

S3F4A1HR

**16/32-BIT RISC
MICROPROCESSOR
USER'S MANUAL**

Revision 1.0



ELECTRONICS

Important Notice

The information in this publication has been carefully checked and is believed to be entirely accurate at the time of publication. Samsung assumes no responsibility, however, for possible errors or omissions, or for any consequences resulting from the use of the information contained herein.

Samsung reserves the right to make changes in its products or product specifications with the intent to improve function or design at any time and without notice and is not required to update this documentation to reflect such changes.

This publication does not convey to a purchaser of semiconductor devices described herein any license under the patent rights of Samsung or others.

Samsung makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Samsung assume any liability arising out of the application or use of any product or circuit and specifically disclaims any and all liability, including without limitation any consequential or incidental damages.

**S3F4A1HR 16/32-Bit RISC Microprocessor
User's Manual, Revision 1.0
Publication Number: 21-S3-F4A1HR-022007**

© 2007 Samsung Electronics

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electric or mechanical, by photocopying, recording, or otherwise, without the prior written consent of Samsung Electronics.

Samsung Electronics' microcontroller business has been awarded full ISO-14001 certification (BSI Certificate No. FM24653). All semiconductor products are designed and manufactured in accordance with the highest quality standards and objectives.

Samsung Electronics Co., Ltd.
San #24 Nongseo-Dong, Giheung-Gu
Yongin-City, Gyunggi-Do, Korea
C.P.O. Box #37, Suwon 446-711

TEL: (82) (31) 209-4956
FAX: (82) (31) 209-3262
Home-Page URL: [Http://www.samsungsemi.com](http://www.samsungsemi.com)

Printed in the Republic of Korea

"Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by the customer's technical experts.

Samsung products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, for other applications intended to support or sustain life, or for any other application in which the failure of the Samsung product could create a situation where personal injury or death may occur.

Should the Buyer purchase or use a Samsung product for any such unintended or unauthorized application, the Buyer shall indemnify and hold Samsung and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, expenses, and reasonable attorney fees arising out of, either directly or indirectly, any claim of personal injury or death that may be associated with such unintended or unauthorized use, even if such claim alleges that Samsung was negligent regarding the design or manufacture of said product.

Table of Contents

Chapter 1 Product Overview

1. Overview	1-1
1.1 Functional Description	1-1
1.2 Features.....	1-2
1.3 Block Diagram	1-4
1.4 Architectural Overview.....	1-5

Chapter 2 Pin Configuration

1. Pin Configuration.....	2-1
2. Pin Assignments.....	2-2
3. Pin Description	2-5

Chapter 3 Memory MAP

Chapter 4 Module Generic Functions

1. Registers access.....	4-1
1.1 Enable / Disable / Status Registers	4-1
1.2 Key Access to Registers.....	4-2
1.3 Registers Undefined Bits	4-2
1.4 Ghost Registers	4-3
2. Power Management Block	4-4

Chapter 5 Analog To Digital Converter

1. Overview	5-1
1.1 Functional Description	5-1
1.2 Block Diagram	5-2
2. External Pin Description.....	5-3
3. Functional Operation.....	5-4
3.1 ADC Detailed Functionalities.....	5-4
3.2 Software Sequence for Conversion.....	5-9
4. Registers Description	5-10

Table of Contents (Continued)

Chapter 6 Capture Module (CAPT)

1. Overview.....	6-1
1.1 Functional Description	6-1
1.2 External Pin Description.....	6-1
2. Functional Operation	6-2
2.1 Mode of Operation	6-2
2.2 Capture's limits.....	6-2
2.3 Programming Examples.....	6-3
3. Registers Description.....	6-4

Chapter 7 Clock Manager

1. Overview.....	7-1
1.1 Functional Description	7-1
1.2 Block Diagram.....	7-2
2. Functional Operation	7-9
2.1 Clock Management.....	7-9
2.2 Clock and Power Supply Monitoring.....	7-57
2.3 Reset Management.....	7-57
3. Registers Description.....	7-59

Chapter 8 Controller Area Network (CANB)

1. Overview.....	8-1
1.1 Functional Description	8-1
1.2 Block Diagram.....	8-2
2. Interface Description.....	8-3
2.1 External Pin Description.....	8-3
3. Functional Operation	8-4
3.1 CAN Protocol	8-4
3.2 Operating Modes.....	8-27
3.3 CAN Application.....	8-34
4. Registers Description.....	8-54

Table of Contents (Continued)

Chapter 9 Data Flash Controller (DFC)

1. Overview	9-1
1.1 Functional Description	9-1
1.2 Block Diagram	9-2
2. Functional Operation	9-3
2.1 Internal Data Flash Controller	9-3
2.2 Read Access	9-4
2.3 Erase Cycle	9-5
2.4 Write Cycle	9-9
2.5 Standby Mode	9-10
2.6 Interrupts Generation	9-10
2.7 Data Abort Generation	9-10
3. Registers Description	9-11

Chapter 10 General Purpose I/O (PIO)

1. Overview	10-1
1.1 Functional Description	10-1
1.2 Block Diagram	10-1
2. Functional Operation	10-2
2.1 PIO Controller Block	10-2
3. Registers Description	10-4

Chapter 11 General Purpose Timer (GPT)

1. Overview	11-1
1.1 Functional Description	11-1
1.2 Block Diagram	11-2
2. External Pin Description	11-3
3. Functional Operation	11-4
3.1 General Description	11-4
3.2 Timer in Capture Mode	11-10
3.3 Timer in Waveform Mode	11-14
3.4 Timer Block Programming	11-20
3.5 Programming Examples	11-22
3.5 Programming Examples	11-22
4. Registers Description	11-23

Table of Contents (Continued)

Chapter 12 Generic Interrupt Controller (GIC)

1. Overview.....	12-1
1.1 Functional Description	12-1
1.2 Block Diagram.....	12-3
1.3 External Pin Description.....	12-3
2. Functional Operation	12-4
2.1 Interrupt Handling	12-4
2.2 Standard Interrupt Sequence.....	12-6
2.3 Fast Interrupt Sequence	12-7
2.4 Spurious Interrupt Sequence	12-8
3. Registers Description.....	12-9

Chapter 13 I/O Configuration (IOCONF)

1. Overview.....	13-1
1.1 Functional Description	13-1
1.2 Block Diagram.....	13-1
2. Functional Operation	13-2
2.1 I/O Configuration	13-2
2.2 Pinout Configuration	13-2
3. Registers Description.....	13-9

Chapter 14 I2C

1. Overview.....	14-1
1.1 Functional Description	14-1
1.2 Block Diagram.....	14-1
2. External Pin Description	14-2
3. Functional Operation	14-3
3.1 I2C Bus Concept.....	14-3
4. Registers Description.....	14-17
4. Timings	14-41

Table of Contents (Continued)

Chapter 15 Interleave Program Flash Memory

1. Overview	15-1
1.1 Functional Description	15-1
1.2 Block Diagram	15-2
2. Functional Operation	15-3
2.1 Interleaved Flash	15-3
2.2 Read Access	15-6
2.3 Erase Cycle	15-7
2.4 Write Cycle	15-10
2.5 Interrupts Generation	15-11
2.6 Data Abort Generation	15-11
3. Registers Description	15-12

Chapter 16 Internal RAM Controller (IRC)

1. Overview	16-1
1.1 Functional Description	16-1
2. Registers Description	16-2

Chapter 17 LCD Controller (LCDC)

1. Overview	17-1
1.1 Functional Description	17-1
2. External Pin Description	17-1
3. Functional Operation	17-2
3.1 LCD Controller	17-2
4. Timings	17-5
4.1 Static Mode	17-5
4.2 1/2 Duty, 1/2 Bias Mode	17-6
4.3 1/3 Duty, 1/2 Bias Mode	17-8
4.4 1/3 Duty, 1/3 Bias Mode	17-10
4.5 1/4 Duty, 1/3 Bias Mode	17-12
5. Registers Description	17-19

Table of Contents (Continued)

Chapter 18 Lite Direct Memory Access (LDMA)

1. Overview.....	18-1
1.1 Functional Description	18-1
1.2 Block Diagram.....	18-1
2. Functional Operation	18-2
2.1 Lite Direct Memory Access	18-2
2.2 LDMA with Hardware Trigger.....	18-3
2.3 LDMA with Software Trigger	18-5
2.4 Priority Management.....	18-6
2.5 LDMA User Interface	18-7
2.6 Power Management.....	18-9
2.7 Debug Mode.....	18-9
3. Registers Description.....	18-10

Chapter 19 Pulse Width Modulation (PWM)

1. Overview.....	19-1
1.1 Functional Description	19-1
1.2 Block Diagram.....	19-1
2. External Pin Description	19-2
3. Functional Operation	19-2
3.1 PWM Parameters.....	19-2
3.2 Programming Examples.....	19-2
4. Registers Description.....	19-4

Chapter 20 Serial Peripheral Interface 16-Bit (SPI16)

1. Overview.....	20-1
1.1 Functional Description	20-1
2. External Pin Description	20-1
2.1 Pin Description	20-2
3. Functional Operation	20-2
3.1 Master mode	20-2
3.2 Slave Mode	20-6
4. Timings	20-7
4.1 Timings Module.....	20-7
4.2 Timings Values.....	20-9
5. Registers Description.....	20-10

Table of Contents (Continued)

Chapter 21 Serial Peripheral Interface 8-Bit (SPI8)

1. Overview	21-1
1.1 Functional Description	21-1
1.2 Block Diagram	21-2
2. External Pin Description	21-2
3. Functional Operation	21-3
3.1 Master Mode	21-3
3.2 Slave Mode	21-7
4. Timings	21-8
4.1 Timings Module	21-8
5. Registers Description	21-11

Chapter 22 Simple Timer (ST)

1. Overview	22-1
1.1 Functional Description	22-1
2. Functional Operation	22-2
2.1 General Description	22-2
2.2 Example	22-4
3. Registers Description	22-5

Chapter 23 Special Function Module (SFM)

1. Overview	23-1
1.1 Functional Description	23-1
2. Registers Description	23-2

Chapter 24 Stamp Timer (STT)

1. Overview	24-1
1.1 Functional Description	24-1
1.2 Block Diagram	24-1
2. Functional Operation	24-2
2.1 Timer Functionality	24-2
2.2 Programming Examples	24-3
3. Registers Description	24-4

Table of Contents (Continued)

Chapter 25 Stepper Motor Controller (SMC)

1. Overview.....	25-1
1.1 Functional Description	25-1
1.2 Block Diagram.....	25-2
2. External Pin Description	25-3
3. Functional Operation	25-3
3.1 Operation Overview	25-3
3.2 PWM Mode	25-3
3.3 Stepper Motor Mode	25-5
3.4 Programming Examples.....	25-13
4. Registers Description.....	25-14

Chapter 26 Universal Sync/Async Receiver/Transmitter (USART)

1. Overview.....	26-1
1.1 Functional Description	26-1
1.2 Block Diagram.....	26-2
2. External Pin Description	26-2
3. Functional Operation	26-3
3.1 Baud Rate Generator.....	26-3
3.2 Receiver	26-6
3.3 Transmitter	26-9
3.4 Break.....	26-10
3.5 Interrupts.....	26-10
3.6 Test Modes	26-10
3.7 LIN Protocol (Compliant with LIN1.2 and LIN2.0 Releases).....	26-11
3.8 Smart-Card Protocol	26-13
3.9 Programming Examples.....	26-14
4. Registers Description.....	25-16

Chapter 27 Watchdog (WD)

1. Overview.....	27-1
1.1 Functional Description	27-1
2. Functional Operation	27-2
2.1 Watchdog Functionality.....	27-2
2.2 Example	27-4
3. Registers Description.....	27-5

Table of Contents (Continued)

Chapter 28 Electrical Data

1. Absolute Maximum Ratings	28-1
2. Recommended Operating Conditions	28-2
3. D.C. Electrical Characteristics.....	28-3

Chapter 29 Package Specifications

List of Figures

Figure Number	Title	Page Number
1-1	S3F4A1HR Block Diagram	1-3
2-1	Pin Configuration.....	2-1
3-1	S3F4A1HR Default Memory Map after Reset.....	3-1
5-1	ADC Block Diagram	5-2
6-1	Capture Pin Resynchronization	6-2
7-1	Clock Manager Block Diagram	7-2
7-2	Clock Manager State Machine.....	7-4
7-3	Transition Diagram: Hardware Reset to NORMAL Mode	7-10
7-4	Transition Diagram: Reset to CKFAIL mode.....	7-11
7-5	Normal Mode to High Speed Mode.....	7-12
7-6	Normal Mode to IDLE-Normal Speed Mode	7-13
7-7	IDLE-Normal Mode to Normal Speed Mode	7-14
7-8	Transition Diagram: NORMAL Mode to SLOW mode	7-15
7-9	Normal Mode to Low Power Mode (LF_SEL = = 0).....	7-16
7-10	Transition Diagram: NORMAL Mode to LOWPOWER Mode (LFSEL=1).....	7-17
7-11	Transition Diagram: NORMAL Mode to HALT Mode (LFSEL=0)	7-18
7-12	Transition Diagram: NORMAL Mode to HALT Mode (LFSEL=1)	7-19
7-13	Transition Diagram: NORMAL Mode to STOP Mode	7-20
7-14	Transition Diagram: LOWPOWER Mode to IDLE-LOWPOWER Mode (LFSEL=0)	7-21
7-15	Transition Diagram: LOWPOWER Mode to IDLE-LOWPOWER Mode (LFSEL=1)	7-22
7-16	Transition Diagram: IDLE-LOWPOWER Mode to LOWPOWER Mode (LFSEL=0)	7-23
7-17	Transition Diagram: IDLE-LOWPOWER Mode to LOWPOWER Mode (LFSEL=1)	7-24
7-18	Transition Diagram: LOWPOWER Mode to NORMAL Mode (LFSEL=0).....	7-25
7-19	Transition Diagram: LOWPOWER Mode to NORMAL Mode (LFSEL=1).....	7-26
7-20	Transition Diagram: LOWPOWER Mode to HIGHSPEED Mode (LFSEL=0).....	7-27
7-21	Transition Diagram: LOWPOWER Mode to HIGHSPEED Mode (LFSEL=1).....	7-28
7-22	Transition Diagram: LOWPOWER Mode to SLOW Mode	7-29
7-23	Transition Diagram: LOWPOWER Mode to HALT Mode (LFSEL==0).....	7-30
7-24	Transition Diagram: LOWPOWER Mode to HALT Mode (LFSEL==1).....	7-31
7-25	Transition Diagram: SLOW Mode to IDLE-SLOW Mode	7-32
7-26	Transition Diagram: IDLE-SLOW Mode to SLOW Mode	7-33
7-27	Transition Diagram: SLOW Mode to NORMAL Mode	7-34
7-28	Transition Diagram: SLOW Mode to HIGHSPEED Mode.....	7-35
7-29	Transition Diagram: SLOW Mode to LOWPOWER Mode	7-36
7-30	Transition Diagram: SLOW Mode to HALT Mode.....	7-37
7-31	Transition Diagram: HIGHSPEED Mode to IDLE-HIGHSPEED Mode.....	7-38
7-32	Transition Diagram: HIGHSPEED Mode to IDLE-HIGHSPEED Mode.....	7-39
7-33	Transition Diagram: HIGHSPEED Mode to NORMAL Mode.....	7-40
7-34	Transition Diagram: HIGHSPEED Mode to SLOW Mode.....	7-41
7-35	Transition Diagram: HIGHSPEED Mode to LOWPOWER Mode (LFSEL=0).....	7-42

List of Figures (Continued)

Figure Number	Title	Page Number
7-36	Transition Diagram: HIGHSPEED Mode to LOWPOWER Mode (LFSEL=1)	7-43
7-37	Transition Diagram: HIGHSPEED Mode to HALT Mode (LFSEL=0).....	7-44
7-38	Transition Diagram: HIGHSPEED Mode to HALT Mode (LFSEL=1).....	7-45
7-39	Transition Diagram: HALT Mode to NORMAL Mode (LFSEL=0).....	7-46
7-40	Transition Diagram: HALT Mode to NORMAL Mode (LFSEL=1).....	7-47
7-41	Transition Diagram: HALT Mode to HIGHSPEED Mode (LFSEL=0).....	7-48
7-42	Transition Diagram: HALT Mode to HIGHSPEED Mode (LFSEL=1).....	7-49
7-43	Transition Diagram: HALT Mode to SLOW Mode	7-50
7-44	Transition Diagram: HALT Mode to LOWPOWER Mode (LFSEL=0)	7-51
7-45	Transition Diagram: HALT Mode to LOWPOWER Mode (LFSEL=1)	7-52
7-46	Transition Diagram: STOP Mode to NORMAL Mode (LFSEL=0)	7-53
7-47	Transition Diagram: STOP Mode to NORMAL Mode (LFSEL=1)	7-54
7-48	Low Frequency Transition: DIVOUT to RINGCLK	7-55
7-49	Low Frequency Transition: RINGCLK to DIVOUT	7-56
7-50	Reset Manager Diagram	7-58
7-51	ARM Instruction Follows a Peripheral Write Access Instruction	7-68
7-52	Peripheral access follows a peripheral write access instruction	7-69
8-1	CAN Block Diagram.....	8-2
8-2	CAN Layers Description	8-6
8-3	Data Frame.....	8-10
8-4	Standard Format.....	8-11
8-5	Extended Format.....	8-11
8-6	CAN Control Field.....	8-12
8-7	CRC Field.....	8-13
8-8	ACK Field	8-14
8-9	Remote Frame.....	8-15
8-10	Error Frame	8-16
8-11	Overload Frame.....	8-17
8-12	Interframe Space for Receiver	8-18
8-13	Interframe Space for Transmitter	8-18
8-14	Partition of Bit Time	8-23
8-15	Example of Nominal Bit Time	8-25
8-16	CAN Core in Silent Mode	8-30
8-17	CAN Core in Loop Back Mode	8-31
8-18	CAN Core in Loop Back Combined with Silent Mode	8-32
8-19	Data Transfer Between IFx Registers and Message RAM	8-35
8-20	CPU Handling of a FIFO Buffer.....	8-41
8-21	Bit Timing.....	8-43
8-22	Propagation Time Segment.....	8-44
8-23	Synchronization on Late and Early Edges	8-46
8-24	Filtering of Short Dominant Spikes.....	8-48
8-25	Structure of the CAN Controller Core.....	8-50
8-26	CAN Bit Time Built up from Time Quantum	8-62
8-27	Open Drain Mode	8-95

List of Figures (Continued)

Figure Number	Title	Page Number
9-1	Data Flash Memory Controller Block Diagram	9-2
9-2	Data Flash Segmentation	9-3
9-3	Read Access Timing Diagram	9-4
9-4	Chip Erase Flow Chart	9-6
9-5	Sector Erase Flow Chart	9-7
9-6	Page Erase Flow Chart	9-8
9-7	Write Flow Chart	9-9
10-1	PIO Block Diagram.....	10-1
11-1	General Purpose Timer (3 channels) Block Diagram	11-2
11-2	Clock Selection Block Diagram	11-5
11-3	Burst Clock Timing Diagram	11-6
11-4	Counter Reset Diagram	11-7
11-5	TIOA Pulse.....	11-11
11-6	TIOA Edges.....	11-12
11-7	TIOA Pulse or Period	11-13
11-8	Counter on TCLK	11-13
11-9	Dual Pulse Width Modulation.....	11-15
11-10	2 Square Signals.....	11-16
11-11	Pulse Generation	11-17
11-12	Single Waveform with Trigger on TIOB	11-18
11-13	Event Counter	11-19
11-14	General Purpose Timer Block Programming Diagram	11-20
11-15	Application with General Purpose Timer Block Programming.....	11-21
12-1	Generic Interrupt Controller Block Diagram.....	12-3
13-1	I/O Configuration Module Block Diagram.....	13-1
14-1	I2C Block Diagram	14-1
14-2	Data Validity	14-5
14-3	Start and Stop Conditions	14-6
14-4	Data Transfer on the I2C Bus	14-7
14-5	Acknowledge.....	14-8
14-6	A master-Transmitter Addresses a Slave	14-10
14-7	A master-Transmitter Addresses a Slave-Receiver with a 10–bits Address	14-16
14-8	Hold and Setup Delays	14-40

List of Figures (Continued)

Figure Number	Title	Page Number
15-1	Interleaved Program Flash Memory Controller Block Diagram.....	15-2
15-2	Interleaved Memory.....	15-4
15-3	Flash Data Out Timing	15-5
15-4	Chip Erase Flow Chart	15-8
15-5	Sector Erase Flow Chart	15-9
15-6	Write Flow Chart.....	15-10
17-1	LCD Controller Block Diagram	17-2
17-2	LCD Bias Driver.....	17-4
17-3	COM/SEG Signals Under Static Mode.....	17-5
17-4	COM/SEG Signals Under 1/2 Duty 1/2 Bias Mode	17-7
17-5	COM/SEG Signals Under 1/3 Duty 1/2 Bias Mode	17-9
17-6	COM/SEG Signals Under 1/3 Duty 1/3 Bias Mode	17-11
17-7	COM/SEG Signals Under 1/4 Duty 1/3 Bias Mode	17-13
17-8	LCD Connection Example in Static Mode	17-14
17-9	LCD Connection Example at 1/2 Duty, 1/2 Bias	17-15
17-10	LCD Connection Example at 1/3 Duty, 1/2 Bias	17-16
17-11	LCD Connection Example at 1/3 Duty, 1/3 Bias	17-17
17-12	LCD Connection Example at 1/4 Duty, 1/3 Bias	17-18
18-1	LDMA Block Diagram	18-1
18-2	LDMA with Peripherals	18-2
18-3	Example : Transfers From a Memory to a Peripheral	18-4
18-4	Example : Transfers From a Peripheral to a Memory	18-5
19-1	PWM Block Diagram	19-1
19-2	Example.....	19-4
20-1	SPI16 Block Diagram in Master Mode	20-4
20-2	SPI16 Flow Diagram in Master Mode.....	20-5
20-3	SIP16 in Slave Mode.....	20-6
20-4	SPI16 Block Diagram in Master Mode, phase=0	20-7
20-5	SPI16 in Master Mode, phase=1	20-7
20-6	DLYBCS, DLYBS and DLYBCT	20-8
20-7	SPI16 Timings	20-8

List of Figures (Continued)

Figure Number	Title	Page Number
21-1	SPI8 Block Diagram	21-2
21-2	SPI8 Flow Chart in Master Mode	21-5
21-3	SPI8 in Master Mode.....	21-6
21-4	SPI8 in Slave Mode.....	21-7
21-5	SPI8 in Master Mode, Phase=0	21-8
21-6	SPI8 in Master Mode, Phase=1	21-8
21-7	SPI8 Timings.....	21-9
22-1	Simple Timer Block Diagram	22-3
24-1	Simple Timer Block Diagram	24-1
25-1	SMC Block Diagram.....	25-2
25-2	Stepper Motor Connection	25-5
25-3	One Phase on Full Stepping	25-6
25-4	Two Phase on Full Stepping	25-7
25-5	Half Stepping.....	25-8
25-6	Cosinus/Sinus Microstepping.....	25-9
25-7	High Torque Microstepping.....	25-10
25-8	PWM Example	25-13
26-1	USART Block Diagram.....	26-2
26-2	USART Baud Rate Generator Block Diagram	26-4
26-3	Asynchronous Mode, Start Bit Detection	26-6
26-4	Asynchronous Mode, Character Reception	26-6
26-5	Synchronous Mode, Character Reception	26-7
26-6	IDLE Flag	26-8
26-7	Synchronous and Asynchronous Modes, Character Transmission.....	26-9
26-8	Message Characteristics.....	26-12
26-9	Smart-Card Transmission Error	26-13
26-10	Error Signaling on Reception	26-14
27-1	Watchdog Block Diagram	27-3
29-1	100 Pin Package Dimensions	29-1

List of Tables

Table Number	Title	Page Number
2-1	Pin Assignments–Pin Number Order	2-2
2-2	Pin Description	2-7
3-1	The Base Address of Peripheral Special Registers	3-1
5-1	ADC Pin Description	5-3
5-2	NBRCH[3:0] Values and the Number of Conversions	5-4
5-3	CVx Values and Selected Input	5-5
5-4	Status Bits Reflecting Power Management.....	5-8
5-5	ADC Memory Map.....	5-10
5-6	NBRCH[3:0] Values and the Number of Conversions	5-16
5-7	Ready for Conversion	5-18
6-1	CAPT Pin Description	6-1
6-2	Capture Special Function Registers	6-4
7-1	Clock Sources	7-3
7-2	Derived Clock Sources.....	7-3
7-3	Oscillators and PLL Activity	7-5
7-4	Clock Transition	7-6
7-5	Reset Sources.....	7-57
7-6	Clock Manager Memory Map.....	7-59
7-7	Multiplier Parameter	7-63
7-8	Post Scalar Parameter.....	7-64
7-9	Pre Divider Parameter.....	7-64
7-10	PLLPRE/PMUL/PLLPOST Allowed Values	7-65
7-11	Peripheral Divider.....	7-67
7-12	Clock Manager Modes	7-70
7-13	Core-Clock Divider	7-72
7-14	Low Frequency Oscillator Clock Divider	7-73
8-1	CAN Pin Description	8-3
8-2	Identifier Length Within Standard and Extended Frame.....	8-9
8-3	RTR Bit.....	8-12
8-4	IDE Bit	8-12
8-5	Data Length Code	8-13
8-6	Interface Register Sets.....	8-28
8-7	Message Object Content	8-29
8-8	Initialization of a Transmit Object.....	8-37
8-9	Initialization of a Receive Object.....	8-38
8-10	Parameters of the CAN Bit Time.....	8-43
8-11	CAN Special Function Register	8-54
8-12	Bit rate and Minimal Core Frequency	8-62
8-13	Control of CAN_TX Pin	8-95

List of Tables (Continued)

Table Number	Title	Page Number
9-1	DFC Special Function Registers	9-11
10-1	PIO Special Function Registers	10-4
11-1	Pin Descriptions Corresponding to Operation Modes	11-3
11-2	Available Interrupts.....	11-9
11-3	GPT Special Function Registers	11-23
11-4	Multi Channels Control Registers Memory Map.....	11-23
11-5	Clock Select.....	11-28
11-6	Burst	11-29
11-7	External Trigger Edge	11-30
11-8	Load Rx	11-30
11-9	Clock Select.....	11-31
11-10	Burst	11-32
11-11	External Event Edge.....	11-33
11-12	External Event	11-33
11-13	Effect on TIOA Event.....	11-34
11-14	Effect on TIOB Event.....	11-35
12-1	Interrupt Sources	12-1
12-2	Generic Interrupt Controller Pin Description	12-3
12-3	Priority Levels	12-4
12-4	GIC Special Function Registers	12-9
12-5	Interrupt Source Type Field.....	12-10
13-1	I/O Configuration	13-2
13-2	IOCONF Special Function Registers.....	13-9

List of Tables (Continued)

Table Number	Title	Page Number
14-1	I2C Pin Description	14-2
14-2	Examples of Baud Rate Configuration	14-4
14-3	Definition of Bytes in First Byte	14-11
14-4	I2C Special Function Registers.....	14-17
14-5	Values of FSCL in kHz Depending of PRV, FAST and PCLK	14-24
14-6	Master/Transmitter Mode Status Codes	14-26
14-7	Master/Receiver Mode Status Codes	14-27
14-8	Slave/Receiver Mode Status Codes	14-29
14-9	Slave/Transmitter Mode Status Codes	14-32
14-10	Miscellaneous Status Codes.....	14-34
14-11	Timing Requirements	14-41
15-1	IFC Special Function Registers	15-12
15-2	Sector Number	15-14
16-1	IRC Special Function Registers	16-2

List of Tables (Continued)

Table Number	Title	Page Number
17-1	LCD Pin Description	17-1
17-2	LCDC Special Function Registers.....	17-19
18-1	LDMA Special Function Registers.....	18-10
18-2	LDMA Transfers Size	18-21
19-1	PWM Pin Description	19-2
19-2	PWM Special Function Registers.....	19-5
20-1	SPI16 Pin Description	20-1
20-2	SPI16 Timing Values	20-9
20-3	SPI16 Special Function Registers.....	20-10
20-4	SPI Bits Per Transfer.....	20-26
20-5	Baudrate	20-27
21-1	SPI8 Pin Description	21-2
21-2	SPI8 Timing Values	21-10
21-3	SPI8 Special Function Registers.....	21-11
21-4	SPI Bits Per Transfer.....	21-26
21-5	Baudrate	21-26
21-6	Delay Before SPCK.....	21-27
21-7	Delay between Consecutive Transfers	21-27
22-1	Simple Timer Special Function Registers	22-5
23-1	SFM Special Function Registers	23-2
23-2	Type of Chip	23-3
23-3	Chip Architecture	23-4
23-4	Memory type.....	23-5
23-5	Memory type.....	23-5
24-1	Stamp Timer Special Function Registers.....	24-4

List of Tables (Continued)

Table Number	Title	Page Number
25-1	PWM Pin Description	25-3
25-2	PWM Limits Frequency	25-4
25-3	SMC Special Function Registers	25-14
25-4	Relation Between NCM[2:0] and the Number of PWM Cycle by microstep	25-21
25-5	Relation Between NMSQ[1:0] and the Number microstep by Sinusoid Quarter	25-21
25-6	Relation Between DM[2:0] and the Driving Method.....	25-22
26-1	USART Pin Description.....	26-2
26-2	Asynchronous Mode (SYNC = 0).....	26-4
26-3	Synchronous Mode (SYNC = 1).....	26-5
26-4	USART Special Function Registers	26-16
26-5	SENDTIME Configuration Field	26-22
26-6	CLKS Clock Selection Field	26-23
26-7	Character Length Field.....	26-23
26-8	Parity Type Field	26-23
26-9	NBSTOP Configuration Field	26-24
26-10	Channel Mode Field	26-24
26-11	Clock Divisor Field	26-38
26-12	Time-Out Configuration Field.....	26-39
26-13	Time-Guard Configuration Field.....	26-40
26-14	Number of Data Field for the LIN1.2 Release.....	26-41
27-1	Watchdog Special Function Registers	27-5
27-2	Watchdog Clock Divider.....	27-10
28-1	Absolute Maximum Ratings	28-1
28-2	Recommended Operating Conditions.....	28-2
28-3	D.C. Electrical Characteristics (5V I/O).....	28-3
28-4	D.C. Electrical Characteristics (3.3V I/O).....	28-5
28-5	Current Consumption	28-7
28-6	Oscillation Characteristics.....	28-7
28-7	Oscillation Stabilization Time	28-8
28-8	Internal RC Oscillation Characteristics	28-8
28-9	LVD Characteristics	28-8
28-10	Input/Output Capacitance	28-9
28-11	A.C. Electrical Characteristics.....	28-9
28-12	A.C. Electrical Characteristics for Internal Flash ROM	28-10
28-13	ADC Electrical Characteristics	28-10
28-14	D.C. Electrical Characteristics for LCD Controller	28-11

List of Important Notice

Notice Number	Title	Page Number
2-1	Regarding SPGM_nBOOT pin connection in flash writing mode	2-6
5-1	Regarding STOP command in ADC continuous mode	5-6
7-1	Regarding writing access to CM_PSTR register	7-62
7-2	Regarding writing access to CM_PDPR register	7-63
7-3	Regarding PLLPRE, PMUL and PLLPOST values	7-65
7-4	Regarding the max frequency of APB(PCLK)	7-65
7-5	Regarding CM_DIVBR register in case of external clock.....	7-67
7-6	Regarding the clock frequency between PCLK and SCLK	7-68
7-7	Regarding clock management in case of Slow, Low power, Halt mode	7-70
7-8	Regarding a mode transition and STABLE interrupt.....	7-70
7-9	Regarding a divided master clock frquency	7-72
7-10	Regarding changing MDIV value of CM_MDIVR register	7-72
7-11	Regarding changing LDIV value of CM_MDIVR register	7-73
7-12	Regarding a configuration and changing the low frequency	7-74
7-13	Regarding LFUSED bit of CM_STR register.....	7-74
7-14	Regarding STOP mode and STOPMODE bit	7-75
8-1	Regarding the message object number 32	8-28
8-2	Regarding the message object number 32	8-81
9-1	Regarding the control standby mode by STANDEN bit	9-17
12-1	Regarding the clear of pending interrupts.....	12-16
15-1	Regarding SPEEDMODE bit and switching Mode	15-16
17-1	Regarding LCD Pin configuration.....	17-1
18-1	Regarding transfer huge number of data	18-5
18-2	Regarding the channel priority during transfer	18-6
18-3	Regarding the transfer size of LDMA	18-7
18-4	Regarding LDMA register modification with debug state	18-9
18-5	Regarding writing access LDMA_MR register	18-21
18-6	Regarding CHEN bit of LDMA status register	18-23
18-7	Regarding writing access LDMA_ASRCR register	18-27
18-8	Regarding writing access LDMA_ADSTR register.....	18-28
18-9	Regarding writing access LDMA_CNTRX register.....	18-29

List of Important Notice (Continued)

Notice Number	Title	Page Number
22-1	Regarding the condition between LFCLK and PCLK for ST0.....	22-9
22-2	Regarding simple timer en/disable with software reset	22-9
22-3	Regarding restart the simple timer.....	22-11
22-4	Regarding the condition for writing access ST_PR0 register	22-15
22-5	Regarding the value of ST_CT0 register in auto-reload mode	22-16
22-6	Regarding the condition for writing access ST_PR1 register	22-17
22-7	Regarding the value of ST_CT1 register in auto-reload mode	22-18
24-1	Regarding internal oscillator clock for STT	24-3
24-2	Regarding the condition of LFCLK clock for STT	24-8
24-3	Regarding writing access CNTRST bit in STT_MR register	24-9
24-4	Regarding writing access STT_CNT register and invalid data	24-16
24-5	Regarding writing access STT_ALR register and invalid data.....	24-17
25-1	Regarding the condition to write the value into SMC_DLY0/1 register.....	25-28
25-2	Regarding the condition to write the value into SMC_PUL0/1 register.....	25-29
25-3	Regarding the condition to write the value into SMC_TPR register	25-30
26-1	Regarding non-effected US_PMSR register by software reset	26-19
26-2	Regarding setting LIN2_0 bit during message transfer	26-22
26-3	Regarding RXRDY bit clear by reading US_RHR register	26-36
26-4	Regarding setting US_BRGR value under the synchronous mode.....	26-38
26-5	Regarding timeout operation in receiver disable and re-enable	26-39
26-6	Regarding writing access US_LIR register during a message transfer	26-41
26-7	Regarding writing access US_DFWR0 during a message transfer.....	26-42
26-8	Regarding writing access US_DFWR1 during a message transfer.....	26-43
26-9	Regarding writing access US_SBLR during a message transfer	26-46

1

PRODUCT OVERVIEW

1. OVERVIEW

1.1 FUNCTIONAL DESCRIPTION

SAMSUNG S3F4A1HR 16/32-bit RISC micro-controller is a cost-effective and high-performance solution for automotive segment, especially for dashboard and body application.

An outstanding feature of the S3F4A1HR is its CPU core, a 16/32-bit RISC processor (ARM7TDMI-S) designed by advanced RISC machines, Ltd. The ARM7TDMI-S core is a low-power, general-purpose, microprocessor macro-cell which was developed for the use in application-specific and customer-specific integrated circuits. Its simple, elegant, and fully static design is particularly suitable for cost-sensitive and power-sensitive applications.

Other feature is an integration of 512Kbytes and 32Kbytes NOR type flash, 16Kbytes SRAM, CAN, LIN, LCD, and Stepper Motor Controller. By providing a complete set of common system peripherals, the S3F4A1HR can minimize the overall system costs and eliminate the need to configure additional components, externally.

The S3F4A1HR is to be developed by using the ARM7TDMI-S core, CMOS standard cell, and data path compiler. Most of the on-chip function blocks will be designed using an HDL synthesizer. The S3F4A1HR will be fully verified in SAMSUNG ASIC test environment including the internal qualification assurance process.

1.2 FEATURES

Architecture

- ARM7TDMI-S CPU Core
- 16/32-bit RISC architecture
- JTAG-based debugging solution

Program Flash Memory

- 512Kbytes internal NOR Flash ROM
- Support high and normal speed mode

Data Flash Memory

- 32Kbytes internal NOR Flash ROM

RAM Memory

- 16Kbytes internal SRAM
- Support the changeable base address

Clock Generator with Configurable PLL

- Programmable clock synthesizer (Max. 40MHz)

Lite Direct Memory Access (LDMA)

- 7 channels
- Transfer from peripheral to memory
- Transfer from memory to memory

Interrupt Controller (GIC)

- 64 interrupt lines
- 44 internal and 12 external interrupt lines
- 16-Level priority vectored interrupt

General Purpose I/O

- 74 multiplexed GPIO

16bit General Purpose Timer (GPT)

- 3 channels : GPT0, GPT1, GPT2
- 3 configurable modes: Counter, PWM, Capture
- Support capture/compares

16bit Simple Timer (ST)

- 2 channels simple timer : ST1
- 2 channels simple timer : ST0
- ST1 on Core Clock
- ST0 on Low Frequency Clock

CAN Controller

- 2 channels with 32buffers : CAN0, CAN1
- Support CAN 2.0A and 2.0B Full Speed
- Stampable message

CAPTURE

- 1 x 16bit capture module : CAPT0
- 1 LDMA channel

Pulse Width Modulations (PWM)

- 2 channels 16bit PWM : PWM0, PWM1
- 8 channels 8bit PWM : PWM2~PWM5

Stepper Motor Controllers (SMC)

- 4 channels : SM[00:03] ~ SM[30:33]
- Normal/Wave/Half-step/Micro-stepping Mode
- Micro-stepping cosinus/sinus & high torque

Stamp Timer (STT)

- 2 channels 32-bit Timer
- Alarm interrupt

A/D Converters (ADC)

- 16 channel analog inputs : AIN[15:0]
- 10bit resolution, Max 500KSps conversion rate
- 1 LDMA channel

UART- LIN

- 3 channels : UART0, UART1, USART0
- Support hardware LIN 1.2 & 2.0
- Support 5,6,7,8, and 9bit data length
- Support for J1587 protocol
- 1 channel, USART0, support synchronous transfer
- 3 LDMA channels

IIC Bus Interface (I2C)

- 2 channels, Multi-master IIC-Bus
- Serial, 8-bit oriented and bi-directional data transfers can be made at up to 100Kbit/s in stand and mode or up to 400Kbit/s in fast mode

1.2 FEATURES (Continued)

LCD Controller (LCDC)

- 4com x 30 segment
- Static, 1/2 and 1/3 bias mode

SPI Bus Interface

- 1 to 8-bit programmable data length, SPI0
- 8 to 16-bit programmable data length, SPI1
- Support Master and Slave mode
- 4 LDMA channels

Watchdog (WD)

- Programmable watchdog timer

Clock Manager (CM)

- Internal Ring oscillator (Typ.1MHz)
- CPU & Peripherals can be deactivated individually
- Clock monitor

Low Voltage Detector (LVD)

- Internal reset generation: Typ. 2.4V
- Interrupt generation: Typ. 4.2V

Power on Reset (POR)

Operating Frequency Range

- 4 ~ 6MHz by External Crystal
- 12 ~ 40MHz by PLL

Operating Voltage Range

- 3.0 ~ 5.5V except ADC, SMC
- ADC, Stepper Motor: 4.5 ~ 5.5V

Operating Temperature Range

- -40 ~ 105°C(not including stepper motor driver)
- -40 ~ 85°C(including stepper motor driver)

Available in 100 TQFP Package

1.3 BLOCK DIAGRAM

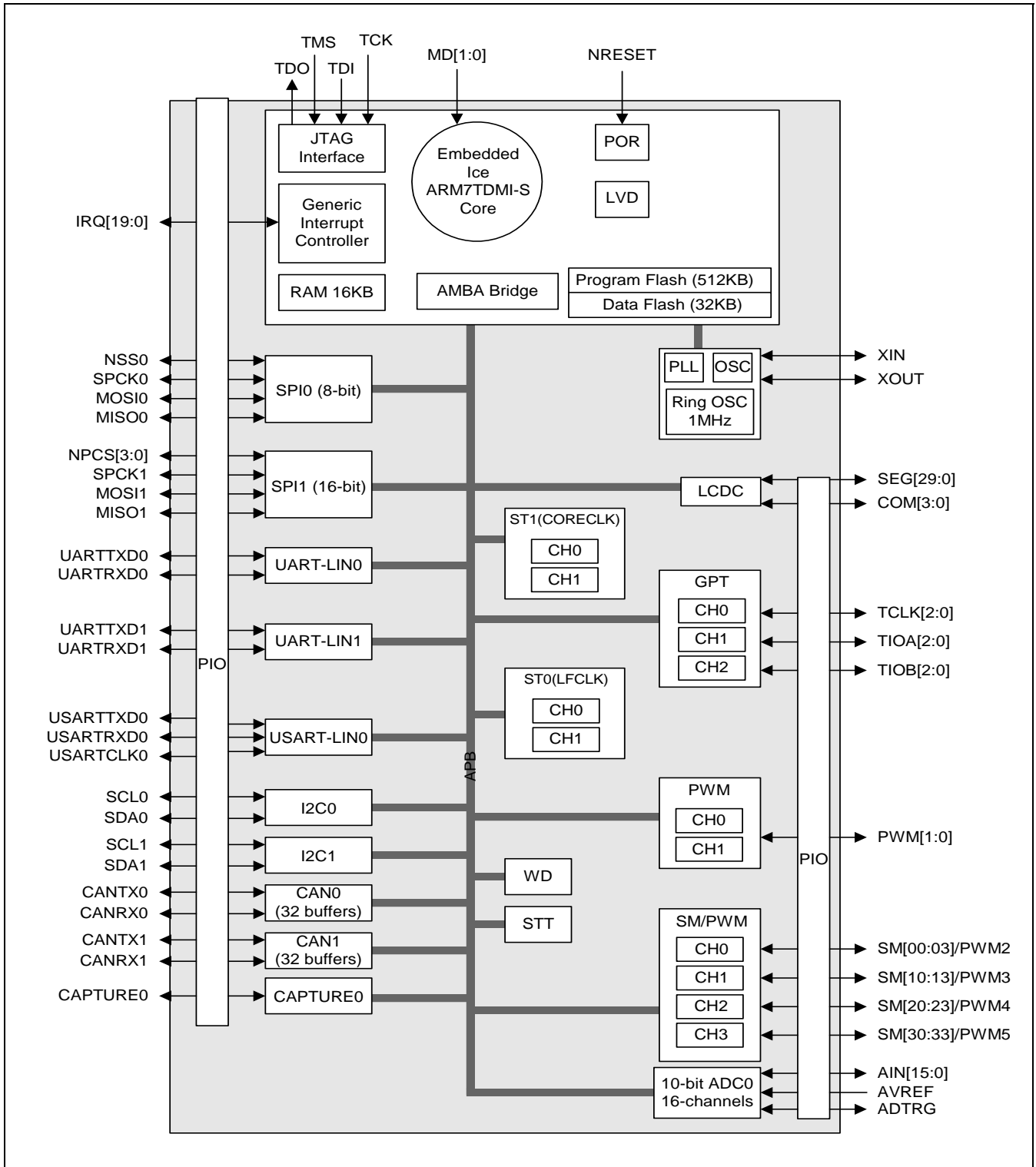


Figure 1-1. S3F4A1HR Block Diagram

1.4 ARCHITECTURAL OVERVIEW

The S3F4A1HR architecture consists of 2 main buses, the Advanced System Bus (ASB) and the Advanced Peripheral Bus (APB). The ASB is designed for maximum performance. It interfaces the processor to the on-chip 32-bit memories. The APB is designed for accesses to on-chip peripherals and is optimized for low power consumption. The AMBA bridge provides an interface between the ASB and the APB.

The S3F4A1HR peripherals are designed to be programmed with a minimum number of instructions. Each peripheral has a space of 16K bytes address allocated in the upper 3M bytes of the 4G bytes address space. Except for the interrupt controller, the peripheral base address is the lowest address of its memory space. The peripheral register set is composed of control, mode, data, status and interrupt registers.

The S3F4A1HR microcontroller operates in little-endian mode. The processor's internal architecture and the ARM and THUMB instruction sets are described in the ARM7TDMI-S data sheet. The memory map and the on-chip peripherals are described in the sub-sequent sections of this data sheet.

The ARM Standard In-Circuit-Emulation debug interface is supported via the ICE port of the S3F4A1HR microcontroller (this is a standard IEEE 1149.1 JTAG Boundary Scan interface).

2 PIN CONFIGURATION

1. PIN CONFIGURATION

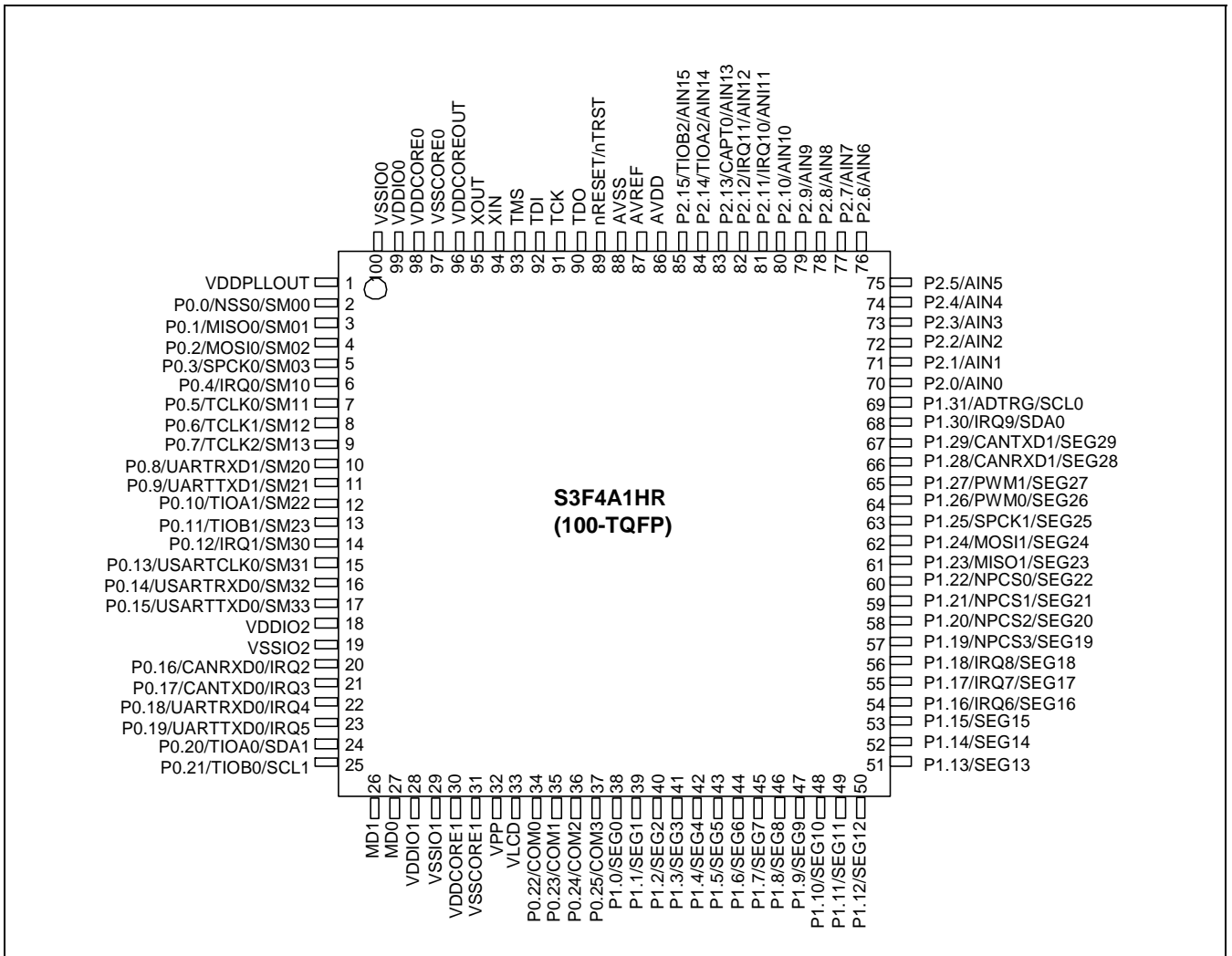


Figure 2-1. Pin Configuration

2. PIN ASSIGNMENTS

Table 2-1. Pin Assignments–Pin Number Order

Num	Name 1	Name 2	Name 3	Flash
1	VDDPLLOUT	–	–	–
2	P0.0	NSS0	SM00	–
3	P0.1	MISO0	SM01	–
4	P0.2	MOSI0	SM02	–
5	P0.3	SPCK0	SM03	–
6	P0.4	IRQ0	SM10	–
7	P0.5	TCLK0	SM11	–
8	P0.6	TCLK1	SM12	–
9	P0.7	TCLK2	SM13	–
10	P0.8	UARTRXD1	SM20	–
11	P0.9	UARTTXD1	SM21	–
12	P0.10	TIOA1	SM22	–
13	P0.11	TIOB1	SM23	–
14	P0.12	IRQ1	SM30	–
15	P0.13	USARTCLK0	SM31	–
16	P0.14	USARTRXD0	SM32	–
17	P0.15	UARTTXD0	SM33	–
18	VDDIO2	–	–	–
19	VSSIO2	–	–	–
20	P0.16	CANRXD0	IRQ2	–
21	P0.17	CANTXD0	IRQ3	–
22	P0.18	UARTRXD0	IRQ4	–
23	P0.19	UARTTXD0	IRQ5	–
24	P0.20	TIOA0	SDA1	–
25	P0.21	TIOB0	SCL1	–
26	MD1	–	–	–
27	MD0	–	–	–
28	VDDIO1	–	–	–
29	VSSIO1	–	–	–
30	VDDCORE1	–	–	–
31	VSSCORE1	–	–	–
32	VPP	–	–	–
33	VLCD	–	–	–

Table 2-1. Pin Assignments–Pin Number Order (Continued)

Num	Name 1	Name 2	Name 3	Flash
34	P0.22	–	COM0	–
35	P0.23	–	COM1	–
36	P0.24	–	COM2	–
37	P0.25	–	COM3	–
38	P1.0	–	SEG0	FS_CLK(I)
39	P1.1	–	SEG1	FS_DIO(I/O)
40	P1.2	–	SEG2	SPGM_nBOOT
41	P1.3	–	SEG3	–
42	P1.4	–	SEG4	–
43	P1.5	–	SEG5	–
44	P1.6	–	SEG6	–
45	P1.7	–	SEG7	–
46	P1.8	–	SEG8	–
47	P1.9	–	SEG9	–
48	P1.10	–	SEG10	–
49	P1.11	–	SEG11	–
50	P1.12	–	SEG12	–
51	P1.13	–	SEG13	–
52	P1.14	–	SEG14	–
53	P1.15	–	SEG15	–
54	P1.16	IRQ6	SEG16	–
55	P1.17	IRQ7	SEG17	–
56	P1.18	IRQ8	SEG18	–
57	P1.19	NPCS3	SEG19	–
58	P1.20	NPCS2	SEG20	–
59	P1.21	NPCS1	SEG21	–
60	P1.22	NPCS0	SEG22	–
61	P1.23	MISO1	SEG23	–
62	P1.24	MOSI1	SEG24	–
63	P1.25	SPCK1	SEG25	–
64	P1.26	PWM0	SEG26	–
65	P1.27	PWM1	SEG27	–
66	P1.28	CANRXD1	SEG28	–
67	P1.29	CANTXD1	SEG29	–

Table 2-1. Pin Assignments–Pin Number Order (Continued)

Num	Name 1	Name 2	Name 3	Flash
68	P1.30	IRQ9	SDA0	–
69	P1.31	ADTRG	SCL0	–
70	P2.0	–	AIN0	–
71	P2.1	–	AIN1	–
72	P2.2	–	AIN2	–
73	P2.3	–	AIN3	–
74	P2.4	–	AIN4	–
75	P2.5	–	AIN5	–
76	P2.6	–	AIN6	–
77	P2.7	–	AIN7	–
78	P2.8	–	AIN8	–
79	P2.9	–	AIN9	–
80	P2.10	–	AIN10	–
81	P2.11	IRQ10	AIN11	–
82	P2.12	IRQ11	AIN12	–
83	P2.13	CAPT0	AIN13	–
84	P2.14	TIOA2	AIN14	–
85	P2.15	TIOB2	AIN15	–
86	AVDD	–	–	–
87	AVREF	–	–	–
88	AVSS	–	–	–
89	nRESET	nTRST	–	–
90	TDO	–	–	–
91	TCK	–	–	–
92	TDI	–	–	–
93	TMS	–	–	–
94	XIN	–	–	–
95	XOUT	–	–	–
96	VDDCOREOUT	–	–	–
97	VSSCORE0	–	–	–
98	VDDCORE0	–	–	–
99	VDDIO0	–	–	–
100	VSSIO0	–	–	–

2-1. IMPORTANT NOTICE

SPGM_nBOOT must be connected to **GND** in the flash writing by SPGM mode.

3. PIN DESCRIPTION

Table 2-2. Pin Description

Module	Pin Name	Function	I/O Type	Active Level	Comments
GIC	IRQ[11:0]	External interrupt request	I	Low	Multiplexed with I/O
RESET	NRST	Hardware reset	I	Low	Schmitt trigger, internal filter
LCD	COM[3:0]	LCD common signal	O	–	Multiplexed with peripheral module
	SEG[29:0]	LCD segment	O	–	Multiplexed with peripheral module
	VLCD	LCD Bias	Power	–	–
CLOCK	XIN	Crystal or oscillator input	I	–	–
	XOUT	Oscillator output	O	–	–
PIO	P0.X P1.X P2.X	General purpose I/O multiplexed	I/O	–	Multiplexed with peripheral module
PWM	PWM[1:0]	Pulse width modulation output	O	–	Multiplexed with I/O
SMC	SM[3:0][3:0]	Stepper motor output	O	–	SM[3:0][1:0] can be used as PWM
GPT	TCLK[2:0]	Timer external clock	O	–	Multiplexed with I/O
	TIOA[2:0]	Multipurpose timer I/O pin A	I/O	–	Multiplexed with I/O
	TIOB[2:0]	Multipurpose timer I/O pin B	I/O	–	Multiplexed with I/O
ADC	AIN[15:0]	ADC Input channels [15:0]	Analog I	–	Multiplexed with I/O
	ADTRG	ADC external trigger	I	–	Multiplexed with I/O
I2C	SDA[1:0]	Serial data	I/O	–	Multiplexed with I/O
	SCL[1:0]	Serial clock	I/O	–	Multiplexed with I/O

Table 2-2. Pin Description (Continued)

Module	Pin Name	Function	I/O Type	Active Level	Comments
U(S)ART	UARTRXD0/1, USARTRXD0	Received signal	I	–	Multiplexed with I/O
	UARTTXD0/1, USARTTXD0	Transmit signal	O	–	Multiplexed with I/O
	USARTCLK0	Clock signal	I/O	–	Multiplexed with I/O
CAPT	CAPT0	Capture input	I	–	Multiplexed with I/O
SPI0	SPCK0	SPI clock	I/O	–	Multiplexed with I/O
	MISO0	Master in slave out	I/O	–	Multiplexed with I/O
	MOSI0	Master out slave in	I/O	–	Multiplexed with I/O
	NSS0	Peripheral chip select	O	Low	Multiplexed with I/O
SPI1	SPCK1	SPI clock	I/O	–	Multiplexed with I/O
	MISO1	Master in slave out	I/O	–	Multiplexed with I/O
	MOSI1	Master out slave in	I/O	–	Multiplexed with I/O
	NPCS[3:0]	Peripheral chip select	O	Low	Multiplexed with I/O
CAN	CANTX0	Transmit line	O	Low	Multiplexed with I/O
	CANRX0	Receive line	I	Low	Multiplexed with I/O
	CANTX1	Transmit line	O	Low	Multiplexed with I/O
	CANRX1	Receive line	I	Low	Multiplexed with I/O
JTAG	TMS	Test mode select	I	–	internal pull-up
	TDI	Test data in	I	–	internal pull-up
	TDO	Test data out	O	–	–
	TCK	Test clock	I	–	(note 1)
MODE	MD[1:0]	Test reserved pin	I	–	Internal pull-down (note 2)

NOTES:

1. TCK is input floating. So TCK must be connected to VDD or GND through the resistor to reduce current consumption in the power down mode.
- 2.

Pin Name	Value	Mode Setting
MD[1:0]	0 0	Normal/ debug mode
	0 1	Flash writing mode (Tool mode)
	1 0	External JTAG (Boundary SCAN)
	1 1	SCAN mode (Only for test)

Table 2-2. Pin Description (Continued)

Module	Pin Name	Function	I/O Type	Active Level	Comments
Power	VDDCORE[1:0]	Core DC supply voltage	I	–	See recommended operating condition
	VSSCORE[1:0]	Core ground voltage	I	–	
	VDDIO[2:0]	I/O block DC supply voltage	I	–	See recommended operating condition
	VSSIO[2:0]	I/O block ground voltage	I	–	
	VDDCOREOUT	From internal regulator	O	–	Connected to GND through a 1uF capacitor
	AVREF	ADC DC reference voltage	I	–	See recommended operating condition
	AVDD	ADC DC supply voltage	I	–	See recommended operating condition
	AVSS	ADC ground voltage	I	–	
	VLCD	DC supply voltage for LCD	I	–	See recommended operating condition
	VPP	Internal flash power test pin	–	–	No connect
	VDDPLL0UT	From internal regulator	O	–	Connected to GND through a 1uF capacitor

NOTE: VLCD must be greater than VDDCORE and VDDIO.

3

MEMORY MAP

When the S3F4A1HR micro-controller is reset, the ARM core is in boot mode to access the internal flash at address 0x00000000. The internal RAM is located at address 0x00300000 and the internal data flash is located at address 0x80000000. But, internal data flash is disabled at reset.

Memory Space	Application	Abort
0xFFFFFFFF – 0xFFE00000	Peripheral devices memory	NO
0xFFDFFFFFF – 0X80008000	Reserved	YES
0X80007FFF – 0X80000000	32Kbytes disabled data FLASH	YES
0X7FFFFFFF – 0X00304000	Reserved	YES
0X00303FFF – 0X00300000	16kbytes internal RAM	NO
0X002FFFFFF – 0X00080000	Reserved	YES
0X0007FFFF – 0X00000000	512kbytes internal program FLASH	NO

Figure 3-1. S3F4A1HR Default Memory Map after Reset

Table 3-1. The Base Address of Peripheral Special Registers

<i>Peripheral</i>	<i>Base Address</i>
DFC	0xFFE00000
IFC	0xFFE04000
PWM (2 channels)	0xFFE08000
ADC 10-bit (16 channels)	0xFFE0C000
SPI0 (8-bit)	0xFFE10000
Watchdog	0xFFE14000
CAN0 (32 buffers)	0xFFE18000
GPT (3 Channels)	0xFFE1C000
ST0 (2 channels)	0xFFE20000
ST1 (2 channels)	0xFFE24000
UART0	0xFFE28000
IOCONF	0xFFE2C000
STT	0xFFE30000
UART1	0xFFE34000
USART0	0xFFE38000
CAN1 (32 buffers)	0xFFE3C000
CAPT0	0xFFE48000
I2C0	0xFFE50000
I2C1	0xFFE54000
LCDC	0xFFE58000
SPI1 (16-bit)	0xFFE60000
PIO0	0xFFE64000
PIO1	0xFFE68000
PIO2	0xFFE6C000
SMC0	0xFFE74000
SMC1	0xFFE78000
SMC2	0xFFE7C000
SMC3	0xFFE80000
SFM	0xFFFFE4000
CM	0xFFFFE8000
IRC	0xFFFFF0000
LDMAC	0xFFFFF8000
GIC	0xFFFFF000

4

MODULE GENERIC FUNCTIONS

1. REGISTERS ACCESS

1.1 ENABLE / DISABLE / STATUS REGISTERS

In order to reduce code size and subsequently increase speed when accessing internal peripherals, most of the registers have been split into 3 address locations:

- The first address location (Enable or Set register) is used to set a bit to a logical 1.
- The second address location (Disable or Clear register) is used to set a bit to a logical 0.
- The third address location (Status register or Mask register) gives the current state of the bit.

To set a bit to a logical 1 in the Status or Mask register, a write command in the Enable or Set register must be performed with the corresponding bit at a logical 1.

To set a bit to a logical 0 in the Status or Mask register, a write command in the Disable or Clear register must be performed with the corresponding bit at a logical 1.

Example:

When supposing that the value of GPT_PSR register is 0x00000000, to enable the TIOB and TCLK pins as PIOs in the GPT block, 0x00050000 must be written in the GPT_PER register. The value read in the GPT_PSR register will be 0x00050000.

Now if the software wants to disable the TIOB pin as a PIO (i.e. enable it for GPT use), a write access to the GPT_PSR register with the value 0x00010000 must be performed. The new value read in the GPT_PSR register will be 0x00010000.

In the following chapters, registers sharing the same behavior will be shown in a single page with one bit table for the Enable or Set and Disable or Clear registers and one bit table for the Status or Mask register. The bit description will be documented after the Status or Mask register.

1.2 KEY ACCESS TO REGISTERS

Some bits in registers can only be set to a value (0 or 1) only if the right key is written at the same time.

Example:

The RSTALW bit in the WD_PWR register can be set to a logical 0 or 1 only if the KEY[7:0] bits are equal to 0x91.

To enable restart mode in the Watchdog, 0x91000001 must be written in the WD_PWR register.

To disable restart mode in the Watchdog, 0x91000000 must be written in the WD_PWR register.

1.3 REGISTERS UNDEFINED BITS

- Undefined bits

Each undefined bit is read as a zero, and writing undefined bits has no consequence. They are marked as '---' and grayed.

- Reserved bits

Writing reserved bit with a different value than the reset value has unpredictable effect. Writing '0' has no effect except express specification. Read back value is the reset state and is 1 or 0 depending on the bit. There are marked 'reserved'.

1.4 GHOST REGISTERS

S3F4A1HR micro-controller integrates an ICE (In-Circuit Emulation) interface that is associated with a JTAG connection and a software debugger that provides powerful debug possibility.

Effectively,

- A running program can be stopped,
- Internal registers and internal / external memories can be monitored,
- Instructions can be added when the core is stopped,
- And also, the program can be resumed.

However, some S3F4A1HR registers are 'read-active', which means that reading such registers can affect state of other registers. This is an usual and wanted register's behavior. For example, in the ADC module, the bit EOC (End Of Conversion) is automatically cleared when the DR (numerical value of the input converted) is read. The aim is to cut off the amount of code needed in an application.

Meanwhile, when debugging software, users can be interested in monitoring the value of such registers, without modify state of another register. For this purpose, for each module, a ghost register field has been implemented in the design.

Users reading in this ghost field will not affect value of any other registers.

Ghost registers are not 'read-active', and are mirrors of original registers. They are located in memory by inverting the 14th bit in the module base address. For example, base address of ADC module is 0xFFE0C000, so ghost registers base address of ADC is 0xFFE0E000. By reading this ghost field, users do not disturb the behavior of ADC module.

Ghost registers exist for all modules.

Find below "read-active" registers list:

Module	"Read-Active" Registers	Effect
GIC	GIC_IVR	Clears IRQ interrupt if present at the GIC
	GIC_FVR	Clears FIQ interrupt if present at the GIC
ADC	ADC_DR	Clears EOC bit in ADC_SR register if set
SPI	SPI_RDR	Clears RDRF bit in SPI_SR register if set

2. POWER MANAGEMENT BLOCK

In order to reduce power consumption, the S3F4A1HR micro-controller provides a power management block in some peripherals used to switch on/off the peripheral clocks (peripheral and PIO block).

This function is independent of the Power Reset Controller (peripheral) used to switch on/off the ARM7TDMI core.

3 registers are provided:

- PERIPHERAL_ECR (at peripheral offset 0x0050) enables the clock
- PERIPHERAL_DCR (at peripheral offset 0x0054) disables the clock
- PERIPHERAL_PMSR (at peripheral offset 0x0058) gives the status of the clock

2 bits are provided in these registers:

- Bit 0 controls the PIO block of the peripheral
- Bit 1 controls the peripheral function

When the peripheral clock is disabled, the clock is immediately stopped. When the clock is re-enabled, the peripheral controller resumes action where it left off.

The table below lists modules which have power management blocks

Module	Power Management Block Present
IRC	No
I2C	Yes
STT	No
SMC	Yes
LCDC	Yes
PIO	Yes
LDMAC	Yes
IFC	Yes
U(S)ART	Yes
ARM core	Yes
SFM	No
WD	No
SPI	Yes
ADC	Yes
GPT	Yes
PWM	Yes
CAN	Yes
ST	Yes
CM	No
GIC	No

5

ANALOG TO DIGITAL CONVERTER

1. OVERVIEW

1.1 FUNCTIONAL DESCRIPTION

The 10-bit Analog to Digital Converter (ADC) module provides the following features:

- 16 x analog inputs: AIN[15:0]
- LDMA transfer
- A programmable conversions sequence: Sequence of analog inputs to be converted. This allows the user to make conversions of some of the 16 inputs in its own order. The length of a sequence (number of conversion) is defined by setting the NBRCH field in the ADC_MR. The composition of sequence is defined in the ADC_CMCR register.
- 2 conversion modes: 'one shot' (or single) mode or 'continuous mode'.
 - In 'one shot' mode, inputs specified in the conversions sequence are successively converted after the start command, conversion results are successively stored in the data register and then ADC stops.
 - In 'continuous mode', inputs specified in the conversions sequence are converted one after the other continuously until a stop is requested. Each time a conversions sequence is completed, the ADC re-starts conversion of the selected inputs in the order of the sequence. In this mode, the microprocessor starts the ADC which is then completely independent.
- Conversions are started by the CPU. Conversions can also be started by an external device using a dedicated input pin. (ADTRG)
- **Analog clock** frequency can be tuned (**less than maximum 2.5MHz**) whichever system clock frequency. This feature allows to configure the sampling frequency of analog inputs. Note that 5 analog clock cycles are required to perform a single conversion.
- Interrupt line connected to the GIC.
- Power management features allowing to reduce power consumption.

1.2 BLOCK DIAGRAM

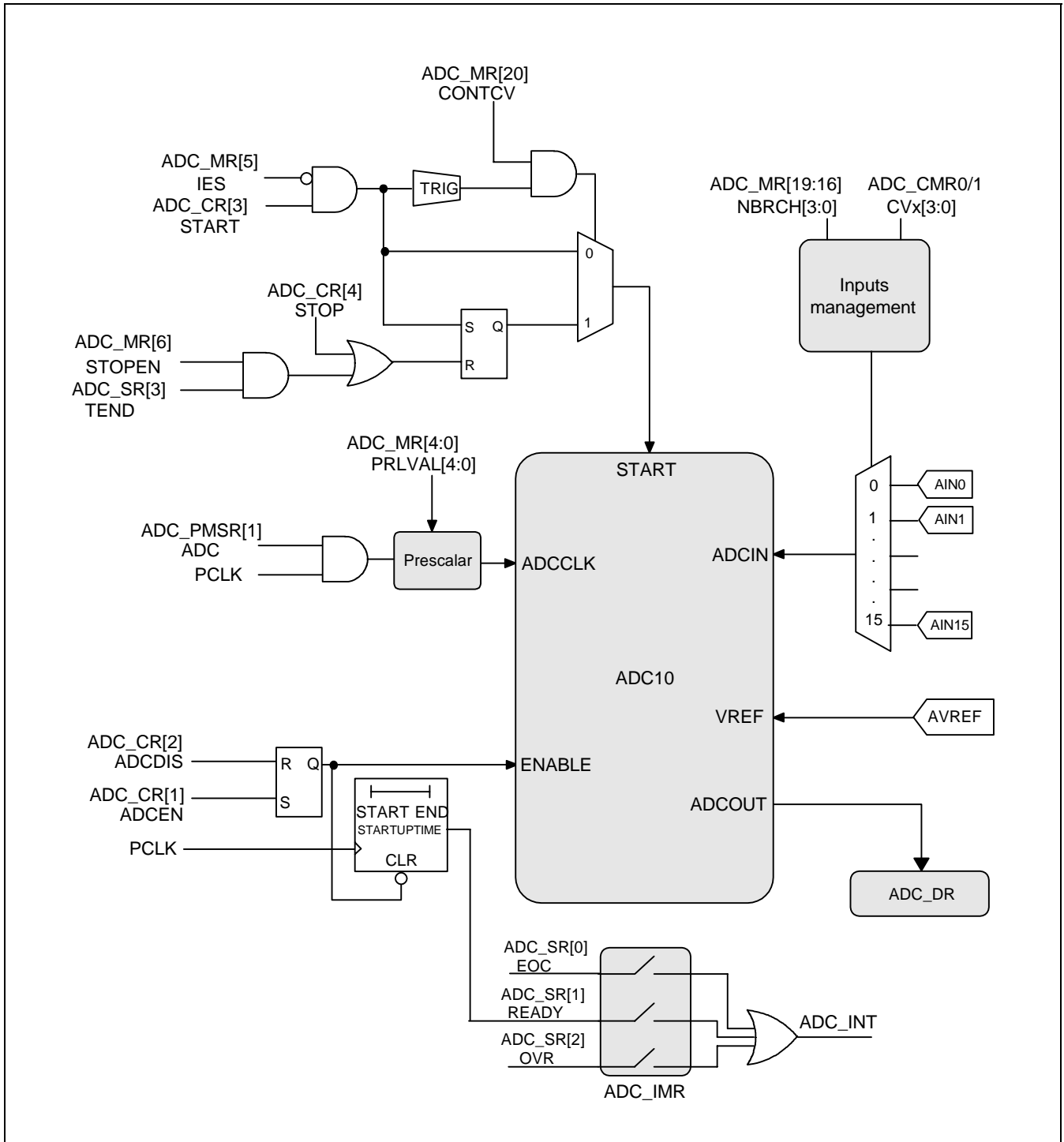


Figure 5-1. ADC Block Diagram

2. EXTERNAL PIN DESCRIPTION

Table 5-1. ADC Pin Description

Pin Name	Function	I/O Type	Active Level	Comments
AIN[15:0]	Analog input	Analog Input	–	–
AVREF	Analog reference voltage	Analog Input	–	–
ADTRG	External start	Digital Input	High	–

3. FUNCTIONAL OPERATION

3.1 ADC DETAILED FUNCTIONALITIES

3.1.1 Conversion Sequence Definition

A conversions sequence is a sequence of analog inputs to be converted. User can configure ADC block to make conversions of some of the 16 inputs in its own order.

The length of the sequence (in other terms the number of conversions) is defined by setting the NBRCH field in the ADC_MR. The following table gives the relation between the NBRCH field and the number of conversion performed in a sequence:

Table 5-2. NBRCH[3:0] Values and the Number of Conversions

NBRCH[3:0]	Number of Conversions
0000	1
0001	2
0010	3
0011	4
0100	5
0101	6
0110	7
0111	8
1000	9
1001	10
1010	11
1011	12
1100	13
1101	14
1110	15
1111	16

This means that, even configured in 'one shot' mode, the ADC will run the specified number of conversion after a start request. The data register will be updated with the conversion results by the end of each conversion which composes the sequence.

The composition of a sequence is programmed in the ADC_CM Rx register. The CV1 field defines the first input to be converted in the sequence. The CV2 field defines the second input to be converted and so on. Find in the table below the relation between the CVx values and input selected:

Table 5-3. CVx Values and Selected Input

CVx Values	Selected Analog Input	Selected Pin
0000	Input 0	AIN 0
0001	Input 1	AIN 1
0010	Input 2	AIN 2
0011	Input 3	AIN 3
0100	Input 4	AIN 4
0101	Input 5	AIN 5
0110	Input 6	AIN 6
0111	Input 7	AIN 7
1000	Input 8	AIN 8
1001	Input 9	AIN 9
1010	Input 10	AIN 10
1011	Input 11	AIN 11
1100	Input 12	AIN 12
1101	Input 13	AIN 13
1110	Input 14	AIN 14
1111	Input 15	AIN 15

For example, assuming that:

- NBRCH = 0x2,
- CV1 = 0x5, CV2 = 0x2 and CV3 = 0x0

After a start conversion request, ADC converts input 5 (AIN5), followed by input 2 (AIN2) and it finishes by converting input 0 (AIN0).

3.1.2 One shot or Continuous Conversion Mode

The ADC can be programmed in two modes: one shot and continuous conversion mode.

One shot conversion mode is enabled by setting CONTCV bit of mode register to '0'. In this mode, upon conversion start request, the ADC performs the only complete conversions sequence and then stops and waits for another start request.

The ADC can not be stopped until it finishes the conversions sequence.

Continuous conversion mode is enabled by setting CONTCV bit of mode register to '1'. In this mode, upon conversion start request, the ADC repetitively performs conversions sequences until it is forced to stopped. To stop continuous conversion, the CPU must write STOP bit in the control register.

When a stop is requested, the ADC finishes its current conversion and updates the data register with this last conversion result. No other conversions are performed even if the conversions sequence is not finished.

5-1. IMPORTANT NOTICE

However, user should be vigilant, because after a stop command in continuous mode, the ADC finishes the on-going conversion and this may look like to an extra conversion.

3.1.3 ADC Start Sources

Conversions are started by the CPU (writing the START bit in the ADC_CR).

Conversions can also be started by an external device using a dedicated input pin (ADTRG).

Internal start is enabled if IES bit of mode register is '0'. A conversion request is initiated by writing START bit of control register.

External start is enabled if IES bit of mode register is '1'. A conversion request is initiated each time a rising edge is detected on "ADTRG" pin. The following constraints apply on "ADTRG" input:

- There is no setup / hold constraint on "start" versus system clock as far as anti- metastability structure is provided by ADC.
- "ADTRG" signal must be stable more than one system clock period.

3.1.4 Analog Cell Clock Frequency

As specified in the electrical characteristics (see Table "ADC characteristics"), the analog cell clock frequency is limited to 2.5MHz, whereas system clock is usually far higher. So, the ADC module provides a clock frequency divider based on a 6-bit counter. The following expression gives the relation between system clock frequency, analog cell clock frequency and the PRLVAL field of mode register:

If PRLVAL is set to '0' or '1', then $FANA = FCLK_B / 4$
else for any other values of PRLVAL, $FANA = FCLK_B / (2 * PRLVAL)$

The PRLVAL data must be chosen in order to ensure that analog frequency is lower than 2.5MHz. Note that 5 analog clock cycles are required to perform a single conversion. The ADC clock rate must not be exceeded 2.5 MHz.

3.1.5 Power Management

The ADC peripheral includes power management features that can be used to minimize power consumption. Power can be saved on two sides: analog and digital.

- Analog power saving: To reduce analog power consumption, CPU shall disable the ADC module (write ADCDIS bit in ADC_CR) which has for effect to set the analog cell to 'standby' mode.
- Digital power saving: To reduce digital power consumption, CPU shall disable the ADC clock (write ADC bit in ADC_DCR) which has for effect to disable all incoming clocks. Then, digital consumption is reduced close to 0. Note that when the clock is disabled, any write access to one of the ADC registers is ineffective except for the 'Enable Clock Register'. Read access to registers is still enabled.

Then, in order to have the ADC peripheral set to lowest consumption mode, it is necessary to disable it (write ADCDIS bit in ADC_CR) first and then switch off the clock (write ADC bit in ADC_DCR). The other way round, in order to completely leave the lowest power consumption mode, it is necessary to enable the clock (write ADC bit in ADC_ECR) first and then enable the ADC (write ADCEN bit in ADC_CR).

The following table summarizes the status bits reflecting power management:

Table 5-4. Status Bits Reflecting Power Management

Status Bit in Register	When set to "1"	When set to "0"
ADC bit in ADC_PMSR	Clock is enabled	Clock is disabled. Digital power saving
ADCENS bit in ADC_SR	Analog cell is active	Analog cell is in standby mode. Analog power saving

3.1.6 Interrupts

The ADC peripheral generates an interrupt if at least one of EOC, READY, OVR or TEND bit is active (set to '1') in the status register while it is enabled (corresponding bit in ADC_IMR is read as '1').

Each interrupt bit can be enabled or disabled using respectively the interrupt enable register and interrupt disable register.

3.1.7 Conversion Details

3.1.7.1 EOC Flag (End Of Conversion)

The EOC bit of the status register flags the presence of a new valid data in data register.

- If EOC is '0', it means that no conversion has been done since last reset of this bit or that the last conversion result has been read by the CPU in the data register.
- If EOC is '1', it means that a conversion has been completed and the new conversion result has not been read yet in data register.

EOC flag is reset to '0' each time a read access is performed in data register (ADC_DR).

3.1.7.2 READY Flag

The READY bit of the status register means that the ADC is ready to accept any conversion start request. This bit is read as '0' while ADC is currently running a conversion.

3.1.7.3 OVR Flag (Overrun)

This flag indicates that a new data overwrites a previous converted data which has not been read. The previous data has been lost.

OVR flag can be clear by CPU (writing OVR bit in clear status register).

3.2 SOFTWARE SEQUENCE FOR CONVERSION

The following lines list the basic sequence of operations after a reset for using ADC peripheral:

- Enable the clock in the ADC_ECR.
- Configure ADC mode in the ADC_MR. PRLVAL field shall be programmed in order to have an analog frequency clock less than 2.5MHz. Indicate if the conversion is or not continuous. Define the conversions sequence:
number of conversion (NBRCH field in ADC_MR) and which inputs are converted (CVx fields in ADC_CMRx).
- Enable ADC (ADC_EN in ADC_CR).
- Wait for READY bit in ADC_SR. When the flag is set, the ADC is ready to start conversion. An interrupt can be generated when the READY flag rises up if the corresponding bit is enabled in the ADC_IMR.
- Initiate conversion by writing START bit in the ADC_CR.
- ADC selects the analog input associated with conversion number 1 of conversion sequence.
- The analog input voltage is sampled and conversion completes after 5 ADC clock cycles from the start command. The digital 10-bit conversion result is stored into ADC_DR and the EOC bit in ADC_SR rises up. If EOC flag was already set, then the OVR bit is also set.
- The CPU can then read the digital value in ADC_DR, which automatically clears the EOC. If the CPU decides that no more data should be converted in continuous mode, then it respectively writes the STOP bit. In this case, the ADC stops operating and waits for next start request. Note that in 'one shot' mode, the ADC can not be stopped until the end of all conversions specified in the conversion sequence.
- If NBRCH is non-null, the ADC selects the analog input associated to the next conversion number and the operation flow restarts from step 6.
- If CONTCV is read as '1', the ADC restarts another conversion sequence from step 5.

4. REGISTERS DESCRIPTION

Base Address – 0xFFE0C000

Table 5-5. ADC Special Function Registers

Offset Address	Name	Description	R/W	Reset State
0x000 – 0x04C	–	Reserved	–	–
0x050	ADC_ECR	Enable clock register	W	–
0x054	ADC_DCR	Disable clock register	W	–
0x058	ADC_PMSR	Power management status register	R	(note)
0x05C	–	Reserved	–	–
0x060	ADC_CR	Control register	W	–
0x064	ADC_MR	Mode register	R/W	0x00000000
0x068	–	Reserved	–	–
0x06C	ADC_CSR	Clear status register	W	–
0x070	ADC_SR	Status register	R	0x00000000
0x074	ADC_IER	Interrupt enable register	W	–
0x078	ADC_IDR	Interrupt disable register	W	–
0x07C	ADC_IMR	Interrupt mask register	R	0x00000000
0x080	ADC_CMRO	Conversion mode register 0	R/W	0x00000000
0x084	ADC_CMRI	Conversion mode register 1	R/W	0x00000000
0x088	ADC_DR	Convert data register	R	0x00000000

NOTE: The reset value of the register depends on IP identifier code.

ADC Enable Clock Register

ADC_ECR (0x050)

Access: Write only

31	30	29	28	27	26	25	24
DBGEN	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	ADC	–
W	W	W	W	W	W	W	W

- **ADC : ADC clock enable**

0: No effect

1: Enable ADC clock

- **DBGEN : Debug mode enable**

0: No effect

1: Enable debug mode

ADC Disable Clock Register

ADC_DCR (0x054)

Access: Write only

31	30	29	28	27	26	25	24
DBGEN	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	ADC	–
W	W	W	W	W	W	W	W

- **ADC : ADC clock disable**

0: No effect

1: Disable ADC clock

- **DBGEN : Debug mode disable**

0: No effect

1: Disable debug mode

ADC Power Management Status Register

ADC_PMSR (0x058)

Access: Read only

31	30	29	28	27	26	25	24
DBGEN	–	IPICODE[25:20]					
R-0	R-0	R-U	R-U	R-U	R-U	R-U	R-U
23	22	21	20	19	18	17	16
IPICODE[19:12]							
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U
15	14	13	12	11	10	9	8
IPICODE[11:4]							
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U
7	6	5	4	3	2	1	0
IPICODE[3:0]				–	–	ADC	–
R-U	R-U	R-U	R-U	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **ADC : ADC clock status**

0: ADC clock disabled.

1: ADC clock enabled.

- **IPICODE[25:0] : IP identifier code**

This field contains the version number of the module, coded on 26bits.

- **DBGEN : Debug mode**

0: ADC is not halted during ARM core debug mode.

1: ADC is halted during ARM core debug mode.

ADC Control Register **ADC_CR (0x060)** **Access: Write only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	STOP	START	ADCDIS	ADCEN	SWRST
W	W	W	W	W	W	W	W

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **SWRST : ADC software reset**

0: No effect.

1: Reset of the ADC peripheral.

When a software reset is performed, all the registers of the peripheral are reset except ADC_PMSR (Power Management Status) register.

- **ADCEN : ADC enable**

0: No effect.

1: ADC is enabled for conversion.

- **ADCDIS : ADC disable**

0: No effect.

1: ADC is disabled (Standby Mode).

In case both ADCEN and ADCDIS are equal to one when the control register is written, the ADC will be disabled.

- **START : Start conversion**

0: No analog to digital conversion to be started.

1: Begin analog to digital conversion, clears EOC bit.

NOTE: Before starting conversions, users should ensure that ADC is ready for conversion (READY bit is set to logical one in ADC_SR).

- **STOP : Stop conversion in continuous conversion**

0: No effect.

1: Stop the continuous conversion.

ADC Mode Register

ADC_MR (0x064)

Access: Read/Write

31	30	29	28	27	26	25	24	
–	–	–	–	–	–	–	–	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
23	22	21	20	19	18	17	16	
–	–	–	CONTCV	NBRCH[3:0]				
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
15	14	13	12	11	10	9	8	
–	–	–	–	–	–	–	–	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
7	6	5	4	3	2	1	0	
–	–	IES	PRLVAL[4:0]					–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

W: Write

R: Read

-0: After reset

-1: After reset

-U: Undefined after reset

- **PRLVAL[4:0] : Preload Value**

A division of the Master clock (Coreclk) determines ADC_clk. The preload value is chosen by the user to adapt the Master clock to the ADC peripheral as well as possible. It is the start value of the down counter. The LSB are fixed to 0 because this value has to be even parity to guaranty a duty cycle of ½, so the user has only to initialize the 5 MSB.

If (PRLVAL == 0 or PRLVAL == 1) FADC = PCLK / 4

Else FADC = PCLK / (2*PRLVAL)

NOTE: The clock rate to the ADC must not be exceeded 2.5 MHz.

- **IES : Internal/External Start**

0: Internal start.

1: External start.

- **NBRCH[3:0] : Number of Conversions**

NOTE: Even in one shot mode, ADC will run multiple conversions if the NBRCH[3:0] is greater than 0000b.

Table 5-6. NBRCH[3:0] Values and the Number of Conversions

NBRCH[3:0]	Number of Conversions
0000	1
0001	2
0010	3
0011	4
0100	5
0101	6
0110	7
0111	8
1000	9
1001	10
1010	11
1011	12
1100	13
1101	14
1110	15
1111	16

- **CONTCV : Continuous Conversion**

- 0: One shot mode. ADC converts as much inputs as specified by the NBRCH[3:0] in the order specified in the ADC_CMCR, and stops.
- 1: Continuous mode. ADC converts as much inputs as specified by the NBRCH[3:0] in the order specified in the ADC_CMCR, and repeats. This bit is initialized to 0.

NOTE: In continuous mode, after a stop command, the ADC finishes the on-going conversion and this may look like to an extra conversion.

ADC Clear Status Register

ADC_CSR (0x06C)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	OVR	–	–
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **OVR : Clear overrun interrupt**

0: No effect.

1: Clear OVR interrupt.

ADC Status Register **ADC_SR (0x70)** **Access: Read only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	CTCVS	ADCENS
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
–	–	–	–	–	OVR	READY	EOC
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

W: Write R: Read -0: After reset -1: After reset -U: Undefined after reset

- **EOC : End of conversion**

0: Conversion not complete or inactive.

1: Conversion complete, data in ADC_DR is valid. This bit is cleared when the ADC_DR is read.

- **READY : ADC ready for conversion**

0: ADC ignores start or stop command: it is not ready for conversion or is converting data.

1: ADC is ready to start a conversion.

To explain more completely the ready flag, let's call "working" the event high when ADC is converting data and "analog_ready" the event low when analog part is disabled or in initializing phase.

Table 5-7. Ready for Conversion

analog_ready	working	ready_flag
0	0	0
0	1	0
1	0	1
1	1	0

- **OVR : Overrun**

0: Zero or One data has been converted by ADC since last ADC_DR read.

1: At least two data has been converted by ADC since last ADC_DR read.

- **ADCENS : ADC enable status**

0: ADC is disabled.

1: ADC is enabled.

- **CTCVS : Continuous mode status**

0: One shot mode with help of microprocessor.

1: Continuous mode, the peripheral is stand-alone. This bit is initialized to 0 and changes when there is a change of mode. This bit never generates any interruption.

ADC Interrupt Enable Register

ADC_IER (0x074)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	OVR	READY	EOC
W	W	W	W	W	W	W	W

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **EOC : End of conversion interrupt enable**

0: No effect

1: Enable EOC interrupt

- **READY : ADC ready for conversion interrupt enable**

0: No effect

1: Enable READY interrupt

- **OVR : Overrun interrupt enable**

0: No effect

1: Enable OVR interrupt

ADC Interrupt Disable Register

ADC_IDR (0x078)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	OVR	READY	EOC
W	W	W	W	W	W	W	W

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **EOC : End of conversion interrupt disable**

0: No effect

1: Disable EOC interrupt

- **READY : ADC ready for conversion interrupt disable**

0: No effect

1: Disable READY interrupt

- **OVR : Overrun interrupt disable**

0: No effect

1: Disable OVR interrupt

ADC Interrupt Mask Register **ADC_IMR (0x07C)** **Access: Read only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
–	–	–	–	–	OVR	READY	EOC
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **EOC : End of conversion interrupt mask**

0: EOC interrupt is disabled.

1: EOC interrupt is enabled.

- **READY : ADC ready for conversion interrupt mask**

0: READY interrupt is disabled.

1: READY interrupt is enabled.

- **OVR : Overrun interrupt mask**

0: OVR interrupt is disabled.

1: OVR interrupt is enabled.

ADC Conversion Mode Register 0

ADC_CMRO (0x080)

Access: Read/Write

31	30	29	28	27	26	25	24
CV8[3:0]				CV7[3:0]			
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
CV6[3:0]				CV5[3:0]			
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
CV4[3:0]				CV3[3:0]			
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
CV2[3:0]				CV1[3:0]			
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- CVx[3:0] : Analog Input Selection**

x = {1, 2, 3, 4, 5, 6, 7, 8} is the conversion number.

CVx Values	Selected Analog Input	Selected Analog Pin
0000	Input 0	AIN 0
0001	Input 1	AIN 1
0010	Input 2	AIN 2
0011	Input 3	AIN 3
0100	Input 4	AIN 4
0101	Input 5	AIN 5
0110	Input 6	AIN 6
0111	Input 7	AIN 7
1000	Input 8	AIN 8
1001	Input 9	AIN 9
1010	Input 10	AIN 10
1011	Input 11	AIN 11
1100	Input 12	AIN 12
1101	Input 13	AIN 13
1110	Input 14	AIN 14
1111	Input 15	AIN 15

ADC Conversion Mode Register 1

ADC_CM1 (0x084)

Access: Read/Write

31	30	29	28	27	26	25	24
CV16[3:0]				CV15[3:0]			
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
CV14[3:0]				CV13[3:0]			
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
CV12[3:0]				CV11[3:0]			
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
CV10[3:0]				CV9[3:0]			
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- CVx[3:0] : Analog Input Selection**

x = {9, 10, 11, 12, 13, 14, 15, 16} is the conversion number.

CVx Values	Selected Analog Input	Selected Analog Pin
0000	Input 0	AIN 0
0001	Input 1	AIN 1
0010	Input 2	AIN 2
0011	Input 3	AIN 3
0100	Input 4	AIN 4
0101	Input 5	AIN 5
0110	Input 6	AIN 6
0111	Input 7	AIN 7
1000	Input 8	AIN 8
1001	Input 9	AIN 9
1010	Input 10	AIN 10
1011	Input 11	AIN 11
1100	Input 12	AIN 12
1101	Input 13	AIN 13
1110	Input 14	AIN 14
1111	Input 15	AIN 15

6

CAPTURE MODULE (CAPT)

1. OVERVIEW

1.1 FUNCTIONAL DESCRIPTION

The capture module is a frame analyzer. It stores the period duration or the high and low level duration by 15bit capture counter (CAP_DR). Those durations are described in number of counter cycle (CAPTCLK). The capture clock CAPTCLK is equal to PCLK divided by a configurable pre-scalar value. The capture allows a data transfer with the LDMA.

1.2 EXTERNAL PIN DESCRIPTION

Table 6-1. CAPT Pin Description

Pin Name	Function	I/O Type	Active Level	Comments
CAPT0	Capture input	I	–	–

2. FUNCTIONAL OPERATION

2.1 MODE OF OPERATION

2.1.1 General Description

The capture clock frequency is equal to:

$CAPTCLK = PCLK / 2^{PRESCALAR[3:0]} - 1$, where :

- PCLK is the peripheral clock frequency
- PRESCALAR[3:0] is a 16 bits data written in the CAPT_MR

It is possible to choose among 3 modes of measurement:

- To measure the duration between each edges (positive and negative).
- To measure the duration between positives edges (period).
- To measure the duration between negatives edges (period).

If an overrun occurs, it is possible to choose to overwrite the data stored in the Data Register (CAPT_DR) or to stop the data acquisition through the mode register (CAPT_MR).

The DATACAPT bit (in the CAPT_SR register) is automatically cleared (i.e. set to a logical 0) after reading of the data register.

It is recommended to disable the capture module after every modification of the CAPT_MR register. Otherwise the first measure could be false.

When the capture is disabled, the capture counter is reset.

2.2 CAPTURE'S LIMITS

To avoid a capture to miss frame edges, the capture counter clock frequency CAPTCLK must be chosen so that:

$CAPTCLK > 2/t_{b2edge_MIN}$, where t_{b2edge_MIN} is the minimum "time between two edges" observed in the frame. Moreover the capture detects each frame edge with a delay equal to the CAPTCLK period.

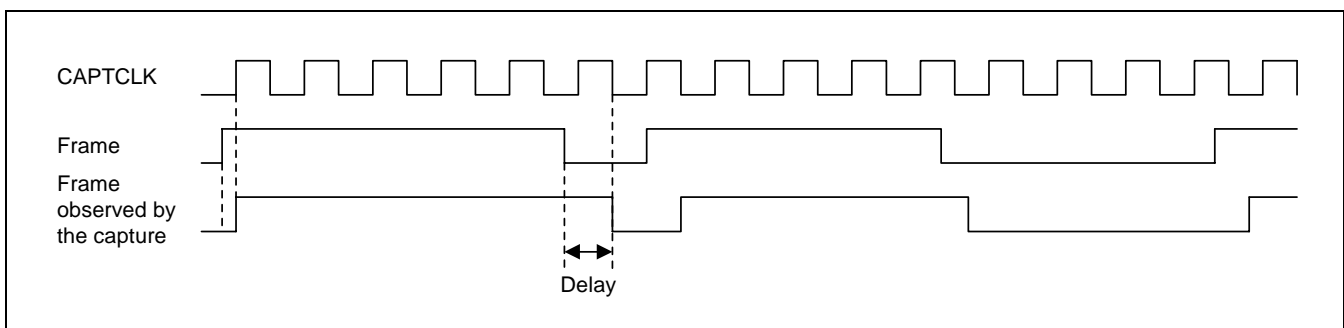


Figure 6-1. Capture Pin Resynchronization

2.3 PROGRAMMING EXAMPLES

Example for the usage of the capture:

Capture of one period of a signal using the LDMA and the associated interrupt. The PCLK frequency is equal to 20MHz; the capture clock CAPTCLK maximum frequency is 10MHz (CAPT_MR PRESCALAR[3:0] = 0). In these conditions, the observed signal maximum frequency is 5MHz.

Configuration:

- Capture clock enable by writing bit CAP in CAPT_ECR
- Software reset of the capture peripheral to be in a known state by writing bit SWRST in CAPT_CR.
- Capture mode register (CAPT_MR) configuration: PRESCALAR[3:0] field set to 0 so that CAPTCLK = 10MHz. Set ONESHOT to 1. OVERMODE is ignored because the capture will stop after its first acquisition. MEASMODE=2 to measure the duration between two positive edges.

- Enable the CAPT by writing bit CAPEN in CAPT_CR

Configuration of CAPT_IER: An interrupt is generated at the end of the capture. When the LDMA will finish the capture, an interrupt will be generated. GIC must be configured

- If status bit CAPENS is set in CAPT_SR start the capture by writing bit STARTCAPT in CAPT_CR, otherwise users should wait the CAPENS flag, consequence of bit CAPEN in CAPT_CR.

Interruption Handling:

- IRQ entry and call C function.
- Read CAPT_SR and verify the source of the interrupt.
- Clear the corresponding interrupt at peripheral level by writing in the CAPT_CSR.

Interrupt treatment: Read the duration between the two positive edges in the received memory space programmed in the LDMA. The duration is expressed by the number of capture clock (duration/10 MHz).

- IRQ exit

3. REGISTERS DESCRIPTION

Base Addresses – CAPT0 : 0xFFE48000

Table 6-2. CAPTURE Special Function Registers

Offset Address	Name	Description	R/W	Reset State
0x0000 – 0x004C	–	Reserved	–	–
0x0050	CAPT_ECR	Enable clock register	W	–
0x0054	CAPT_DCR	Disable clock register	W	–
0x0058	CAPT_PMSR	Power management status register	R	(note)
0x005C	–	Reserved	–	–
0x0060	CAPT_CR	Control register	W	–
0x0064	CAPT_MR	Mode register	R	0x00000000
0x0068	–	Reserved	–	–
0x006C	CAPT_CSR	Clear status register	W	–
0x0070	CAPT_SR	Status register	R	0x00000000
0x0074	CAPT_IER	Interrupt enable register	W	–
0x0078	CAPT_IDR	Interrupt disable register	W	–
0x007C	CAPT_IMR	Interrupt mask register	R	0x00000000
0x0080	CAPT_DR	Data register	R	0x00000000

NOTE: The reset value of register depends on IP identifier code.

CAPTURE Enable Clock Register

CAP_ECR(0x0050)

Access: Write only

31	30	29	28	27	26	25	24
DBGEN	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	CAP	–
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **CAP : Capture clock enable**

0: No effect

1: Enable Capture clock

- **DBGEN : Debug mode enable**

0: No effect

1: Enable debug mode.

CAPTURE Disable Clock Register

CAP_DCR (0x0054)

Access: Write only

31	30	29	28	27	26	25	24
DBGEN	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	CAP	–
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **CAP : Capture clock disable**

0: No effect

1: Disable Capture clock

- **DBGEN : Debug mode disable**

0: No effect

1: Disable debug mode.

CAPTURE Power Management Status Register CAP_PMSR (0x0058)

Access: Read only

31	30	29	28	27	26	25	24
DBGEN	–	IPICODE[25:20]					
R-0	R-0	R-U	R-U	R-U	R-U	R-U	R-U
23	22	21	20	19	18	17	16
IPICODE[19:12]							
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U
15	14	13	12	11	10	9	8
IPICODE[11:4]							
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U
7	6	5	4	3	2	1	0
IPICODE[3:0]				–	–	CAP	–
R-U	R-U	R-U	R-U	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **CAP : Capture clock status**

0: Capture clock disabled.

1: Capture clock enabled.

- **IPICODE[25:0]**

This field contains the version number of the module, coded on 26 bits.

- **DBGEN : Debug mode**

0: Capture module is not halted during ARM core debug mode.

1: Capture module is halted during ARM core debug mode.

CAPTURE Control Register **CAP_CR(0x0060)** **Access: Write only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	STARTCAPT	CAPDIS	CAPEN	SWRST
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **SWRST : CAPTURE software reset**

0: No effect

1: Reset the CAPTURE

A software triggered by hardware reset of the CAPTURE is performed. It reset all the registers.

- **CAPEN: CAPTURE enable**

0: No effect.

1: Enables the CAPTURE.

- **CAPDIS : CAPTURE disable**

0: No effect.

1: Disables the CAPTURE.

In case both CAPEN and CAPDIS are equal to one when the control register is written the CAPTURE will be disabled.

- **STARTCAPT : START CAPTURE**

0: No effect.

1: The CAPTURE starts a new capture. The start is effective only if capture is previously enabled. (CAPEN is set.)

CAPTURE Mode Register CAP_MR (0x0064) Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
ONESHOT	OVERMODE	MEASMODE		PRESCALAR			
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

• **PRESCALAR[3:0] : pre-scalar factor for the CAPTCLK**

$$CAPTCLK = PLCK / 2^{PRESCALAR+1}$$

• **MEASMODE[1:0] : Measurement Mode**

MEASMODE[1:0]		Measure realized
0	X	Measure between each edges (positive and negative)
1	0	Measure between positives edges.
1	1	Measure between negatives edges.

• **OVERMODE : Overrun mode**

- 0: When the overrun is happened, the capture stops writing on the Data Register. If DATACAPT bit of the CAPT_SR is enabled and the CAPTURE module receives a new data, data register will not be refreshed.
- 1: When the overrun is happened, the capture does not stop writing on the Data Register.

• **ONESHOT : One Shot**

- 0: The capture still captures a frame variation.
- 1: The module captures a frame variation and stop. To ask for other capture, the STARTCAPT bit has to be set to 1 in the CAP_CR.

CAPTURE Clear Status Register

CAP_CSR (0x006C)

Access : Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	OVERFLOW	OVERRUN	–
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **OVERRUN: Clear Overrun Interrupt**

0: No effect.

1: Clear the OVERRUN interrupt.

- **OVERFLOW: Clear Overflow Interrupt**

0: No effect.

1: Clear the OVERFLOW interrupt.

CAPTURE Status Register **CAP_SR (0x0070)** **Access: Read only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	CAPENS
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
–	–	–	–	DATACAPT	OVERFLOW	OVERRUN	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **OVERRUN: Over run**

0: No effect

1: An overrun has occurred.

Overrun indicates a valid data was not read when an overwriting occurred.

- **OVERFLOW: Over flow**

0: No effect

1: An overflow has occurred.

Overflow indicates that the counter of the duration was saturated.

- **DATACAPT : Data Captured**

0: No effect

1 Data in CAP_DR has to be read.

This bit is cleared by reading the CAP_DR register.

- **CAPENS : Capture enable status**

0: Capture is disabled.

1: Capture is enabled.

CAPTURE Interrupt Enable Register

CAP_IER (0x0074)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	DATACAPT	OVERFLOW	OVERRUN	–
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **OVERRUN : Over run interrupt enable**
0: No effect
1: Enables the OVERRUN interrupt.
- **OVERFLOW : Over flow interrupt enable**
0: No effect
1: Enables the OVERFLOW interrupt.
- **DATACAPT : Data captured interrupt enable**
0: No effect.
1: Enables DATACAPT interrupt.

CAPTURE Interrupt Disable Register

CAP_IDR (0x0078)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	DATACAPT	OVERFLOW	OVERRUN	–
W	W	W	W	W	W	W	W

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **OVERRUN : Over run interrupt disable**

0: No effect

1: Disables the OVERRUN interrupt.

- **OVERFLOW : Over flow interrupt disable**

0: No effect

1: Disables the OVERFLOW interrupt.

- **DATACAPT : Data captured interrupt disable**

0: No effect.

1: Disables DATACAPT interrupt.

CAPTURE Interrupt Mask Register

CAP_IMR (0x007C)

Access: Read only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
–	–	–	–	DATACAPT	OVERFLOW	OVERRUN	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **OVERRUN : Over run interrupt mask**

0: OVERRUN Interrupt is disabled.

1: OVERRUN interrupt is enabled.

- **OVERFLOW : Over flow interrupt mask**

0: OVERFLOW Interrupt is disabled.

1: OVERFLOW interrupt is enabled.

- **DATACAPT : Data captured interrupt mask**

0: DATACAPT interrupt is disabled.

1: DATACAPT interrupt is enabled.

CAPTURE DATA Register

CAP_DR (0x0080)

Access: Read only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
LEVEL	DURATION						
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
DURATION							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset***NOTES:**

1. When reading this register, DATACAPT bit is clear in the CAPT_SR.
2. When debugging, to avoid clearing DATACAPT bit, users should use ghost registers.

- **DURATION : Capture duration**

Numbers of CAPTCLK during which the output is at the level indicates by the LEVEL bit, or during a frame period, depending of MEASMODE in CAP_MR.

- **LEVEL : Level measured (NOTE)**

- 0: The duration concerns a low level
 1: The duration concerns a high level

NOTE: That if the MEASMODE[1:0] (of the Mode register) equal 1X the LEVEL bit is used as a Duration bit

7

CLOCK MANAGER

1. OVERVIEW

1.1 FUNCTIONAL DESCRIPTION

This module delivers all clock, clock enables and reset signals used by on-chip blocks composing an emulated ARM-based device.

The Clock Manager provides:

- 1 MHz internal low frequency oscillator (Internal Ring Oscillator)
- 4 MHz or 6 MHz master oscillator
- Programmable PLL operational frequency up to 40 MHz
- Programmable master clock divider
- Programmable ring clock divider
- Master oscillator failure detection feature (referred as "clock monitor" in this manual).
- Internal Power-On-Reset
- Power supply monitoring by LVD

The Clock Manager manages the clock sources selection and transitions:

- To manage the master oscillator stabilization time
- To manage the low frequency oscillator stabilization time
- To manage the PLL stabilization time
- To manage the peripheral clock
- To manage the system clock
- To manage the GIC clock
- To manage the ARM Core clock

The clock manager manages the circuit for reset sources:

- Power-On-Reset
- External asynchronous hardware reset
- Power supply monitor reset
- Clock monitor reset
- Watchdog reset

1.2 BLOCK DIAGRAM

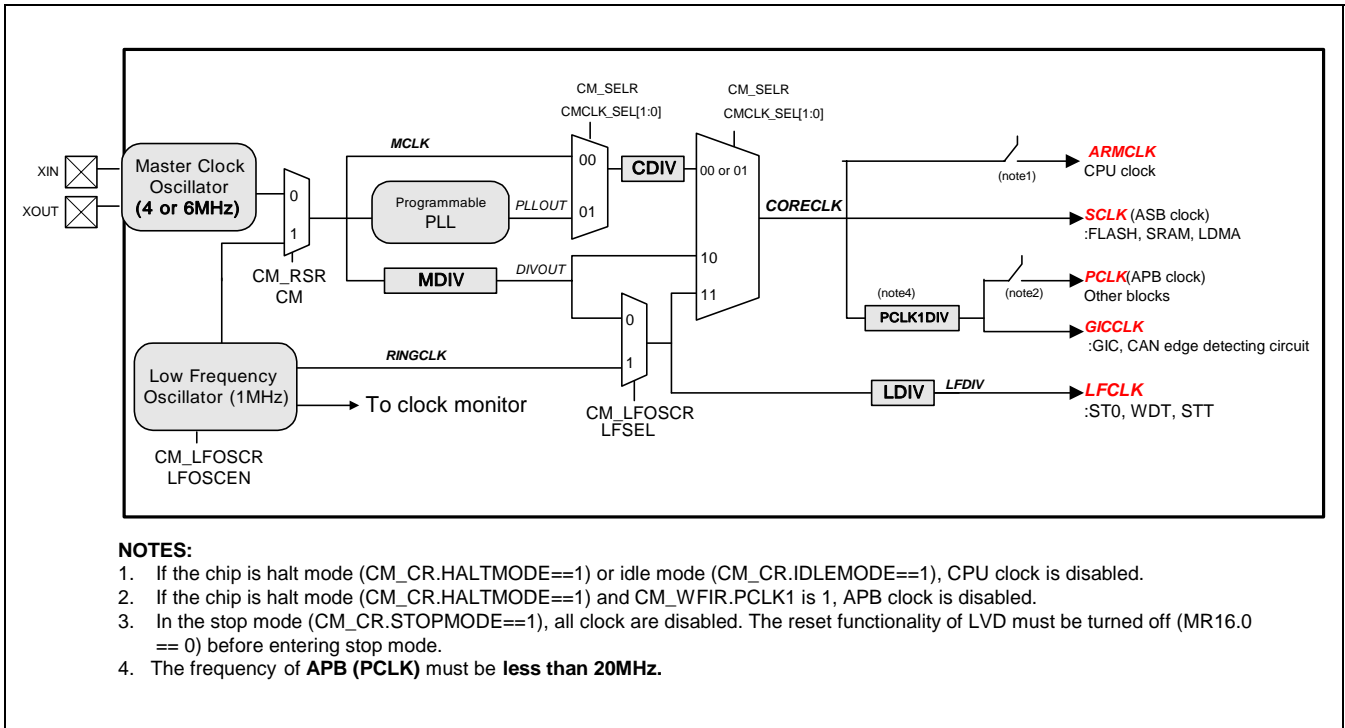


Figure 7-1. Clock Manager Block Diagram

The clock manager allows the user to select between the different operating modes: Normal Mode (CMCLK_SEL == 00), High-speed Mode (CMCLK_SEL == 01), Slow Mode (CMCLK_SEL == 10), Lower Power Mode (CMCLK_SEL == 11), HALT Mode, IDLE mode, STOP mode. Moreover, upon master oscillator failure detection, the clock manager will force the circuit to reset and restart in CKFAIL mode.

As shown in previous synoptic, the clock-manager is responsible for the generation of 5 global clocks:

Table 7-1. Clock Sources

Name	Definition	Comments
SCLK	ARM7 sub-system clock	This clock is running actually the clock used by all internal modules connected to the embedded ASB system bus (except the ARM7 core).
ARMCLK	Clock used by the ARM7TDMI core	This clock is almost the same as SCLK (same frequency) except that it is always cut in HALT mode and IDLE mode whereas SCLK may be left active.
PCLK	Peripheral clock source	This clock is used by all APB bus peripherals. From this clock source, it is possible to defined locally in each peripheral is actually using the clock by programming the module level power management register (PMSR).
GICCLK	Clock dedicated to GIC (Generic Interrupt Controller)	This clock is almost the same as PCLK (same frequency) except that it is always running in HALT mode whereas PCLK may be cut.
LFCLK	Low frequency clock	This clock is always running and it is generated from divided main clock or divided low frequency oscillator. This clock can be programmed through the CM_MDIVR register.(refer to this register's description for details)

Each of those global clocks will be derived from the following clock sources:

Table 7-2. Derived Clock Sources

Name	Definition	Comments
MCLK	This clock is the output of the master oscillator	The master oscillator is an internal oscillator using an external crystal connected on XIN/XOUT pins.
RINGCLK	This clock is the output of the ring-oscillator	The ring oscillator is a fully embedded 1 MHz (typical) oscillator.
PLLOUT	This clock is output of the internal programmable PLL	The PLL output frequency can be programmed up to 40 MHz. This is done through the CM_PDPR register.(refer to this register's description for details)
DIVOUT	This clock is derived from MCLK through frequency divider	The DIVOUT frequency can be programmed through the CM_MDIVR register (refer to this register's description for details).
LFDIV	This clock is derived from RINGCLK through frequency divider	The LFDIV frequency can be programmed through the CM_MDIVR register (refer to this register's description for details).

The following drawing displays the operating mode state diagram with associated possible transitions. Each of those transitions will be detailed further in this document.

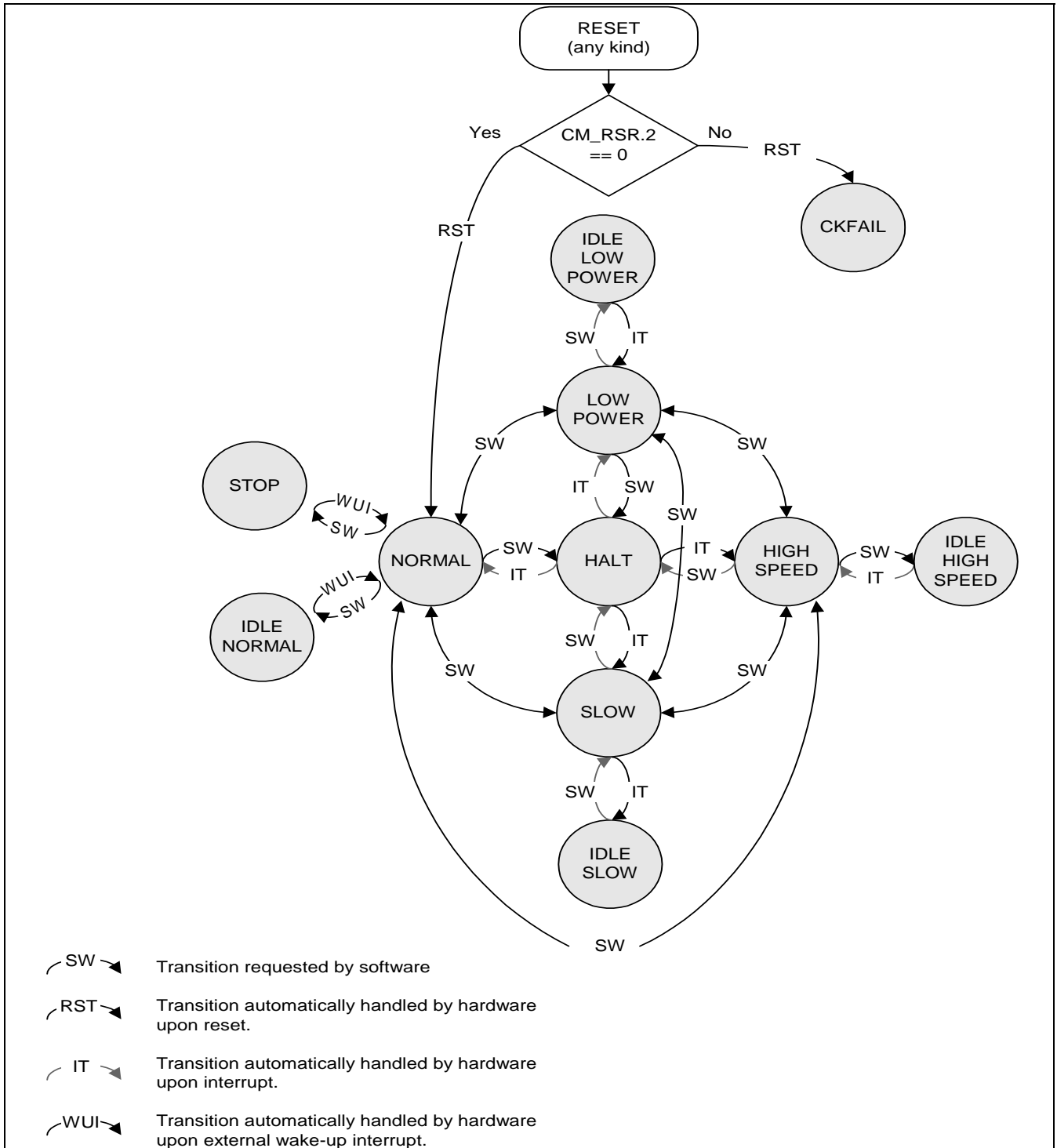


Figure 7-2. Clock Manager State Machine

As introduced previously, the clock-manager manages a master oscillator, a programmable PLL and an internal low frequency oscillator. Each of them shall be enabled or disabled according to the software programming and the current operating mode.

The effect of each of the previous operating modes is given in the following tables

Table 7-3. Oscillators and PLL Activity

Mode	LFOSCEN ^(note1)	LFSEL ^(note1)	Ring Oscillator	Master Oscillator	PLL
NORMAL, IDLE NORMAL	0	0	Disabled	Enabled	Disabled
		1	Forbidden ^(note2)		
	1	0	Enabled	Enabled	Disabled
		1	Enabled	Enabled	Disabled
SLOW, IDLE SLOW	0	0	Forbidden ^(note2)		
	1	0	Forbidden ^(note2)		
		1	Enabled	Enabled	Disabled
HALT, LOWPOWER, IDLE LOWPOWRER	0	0	Disabled	Enabled	Disabled
		1	Forbidden ^(note2)		
	1	0	Enabled	Enabled	Disabled
		1	Enabled	Disabled	Disabled
HIGHSPEED, IDLE HIGHSPEED	0	0	Disabled	Enabled	Enabled
		1	Forbidden ^(note2)		
	1	0	Enabled	Enabled	Enabled
		1	Enabled	Enabled	Enabled
STOP	0 or 1	0 or 1	Disabled	Disabled	Disabled
CKFAIL	0	Forbidden ^(note2)			
	1	0 or 1	Enabled	Disabled	Disabled

NOTES:

1. LFOSCEN and LFSEL bits are part of the CM_LFOSCR register and can be written only in the normal mode.
2. The software must avoid this situation, otherwise unpredictable effect will occur inside the device.

Also, the clock manager manages the 5 clock sources introduced above according to the following table:

The information given in this table is true outside any transition period (i.e. when the selected clock manager state is fully established). You will see in following paragraphs that some clocks may be generated differently during state transitions.

Table 7-4. Clock Transition

Mode	LFOSCEN (note1)	LFSEL (note1)	ARMCLK	SCLK	PCLK (note2)	GICCLK	LFCLK
NORMAL	0	0	MCLK/ (CDIV+1)	MCLK/ (CDIV+1)	MCLK/ C_PCLK_DIV	MCLK/ C_PCLK_DIV	DIVOUT/ (LDIV+1)
		1	Forbidden (note3)				
	1	0	MCLK/ (CDIV+1)	MCLK/ (CDIV+1)	MCLK/ C_PCLK_DIV	MCLK/ C_PCLK_DIV	DIVOUT/ (LDIV+1)
		1	MCLK/ (CDIV+1)	MCLK/ (CDIV+1)	MCLK/ C_PCLK_DIV	MCLK/ C_PCLK_DIV	RINGCLK/ (LDIV+1)
IDLE NORMAL	0	0	Off	MCLK/ (CDIV+1)	MCLK/ C_PCLK_DIV	MCLK/ C_PCLK_DIV	DIVOUT/ (LDIV+1)
		1	Forbidden (note3)				
	1	0	Off	MCLK/ (CDIV+1)	MCLK/ C_PCLK_DIV	MCLK/ C_PCLK_DIV	DIVOUT/ (LDIV+1)
		1	Off	MCLK/ (CDIV+1)	MCLK/ C_PCLK_DIV	MCLK/ C_PCLK_DIV	RINGCLK/ (LDIV+1)
HIGH SPEED	0	0	PLLOUT/ (CDIV+1)	PLLOUT/ (CDIV+1)	PLLOUT/ C_PCLK_DIV	PLLOUT/ C_PCLK_DIV	DIVOUT/ (LDIV+1)
		1	Forbidden (note3)				
	1	0	PLLOUT/ (CDIV+1)	PLLOUT/ (CDIV+1)	PLLOUT/ C_PCLK_DIV	PLLOUT/ C_PCLK_DIV	DIVOUT/ (LDIV+1)
		1	PLLOUT/ (CDIV+1)	PLLOUT/ (CDIV+1)	PLLOUT/ C_PCLK_DIV	PLLOUT/ C_PCLK_DIV	RINGCLK/ (LDIV+1)
IDLE HIGH SPEED	0	0	Off	PLLOUT/ (CDIV+1)	PLLOUT/ C_PCLK_DIV	PLLOUT/ C_PCLK_DIV	DIVOUT/ (LDIV+1)
		1	Forbidden (note3)				
	1	0	Off	PLLOUT/ (CDIV+1)	PLLOUT/ C_PCLK_DIV	PLLOUT/ C_PCLK_DIV	DIVOUT/ (LDIV+1)
		1	Off	PLLOUT/ (CDIV+1)	PLLOUT/ C_PCLK_DIV	PLLOUT/ C_PCLK_DIV	RINGCLK/ (LDIV+1)
SLOW	0	Forbidden (note3)					
	1	0	Forbidden (note3)				
		1	DIVOUT	DIVOUT	DIVOUT/ (PCLK1DIV+1)	DIVOUT/ (PCLK1DIV+1)	RINGCLK/ (LDIV+1)

Table 7-4. Clock Transition (Continued)

Mode	LFOSCN (note1)	LFSEL (note1)	ARMCLK	SCLK	PCLK (note2)	GICCLK	LFCLK
IDLE SLOW	0	Forbidden (note3)					
	1	0	Forbidden (note3)				
		1	Off	DIVOUT	DIVOUT/ (PCLK1DIV+1)	DIVOUT/ (PCLK1DIV+1)	RINGCLK/ (LDIV+1)
LOW POWER	0	0	DIVOUT	DIVOUT	DIVOUT/ (PCLK1DIV+1)	DIVOUT/ (PCLK1DIV+1)	DIVOUT/ (LDIV+1)
		1	Forbidden (note3)				
	1	0	DIVOUT	DIVOUT	DIVOUT/ (PCLK1DIV+1)	DIVOUT/ (PCLK1DIV+1)	DIVOUT/ (LDIV+1)
		1	RINGCLK	RINGCLK	RINGCLK/ (PCLK1DIV+1)	RINGCLK/ (PCLK1DIV+1)	RINGCLK/ (LDIV+1)
IDLE LOW POWER	0	0	Off	DIVOUT	DIVOUT/ (PCLK1DIV+1)	DIVOUT/ (PCLK1DIV+1)	DIVOUT/ (LDIV+1)
		1	Forbidden (note3)				
	1	0	Off	DIVOUT	DIVOUT/ (PCLK1DIV+1)	DIVOUT/ (PCLK1DIV+1)	DIVOUT/ (LDIV+1)
		1	Off	RINGCLK	RINGCLK/ (PCLK1DIV+1)	RINGCLK/ (PCLK1DIV+1)	RINGCLK/ (LDIV+1)
HALT	0	0	Off	DIVOUT	Software control (note4)	DIVOUT/ (PCLK1DIV+1)	DIVOUT/ (LDIV+1)
		1	Forbidden (note3)				
	1	0	Off	DIVOUT	Software control (note4)	DIVOUT/ (PCLK1DIV+1)	DIVOUT/ (LDIV+1)
		1	Off	RINGCLK	Software control (note4)	RINGCLK/ (PCLK1DIV+1)	RINGCLK/ (LDIV+1)
STOP	0 or 1	0 or 1	Off	Off	Off	Off	Off
CKFAIL	0	Forbidden (note3)					
	1	0 or 1	RINGCLK/ (CDIV+1)	RINGCLK/ (CDIV+1)	RINGCLK/ C_PCLK_DIV	RINGCLK/ C_PCLK_DIV	RINGCLK/ (LDIV+1)

NOTES

1. LFOSCN and LFSEL bits are part of the CM_LFOSCR register and can be written only in the normal mode.
2. PCLK and GICCLK are derived from master clock through programmable frequency divider controlled by PCLK1DIV field accessible in CM_DIVBR register.
3. The software must avoid this clock mode condition, otherwise unpredictable effect will occur inside the device.
4. In HALT mode, PCLK may be "on" or "off" depending on the value of PCLK1 in CM_WIFR register
5. $C_PCLK_DIV = (CDIV+1) \times (PCLK1DIV+1)$
6. In any mode, the divided master clock frequency (CORECLK) must always be greater than the low frequency oscillator (RINGCLK).
7. It is recommended to check stable bit when changing mode. But when user switch from HIGH SPEED mode to NORMAL mode, user must be sure the switch has finished when the SPEED mode (refer to IFC module) is disabled (SPEEDMODE bit=0 in IFC_MR register). So when changing mode, you should check 'STABLE' bit of CM_SR.

Program the following steps:

- Clear stable interrupt
- Enable High speed mode for the FLASH controller
- Go to HIGH SPEED mode
- "Wait stable rise"**
- Clear stable status
- Go to NORMAL mode
- "Wait stable rise"**
- Disable high speed mode for the FLASH controller
- Clear stable status

8. When waking-up from IDLE/HALT, user must insert NOPx4. In IDLE/HALT Mode CPU and Flash/SRAM clock is disabled. When asserted wake-up signal, CPU and Memory is released from IDLE/HALT mode. But CPU is faster than FLASH/SRAM memory. So CPU can read FLASH/SRAM being under wait-state. To protect this, user must have some-wait by using NOP.

2. FUNCTIONAL OPERATION

2.1 CLOCK MANAGEMENT

2.1.1 General Description

At power-up, the master and the internal low frequency oscillator are enabled by default. After reset, the clock manager operates in the NORMAL mode and the PLL is disabled. Only in the NORMAL mode, the user can change configuration of the low frequency oscillator with the CM_LFOSCR register.

The CMCLKSEL field in the CM_SELR register and HALTMODE, IDLEMODE, STOPMODE fields in CM_CR register allows the software to select amongst the 7 user programmable modes: NORMAL, HIGH SPEED, SLOW, LOW POWER, HALT, IDLE and STOP.

Before switching to a different mode, software has to ensure that the clock manager is not already switching from on mode to another one (i. e current mode is already stable).

The IDLEMODE in CM_CR register allows the software to disable the ARM7 clock, all other clock in the circuit stay unchanged (ARM7 will start again on interrupt). Thus 4 additional mode are defined : IDLE NORMAL, IDLE SLOW, IDLE LOW POWER and IDLE HIGH SPEED.

- In NORMAL mode, the main clocks are directly generated from the master oscillator. The PLL is disabled.
- In the HIGH SPEED mode, the PLL is automatically enabled. Before selecting this mode, the user has to configure the PLL stabilization time in the CM_PSTR register, the PLL divider parameters (PLL_PRE, PMUL and PLL_POST fields in the CM_PDPR register). The PLL_ST bit in the CM_STR register is used as flag to indicate that the PLL is enabled and stabilized.
- In the SLOW mode, the main clock is generated from the divided master clock. The MDIV division factor is configured via the CM_MDIVR register. The real division factor corresponds to the MDIV+1 value. This mode is allowed only if the low frequency oscillator is enabled and used.
- In the LOW POWER mode (and in the HALT mode), the main clocks are generated from the low frequency oscillator or from the master oscillator divided by MDIV+1. When low frequency oscillator is enabled, the master oscillator is automatically shut off. The user must not forget to configure the master oscillator stabilization time into the CM_OSTR register before the clock manager will switch to the SLOW, NORMAL or HIGH SPEED mode from LOW POWER mode.
- In the HALT mode, all clocks are disabled except the GIC clock, the SCLK and the PCLK. But, the user can stop the PCLK with CM_WIFR in the HALT mode. Interrupt will switch operating mode from HALT mode to the previous mode in CM_SELR register.
- In the STOP mode, all clocks are disabled. Only an external interrupt will allow to switch to the normal mode from STOP mode. The STOP mode must be entered only from Normal mode, otherwise unexpected effect may occur in the device.

If enabled in CM_MR register, the embedded clock monitor will continuously check the master oscillator operation. Upon master oscillator failure, the clock manager will force the circuit to reset and re-start in CKFAIL mode. The software can be informed of the oscillator failure by reading the CM bit in CM_RSR register.

2.1.2 Modes Transition Description

Hardware Reset to NORMAL mode

- Initial State :
 - Hardware reset sources : nRESET, LVD and Watchdog timer
- Transition trigger :
 - Transition is initialized whenever reset condition disappears
- Transition notes :
 - In case of power-up reset or if the master oscillator was off, the circuit shall wait for oscillator stabilization time, counting 65535 periods of master oscillator clock.
- Final State :
 - After reset transition, the circuit starts in NORMAL mode using the master oscillator as clock source for all internal clocks, and the divider factor PCLK1DIV and CDIV parameters are at their reset values. LFCLK is DIVOUT/(LDIV+1) clock.

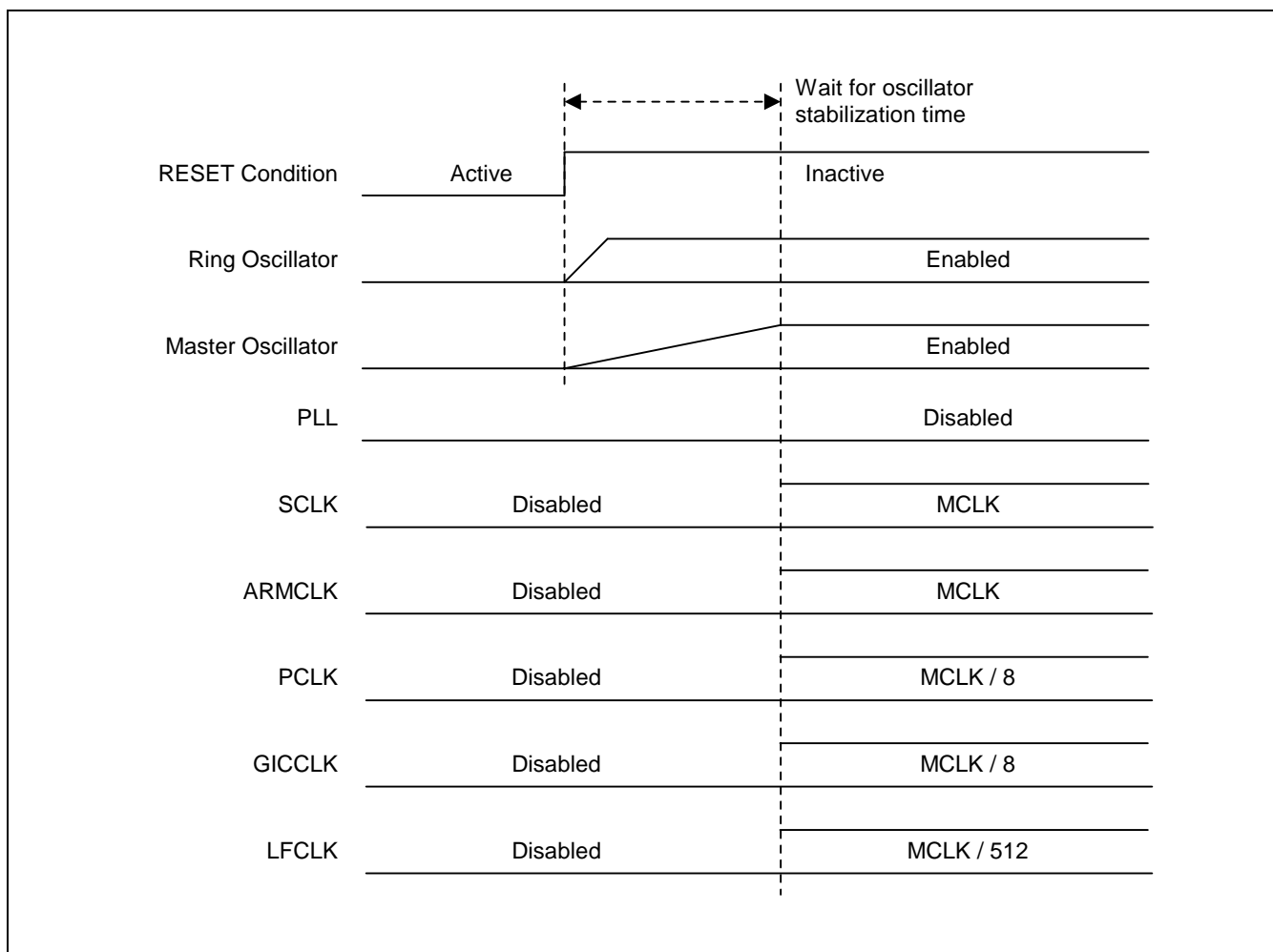


Figure 7-3. Transition Diagram: Hardware Reset to NORMAL Mode

Clock monitor reset to CKFAIL mode

- Initial State :
 - The circuit may be in any state before this transition.
 - The Clock monitor feature must be activated. (CM_EN must be set to 1 in CM_MR register)
 - The internal low frequency oscillator must be enabled. (LFOSCEN must be set to 1 in CM_LFOSCR register)
- Transition trigger :
 - The transition occurs whenever clock monitor detects a malfunction of master clock.
- Final State :
 - After clock monitor reset, the circuit enters the CKFAIL mode, which is functionally equivalent to NORMAL mode except that all clocks are derived from internal low frequency oscillator.
- Important note :
 - User should not try to change to any other modes by writing the CMCLK_SEL field in the CKFAIL mode. Otherwise, unpredictable effects may occur inside the device.

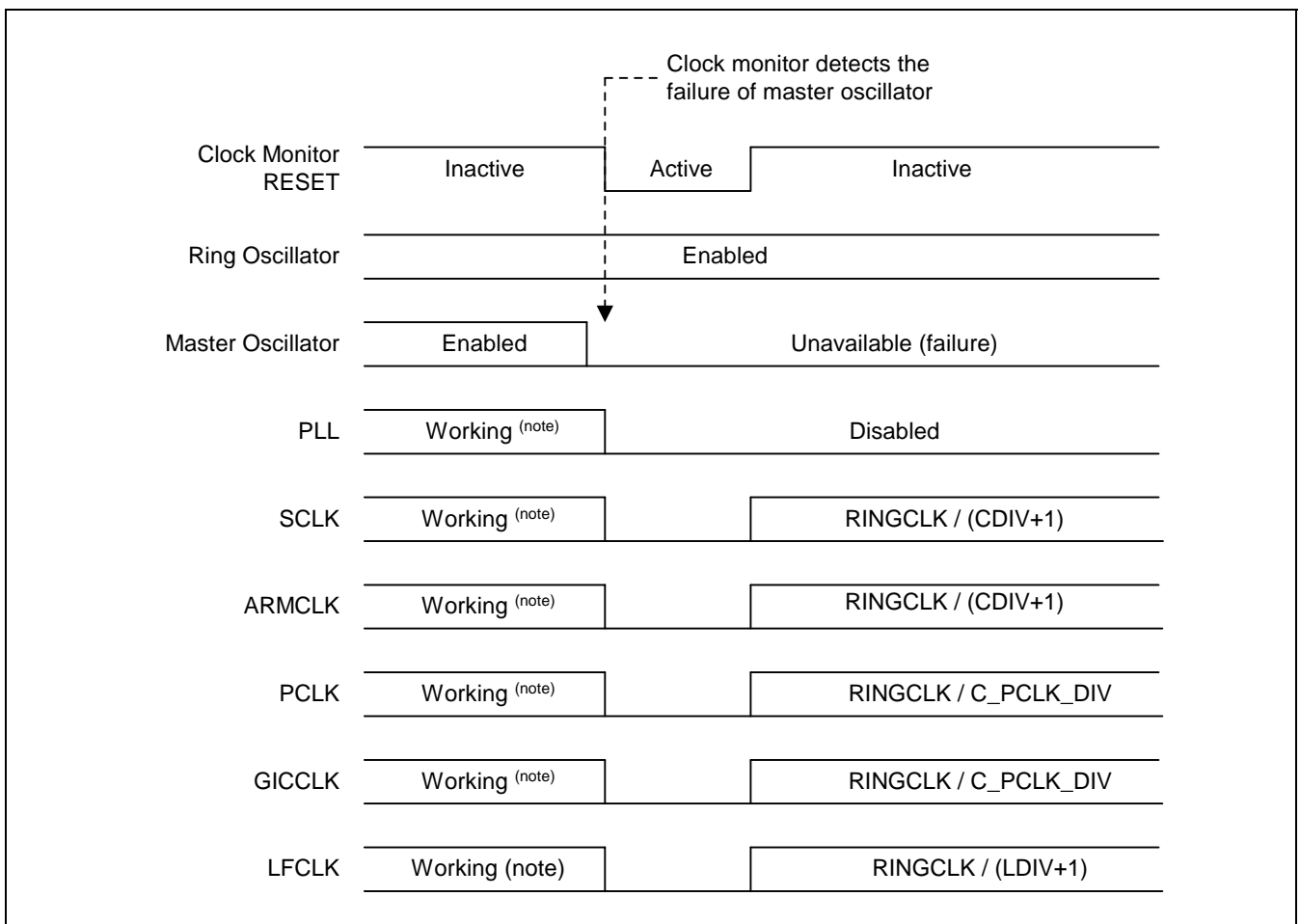


Figure 7-4. Transition Diagram: Reset to CKFAIL mode

NOTE: Clock status before clock monitor reset depends on the state of the device.

NORMAL mode to HIGHSPEED mode

- Initial State :
 - It is assumed that the circuit is in NORMAL mode before transition.
 - The software must have previously written the appropriate values in CM_PDPR register (PLL divider parameters) and in CM_PSTR register (PLL stabilization time). This is very important because those registers do not provide valid initial PLL parameters after reset.
- Transition trigger :
 - The transition is triggered upon software control when HIGHSPEED code is written in CM_SELR register.
- Final State :
 - After automatic transition, the circuit resumes in HIGHSPEED mode, where the system clock is derived from PLL output.
 - The status of LFCLK depends on LFSEL field as described below.
 - The STABLE interrupt occurs (if programmed)

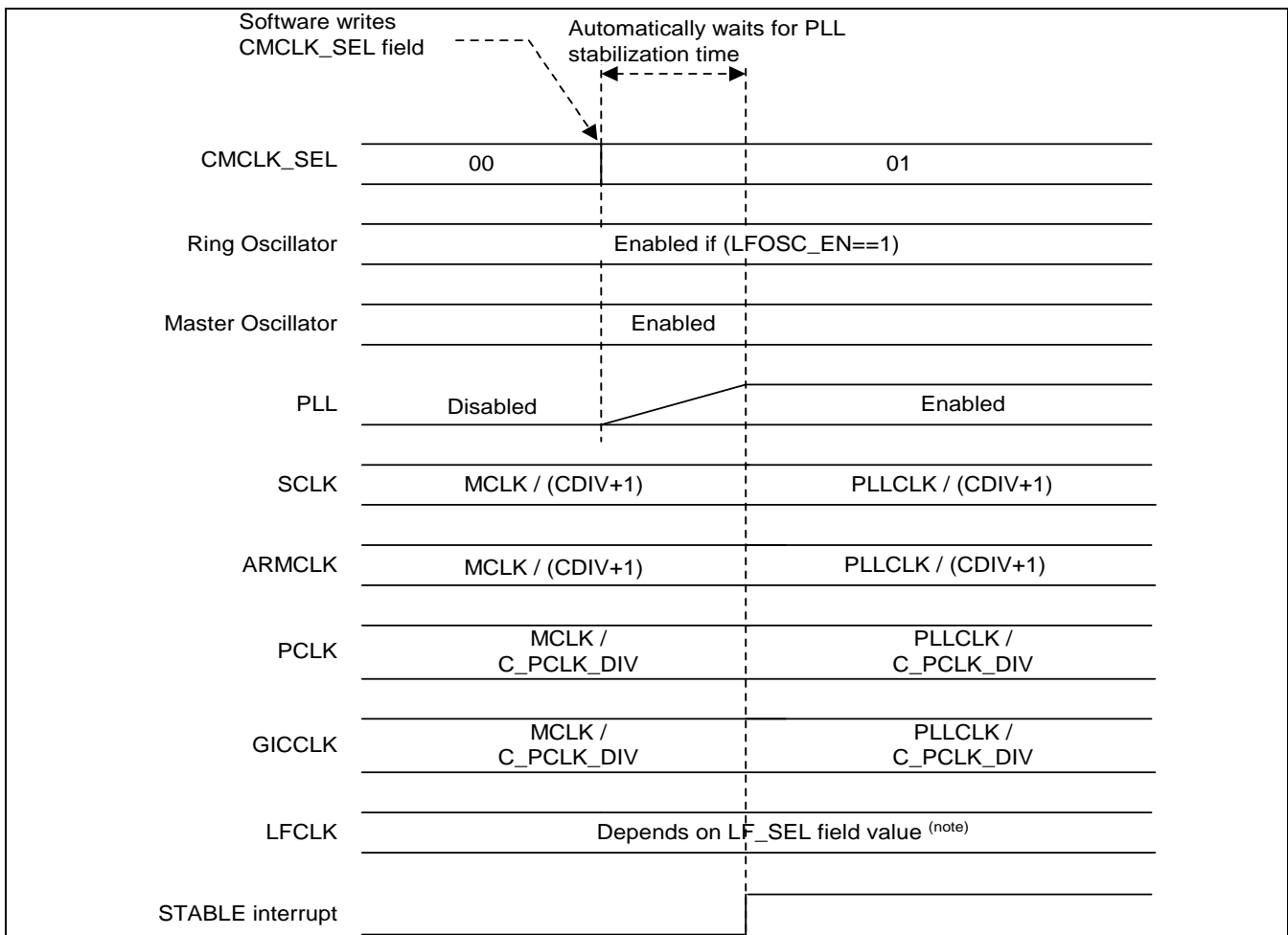


Figure 7-5. Normal Mode to High Speed Mode

NOTE: If (LS_SEL == 0) then LFCLK is : $MCLK / ((MDIV+1) \times (LDIV+1))$
 Else : LFCLK is : $RINGCLK / (LDIV+1)$

NORMAL mode to IDLE-NORMAL mode

- Initial State :
 - It is assumed that the circuit is in NORMAL mode before transition.
- Transition trigger :
 - The transition is triggered upon software control when IDLEMODE bit is set in CM_CR register.
- Final State :
 - After automatic transition, the circuit resumes in IDLE-NORMAL mode, which is the same as NORMAL mode except that ARM7 clock is disabled.

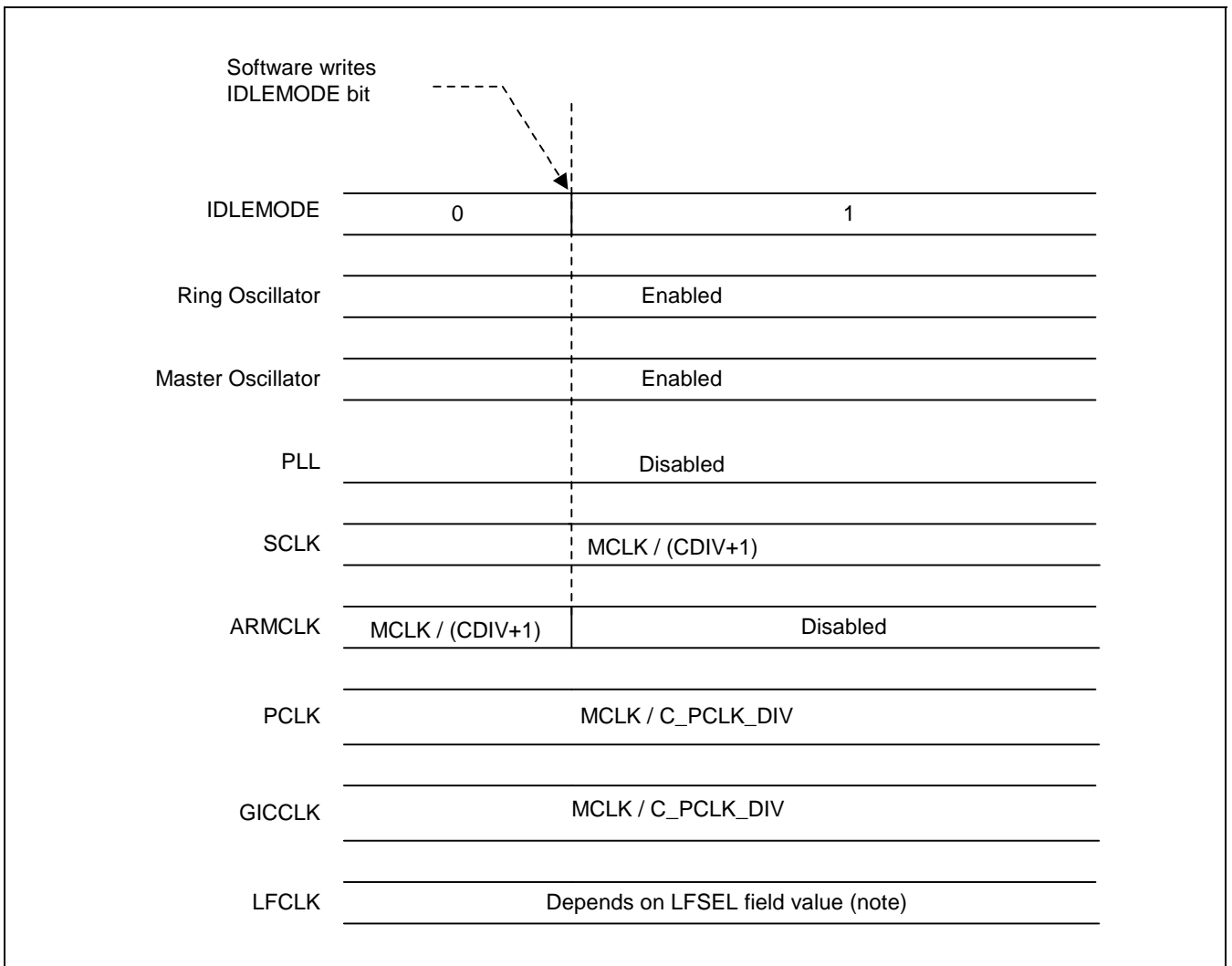


Figure 7-6. Normal Mode to IDLE-Normal Speed Mode

NOTE: If (LS_SEL == 0) then LFCLK is : $MCLK / ((MDIV+1) \times (LDIV+1))$
 Else : LFCLK is : $RINGCLK / (LDIV+1)$

IDLE-NORMAL mode to NORMAL mode

- Initial State :
– It is assumed that the circuit is in IDLE-NORMAL mode before transition.
- Transition trigger :
– The transition is triggered upon any valid interrupt from GIC (interrupt controller)
- Final State :
– After automatic transition, the circuit resumes in NORMAL mode in the same state it was before entering the IDLE-NORMAL mode.

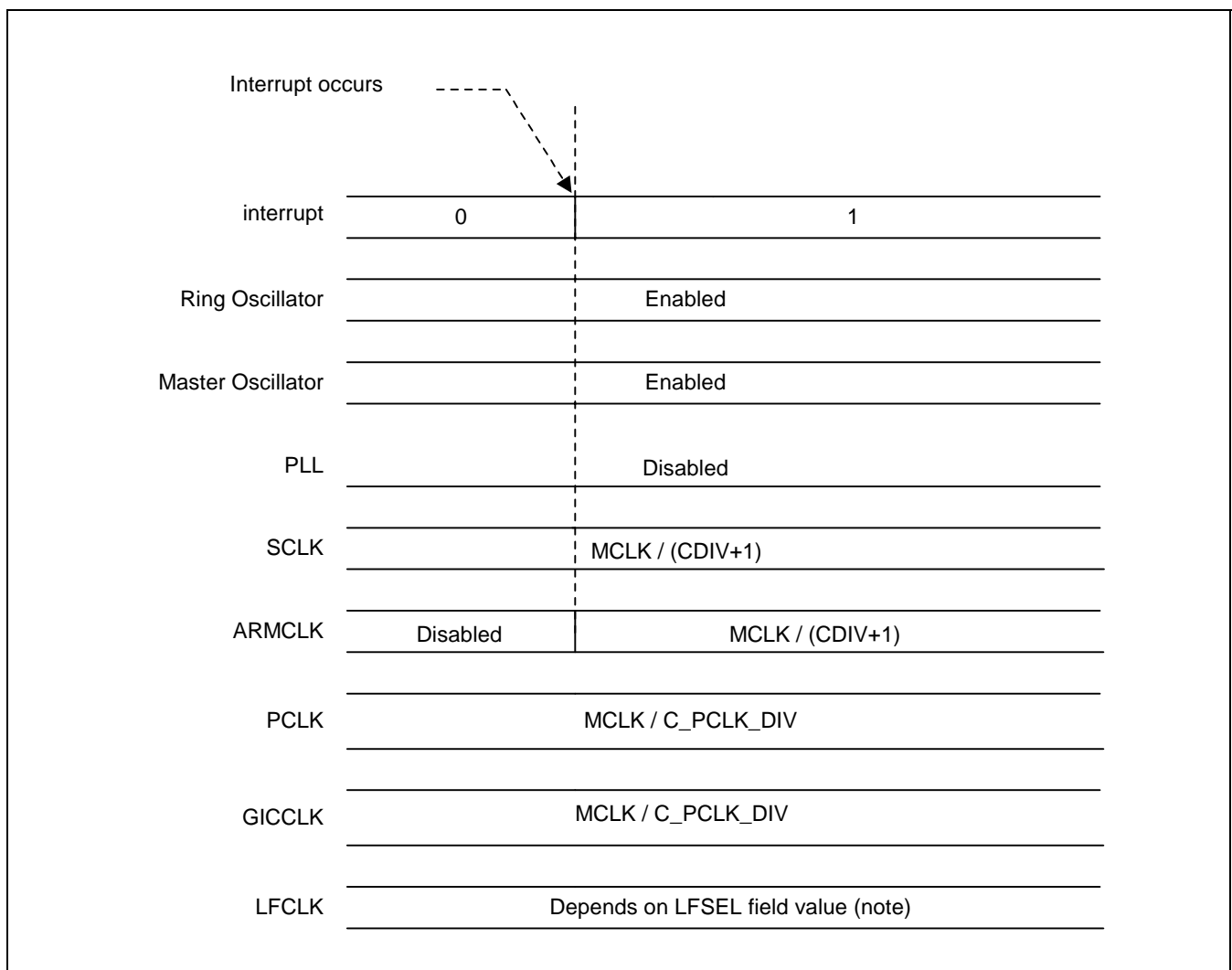


Figure 7-7. IDLE-Normal Mode to Normal Speed Mode

NOTE: If (LS_SEL == 0) then LFCLK is : $MCLK / ((MDIV+1) \times (LDIV+1))$
 Else : LFCLK is : $RINGCLK / (LDIV+1)$

NORMAL mode to SLOW mode

- Initial State :
 - It is assumed that the circuit is in NORMAL mode before transition.
 - The software must have selected the low-frequency from ring-oscillator (LFOSCEN is set to 1, and LFSEL is set to 1). All other values are forbidden for the SLOW mode selection.
- Transition trigger :
 - The transition is triggered upon software control when SLOW mode code is written in CM_SEL register.
- Final State :
 - After automatic transition, the circuit resumes in SLOW mode, where the system clock is derived from master oscillator through frequency divider.
 - LFCLK clock is RINGCLK / (LDIV+1).
 - STABLE interrupt occurs.(if programmed)

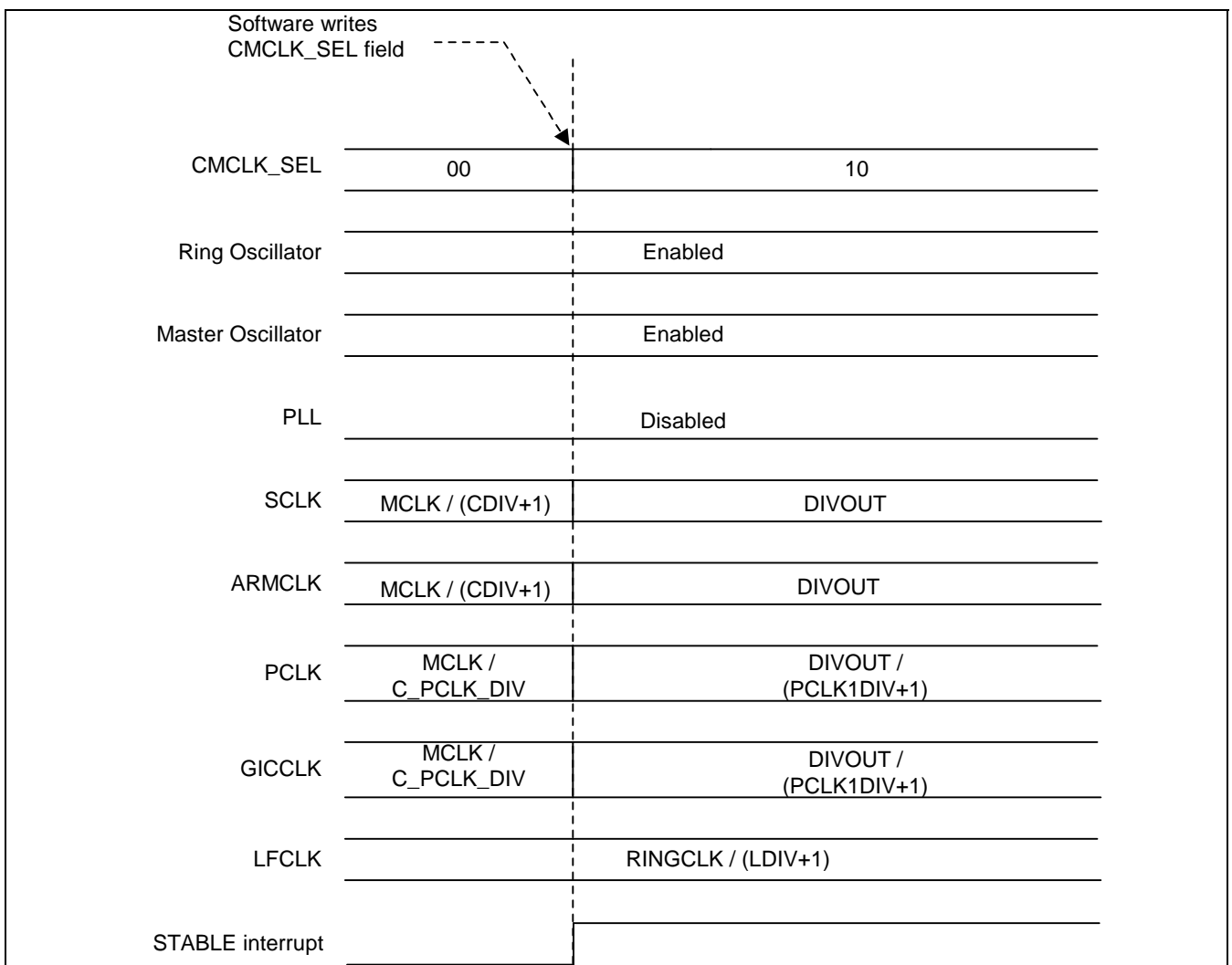


Figure 7-8. Transition Diagram: NORMAL Mode to SLOW mode

NORMAL mode to LOWPOWER mode

- Initial State :
 - It is assumed that the circuit is in NORMAL mode before transition.
 - The software must have previously written the appropriate value in CM_OSTR register (master oscillator stabilization time)
 - The software must have previously selected the Low-Frequency clock strategy in CM_LFOSCR register. Note that this register must be written only in NORMAL mode.
- Transition trigger :
 - The transition is triggered upon software control when LOWPOWER mode code is written in CM_SEL register.
- Final State :
 - After automatic transition, the circuit resumes in LOWPOWER mode, where the system clock is derived from LFCLK clock.
 - LFCLK clock status depends on the LFSEL field value
 - STABLE interrupt occurs.(if programmed)

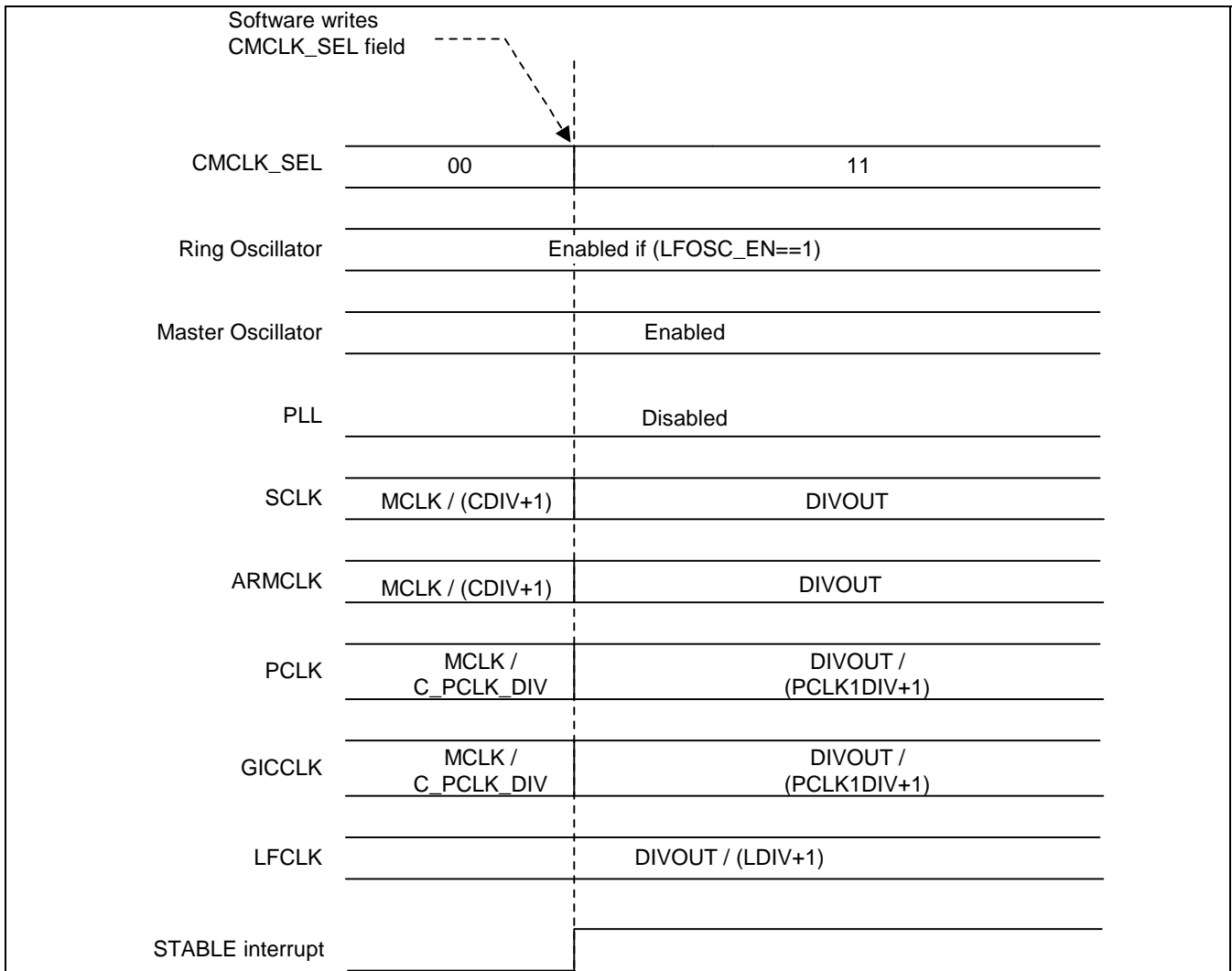


Figure 7-9. Normal Mode to Low Power Mode (LF_SEL = = 0)

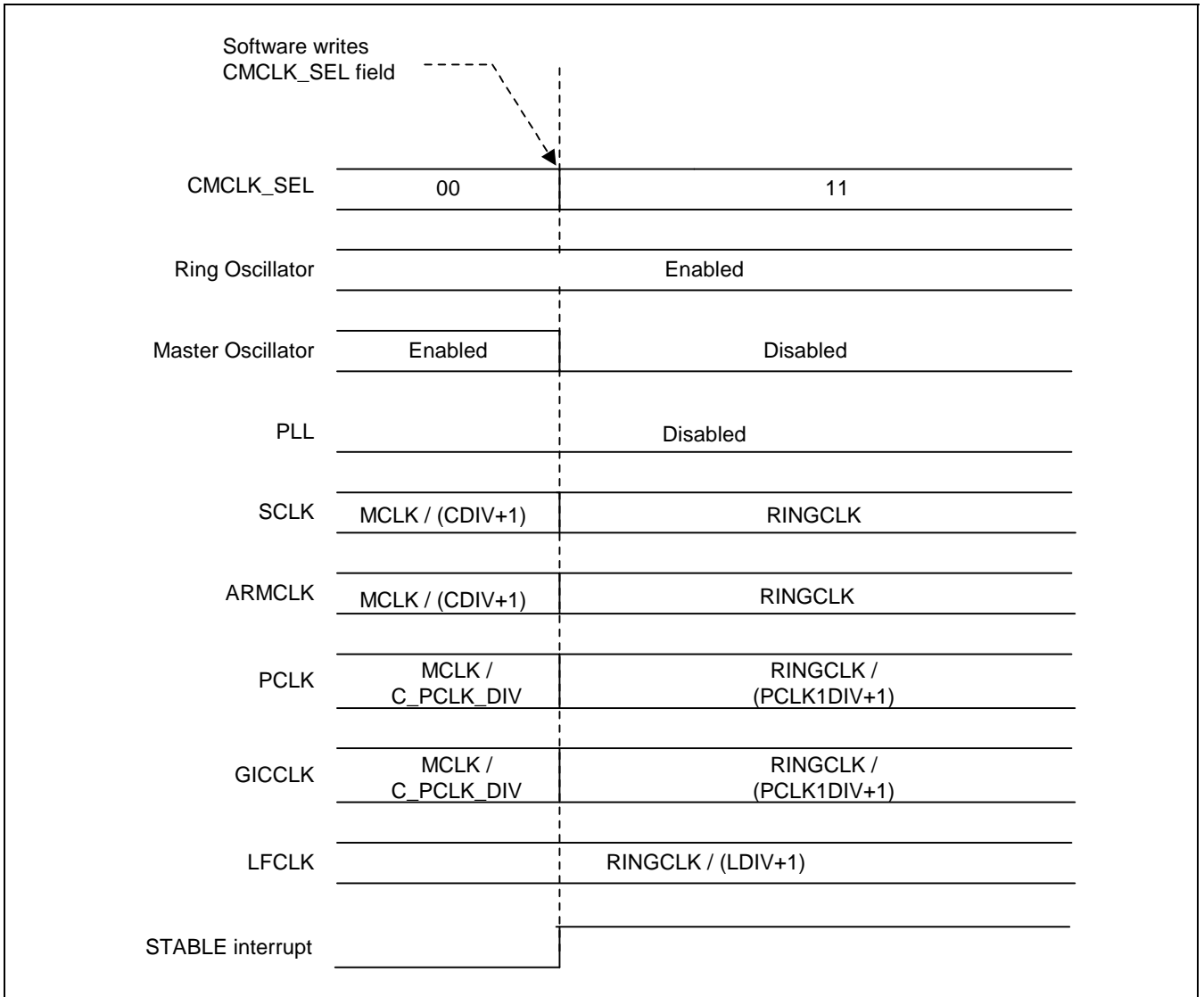


Figure 7-10. Transition Diagram: NORMAL Mode to LOWPOWER Mode (LFSEL=1)

NORMAL mode to HALT mode

- Initial State :
 - It is assumed that the circuit is in NORMAL mode before transition.
 - The software must have previously selected the Low-Frequency clock strategy in CM_LFOSCR register. Note that this register must be written only in NORMAL mode.
 - The software must have previously written the appropriate value in CM_OSTR register (master oscillator stabilization time)
- Transition trigger :
 - The transition is triggered upon software control when HALTMODE is set in CM_CR register.
- Final State :
 - After automatic transition, the circuit resumes in HALT mode, where the ARM clock is disabled, the SCLK clock is derived from LFCLK, the GIC clock is derived from LFCLK and others system clocks activity depends on CM_WFIR value.
 - LFCLK clock status depends on the LFSEL field value.
 - STABLE interrupt occurs.(if programmed)

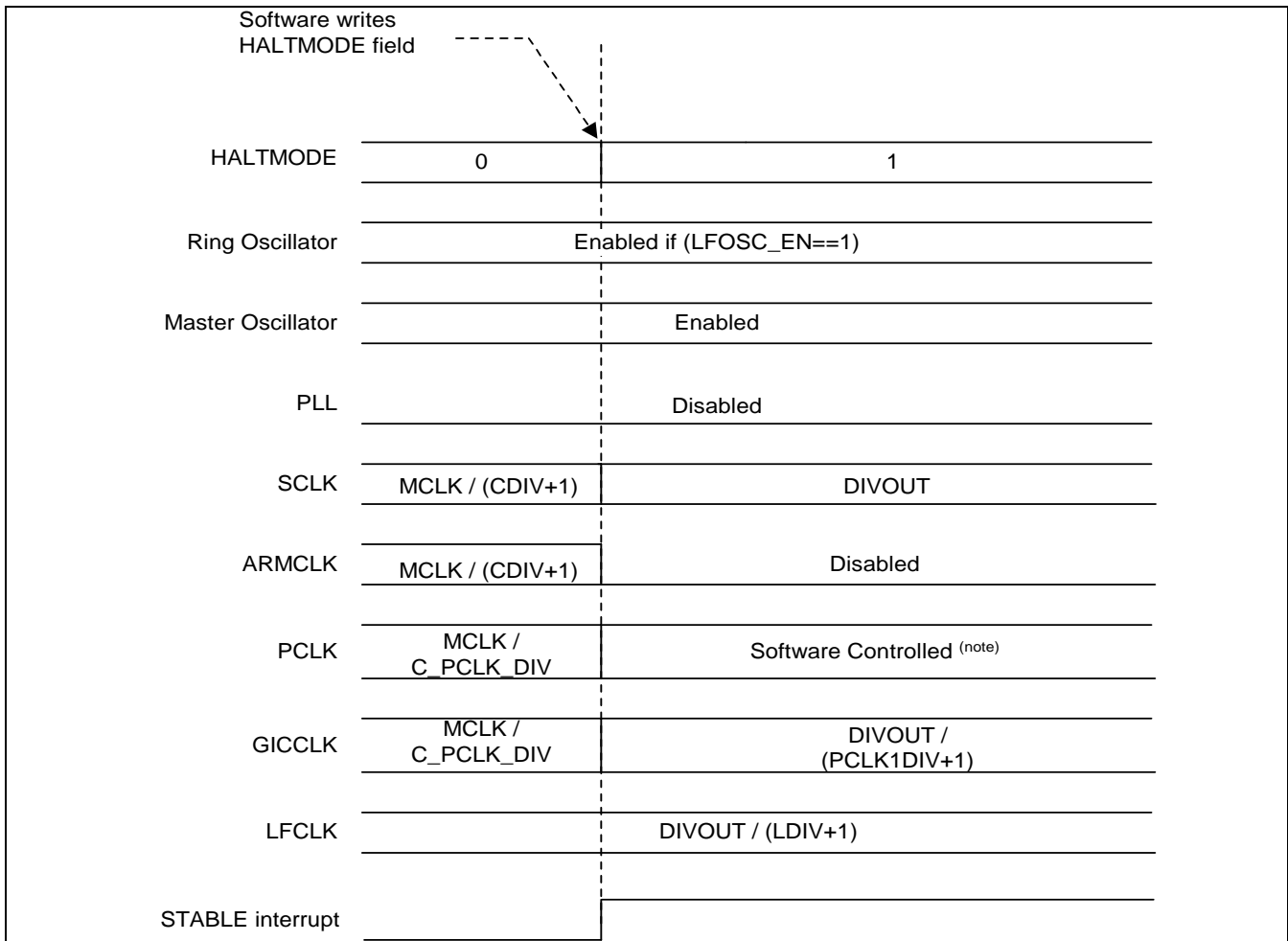


Figure 7-11. Transition Diagram: NORMAL Mode to HALT Mode (LFSEL=0)

NOTE: PCLK activity depends on the CM_WIFR register value : either (DIVOUT / (PCLK1DIV+1)) or disabled

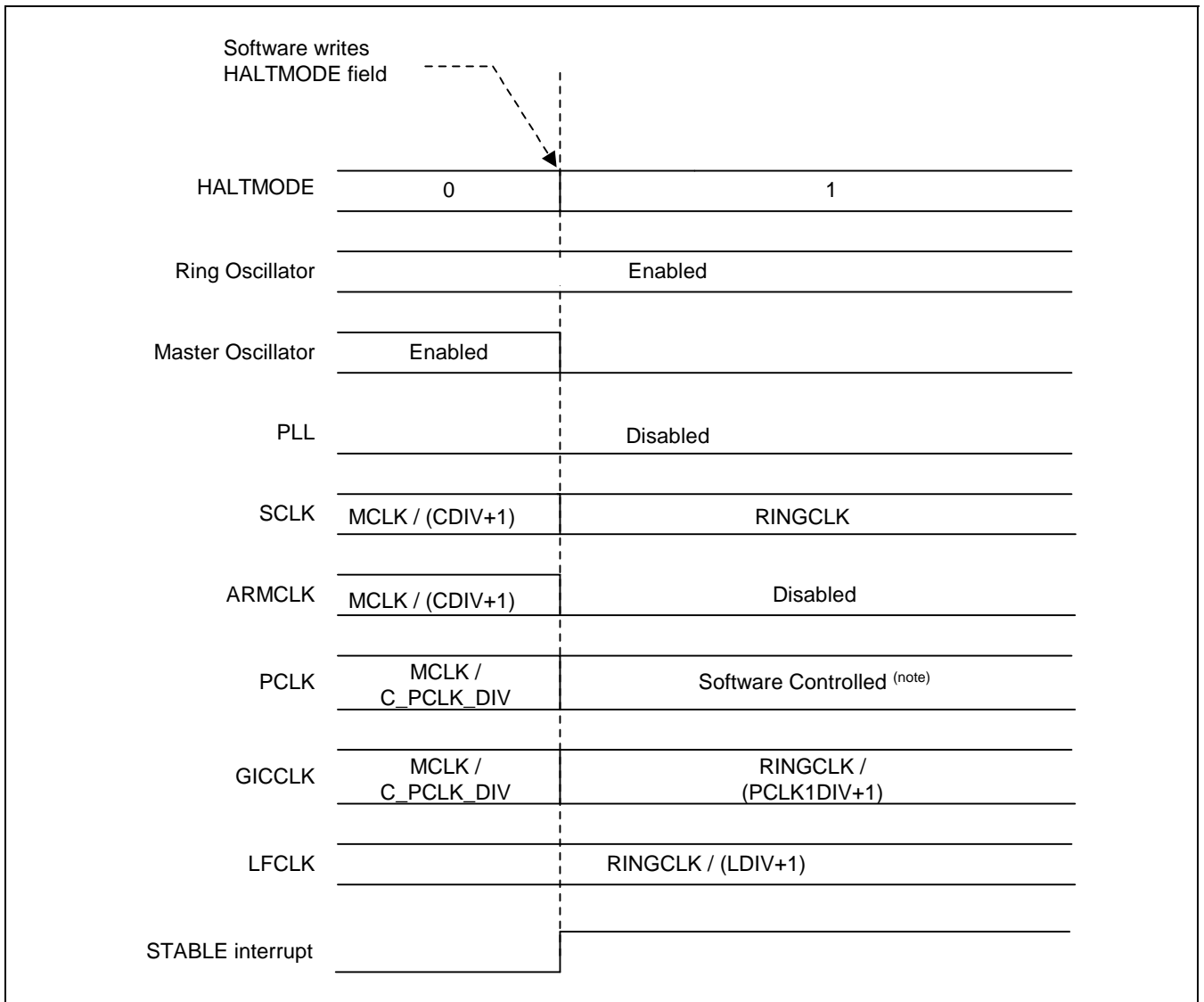


Figure 7-12. Transition Diagram: NORMAL Mode to HALT Mode (LFSEL=1)

NOTE: PCLK activity depends on the CM_WIFR register value : either (RINGCLK / (PCLK1DIV+1)) or disabled

NORMAL mode to STOP mode

- Initial State :
 - It is assumed that the circuit is in NORMAL mode before transition.
 - The software must have previously written the appropriate value in CM_OSTR register (master oscillator stabilization time)
 - The software must have previously selected the Low-Frequency clock strategy in CM_LFOSCR register. Note that this register must be written only in NORMAL mode.
- Transition trigger :
 - The transition is triggered upon software control when STOMODE is set in CM_CR register.
- Final State :
 - After automatic transition, the chip is switched to STOP mode, where all oscillators are disabled. The circuit shall only wake-up on external wake-up interrupts. (Refer to P14-46)

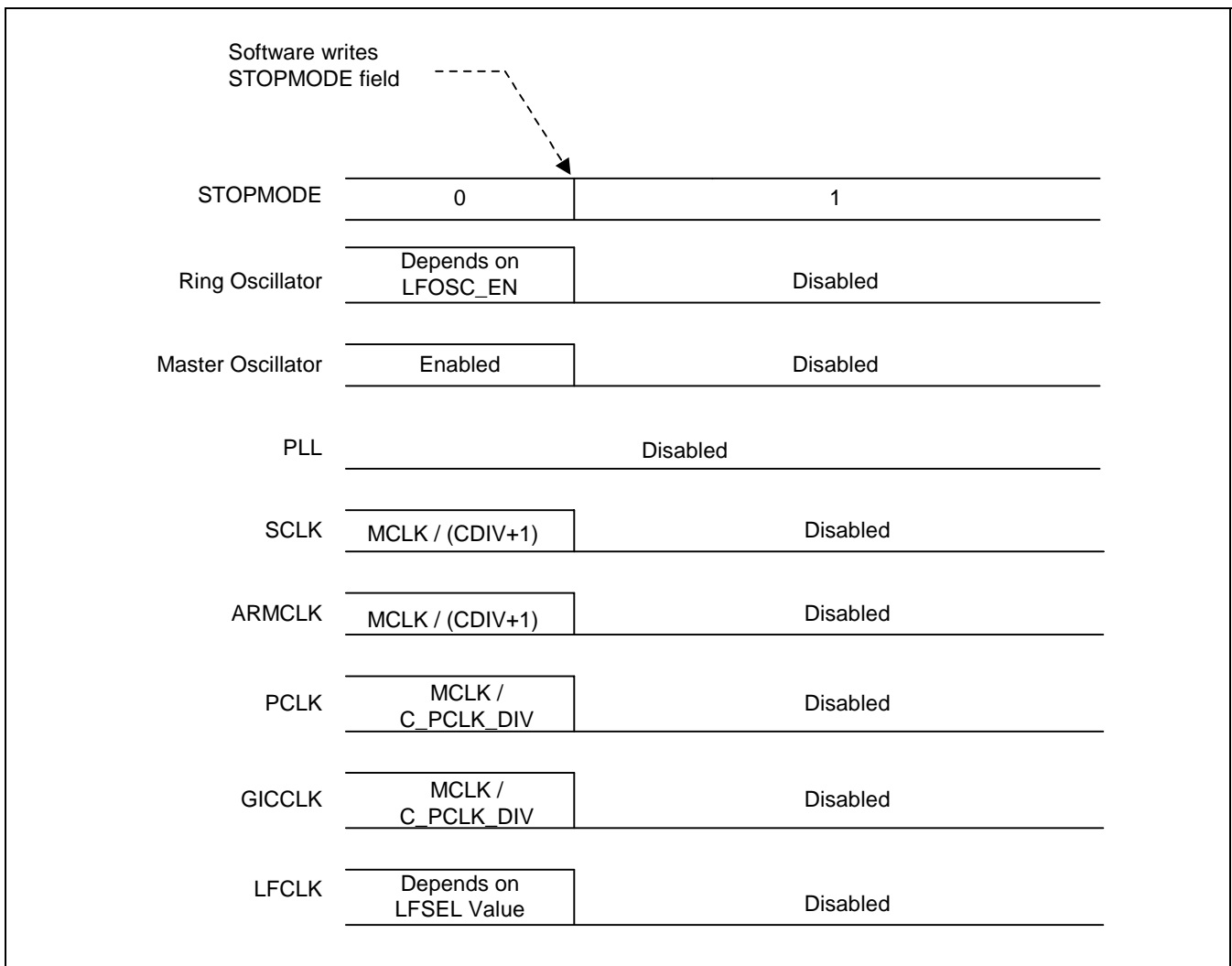


Figure 7-13. Transition Diagram: NORMAL Mode to STOP Mode

LOWPOWER mode to IDLE-LOWPOWER mode

- Initial State :
 - It is assumed that the circuit is in LOWPOWER mode before transition.
- Transition trigger :
 - The transition is triggered upon software control when IDLEMODE is set in CM_CR register
- Final State :
 - After automatic transition, the circuit resumes to IDLE-LOWPOWER mode which is the same as LOWPOWER mode except that ARM7 clock is disabled.

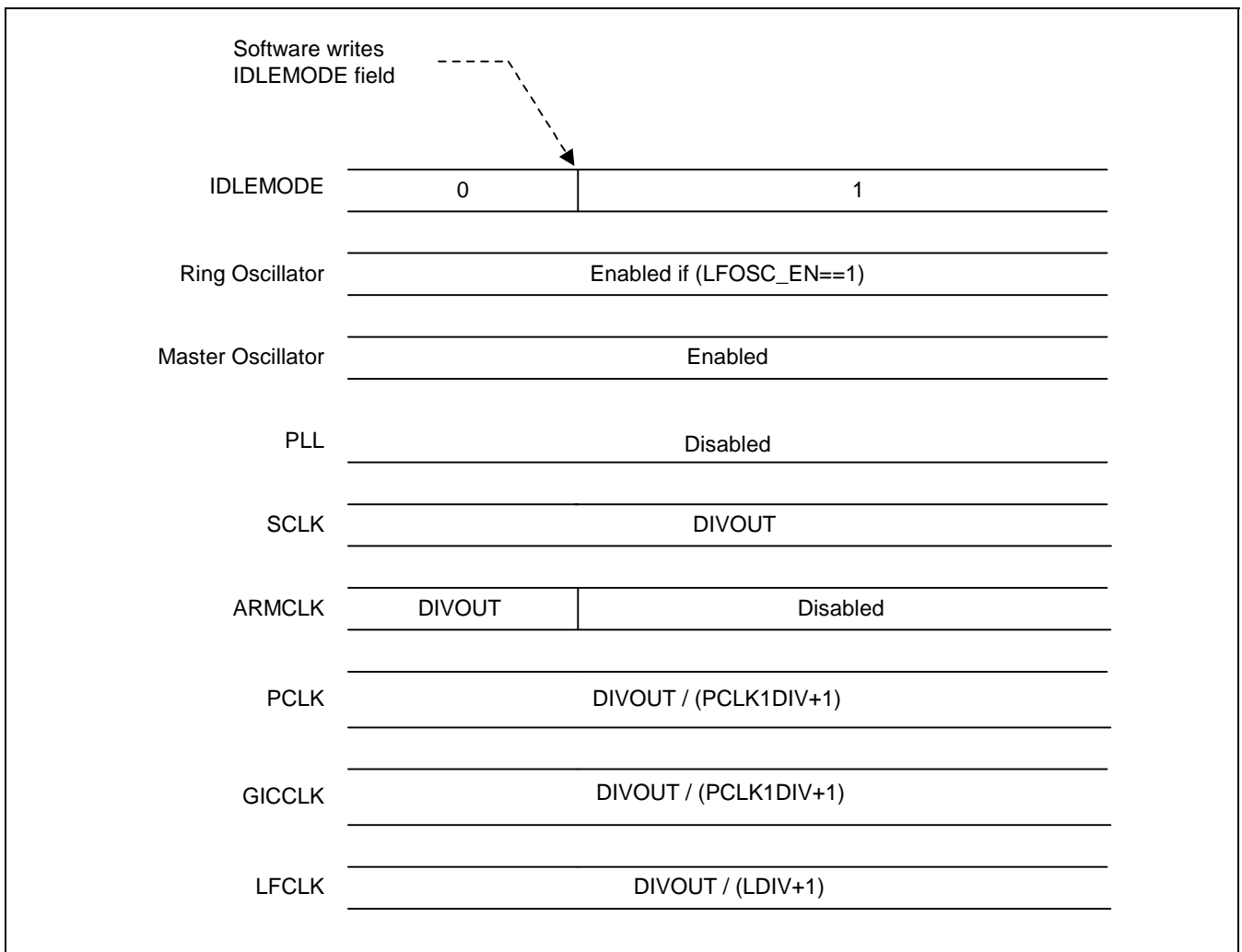


Figure 7-14. Transition Diagram: LOWPOWER Mode to IDLE-LOWPOWER Mode (LFSEL=0)

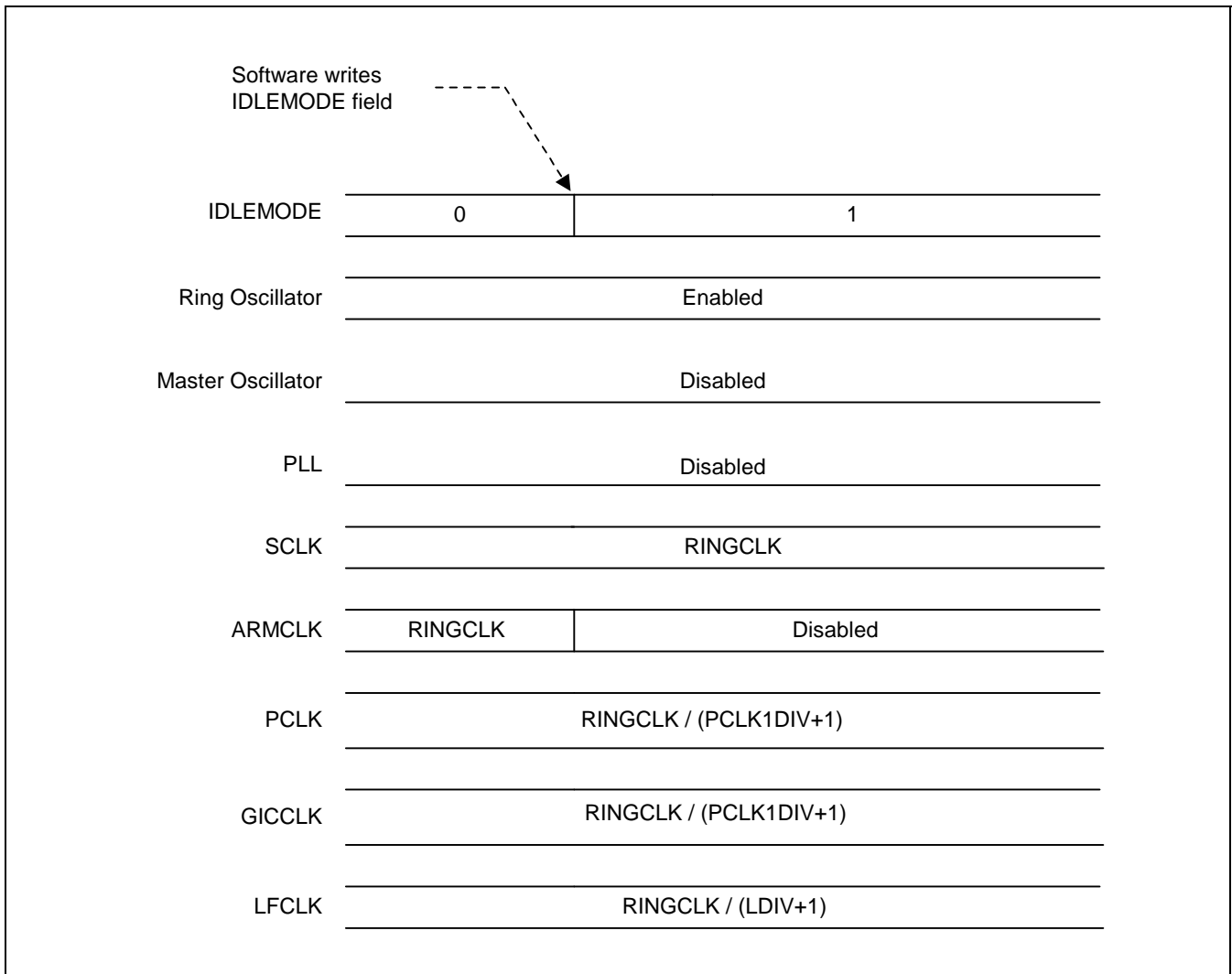


Figure 7-15. Transition Diagram: LOWPOWER Mode to IDLE-LOWPOWER Mode (LFSEL=1)

IDLE-LOWPOWER to LOWPOWER mode

- Initial State :
 - It is assumed that the circuit is in IDLE-LOWPOWER mode before transition.
- Transition trigger :
 - The transition is triggered upon any valid interrupt from GIC (interrupt controller)
- Final State :
 - After automatic transition, the circuit resumes to LOWPOWER mode in the same state it was before entering the IDLE-LOWPOWER mode.

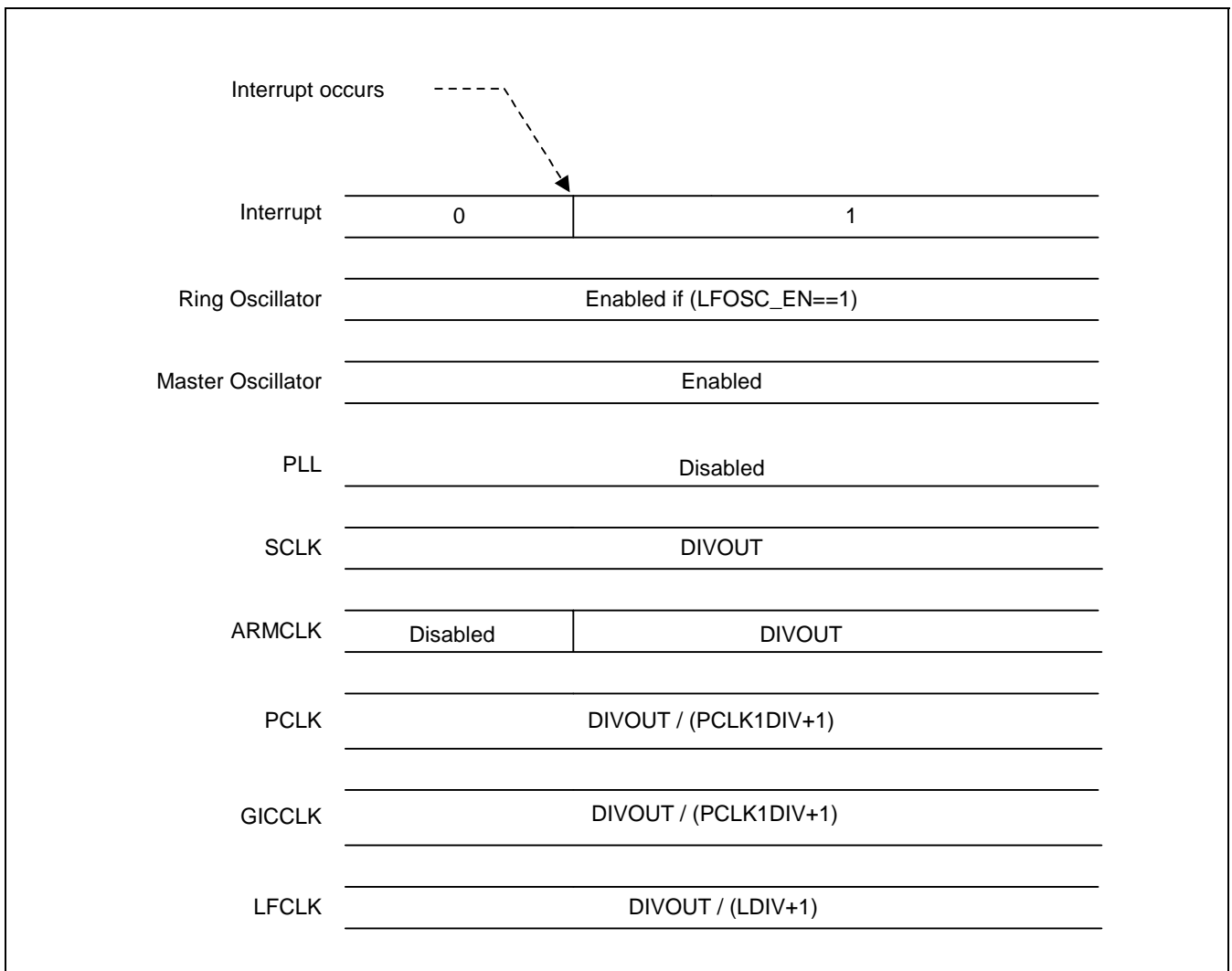


Figure 7-16. Transition Diagram: IDLE-LOWPOWER Mode to LOWPOWER Mode (LFSEL=0)

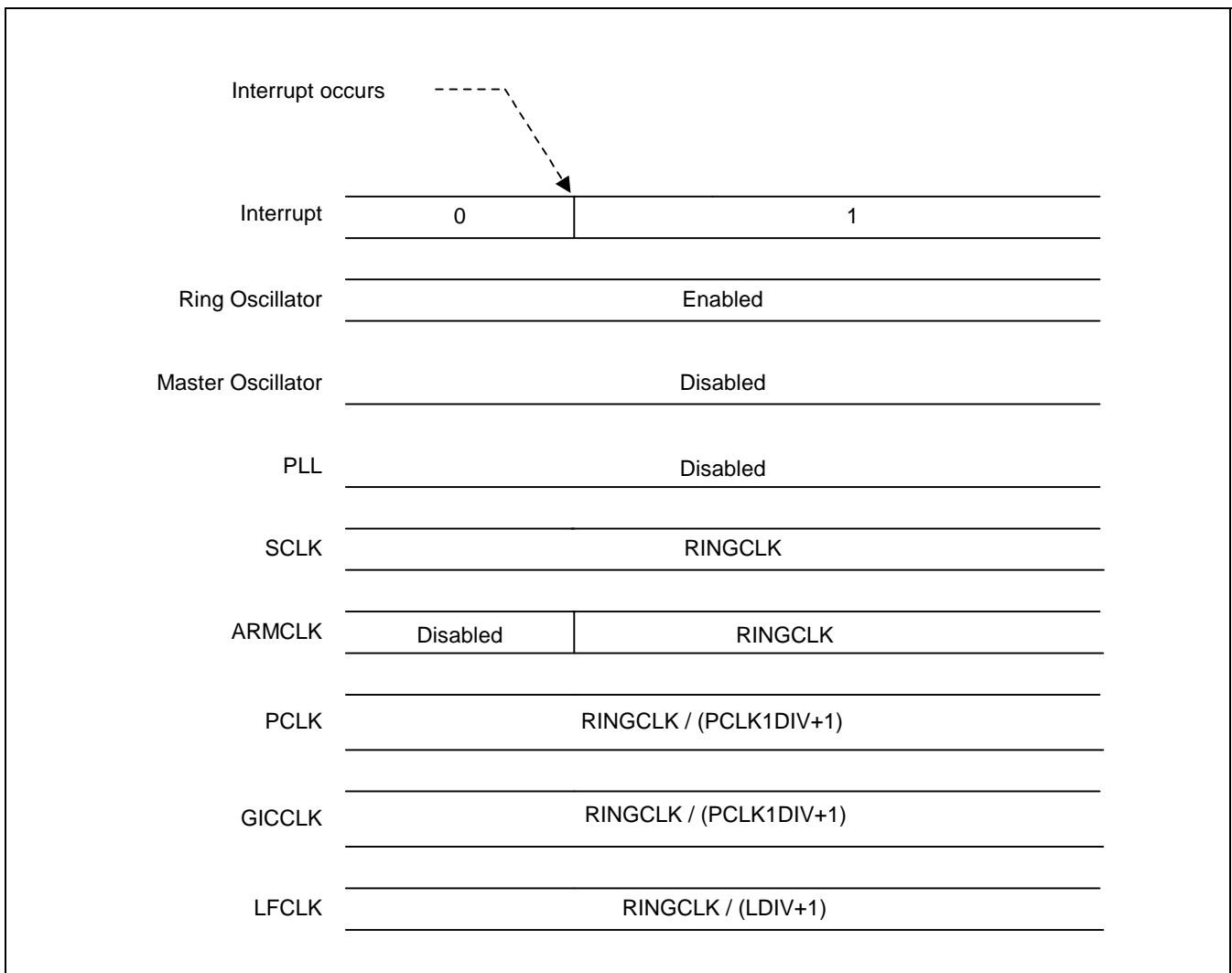


Figure 7-17. Transition Diagram: IDLE-LOWPOWER Mode to LOWPOWER Mode (LFSEL=1)

LOWPOWER mode to NORMAL mode

- Initial State :
 - It is assumed that the circuit is in LOWPOWER mode before transition.
- Transition trigger :
 - The transition is triggered upon software control when NORMAL mode code is written in CM_SEL register
- Final State :
 - After automatic transition, the circuit resumes to NORMAL mode where all system clocks are derived from master oscillator
 - LFCLK clock status depends on the LFSEL value.
 - STABLE interrupt occurs.(if programmed)

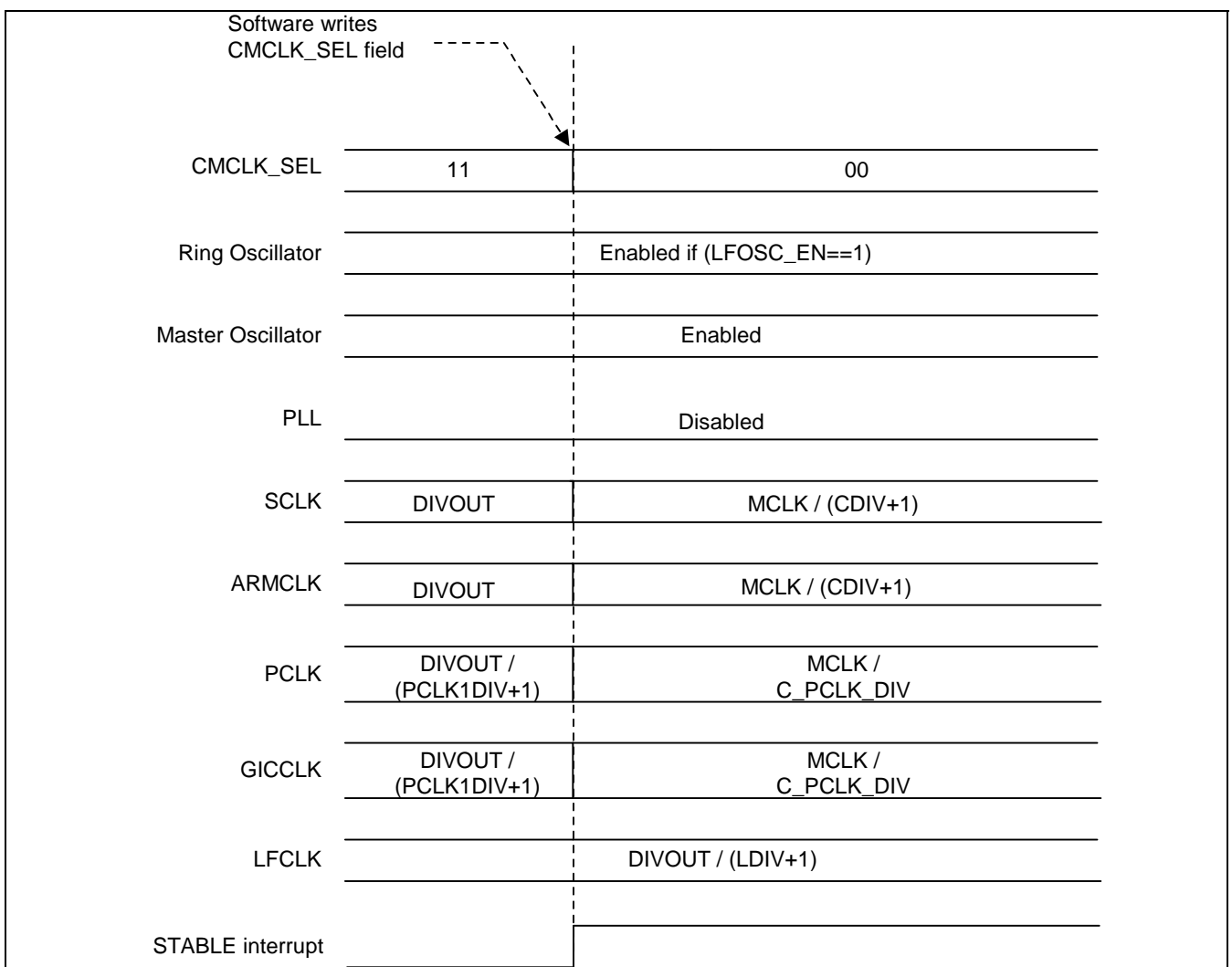


Figure 7-18. Transition Diagram: LOWPOWER Mode to NORMAL Mode (LFSEL=0)

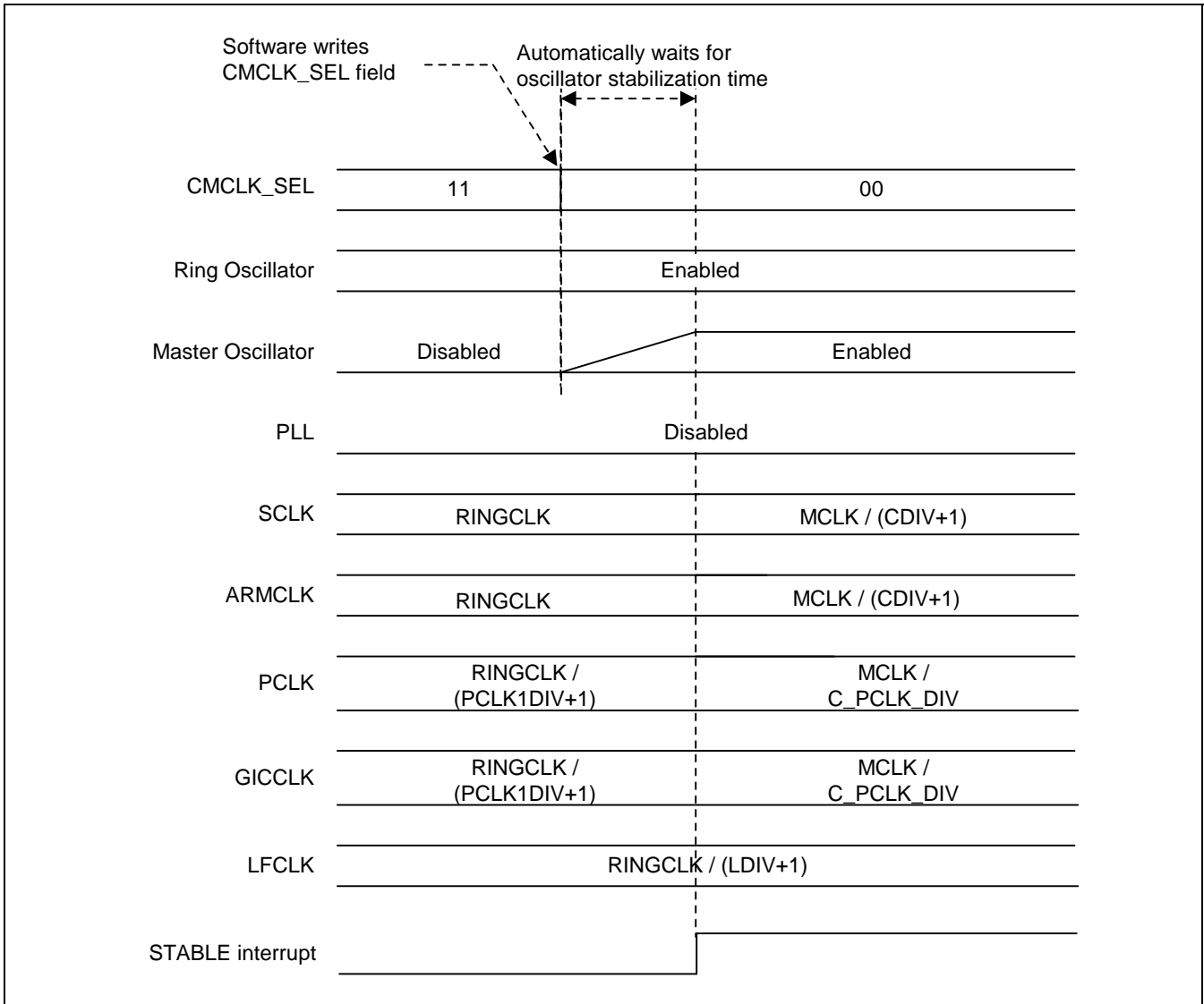


Figure 7-19. Transition Diagram: LOWPOWER Mode to NORMAL Mode (LFSEL=1)

LOWPOWER mode to HIGHSPEED mode

- Initial State :
 - It is assumed that the circuit is in LOWPOWER mode before transition.
 - The software must have previously written the appropriate values in CM_PDPR register (PLL dividers parameters), CM_PSTR register (PLL stabilization time) and CM_OSTR (master oscillator stabilization time). This is very important because those registers do not provide valid initial PLL parameters after reset.
- Transition trigger :
 - The transition is triggered upon software control when HIGHSPEED code is written in CM_SELR register.
- Final State :
 - After automatic transition, the circuit resumes in HIGHSPEED mode, where the system clock is derived from PLL output.
 - The status of LFCLK depends on LFSEL field as described bellow.
 - STABLE interrupt occurs.(if programmed)
- Important note :
 - During transition, the previous clock is maintained until the master oscillator (if LFSEL=1) and the PLL are stabilized. Then the high-speed clock is provided to the circuit and the STABLE interrupt occurs.

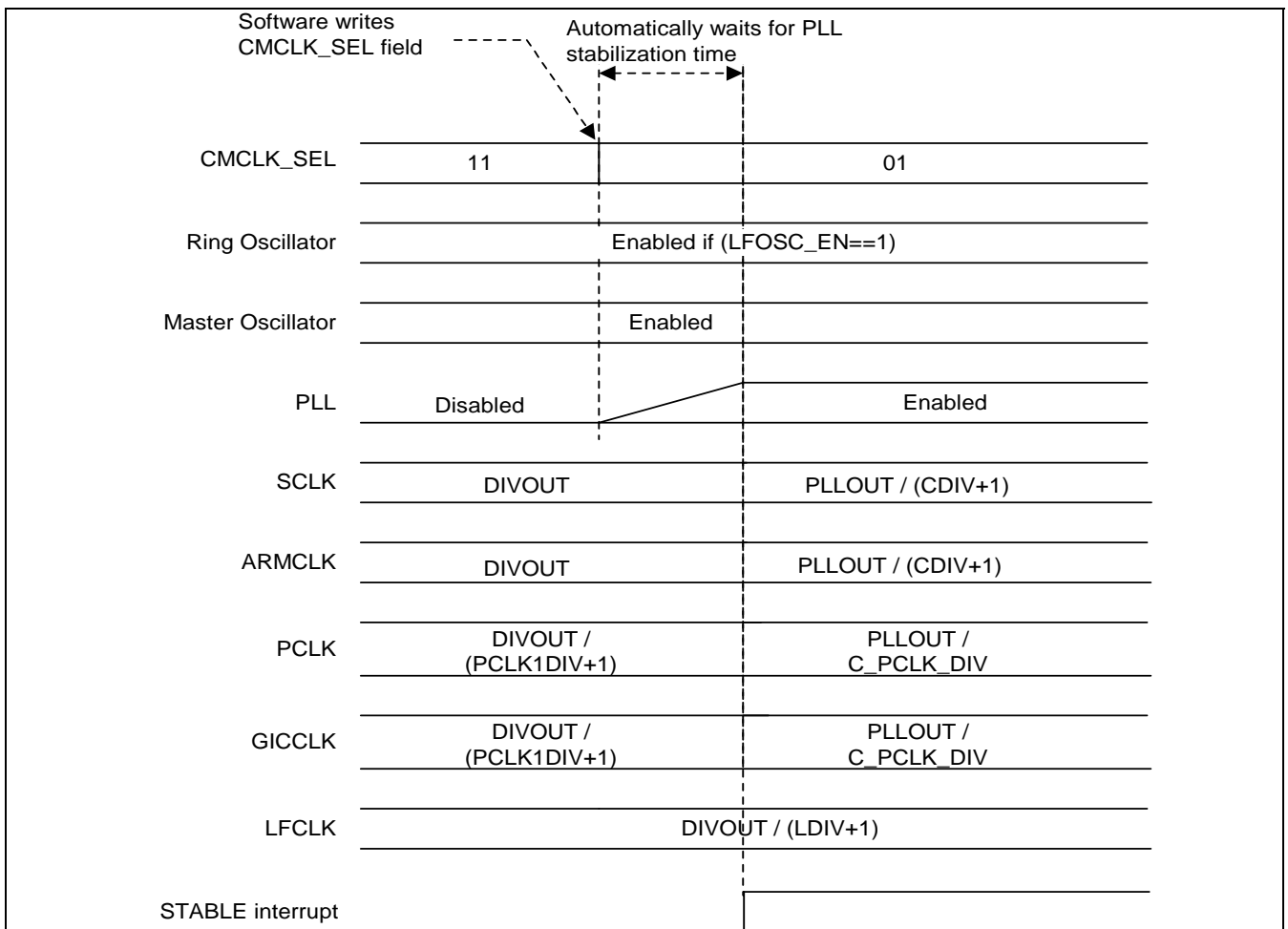


Figure 7-20. Transition Diagram: LOWPOWER Mode to HIGHSPEED Mode (LFSEL=0)

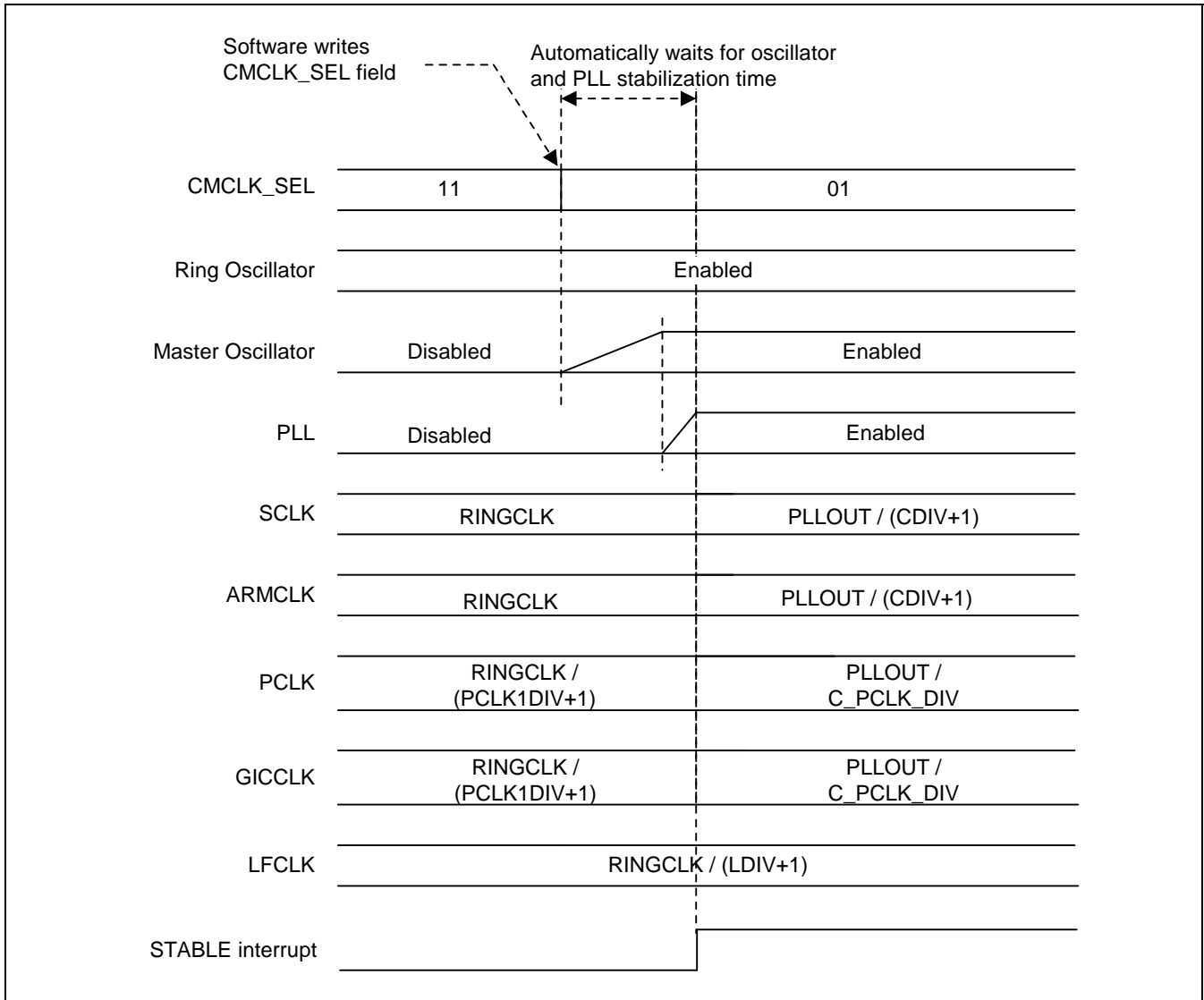


Figure 7-21. Transition Diagram: LOWPOWER Mode to HIGHSPEED Mode (LFSEL=1)

LOWPOWER mode to SLOW mode

- Initial State :
 - It is assumed that the circuit is in LOWPOWER mode before transition.
 - The software must have selected the low-frequency from ring-oscillator (LFOSCEN set to 1, and LFSEL set to 1) All other values are forbidden for the SLOW mode selection.
 - The software must have previously written the appropriate value in CM_OSTR register (master oscillator stabilization time)
- Transition trigger :
 - The transition is triggered upon software control when SLOW mode code is written in CM_SEL register.
- Final State :
 - After automatic transition, the circuit resumes in SLOW mode, where the system clock is derived from master oscillator through frequency divider.
 - LFCLK clock is RINGCLK / (LDIV+1).
 - STABLE interrupt occurs.(if programmed)
- Important note :
 - During transition, the system clock remains derived from RINGCLK. Then, when the main oscillator is stable, the STABLE interrupt is issued so that the software can be informed of the effective clock source change.

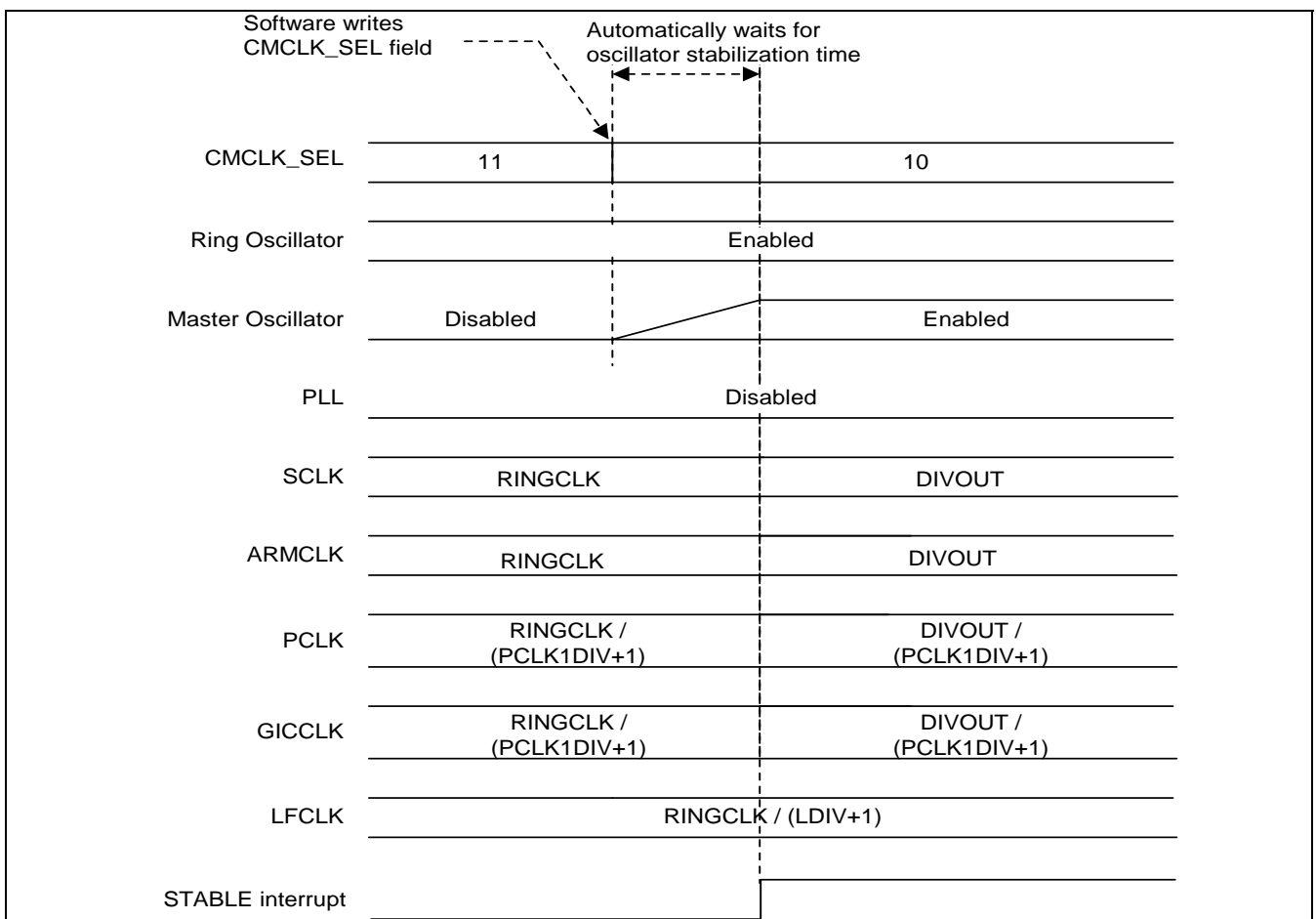


Figure 7-22. Transition Diagram: LOWPOWER Mode to SLOW Mode

LOWPOWER mode to HALT mode

- Initial State :
 - It is assumed that the circuit is in LOWPOWER mode before transition.
- Transition trigger :
 - The transition is triggered upon software control when the HALTMODE is set in CM_CR register.
- Final State :
 - After automatic transition, the circuit resumes in HALT mode, where the ARM clock is disabled, the SCLK clock is derived from LFCLK, the GIC clock is derived from LFCLK and others system clocks activity depends on CM_WFIR value.
 - LFCLK clock status depends on the LFSEL field value.
 - STABLE interrupt occurs.(if programmed)

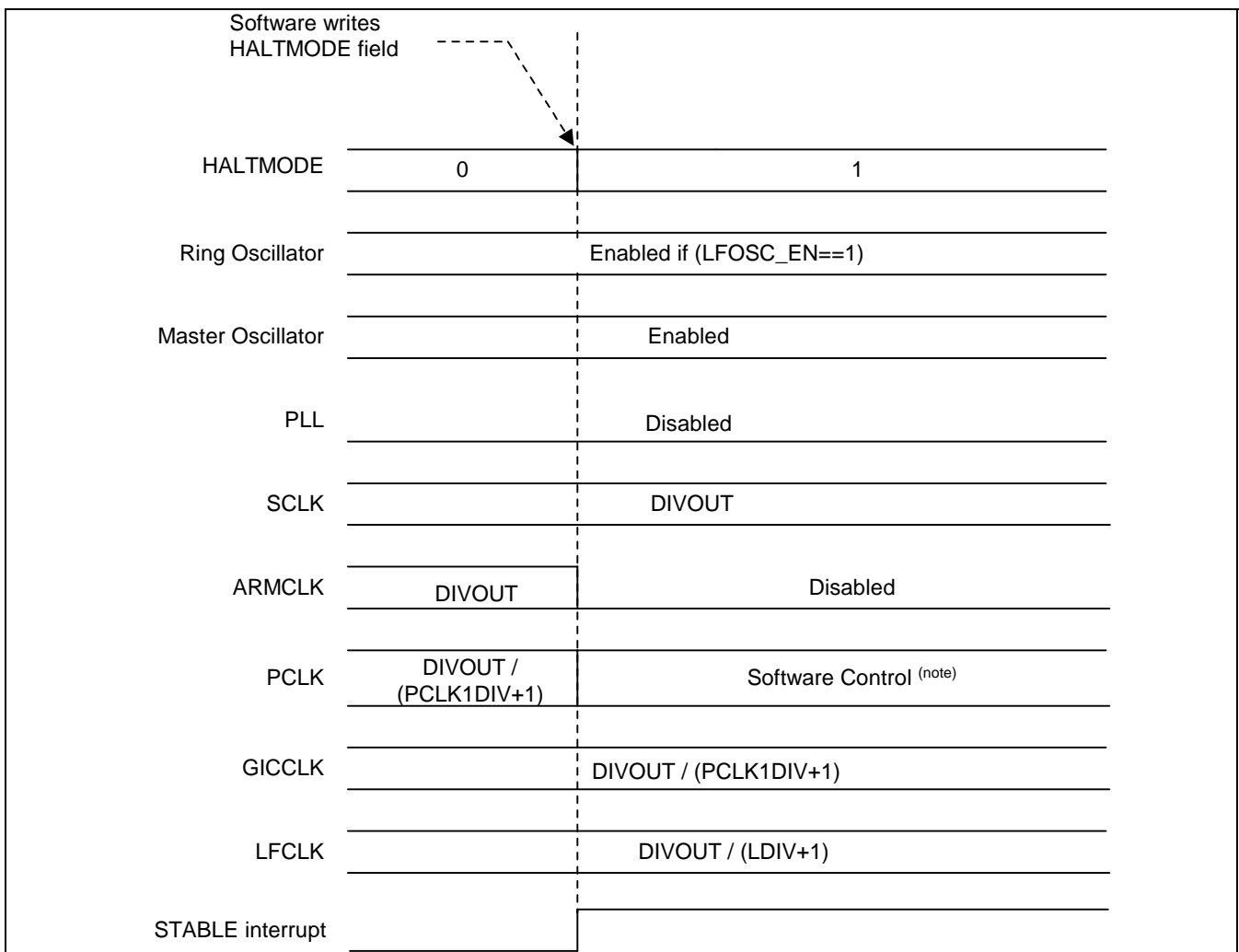


Figure 7-23. Transition Diagram: LOWPOWER Mode to HALT Mode (LFSEL==0)

NOTE: PCLK activity depends on the CM_WIFR register value : either (DIVOUT / (PCLK1DIV+1)) or disabled

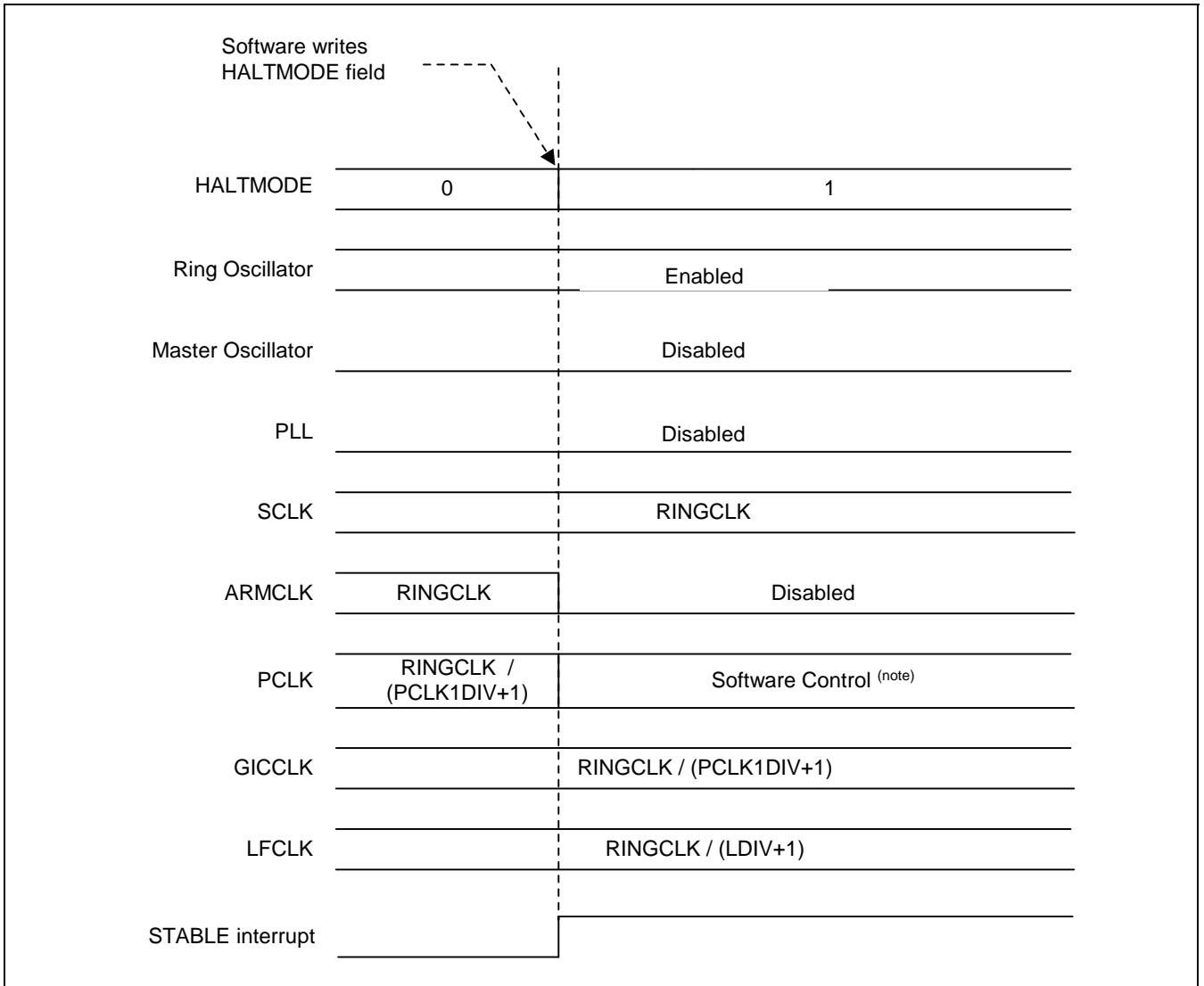
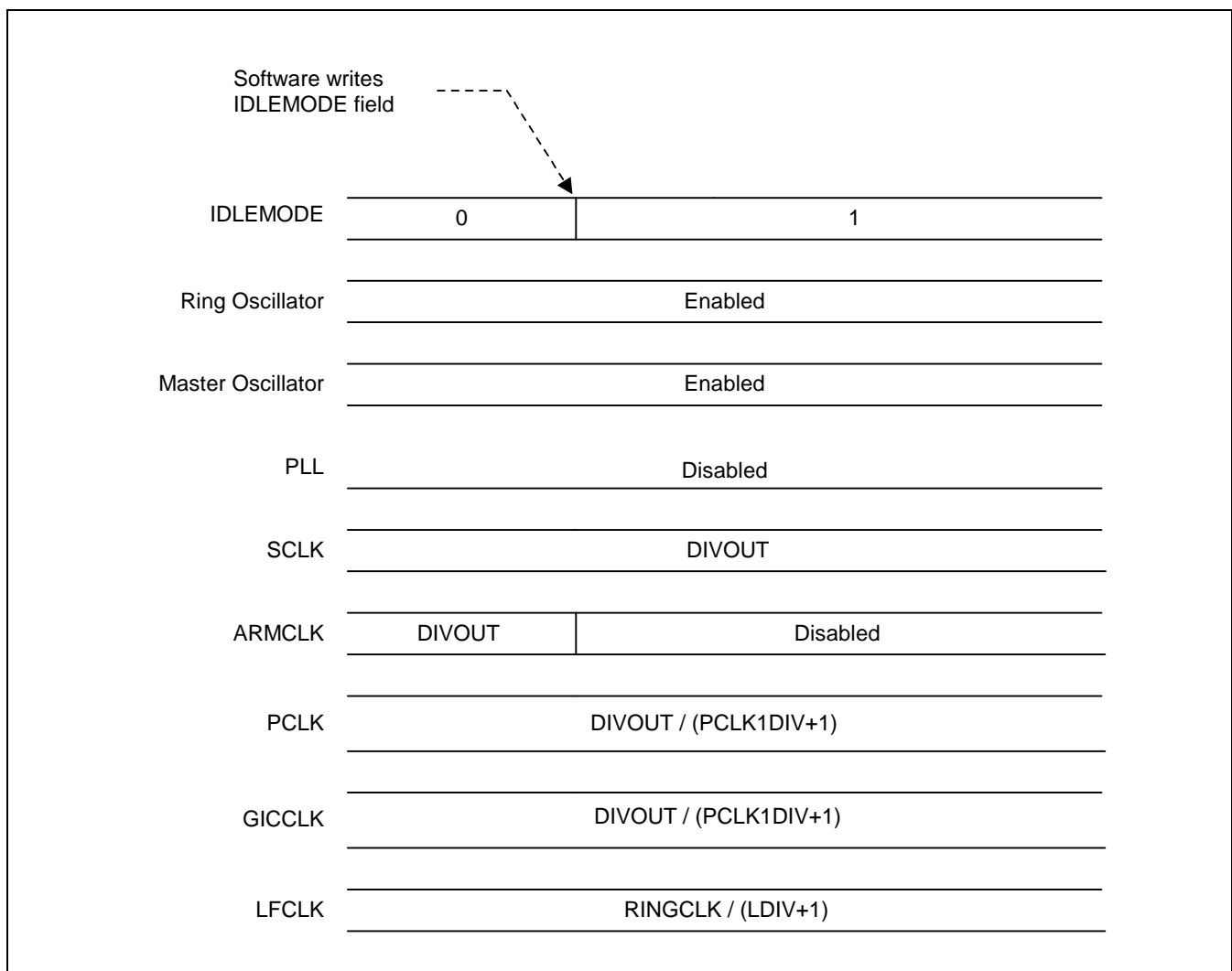


Figure 7-24. Transition Diagram: LOWPOWER Mode to HALT Mode (LFSEL==1)

NOTE: PCLK activity depends on the CM_WIFR register value : either (RINGCLK / (PCLK1DIV+1)) or disabled

SLOW mode to IDLE-SLOW mode

- Initial State :
 - It is assumed that the circuit is in SLOW mode before transition.
- Transition trigger :
 - The transition is triggered upon software control when IDLEMODE is set in CM_CR register
- Final State :
 - After automatic transition, the circuit resumes to IDLE-SLOW mode which is the same as SLOW mode except that ARM7 clock is disabled.

**Figure 7-25. Transition Diagram: SLOW Mode to IDLE-SLOW Mode**

IDLE-SLOW to SLOW mode

- Initial State :
 - It is assumed that the circuit is in IDLE-SLOW mode before transition.
- Transition trigger :
 - The transition is triggered upon any valid interrupt from GIC (interrupt controller)
- Final State :
 - After automatic transition, the circuit resumes to SLOW mode in the same state it was before entering the IDLE-SLOW mode.

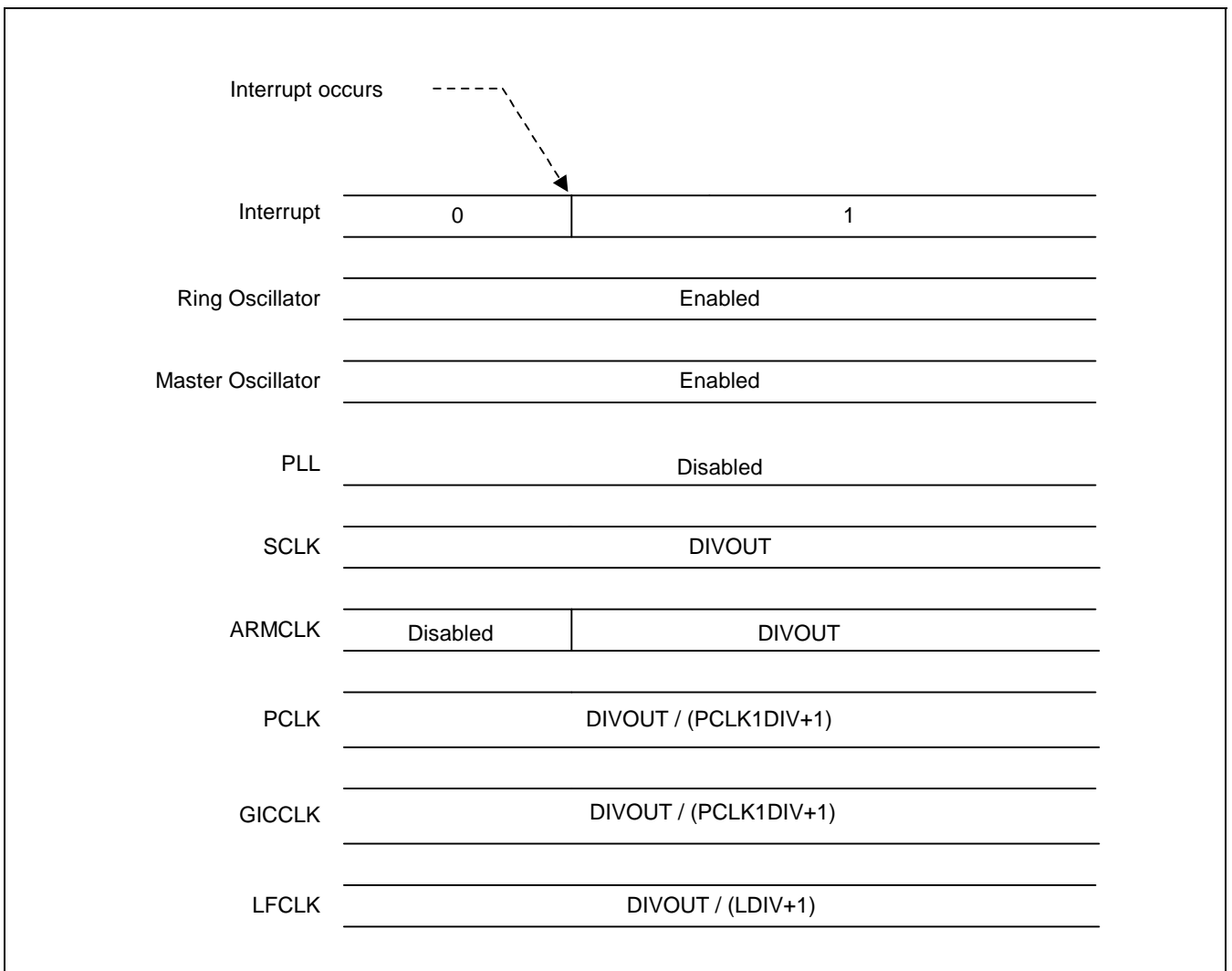


Figure 7-26. Transition Diagram: IDLE-SLOW Mode to SLOW Mode

SLOW mode to NORMAL mde

- Initial State :
 - It is assumed that the circuit is in SLOW mode before transition
- Transition trigger :
 - The transition is triggered upon software control when NORMAL mode code is written in CM_SEL register.
- Final State :
 - After automatic transition, the circuit resumes in NORMAL mode, where the system clock is derived from master oscillator. LFCLK clock is the same as ring oscillator output.
 - STABLE interrupt occurs.(if programmed)

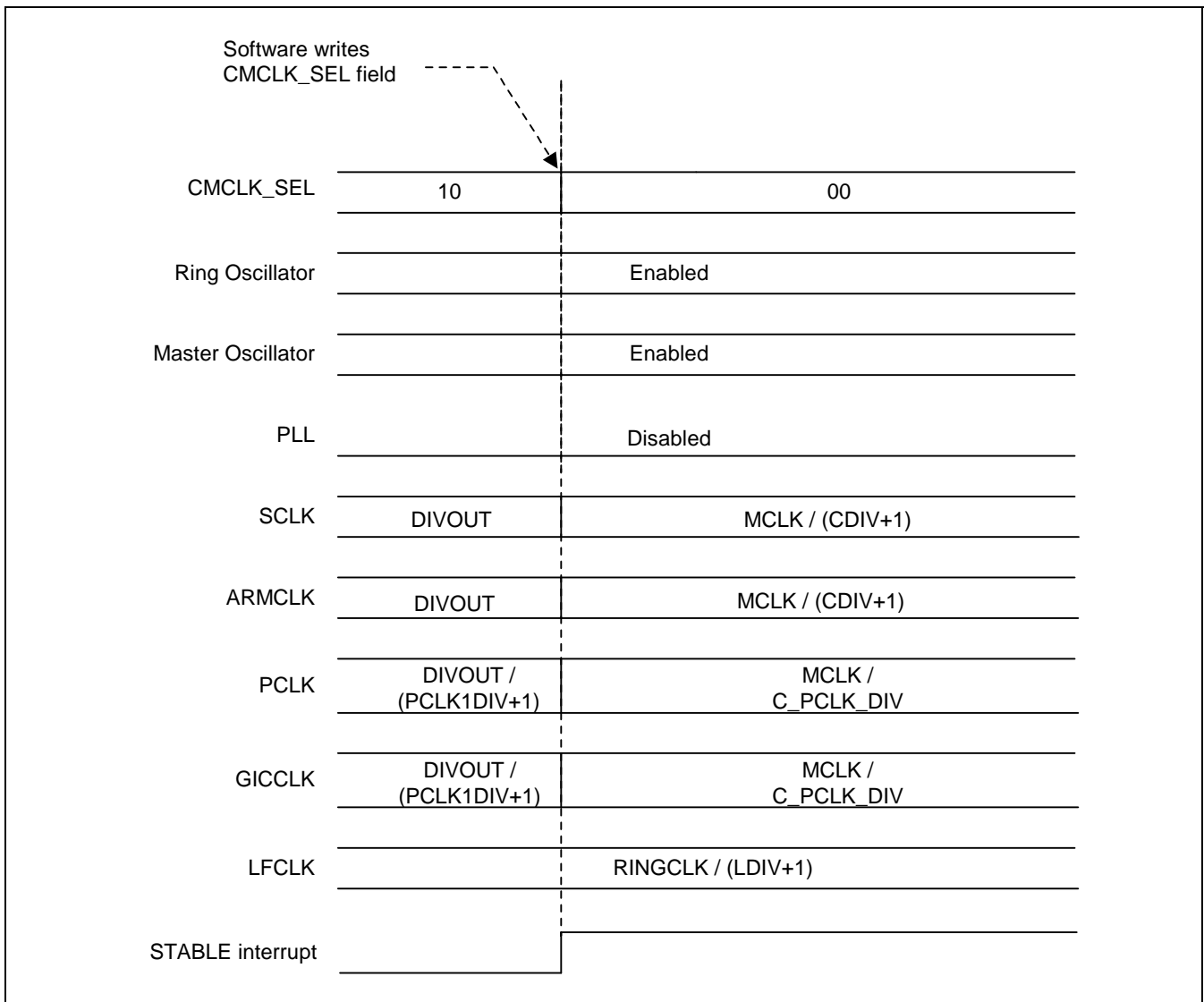


Figure 7-27. Transition Diagram: SLOW Mode to NORMAL Mode

SLOW mode to HIGHSPEED mode

- Initial State :
 - It is assumed that the circuit is in SLOW mode before transition.
 - The software must have previously written the appropriate values in CM_PDPR register (PLL divider parameters) and in CM_PSTR register (PLL stabilization time). This is very important because those registers do not provide valid initial PLL parameters after reset.
- Transition trigger :
 - The transition is triggered upon software control when HIGHSPEED mode code is written in CM_SEL register.
 - During the PLL stabilization time, the slow mode clock is maintained into the circuit.
- Final State :
 - After automatic transition, the circuit resumes in HIGHSPEED mode, where the system clock is derived from PLL output.
 - LFCLK clock is RINGCLK / (LDIV+1).
 - STABLE interrupt occurs.(if programmed)

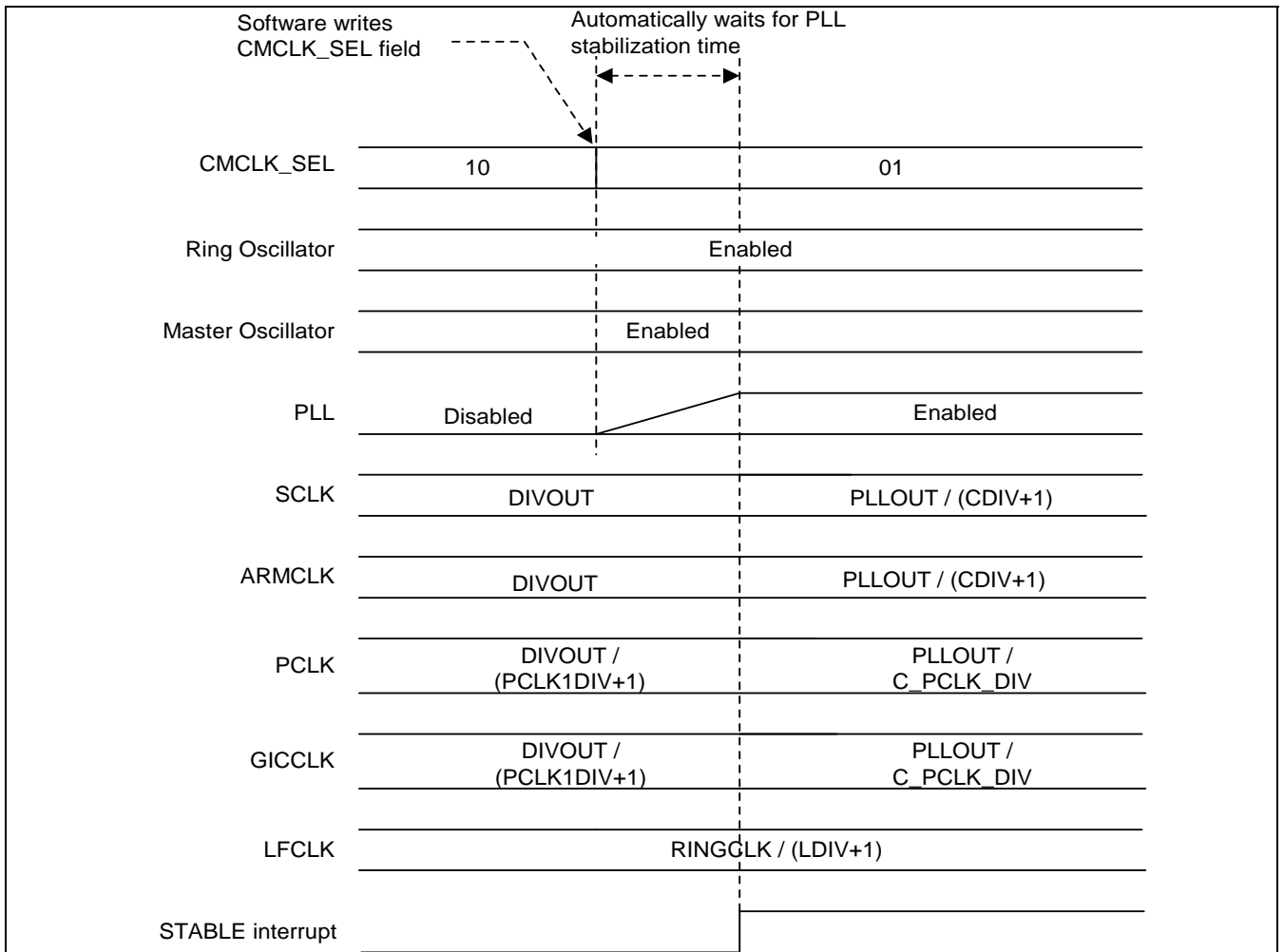


Figure 7-28. Transition Diagram: SLOW Mode to HIGHSPEED Mode

SLOW mode to LOWPOWER mode

- Initial State :
 - It is assumed that the circuit is in SLOW mode before transition.
- Transition trigger :
 - The transition is triggered upon software control when LOWPOWER mode code is written in CM_SEL register.
- Final State :
 - After automatic transition, the circuit resumes in LOWPOWER mode, where the all clocks are derived from LFCLK clock.
 - LFCLK clock is RINGCLK / (LDIV+1).
 - STABLE interrupt occurs.(if programmed)

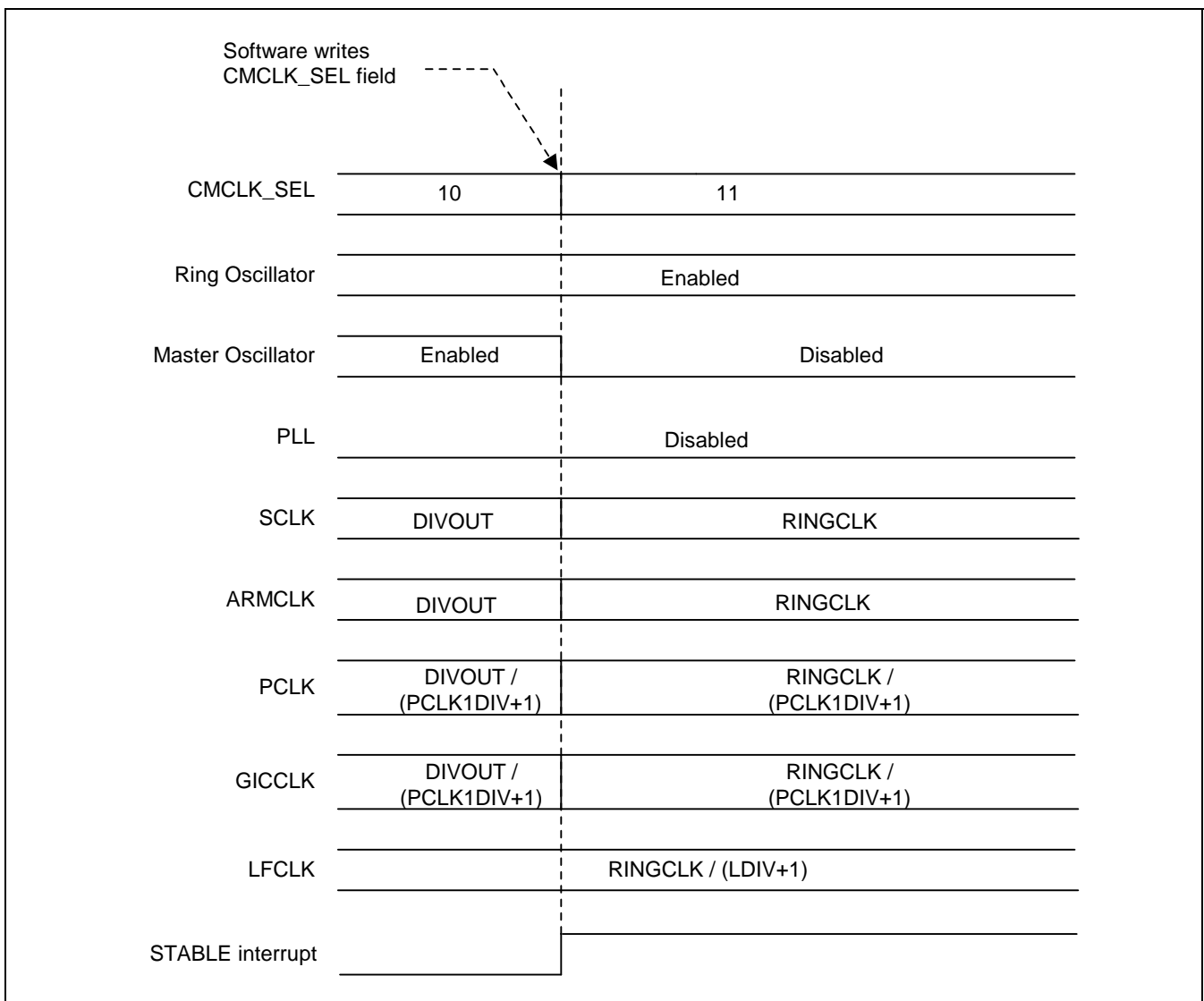


Figure 7-29. Transition Diagram: SLOW Mode to LOWPOWER Mode

SLOW mode to HALT mode

- Initial State :
 - It is assumed that the circuit is in SLOW mode before transition.
- Transition trigger :
 - The transition is triggered upon software control when HALTMODE bit is set in CM_CR register.
- Final State :
 - After automatic transition, the circuit resumes in HALTMODE mode, where the system clock is derived from ring oscillator.
 - LFCLK clock is $RINGCLK / (LDIV+1)$.
 - STABLE interrupt occurs.(if programmed)

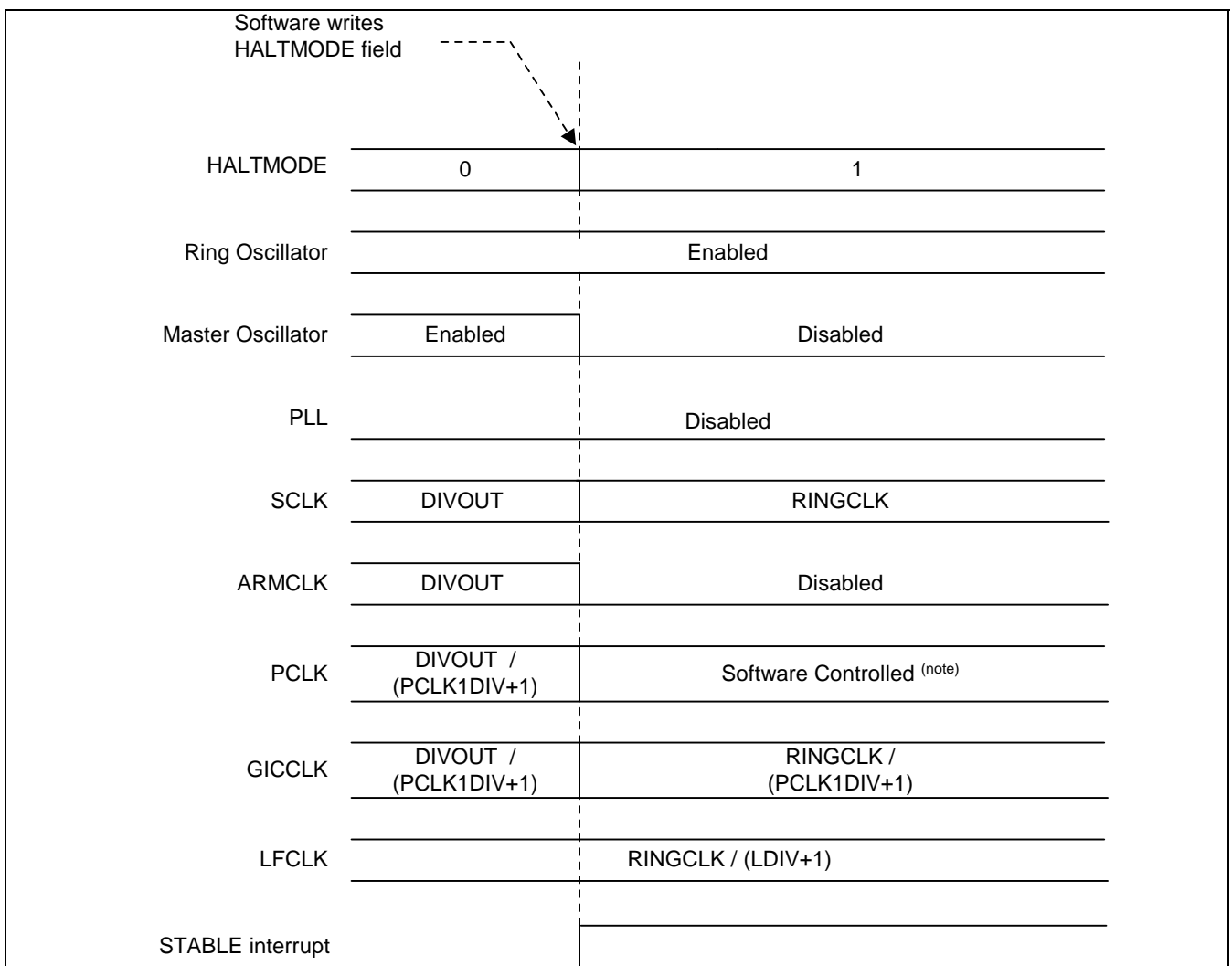


Figure 7-30. Transition Diagram: SLOW Mode to HALT Mode

NOTE: PCLK activity depends on the CM_WIFR register value : either $(RINGCLK / (PCLK1DIV+1))$ or disabled.

HIGHSPEED mode to IDLE-HIGHSPEED mode

- Initial State :
 - It is assumed that the circuit is in HIGHSPEED mode before transition.
- Transition trigger :
 - The transition is triggered upon software control when IDLEMODE is set in CM_CR register
- Final State :
 - After automatic transition, the circuit resumes to IDLE-HIGHSPEED mode which is the same as HIGHSPEED mode except that ARM7 clock is disabled.

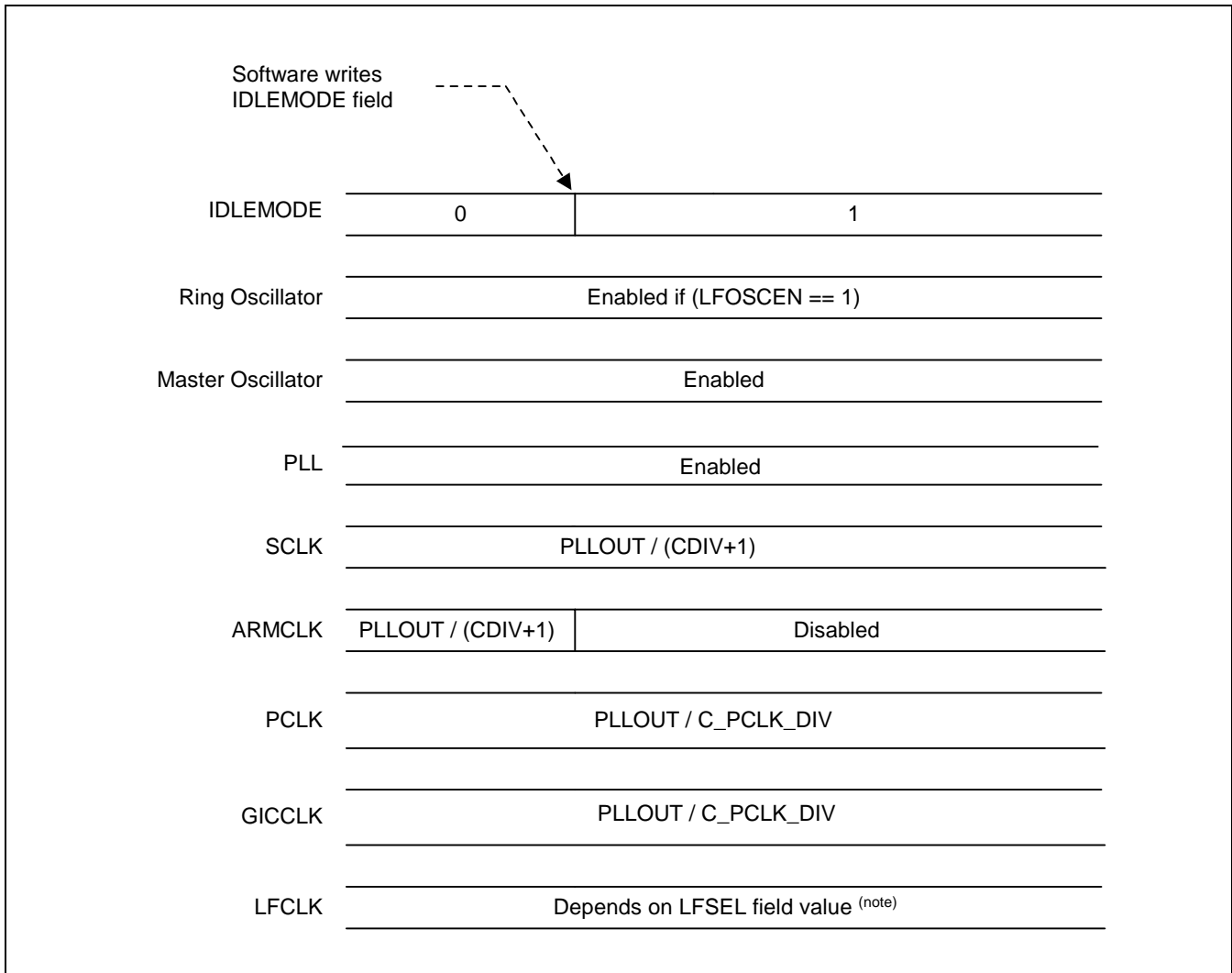
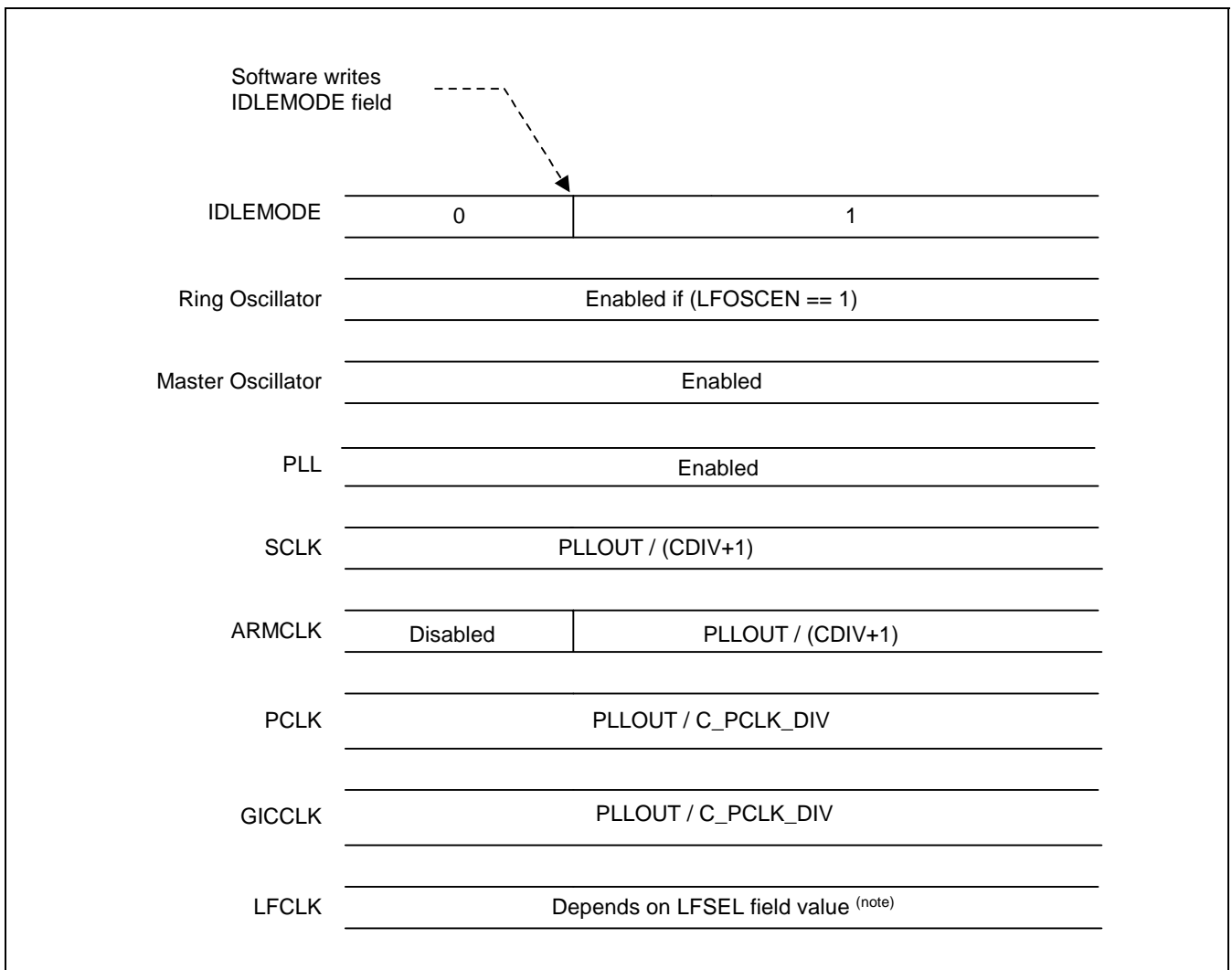


Figure 7-31. Transition Diagram: HIGHSPEED Mode to IDLE-HIGHSPEED Mode

NOTE: If (LS_SEL == 0) then LFCLK is : $MCLK / ((MDIV+1) \times (LDIV+1))$
 Else : LFCLK is : $RINGCLK / (LDIV+1)$

IDLE-HIGHSPEED mode to HIGHSPEED mode

- Initial State :
 - It is assumed that the circuit is in IDLE-HIGHSPEED mode before transition.
- Transition trigger :
 - The transition is triggered upon any valid interrupt from GIC (interrupt controller)
- Final State :
 - After automatic transition, the circuit resumes to HIGHSPEED mode in the same state it was before entering the IDLE-HIGHSPEED mode.

**Figure 7-32. Transition Diagram: HIGHSPEED Mode to IDLE-HIGHSPEED Mode**

NOTE: If (LS_SEL == 0) then LFCLK is : $MCLK / ((MDIV+1) \times (LDIV+1))$
 Else : LFCLK is : $RINGCLK / (LDIV+1)$

HIGHSPEED mode to NORMAL mode

- Initial State :
 - It is assumed that the circuit is in HIGHSPEED mode before transition.
- Transition trigger :
 - The transition is triggered upon software control when NORMAL mode code is written in CM_SEL register.
- Final State :
 - After automatic transition, the circuit resumes in NORMAL mode, where all system clocks are derived from master oscillator clock.
 - LFCLK clock status depends on LFSEL field value.
 - STABLE interrupt occurs.(if programmed)

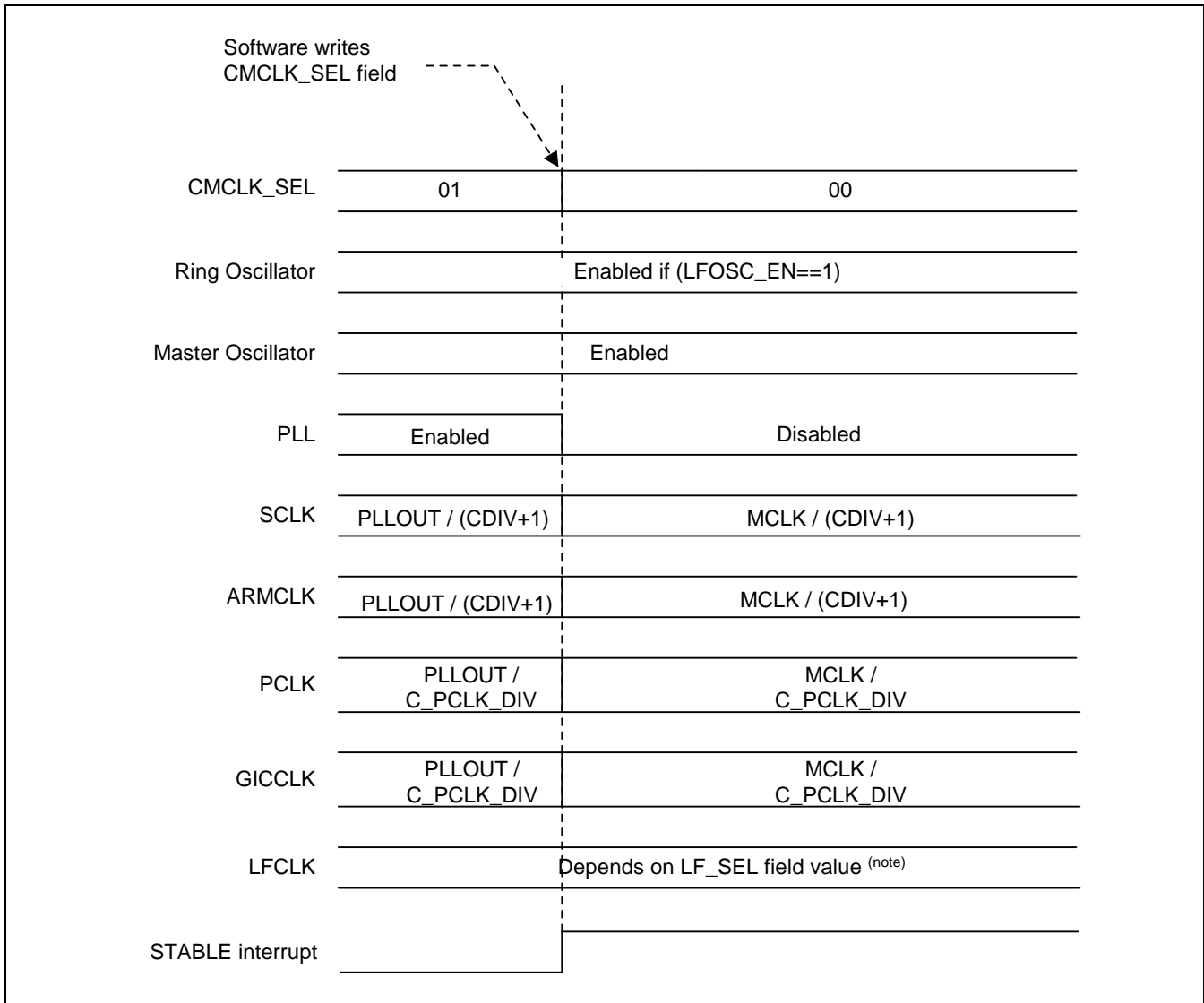


Figure 7-33. Transition Diagram: HIGHSPEED Mode to NORMAL Mode

NOTE: If (LS_SEL == 0) then LFCLK is : $MCLK / ((MDIV+1) \times (LDIV+1))$
 Else : LFCLK is : $RINGCLK / (LDIV+1)$

HIGHSPEED mode to SLOW mode

- Initial State :
 - It is assumed that the circuit is in HIGHSPEED mode before transition.
 - The software must have selected the low-frequency from ring-oscillator (LFOSCEN set to 1, and LFSEL set to 1).
 - All other values are forbidden for the SLOW mode selection.
- Transition trigger :
 - The transition is triggered upon software control when SLOW mode code is written in CM_SEL register.
- Final State :
 - After automatic transition, the circuit resumes in SLOW mode, where the system clock is derived from master oscillator through frequency divider.
 - LFCLK clock is RINGCLK / (LDIV+1).
 - STABLE interrupt occurs.(if programmed)

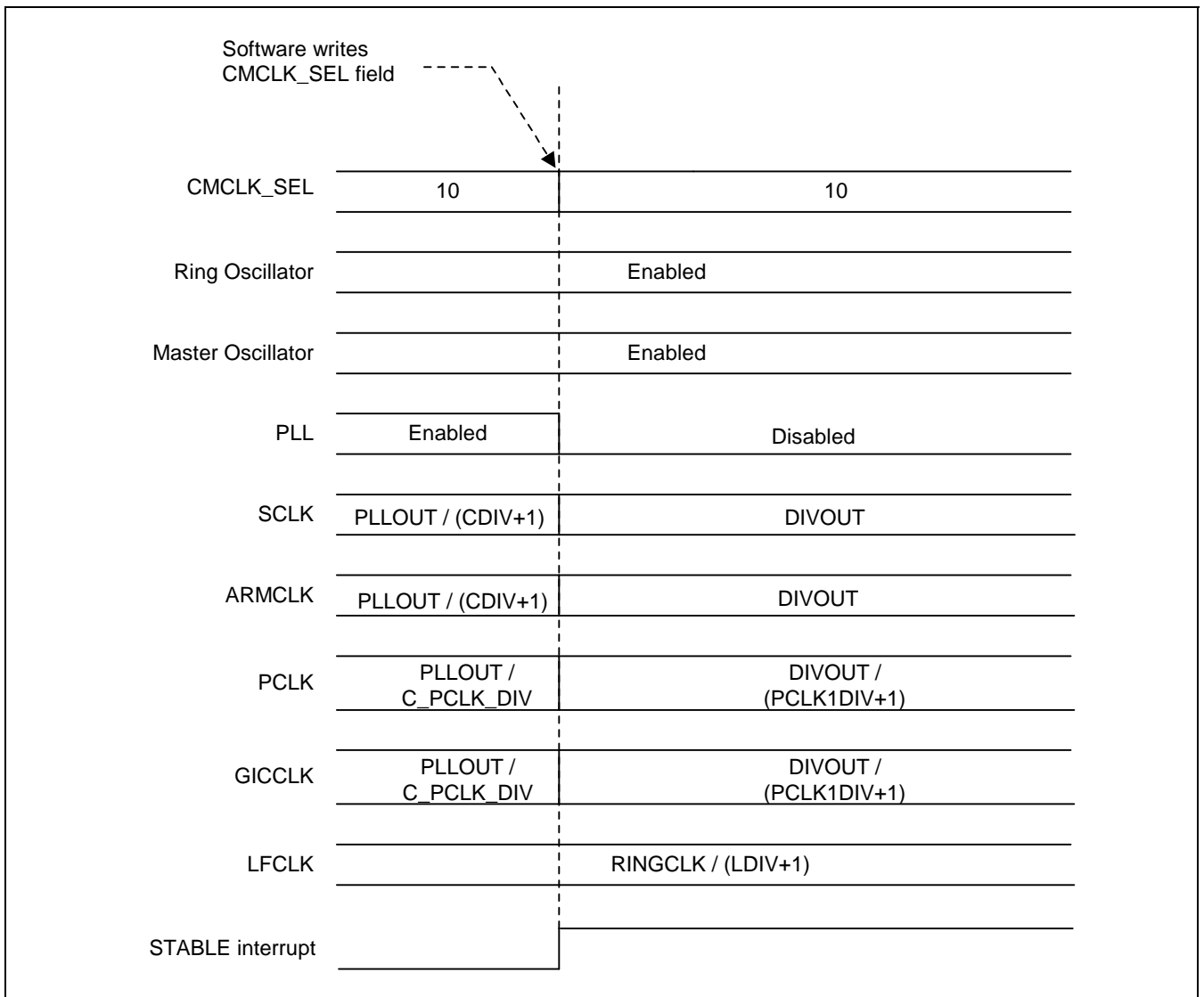


Figure 7-34. Transition Diagram: HIGHSPEED Mode to SLOW Mode

HIGHSPEED mode to LOWPOWER mode

- Initial State :
 - It is assumed that the circuit is in HIGHSPEED mode before transition.
 - The software must have previously written the appropriate value in CM_OSTR register (master oscillator stabilization time)
 - The software must have previously selected the Low-Frequency clock strategy in CM_LFOSCR register. Note that this register must be written only in NORMAL mode.
- Transition trigger :
 - The transition is triggered upon software control when LOWPOWER mode code is written in CM_SEL register.
- Final State :
 - After automatic transition, the circuit resumes in LOWPOWER mode, where all clocks are derived from LFCLK clock.
 - LFCLK clock status depends on the LFSEL field value.
 - STABLE interrupt occurs.(if programmed)

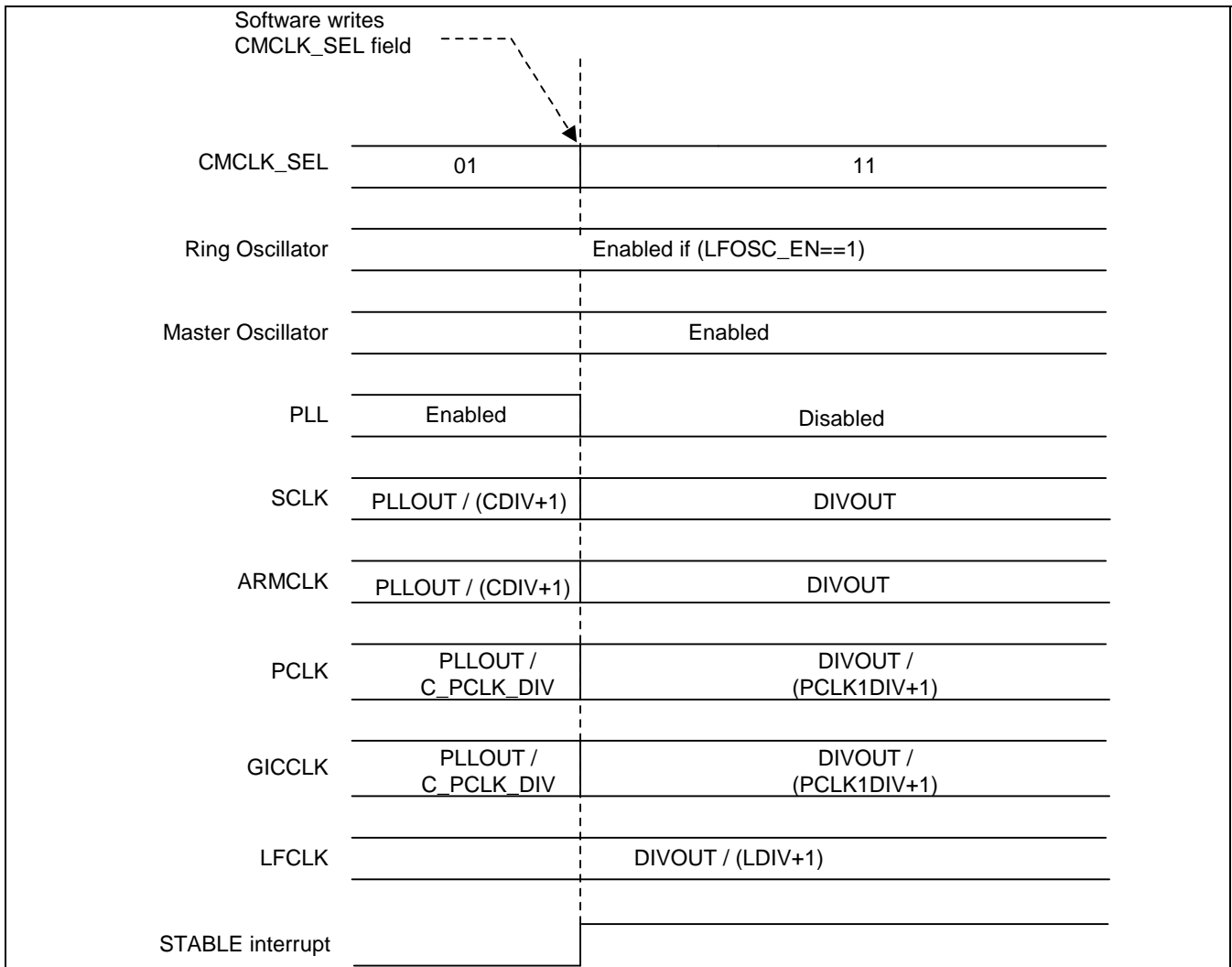


Figure 7-35. Transition Diagram: HIGHSPEED Mode to LOWPOWER Mode (LFSEL=0)

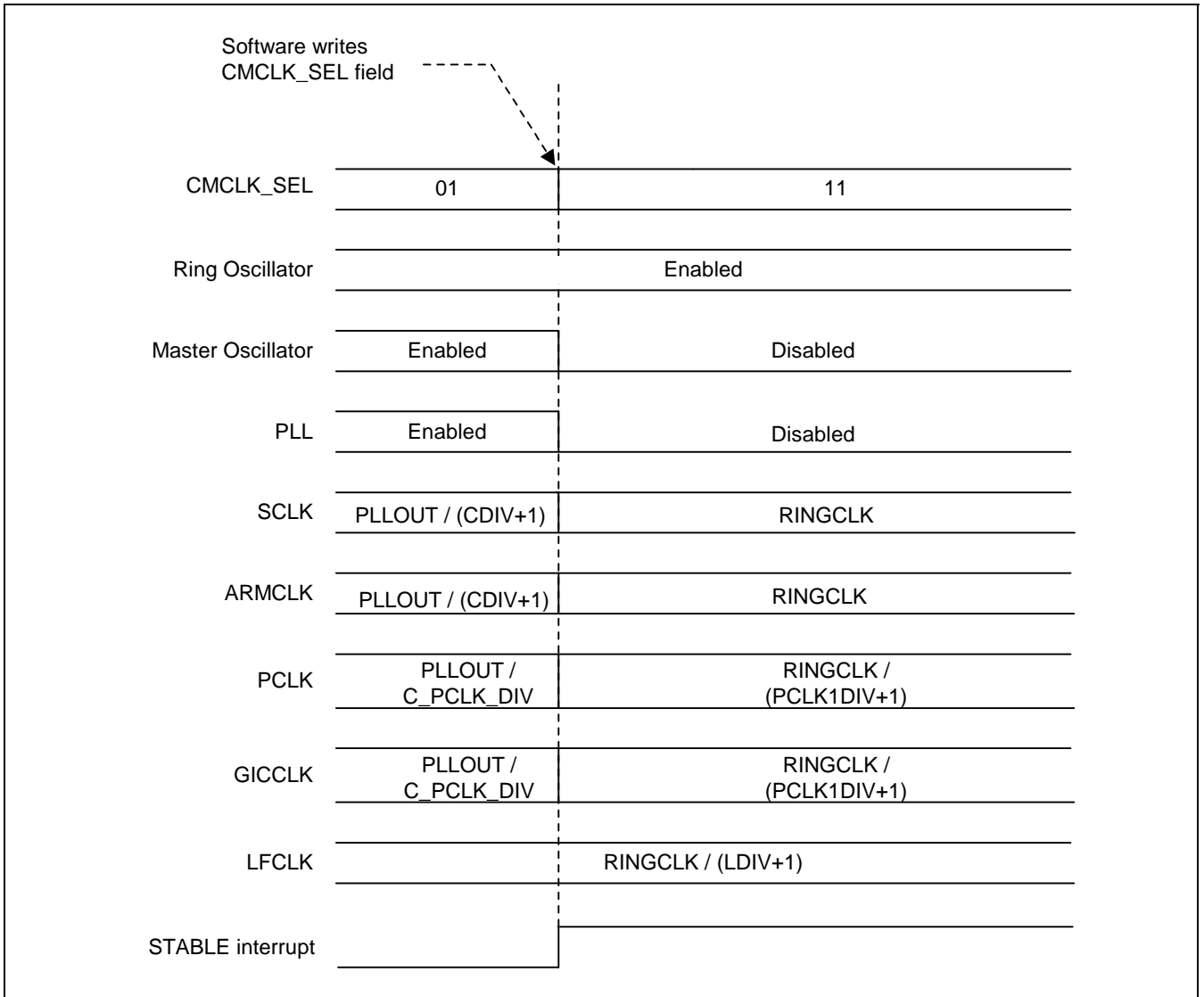


Figure 7-36. Transition Diagram: HIGH SPEED Mode to LOW POWER Mode (LFSEL=1)

HIGHSPEED mode to HALT mode

- Initial State :
 - It is assumed that the circuit is in HIGHSPEED mode before transition.
 - The software must have previously written the appropriate values in CM_PDPR register (PLL dividers parameters), CM_PSTR register (PLL stabilization time) and CM_OSTR (master oscillator stabilization time). This is very important because those registers do not provide valid initial PLL parameters after reset.
 - The software must have previously selected the Low-Frequency clock strategy in CM_LFOSCR register. Note that this register must be written only in NORMAL mode.
- Transition trigger :
 - The transition is triggered upon software control when HALTMODE is set in CM_CR register.
- Final State :
 - After automatic transition, the circuit resumes in HALT mode, where the ARM clock is disabled, the SCLK clock is derived from LFCLK, the GIC clock is derived from LFCLK and others system clocks activity depends on CM_WIFR value.
 - LFCLK clock status depends on the LFSEL field value.
 - STABLE interrupt occurs.(if programmed)

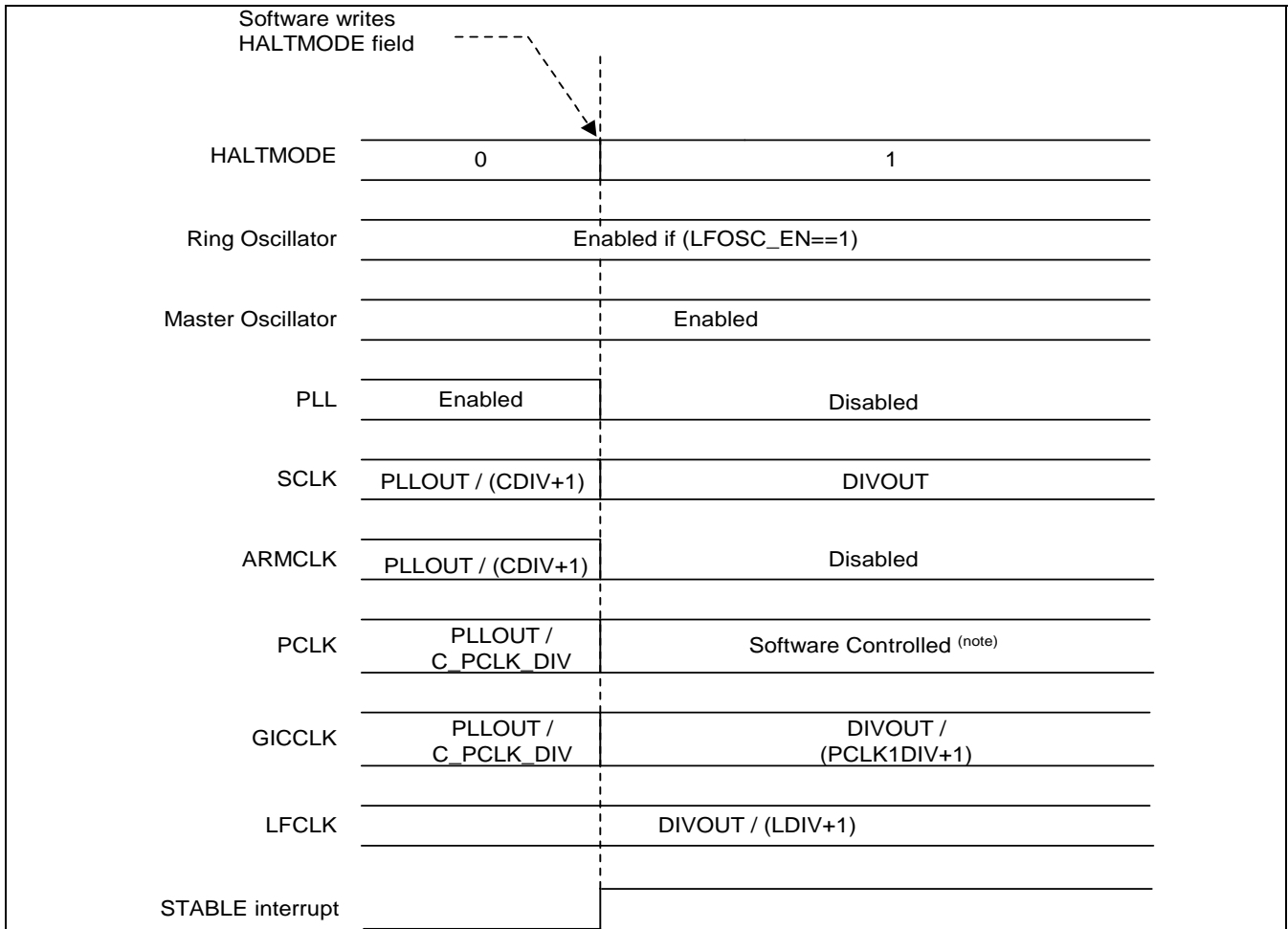


Figure 7-37. Transition Diagram: HIGHSPEED Mode to HALT Mode (LFSEL=0)

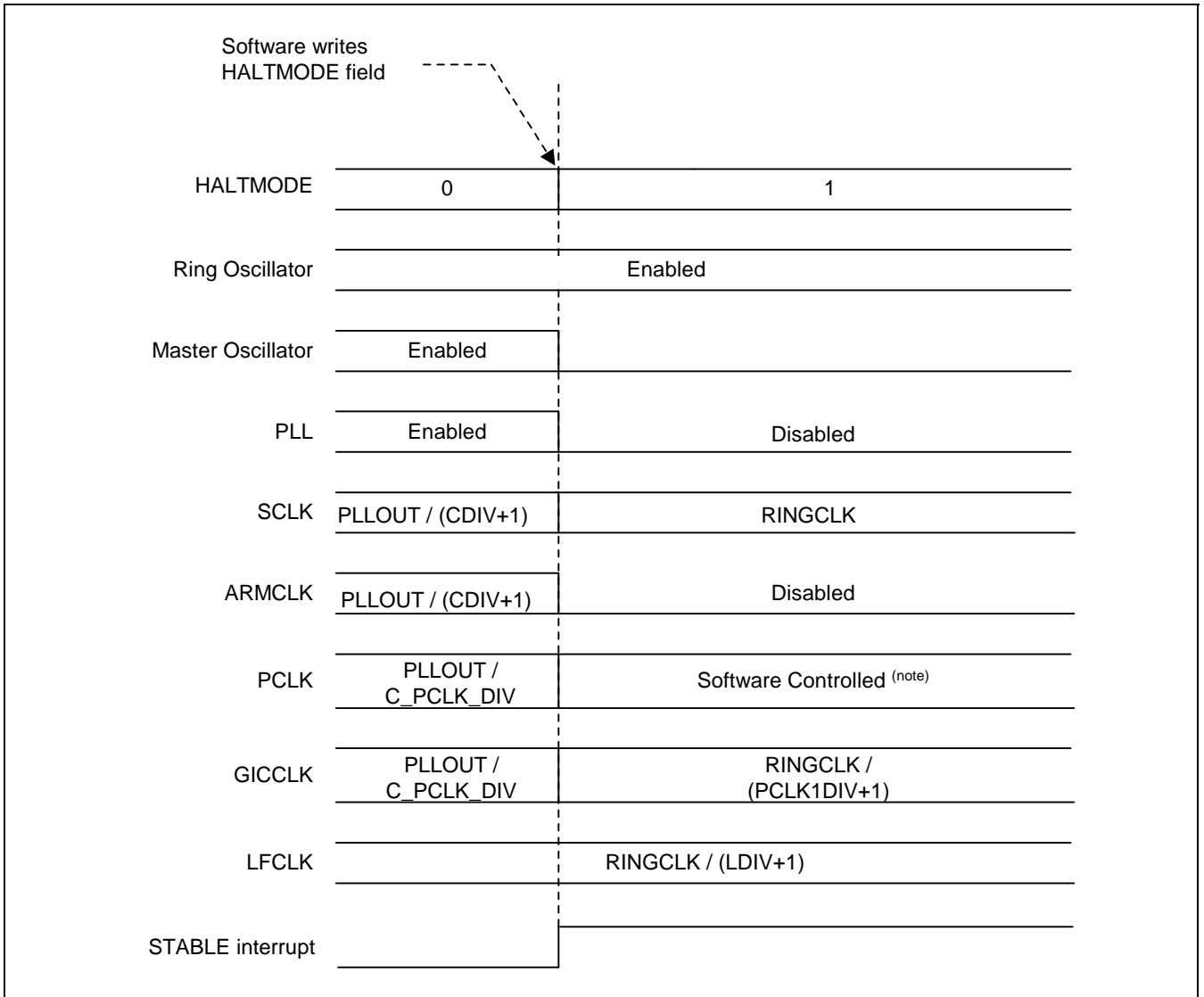


Figure 7-38. Transition Diagram: HIGH SPEED Mode to HALT Mode (LFSEL=1)

NOTE: PCLK activity depends on the CM_WIFR register value : either (DIVOUT / (PCLK1DIV+1)) or disabled

HALT mode to NORMAL mode

- Initial State :
 - It is assumed that the circuit is in HALT mode before transition.
 - It is also assumed that the circuit was in NORMAL mode before entering the HALT mode
 - The software must have previously written the appropriate value in CM_OSTR register (master oscillator stabilization time)
- Transition trigger :
 - The transition is triggered upon valid interrupt received by GIC module.
- Final State :
 - After automatic transition, the circuit resumes to NORMAL mode where all system clocks are derived from master oscillator.
 - LFCLK clock status depends on the LFSEL value.
 - STABLE interrupt occurs.(if programmed)
- Important note :
 - During transition, when LFSEL==1, the system clock remains derived from RINGCLK. Then, when the main oscillator is stable, the STABLE interrupt is issued so that the software can be informed of the effective clock source change.

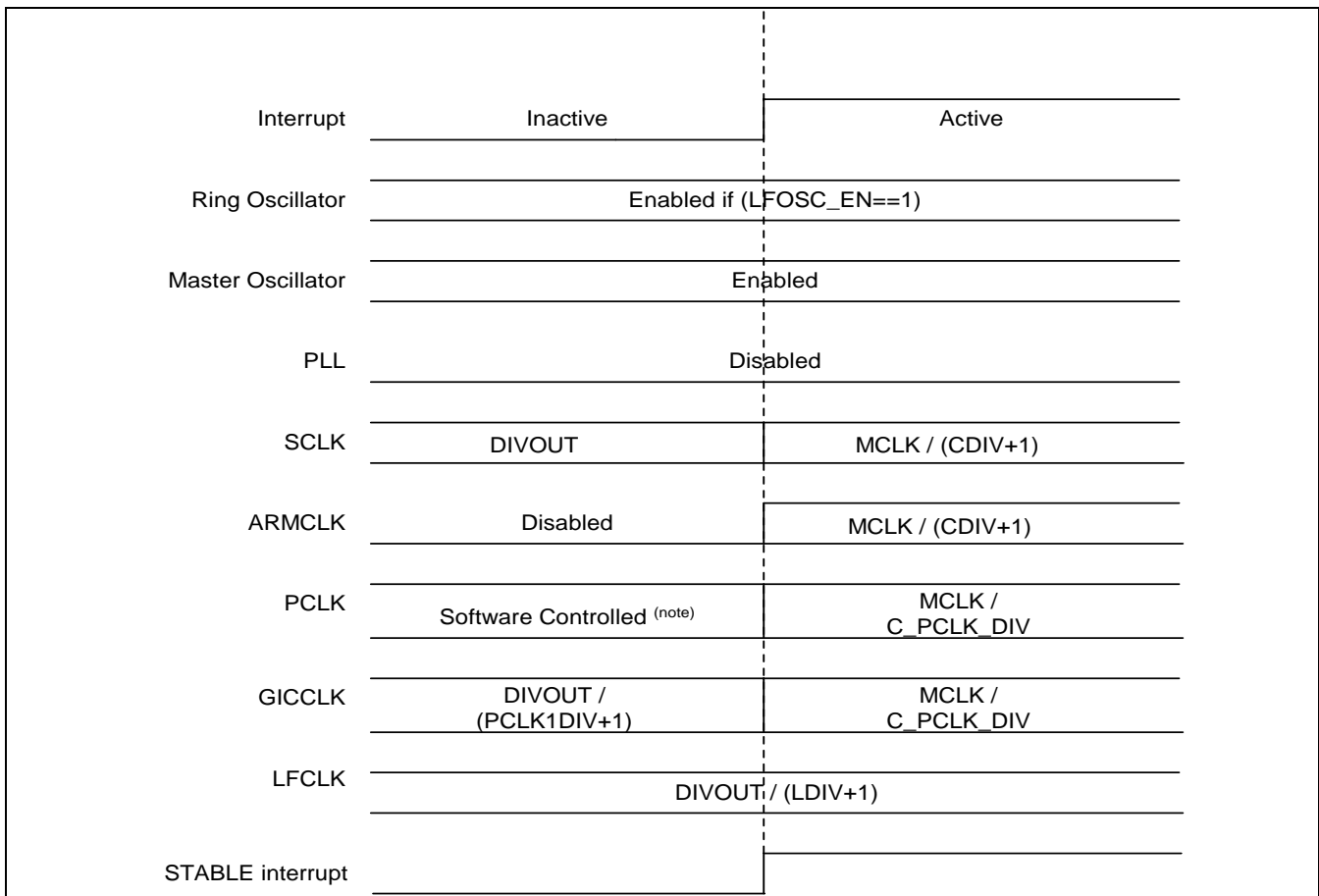


Figure 7-39. Transition Diagram: HALT Mode to NORMAL Mode (LFSEL=0)

NOTE: PCLK activity depends on the CM_WIFR register value : either (DIVOUT / (PCLK1DIV+1)) or disabled.

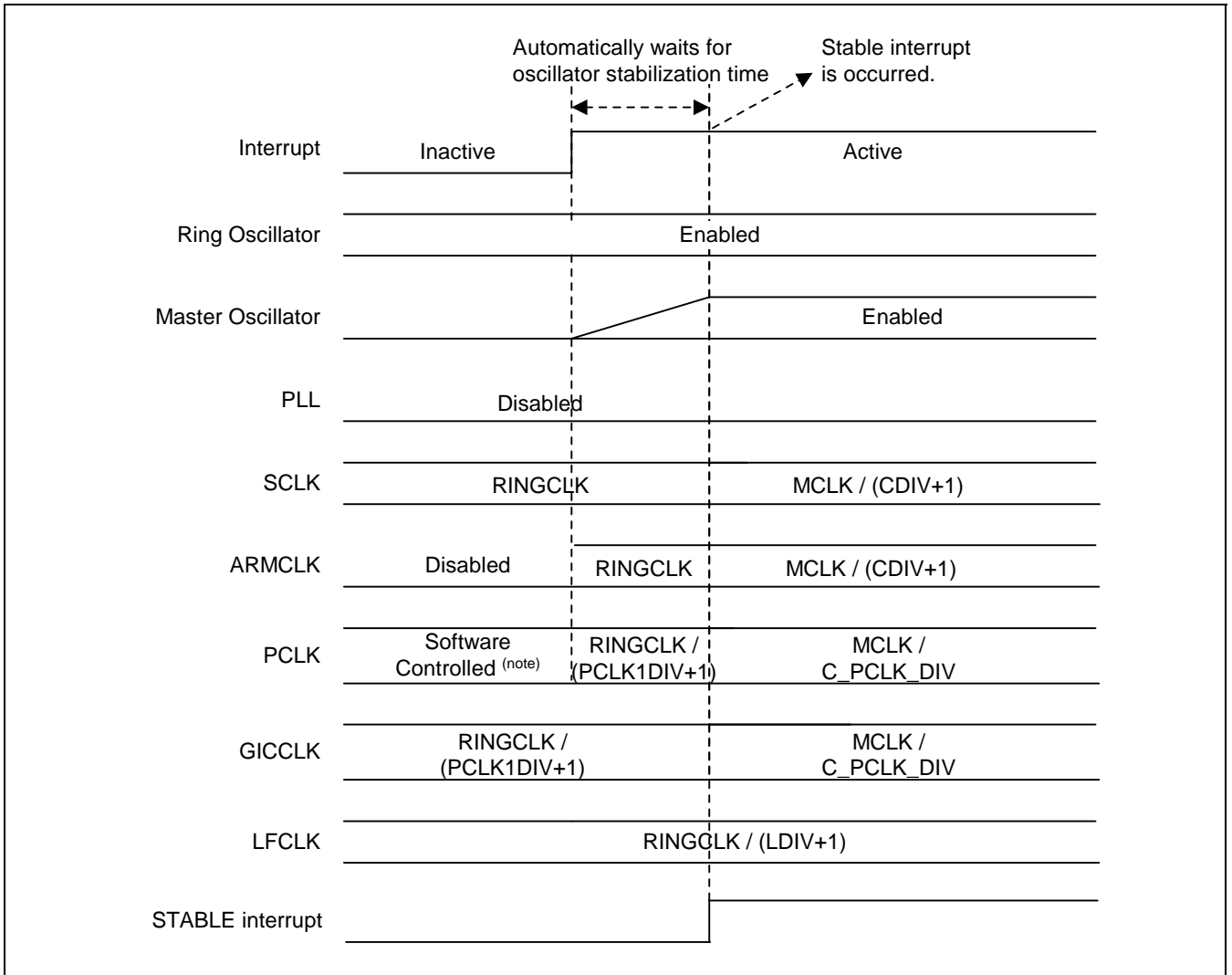


Figure 7-40. Transition Diagram: HALT Mode to NORMAL Mode (LFSEL=1)

NOTE: PCLK activity depends on the CM_WIFR register value : either (RINGCLK / (PCLK1DIV+1)) or disabled

HALT mode to HIGH SPEED mode

- Initial State :
 - It is assumed that the circuit is in HALT mode before transition.
 - It is also assumed that the circuit was in HIGH SPEED mode before entering the HALT mode
 - The software must have previously written the appropriate values in CM_PDPR register (PLL dividers parameters), CM_PSTR register (PLL stabilization time) and CM_OSTR (master oscillator stabilization time). This is very important because those registers do not provide valid initial PLL parameters after reset.
- Transition trigger :
 - The transition is triggered upon valid interrupt received by GIC module.
 - During the master oscillator(if LFSEL==1) and PLL stabilization time, the low frequency clock is always still provided to the circuit.
- Final State :
 - After automatic transition, the circuit resumes to HIGH SPEED mode where all system clocks are derived from PLL output;
 - LFCLK clock status depends on the LFSEL value.
 - STABLE interrupt occurs.(if programmed)

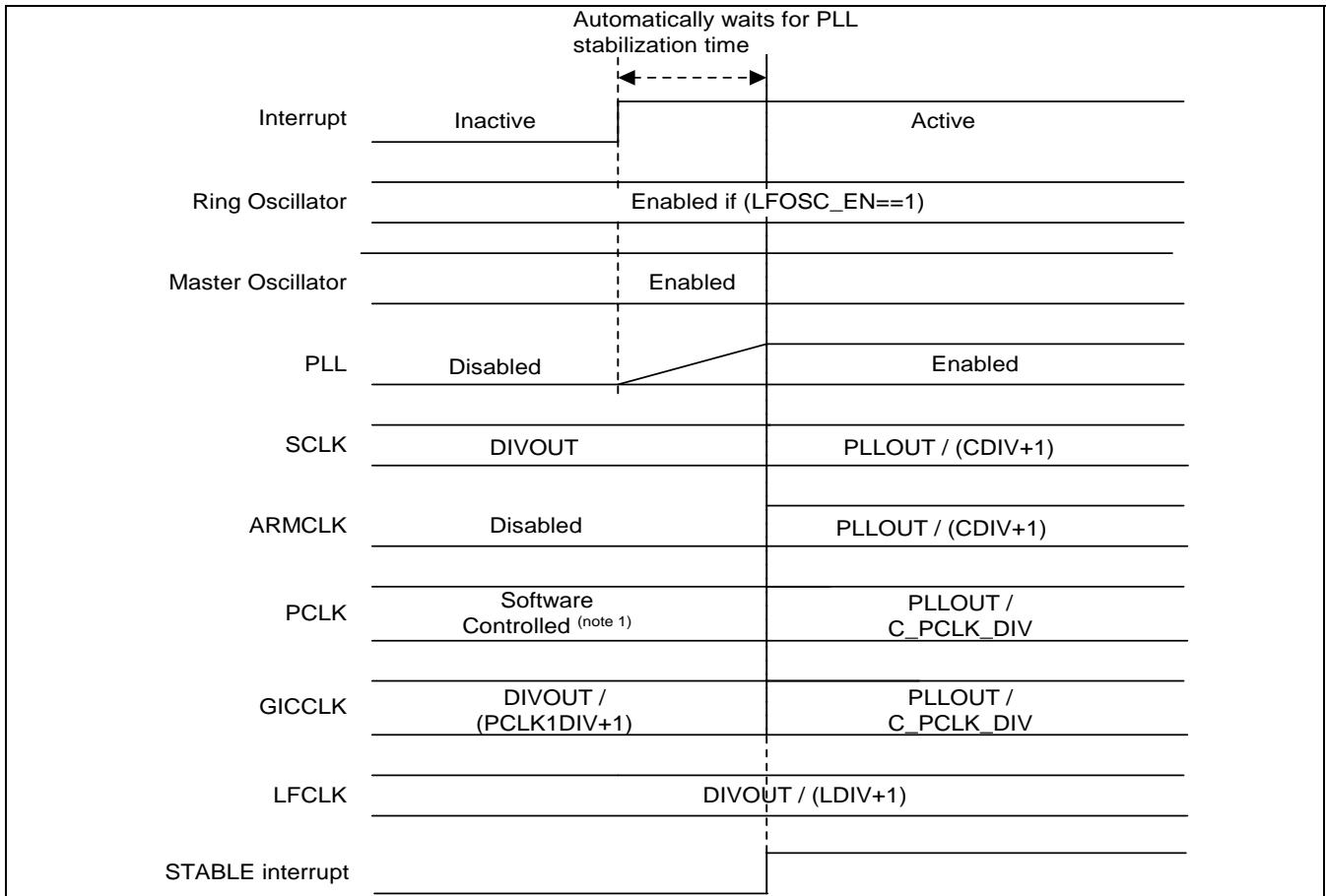


Figure 7-41. Transition Diagram: HALT Mode to HIGH SPEED Mode (LFSEL=0)

NOTE: PCLK activity depends on the CM_WIFR register value : either (DIVOUT / (PCLK1DIV+1)) or disabled.

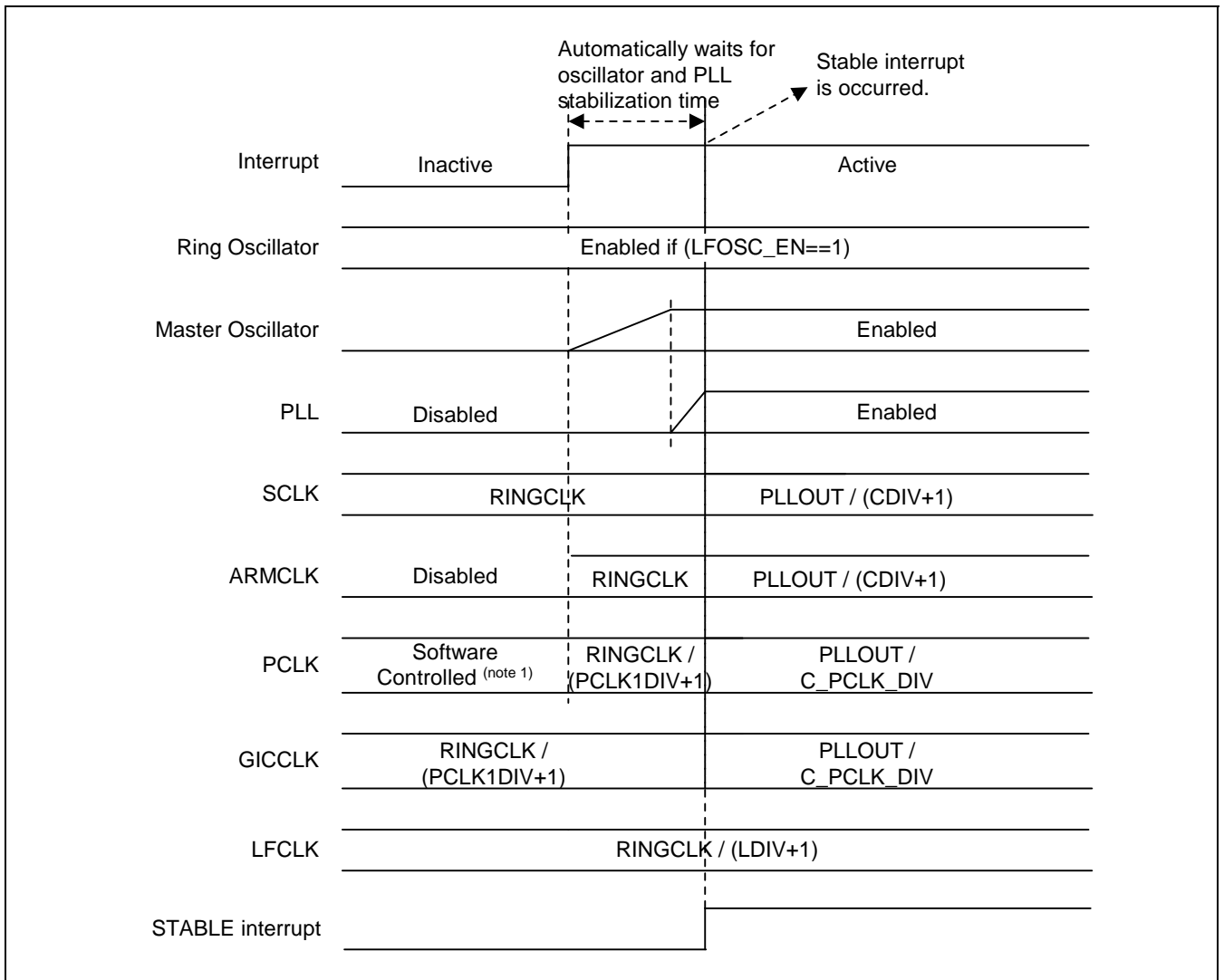


Figure 7-42. Transition Diagram: HALT Mode to HIGH SPEED Mode (LFSEL=1)

NOTE: PCLK activity depends on the CM_WIFR register value : either (RINGCLK / (PCLK1DIV+1)) or disabled.

HALT mode to SLOW mode

- Initial State :
 - It is assumed that the circuit is in HALT mode before transition.
 - It is also assumed that the circuit was in SLOW mode before entering the HALT mode
 - The software must have previously written the appropriate value in CM_OSTR register (master oscillator stabilization time)
- Transition trigger :
 - The transition is triggered upon valid interrupt received by GIC module.
- Final State :
 - After automatic transition, the circuit resumes in SLOW mode, where the system clock is derived from master oscillator through frequency divider.
 - LFCLK clock is RINGCLK / (LDIV+1).
 - STABLE interrupt occurs.(if programmed)
- Important note :
 - During transition, when LFSEL==1, the system clock remains derived from RINGCLK. Then, when the main oscillator is stable, the STABLE interrupt is issued so that the software can be informed of the effective clock source change.

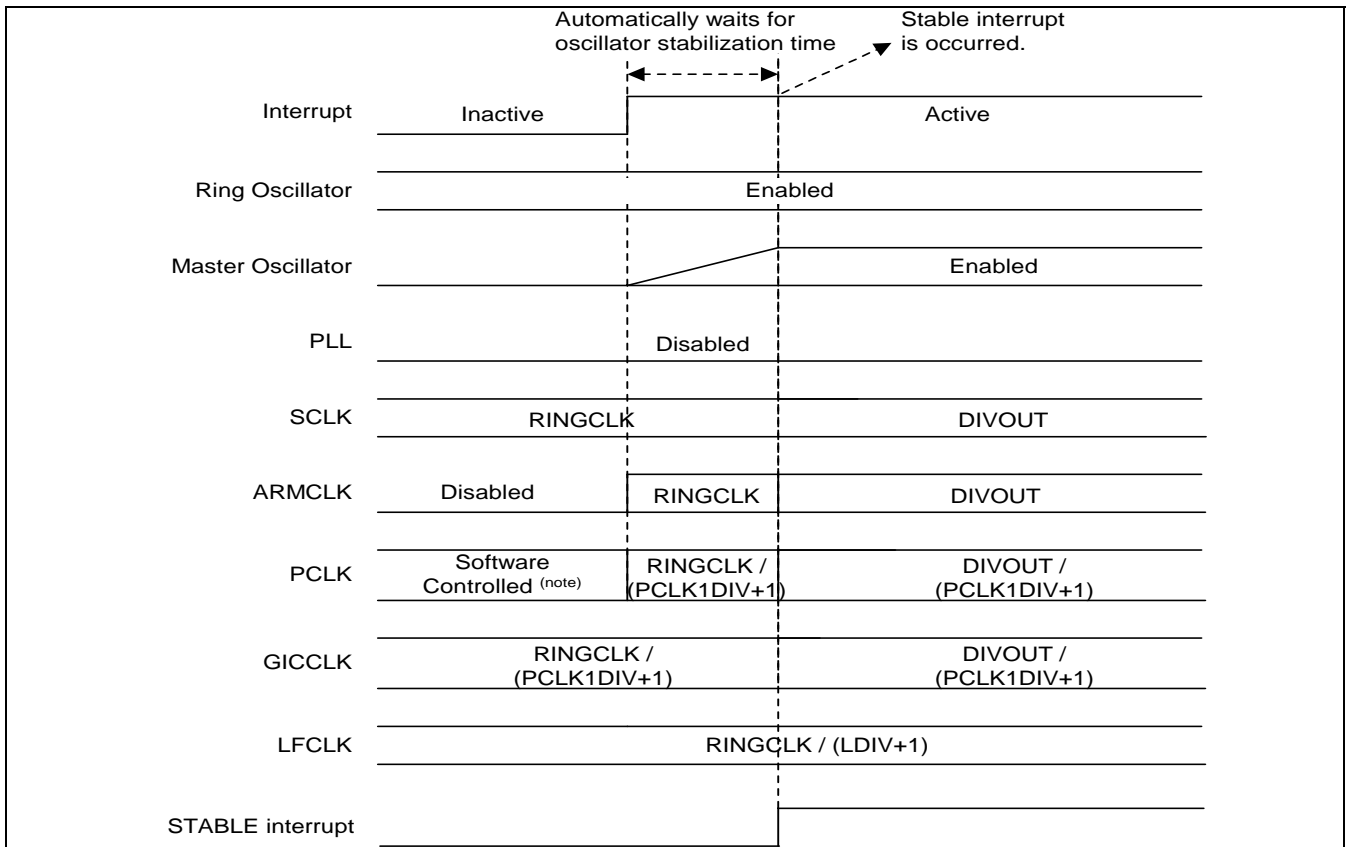


Figure 7-43. Transition Diagram: HALT Mode to SLOW Mode

NOTE: PCLK activity depends on the CM_WIFR register value : either (RINGCLK / (PCLK1DIV+1)) or disabled.

HALT mode to LOWPOWER mode

- Initial State :
 - It is assumed that the circuit is in HALT mode before transition.
 - It is also assumed that the circuit was in LOWPOWER mode before entering the HALT mode
- Transition trigger :
 - The transition is triggered upon valid interrupt received by GIC module.
- Final State :
 - After automatic transition, the circuit resumes in LOWPOWER mode, where the system clock is derived from LFCLK clock.
 - LFCLK clock status depends on the LFSEL field value.
 - STABLE interrupt occurs.(if programmed)

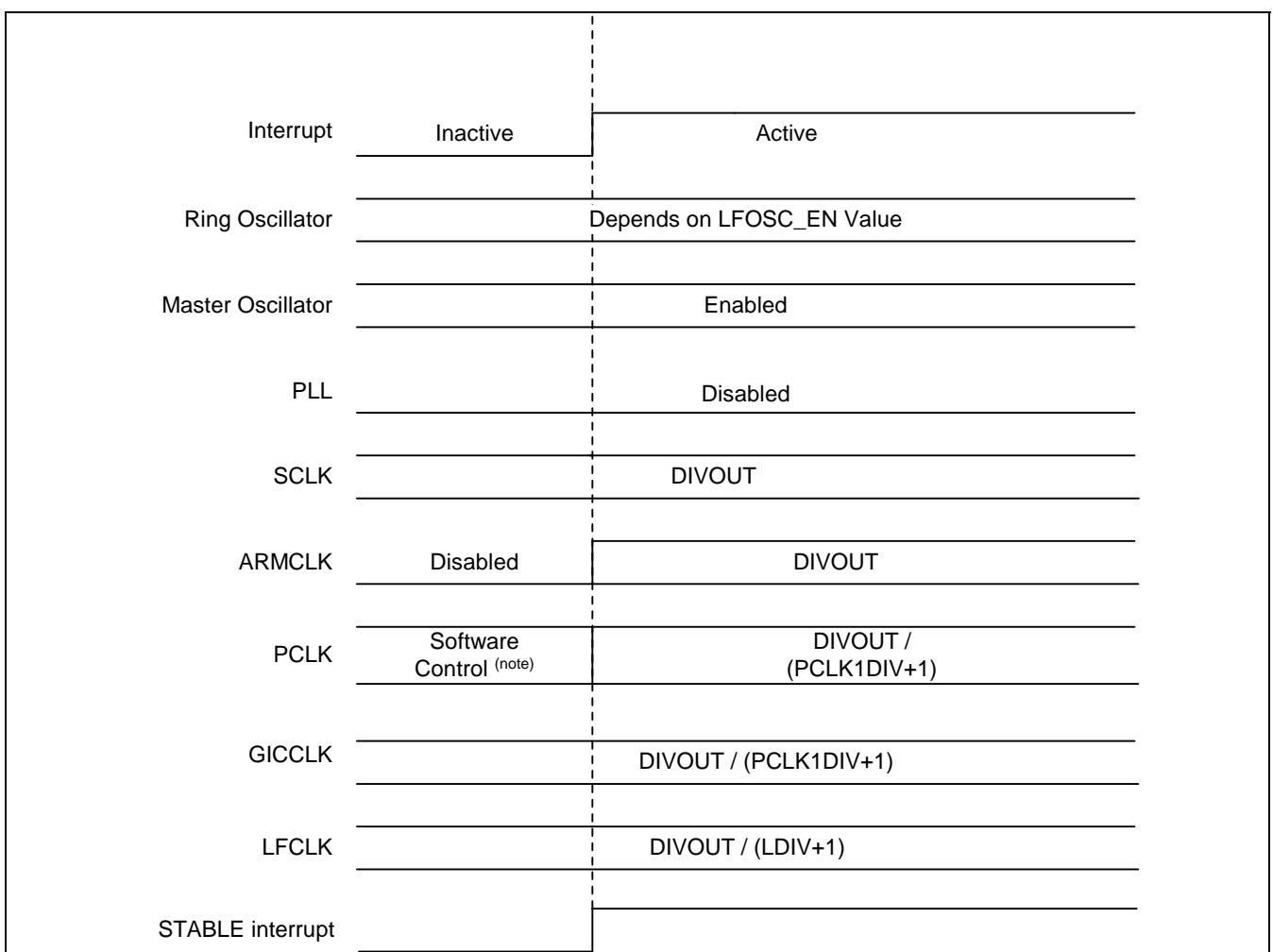


Figure 7-44. Transition Diagram: HALT Mode to LOWPOWER Mode (LFSEL=0)

NOTE: PCLK activity depends on the CM_WIFR register value : either (DIVOUT / (PCLK1DIV+1)) or disabled

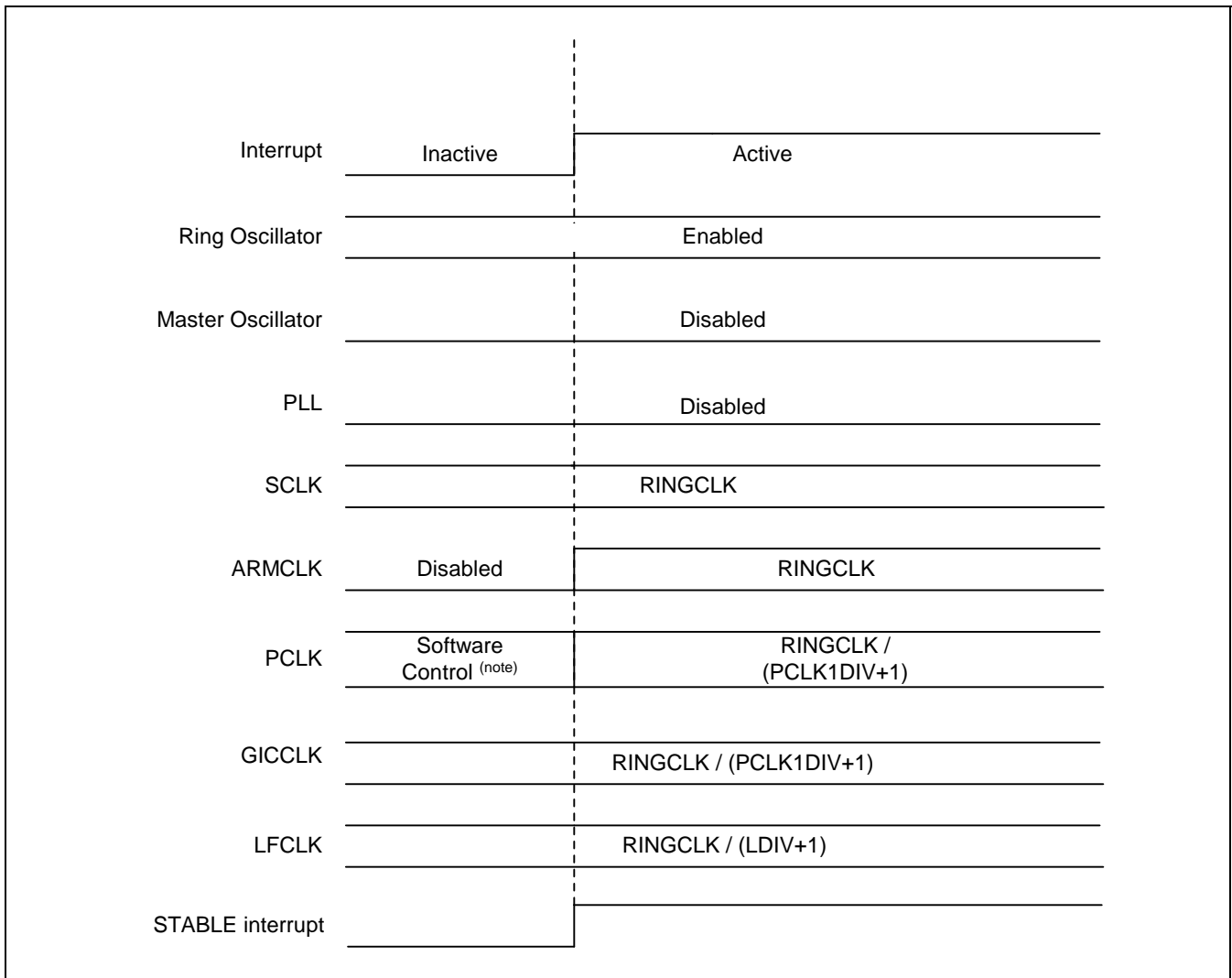


Figure 7-45. Transition Diagram: HALT Mode to LOWPOWER Mode (LFSEL=1)

NOTE: PCLK activity depends on the CM_WIFR register value : either (RINGCLK / (PCLK1DIV+1)) or disabled.

STOP mode to NORMAL mode

- Initial State :
 - It is assumed that the circuit is in STOP mode before transition.
 - The software must have previously written the appropriate value in CM_OSTR register (master oscillator stabilization time)
- Transition trigger :
 - The transition is triggered upon valid interrupt received by GIC module.
- Final State :
 - After automatic transition, the circuit resumes to NORMAL mode where all system clocks are derived from master oscillator
 - LFCLK clock status depends on the LFSEL value.

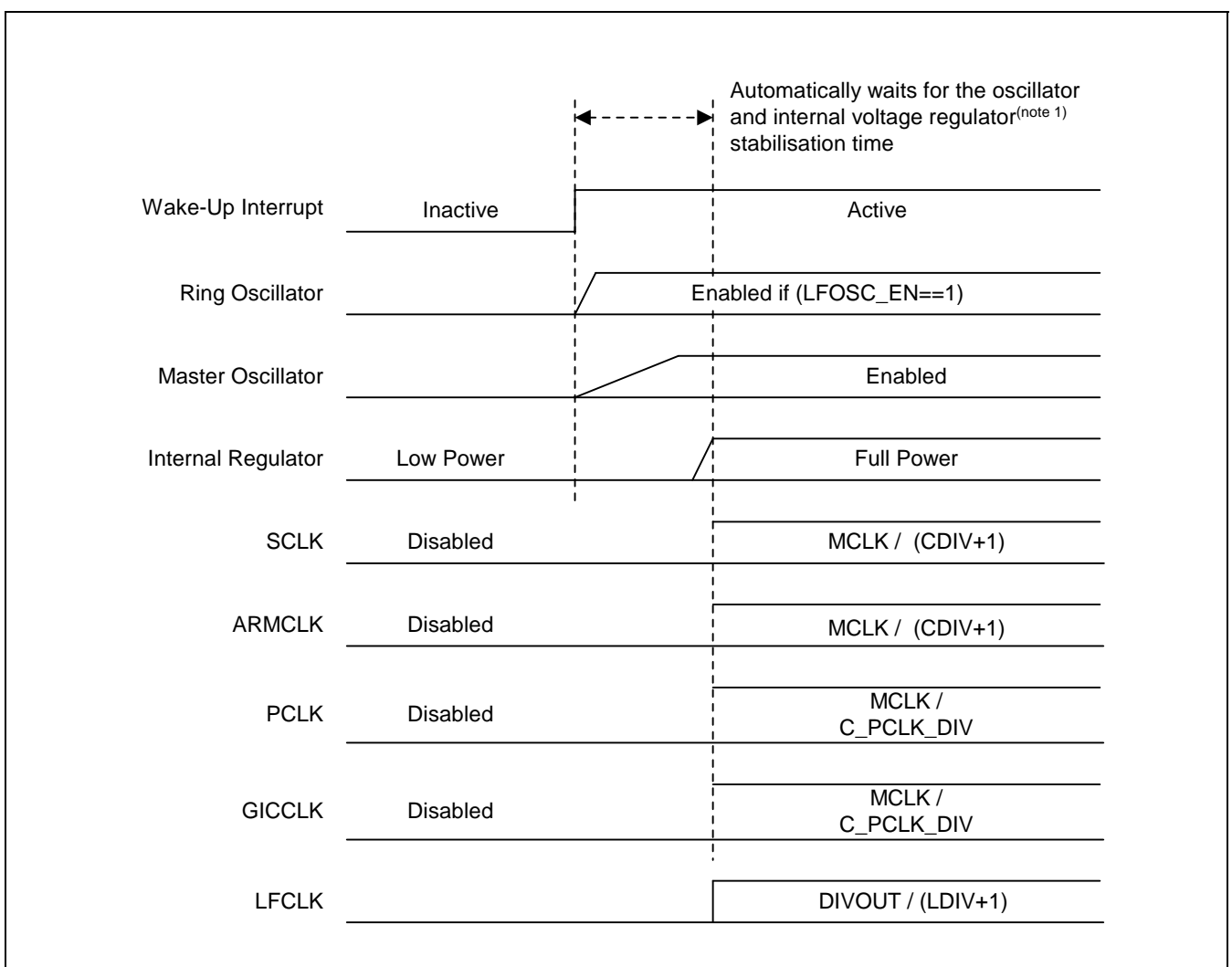


Figure 7-46. Transition Diagram: STOP Mode to NORMAL Mode (LFSEL=0)

NOTE: Internal voltage regulator stabilization time equals 64 MCLK periods. This duration is added to the master oscillator stabilization time.

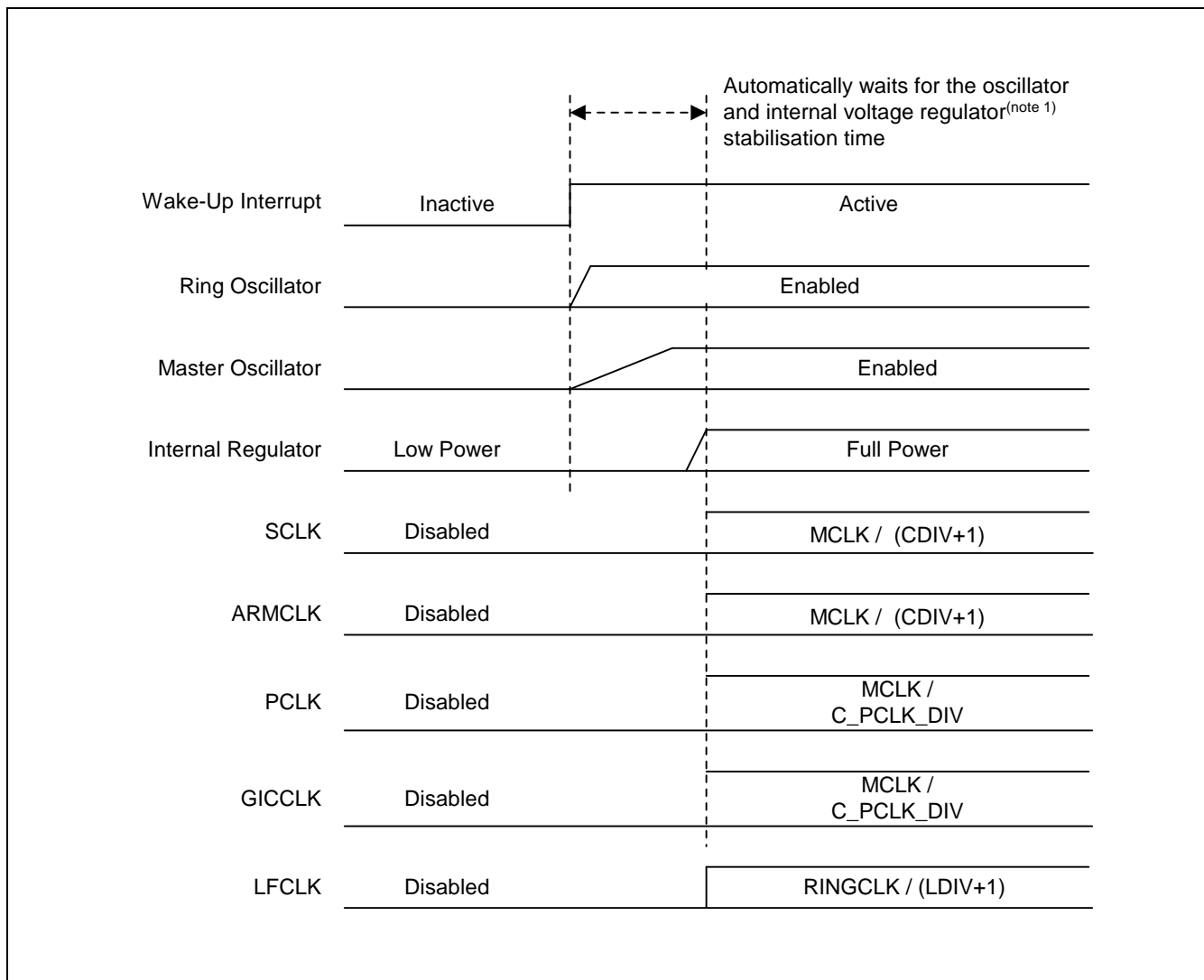


Figure 7-47. Transition Diagram: STOP Mode to NORMAL Mode (LFSEL=1)

NOTE: Internal voltage regulator stabilization time equals 64 MCLK periods. This duration is added to the master oscillator stabilization time.

2.1.3 Description of the Low Frequency Clock Transition

The user can change the low frequency clock source only in the NORMAL mode. The low frequency clock configuration is programmed in CM_LFOSCR register. After each reset, the low frequency oscillator is always enabled and DIVOUT is selected as the LFCLK. It is useful to confirm the use of the oscillator with LFSEL=1 or to disable it to use DIVOUT with LFOSCEN=0. When the user wants to enable and use the low frequency oscillator, it is necessary to program the stabilization time value (LF_ST bit in CM_LFOSCR register) and switch the register LFSEL and LFOSCEN bits to 1. The clock manager automatically waits the oscillator stabilization before propagating the low frequency clock.

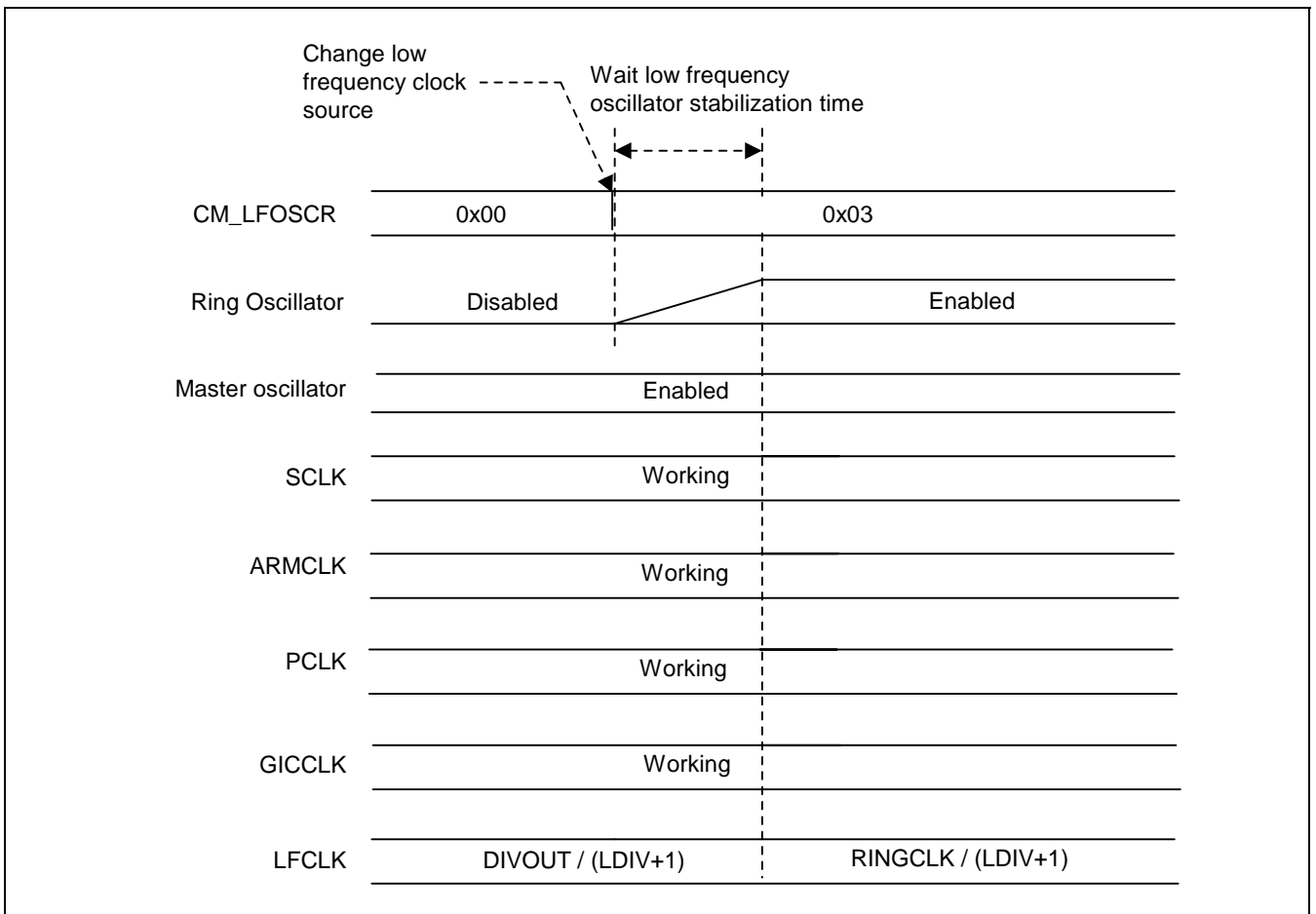


Figure 7-48. Low Frequency Transition: DIVOUT to RINGCLK

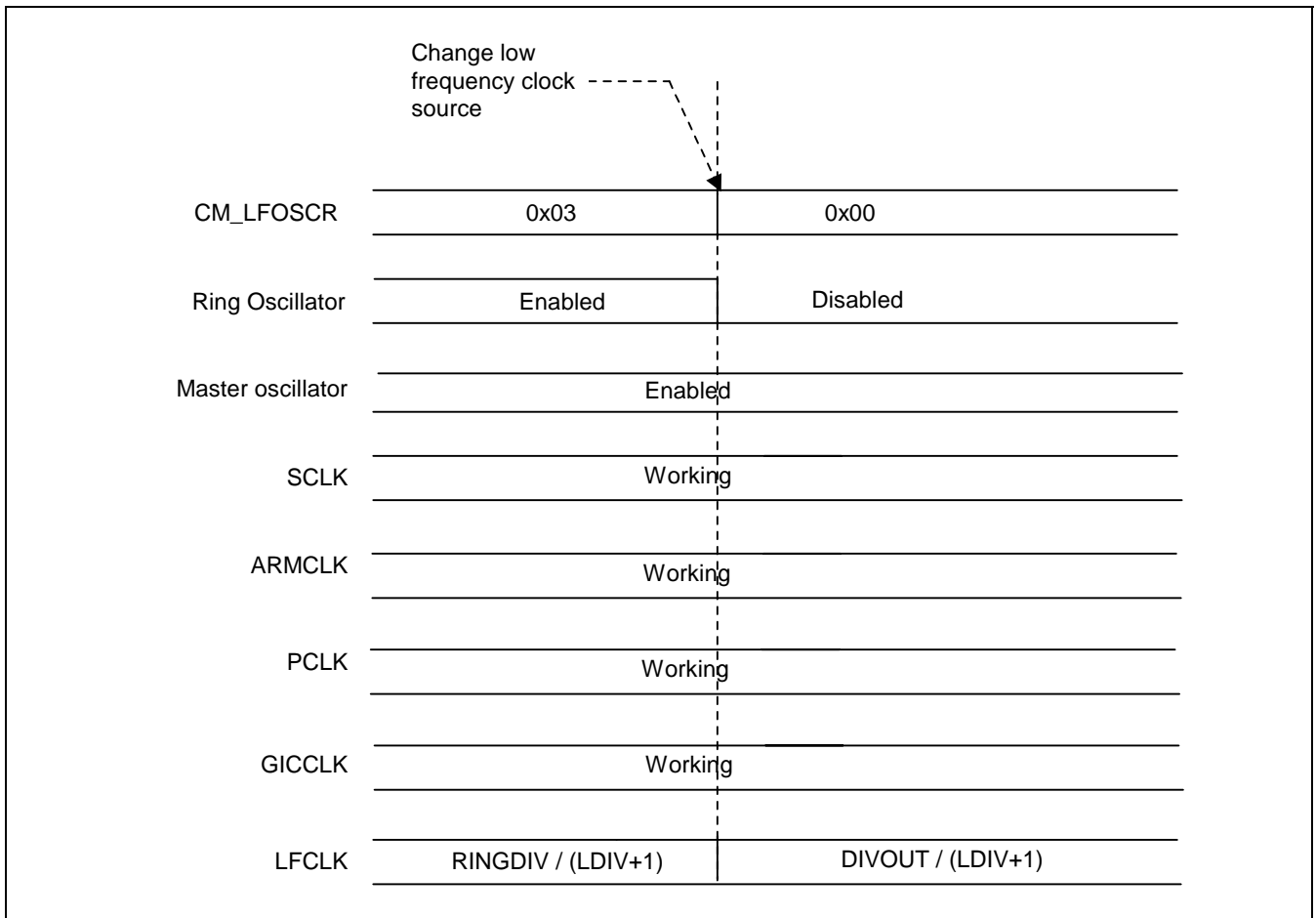


Figure 7-49. Low frequency transition: RINGCLK to DIVOUT

2.2 CLOCK AND POWER SUPPLY MONITORING

2.2.1 POWER Supply Monitoring (LVD)

The LVD provides an internal Power supply monitoring system.

This one is able to generate an interrupt or a reset according to the following condition :

- If power supply voltage fall under typ. 2.4V, then the LVD will generates reset. Reset condition will remain until power supply voltage rises over the preceding value.
- If power supply voltage fall under typ. 4.3V, then the LVD will generate an interrupt.

Both LVD reset and interrupts can be enabled/disabled separately by setting the appropriate values in dedicated register of IOCONF module.

2.2.2 Master Clock Monitoring

The clock monitor provides an internal master clock monitoring system.

This one is able to detect crystal oscillator failure, and generate a reset when this condition occurs. This feature can be enabled/disabled by setting the appropriate value in CM_MR register.

NOTES:

1. Clock-monitoring is enabled after reset.
2. The clock monitoring feature is available only if the internal ring oscillator is enabled (LFOSCEN bit set to 1 in CM_LFOCSR register)

2.3 RESET MANAGEMENT

2.3.1 General Description

The embedded reset controller is responsible for the management of the different reset sources available in the circuit.

The following table gives a list of those reset sources :

Table 7-5. Reset Sources

Name	Description
nRESET	This is the hardware reset (active low) given by external device on nRESET pin.
CMRST	This reset is generated internally by the clock-monitor feature. This feature can be enabled/disabled by software by setting the appropriate value in IOCONF module.
LVDRST	This reset is generated internally by the power supply monitor feature (LVD). This feature can be enabled/disabled by software by setting the appropriate value in IOCONF module.
WDRST	This reset is generated internally by the watchdog timer (WD). Please refer to the WD section for details on watchdog module.

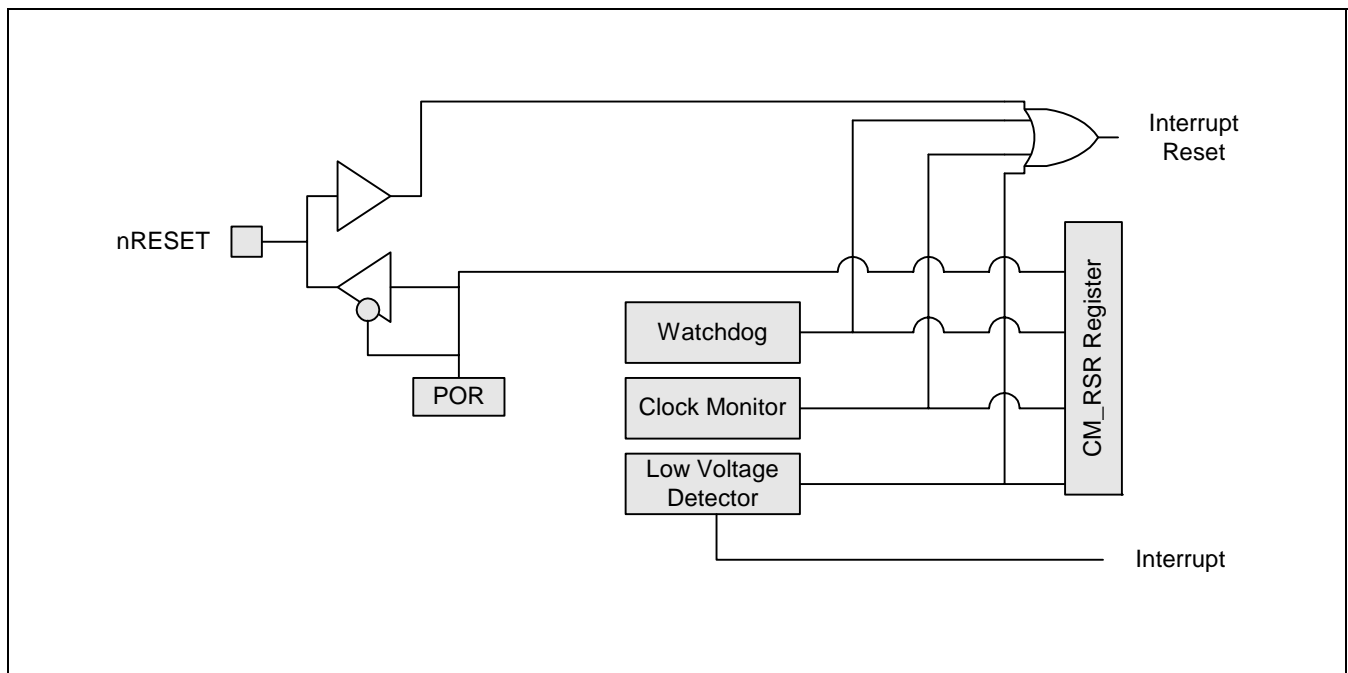


Figure 7-50. Reset Manager Diagram

The preceding drawing shows that each of the LVDRST, CMRST, WDRST has a corresponding status bit in CM_RSR register. This can be used by the software to detect the exact source of reset at boot time. (if all status bits are cleared, this means that external nRESET has been applied)

3. REGISTERS DESCRIPTION

Base Address – 0xFFFE8000

Table 7-6. Clock Manager Special Function Registers

Offset Address	Name	Description	R/W	Reset State
0x000	CM_STR	Oscillator and PLL status	R	0x00000006
0x004	–	Reserved	–	–
0x008	CM_WFIR	Master clock wait for interrupt	R/W	0x00000000
0x00C	CM_PSTR	PLL stabilization time	R/W	0x00000154
0x010	CM_PDPR	PLL divider parameters	R/W	0x00XXXXXX
0x014	CM_OSTR	Master oscillator stabilization time	R/W	0x00002EE0
0x018	–	Reserved	–	–
0x01C	CM_DIVBR	Main clock divider	R/W	0x00077777
0x020	CM_SELR	System clock selection	R/W	0x00000000
0x024	CM_RSR	Reset status	R	0x0000000X
0x028	CM_MDIVR	Master oscillator clock divider	R/W	0x000001FF
0x02C	CM_LFOSCR	Low frequency oscillator control	R/W	0x00000001
0x030	CM_CR	Control register	W	0x00000000
0x034 – 0x060	–	Reserved	–	–
0x064	CM_MR	Mode register	R/W	0x00000000
0x068	–	Reserved	–	–
0x06C	CM_CSR	Clear status register	W	–
0x070	CM_SR	Status register	R	0x00000000
0x074	CM_IER	Interrupt enable register	W	–
0x078	CM_IDR	Interrupt disable register	W	–
0x07C	CM_IMR	Interrupt mask register	R	0x00000000

CLKMNGR Oscillator and PLL Status Register CM_STR(0x000)

Access: Read only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
–	–	–	–	LFUSED	LFOST	OSCST	PLLST
R-0	R-0	R-0	R-0	R-0	R-1	R-1	R-0

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **PLLST : PLL stabilization status**

0: The PLL is disabled.

1: The PLL is enabled and stabilized.

- **OSCST : Master oscillator stabilization status**

This flag is set to one when the master oscillator counter has occurred at the configured value in the CM_OSTR register.

0: The master oscillator is disabled.

1: The master oscillator is enabled and stabilized.

- **LFOST : Low frequency oscillator stabilization status**

This flag is set to one when the low frequency oscillator counter has reached at the configured value in the CM_LFOSCR register.

0: The low frequency oscillator is disabled.

1: The low frequency oscillator is enabled and stabilized.

- **LFUSED : Low frequency oscillator use status**

This flag is set to one when the low frequency oscillator is stable and used as the low frequency clock.

0: The low frequency oscillator is unused.

1: The low frequency oscillator is stable and used.

CLKMNGR Wait for Interrupt

CM_WFIR(0x008)

Access: Read/Write

31	30	29	28	27	26	25	24
WFIKEY[15:8]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
WFIKEY [7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
–	–	PCLK1	Reserved	Reserved	Reserved	Reserved	Reserved
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **PCLK1 : Enable/Disable the PCLK1 clocks of peripheral clock in the HALT mode**

0: If the circuit is in HALT mode (HALTMODE bit written to 1 in CM_CR), and if the peripheral clock is active (see PMSR register of each peripheral), the peripheral clock is running.

1: If the circuit is in HALT mode (HALTMODE bit written to 1 in CM_CR), the peripheral clock is cut.

- **WFIKEY[15:0]**

Any write in the CM_WFIR register bits will only be effective if the WFIKEY field is equal to 0x80A4.

CLKMNGR PLL Stabilization Time			CM_PSTR(0x00C)				Access: Read/Write	
31	30	29	28	27	26	25	24	
PLLKEY[15:8]								
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
23	22	21	20	19	18	17	16	
PLLKEY[7:0]								
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
15	14	13	12	11	10	9	8	
–	–	–	–	–	PST[10:8]			
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	
7	6	5	4	3	2	1	0	
PST[7:0]								
R/W-0	R/W-1	R/W-0	R/W-1	R/W-0	R/W-1	R/W-0	R/W-0	

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

7-1. IMPORTANT NOTICE

This register will be writable access only when the PLL is disabled otherwise the register will be only readable.

- **PST[10:0] : PLL stabilization time**

PST register value = (PLL stabilization time / (MCLK period × 256)) – 5

MCLK is the master clock from the master oscillator. The default value is the maximum startup time 0x154. The stabilization time of PLL is maximum 150us.

- **PLLKEY[15:0] : Key for write access into the CM_PSTR register**

Any write in the CM_PSTR register bits will only be effective if the PLLKEY field is equal to 0x59C1.

CLKMNGR PLL Divider Parameters			CM_PDPR(0x010)			Access: Read /Write	
31	30	29	28	27	26	25	24
PDPKEY[15:8]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
PDPKEY[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
PLL_PRE[5:0]					PLL_POST[1:0]		
R/W-X	R/W-X	R/W-X	R/W-X	R/W-X	R/W-X	R/W-X	R/W-X
7	6	5	4	3	2	1	0
PMUL[7:0]							
R/W-X	R/W-X	R/W-X	R/W-X	R/W-X	R/W-X	R/W-X	R/W-X

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

7-2. IMPORTANT NOTICE

This register will be writable access only when the PLL is disabled otherwise the register will be readable.

- **PMUL[7:0] : PLL multiplier parameter**

These bits select the PLL multiplier. This value depends on the external crystal oscillator. It is not possible to write the zero value in this field.

In all case, the PLL output frequency must be between 12 to 40 MHz.

Table 7-7. Multiplier Parameter

PMUL[7:0]	PLL Multiplier $M = PMUL[7:0] + 8$
0	Unchanged
1	9
2	10
3	11
---	---
FF	263

- **PLL_POST[1:0] : Post scalar parameter**

This field configures the post scalar factor.

Table 7-8. Post Scalar Parameter

PLL_POST[1:0]	Post Divider Factor $S = 2^{\text{PLL_POST}[1:0]}$
0	1
1	2
2	4
3	8

- **PLL_PRE[5:0] : Pre divider parameter**

This parameter configures the pre divider.

Table 7-9. Pre Divider Parameter

PLL_PRE[5:0]	Pre Divider Factor $P = \text{PLL_PRE}[5:0] + 2$
0	2
1	3
2	4
3	5
---	---
3F	65

- **PDPKEY[15:0] : Key for write access into the CM_PDPR register**

Any write in the CM_PDPR register bits will only be effective if the PDPKEY field is equal to 0x7AB2.

7-3. IMPORTANT NOTICE

For PLL stability reasons, the PLLPRE/PMUL/PLLPOST values are strictly restricted to the range of values provided in following table. $F_{out} = (M \times F_{in}) / (P \times S)$

Table 7-10. PLLPRE/PMUL/PLLPOST Allowed Values

Fin (MHz)	Fout(MHz)	P	M	S	PLLPRE[5:0]	PMUL[7:0]	PLLPOST[1:0]
4	12	3	36	4	000001	00011100	10
	16	3	48	4	000001	00101000	10
	20	3	60	4	000001	00110100	10
	24	3	36	2	000001	00011100	01
	28	3	42	2	000001	00100010	01
	32	3	48	2	000001	00101000	01
	36	3	54	2	000001	00101110	01
	40	3	60	2	000001	00110100	01
6	12	3	24	4	000001	00010000	10
	18	3	36	4	000001	00011100	10
	24	3	24	2	000001	00010000	01
	30	3	30	2	000001	00010110	01
	36	3	36	2	000001	00011100	01
	40	3	40	2	000001	00100000	01

7-4. IMPORTANT NOTICE

The frequency of APB (PCLK) must be less than 20MHz.

CLKMNGR Oscillator Stabilization Time				CM_OSTR(0x014)				Access: Read /Write
31	30	29	28	27	26	25	24	
OSTKEY[15:8]								
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
23	22	21	20	19	18	17	16	
OSTKEY [7:0]								
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
15	14	13	12	11	10	9	8	
OST[15:8]								
R/W-0	R/W-0	R/W-1	R/W-0	R/W-1	R/W-1	R/W-1	R/W-0	
7	6	5	4	3	2	1	0	
OST[7:0]								
R/W-1	R/W-1	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **OST[15:0] : Master oscillator stabilization time**

This field is the number of low frequency oscillator cycle for respect the master oscillator stabilization time. During reset, this register will be set at 0x2EE0 as default value (worst case).

Master Oscillator stabilization time = OST[15:0] * LFCLK period

- **OSTKEY[15:0] : Key for write access into the CM_OSTR register**

Any write in the CM_OSTR register bits will only be effective if the OSTKEY field is equal to 0xFA4B.

CLKMNGR Master Clock Divider **CM_DIVBR(0x01C)** **Access: Read /Write**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	Reserved[2:0]		
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1	R/W-1
15	14	13	12	11	10	9	8
–	Reserved[2:0]			–	Reserved[2:0]		
R/W-0	R/W-1	R/W-1	R/W-1	R/W-0	R/W-1	R/W-1	R/W-1
7	6	5	4	3	2	1	0
–	Reserved[2:0]			–	PCLK1DIV[2:0]		
R/W-0	R/W-1	R/W-1	R/W-1	R/W-0	R/W-1	R/W-1	R/W-1

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

7-5. IMPORTANT NOTICE

The division ratio is similar (same programmable field) for the external clocks and for the gated external clocks used for internal domain.

- PCLK1DIV[2:0] : Peripheral divider**

This field selects the division ratio between the CORECLK system clock and the peripheral clock domain.

Table 7-11. Peripheral Divider

PCLK1DIV[2:0]	Division Ratio Between CORECLK and Peripheral Clock
000	1
001	2
010	3
---	---
111	8

7-6. IMPORTANT NOTICE

1. When Peripheral clock(PCLK) is lower than the System clock(SCLK) and the ARM instruction follows a peripheral write access instruction, the next ARM instruction can be executed before the end of the peripheral write access.
2. Software have to ensure that this is not critical for this application or it should add a dummy peripheral read access before executing the ARM instruction.
3. For example, a peripheral write access is attempted to unlock the flash write protection. The following diagrams show the arm state when an ARM instruction follows the peripheral write access instruction and when a peripheral 'dummy' access instruction follows the peripheral write access instruction. In this example, PCLK = SCLK/4.

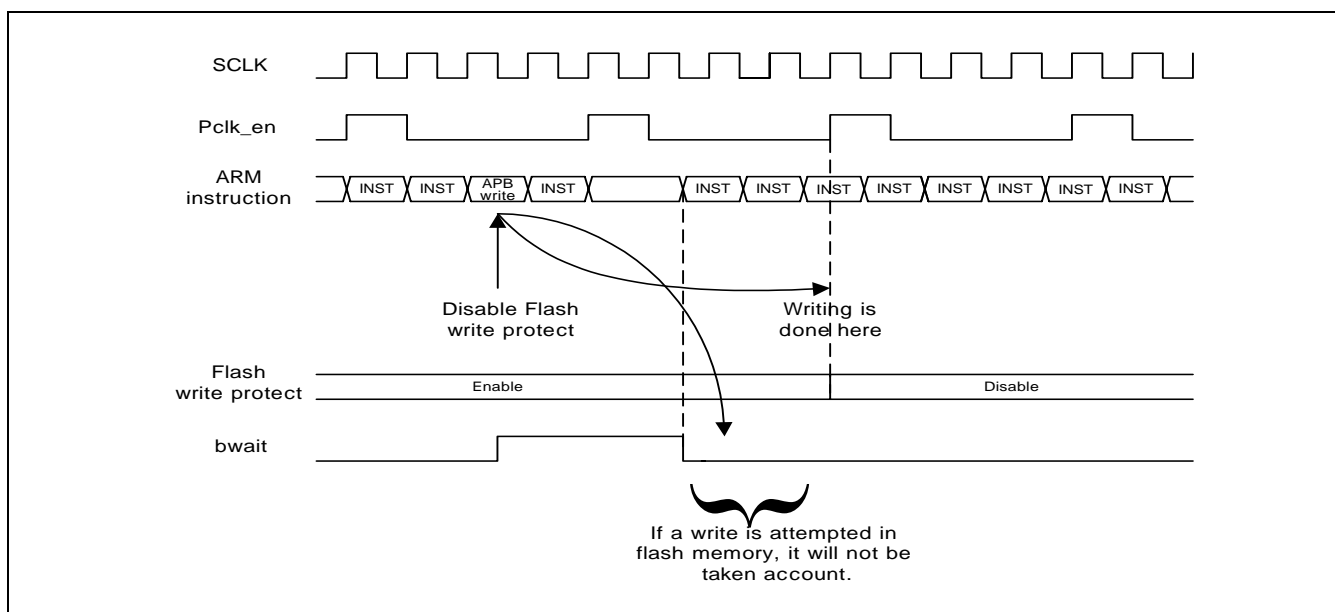


Figure 7-51. ARM Instruction Follows a Peripheral Write Access Instruction

The following example shows the arm state when a peripheral access instruction follows a peripheral write access instruction.

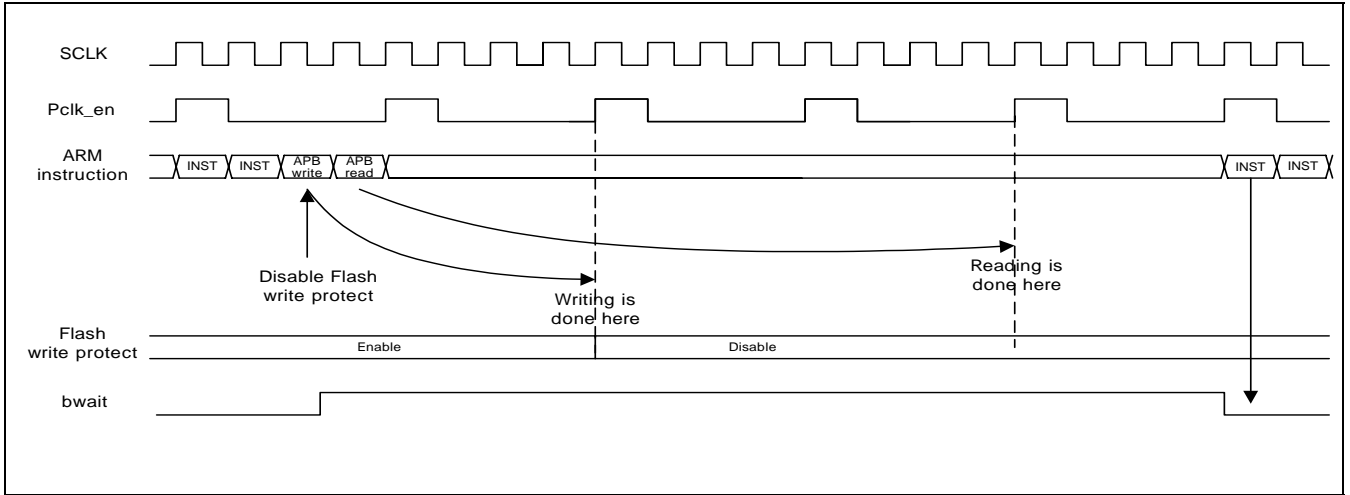


Figure 7-52. Peripheral Access Follows a Peripheral Write Access Instruction

CLKMNGR System Clock Selection **CM_SELR(0x020)** **Access: Read /Write**

31	30	29	28	27	26	25	24
SELKEY[15:8]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
SELKEY[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	CMCLK_SEL[1:0]	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

7-7. IMPORTANT NOTICE

The SLOW mode can not be used when there is not low frequency oscillator. It is equivalent to LOW POWER mode.

When the HALTMODE bit is set to 1, the clock manager will switch to the HALT mode (i.e. to LOW POWER mode).

Do not forget to set the correct value of master oscillator stabilization time before exiting LOW POWER mode.

- **CMCLK_SEL[1:0] : Select between different clocks**

This field selects the clock manager mode.

Table 7-12. Clock Manager Modes

CMCLK_SEL[1:0]	System Clock From	Description
00	Master oscillator or external clock	Normal mode
01	PLL output (multiplication of master or external clock)	High speed mode
10	Master or external clock divided by (MDIV+1)	Slow mode
11	Low frequency oscillator or master clock divided by (MDIV+1)	Low power mode

7-8. IMPORTANT NOTICE

The end of the mode transition is indicated with the STABLE interrupt (see CM_SR register) which goes to high level when the system clock frequency is stabilized and provided to the circuit.

- **SELKEY[15:0] : Key for write access into the CM_SELR register**

Any write in the CM_SELR register bits will only be effective if the SELKEY field is equal to 0xD0C9.

CLKMNGR Reset Status

CM_RSR(0x024)

Access: Read only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
–	–	–	–	–	LVD	CM	WD
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **WD : Internal reset from watchdog**

0: The last reset is not caused by the watchdog module.

1: The last reset is caused by the watchdog module.

- **CM : Internal reset from Clock Monitor**

0: The last reset is not caused by the Clock Monitor module.

1: The last reset is caused by the Clock Monitor module.

- **LVD : Internal reset from LVD**

0: The last reset is not caused by the LVD module.

1: The last reset is caused by the LVD module.

CLKMNGR Master Oscillator Clock Divider CM_MDIVR(0x028) Access: Read/Write

31	30	29	28	27	26	25	24
MDIVKEY[15:8]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
MDIVKEY[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
LDIV[2:0]			CDIV[2:0]			-	MDIV[8]
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1
7	6	5	4	3	2	1	0
MDIV[7:0]							
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

7-9. IMPORTANT NOTICE

In any mode, the divided master clock frequency must always be greater than low frequency oscillator.

• **MDIV[8:0] : Master clock divider**

This field is used to divide the master clock in order to generate the low frequency clock. The MDIV=0 value is no effect i.e. the previous value is unchanged.

7-10. IMPORTANT NOTICE

The MDIV can be changed only if LSEL=0 (CM_LFOSCR register) and the clock manager is not in LOW POWER mode or if LFSEL=1 (CM_LFOSCR register) and the clock manager is not in SLOW mode.

• **CDIV[2:0] : Core clock divider**

This field is used to divide the PLLOUT or MCLK clocks.

Table 7-13. Core-clock Divider

CDIV[2:0]	Frequency division ratio between PLLOUT(in high speed mode) or MCLK(in normal mode) and CORECLK clock
000	1
001	2
010	3
- - -	- - -
111	8

- **LDIV[2:0] : Low frequency oscillator clock divider**

This field is used to divide the ring oscillator clock frequency in order to have a real frequency clock when ring oscillator is used for LFDIV generation.

Table 7-14. Low Frequency Oscillator Clock Divider

LDIV[2:0]	Division Ratio Between RINGCLK or DIVOUT and LFDIV clock
000	1
001	2
010	3
---	---
111	8

7-11. IMPORTANT NOTICE

The LDIV can be changed if the clock manager is not in LOW POWER mode.

- **MDIVKEY[15:0] : Key for write access into the CM_MDIVR register**

Any write in the CM_MDIVR register bits will only be effective if the MDIVKEY field is equal to 0xACDC.

CLKMNGR Low Frequency Oscillator Control CM_LFOSCR(0x02C)**Access: Read/Write**

31	30	29	28	27	26	25	24
LFOSCKEY[15:8]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
LFOSCKEY[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
LF_ST[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
-	-	-	-	-	-	LFSEL	LFOSCEN
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

7-12. IMPORTANT NOTICE

- .7 - .2 are reserved bits and must be set to '0'.
- The low frequency configuration must be set after every reset. These bits can be changed only in the NORMAL mode.

- LFOSCEN : Enable/Disable the low frequency oscillator**

- 0: Disable the low frequency oscillator
1: Enable the low frequency oscillator

7-13. IMPORTANT NOTICE

After enabling the low frequency oscillator, software has to ensure that the LFUSED bit in CM_STR register is set to 1.

- LFSEL : Low frequency clock selection**

- 0: Select the master clock divided by MDIV+1 as low frequency clock.
1: Select the low frequency oscillator clock as low frequency clock.

- LF_ST[7:0] : Low frequency stabilization time**

This value is the number of master clock cycles counted to wait the low frequency stabilization. When this value is reached, the LFOST bit in CM_STR is set.

$$LF_ST[7:0] = (LF_{st} / (MCLK \text{ period} \times 128)) - 3$$

- LFOSCKEY[15:0] : Key for write access into the CM_LFOSCR register**

Any write in the CM_LFOSCR register bits will only be effective if the LFOSCKEY field is equal to 0xA34C.

CLKMNGR Control Register **CM_CR(0x030)** **Access: Write only**

31	30	29	28	27	26	25	24
CRKEY[15:8]							
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
23	22	21	20	19	18	17	16
CRKEY[7:0]							
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
7	6	5	4	3	2	1	0
–	–	IDLEMODE	STOPMODE	–	–	–	HALTMODE
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- HALTMODE : Stop ARM clock. This bit is set only, the circuit will come out of HALT mode upon interrupt.**
 - 0: No effect
 - 1: Enter into HALT mode
- STOPMODE : Stop all clocks. The circuit resumes form STOPMODE mode upon external wake-up interrupt.**
 - 0: No effect
 - 1: Enter into STOP mode
- IDLEMODE : Stop ARM clock. All other clocks are left unchanged. This bit is set only, the circuit will come out of IDLE mode upon interrupt.**
 - 0: No effect
 - 1: Enter into IDLE mode
- CRKEY[15:0] : Key for write access into the CM_CR register**

Any write in the CM_CR register bits will only be effective if the CRKEY field is equal to 0x678F.

7-14. IMPORTANT NOTICE

This STOPMODE bit must be set only in Normal mode. In other words, STOP mode can be entered only in the normal mode.

SPEEDMODE of IFC_MR must be set to 0 before entering stop mode because of current consumption.

There are 4 external interrupts for wake up source from STOP mode.

The LDMA operation related to flash memory must be finished before entering HALT mode.

CLKMNGR Mode Register				CM_MR(0x064)				Access: Write
31	30	29	28	27	26	25	24	
MRKEY[15:8]								
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	
23	22	21	20	19	18	17	16	
MRKEY[7:0]								
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	
15	14	13	12	11	10	9	8	
–	–	–	–	–	–	–	–	
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	
7	6	5	4	3	2	1	0	
–	–	–	–	–	–	–	CM_EN	
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **CM_EN : Clock Monitor Enable.**

0: Disable clock monitor.

1: Enable clock monitor.

- **MRKEY[15:0] : Key for write access into the CM_MR register**

Any write in the CM_MR register bits will only be effective if the MRKEY field is equal to 0x1505.

CM Clear Status Register

CM_CSR (0x06C)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	STABLE
W	W	W	W	W	W	W	W

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **STABLE** : Clear stable interrupt

0: No effect.

1: Clear Stable interrupt.

CM Status Register **CM_SR (0x70)** **Access: Read only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	STABLE
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: After reset**-1: After reset**-U: Undefined after reset*

- **STABLE : Stable interrupt**

0: The requested clock frequency is not yet available. The master oscillator or PLL is not stabilized.

1: The master oscillator and/or PLL are stabilized. The requested clock frequency is available.

CM Interrupt Enable Register

CM_IER (0x074)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	STABLE
W	W	W	W	W	W	W	W

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **STABLE** : Stable interrupt enable

0: No effect

1: Enable STABLE interrupt

CM Interrupt Disable Register

CM_IDR (0x078)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	STABLE
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- STABLE : Stable interrupt disable**

0: No effect

1: Disable STABLE interrupt

CM Interrupt Mask Register

CM_IMR (0x07C)

Access: Read only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	STABLE
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **STABLE : Stable interrupt mask**

0: STABLE interrupt is disabled.

1: STABLE interrupt is enabled.

8

CONTROLLER AREA NETWORK (CANB)

1. OVERVIEW

1.1 FUNCTIONAL DESCRIPTION

The CAN module consists of the components CAN Core, Message RAM, Message Handler, Control Registers and Module Interface.

The CAN Core performs communication according to the CAN protocol version 2.0 part A and B. The bit rate can be programmed to values up to 1MBit/s depending on the used technology. For the connection to the physical layer additional transceiver hardware is required.

For communication on a CAN network, individual Message Objects are configured. The Message Objects and Identifier Masks for acceptance filtering of received messages are stored in the Message RAM.

All functions concerning the handling of messages are implemented in the Message Handler. Those functions are the acceptance filtering, the transfer of messages between the CAN Core and the Message RAM, and the handling of transmission requests as well as the generation of the module interrupt.

The CAN implements the following features:

- Supports CAN protocol version 2.0 part A and B
- Bit rates up to 1 MBit/s
- 32 Message Objects
- Each Message Object has its own identifier mask
- Programmable FIFO mode (concatenation of Message Objects)
- Maskable interrupt
- Disabled Automatic Retransmission mode for Time Triggered CAN applications
- Programmable loop-back mode for self-test operation
- Power management block and wake up mode allowing optimization of power consumption
- CAN_TX output pin configurable in open drain allowing connection without external transceiver

1.2 BLOCK DIAGRAM

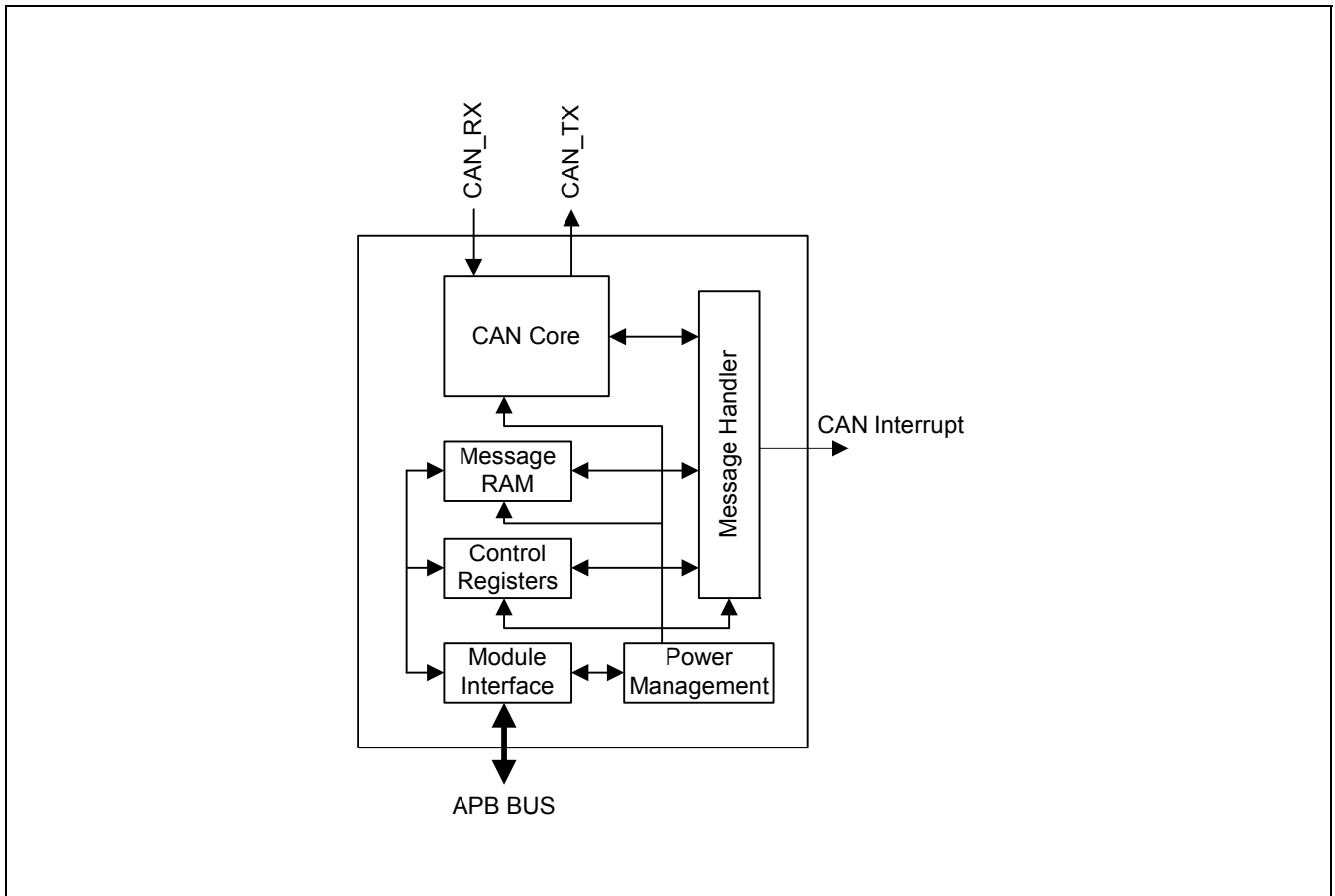


Figure 8-1. CAN Block Diagram

2. INTERFACE DESCRIPTION

2.1 EXTERNAL PIN DESCRIPTION

Table 8-1. CAN Pin Description

Pin Name	Function	I/O Type	Active Level	Comments
CANTXD[3:0]	Transmit line	O	Low	–
CANRXD[3:0]	Receive line	I	Low	–

3. FUNCTIONAL OPERATION

3.1 CAN PROTOCOL

3.1.1 Introduction

The Controller Area Network (CAN) is a serial communication protocol which efficiently supports distributed real-time control with a very high level of security.

Its domain of application ranges from high speed networks to low-cost multiplex wiring. For example, in automotive, electronics, engine control units, sensors, anti-skid systems, etc. may be connected using CAN with bit rates up to 1 Mbit/s. At the same time, it is cost effective to build into vehicle body electronics such as lamp clusters, electric windows, etc... to replace the wiring harness otherwise required.

The intention of this specification is to achieve compatibility between any two CAN implementations. Compatibility, however, has different aspects regarding e.g. electrical features and the interpretation of data to be transferred.

To achieve design transparency and implementation flexibility, CAN has been subdivided into different layers according to the ISO/OSI reference model :

- The Data Link Layer
 - The Logical Link Control (LLC) sub-layer
 - The Medium Access Control (MAC) sub-layer
- The Physical Layer

NOTE: In previous versions of CAN specification, services and functions of the LLC and MAC sub-layers of the Data Link Layer were described in layers denoted as 'object layer' and 'transfer layer'.

The scope of the LLC sub-layer includes : determining which messages received by the LLC sub-layer are actually to be accepted ; providing services for data transfer and for remote data request ; and providing the means for recovery management and overload notifications. There is much freedom in defining object handling.

The scope of the MAC sub-layer mainly is the transfer protocol, i.e. controlling the framing, performing arbitration, error checking, error and fault confinement. Within the MAC sub-layer it is decided whether the bus is free for starting signaling a new transmission or whether a reception is just starting. Also some general features of the bit timing are regarded as part of the MAC sub-layer. It is in the nature of the MAC sub-layer that there is no freedom for modifications.

The scope of the physical layer is the actual transfer of the bits between the different nodes with respect to all electrical properties. Within a network the physical layer, of course, has to be the same for all nodes. There are, however, many possible implementations of physical layer.

The scope of this specification is to define the MAC sub-layer and a small part of the LLC sub-layer of the Data Link Layer and to describe the consequences of the CAN protocol on the surrounding layers.

3.1.2 Basic Concepts

3.1.2.1 CAN Properties

CAN has the following properties:

- Prioritization of messages,
- Guarantee of latency times,
- Configuration flexibility,
- Multicast reception with time synchronization,
- System wide data consistency,
- Multi-master,
- Error detection and error signaling,
- Automatic retransmission of corrupted messages as soon as the bus is idle again,
- Distinction between temporary errors and permanent failures of nodes and autonomous switching off of defect nodes.

3.1.2.2 Layered Structure of a CAN Node

Layered architecture of CAN according to the ISO/OSI reference model :

- The LLC sub-layer is concerned with message filtering, overload notification and recovery management.
- The MAC sub-layer represents the kernel of the CAN protocol. It presents messages received from the LLC sub-layer and accepts messages to be transmitted to the LLC sub-layer. The MAC sub-layer is responsible for Message Framing, Arbitration, Acknowledgement, Error Detection and Signaling. The MAC sub-layer are supervised by a management entity called Fault Confinement which is self-checking mechanism for distinguishing short disturbances from permanent failures.
- The physical layer defines how signals are actually transmitted and therefore deals with the description of bit timing, bit encoding, and synchronization. Within this specification the physical layer is not defined, as it will vary according to the requirements of individual applications (for example, transmission medium and signal level implementations).

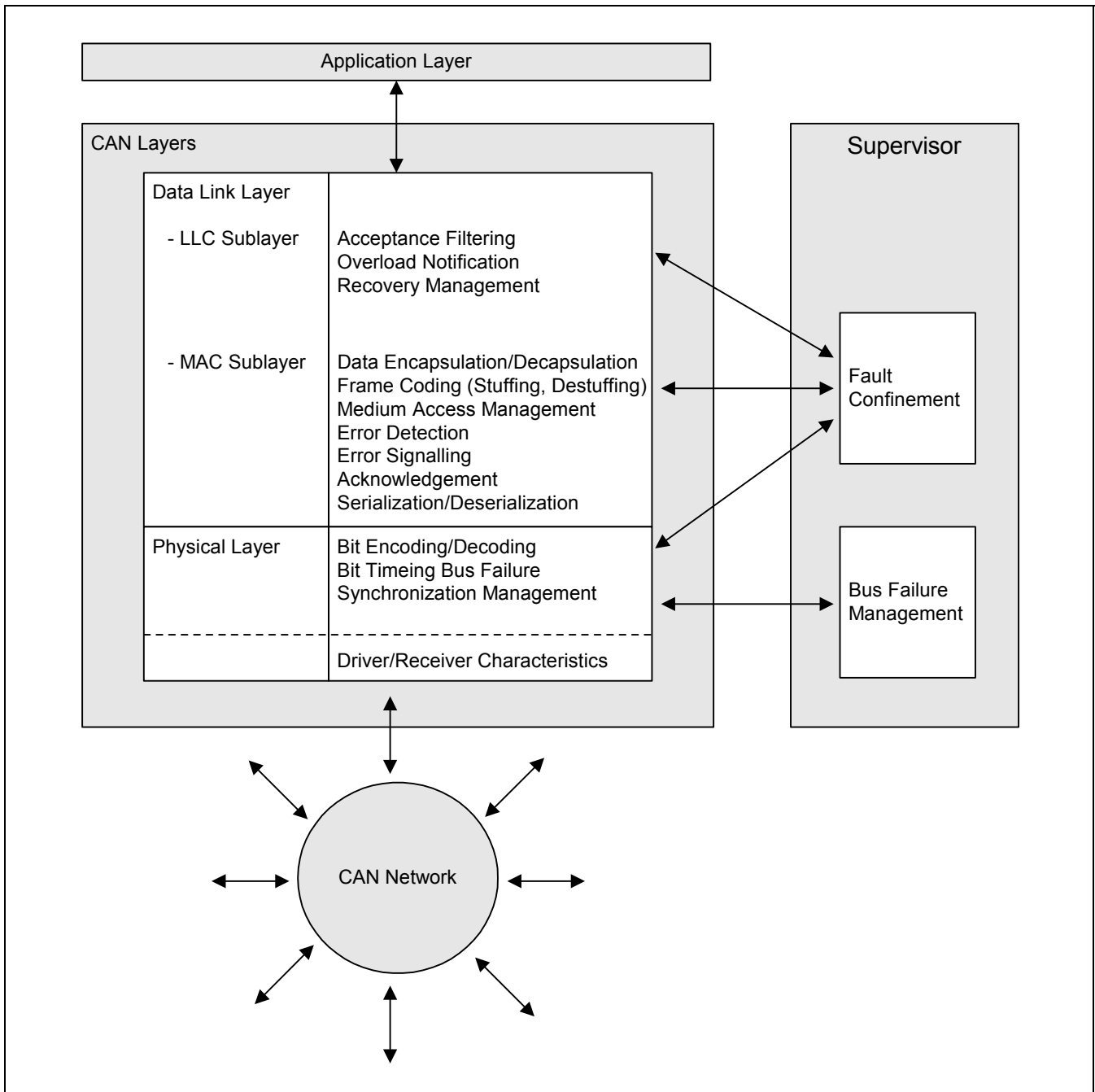


Figure 8-2. CAN Layers Description

LLC = Logical Link Control
 MAC = Medium Access Control

The scope of this specification is to define the Data Link Layer and the consequences of the CAN protocol on the surrounding layers.

3.1.2.3 Messages

Information on the bus is sent in fixed format messages of different but limited length (see section “Message transfer” below). When the bus is free, any connected unit may start to transmit a new message.

3.1.2.4 Information Routing

In CAN systems, a CAN node does not make use of any information about the system configuration (e.g. node addresses). This has several important consequences:

- System flexibility : nodes can be added to the CAN network without requiring any change in the software or hardware of any node and application layer.
- Message routing : the content of a message is named by an identifier. The identifier does not indicate the destination of the message, but describes the meaning of the data, so that all nodes in the network are able to decide by Message Filtering whether the data is to be acted upon by them or not.
- Multicast : as a consequence of the concept of Message Filtering any number of nodes can receive and simultaneously act upon the same message.
- Data consistency : within a CAN network it is guaranteed that a message is simultaneously accepted either by all nodes or by no node. Thus data consistency of a system is achieved by the concepts of multicast and by error handling.

3.1.2.5 Bit Rate

The speed of CAN may be different in different systems. However, in a given system the bit-rate is uniform and fixed.

3.1.2.6 Priorities

The identifier defines a static message priority during bus access.

3.1.2.7 Remote Data Request

By sending a Remote Frame a node requiring data may request another node to send the corresponding Data Frame. The Data Frame and the corresponding Remote Frame are named by the same Identifier.

3.1.2.8 Multi-Master

When the bus is free any unit may start to transmit a message. The unit with the message of highest priority to be transmitted gains bus access.

3.1.2.9 Arbitration

Whenever the bus is free, any unit may start to transmit a message. If two or more units start transmitting messages at the same time, the bus access conflict is resolved by bit-wise arbitration using the Identifier. The mechanism of arbitration guarantees that neither information nor time is lost. If a Data Frame and a Remote Frame with the same Identifier are initiated at the same time, the Data Frame prevails over the Remote Frame. During arbitration every transmitter compares the level of the bit transmitted with the level that is monitored on the bus. If these levels are equal, the unit may continue to send. When a 'recessive' level is sent and a 'dominant' level is monitored (see Bus Values), the unit has lost arbitration and must withdraw without sending one more bit.

3.1.2.10 Safety

In order to achieve the utmost safety of data transfer, powerful measures for error detection, signalling and self-checking are implemented in every CAN node.

3.1.2.10.1 Error Detection

For detecting errors the following measures have been taken:

- Monitoring (transmitters compare the bit levels to be transmitted with the bit levels detected on bus).
- Cyclic Redundancy Check (CRC).
- Bit stuffing.
- Message frame check.

3.1.2.10.2 Performance of Error Detection

The error detection mechanisms have the following properties:

- All global errors are detected (by means of monitoring).
- All local errors at transmitters are detected (by means of monitoring).
- Up to 5 randomly distributed errors in a message are detected (by means of CRC).
- Burst errors of length less than 15 in a message are detected (by means of CRC).
- Errors of any odd number in a message are detected (by means of CRC).

Total residual error probability for undetected corrupted messages is less than: message error rate $\times 4.7 \times 10^{-11}$.

3.1.2.11 Error Signaling and Recovery Time

Corrupted messages are flagged by any node detecting an error. Such messages are aborted and will be retransmitted automatically. The recovery time from detecting an error until the start of the next message is at most 31 bit times, if there is no further error.

3.1.2.12 Fault Confinement

CAN nodes are able to distinguish short disturbances from permanent failures. Defective nodes are switched off.

3.1.2.13 Connections

The CAN serial communication link is a bus to which a number of units may be connected. This number has no theoretical limit. Practically the total number of units will be limited by delay times and/or electrical loads on the bus line.

3.1.2.14 Single Channel

The bus consists of a single bi-directional channel that carries bits. From this data resynchronization information can be derived. The way in which this channel is implemented is not fixed in this specification. E.g. single wire (plus ground), two differential wires, optical fibers, etc.

3.1.2.15 Bus Values

The bus can have one of two complementary logical values: 'dominant' or 'recessive'. During simultaneous transmission of 'dominant' and 'recessive' bits, the resulting bus value will be 'dominant'. For example, in case of a wired-AND implementation of the bus, the 'dominant' level would be represented by a logical '0' and the 'recessive' level by a logical '1'. Physical states (e.g. electrical voltage, light) that represent the logical levels are not given in this specification.

3.1.2.16 Acknowledgment

All receivers check the consistency of the message being received and will acknowledge a consistent message and flag an inconsistent message.

3.1.2.17 Sleep Mode / Wake-up

To reduce the system's power consumption, a CAN-device may be set into sleep mode without any internal activity and with disconnected bus drivers. The sleep mode is finished with a wake-up by any bus activity or by internal conditions of the system. On wake-up, the internal activity is restarted, although the MAC sublayer will be waiting for the system's oscillator to stabilize and it will then wait until it has synchronized itself to the bus activity (by checking for eleven consecutive 'recessive' bits), before the bus drivers are set to "on-bus" again.

3.1.3 Message Transfer

3.1.3.1 Definition of Transmitter/Receiver

A node originating a message is called transmitter of that message. The node stays transmitter until the bus is idle or the node loses arbitration. A node is called receiver of a message, if it is not transmitter of that message and the bus is not idle.

3.1.3.2 Frame Formats

There are two different formats which differ in the length of the identifier field:

Table 8-2. Identifier Length Within Standard and Extended Frame

Frame Format	Identifier Length
Standard frame	11 bit
Extended frame	29 bit

3.1.3.3 Frame Types

Message transfer is manifested and controlled by four different frame types :

- A data frame carries data from a transmitter to the receivers.
- A remote frame is transmitted by a bus unit to request the transmission of the data frame with the same identifier.
- An error frame is transmitted by any unit on detecting a bus error.
- An overload frame is used to provide for an extra delay between the preceding and the succeeding data or remote frames.

Data frames and remote frames can be used both in standard frame format and extended frame format. They are separated from preceding frames by an Interframe space.

3.1.3.3.1 Data frame

A data frame is composed of seven different bit fields :

- Start Of Frame (SOF),
- Arbitration field,
- Control field,
- Data field,
- CRC field,
- ACK field,
- End Of Frame (EOF).

The data field can be of length zero.

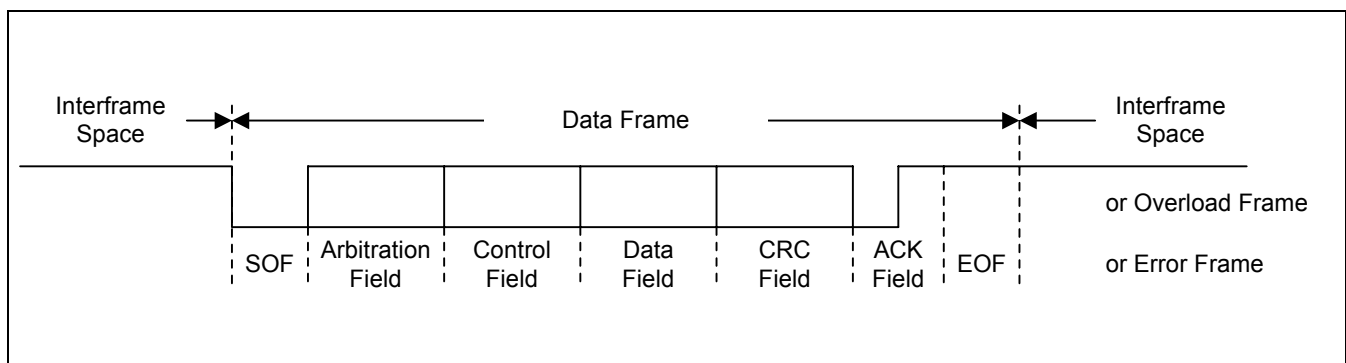


Figure 8-3. Data Frame

3.1.3.3.1.1 Start of frame (standard and extended format)

The Start of Frame (SOF) marks the beginning of Data Frames and Remote Frames. It consists of a single 'dominant' bit.

A node is only allowed to start transmission when the bus is idle (see 'Interframe Spacing'). All nodes have to synchronize to the leading edge caused by Start of Frame (see 'Hard Synchronization') of the node starting transmission first.

3.1.3.3.1.2 Arbitration field

The format of the Arbitration Field is different for standard format and extended format frames.

- In standard format the Arbitration Field consists of the 11 bit identifier and the RTR bit. The identifier bits are denoted ID[28:18].

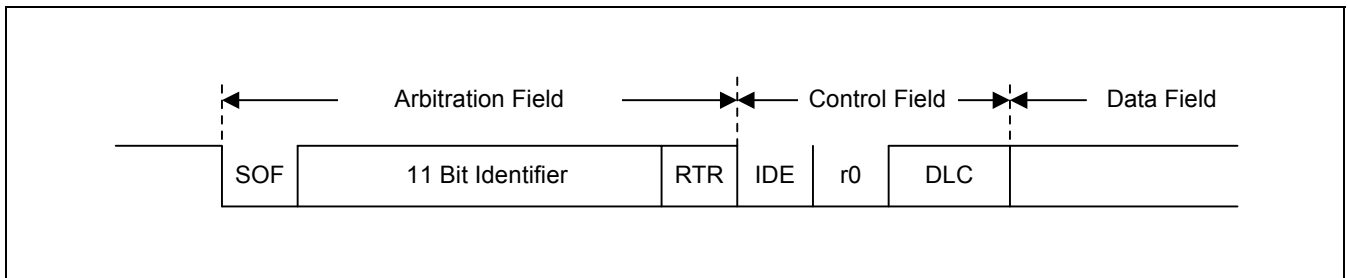


Figure 8-4. Standard Format

- In Extended Format the Arbitration Field consists of the 29 bit Identifier, the SRR bit, the IDE bit, and the RTR bit. The Identifier bits are denoted ID[28:0].

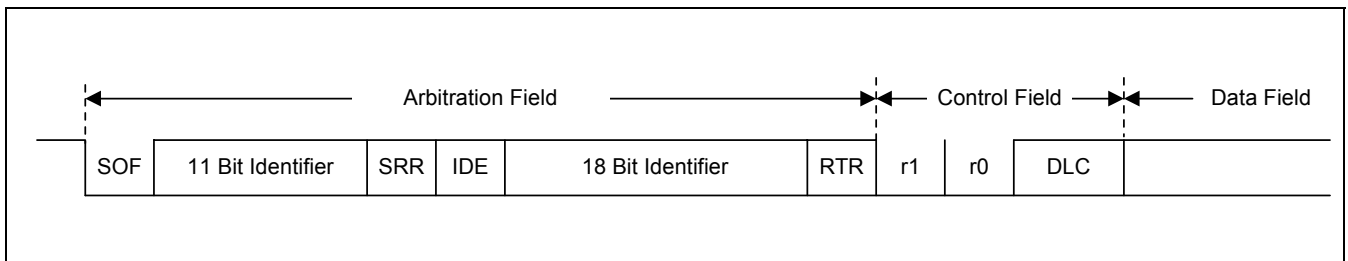


Figure 8-5. Extended Format

In order to distinguish between Standard Format and Extended Format the reserved bit r1 in previous CAN specifications version 1.0-1.2 now is denoted as IDE bit.

— Identifier

IDENTIFIER - Standard Format

The IDENTIFIER's length is 11 bits and corresponds to the base ID in Extended Format. These bits are transmitted in the order from ID28 to ID18. The least significant bit is ID18. The 7 most significant bits ID[28:22] must not be all 'recessive'.

IDENTIFIER - Extended Format

In contrast to the Standard Format the Extended Format consists of 29 bits. The format comprises two sections: Base ID with 11 bits and the Extended ID with 18 bits.

⇒ *Base ID*: the Base ID consists of 11 bits. It is transmitted in the order from ID28 to ID18. It is equivalent to format of the Standard Identifier. The base ID defines the Extended Frame's base priority.

⇒ *Extended ID*: the Extended ID consists of 18 bits. It is transmitted in the order of ID17 to ID0.

In a Standard Frame the IDENTIFIER is followed by the RTR bit.

— RTR bit (Standard and Extended Format)

Find in the table below the RTR bit logical value according to frame type:

Table 8-3. RTR Bit

Frame Type	RTR Bit Logical Value
Data frame	'dominant'
Remote frame	'recessive'

In an Extended Frame the base ID is transmitted first, followed by the IDE bit and the SRR bit. The extended ID is transmitted after the SRR bit.

— SRR bit

The SRR bit (Substitute Remote Request) is a recessive bit. It is transmitted in Extended Frames at the position of the RTR bit in Standard Frames and so substitutes the RTR bit in the Standard Frame.

Therefore, collisions of a Standard Frame and an Extended Frame, the base ID (see 'Extended IDENTIFIER' below) of which is the same as the Standard Frame's Identifier, are resolved in such a way that the Standard Frame prevails the Extended Frame.

— IDE bit (Extended Format)

Find in the table below the IDE bit logical value and membership according to frame format type :

Table 8-4. IDE Bit

Frame Format Type	IDE Bit Belongs to	IDE Bit Logical Value
Standard frame format	the control field	'dominant'
Extended frame format	the arbitration field	'recessive'

3.1.3.3.1.3 Control field (Standard and Extended Format)

The CONTROL FIELD consists of six bits. The format of the CONTROL FIELD is different for Standard Format and Extended Format. Frames in Standard Format include the DATA LENGTH CODE, the IDE bit, which is transmitted 'dominant' (see above), and the reserved bit r0. Frames in Extended Format include the DATA LENGTH CODE and two reserved bits r1 and r0. The reserved bits have to be sent 'dominant', but Receivers accept 'dominant' and 'recessive' bits in all combinations.

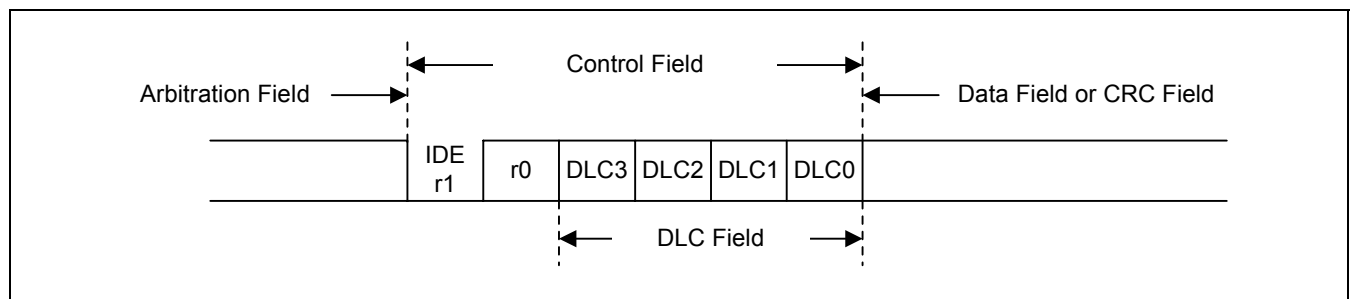


Figure 8-6. CAN Control Field

— Data length code (Standard and Extended format)

The number of bytes in the DATA FIELD is indicated by the DATA LENGTH CODE. The DLC field is four bits wide and is transmitted within the CONTROL FIELD. Coding of the number of data bytes by the DLC field is described in the table below:

Table 8-5. Data Length Code (note)

Number of Data (Bytes)	Data Length Code			
	DLC3	DLC2	DLC1	DLC0
0	D	D	D	D
1	D	D	D	R
2	D	D	R	D
3	D	D	R	R
4	D	R	D	D
5	D	R	D	R
6	D	R	R	D
7	D	R	R	R
8	R	D	D	D

NOTE: In table, D means dominant bit, R means recessive bit.

3.1.3.3.1.4 Data frame (Standard and Extended Format)

The admissible numbers of data bytes is 0 to 8. Other values may not be used.

3.1.3.3.1.5 Data field (Standard and Extended Format)

The DATA FIELD consists of the data to be transferred within a DATA FRAME. It can contain from 0 to 8 bytes, which each contain 8 bits which are transferred MSB first.

3.1.3.3.1.6 CRC field (Standard and Extended Format)

The CRC (Cyclic Redundancy Check) FIELD contains the CRC SEQUENCE followed by a CRC DELIMITER.

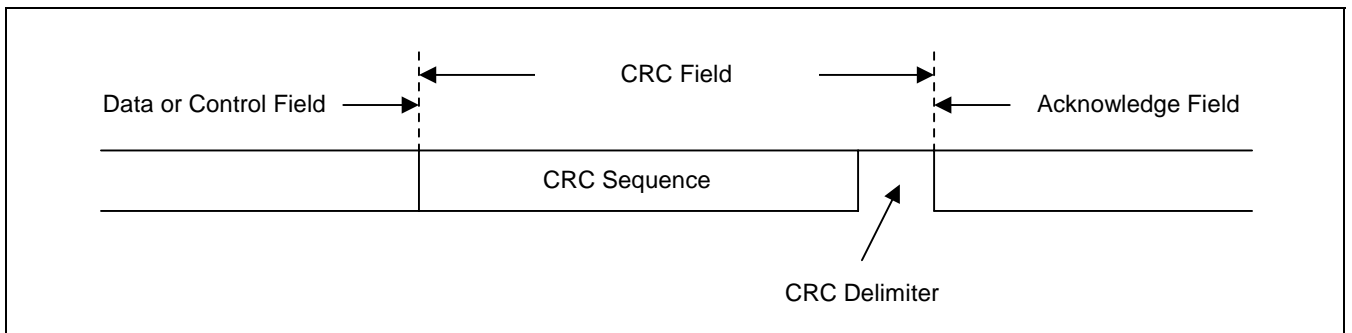


Figure 8-7. CRC Field

— CRC sequence (Standard and Extended Format)

The frame check sequence is derived from a cyclic redundancy code best suited for frames with bit counts less than 127 bits (BCH code).

In order to carry out the CRC calculation, the polynomial to be divided is defined as the polynomial, the coefficients of which are given by the de-stuffed bit stream consisting of START OF FRAME, ARBITRATION FIELD, CONTROL FIELD, DATA FIELD (if present) and, for the 15 lowest coefficients, by 0. This polynomial is divided (the coefficients are calculated modulo-2) by the generator-polynomial :

$$X^{15} + X^{14} + X^{10} + X^8 + X^7 + X^4 + X^3 + 1.$$

The remainder of this polynomial division is the CRC SEQUENCE transmitted over the bus.

In order to implement this function, a 15 bit shift register CRC_RG(14:0) can be used. If NXTBIT denotes the next bit of the bit stream, given by the de-stuffed bit sequence from START OF FRAME until the end of the DATA FIELD, the CRC SEQUENCE is calculated as follows :

```
CRC_RG = 0 // initialize shift register
REPEAT CRCNXT = NXTBIT EXOR CRC_RG[14]
CRC_RG[14:1] = CRC_RG[13:0] // shift left by
CRC_RG[0] = 0 // 1 position
IF CRCNXT THEN
CRC_RG[14:0] = CRC_RG[14:0] EXOR 0x4599
ENDIF
UNTIL (CRC sequence starts or there is an ERROR condition)
```

After the transmission/reception of the last bit of the DATA FIELD, CRC_RG contains the CRC sequence.

— CRC delimiter (standard and extended format)

The CRC SEQUENCE is followed by the CRC DELIMITER which consists of a single 'recessive' bit.

3.1.3.3.1.7 ACK Field (Standard and Extended Format)

The ACK (acknowledgement) FIELD is two bits long and contains the ACK SLOT and the ACK DELIMITER. In the ACK FIELD the transmitting node sends two 'recessive' bits.

A RECEIVER which has received a valid message correctly, reports this to the TRANSMITTER by sending a 'dominant' bit during the ACK SLOT (it sends 'ACK').

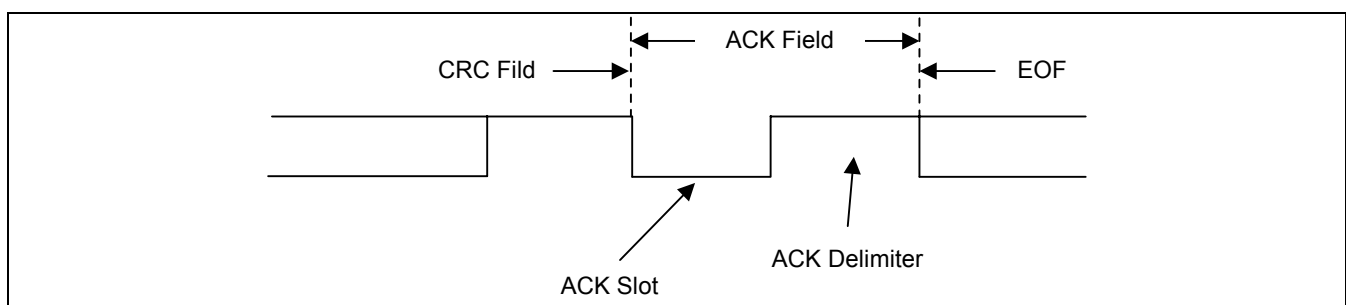


Figure 8-8. ACK Field

— ACK SLOT:

All stations having received the matching CRC SEQUENCE report this within the ACK SLOT by superscribing the 'recessive' bit of the TRANSMITTER by a 'dominant' bit.

— ACK DELIMITER :

The ACK DELIMITER is the second bit of the ACK FIELD and has to be a 'recessive' bit. As a consequence, the ACK SLOT is surrounded by two 'recessive' bits (CRC DELIMITER, ACK DELIMITER).

3.1.3.3.1.8 End of Frame (Standard and Extended Format)

Each DATA FRAME and REMOTE FRAME is delimited by seven 'recessive' bits. These bits form the END OF FRAME SEQUENCE (EOF).

3.1.3.3.2 Remote Frame

A node acting as a RECEIVER for certain data can initiate the transmission of the respective data by its source node by sending a REMOTE FRAME. A REMOTE FRAME exists both in Standard Format and in Extended Format. In both case it is composed of six different bit fields :

Start Of Frame (SOF)

- arbitration field
- control field
- CRC field
- ACK field
- End Of Frame

Contrary to DATA FRAMEs, the RTR bit of REMOTE FRAMEs is 'recessive'. There is no DATA FIELD, independent of the values of the DATA LENGTH CODE which may be signed any value within the admissible range from 0 to 8. The value is the DATA LENGTH CODE of the corresponding DATA FRAME.

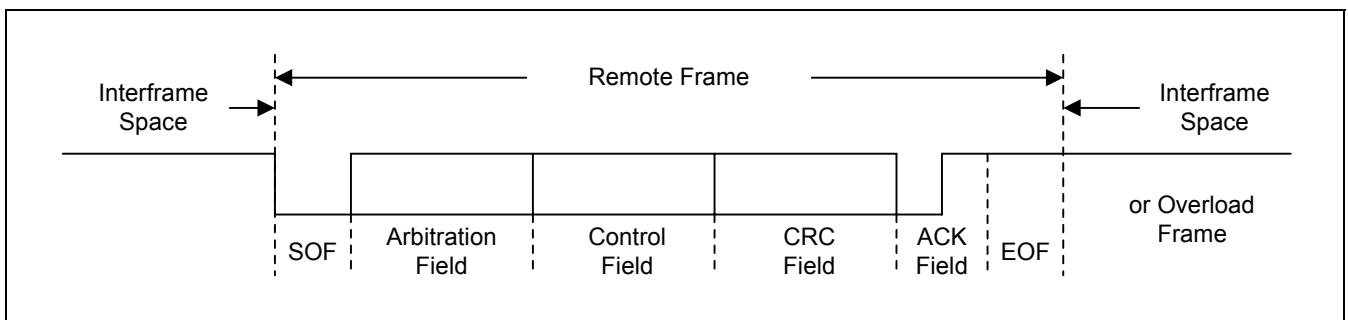


Figure 8-9. Remote Frame

The polarity of the RTR bit indicates whether a transmitted frame is a DATA FRAME (RTR bit 'dominant') or a REMOTE FRAME (RTR bit 'recessive').

3.1.3.3.3 Error frame

The ERROR FRAME consists of two different fields. The first field is given by the superposition of ERROR FLAGS contributed from different nodes. The following second field is the ERROR DELIMITER. In order to terminate an ERROR FRAME correctly, an 'error passive' node may need the bus to be 'bus idle' for at least three bit times (if there is a local error at an 'error passive' receiver). Therefore the bus should not be loaded to 100%.

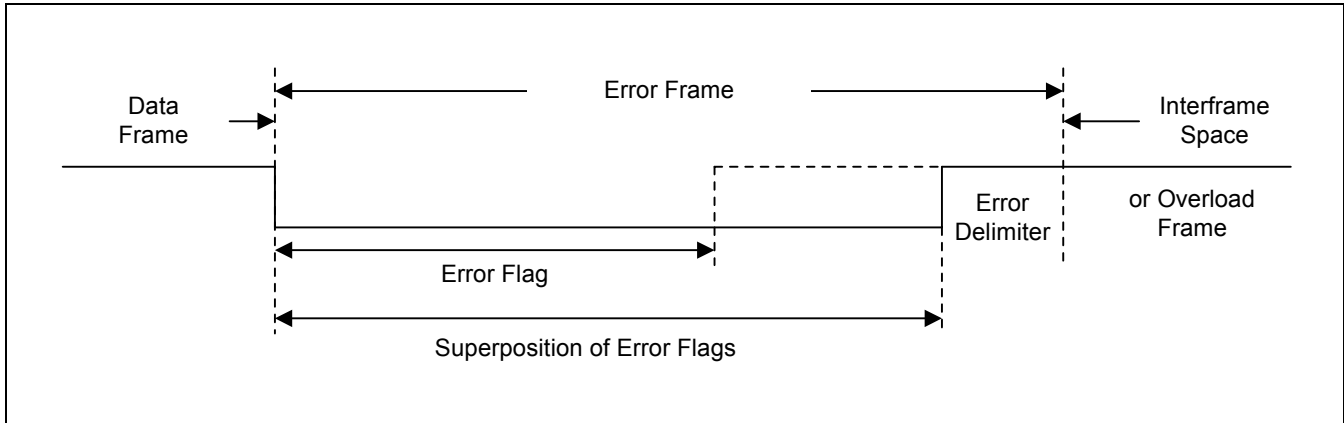


Figure 8-10. Error Frame

3.1.3.3.3.1 Error Flag

There are two forms of an ERROR FLAG: an ACTIVE ERROR FLAG and a PASSIVE ERROR FLAG.

- The ACTIVE ERROR FLAG consists of six consecutive 'dominant' bits.
- The PASSIVE ERROR FLAG consists of six consecutive 'recessive' bits unless it is overwritten by 'dominant' bits from other nodes.

An 'error active' node detecting an error condition signals this by transmission of an ACTIVE ERROR FLAG. The ERROR FLAG's form violates the law of bit stuffing (see CODING) applied to all fields from START OF FRAME to CRC DELIMITER or destroys the fixed form ACK FIELD or END OF FRAME field. As a consequence, all other nodes detect an error condition and on their part start transmission of an ERROR FLAG. So the sequence of 'dominant' bits which actually can be monitored on the bus results from a superposition of different ERROR FLAGS transmitted by individual nodes. The total length of this sequence varies between a minimum of six and a maximum of twelve bits.

An 'error passive' node detecting an error condition tries to signal this by transmission of a PASSIVE ERROR FLAG. The 'error passive' node waits for six consecutive bits of equal polarity, beginning at the start of the PASSIVE ERROR FLAG. The PASSIVE ERROR FLAG is complete when these six equal bits have been detected.

3.1.3.3.3.2 Error Delimiter

The ERROR DELIMITER consists of eight 'recessive' bits.

After transmission of an ERROR FLAG each node sends 'recessive' bits and monitors the bus until it detects a 'recessive' bit. Afterwards it starts transmitting seven more 'recessive' bits.

3.1.3.3.4 Overload frame

The OVERLOAD FRAME contains the two bit fields OVERLOAD FLAG and OVERLOAD DELIMITER. There are two kinds of OVERLOAD conditions, which both lead to the transmission of an OVERLOAD FLAG :

- The internal conditions of a receiver, which requires a delay of the next DATA FRAME or REMOTE FRAME.
- Detection of a 'dominant' bit at the first and second bit of INTERMISSION.

If a CAN node samples a dominant bit at the eighth bit (the last one) of an ERROR DELIMITER or OVERLOAD DELIMITER, it will start transmitting an OVERLOAD FRAME (not an ERROR FRAME). The Error Counters will not be incremented.

The start of an OVERLOAD FRAME due to OVERLOAD condition 1 is only allowed to be started at the first bit time of an expected INTERMISSION, whereas OVERLOAD FRAMEs due to OVERLOAD condition 2 and condition 3 start one bit after detecting the 'dominant' bit.

At most two OVERLOAD FRAMEs may be generated to delay the next DATA or REMOTE FRAME.

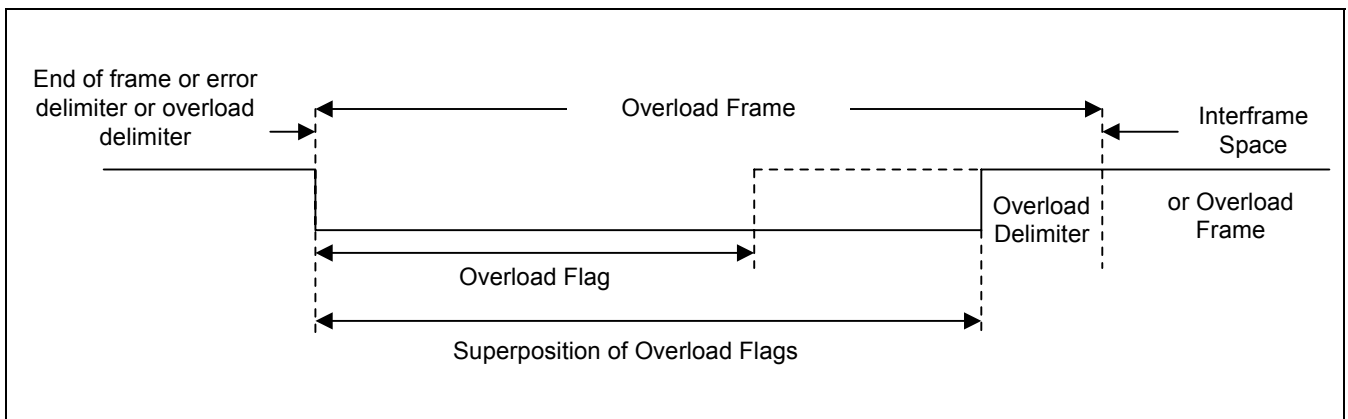


Figure 8-11. Overload Frame

3.1.3.3.4.1 Overload flag

Consists of six 'dominant' bits. The overall form corresponds to that of the ACTIVE ERROR FLAG.

The OVERLOAD FLAG's form destroys the fixed form of the INTERMISSION field. As a consequence, all other nodes also detect an OVERLOAD condition and on their part start transmission of an OVERLOAD FLAG. In case that there is a 'dominant' bit detected during the 3rd bit of INTERMISSION then it will interpret this bit as START OF FRAME.

NOTE

Controllers based on the CAN Specification version 1.0 and 1.1 have another interpretation of the 3rd bit of INTERMISSION: if a dominant bit was detected locally at some node, the other nodes will not interpret the OVERLOAD FLAG correctly, but interpret the first of these six 'dominant' bits as START OF FRAME ; the sixth 'dominant' bit violates the rule of bit stuffing causing an error condition.

3.1.3.3.4.2 Overload Delimiter

Consists of eight 'recessive' bits.

The OVERLOAD DELIMITER is of the same form as the ERROR DELIMITER. After transmission of an OVERLOAD FLAG the node monitors the bus until it detects a transition from a 'dominant' to a 'recessive' bit. At this point of time every bus node has finished sending its OVERLOAD FLAG and all nodes start transmission of seven more 'recessive' bits in coincidence.

3.1.3.3.5 Interframe Spacing

DATA FRAMES and REMOTE FRAMES are separated from preceding frames whatever type they are (DATA FRAME, REMOTE FRAME, ERROR FRAME, OVERLOAD FRAME) by a bit field called INTERFRAME SPACE. In contrast, OVERLOAD FRAMES and ERROR FRAMES are not preceded by an INTERFRAME SPACE and multiple OVERLOAD FRAMES are not separated by an INTERFRAME SPACE.

3.1.3.3.5.1 Interframe Space

Contains the bit fields INTERMISSION and BUS IDLE and, for 'error passive' nodes, which have been TRANSMITTER of the previous message, SUSPEND TRANSMISSION.

For nodes which are not 'error passive' or have been RECEIVER of the previous message:

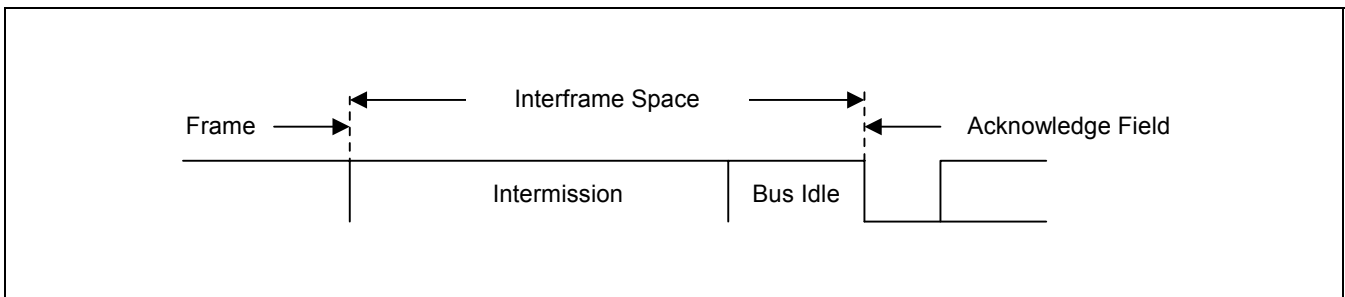


Figure 8-12. Interframe space for Receiver

For 'error passive' nodes which have been TRANSMITTER of the previous message:

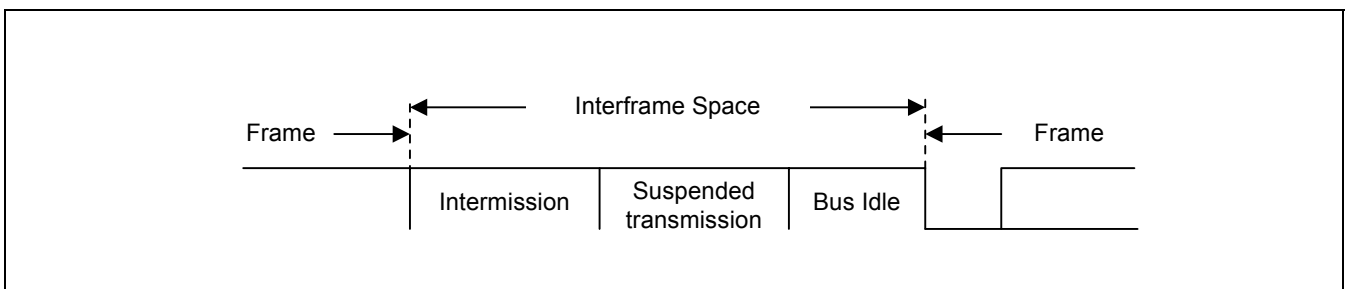


Figure 8-13. Interframe space for transmitter

3.1.3.3.5.2 Intermission

Consists of three 'recessive' bits.

During INTERMISSION the only action to be taken is signaling an OVERLOAD condition and no node is allowed to actively start a transmission of a DATA FRAME or REMOTE FRAME.

NOTE

if a CAN node has a message waiting for transmission and it samples a dominant bit at the third bit of INTERMISSION, it will interpret this as a START OF FRAME bit, and, with the next bit, start transmitting its message with the first bit of its IDENTIFIER without first transmitting a START OF FRAME bit and without becoming receiver.

3.1.3.3.5.3 Bus Idle

The period of BUS IDLE may be of arbitrary length. The bus is recognized to be free and any node having something to transmit can access the bus. A message, which is pending for transmission during the transmission of another message, is started in the first bit following INTERMISSION.

The detection of a 'dominant' bit on the bus is interpreted as START OF FRAME.

3.1.3.3.5.4 Suspend transmission

After an 'error passive' node has transmitted a message, it sends eight 'recessive' bits following INTERMISSION, before starting to transmit a further message or recognizing the bus to be idle. If meanwhile a transmission (caused by another node) starts, the node will become receiver of this message.

3.1.3.4 Conformance with Regard to Frame Formats

The Standard Format is equivalent to the Data/Remote Frame Format as it is described in the CAN Specification 1.2. In contrast the Extended Format is a new feature of the CAN protocol. In order to allow the design of relatively simple controllers, the implementation of the Extended Format to its full extend is not required (e.g. send messages or accept data from messages in Extended Format), whereas the Standard Format must be supported without restriction.

New controllers are considered to be in conformance with this CAN Specification, if they have at least the following properties with respect to the Frame Formats defined before.

- Every new controller supports the Standard Format ;
- Every new controller can receive messages of the Extended Format. This requires that Extended Frames are not destroyed just because of their format. It is, however, not required that the Extended Format must be supported by new controllers.

3.1.4 Message Filtering

Message filtering is based upon the whole Identifier. Mask registers that allow any Identifier bit to be set 'don't care' for message filtering, may be used to select groups of Identifiers to be mapped into the attached receive buffers. If mask registers are implemented every bit of the mask registers must be programmable, i.e. they can be enabled or disabled for message filtering. The length of the mask register can comprise the whole IDENTIFIER or only part of it.

3.1.5 Message Validation

The point of time at which a message is taken to be valid, is different for the transmitter and the receivers of the message.

3.1.5.1 Transmitter

The message is valid for the transmitter, if there is no error until the end of END OF FRAME. If a message is corrupted, retransmission will follow automatically and according to prioritization. In order to be able to compete for bus access with other messages, retransmission has to start as soon as the bus is idle.

3.1.5.2 Receivers

The message is valid for the receivers, if there is no error until the last but one bit of END OF FRAME. The value of the last bit of END OF FRAME is treated as 'don't care', a dominant value does not lead to a FORM ERROR.

3.1.6 Coding

BIT STREAM CODING : the frame segments START OF FRAME, ARBITRATION FIELD, CONTROL FIELD, DATA FIELD and CRC SEQUENCE are coded by the method of bit stuffing. Whenever a transmitter detects five consecutive bits of identical value in the bit stream to be transmitted it automatically inserts a complementary bit in the actual transmitted bit stream.

The remaining bit fields of the DATA FRAME or REMOTE FRAME (CRC DELIMITER, ACK FIELD and END OF FRAME) are of fixed form and not stuffed. The ERROR FRAME and the OVERLOAD FRAME are of fixed form as well and not coded by the method of bit stuffing.

The bit stream in a message is coded according to the Non-Return-to-Zero (NRZ) method. This means, that during the total bit time the generated bit level is either 'dominant' or 'recessive'.

3.1.7 Error Handling

3.1.7.1 Error detection

There are 5 different error types (which are not mutually exclusive) :

- **BIT ERROR:** A unit that is sending a bit on the bus also monitors the bus. A BIT ERROR has to be detected at that bit time, when the bit value that is monitored is different from the bit value that is sent. An exception is the sending of a 'recessive' bit during the stuffed bit stream of the ARBITRATION FIELD or during the ACK SLOT. Then no BIT ERROR occurs when a 'dominant' bit is monitored. A TRANSMITTER sending a PASSIVE ERROR FLAG and detecting a 'dominant' bit does not interpret this as a BIT ERROR.
- **STUFF ERROR:** A STUFF ERROR has to be detected at the bit time of the 6th consecutive equal bit level in a message field that should be coded by the method of bit stuffing.
- **CRC ERROR:** The CRC sequence consists of the result of the CRC calculation by the transmitter. The receivers calculate the CRC in the same way as the transmitter. A CRC ERROR has to be detected, if the calculated result is not the same as that received in the CRC sequence.
- **FORM ERROR:** A FORM ERROR has to be detected when a fixed-form bit field contains one or more illegal bits. (Note, that for a Receiver a dominant bit during the last bit of END OF FRAME is not treated as FORM ERROR).
- **ACKNOWLEDGEMENT ERROR:** An ACKNOWLEDGEMENT ERROR has to be detected by a transmitter whenever it does not monitor a 'dominant' bit during ACK SLOT.

3.1.7.2 Error Signaling

A node detecting 'an error condition' signals this by transmitting an ERROR FLAG. For an 'error active' node it is an ACTIVE ERROR FLAG, for an 'error passive' node it is a PASSIVE ERROR FLAG. Whenever a BIT ERROR, a STUFF ERROR, a FORM ERROR or an ACKNOWLEDGEMENT ERROR is detected by any node, transmission of an ERROR FLAG is started at the respective node at the next bit.

Whenever a CRC ERROR is detected, transmission of an ERROR FLAG starts at the bit following the ACK DELIMITER, unless an ERROR FLAG for another error condition has already been started.

3.1.8 Fault Confinement

With respect to fault confinement a unit may be in one of three states :

- 'error active'
- 'error passive'
- 'bus off'

An 'error active' unit can normally take part in bus communication and sends an ACTIVE ERROR FLAG when an error has been detected.

An 'error passive' unit must not send an ACTIVE ERROR FLAG. It takes part in bus communication, but when an error has been detected only a PASSIVE ERROR FLAG is sent. Also after a transmission, an 'error passive' unit will wait before initiating a further transmission (See SUSPEND TRANSMISSION).

A 'bus off' unit is not allowed to have any influence on the bus (E.g. output drivers switched off).

For fault confinement two counts are implemented in every bus unit :

- TRANSMIT ERROR COUNT
- RECEIVE ERROR COUNT

These counts are modified according to the following rules (note that more than one rule may apply during a given message transfer) :

- When a RECEIVER detects an error, the RECEIVE ERROR COUNT will be increased by 1, except when the detected error was a BIT ERROR during the sending of an ACTIVE ERROR FLAG or an OVERLOAD FLAG.
- When a RECEIVER detects a 'dominant' bit as the first bit after sending an ERROR FLAG the RECEIVE ERROR COUNT will be increased by 8.
- When a TRANSMITTER sends an ERROR FLAG the TRANSMIT ERROR COUNT is increased by 8 except :
 - Exception 1 : If the TRANSMITTER is 'error passive' and detects an ACKNOWLEDGEMENT ERROR because of not detecting a 'dominant' ACK and does not detect a 'dominant' bit while sending its PASSIVE ERROR FLAG.
 - Exception 2 : If the TRANSMITTER sends an ERROR FLAG because a STUFF ERROR occurred during ARBITRATION, and should have been 'recessive', and has been sent as 'recessive' but monitored as 'dominant'.

In exceptions 1 and 2 the TRANSMIT ERROR COUNT is not changed.

- If an TRANSMITTER detects a BIT ERROR while sending an ACTIVE ERROR FLAG or an OVERLOAD FLAG the TRANSMIT ERROR COUNT is increased by 8.
- If an RECEIVER detects a BIT ERROR while sending an ACTIVE ERROR FLAG or an OVERLOAD FLAG the RECEIVE ERROR COUNT is increased by 8.
- Any node tolerates up to 7 consecutive 'dominant' bits after sending an ACTIVE ERROR FLAG, PASSIVE ERROR FLAG or OVERLOAD FLAG. After detecting the 14th consecutive 'dominant' bit (in case of an ACTIVE ERROR FLAG or an OVERLOAD FLAG) or after detecting the 8th consecutive 'dominant' bit following a PASSIVE ERROR FLAG, and after each sequence of additional eight consecutive 'dominant' bits every TRANSMITTER increases its TRANSMIT ERROR COUNT by 8 and every RECEIVER increases its RECEIVE ERROR COUNT by 8.
- After the successful transmission of a message (getting ACK and no error until END OF FRAME is finished) the TRANSMIT ERROR COUNT is decreased by 1 unless it was already 0.
- After the successful reception of a message (reception without error up to the ACK SLOT and the successful sending of the ACK bit), the RECEIVE ERROR COUNT is decreased by 1, if it was between 1 and 127. If the RECEIVE ERROR COUNT was 0, it stays 0, and if it was greater than 127, then it will be set to a value between 119 and 127.
- A node is 'error passive' when the TRANSMIT ERROR COUNT equals or exceeds 128, or when the RECEIVE ERROR COUNT equals or exceeds 128. An error condition letting a node become 'error passive' causes the node to send an ACTIVE ERROR FLAG.
- A node is 'bus off' when the TRANSMIT ERROR COUNT is greater than or equal to 256.
- An 'error passive' node becomes 'error active' again when both the TRANSMIT ERROR COUNT and the RECEIVE ERROR COUNT are less than or equal to 127.
- An node which is 'bus off' is permitted to become 'error active' (no longer 'bus off') with its error counters both set to 0 after 128 occurrences of 11 consecutive 'recessive' bits have been monitored on the bus.

NOTES

1. An error count value greater than about 96 indicates a heavily disturbed bus. It may be of advantage to provide means to test for this condition.
2. Start-up / Wake-up : If during system start-up only 1 node is online, and if this node transmits some message, it will get no acknowledgement, detect an error and repeat the message. It can become 'error passive' but not 'bus off' due to this reason.

3.1.9 Oscillator tolerance

A maximum oscillator tolerance of 1.58% is given and therefore a ceramic resonator can be used at a bus speed of up to 125 Kbits/s as a rule of thumb. For a more precise evaluation refer to: Dais, S; Chapman, M:

"Impact of Bit Representation on Transport Capacity and Clock Accuracy in Serial Data Streams",

SAE Technical Paper Series 890532, Multiplexing in Automobile SP-773, March 1989.

For the full bus speed range of the CAN protocol, a quartz oscillator is required.

The chip of the CAN network with the highest requirement for its oscillator accuracy determines the oscillator accuracy which is required from all the other nodes.

NOTE

CAN controllers following this CAN Specification and controllers following the previous versions 1.0 and 1.1, used in one and the same network, must all be equipped with a quartz oscillator. That means ceramic resonators can only be used in a network with all the nodes of the network following CAN Protocol Specification versions 1.2 or later.

3.1.10 Bit Timing Requirements

3.1.10.1 Nominal bit rate

The Nominal Bit Rate is the number of bits per second transmitted in the absence of resynchronization by an ideal transmitter.

3.1.10.2 Nominal bit time

The nominal bit time of the network is uniform throughout the network and is given by:

$$\text{NOMINAL BIT TIME} = 1 / \text{NOMINAL BIT RATE}$$

The Nominal Bit Time can be thought of as being divided into separate non-overlapping time segments. These segments are:

- Synchronization Segment (SYNC_SEG)
- Propagation Time Segment (PROP_SEG)
- Phase Buffer Segment1 (PHASE_SEG1)
- Phase Buffer Segment2 (PHASE_SEG2)

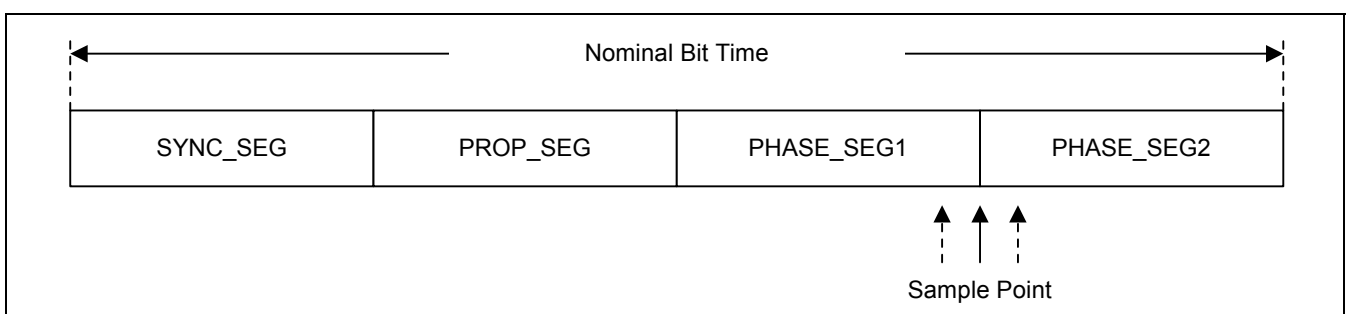


Figure 8-14. Partition of Bit Time

The duration of each segment is set in the mode register and is expressed as a multiple of time quantum (TCAN).

3.1.10.3 Time Quantum

The TIME QUANTUM (T_Q) is a fixed unit of time derived from the oscillator period. It is programmable by the way of a pre-scalar, called the BAUD RATE PRESCALAR (BD[9:0] in CAN_MR register). The formula between T_Q , CORECLK and BD[9:0] is the following :

$$T_Q = (BD[9:0] + 1) / CORECLK$$

Each segment (SYNC_SEG, PROP_SEG, PHASE_SEG1 and PHASE_SEG2) is an integer multiple of TIME QUANTUM (T_Q).

3.1.10.4 Synchronization Segment

This part of the bit time is used to synchronize the various nodes on the bus. An edge is expected to lie within this segment. The duration of the synchronization segment, SYNC_SEG, is not programmable and is fixed at one time quantum.

3.1.10.5 Propagation Segment

This part of the bit time is used to compensate for the physical delay times within the network. It is twice the sum of the signal's propagation time on the bus line, the input comparator delay, and the output driver delay.

The duration of the propagation segment PROP_SEG may be between 1 and 8 time quanta. It is programmable by the way of the PROP bits in the MR, and it is equal to :

3.1.10.6 Phase Segment 1, Phase Segment 2

These PHASE BUFFER SEGMENTS are used to compensate for edge phase errors. These segments can be lengthened or shortened by resynchronization.

3.1.10.7 Sample point

The SAMPLE POINT is the point of time at which the bus level is read and interpreted as the value of that respective bit. It's location is at the end of PHASE_SEG1.

3.1.10.8 Information processing time (IPT)

The INFORMATION PROCESSING TIME is the time segment starting with the SAMPLE POINT reserved for calculation the subsequent bit level.

3.1.10.9 Length of Time Segments

- SYNC_SEG is 1 TIME QUANTUM long.
- PROP_SEG is programmable to be 1,2,...,8 TIME QUANTA long.
- PHASE_SEG1 is programmable to be 1,2,...,8 TIME QUANTA long.
- PHASE_SEG2 is the maximum of PHASE_SEG1 and the INFORMATION PROCESSING TIME.
- The INFORMATION PROCESSING TIME is less than or equal to 2 TIME QUANTA long.

The total number of TIME QUANTA in a bit time has to be programmable at least from 8 to 25.

NOTE

It is often intended that control units do not make use of different oscillators for the local CPU and its communication device. Therefore the oscillator frequency of a CAN device tends to be that of the local CPU and is determined by the requirements of the control unit. In order to derive the desired bit rate, programmability of the bit timing is necessary. In case of CAN implementations that are designed for use without a local CPU the bit timing cannot be programmable. On the other hand these devices allow to choose an external oscillator in such a way that the device is adjusted to the appropriate bit rate so that the programmability is dispensable for such components. The position of the sample point, however, should be selected in common for all nodes. Therefore the bit timing of CAN devices without local CPU should be compatible to the following definition of the bit time:

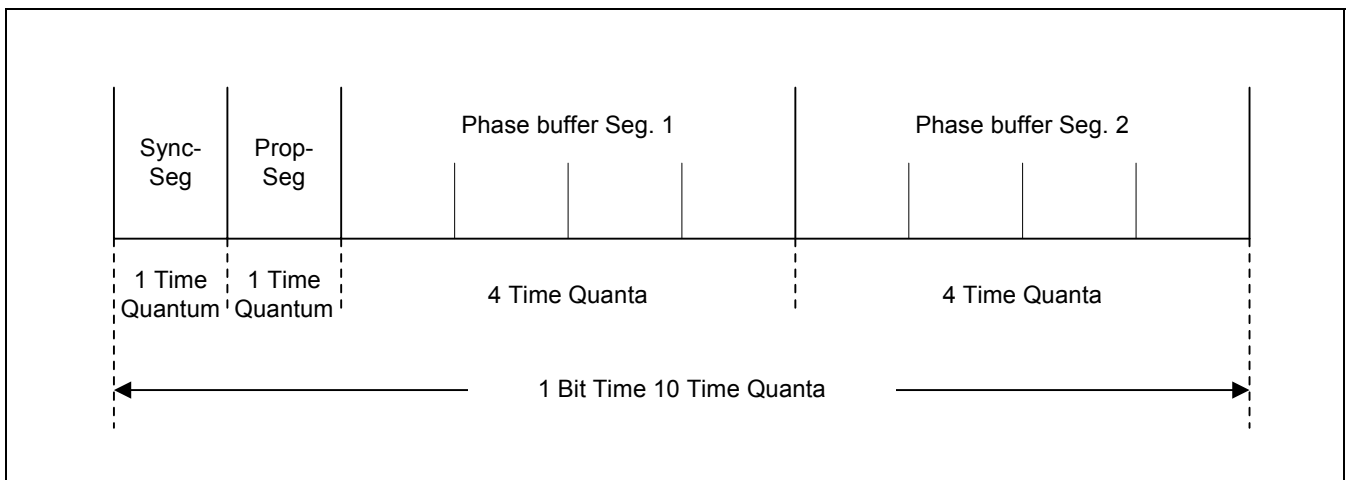


Figure 8-15. Example of Nominal Bit Time

3.1.10.10 Hard Synchronization

After a HARD SYNCHRONIZATION the internal bit time is restarted with SYNC_SEG. Thus HARD SYNCHRONIZATION forces the edge which has caused the HARD SYNCHRONIZATION to lie within the SYNCHRONIZATION SEGMENT of the restarted bit time.

3.1.10.11 Resynchronization Jump Width

As a result of RESYNCHRONIZATION PHASE_SEG1 may be lengthened or PHASE_SEG2 may be shortened. The amount of lengthening or shortening of the PHASE BUFFER SEGMENTS has an upper bound given by the RESYNCHRONIZATION JUMP WIDTH. The RESYNCHRONIZATION JUMP WIDTH shall be programmable between 1 and $\min(4, \text{PHASE_SEG1})$. Clocking information may be derived from transitions from one bit value to the other. The property that only a fixed maximum number of successive bits have the same value provides the possibility of resynchronizing a bus unit to the bit stream during a frame. The maximum length between two transitions which can be used for resynchronization is 29 bit times.

3.1.10.12 Phase Error of an Edge

The PHASE ERROR of an edge is given by the position of the edge relative to SYNC_SEG, measured in TIME QUANTA. The sign of PHASE ERROR is defined as follows:

- $e = 0$ if the edge lies within SYNC_SEG.
- $e > 0$ if the edge lies before the SAMPLE POINT.
- $e < 0$ if the edge lies after the SAMPLE POINT of the previous bit.

3.1.10.13 Resynchronization

The effect of a RESYNCHRONIZATION is the same as that of a HARD SYNCHRONIZATION, when the magnitude of the PHASE ERROR of the edge which causes the RESYNCHRONIZATION is less than or equal to the programmed value of the RESYNCHRONIZATION JUMP WIDTH. When the magnitude of the PHASE ERROR is larger than the RESYNCHRONIZATION JUMP WIDTH,

- and if the PHASE ERROR is positive, then PHASE_SEG1 is lengthened by an amount equal to the RESYNCHRONIZATION JUMP WIDTH.
- and if the PHASE ERROR is negative, then PHASE_SEG2 is shortened by an amount equal to the RESYNCHRONIZATION JUMP WIDTH.

3.1.10.14 Synchronization Rules

HARD SYNCHRONIZATION and RESYNCHRONIZATION are the two forms of SYNCHRONIZATION. They obey the following rules:

- Only one SYNCHRONIZATION within one bit time is allowed.
- An edge will be used for SYNCHRONIZATION only if the value detected at the previous SAMPLE POINT (previous read bus value) differs from the bus value immediately after the edge.
- HARD SYNCHRONIZATION is performed whenever there is a 'recessive' to 'dominant' edge during BUS IDLE.
- All other 'recessive' to 'dominant' edges fulfilling the rules 1 and 2 will be used for RESYNCHRONIZATION with the exception that a node transmitting a dominant bit will not perform a RESYNCHRONIZATION as a result of a 'recessive' to 'dominant' edge with a positive PHASE ERROR, if only 'recessive' to 'dominant' edges are used for resynchronization.

3.2 OPERATING MODES

3.2.1 Software Initialization

The software initialization is started by disabling CAN controller, either by software (setting the bit CANDIS in the CAN Control Register) or by a hardware reset, or by going bus off.

While CAN is disabled, all message transfer from and to the CAN bus is stopped, the status of the CAN bus output CAN_TX is recessive (HIGH). The counter of the ERCCR(see P8-78) is unchanged. Disabling CAN does not change any configuration register.

To initialize the CAN Controller, the CPU has to set up the Mode Register and each Message Object. If a Message Object is not needed, it is sufficient to set MSGVAL of CAN_IRx to not valid(See P8-85). Otherwise, the whole Message Object has to be initialized because the RAM is not initialized at power up.

Access to the Mode Register for the configuration of the bit timing is allowed when CAN is disabled (CANENS of CAN_SR is reset to zero) and bit CCENS in the CAN_SR is set(See P8-66).

Enabling the CAN (by CPU only) finishes the software initialization(CANEN of CAN_CR is set to '1'). Afterwards the Bit Stream Processor (BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (Bus Idle) before it can take part in bus activities and starts the message transfer.

The initialization of the Message Objects is independent of bit CANENS of CAN_SR and can be done on the fly, but the Message Objects should all be configured to particular identifiers or set to not valid(MSGVAL of CAN_IRx is set to '0') before the BSP starts the message transfer. To change the configuration of a Message Object during normal operation, the CPU has to start by setting MSGVAL of CAN_IRx to 0. When the configuration is completed, MSGVAL is set to valid again.

3.2.2 CAN message Transfer

Once the CAN is initialized and CAN is enabled, the CAN Core synchronizes itself to the CAN bus and starts the message transfer.

Received messages are stored into their appropriate Message Objects if they pass the Message Handler's acceptance filtering. The whole message including all arbitration bits, DLC and eight data bytes is stored into the Message Object.

If the Identifier Mask is used, the arbitration bits which are masked to "don't care" may be overwritten in the Message Object.

The CPU may read or write each message any time via the Interface Registers, the Message Handler guarantees data consistency in case of concurrent accesses.

Messages to be transmitted are updated by the CPU. If a permanent Message Object (arbitration and control bits set up during configuration) exists for the message, only the data bytes are updated and then TXRQST bit with NEWDAT bit are set to start the transmission(See P8-86). If several transmit messages are assigned to the same Message Object (when the number of Message Objects is not sufficient), the whole Message Object has to be configured before the transmission of this message is requested.

The transmission of any number of Message Objects may be requested at the same time, they are transmitted subsequently according to their internal priority(See P8-37). Messages may be updated or set to not valid any time, even when their requested transmission is still pending. The old data will be discarded when a message is updated before its pending transmission has started.

Depending on the configuration of the Message Object, the transmission of a message may be requested automatically by the reception of a remote frame with a matching identifier.

8-1. IMPORTANT NOTICE

The message object number 32 shall not be used in transmission.

3.2.3 Message Interface Register Sets

There are two sets of Interface Registers which are used to control the CPU access to the Message RAM. The Interface Registers avoid conflicts between CPU access to the Message RAM and CAN message reception and transmission by buffering the data to be transferred. A complete Message Object (See 3.2.4 Message Object) or part of the Message Object may be transferred between the Message RAM and the IFx Message Buffer registers in one single transfer.

The function of the two interface register sets is identical (except for BASIC mode). They can be used the way that one set of registers is used for the data to transfer to the Message RAM while the other set of registers is used for the data transfer from the Message RAM, allowing both processes to be interrupted by each other. Table 8-6 gives an overview of the two Interface Register sets.

Each set of Interface Registers consists of Message Buffer Registers controlled by their own Transfer Management Registers. The Transfer Management Register specifies the direction of the data transfer, which part of a Message Object will be transferred and is used to select a Message Object to the Message RAM as a target or source for the transfer and to start the action specified.

Table 8-6. Interface Register Sets

Offset	Interface 0 Registers	Name	Access
0x0100	Interface 0 Transfer Management Register	CAN_TMR0	Read/Write
0x0104	Interface 0 Data A Register	CAN_DAR0	Read/Write
0x0108	Interface 0 Data B Register	CAN_DBR0	Read/Write
0x010C	Interface 0 Mask Register	CAN_MSKR0	Read/Write
0x0110	Interface 0 Identifier Register	CAN_IR0	Read/Write
0x0114	Interface 0 Message Control Register	CAN_MCR0	Read/Write
0x0118	Interface 0 Stamp Register	CAN_STPR0	Read only
Offset	Interface 1 Registers	Name	Access
0x120	Interface 1 Transfer Management Register	CAN_TMR1	Read/Write
0x124	Interface 1 Data A Register	CAN_DAR1	Read/Write
0x128	Interface 1 Data B Register	CAN_DBR1	Read/Write
0x12C	Interface 1 Mask Register	CAN_MSKR1	Read/Write
0x130	Interface 1 Identifier Register	CAN_IR1	Read/Write
0x134	Interface 1 Message Control Register	CAN_MCR1	Read/Write
0x138	Interface 1 Stamp Register	CAN_STPR1	Read only

3.2.4 Message Object

There are 32 Message Objects in Message RAM. They cannot be accessed directly by the CPU, these accesses are handled via the IFx Interface Registers.

The table below gives an overview of the Message Object content.

Table 8-7. Message Object Content

Name	Description
DAR	First data bytes message
DBR	Last data bytes message
MSKR	Identifier mask
IR	Message identifier
MCR	Message control
STPR	Stamp date of message

3.2.5 Disabled Automatic Retransmission

According to the CAN Specification (see ISO11898, 6.3.3 Recovery Management), the CAN provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. The frame transmission service will not be confirmed to the user before the transmission is successfully completed. By default, this means for automatic retransmission is enabled. It can be disabled to enable the CAN to work within a Time Triggered CAN (TTCAN, see ISO11898-1) environment.

The Automatic Retransmission mode is disabled by programming bit AR in the CAN Mode Register (CAN_MR) to zero. In this operation mode the programmer has to consider the different behavior of bits TXRQST and NEWDAT in the Control Registers of the Message Buffers(See P8-86) :

- When a transmission starts bit TXRQST of the respective Message Buffer is reset, while bit NEWDAT remains set.
- When the transmission completed successfully bit NEWDAT is reset.
- When a transmission failed (lost arbitration or error) bit NEWDAT remains set. To restart the transmission the CPU has to set TXRQST back to one.

3.2.6 Test Mode

The CAN module has three test mode that can be combined :

- the Silent mode,
- the Loop back mode,
- the Basic mode.

Moreover, the pin CAN_TX can be controlled by software. It can also be configured in open drain allowing to connect directly CAN controllers together without external transceiver.

The pin CAN_TX control and test modes are enabled through the Test Register. Write access in the Test Register is protected by a control access key. To enable or disable a test function, it is necessary to write the correct bit pattern (TESTKEY = 0x0C75) to the control access key bits at the same time as the control bits are written.

The bit RX monitors the state of pin CAN_RX and therefore is only readable.

3.2.7 Silent Mode

The CAN Core can be set in Silent Mode by programming the Test Register bit SILENT to one.

In Silent Mode, the CAN is able to receive valid data frames and valid remote frames, but it sends only recessive bits on the CAN bus and it cannot start a transmission. If the CAN Core is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the CAN Core monitors this dominant bit, although the CAN bus may remain in recessive state. The Silent Mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits (Acknowledge Bits, Error Frames). The figure below shows the connection of signals CAN_TX and CAN_RX to the CAN Core in Silent Mode.

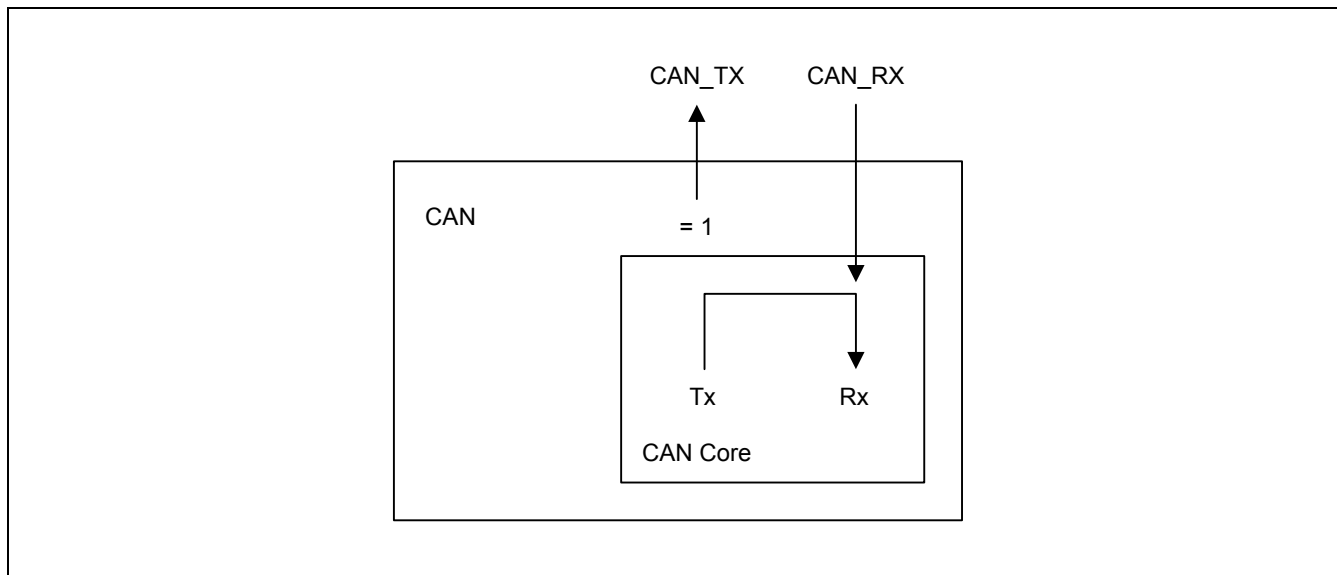


Figure 8-16. CAN Core in Silent Mode

In ISO 11898-1, the Silent Mode is called the Bus Monitoring Mode.

3.2.8 Loop Back Mode

The CAN Core can be set in Loop Back Mode by programming the Test Register bit LBACK to one. In Loop Back Mode, the CAN Core treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into a Receive Buffer. The figure below shows the connection of signals CAN_TX and CAN_RX to the CAN Core in Loop Back Mode.

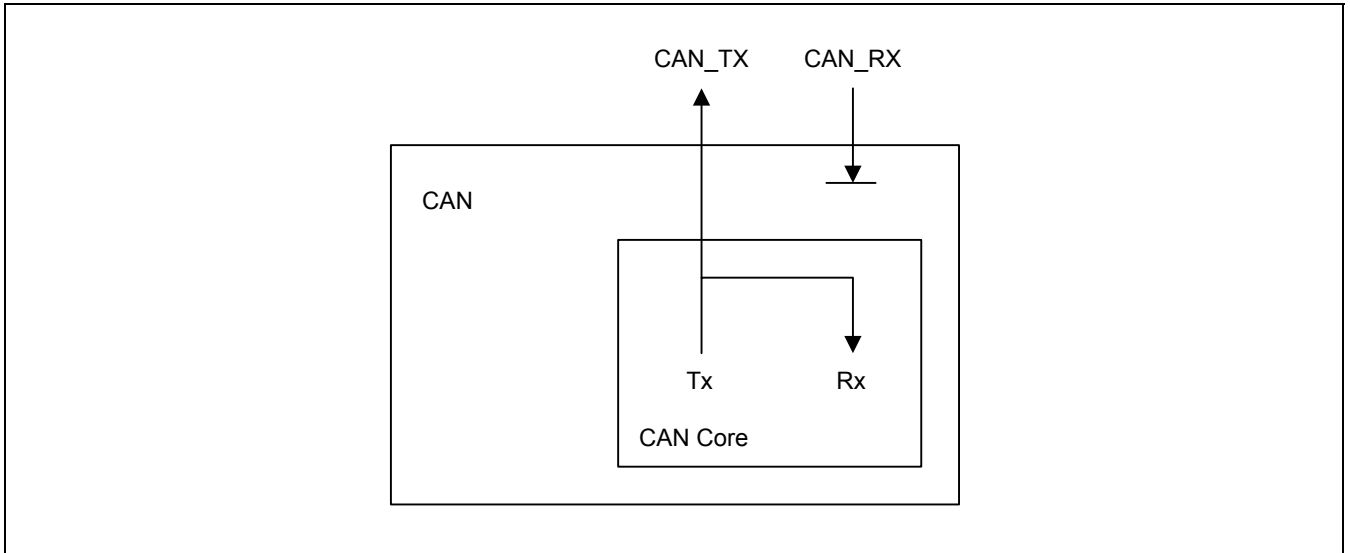


Figure 8-17. CAN Core in Loop Back Mode

This mode is provided for self-test functions. To be independent from external stimulation, the CAN Core ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in Loop Back Mode. In this mode the CAN Core performs an internal feedback from its TX output to its RX input. The actual value of the CAN_RX input pin is disregarded by the CAN Core. The transmitted messages can be monitored at the CAN_TX pin.

3.2.9 Loop Back Mode Combined With Silent Mode

It is also possible to combine Loop Back Mode and Silent Mode by programming bits LBACK and SILENT to one at the same time. This mode can be used for a “Hot Self test”, meaning the CAN can be tested without affecting a running CAN system connected to the pins CAN_TX and CAN_RX. In this mode the CAN_RX pin is disconnected from the CAN Core and the CAN_TX pin is held recessive. Figure 8-18 shows the connection of signals CAN_TX and CAN_RX to the CAN Core in case of the combination of Loop Back Mode with Silent Mode.

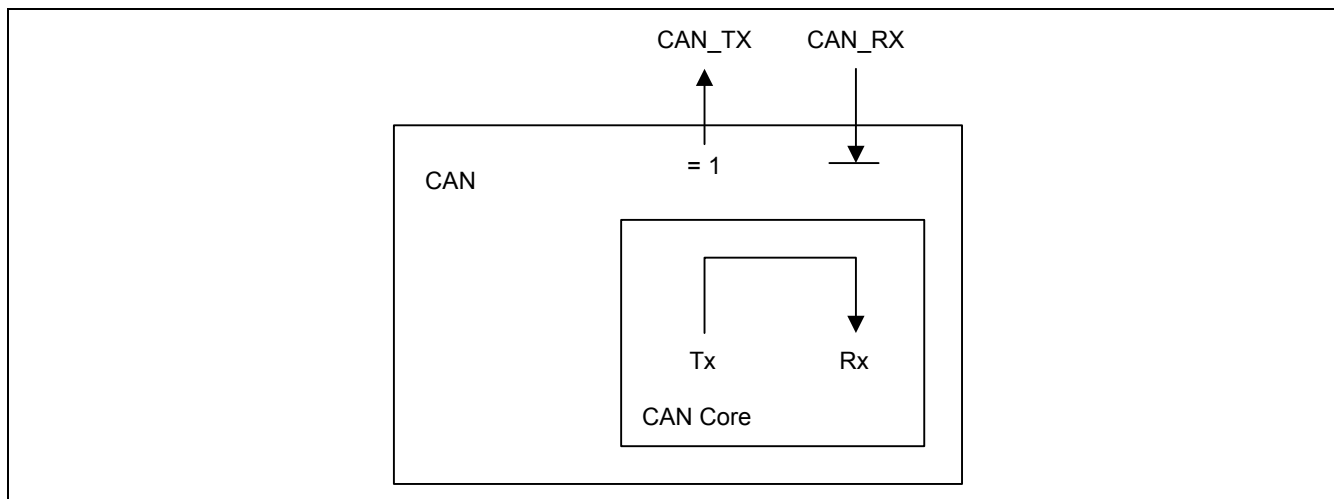


Figure 8-18. CAN Core in Loop Back Combined with Silent Mode

3.2.10 Basic Mode

The CAN Core can be set in Basic Mode by programming the Test Register bit BASIC to one. In this mode the CAN module runs without the Message RAM.

The IF0 Registers are used as Transmit Buffer. The transmission of the contents of the IF0 Registers is requested by writing the RQBTX bit of the CAN_CR register to '1'. The IF0 Registers are locked while the BTXPD bit is set in the CAN_SR register. The BTXPD bit indicates that the transmission is pending.

As soon the CAN bus is idle, the IF0 Registers are loaded into the shift register of the CAN Core and the transmission is started. When the transmission has completed, the BTXPD bit is reset and the locked IF0 Registers are released.

A pending transmission can be aborted at any time by writing the ABBTX bit of the CAN_CR register. If the CPU aborted a pending transmission, a possible retransmission in case of lost arbitration or in case of an error is disabled.

The IF1 Registers are used as Receive Buffer. After the reception of a message the contents of the shift register is stored into the IF1 Registers, without any acceptance filtering.

Additionally, the actual contents of the shift register can be monitored during the message transfer. Each time a read Message Object is initiated by writing STSR bit of the CAN_CR register, the contents of the shift register is stored into the IF1 Registers.

In Basic Mode the evaluation of all Message Object related control and status bits and of the control bits of the Ifx Transfer Management Registers is turned off. The message number of the Transfer Management registers is not evaluated. The NEWDAT and MSGLSST bits of the IF1 Message Control Register retain their function, DLC will show the received DLC, the other control bits will be read as '0'.

3.2.11 Control of pin CAN_TX

Four output functions are available for the CAN transmit pin CAN_TX. Additionally to its default function – the serial data output – it can drive the CAN Sample Point signal to monitor CAN Core's bit timing and it can drive constant dominant or recessive values. The last two functions, combined with the readable CAN receive pin CAN_RX, can be used to check the physical layer of CAN bus.

The output mode of pin CAN_TX is selected by programming the Test Register bits TX1 and TX0 as described in Test Register definition. (See P8-93)

The three test functions for pin CAN_TX interfere with all CAN protocol functions. CAN_TX must be left in its default function when CAN message transfer or any of the test modes Loop Back Mode, Silent Mode, or Basic Mode are selected.

CAN_TX pin can be programmed as open drain (it can only drive a low level). This configuration is done by setting TXOPD to '1' in CAN_TSTR register. User should disable CAN (CANDIS in CAN_CR register) before switching between open drain and normal mode in order to avoid glitch on CAN_TX output. An external pull-up is necessary to guarantee a logic level (logical one) when the pin is not being driven. This feature can be useful in two cases :

- It allows to connect directly our CAN controller to another CAN controller providing the same open drain feature without external transceiver.
- It allows to drive a 5V transceiver whereas the CAN controller is 3V3, using an external 5V pull up resistor.

3.2.12 Bus off Recovery Sequence

When the CAN controller goes in bus off mode, it becomes disable (CANENS is reset) and does not take part any longer in bus activities. To resume in normal operation mode, CPU has to enable the CAN controller (set CANEN bit in CAN_CR register), at this point the CAN controller will wait for 129 successive occurrences of bus idle (a sequence of 11 consecutive recessive bits).

At the end of the bus off recovery sequence, the read and write error counter are reset and the BUSOFF bit is reset. The bus off recovery sequence cannot be shortened by resetting CAN (SWRST set to '1' in CAN_CR).

During the waiting time after CPU has enabled CAN controller, each time a sequence of 11 recessive bits has been monitored, BIT0 bit is set in the status register, enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the proceeding of the bus off recovery sequence.

The bus off recovery sequence can be monitored by reading the REC counter : it shall be reset when CPU enables the CAN, and each time a sequence of 11 consecutive recessive bits occurred on the bus, the REC counter shall be increased by one. During the recovery sequence, BUSOFF, ERPASS and CANENS bits are set while ERWARN bit is reset. (See CAN_SR register, P8-63)

3.3 CAN APPLICATION

3.3.1 Management of Message Objects

The configuration of the Message Objects in the Message RAM will (with the exception of the bits MSGVAL, NEWDAT, ITPND, and TXRQST) not be affected by resetting the chip. All the Message Objects must be initialized by the CPU or they must not be valid (MSGVAL of CAN_IRx = '0') and the bit timing must be configured before the CPU set the CANEN bit in the CAN Control Register(CAN_CR).

The configuration of a Message Object is done by programming Mask, Arbitration, Control and Data field of one of the two interface register sets to the desired values. By writing to the corresponding IFx Transfer Management Register(CAN_TMRx), the IFx Message Buffer Registers are loaded into the addressed Message Object in the Message RAM. A message transfer between the message RAM and the IFx message buffer register is started as soon as the CPU has written in the register CAN_TMRx.

When the CANEN bit in the CAN Control Register is set, the CAN Protocol Controller state machine of the CAN Core and the Message Handler State Machine control the CAN's internal data flow. Received messages that pass the acceptance filtering are stored into the Message RAM, messages with pending transmission request are loaded into the CAN Core's Shift Register and are transmitted via the CAN bus.

The CPU reads received messages and updates messages to be transmitted via the IFx Interface Registers. Depending on the configuration, the CPU is interrupted on certain CAN message and CAN error events.

3.3.2 Message Handler State Machine

The Message Handler controls the data transfer between the RX/TX Shift Register of the CAN Core, the Message RAM and the IFx Registers.

The Message Handler FSM controls the following functions :

- Data Transfer from IFx Registers to the Message RAM.
- Data Transfer from Message RAM to the IFx Registers.
- Data Transfer from Shift Register to the Message RAM.
- Data Transfer from Message RAM to Shift Register.
- Data Transfer from Shift Register to the Acceptance Filtering unit.
- Scanning of Message RAM for a matching Message Object.
- Handling of TXRQST flags. (See CAN_MCRx register, P8-86)
- Handling of interrupts.

3.3.2.1 Data Transfer between interface register and Message RAM

When the CPU initiates a data transfer between the IFx Registers and Message RAM, the Message Handler sets the BUSYx bit in the Status Register to '1'. After the transfer has completed, the BUSYx bit is set back to '0' (see the figure below).

The respective Transfer Management Register(CAN_TMRx register, P8-79) specifies whether a complete Message Object or only parts of it will be transferred. But, due to the structure of the Message RAM it is not possible to write single bits/bytes of one Message Object, it is always necessary to write a complete Message Object into the Message RAM. Therefore the data transfer from the IFx Registers to the Message RAM requires of a read-modify-write cycle. Parts of the Message Object that are not to be changed are read from the Message RAM and then the complete contents of the Message Buffer Registers are into the Message Object.

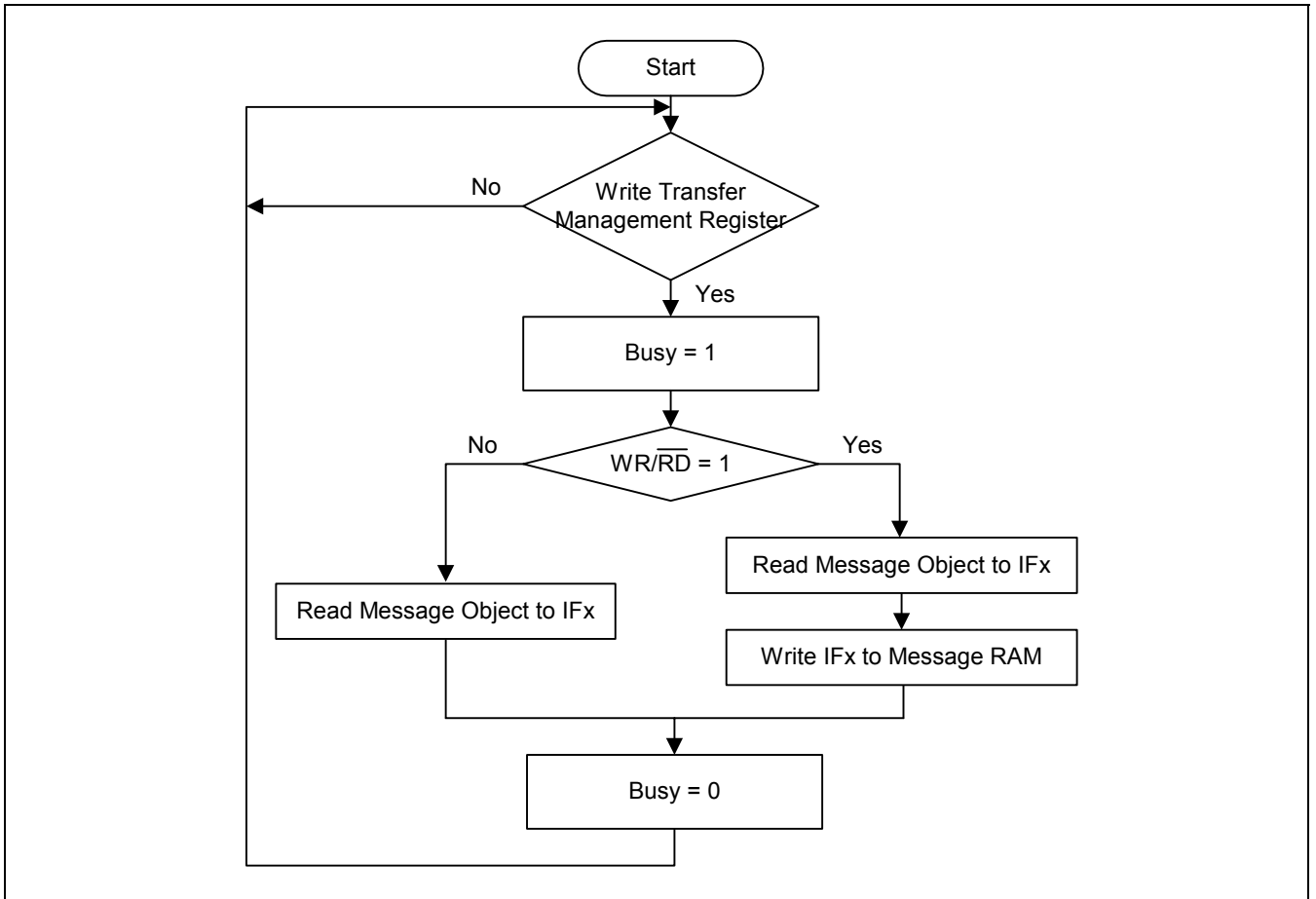


Figure 8-19. Data Transfer Between IFx Registers and Message RAM

After the partial write of a Message Object, that Message Buffer Registers that are not selected in the Transfer Management Register will set to the actual contents of the selected Message Object.

After the partial read of a Message Object, that Message Buffer Registers that are not selected in the Transfer Management Register will be left unchanged.

3.3.2.2 Transmission of Messages from Message object to shift register

If the shift register of the CAN Core cell is ready for loading and if there is no data transfer between the IFx Registers and Message RAM, the MSGVAL bits in the CAN_IRx and TXRQST bits in the CAN_MCRx are taken into consideration. The valid Message Object with the highest priority pending transmission request (see 3.3.2.4) is loaded into the shift register by the Message Handler and the transmission is started. The Message Object's NEWDAT bit is reset (See P8-87).

After a successful transmission and if no new data was written to the Message Object (NEWDAT = '0') since the start of the transmission, the TXRQST bit will be reset (See P8-86). If TXIE is set, ITPND will be set after a successful transmission. If the CAN has lost the arbitration or if an error occurred during the transmission, the message will be retransmitted as soon as the CAN bus is free again. If meanwhile the transmission of a message with higher priority has been requested, the messages will be transmitted in the order of their priority.

3.3.2.3 Acceptance Filtering of Received Messages in the shift register

When the arbitration and control field (ID + IDE + RTR + DLC) of an incoming message is completely shifted into the RX/TX Shift Register of the CAN Core, the Message Handler FSM starts the scanning of the Message RAM for a matching valid Message Object.

To scan the Message RAM for a matching Message Object, the Acceptance Filtering unit is loaded with the arbitration bits from the CAN Core shift register. Then the arbitration and mask fields (including MSGVAL, UMASK, NEWDAT, and OVERWRITE) of Message Object 1 are loaded into the Acceptance Filtering unit and compared with the arbitration field from the shift register. This is repeated with each following Message Object until a matching Message Object is found or until the end of the Message RAM is reached.

If a match occurs, the scanning is stopped and the Message Handler FSM proceeds depending on the type of frame (Data Frame or Remote Frame) received.

3.3.2.3.1 Reception of a Data Frame

The Message Handler FSM stores the message from the CAN Core shift register into the respective Message Object in the Message RAM. Not only the data bytes, but also all arbitration bits and the Data Length Code are stored into the corresponding Message Object. This is implemented to keep the data bytes connected with the identifier even if arbitration mask registers are used.

The NEWDAT bit is set to indicate that new data (not yet seen by the CPU) has been received. The CPU should reset NEWDAT when it reads the Message Object. If at the time of the reception the NEWDAT bit was already set, MSGLST is set to indicate that the previous data (supposedly not seen by the CPU) is lost. If the RXIE bit is set, the ITPND bit is set, causing the Interrupt Register to point to this Message Object.

The TXRQST bit of this Message Object is reset to prevent the transmission of a Remote Frame, while the requested Data Frame has just been received.

3.3.2.3.2 Reception of a Remote Frame

When a Remote Frame is received, three different configurations of the matching Message Object have to be considered:

[1] MDIR of CAN_IRx (P8-84) = '1' (message direction = transmit), RMTEN of CAN_MCRx (P8-87) = '1', UMASK of CAN_MCRx (P8-87) = '1' or '0'. At the reception of a matching Remote Frame, the TXRQST bit of this Message Object is set. The rest of the Message Object remains unchanged.

[2] MDIR of CAN_IRx (P8-84) = '1' (message direction = transmit), RMTEN of CAN_MCRx (P8-87) = '0', UMASK of CAN_MCRx (P8-87) = '0'. At the reception of a matching Remote Frame, the TXRQST bit of this Message Object remains unchanged; the Remote Frame is ignored.

[3] MDIR of CAN_IRx (P8-84) = '1' (message direction = transmit), RMTEN of CAN_MCRx (P8-87) = '0', UMASK of CAN_MCRx (P8-87) = '1'. At the reception of a matching Remote Frame, the TXRQST bit of this Message Object is reset. The arbitration and control field (ID + IDE + RTR + DLC) from the shift register is stored into the Message Object in the Message RAM and the NEWDAT bit of this Message Object is set. The data field of the Message Object remains unchanged; the Remote Frame is treated similar to a received Data Frame.

3.3.2.4 Receive / Transmit priority

The receive/transmit priority for the Message Objects is attached to the message number. Message Object 1 has the highest priority, while Message Object 32 has the lowest priority. If more than one transmission request is pending, they are serviced due to the priority of the corresponding Message Object.

3.3.3 Configuration of a Transmit Object

The table below shows how a Transmit object should be initialized.

Table 8-8. Initialization of a Transmit Object

MSGVAL	ID	DATA	MSK	OVER WRITE	MDIR	NEWDAT	MSGLST	RXIE	TXIE	ITPND	RMTEN	TXRQST
1	Appl	Appl	Appl	1	1	0	0	0	Appl	0	Appl	0

The Identifier Registers (ID and XTD of CAN_IRx, P8-84) are given by the application. They define the identifier and type of the outgoing message. If an 11-bit Identifier ("Standard Frame") is used, it is programmed to ID[28:18]. And, ID[17:0] can then be disregarded.

If the TXIE of CAN_MCRx (P8-87) is set, the ITPND bit will be set after a successful transmission of the Message Object.

If the RMTEN of CAN_MCRx (P8-87) is set, a matching received Remote Frame will cause the TXRQST of CAN_MCRx (P8-87) to be set; the Remote Frame will autonomously be answered by a Data Frame.

The Data Registers (DLC of CAN_MCRx (P8-86), DATA[0:7]) are given by the application, TXRQST and RMTEN may not be set before the data is valid.

The Mask Registers (MSK, UMASK, MXTD, and MMDIR bits) may be used (UMASK='1') to allow groups of Remote Frames with similar identifiers to set the TXRQST bit. Handle with care. The MDIR bit should not be masked. (See P8-83, P8-87)

3.3.4 Updating a Transmit Object

The CPU may update the data bytes of a Transmit object any time via the IFx Interface registers, neither MSGVAL of CAN_IRx (P8-84) nor TXRQST of CAN_MCRx (P8-86) have to be reset before the update.

Even if only a part of the data bytes are to be updated, all four bytes of the corresponding IFx Data A Register or IFx Data B Register have to be valid before the content of that register is transferred to the Message Object. Either the CPU has to write all four bytes into the IFx Data Register or the Message Object is transferred to the IFx Data Register before the CPU writes the new data bytes.

To prevent the reset of TXRQST at the end of a transmission that may already be in progress while the data is updated, NEWDAT has to be set together with TXRQST. For details see section “Transmission of messages from message object to shift register”.

When NEWDAT is set together with TXRQST, NEWDAT will be reset as soon as the new transmission has started.

3.3.5 Configuration of a Receive Object

The table below shows how a Receive object should be initialized.

Table 8-9. Initialization of a Receive Object

MSGVAL	ID	DATA	MSK	OVER WRITE	MDIR	NEWDAT	MSGLST	RXIE	TXIE	ITPND	RMTEN	TXRQST
1	Appl	Appl	Appl	1	0	0	0	Appl	0	0	0	0

The Identifier Registers (ID and XTD bits of CAN_IRx, P8-84) are given by the application. They define the identifier and type of accepted received messages. If an 11-bit Identifier (“Standard Frame”) is used, it is programmed to ID[28:18], ID[17:0] can then be disregarded. When a Data Frame with an 11-bit Identifier is received, ID[17:0] will be set to ‘0’.

If the RXIE bit is set, the ITPND bit will be set when a received Data Frame is accepted and stored in the Message Object.

The Data Length Code (DLC of CAN_MCRx, P8-86) is given by the application. When the Message Handler stores a Data Frame in the Message Object, it will store the received Data Length Code and eight data bytes. If the Data Length Code is less than 8, the remaining bytes of the Message Object will be overwritten by non specified values.

The Mask Registers (UMASK, MXTD, and MMDIR bits) may be used (UMASK='1') to allow groups of Data Frames with similar identifiers to be accepted. For details see “Reception of data frame”(P8-36). The MDIR bit should not be masked in typical applications.

3.3.6 Handling of Received Message

The CPU may read a received message any time while the CAN is not transmitting a message — check TS flag in CAN_SR register (P8-66) — via the IFx Interface registers, the data consistency is guaranteed by the Message Handler state machine.

The CPU will write to the Transfer Management Register. This write will transfer the whole received message from the Message RAM into the Message Buffer Register. Additionally, the bits NEWDAT and ITPND are cleared in the Message RAM (not in the Message Buffer).

If the Message Object uses masks for acceptance filtering, the arbitration bits show which of the matching messages has been received. The actual value of NEWDAT shows whether a new message has been received since last time this Message Object was read.

The actual value of MSGLST shows whether more than one message has been received since last time this Message Object was read. MSGLST will not be automatically reset.

By means of a Remote Frame, the CPU may request another CAN node to provide new data for a receive object. Setting the TXRQST bit of a receive object will cause the transmission of a Remote Frame with the receive object's identifier. (See P8-37)

This Remote Frame triggers the other CAN node to start the transmission of the matching Data Frame. If the matching Data Frame is received before the Remote Frame could be transmitted, the TXRQST bit is automatically reset.

3.3.7 Configuration of a FIFO Buffer

With the exception of the OVERWRITE of CAN_MCRx (P8-86), the configuration of Receive Objects belonging to a FIFO Buffer is the same as the configuration of a (single) Receive Object, see section "Configuration of a Receive object"(P8-38).

To concatenate two or more Message Objects into a FIFO Buffer, the identifiers and masks (if used) of these Message Objects have to be programmed to matching values. Due to the implicit priority of the Message Objects, the Message Object with the lowest number will be the first Message Object of the FIFO Buffer. The OVERWRITE bit of all Message Objects of a FIFO Buffer except the last have to be programmed to zero. The OVERWRITE bits of the last Message Object of a FIFO Buffer is set to one, configuring it as the End of the Block.

3.3.8 Reception of messages with FIFO Buffers

Received messages with identifiers matching to a FIFO Buffer are stored into a Message Object of this FIFO Buffer starting with the Message Object with the lowest message number.

When a message is stored into a Message Object of a FIFO Buffer the NEWDAT bit of this Message Object is set. By setting NEWDAT while OVERWRITE is zero the Message Object is locked for further write accesses by the Message Handler until the CPU has written the NEWDAT bit back to zero.

Messages are stored into a FIFO Buffer until the last Message Object of this FIFO Buffer is reached. If none of the preceding Message Objects is released by writing NEWDAT to zero, all further messages for this FIFO Buffer will be written into the last Message Object of the FIFO Buffer and therefore overwrite previous messages.

Reading from a FIFO Buffer

When the CPU transfers the contents of Message Object to the IFx Message Buffer registers by writing its number to the IFx Transfer Management Register(CAN_TMRx, See P8-79), the bits NEWDAT and ITPND should be reset to zero (TRND = '1' and CLRIT of CAN_TMRx = '1'). The values of these bits in the Message Control Register always reflect the status before resetting the bits.

To assure the correct function of a FIFO Buffer, the CPU should read out the Message Objects starting at the FIFO Object with the lowest message number.

The figure below shows how a set of Message Objects which are concatenated to a FIFO Buffer can be handled by the CPU.

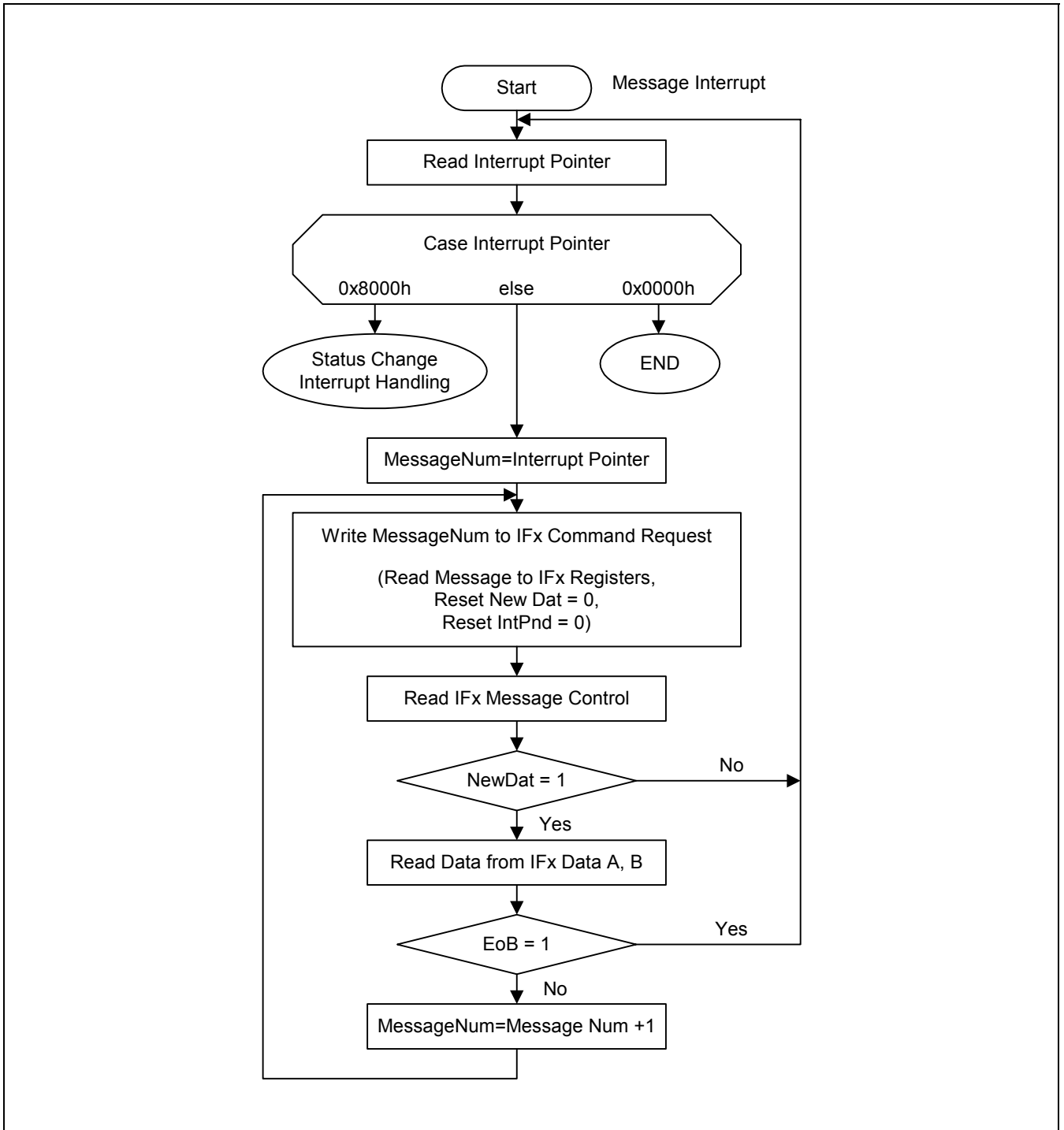


Figure 8-20. CPU Handling of a FIFO Buffer

3.3.9 Handling of Interrupts

A message interrupt is cleared by clearing the Message Object's ITPND bit. The Status Interrupt is cleared by writing the Clear Status Register.

The CPU controls whether a change of the Status Register may cause an interrupt with the Interrupt Mask Register and bits TXIE and RXIE of IFx Message Control Register(CAN_TMRx, See P8-79),

3.3.10 Configuration of the bit Timing

Even if minor errors in the configuration of the CAN bit timing do not result in immediate failure, the performance of a CAN network can be reduced significantly.

In many cases, the CAN bit synchronization will amend a faulty configuration of the CAN bit timing to such a degree that only occasionally an error frame is generated. In the case of arbitration however, when two or more CAN nodes simultaneously try to transmit a frame, a misplaced sample point may cause one of the transmitters to become error passive.

The analysis of such sporadic errors requires a detailed knowledge of the CAN bit synchronization inside a CAN node and of the CAN nodes' interaction on the CAN bus.

3.3.10.1 Bit Time and Bit Rate

CAN supports bit rates in the range of lower than 1 kBit/s up to 1000 kBit/s. Each member of the CAN network has its own clock generator, usually a quartz oscillator. The timing parameter of the bit time (i.e. the reciprocal of the bit rate) can be configured individually for each CAN node, creating a common bit rate even though the CAN nodes' oscillator periods(f_{osc}) maybe different.

The frequencies of these oscillators are not absolutely stable. Small variations are caused by changes in temperature or voltage and by deteriorating components. As long as the variations remain inside a specific oscillator tolerance range(df), the CAN nodes are able to compensate for the different bit rates by resynchronizing to the bit stream.

According to the CAN specification, the bit time is divided into four segments (see Figure 8-21). The Synchronization Segment, the Propagation Time Segment, the Phase Buffer Segment 1, and the Phase Buffer Segment 2. Each segment consists of a specific, programmable number of time quanta (see Table 8-10). The length of the time quantum(t_q), which is the basic time unit of the bit time, is defined by the CAN controller's system clock f_{sys} and the Baud Rate Pre-scalar : $t_q = (BD+1) / f_{sys}$. The CAN's system clock f_{sys} is the frequency of its CAN_CLK input.

The Synchronization Segment Sync_Seg is that part of the bit time where edges of the CAN bus level are expected to occur; the distance between an edge that occurs outside of Sync_Seg and the Sync_Seg is called the phase error of that edge. The Propagation Time Segment Prop_Seg is intended to compensate for the physical delay times within the CAN network. The Phase Buffer Segments Phase_Seg1 and Phase_Seg2 surround the Sample Point. The (Re-)Synchronization Jump Width (SJW) defines how far a resynchronization may move the Sample Point inside the limits defined by the Phase Buffer Segments to compensate for edge phase errors.

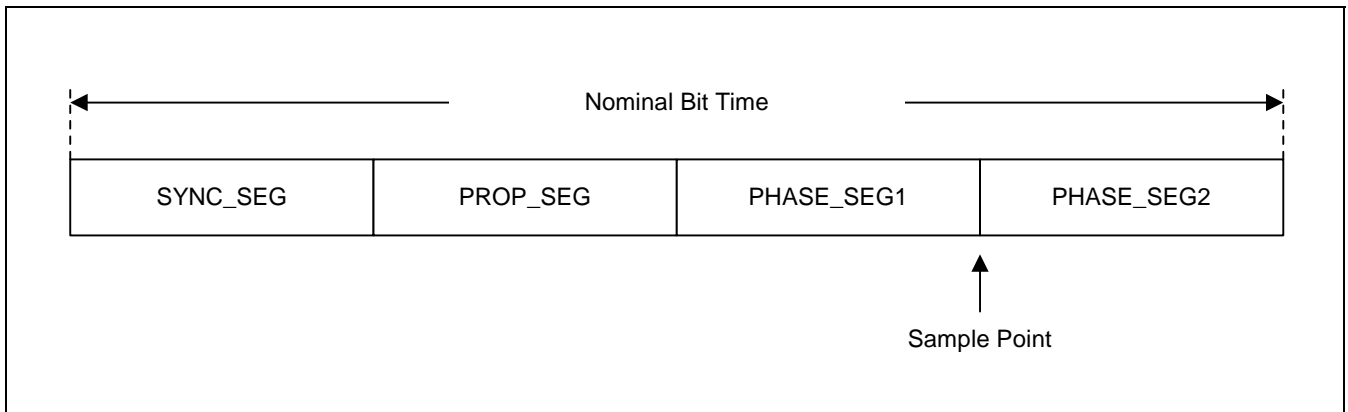


Figure 8-21. Bit Timing

Table 8-10. Parameters of the CAN Bit Time

Parameter	Range	Remark
BRP	[1..32]	Defines the length of the time quantum tq
Sync_Seg	1 tq	Fixed length, synchronization of bus input to system clock
Prop_Seg	[1..8] tq	Compensates for the physical delay times
Phase_Seg1	[1..8] tq	May be lengthened temporarily by synchronization
Phase_Seg2	[1..8] tq	May be shortened temporarily by synchronization
SJW	[1..8] tq	May not be longer than either phase buffer segment
This table describes the minimum programmable ranges required by the CAN protocol		

A given bit rate may be met by different bit time configurations, but for the proper function of the CAN network the physical delay times and the oscillator's tolerance range have to be considered.

3.3.10.2 Propagation Time Segment

This part of the bit time is used to compensate physical delay times within the network. These delay times consist of the signal propagation time on the bus and the internal delay time of the CAN nodes.

Any CAN node synchronized to the bit stream on the CAN bus will be out of phase with the transmitter of that bit stream, caused by the signal propagation time between the two nodes. The CAN protocol's non-destructive bitwise arbitration and the dominant acknowledge bit provided by receivers of CAN messages require that a CAN node transmitting a bit stream must also be able to receive dominant bits transmitted by other CAN nodes that are synchronized to that bit stream. The example in below shows the phase shift and propagation times between two CAN nodes.

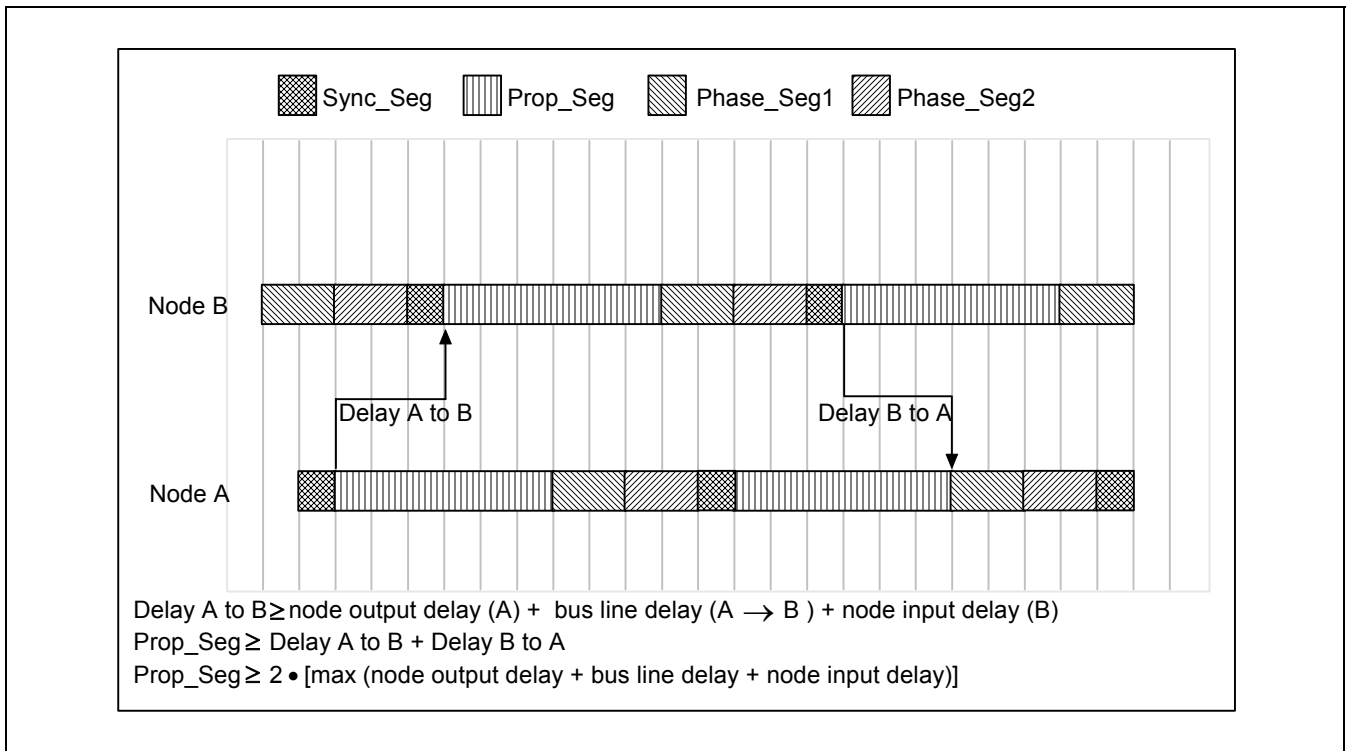


Figure 8-22. Propagation Time Segment

In this example, both nodes A and B are transmitters performing an arbitration for the CAN bus. The node A has sent its Start of Frame bit less than one bit time earlier than node B, therefore node B has synchronized itself to the received edge from recessive to dominant. Since node B has received this edge delay(A to B) after it has been transmitted, B’s bit timing segments are shifted with regard to A. Node B sends an identifier with higher priority and so it will win the arbitration at a specific identifier bit when it transmits a dominant bit while node A transmits a recessive bit. The dominant bit transmitted by node B will arrive at node A after the delay (B to A).

Due to oscillator tolerances, the actual position of node A’s Sample Point can be anywhere inside the nominal range of node A’s Phase Buffer Segments, so the bit transmitted by node B must arrive at node A before the start of Phase_Seg1. This condition defines the length of Prop_Seg.

If the edge from recessive to dominant transmitted by node B would arrive at node A after the start of Phase_Seg1, it could happen that node A samples a recessive bit instead of a dominant bit, resulting in a bit error and the destruction of the current frame by an error flag.

The error occurs only when two nodes arbitrate for the CAN bus that have oscillators of opposite ends of the tolerance range and that are separated by a long bus line; this is an example of a minor error in the bit timing configuration (Prop_Seg to short) that causes sporadic bus errors.

Some CAN implementations provide an optional 3 Sample Mode. The CAN does not. In this mode, the CAN bus input signal passes a digital low-pass filter, using three samples and a majority logic to determine the valid bit value. This results in an additional input delay of $1 t_q$, requiring a longer Prop_Seg.

3.3.10.3 Phase Buffer Segments and Synchronization

The Phase Buffer Segments (Phase_Seg1 and Phase_Seg2) and the Synchronization Jump Width (SJW) are used to compensate for the oscillator tolerance. The Phase Buffer Segments may be lengthened or shortened by synchronization.

Synchronizations occur on edges from recessive to dominant, their purpose is to control the distance between edges and Sample Points.

Edges are detected by sampling the actual bus level in each time quantum and comparing it with the bus level at the previous Sample Point. A synchronization may be done only if a recessive bit was sampled at the previous Sample Point and if the actual time quantum's bus level is dominant.

An edge is synchronous if it occurs inside of Sync_Seg, otherwise the distance between edge and the end of Sync_Seg is the edge phase error, measured in time quanta. If the edge occurs before Sync_Seg, the phase error is negative, else it is positive.

Two types of synchronization exist : Hard Synchronization and Resynchronization. A Hard Synchronization is done once at the start of a frame; inside a frame only Resynchronizations occur.

- *Hard Synchronization*

After a hard synchronization, the bit time is restarted with the end of Sync_Seg, regardless of the edge phase error.

Thus hard synchronization forces the edge which has caused the hard synchronization to lie within the synchronization segment of the restarted bit time.

- *Resynchronization*

Resynchronization leads to a shortening or lengthening of the bit time such that the position of the sample point is shifted with regard to the edge.

When the phase error of the edge which causes Resynchronization is positive, Phase_Seg1 is lengthened. If the magnitude of the phase error is less than SJW, Phase_Seg1 is lengthened by the magnitude of the phase error, else it is lengthened by SJW.

When the phase error of the edge which causes Resynchronization is negative, Phase_Seg2 is shortened. If the magnitude of the phase error is less than SJW, Phase_Seg2 is shortened by the magnitude of the phase error, else it is shortened by SJW.

When the magnitude of the phase error of the edge is less than or equal to the programmed value of SJW, the results of Hard Synchronization and Resynchronization are the same. If the magnitude of the phase error is larger than SJW, the Resynchronization cannot compensate the phase error completely, an error of (phase error - SJW) remains.

Only one synchronization may be done between two Sample Points. The Synchronizations maintain a minimum distance between edges and Sample Points, giving the bus level time to stabilize and filtering out spikes that are shorter than (Prop_Seg + Phase_Seg1).

Apart from noise spikes, most synchronizations are caused by arbitration. All nodes synchronize “hard” on the edge transmitted by the “leading” transceiver that started transmitting first, but due to propagation delay times, they cannot become ideally synchronized. The “leading” transmitter does not necessarily win the arbitration, therefore the receivers have to synchronize themselves to different transmitters that subsequently “take the lead” and that are differently synchronized to the previously “leading” transmitter. The same happens at the acknowledge field, where the transmitter and some of the receivers will have to synchronize to that receiver that “takes the lead” in the transmission of the dominant acknowledge bit.

Synchronizations after the end of the arbitration will be caused by oscillator tolerance, when the differences in the oscillator’s clock periods of transmitter and receivers sum up during the time between synchronizations (at most ten bits). These summarized differences may not be longer than the SJW, limiting the oscillator’s tolerance range. The examples in the figure below show how the Phase Buffer Segments are used to compensate for phase errors.

There are three drawings of each two consecutive bit timings. The upper drawing shows the synchronization on a “late” edge, the lower drawing shows the synchronizations on an “early” edge, and the middle drawing is the reference without synchronization.

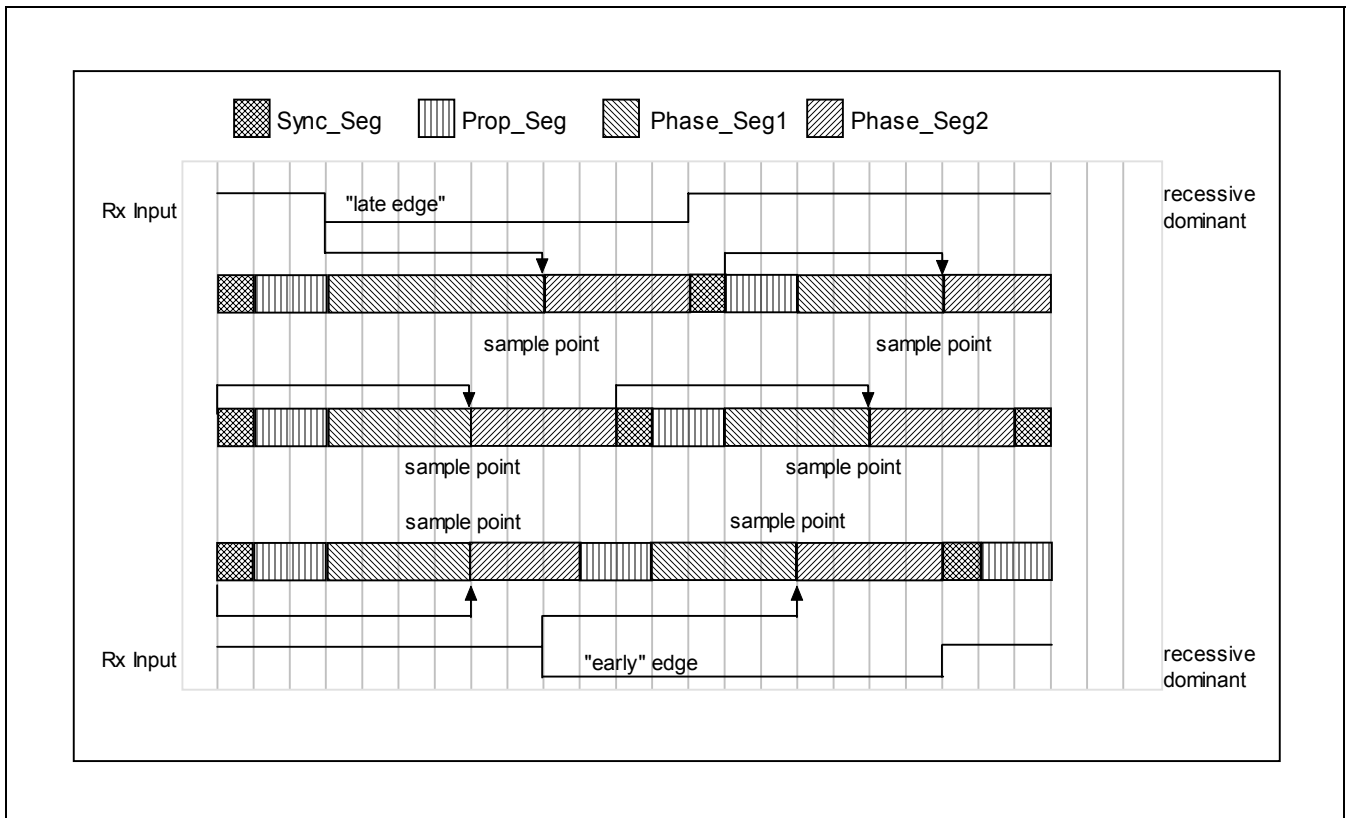


Figure 8-23. Synchronization on Late and Early Edges

In the first example an edge from recessive to dominant occurs at the end of Prop_Seg. The edge is “late” since it occurs after the Sync_Seg. Reacting to the “late” edge, Phase_Seg1 is lengthened so that the distance from the edge to the Sample Point is the same as it would have been from the Sync_Seg to the Sample Point if no edge had occurred. The phase error of this “late” edge is less than SJW, so it is fully compensated and the edge from dominant to recessive at the end of the bit, which is one nominal bit time long, occurs in the Sync_Seg.

In the second example an edge from recessive to dominant occurs during Phase_Seg2. The edge is “early” since it occurs before a Sync_Seg. Reacting to the “early” edge, Phase_Seg2 is shortened and Sync_Seg is omitted, so that the distance from the edge to the Sample Point is the same as it would have been from an Sync_Seg to the Sample Point if no edge had occurred. As in the previous example, the magnitude of this “early” edge’s phase error is less than SJW, so it is fully compensated.

The Phase Buffer Segments are lengthened or shortened temporarily only; at the next bit time, the segments return to their nominal programmed values.

In these examples, the bit timing is seen from the point of view of the CAN implementation’s state machine, where the bit time starts and ends at the Sample Points. The state machine omits Sync_Seg when synchronizing on an “early” edge because it cannot subsequently redefine that time quantum of Phase_Seg2 where the edge occurs to be the Sync_Seg.

The examples in the next figure show how short dominant noise spikes are filtered by synchronizations. In both examples the spike starts at the end of Prop_Seg and has the length of (Prop_Seg + Phase_Seg1).

In the first example, the Synchronization Jump Width is greater than or equal to the phase error of the spike’s edge from recessive to dominant. Therefore the Sample Point is shifted after the end of the spike; a recessive bus level is sampled.

In the second example, SJW is shorter than the phase error, so the Sample Point cannot be shifted far enough ; the dominant spike is sampled as actual bus level.

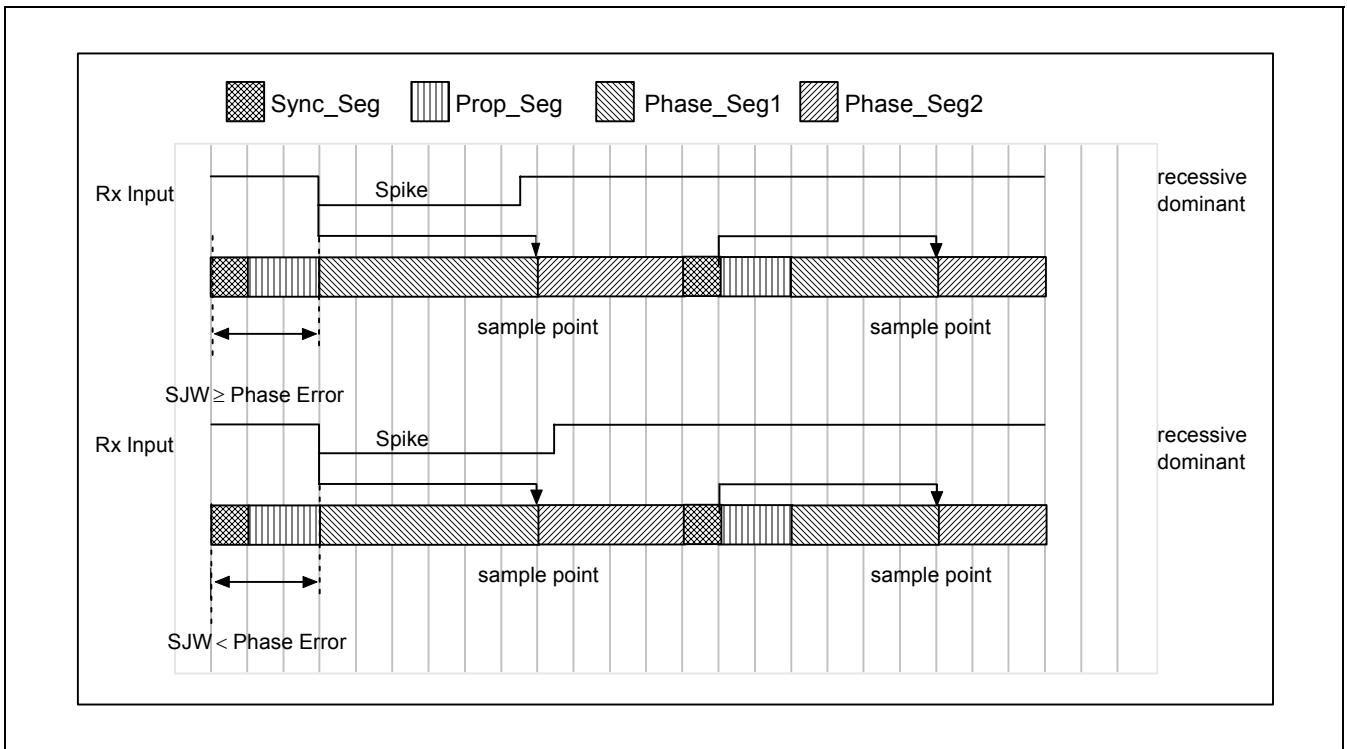


Figure 8-24. Filtering of Short Dominant Spikes

3.3.10.4 Oscillator Tolerance Range

The oscillator tolerance range was increased when the CAN protocol was developed from version 1.1 to version 1.2 (version 1.0 was never implemented in silicon). The option to synchronize on edges from dominant to recessive became obsolete, only edges from recessive to dominant are considered for synchronization. The only CAN controllers to implement protocol version 1.1 have been Intel 82526 and Philips 82C200, both are superseded by successor products.

The protocol update to version 2.0 (A and B) had no influence on the oscillator tolerance.

The tolerance range df for an oscillator's frequency f_{osc} around the nominal frequency f_{nom} with $(1 - df) \cdot f_{nom} \leq f_{osc} \leq (1 + df) \cdot f_{nom}$ depends on the proportions of Phase_Seg1, Phase_Seg2, SJW, and the bit time. The maximum tolerance df is defined by two conditions (both shall be met) :

$$I : df \leq [\min(\text{Phase_Seg1}, \text{Phase_Seg2}) / 2 \cdot (13 \cdot \text{bit_time} - \text{Phase_Seg2})]$$

$$I : df \leq [\text{SJW} / 20 \cdot \text{bit_time}]$$

It has to be considered that SJW may not be larger than the smaller of the Phase Buffer Segments and that the Propagation Time Segment limits that part of the bit time that may be used for the Phase Buffer Segments.

The combination Prop_Seg = 1 and Phase_Seg1 = Phase_Seg2 = SJW = 4 allows the largest possible oscillator tolerance of 1.58%. This combination with a Propagation Time Segment of only 10% of the bit time is not suitable for short bit times; it can be used for bit rates of up to 125 kBit/s (bit time = 8 ms) with a bus length of 40 m.

3.3.10.5 Configuration of the CAN Protocol Controller

In most CAN implementations and also in this CAN, the bit timing configuration is programmed in two register bytes.

The sum of Prop_Seg and Phase_Seg1 (as TSEG1) is combined with Phase_Seg2 (as TSEG2) in one byte, SJW and BRP are combined in the other byte (see the figure below).

In these bit timing registers, the four components TSEG1, TSEG2, SJW, and BRP have to be programmed to a numerical value that is one less than its functional value; so instead of values in the range of [1..n], values in the range of [0..n-1] are programmed. That way, e.g. SJW (functional range of [1..4]) is represented by only two bits.

Therefore the length of the bit time is (programmed values) $[\text{TSEG1} + \text{TSEG2} + 3] \cdot t_q$ or (functional values) $[\text{Sync_Seg} + \text{Prop_Seg} + \text{Phase_Seg1} + \text{Phase_Seg2}] \cdot t_q$.

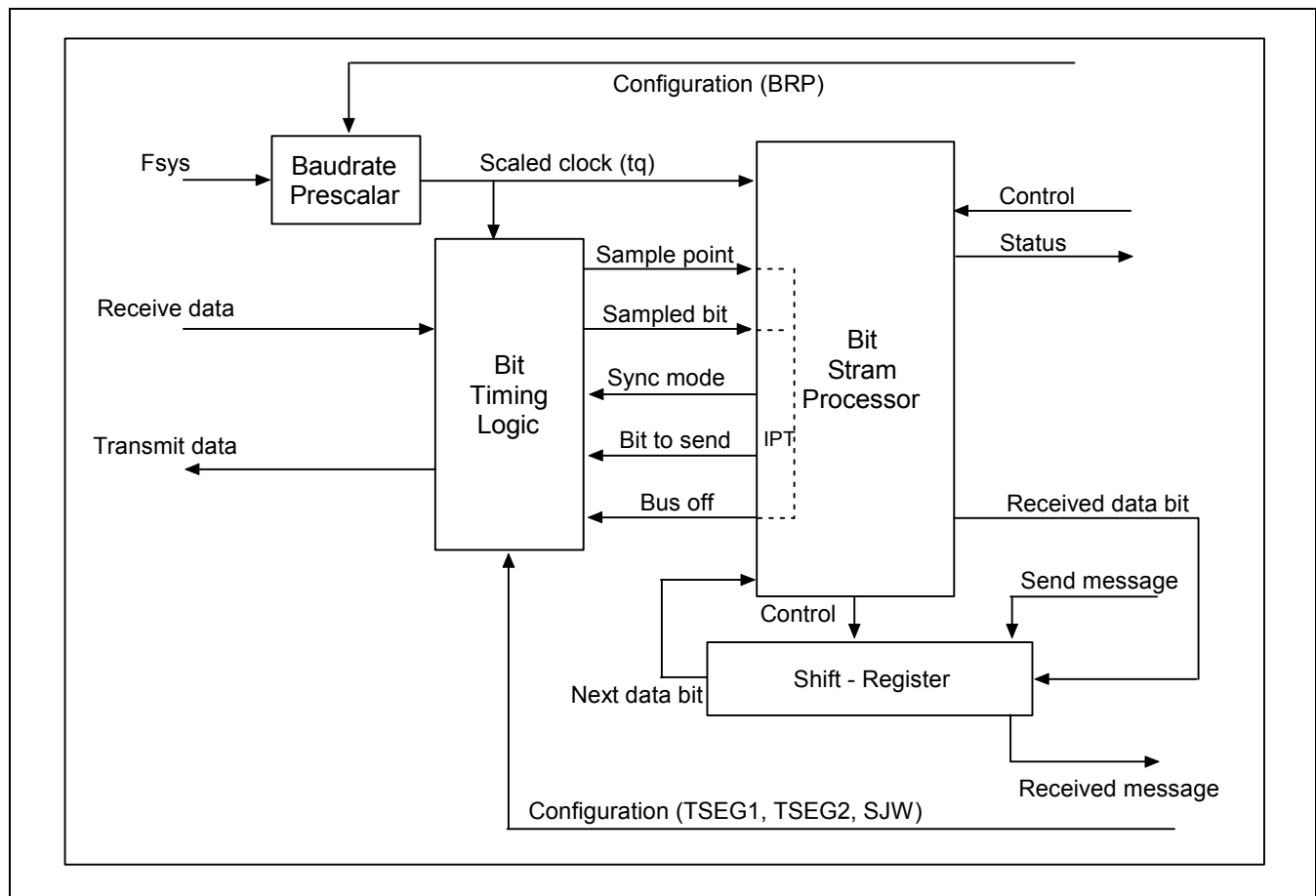


Figure 8-25. Structure of the CAN Controller Core

The data in the bit timing registers are the configuration input of the CAN protocol controller. The Baud Rate Prescaler (configured by BRP) defines the length of the time quantum, the basic time unit of the bit time; the Bit Timing Logic (configured by TSEG1, TSEG2, and SJW) defines the number of time quanta in the bit time.

The processing of the bit time, the calculation of the position of the Sample Point, and occasional synchronizations are controlled by the BTL state machine, which is evaluated once each time quantum. The rest of the CAN protocol controller, the Bit Stream Processor (BSP) state machine is evaluated once each bit time, at the Sample Point.

The Shift Register serializes the messages to be sent and parallelizes received messages. Its loading and shifting is controlled by the BSP.

The BSP translates messages into frames and vice versa. It generates and discards the enclosing fixed format bits, inserts and extracts stuff bits, calculates and checks the CRC code, performs the error management, and decides which type of synchronization is to be used. It is evaluated at the Sample Point and processes the sampled bus input bit. The time after the Sample point that is needed to calculate the next bit to be sent (e.g. data bit, CRC bit, stuff bit, error flag, or idle) is called the Information Processing Time (IPT).

The IPT is application specific but may not be longer than $2 t_q$; this CAN's IPT is $0 t_q$. Its length is the lower limit of the programmed length of Phase_Seg2. In case of a synchronization, Phase_Seg2 may be shortened to a value less than IPT, which does not affect bus timing.

3.3.10.6 Calculation of the Bit Timing Parameters

Usually, the calculation of the bit timing configuration starts with a desired bit rate or bit time. The resulting bit time (1/bit rate) must be an integer multiple of the system clock period.

The bit time may consist of 4 to 25 time quanta, the length of the time quantum t_q is defined by the Baud Rate Pre-scalar with $T_Q = (\text{Baud Rate Pre-scalar} + 1) / f_{\text{sys}}$. Several combinations may lead to the desired bit time, allowing iterations of the following steps.

First part of the bit time to be defined is the Prop_Seg. Its length depends on the delay times measured in the system. A maximum bus length as well as a maximum node delay has to be defined for expandable CAN bus systems. The resulting time for Prop_Seg is converted into time quanta (rounded up to the nearest integer multiple of T_Q).

The Sync_Seg is $1 t_q$ long (fixed), leaving $(\text{bit time} - \text{Prop_Seg} - 1) T_Q$ for the two Phase Buffer Segments. If the number of remaining T_Q is even, the Phase Buffer Segments have the same length, $\text{Phase_Seg2} = \text{Phase_Seg1}$, else $\text{Phase_Seg2} = \text{Phase_Seg1} + 1$.

The minimum nominal length of Phase_Seg2 has to be regarded as well. Phase_Seg2 may not be shorter than the CAN controller's Information Processing Time, which is, depending on the actual implementation, in the range of $[0..2] T_Q$.

The length of the Synchronization Jump Width is set to its maximum value, which is the minimum of 4 and Phase_Seg1.

The oscillator tolerance range necessary for the resulting configuration is calculated by the formulas given in section "Oscillator Tolerance range".

If more than one configuration is possible, that configuration allowing the highest oscillator tolerance range should be chosen.

CAN nodes with different system clocks require different configurations to come to the same bit rate. The calculation of the propagation time in the CAN network, based on the nodes with the longest delay times, is done once for the whole network.

The CAN system's oscillator tolerance range is limited by that node with the lowest tolerance range.

The calculation may show that bus length or bit rate have to be decreased or that the oscillator frequencies' stability has to be increased in order to find a protocol compliant configuration of the CAN bit timing.

3.3.10.6.1 Example for bit timing at high Baud rate

In this example, the frequency of CAN module clock is 10 MHz, BD(Baud rate prescalar) is 0, the bit rate is fixed at 1M Bit/s.

$$T_Q = t_{PCLK} = 100 \text{ ns}$$

$$\text{Bit time} = 1 \text{ us} = 10 \times T_Q$$

Delay of bus driver : 50 ns

Delay of receiver circuit : 30 ns

Delay of bus line (40m) : 220 ns

$$\text{Prog_Seg} = 2 \times (\text{Delay of bus driver}) + 2 \times (\text{Delay of receiver circuit}) + 2 \times (\text{Delay of bus line}) = 600 \text{ ns} = 6 \times T_Q$$

The Sync_Seg is 1 T_Q long(fixed), leaving (bit time – Prop_Seg – 1) t_Q for the two Phase Buffer Segments. If the number of remaining t_Q is even, the Phase Buffer Segments have the same length, Phase_Seg2 = Phase_Seg1, else Phase_Seg2 = Phase_Seg1 + 1

$$T_{\text{Sync_Seg}} = 1 \times T_Q$$

T_Q remaining = 3, then

$$\text{Phase buffer Seg 1} = 1 \times T_Q$$

$$\text{Phase buffer Seg 2} = 2 \times T_Q$$

So :

$$T_{\text{PHS1}} = \text{Prop_Seg} + \text{Phase buffer Seg 1} = 7 \times T_Q$$

$$T_{\text{PHS2}} = \text{Phase buffer Seg 2} = 2 \times T_Q$$

$$\text{Bit time } 1 \text{ us} = T_{\text{Sync_Seg}} + T_{\text{PHS1}} + T_{\text{PHS2}}$$

$$T_{\text{SJW}} = \min(4, \text{Phase buffer Seg 1}) = 1 \times T_Q$$

$$\text{PHSEG2}[2:0] = (T_{\text{PHS2}} / T_Q) - 1 = 1$$

$$\text{PHSEG1}[3:0] = (T_{\text{PHS1}} / T_Q) - 1 = 6$$

AR = 1 (to be compliant with CAN standard)

$$\text{SJW}[1:0] = (T_{\text{SJW}} / T_Q) - 1 = 0$$

$$\text{BD} [9:0] = 0$$

$$\text{CAN_MR} = 0x00164000$$

3.3.10.6.2 Example for bit timing at low Baud rate

In this example, the frequency of CAN module clock is 2 MHz, BD(baud rate prescalar) is 1, the bit rate is fixed at 100K Bit/s.

$$T_Q = 2 \times t_{PCLK} = 1 \text{ us}$$

$$\text{Bit time} = 10 \text{ us} = 10 \times T_Q$$

Delay of bus driver : 200 ns

Delay of receiver circuit : 80 ns

Delay of bus line (40m) : 220 ns

$$\text{Prog_Seg} = 2 \times (\text{Delay of bus driver}) + 2 \times (\text{Delay of receiver circuit}) + 2 \times (\text{Delay of bus line}) = 1 \text{ us} = T_Q$$

The Sync_Seg is 1 tq long(fixed), leaving (bit time – Prop_Seg – 1) tq for the two Phase Buffer Segments. If the number of remaining tq is even, the Phase Buffer Segments have the same length, Phase_Seg2 = Phase_Seg1, else Phase_Seg2 = Phase_Seg1 + 1

$$T_{\text{Sync_Seg}} = 1 \text{ us} = 1 \times T_Q$$

T_Q remaining = 8, then

$$\text{Phase buffer Seg 1} = 4 \times T_Q$$

$$\text{Phase buffer Seg 2} = 4 \times T_Q$$

So :

$$T_{\text{PHS1}} = \text{Prop_Seg} + \text{Phase buffer Seg 1} = 5 \times T_Q$$

$$T_{\text{PHS2}} = \text{Phase buffer Seg 2} = 4 \times T_Q$$

$$\text{Bit time } 10 \text{ us} = T_{\text{Sync_Seg}} + T_{\text{PHS1}} + T_{\text{PHS2}}$$

$$T_{\text{SJW}} = \min(4, \text{Phase buffer Seg 1}) = 4 \times T_Q$$

$$\text{PHSEG2}[2:0] = (T_{\text{PHS2}} / T_Q) - 1 = 3$$

$$\text{PHSEG1}[3:0] = (T_{\text{PHS1}} / T_Q) - 1 = 4$$

AR = 1 (to be compliant with CAN standard)

$$\text{SJW}[1:0] = (T_{\text{SJW}} / T_Q) - 1 = 3$$

$$\text{BD}[9:0] = 1$$

$$\text{CAN_MR} = 0x00347001$$

3.3.3.11 Time stamp

A 32.bit stamp allows to date all messages received.

The 32.bit register forming the second counter in the Stamp Timer (STT) is provided to the CAN module. After each reception of a CAN frame, the value of the current second counter will be automatically written in the corresponding CAN channel CAN_STPRX register.

4. REGISTERS DESCRIPTION

Base Addresses – CAN0 : 0xFFE18000
 CAN1 : 0xFFE3C000
 CAN2 : 0xFFE40000
 CAN3 : 0xFFE44000

Table 8-11. CAN Special Function Register

Offset Address	Name	Description	R/W	Reset State
0x000 – 0x04C	–	Reserved	–	–
0x050	CAN_ECR	Enable clock register	W	–
0x054	CAN_DCR	Disable clock register	W	–
0x058	CAN_PMSR	Power management status register	R	0XXXXXXXX0
0x05C	–	Reserved	–	–
0x060	CAN_CR	Control register	W	–
0x064	CAN_MR	Mode register	R/W	0x00234001
0x068	–	Reserved	–	–
0x06C	CAN_CSR	Clear status register	W	–
0x070	CAN_SR	Status register	R	(note)
0x074	CAN_IER	Interrupt enable register	W	–
0x078	CAN_IDR	Interrupt disable register	W	–
0x07C	CAN_IMR	Interrupt mask register	R	0x00000000
0x080	–	Reserved	–	–
0x084	CAN_ISSR	Interrupt source status register	R	0x00000000
0x088	CAN_SIER	Source interrupt enable register	W	–
0x08C	CAN_SIDR	Source interrupt disable register	W	–
0x090	CAN_SIMR	Source interrupt mask register	R	0x00000000
0x094	CAN_HPIR	Highest priority interrupt register	R	0x00000000
0x098	CAN_ERCR	Error counter register	R	0x00000000
0x09C – 0x0FC	–	Reserved	–	–
0x100	CAN_TMR0	Interface 0 transfer management register	R/W	0x00000001
0x104	CAN_DAR0	Interface 0 data A register	R/W	0x00000000
0x108	CAN_DBR0	Interface 0 data B register	R/W	0x00000000
0x10C	CAN_MSKR0	Interface 0 mask register	R/W	0xDFFFFFFF

Table 8-11. CAN Special Function Register (Continued)

Offset Address	Name	Description	R/W	Reset State
0x110	CAN_IR0	Interface 0 identifier register	R/W	0x00000000
0x114	CAN_MCR0	Interface 0 message control register	R/W	0x00000000
0x118	CAN_STPR0	Interface 0 stamp register	R	0x00000000
0x10C	–	Reserved	–	–
0x120	CAN_TMR1	Interface 1 transfer management register	R/W	0x00000001
0x124	CAN_DAR1	Interface 1 data A register	R/W	0x00000000
0x128	CAN_DBR1	Interface 1 data B register	R/W	0x00000000
0x12C	CAN_MSKR1	Interface 1 mask register	R/W	0xDFFFFFFF
0x130	CAN_IR1	Interface 1 Identifier register	R/W	0x00000000
0x134	CAN_MCR1	Interface 1 message control register	R/W	0x00000000
0x138	CAN_STPR1	Interface 1 stamp register	R	0x00000000
0x13C	–	Reserved	–	–
0x140	CAN_TRR	Transmission request register	R	0x00000000
0x144	CAN_NDR	New data register	R	0x00000000
0x148	CAN_MVR	Message valid register	R	0x00000000
0x14C	CAN_TSTR	Test register	R/W	(note)

NOTE: The value of this depends on CAN_RX pin value.

CAN Enable Clock Register		CAN_ECR (0x050)						Access: Write only	
31	30	29	28	27	26	25	24		
DBGEN	–	–	–	–	–	–	–		
W	W	W	W	W	W	W	W		
23	22	21	20	19	18	17	16		
–	–	–	–	–	–	–	–		
W	W	W	W	W	W	W	W		
15	14	13	12	11	10	9	8		
–	–	–	–	–	–	–	–		
W	W	W	W	W	W	W	W		
7	6	5	4	3	2	1	0		
–	–	–	–	–	–	CAN	–		
W	W	W	W	W	W	W	W		

- **CAN : CAN clock enable**

0: No effect

1: Enable CAN clock

- **DBGEN : Debug mode enable**

0: No effect

1: Enable debug mode

CAN Disable Clock Register

CAN_DCR (0x054)

Access: Write only

31	30	29	28	27	26	25	24
DBGEN	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	CAN	–
W	W	W	W	W	W	W	W

- **CAN : CAN clock disable**

0: No effect

1: Disable CAN clock

- **DBGEN : Debug mode disable**

0: No effect

1: Disable debug mode

CAN Power Management Status Register **CAN_PMSR (0x058)** **Access: Read only**

31	30	29	28	27	26	25	24
DBGEN	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	CAN	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **CAN : CAN clock status**

- 0: CAN clock disabled.
- 1: CAN clock enabled.

NOTES:

1. The CAN_PMSR register is not reset by software reset.
2. Disabling CAN clock while a transfer is occurring will stop immediately the current transfer. This transfer will finish as soon as the CPU enables CAN clock. To avoid incomplete transfer, software should ensure no transfer is occurring when disabling CAN clock.

- **DBGEN : Debug mode**

- 0: Debug mode is disabled : dbgack_sclk input has no influence on CAN function.
- 1: Debug mode is enabled : when the dbgack_sclk input signal is low, the CAN function is left unchanged. When this signal is active high, the CAN function is frozen, and in case a transfer is in progress, the transfer will be finished safely before the CAN is frozen. However, full read/write access to internal register is kept for debug purpose.

CAN Control Register				CAN_CR (0x060)				Access: Write only
31	30	29	28	27	26	25	24	
–	–	–	–	–	–	–	–	
W	W	W	W	W	W	W	W	
23	22	21	20	19	18	17	16	
–	–	–	–	–	–	–	–	
W	W	W	W	W	W	W	W	
15	14	13	12	11	10	9	8	
–	–	–	–	–	STSR	ABBTX	RQBTX	
W	W	W	W	W	W	W	W	
7	6	5	4	3	2	1	0	
–	–	–	CCDIS	CCEN	CANDIS	CANEN	SWRST	
W	W	W	W	W	W	W	W	

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **SWRST : CAN software reset**

0: No effect.

1: Reset the CAN.

A software reset triggered hardware reset of the CAN is performed. It reset all the registers except the CAN_PMSR. The data stored in the Message RAM is not affected by a hardware reset, except MSGVAL, TXRQST, NEWDAT and ITPND bits for each Message Objects. After power-on, the content of the Message RAM is undefined.

- **CANEN : CAN enable**

0: No effect.

1: Enables the CAN. Enabling the CAN finishes the software initialization, transmitter and receiver state machines are reset.

Afterwards the Bit Stream Processor(BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (Bus Idle) before it can take part in bus activities and starts the message transfer. This delay is part of the CAN standard.

If CAN is disabled because it goes in bus off state, CAN could be enabled again, but it will take part in bus activities after 129 successive occurrences of bus idle (a sequence of 11 consecutive recessive bits) occurred on the bus. The CANENS bit is set only at the end of the 129 occurrence of 11 consecutive recessive bits. The bus off recovery sequence can be monitored by reading the REC counter: it shall be reset when CPU enables the CAN, and each time a sequence of 11 consecutive recessive bits occurred on the bus, the REC counter shall be increased by one.

- **CANDIS : CAN disable**

0: No effect.

1: Disables the CAN. No data will be received or transmitted. Disabling CAN module while module is transferring will stop immediately the current transfer. The status of the CAN bus output CAN_TX is 'recessive' (HIGH). The counter of the ERCCR(See P8-78) is unchanged. Disabling CAN does not change any configuration register.

In case both CANEN and CANDIS are equals to one when the control register is written, the CAN will be disabled.

- **CCEN : Configuration change enable**

0: No effect.

1: Enables the write access on Bit timing bits in CAN_MR register (while CAN is disabled : CANENS = 0 in CAN_SR register).

- **CCDIS : Configuration change disable**

0: No effect.

1: Disables the write access on bit timing bits in CAN_MR register.

In case both CCEN and CCDIS are equals to one when the control register is written, the configuration change will be disabled.

- **RQBTX : Request basic transmission**

0: No effect.

1: Request the transmission of the contents of IF0 registers (Basic mode). This bit is used only in basic mode (BASIC = '1' in CAN_TSTR register).

- **ABBTX : Abort basic transmission**

0: No effect.

1: Abort the transmission of the contents of IF0 registers (Basic mode).

This bit is used only in basic mode (BASIC = '1' in CAN_TSTR register).

In case both RQBTX and ABBTX are equals to one when the control register is written, the transmission is aborted.

- **STSR : Store shift register**

0: No effect.

1: The contents of the shift register is stored into the IF1 Registers (Basic mode). It allows to monitor the actual contents of the shift register.

This bit is used only in basic mode (BASIC = '1' in CAN_TSTR register).

CAN Mode Register		CAN_MR (0x064)						Access: Read/Write
31	30	29	28	27	26	25	24	
–	–	–	–	–	–	–	–	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
23	22	21	20	19	18	17	16	
–	RHSEG2[2:0]			PHSEG1[3:0]				
R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1	
15	14	13	12	11	10	9	8	
–	AR	SJW[1:0]		–	–	BD[9:8]		
R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
7	6	5	4	3	2	1	0	
BD[7:0]								
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

All the bits of this register, except AR bit, can be written only if CCENS (Configuration Change Enable) bit is high and CANENS (CAN Enable Status) bit is low. (See CAN_SR register, P8-63)

- **BD[9:0] : Baud rate pre-scalar**

Used for generating the time quantum (T_Q) using the following formula:

$$T_Q = (BD + 1) / PCLK$$

The bit time is built up from a multiple of this quantum.

- **SJW[1:0] : Synchronization jump width**

The CAN controller re-synchronizes on each edge of the transmission. The (SJW + 1) value defines the maximum number of CAN clock cycles a bit period may be shortened or lengthened.

$$T_{SWJ} = T_Q \times (SJW + 1)$$

- **AR : Automatic Retransmission**

0: No Automatic Retransmission

1: Activate Automatic Retransmission (This bit must be set to 1 to be compliant with the CAN standard.)

- **PHSEG1[3:0] : Phase segment 1 value**

The time segment before the sample point. It is the sum of the Propagation Segment and Phase Buffer Segment 1. Zero is a prohibited value and will be written as a one.

$$T_{PHS1} = T_Q \times (PHSEG1 + 1)$$

- **PHSEG2[2:0] : Phase segment 2 value**

The time segment after the sample point.

$$T_{PHS2} = T_Q \times (PHSEG2 + 1)$$

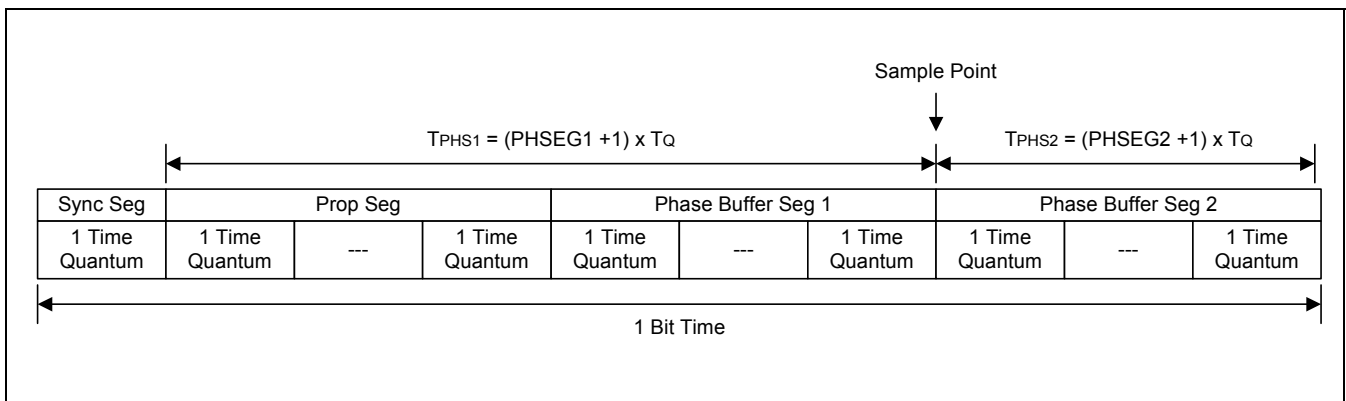


Figure 8-26. CAN Bit Time Built up from Time Quantum

Table 8-12. Bit rate and Minimal Core Frequency

Bit Rate	Minimal PCLK
125 Kbits	1 MHz
250Kbits	2 MHz
500 Kbits	4 MHz
1 Mbits	8 MHz

CAN Clear Status Register				CAN_CSR (0x06C)				Access: Write only
31	30	29	28	27	26	25	24	
–	–	–	–	–	–	–	–	
W	W	W	W	W	W	W	W	
23	22	21	20	19	18	17	16	
–	–	–	–	–	–	–	–	
W	W	W	W	W	W	W	W	
15	14	13	12	11	10	9	8	
CRC	BIT0	BIT1	ACK	FORM	STUFF	TXOK	RXOK	
W	W	W	W	W	W	W	W	
7	6	5	4	3	2	1	0	
–	–	–	ACTVT	BUSOFFTR	ERPASSTR	ERWARNTR	–	
W	W	W	W	W	W	W	W	

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **ERWARNTR : Clear error passive warning transition interrupt**

0: No effect.

1: Clear error passive warning transition interrupt

- **ERPASSTR : Clear error passive transition interrupt**

0: No effect.

1: Clear error passive transition interrupt

- **BUSOFFTR : Clear bus off transition interrupt**

0: No effect.

1: Clear bus off transition interrupt

- **ACTVT : Clear activity interrupt**

0: No effect

1: Clear activity interrupt

- **RXOK : Clear successfully received message interrupt**

0: No effect

1: Clear successfully received message interrupt

- **TXOK : Clear successfully transmit message interrupt**

0: No effect

1: Clear successfully transmit message interrupt

- **STUFF : Clear stuff error interrupt**

0: No effect

1: Clear stuff error interrupt

- **FORM : Clear form error interrupt**

0: No effect

1: Clear form error interrupt

- **ACK : Clear acknowledge error interrupt**

0: No effect

1: Clear acknowledge error interrupt

- **BIT1 : Clear bit to one error interrupt**

0: No effect

1: Clear bit to one error interrupt

- **BIT0 : Clear bit to zero error interrupt**

0: No effect

1: Clear bit to zero error interrupt

- **CRC : Clear CRC error interrupt**

0: No effect

1: Clear CRC error interrupt

CAN Status Register **CAN_SR (0x070)** **Access: Read only**

31	30	29	28	27	26	25	24
-	-	-	-	-	-	BTXPD	CCENS
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
TS	RS	BUSY1	BUSY0	BUSOFF	ERPASS	ERWARN	CANENS
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
CRC	BIT0	BIT1	ACK	FORM	STUFF	TXOK	RXOK
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
-	-	-	ACTVT	BUSOFFTR	ERPASSTR	ERWARNTR	ISS
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

CRC, BIT0, BIT1, ACK, FORM and STUFF bits are mutually exclusives and are updated by the CAN controller at the end of each transfer (reception or transmission).

- **ISS : Interrupt Source Status**

0: No interrupt in any channel.

1: At least one interrupt occurred in a channel and the corresponding bit in Source Interrupt Mask Register (SIMR, P8-76) is enabled (read CAN_ISSR(P8-73) for more information).

- **ERWARNTR : Error passive warning transition**

0: No transition from or to error passive warning occurs since the last reset of this bit.

1: At least one transition from or to error passive warning occurs (one of the error counters reached the error passive warning limit of 96) and this bit has not been reset.

- **ERPASSTR : Error passive transition**

0: No transition between error passive mode since last reset of this bit.

1: At least one transition from or to error passive mode and this bit has not been reset.

- **BUSOFFTR : Bus off transition**

0: No passage in bus off mode since last reset of this bit.

1: At least one transition from or to bus off mode and this bit has not been reset.

- **ACTVT : Activity**

- 0: No activity (no dominant level) has occurred on the CAN_RX pin since the last reset of this bit or CAN clock is enabled.
- 1: Activity (dominant level) has occurred on the CAN_RX pin since the last reset of this bit while CAN clock is disabled (halt mode and / or CAN = '0' in CAN_PMSR register).

- **RXOK : Successfully received a message**

- 0: No message received since last reset of this bit.
- 1: At least one message has been successfully received.

- **TXOK : Successfully transmitted a message**

- 0: No message transmitted since last reset of this bit.
- 1: At least one message has been successfully transmitted.

- **STUFF : Stuff error**

- 0: No stuff error occurred during last message transfer since last reset of this bit.
- 1: During last message transfer, a stuff error occurred – more than five equal bits in a sequence have occurred in a part of a received message where this is not allowed.

- **FORM : Form error**

- 0: No form error occurred during last message transfer since last reset of this bit.
- 1: During last message transfer, a form error occurred : a fixed format part of a received frame has the wrong format.

- **ACK : Acknowledge error**

- 0: No acknowledge error occurred during last message transfer since last reset of this bit.
- 1: During last message transfer, an acknowledge error occurred : the message this CAN transmitted was not acknowledge by another node.

- **BIT1 : Bit to one error**

- 0: No bit to one error occurred during last message transfer since last reset of this bit.
- 1: During last message transfer, a bit to one error occurred : during the transmission of a message, with the exception of the identifier, the device wanted to send a recessive level, but monitored a dominant level.

- **BIT0 : Bit to zero error**

- 0: No bit to zero error occurred during last message transfer since last reset of this bit.
- 1: During last message transfer, a bit to zero error occurred : during the transmission of a message (or acknowledged bit, or active error flag, or overload flag), the device wanted to send a dominant level, but monitored a recessive level. During bus off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the bus off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).

- **CRC : CRC error**

- 0: No CRC error occurred during last message transfer (reception or transmission) since last reset of this bit.
- 1: A CRC error occurred – the CRC check sum was incorrect in the message received – during last message transfer (reception or transmission).

- **CANENS : CAN enable status**

- 0: CAN is disabled. All message transfer from and to CAN bus is stopped, the status of the output pin CAN_TX is recessive and the error counters are unchanged. Disable the CAN does not change any configuration register.
- 1: CAN is enabled. It can take part in bus activities and can start message transfer.

- **ERWARN : Error passive warning**

- 0: No error passive warning.
- 1: At least one of the error counters reached the error passive warning limit of 96.

- **ERPASS : Error passive**

- 0: CAN is not in error passive mode.
- 1: CAN is in error passive mode.

- **BUSOFF : Bus off**

- 0: CAN is not in bus off mode.
- 1: CAN is in bus off mode.

When the CAN controller goes in bus off mode, it becomes disable (CANENS is reset) and does not take part any longer in bus activities. To resume in normal operation mode, CPU has to enable the CAN controller (set CANEN bit in CAN_CR register), at this point the CAN controller will wait for 129 successive occurrences of bus idle (a sequence of 11 consecutive recessive bits). At the end of the bus off recovery sequence, the read and write error counter are reset and the BUSOFF bit is reset. The bus off recovery sequence cannot be shortened by resetting CAN (SWRST set to '1' in CAN_CR).

During the waiting time after CPU has enabled CAN controller, each time a sequence of 11 recessive bits has been monitored, BIT0 bit is set in the status register, enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the proceeding of the bus off recovery sequence.

- **BUSY0 : Busy flag of interface 0**

- 0: Read/write action between interface 0 and Message RAM has finished.
- 1: A transfer – read or write – between interface 0 and Message RAM is in progress.

- **BUSY1 : Busy flag of interface 1**

- 0: Read/write action between interface 1 and Message RAM has finished.
- 1: A transfer – read or write – between interface 1 and Message RAM is in progress.

- **RS : Receive status**

0: If the transmit status is also clear then the CAN controller is idle; otherwise it is in transmit mode.

1: CAN controller entered receive mode from idle, or by losing arbitration during transmission.

NOTE: if CAN controller lost arbitration during transmission, during a CAN clock period, RS and TS bit are set to '1'.

- **TS : Transmit status**

0: If the receive status is also clear then the CAN controller is idle; otherwise it is in receive mode.

1: CAN controller has started to transmit a message.

- **CCENS : Configuration change enable**

0: The CPU has no write access to the bit timing bits in CAN_MR register.

1: The CPU has write access to the bit timing bits in CAN_MR register while CAN is disabled (CANENS = 0).

- **BTXPD : Basic transmission pending**

0: No transmission has been requested in basic mode or the transmission has completed or basic mode has been left.

1: The transmission of the IF0 registers is pending or occurring at that moment.

This bit is used only in basic mode (BASIC = '1' in CAN_TSTR register).

CAN Interrupt Enable Register **CAN_IER (0x074)** **Access: Write only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
CRC	BIT0	BIT1	ACK	FORM	STUFF	TXOK	RXOK
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	ACTVT	BUSOFFTR	ERPASSTR	ERWARNTR	–
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **ERWARNTR : Error passive warning transition interrupt enable**

0: No effect.

1: Enable error passive warning transition interrupt.

- **ERPASSTR : Error passive transition interrupt enable**

0: No effect.

1: Enable error passive transition interrupt.

- **BUSOFFTR : Bus off transition interrupt enable**

0: No effect.

1: Enable bus off transition interrupt.

- **ACTVT : Activity interrupt enable**

0: No effect.

1: Enable activity interrupt.

- **RXOK : Successfully received a message interrupt enable**

0: No effect.

1: Enable successfully received a message interrupt.

- **TXOK : Successfully transmitted a message interrupt enable**

0: No effect.

1: Enable successfully transmit a message interrupt.

- **STUFF : Stuff error interrupt enable**

0: No effect.

1: Enable stuff error interrupt.

- **FORM : Form error interrupt enable**

0: No effect.

1: Enable form error interrupt.

- **ACK : Acknowledge error interrupt enable**

0: No effect.

1: Enable acknowledge error interrupt.

- **BIT1 : Bit to one error interrupt enable**

0: No effect.

1: Enable bit to one error interrupt.

- **BIT0 : Bit to zero error interrupt enable**

0: No effect.

1: Enable bit to zero error interrupt.

- **CRC : CRC error interrupt enable**

0: No effect.

1: Enable CRC error interrupt.

CAN Interrupt Disable Register

CAN_IDR (0x078)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
CRC	BIT0	BIT1	ACK	FORM	STUFF	TXOK	RXOK
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	ACTVT	BUSOFFTR	ERPASSTR	ERWARNTR	–
W	W	W	W	W	W	W	W

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **ERWARNTR : Error passive warning transition interrupt disable**

0: No effect.

1: Disable error passive warning transition interrupt.

- **ERPASSTR : Error passive transition interrupt disable**

0: No effect.

1: Disable error passive transition interrupt.

- **BUSOFFTR : Bus off transition interrupt disable**

0: No effect.

1: Disable bus off transition interrupt.

- **ACTVT : Activity disable interrupt disable**

0: No effect.

1: Disable activity interrupt.

- **RXOK : Successfully received a message interrupt disable**

0: No effect.

1: Disable successfully received a message interrupt.

- **TXOK : Successfully transmitted a message interrupt disable**

0: No effect.

1: Disable successfully transmit a message interrupt.

- **STUFF : Stuff error interrupt disable**

0: No effect.

1: Disable stuff error interrupt

- **FORM : Form error interrupt disable**

0: No effect.

1: Disable form error interrupt.

- **ACK : Acknowledge error interrupt disable**

0: No effect.

1: Disable acknowledge error interrupt.

- **BIT1 : Bit to one error interrupt disable**

0: No effect.

1: Disable bit to one error interrupt.

- **BIT0 : Bit to zero error interrupt disable**

0: No effect.

1: Disable bit to zero error interrupt.

- **CRC : CRC error interrupt disable**

0: No effect.

1: Disable CRC error interrupt.

CAN Interrupt Mask Register

CAN_IMR (0x07C)

Access: Read only

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
CRC	BIT0	BIT1	ACK	FORM	STUFF	TXOK	RXOK
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
-	-	-	ACTVT	BUSOFFTR	ERPASSTR	ERWARNTR	-
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **ERWARNTR : Error passive warning interrupt mask**

0: Error passive warning interrupt is disabled.

1: Error passive warning interrupt is enabled.

- **ERPASSTR : Error passive interrupt mask**

0: Error passive interrupt is disabled.

1: Error passive interrupt is enabled.

- **BUSOFFTR : Bus off interrupt mask**

0: Bus off interrupt is disabled.

1: Bus off interrupt is enabled.

- **ACTVT : Activity interrupt mask**

0: Activity interrupt is disabled.

1: Activity interrupt is enabled.

- **RXOK : Successfully received a message interrupt mask**

0: Successfully received a message interrupt is disabled.

1: Successfully received a message interrupt is enabled.

- **TXOK : Successfully transmitted a message interrupt mask**

0: Successfully transmit a message interrupt is disabled.

1: Successfully transmit a message interrupt is enabled.

- **STUFF : Stuff error interrupt mask**

0: Stuff error interrupt is disabled.

1: Stuff error interrupt is enabled.

- **FORM : Form error interrupt mask**

0: Form error interrupt is disabled.

1: Form error interrupt is enabled.

- **ACK : Acknowledge error interrupt mask**

0: Acknowledge error interrupt is disabled.

1: Acknowledge error interrupt is enabled.

- **BIT1 : Bit to one error interrupt mask**

0: Bit to one error interrupt is disabled.

1: Bit to one error interrupt is enabled.

- **BIT0 : Bit to zero error interrupt mask**

0: Bit to zero error interrupt is disabled.

1: Bit to zero error interrupt is enabled.

- **CRC : CRC error interrupt mask**

0: CRC error interrupt is disabled.

1: CRC error interrupt is enabled.

CAN Interrupt Source Status Register

CAN_ISSR (0x084)

Access: Read only

31	30	29	28	27	26	25	24
CH32	CH31	CH30	CH29	CH28	CH27	CH26	CH25
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
CH24	CH23	CH22	CH21	CH20	CH19	CH18	CH17
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
CH16	CH15	CH14	CH13	CH12	CH11	CH10	CH9
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write R: Read -0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **CHX : Channel X interrupt**

0: No interrupt occurred on channel X.

1: An interrupt occurred on channel X : ITPND = '1' on channel X.

CAN Source Interrupt Enable Register **CAN_SIER (0x088)** **Access: Write only**

31	30	29	28	27	26	25	24
CH32	CH31	CH30	CH29	CH28	CH27	CH26	CH25
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
CH24	CH23	CH22	CH21	CH20	CH19	CH18	CH17
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
CH16	CH15	CH14	CH13	CH12	CH11	CH10	CH9
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **CHX : Channel X interrupt enable**

0: No effect.

1: Enable Channel X interrupt.

CAN Source Interrupt Disable Register

CAN_SIDR (0x08C)

Access: Write only

31	30	29	28	27	26	25	24
CH32	CH31	CH30	CH29	CH28	CH27	CH26	CH25
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
CH24	CH23	CH22	CH21	CH20	CH19	CH18	CH17
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
CH16	CH15	CH14	CH13	CH12	CH11	CH10	CH9
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **CHX : Channel X interrupt disable**

0: No effect.

1: Disable Channel X interrupt.

CAN Source Interrupt Mask Register **CAN_SIMR (0x090)** **Access: Read only**

31	30	29	28	27	26	25	24
CH32	CH31	CH30	CH29	CH28	CH27	CH26	CH25
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
CH24	CH23	CH22	CH21	CH20	CH19	CH18	CH17
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
CH16	CH15	CH14	CH13	CH12	CH11	CH10	CH9
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **CHX : Channel X interrupt mask**

0: Channel X interrupt is disabled.

1: Channel X interrupt is enabled.

CAN Highest Priority Interrupt Register **CAN_HPIR (0x094)** **Access: Read only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
INTID[15:8]							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
INTID[7:0]							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- INTID[15:0] : Channel X interrupt mask**

0: No interrupt is pending.

0x0001-0x0020 : The number of Message Object which caused the interrupt.

0x0021-0x7FFF : reserved.

0x8000 : Status interrupt : at least one interrupt raised up in status register (except ACTVT and ISS) and while enabled (corresponding bit in IMR is set). Read CAN_SR for more information.

0x8001-0xFFFF : reserved.

If several interrupt are pending, the CAN highest priority interrupt register will point to the pending interrupt with the highest priority, disregarding their chronological order. An interrupt remains pending until the CPU has cleared it. The status interrupt has the highest priority. Among the channel interrupt, the Message object's interrupt priority decreases with increasing message number.

CAN Error Counter Register **CAN_ERCR (0x098)** **Access: Read only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
TEC[7:0]							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
REP	REC[6:0]						
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **REC[6:0] : Reception error counter**

Here is the value of the reception error counter.

- **REP : Receive Error Passive**

0: The receive Error Counter is below the error passive level as defined in the CAN specification.

1: The receive Error Counter has reached the error passive level.

- **TEC[7:0] : Transmit error counter**

Here is the value of the transmit error counter.

**CAN Interface X Transfer Management Register CAN_TMR1 (0x100)
CAN_TMR2 (0x120)**

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
CLRIT	TRND	Reserved	AMCR	AIR	AMSKR	ADBR	ADAR
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
WR	–	NUMBER[5:0]					
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

A message transfer between the Message RAM and the IFx Message buffer registers is started as soon as the CPU has written in this register. With this write operation the BUSYx bit is automatically set to '1' in the CAN_SR, and it is reset to '0' once the transfer between the Interface Registers and the Message RAM has completed. Registers of interface X (DAR, DBR, MSKR, IR and MCR) must not be accessed while the BUSYx bit is read as a '1' in the CAN_SR.

Most of the bit of this register have different function depending on WR bit.

8-2. IMPORTANT NOTICE

The message object number 32 shall not be used in transmission.

- **NUMBER[5:0] : Message number**

Message number to read or write. Only values between 0x01 and 0x20 (0x01 and 0x20 included) are valid. When a message number that is not valid (0x00 and 0x21 to 0x3F) is written to this register, the message number is transformed into a valid message value and that Message is transferred.

- **WR : Write or read direction**

- 0: Read : transfer from the message object NUMBER into the selected interface X registers.
- 1: Write : transfer direction from the selected interface X registers to the message object NUMBER.

- **ADAR : Access Data A register**

- 0: Data bytes 0-3 unchanged.
- 1: Transfer data bytes 0-3 between message object and CAN_DARx register.

- **ADBR : Access Data B register**

0: Data bytes 4-7 unchanged.

1: Transfer data bytes 4-7 between message object and CAN_DBRx register.

- **AMSKR : Access mask register**

0: Mask bits unchanged.

1: Transfer mask bits between message object and CAN_MSKRx register.

- **AIR : Access identifier register**

0: Identifier bits unchanged.

1: Transfer identifier, direction, extended and message valid bits between message object and CAN_IRx register.

- **AMCR : Access message control register**

0: Control bits unchanged.

1: Transfer control bits between message object and CAN_MCRx register.

- **TRND : Set TXRQST bit or clear NEWDAT**

0: TXRQST or NEWDAT bit unchanged.

1: When writing (WR = 1), it set TXRQST bit in the message object. When reading (WR = 0), it clears NEWDAT bit in the message object.

In write mode (WR = 1), if a transmission is requested by programming bit TXRQST in this register, bit TXRQST in the CAN_MCRx register will be ignored.

In read mode (WR = 0), a read access to a message object can be combined with the reset of the control bits ITPND and NEWDAT.

The value of these bits transferred to the CAN_MCRx register always reflect the status before resetting these bits.

- **CLRIT : Clear interrupt pending bit**

0: ITPND bit remains unchanged.

1: Clear ITPND bit in the message object.

This bit is ignored in write mode.

CAN Interface X Data A Register

**CAN_DAR1 (0x104)
CAN_DAR2 (0x124)**

Access: Read/Write

31	30	29	28	27	26	25	24
DATA3[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
DATA2[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
DATA1[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
DATA0[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

Access to this register is prohibited while BUSYx bit is set.

- **DATAx[7:0] : Data x of interface X**

Data number x of interface X. In a CAN data frame, DATA0 is the first, DATA7 is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first.

CAN Interface X Data B Register **CAN_DBR1 (0x108)** **Access: Read/Write**
CAN_DBR2 (0x128)

31	30	29	28	27	26	25	24
DATA3[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
DATA2[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
DATA1[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
DATA0[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

Access to this register is prohibited while BUSYx bit is set.

- **DATAx[7:0] : Data x of interface X**

Data number x of interface X. In a CAN data frame, DATA0 is the first, DATA7 is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first.

CAN Interface X Mask Register

CAN_MSKR1 (0x010C)
CAN_MSKR2 (0x012C)

Access: Read/Write

31	30	29	28	27	26	25	24	
MXTD	MMDIR	–	BASEMASK[10:6]					
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	
23	22	21	20	19	18	17	16	
BASEMASK[5:0]						EXTMASK[17:16]		
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	
15	14	13	12	11	10	9	8	
EXTMASK[15:8]								
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	
7	6	5	4	3	2	1	0	
EXTMASK[7:0]								
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

Access to this register is prohibited while BUSYx bit is set.

- **EXTMASK[17:0] : Extended identifier mask**

18-bit mask for extended identifier (only extended identifier). If CAN controller received a 29-bit identifier(extended frame), all mask bits (BASEMASK [28:18] and EXTMASK[17:0]) are used. When a bit within EXTMASK bits is set to one, the corresponding identifier bit is used for acceptance filtering. When a bit within EXTMASK bits is set to zero, the corresponding bit in the identifier of the message object cannot inhibit the match in the acceptance filtering.

- **BASEMASK[10:0] : Base identifier mask**

11-bit mask for base identifier (standard and extended identifier). If CAN controller received a 11-bit identifier (standard frame), the lower bits (EXTMASK[17:0]) are not used. When a bit within BASEMASK bits is set to one, the corresponding identifier bit is used for acceptance filtering. When a bit within BASEMASK bits is set to zero, the corresponding identifier bit cannot inhibit the match in the acceptance filtering.

- **MMDIR : Message direction mask**

0: The MDIR bit of CAN_IRx register has no effect on the acceptance filtering.

1: The MDIR bit of CAN_IRx register is used for acceptance filtering.

- **MXTD : XTD bit mask**

0: The XTD bit of CAN_IRx register has no effect on the acceptance filtering.

1: The XTD bit of CAN_IRx register is used for acceptance filtering.

CAN Interface X Identifier Register **CAN_IR1 (0x110)** **Access: Read/Write**
CAN_IR2 (0x130)

31	30	29	28	27	26	25	24
MSGVAL	XTD	MDIR	BASEID[10:6]				
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
BASEID[5:0]						EXTID[17:16]	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
EXTID[15:8]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
EXTID[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

Access to this register is prohibited while BUSYx bit is set.

- **EXTID[17:0] : Extended identifier of interface X**

18-bit value for extended identifier. If CAN sends a 29-bit identifier, all bits (BASEID and EXTID) are used. If CAN sends a 29-bit identifier, base identifier (BASEID) are sent first, extended identifier (EXTID) later.

- **BASEID[10:0] : Base identifier of interface X**

11-bit value for base identifier (standard and extended identifier). If CAN sends only a 11-bit identifier, the lower bits (ID[17:0]) are not used. If CAN sends a 29-bit identifier, base identifier (BASEID) are sent first, extended identifier (EXTID) later.

- **MDIR : Message direction**

0: Receive. If TXRQST is set to '1', a remote frame with the identifier of this message object is transmitted. If a data frame with matching identifier is received, that message is stored in this message object.

1: Transmit. If TXRQST is set to '1', the respective message object is transmitted as a data frame. On reception of a remote frame with matching identifier, if RMTEN = '1' the TXRQST bit of this message object is set, if RMTEN = '0' and UMASK = '0' the remote frame is ignored, if RMTEN = '0' and UMASK = '1' the arbitration and control field (ID + IDE + RTR + DLC) from the received message are stored into the Message Object in the Message RAM and the NEWDAT bit of this Message Object is set.

- **XTD : Extended identifier**

- 0: The 11-bit (“standard”) identifier will be used for this message object.
- 1: The 29-bit (“extended”) identifier will be used for this message object.

- **MSGVAL : Message Valid**

- 0: The message object is ignored by the message handler.
- 1: The message object is configured and should be considered by the message handler.

The CPU must reset the MSGVAL bit of all unused message objects during initialization before it enables CAN (CANEN in the CAN_CR register). This bit must also be reset before the identifier (BASEID and EXTID), the control bits XTD and MDIR or the data length code DLC are modified, or if the messages object is no longer required.

CAN Interface X Message Control Register **CAN_MCR1 (0x114)** **Access: Read/Write**
CAN_MCR2 (0x134)

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
NEWDAT	MSGLST	ITPND	UMASK	TXIE	RXIE	RMTEN	TXRQST
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
OVERWRITE	–	–	–	DLC[3:0]			
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

Access to this register is prohibited while BUSYx bit is set. In basic mode, for interface 0 (transmit interface), only DLC bits retain their functions. For interface 1 (receive interface), only NEWDAT, MSGLST and DLC bits retains their functions, and when a message is received other bits are read as '0'.

- **DLC[3:0] : Data length code**

This is the number of bytes in the data field of a message (from 0 to 8). This value is updated whenever a frame is received (data or remote).

- **OVERWRITE : Overwrite mode**

0: Message object is configured in normal mode. In normal mode, new frames do not overwrite the previous frame.

1: Message object is configured in overwrite mode. In overwrite mode, new frames overwrite the previous frame.

This bit is used to concatenate two or more messages objects (up to 32) to build a FIFO buffer. For single message objects (not belonging to a FIFO buffer) this bit must always be set to one.

- **TXRQST : Transmit request**

0: This message object is not waiting for transmission.

1: The transmission of this message object is requested and is not yet done.

- **RMTEN : Remote enable**

0: At the reception of a remote frame, TXRQST is left unchanged.

1: At the reception of a remote frame, TXRQST is set.

- **RXIE : Receive interrupt enable**

0: ITPND will be left unchanged after a successful reception of a frame.

1: ITPND will be set after a successful reception of a frame.

- **TXIE : Transmit interrupt enable**

0: ITPND will be left unchanged after a successful transmission of a frame.

1: ITPND will be set after a successful transmission of a frame.

- **UMASK : Use acceptance mask**

0: Mask ignored.

1: Use mask (BASEMASK, EXTMASK, MXTD and MMDIR) for acceptance filtering.

- **ITPND : Interrupt pending**

0: This message is not the source of an interrupt.

1: This message object is the source of an interrupt.

- **MSGLST : Message lost**

0: No message lost since last time this bit was reset by the CPU.

1: The message handler stored a new message into this object when NEWDAT was still set, the CPU lost a message.

This is only valid for message objects with direction = receive.

- **NEWDAT : New data**

0: No new data has been written into the data portion of this message object by the message handler since last time this flag was reset by the CPU.

1: The message handler or the CPU has written new data into the data portion of this message object.

Interface X Stamp Register CAN_STPR0 (0x118) Access: Read only
CAN_STPR1 (0x138)

31	30	29	28	27	26	25	24
STAMP[31:24]							
R	R	R	R	R	R	R	R
23	22	21	20	19	18	17	16
STAMP[23:16]							
R	R	R	R	R	R	R	R
15	14	13	12	11	10	9	8
STAMP[15:8]							
R	R	R	R	R	R	R	R
7	6	5	4	3	2	1	0
STAMP[7:0]							
R	R	R	R	R	R	R	R

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **STAMP[31:0] : Stamp value**

These 32-bits stamp the date at which the message linked with the channel X has been received. The value is copied from the STT second counter register.

CAN Transmission Request Register

CAN_TRR (0x140)

Access: Read only

31	30	29	28	27	26	25	24
CH32	CH31	CH30	CH29	CH28	CH27	CH26	CH25
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
CH24	CH23	CH22	CH21	CH20	CH19	CH18	CH17
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
CH16	CH15	CH14	CH13	CH12	CH11	CH10	CH9
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **CHx : Transmission request on channel x**

0: This message object is not waiting for a transmission.

1: The transmission object of this message object is requested and is not yet done.

CAN New Data Register

CAN_NDR (0x144)

Access: Read only

31	30	29	28	27	26	25	24
CH32	CH31	CH30	CH29	CH28	CH27	CH26	CH25
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
CH24	CH23	CH22	CH21	CH20	CH19	CH18	CH17
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
CH16	CH15	CH14	CH13	CH12	CH11	CH10	CH9
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **CHx : New data on channel x**

- 0: No new data has been written into the data portion of this message object by the message handler since last time this flag was cleared by the CPU.
- 1: The message handler or the CPU has written new data into the data portion of this message object.

CAN Message Valid Register

CAN_MVR (0x148)

Access: Read only

31	30	29	28	27	26	25	24
CH32	CH31	CH30	CH29	CH28	CH27	CH26	CH25
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
CH24	CH23	CH22	CH21	CH20	CH19	CH18	CH17
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
CH16	CH15	CH14	CH13	CH12	CH11	CH10	CH9
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **CHx : Message valid on channel x**

0: This message object is ignored by the message handler.

1: This message object is configured and should be considered by the message handler.

CAN Test Register		CAN_TSTR (0x14C)				Access: Read/Write	
31	30	29	28	27	26	25	24
TSTKEY[15:8]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
TSTKEY[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
–	RX	TXOPD	TX[1:0]		LBACK	SILENT	BASIC
R/W-0	R-U	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **BASIC : Basic mode**

0: Basic mode disabled.

1: Interface 0 registers used as TX buffer, Interface 1 registers used as RX buffer. For further details, see “Basic Mode”.

- **SILENT : Silent mode**

0: Silent mode is disabled.

1: The module is in silent mode.

- **LBACK : Loop Back Mode**

0: Loop back mode is disabled.

1: Loop back mode is enabled.

• **TX[1:0] : Control of CAN_TX pin**

TX must be left in its default function (controlled by CAN core) when CAN message transfer or any of the test modes Loop Back Mode, Silent Mode, or Basic Mode are selected. The three test functions (monitoring, drive '0' and drive '1') interfere with all CAN protocol functions.

Table 8-13. Control of CAN_TX Pin

TX[1:0]	Control
00	CAN_TX is controlled by CAN core
01	Sample point is monitored at CAN_TX pin (CAN controller shall be enabled)
10	CAN_TX pin drives a dominant ('0') value
11	CAN_TX pin drives a recessive ('1') value

• **TXOPD : TX open drain**

0: Output CAN_TX pin is not configured as an open drain.

1: Output CAN_TX pin is configured as an open drain. It allows to connect directly CAN controller without external transceiver or to drive properly a 5V transceiver with a 3V3 CAN controller with an additional external pull up.

NOTE: when switching this bit, user should disable CAN in order to avoid to generate glitch on CAN_TX pin.

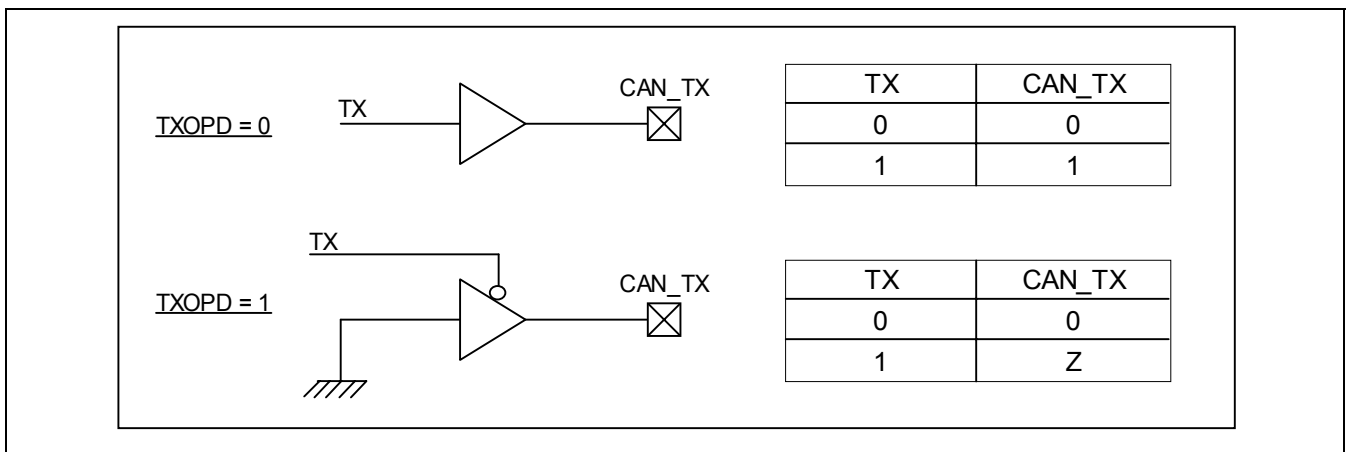


Figure 8-27. Open Drain Mode

• **RX : Monitor the value of CAN_RX pin**

0: The CAN bus is dominant (CAN_RX = '0').

1: The CAN bus is recessive (CAN_RX = '1').

• **TSTKEY[15:0] : Test access key**

Used only when writing CAN_TSTR. TSTKEY is read as 0.

0x0C75 = Write access in CAN_TSTR is allowed.

Other value = Write access in CAN_TSTR is prohibited.

9

DATA FLASH CONTROLLER (DFC)

1. OVERVIEW

1.1 FUNCTIONAL DESCRIPTION

The Internal Data Flash Memory Controller (DFC) is used to access one 32 Kbytes synchronous flash (1024 words x 32 bits).

The DFC supports different working modes:

- Read access in 8, 16 or 32 bits
- Write access (program) in 8, 16 or 32 bits
- Page erase: erase only one 256 bytes page in the flash memory
- Sector erase: erase only one 8 Kbytes sector in the flash memory
- Chip erase: the internal data flash memory is completely erased except the sector 0.
- Standby: used to decrease the power consumption when the flash is not used.

The user can change the module base address. The write access protection is supported through the DFC_MR register. Write access and flash erase are supported.

Interrupts are generated to indicate end of memory erase cycle, end of write cycle and denied access.

1.2 BLOCK DIAGRAM

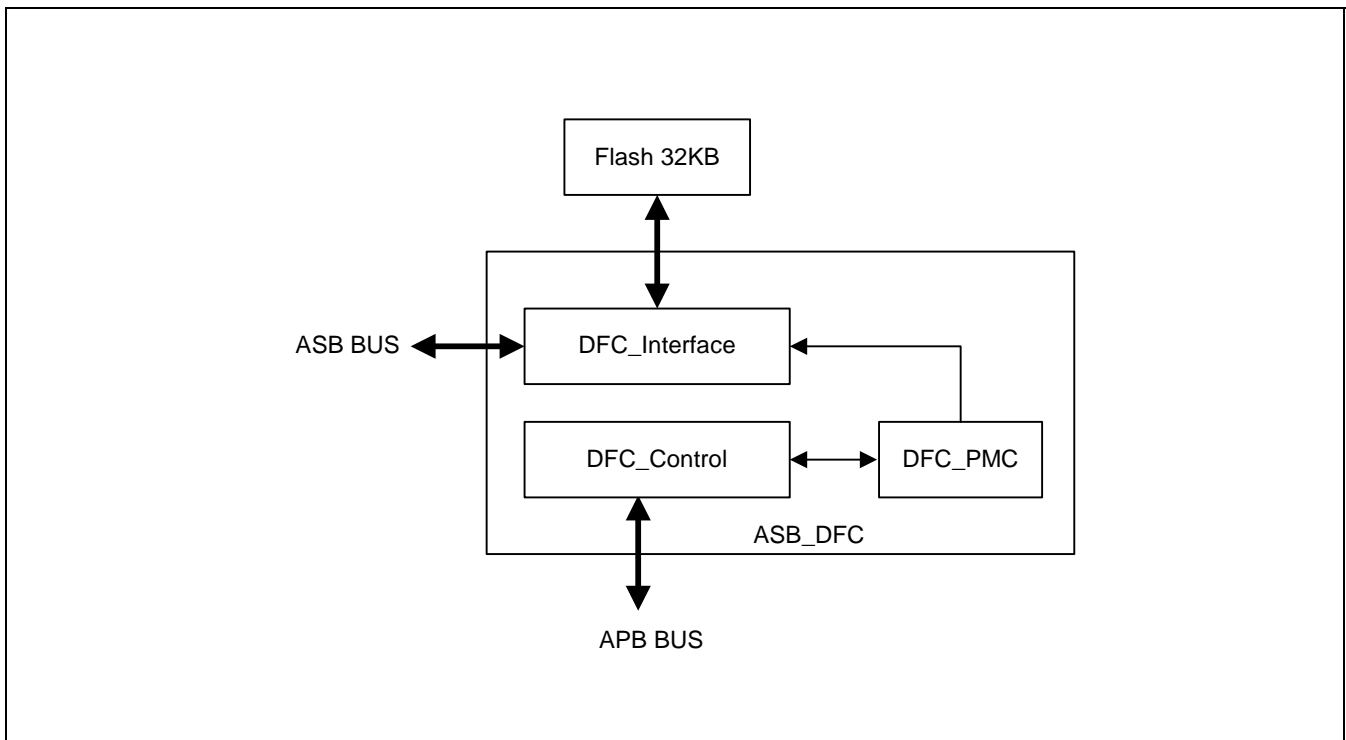


Figure 9-1. Data Flash Memory Controller Block Diagram

2. FUNCTIONAL OPERATION

2.1 INTERNAL DATA FLASH CONTROLLER

2.1.1 General Description

The Internal Data Flash Controller allows to read or write data in one 32 Kbytes Flash Memory (1024 words x 32 bits). These accesses can be done in 8, 16 or 32 bits.

To reduce the power consumption, the DFC can be stopped.

The data flash memory can be erased. There are 3 different erase cycles: page erase, sector erase and chip erase.

2.1.2 Data Flash Segmentation

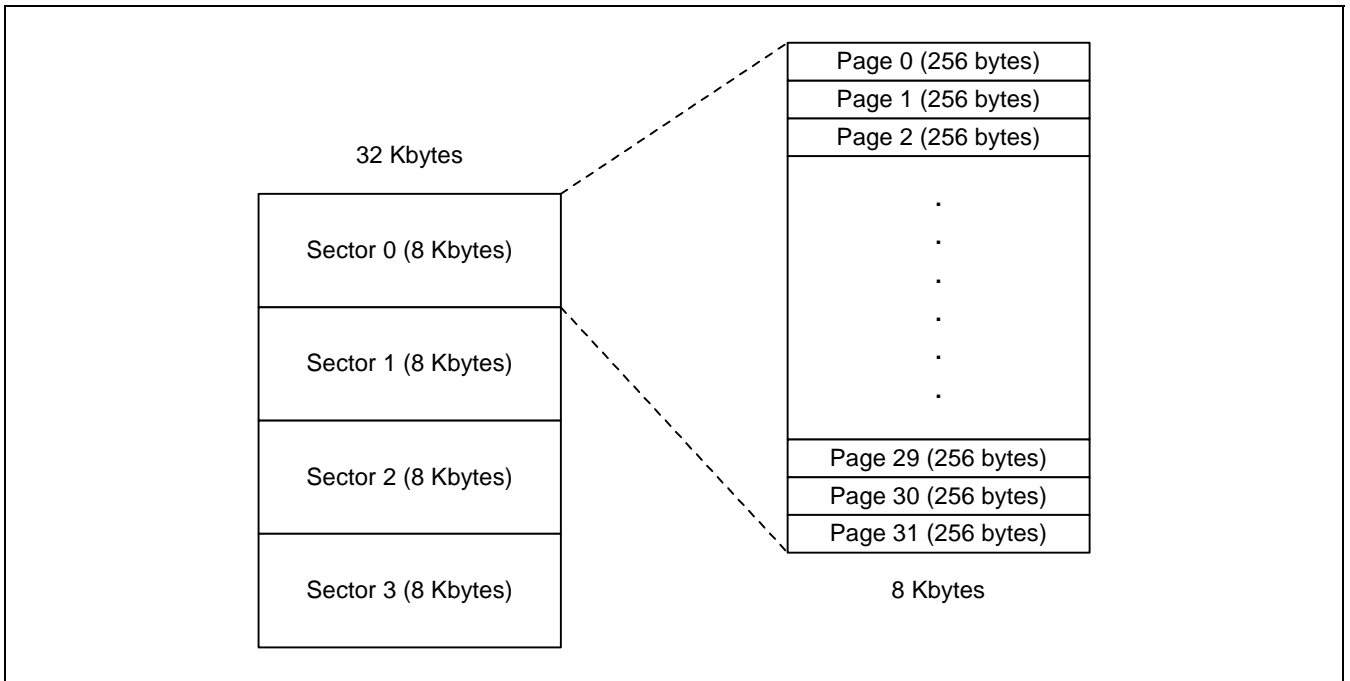


Figure 9-2. Data Flash Segmentation

2.1.3 Address Decoding

The Internal Data Flash Controller is selected when the MSB of the ASB address (baddr[31:24]) match the Base Address (BA[7:0]) programmed in the DFC_MR (DFC Mode Register).

The Base address can be changed to remap the Internal Data Flash Controller.

NOTE

At reset the Base Address is equal to 0x80000000

2.2 READ ACCESS

Read access can be done in 8, 16 or 32 bits.

During a read access the flash clock frequency is equal at system clock frequency divided by 2.

One read access required 2 clocks cycles: each read access requires one wait state.

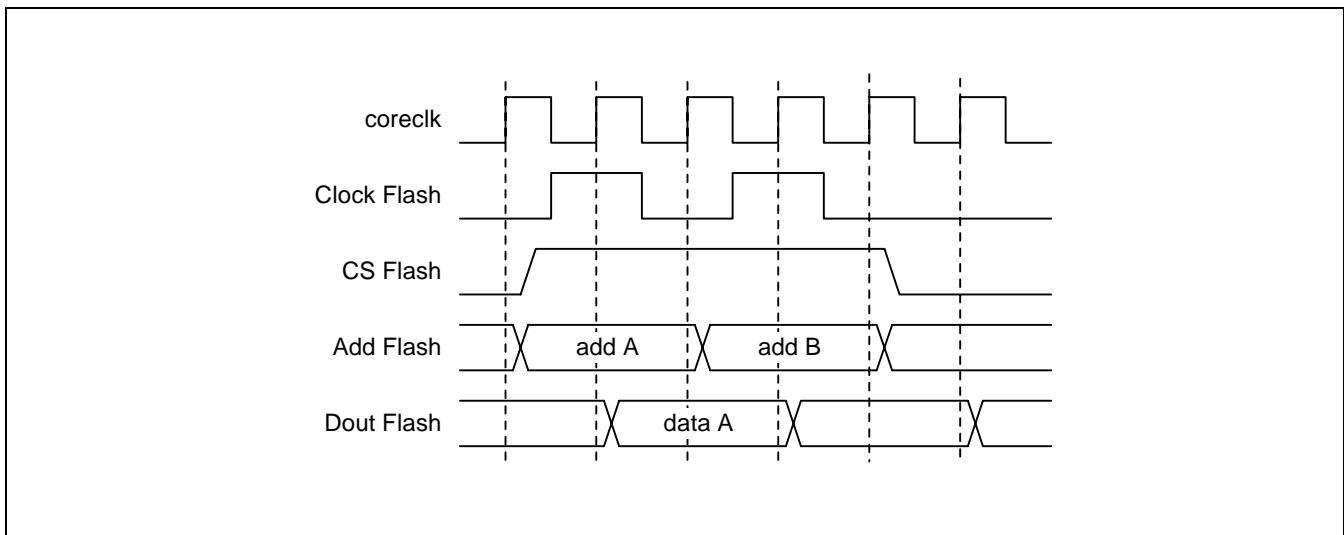


Figure 9-3. Read Access Timing Diagram

2.3 ERASE CYCLE

Data Flash erase must be done in the program flash or SRAM only.

There are 3 types of erase cycle:

- chip erase: the chip erase cycle erases the whole data flash memory except the sector 0. It is done when the ERASE[1:0] bits are set to 0x3 in the DFC_CR (DFC Control Register). The chip erase takes typical 32 ms. When the chip erase is finished, the DFC generates an end of erase interrupt.
- sector erase: the flash memory is divided in 4 sectors of 8 Kbytes. The sector erase cycle erase one sector of data flash memory. It is done by programming the sector number(SECTOR[1:0]) and setting the ERASE[1:0] bits to 0x2 in the DFC_CR (DFC Control Register). The sector erase takes typical 32 ms. When the sector erase cycle is finished, the DFC generates an end of erase interrupt.
- page erase: each flash sector is divided in 32 pages of 256 bytes (there are 128 pages in the data flash memory). The page erase cycle erase one page of flash memory. It is done by programming the page number (PAGE[4:0]) and the sector number (SECTOR[1:0]) and setting the ERASE[1:0] bits to 0x1 in the DFC_CR (DFC Control Register). The page erase takes typical 4 ms. When the page erase cycle is finished, the DFC generates an end of erase interrupt.

During one (page or sector or chip) erase cycle, if a read or write access occurs in the DFC, this access is not taken into account, an abort is generated on the internal bus and a denied access interrupt is generated.

During one (page or sector or chip) erase cycle, if a new erase cycle is programmed, this erase cycle is not taken into account and a denied access interrupt is generated.

If erase (page or sector or chip) cycles are not allowed (bit WPR high in the DFC_MR Register), and an erase cycle is programmed, the erase cycle is ignored and an denied access interrupt is generated.

During one (page or sector or chip) erase cycle, the busy bit in the DFC_SR (DFC Status Register) is set to high and automatically cleared at the end of the erase cycle.

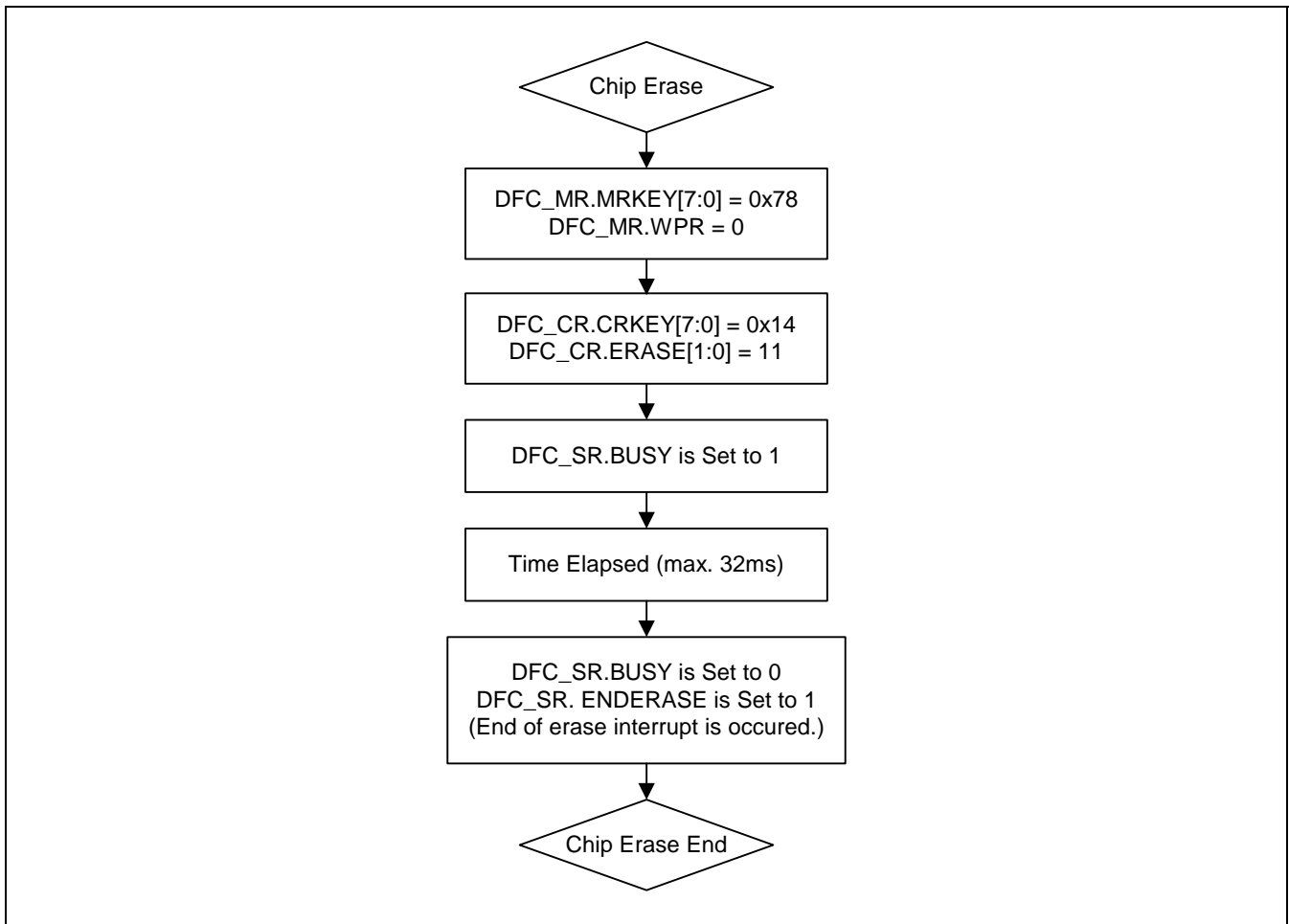


Figure 9-4. Chip Erase Flow Chart

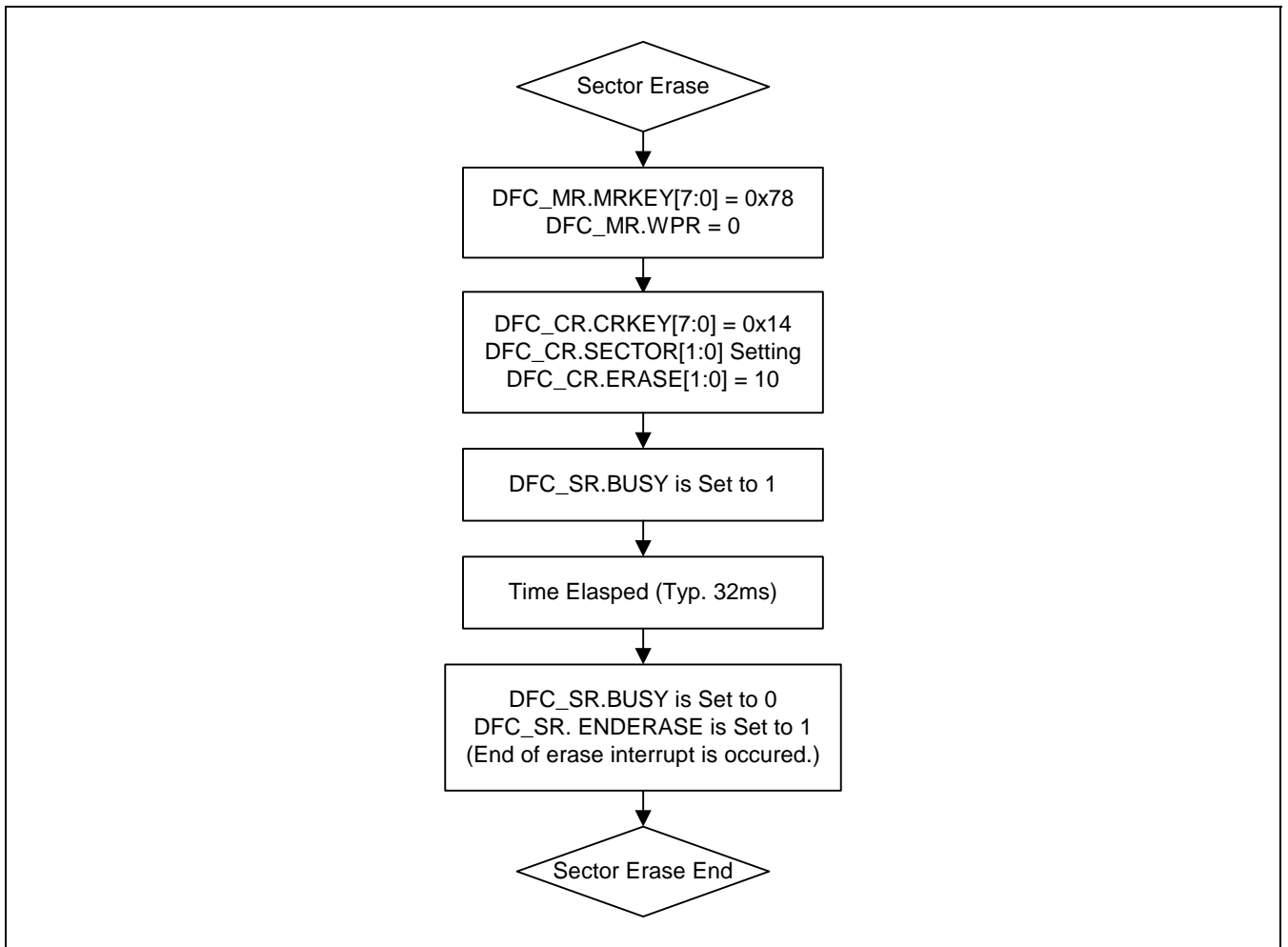


Figure 9-5. Sector Erase Flow Chart

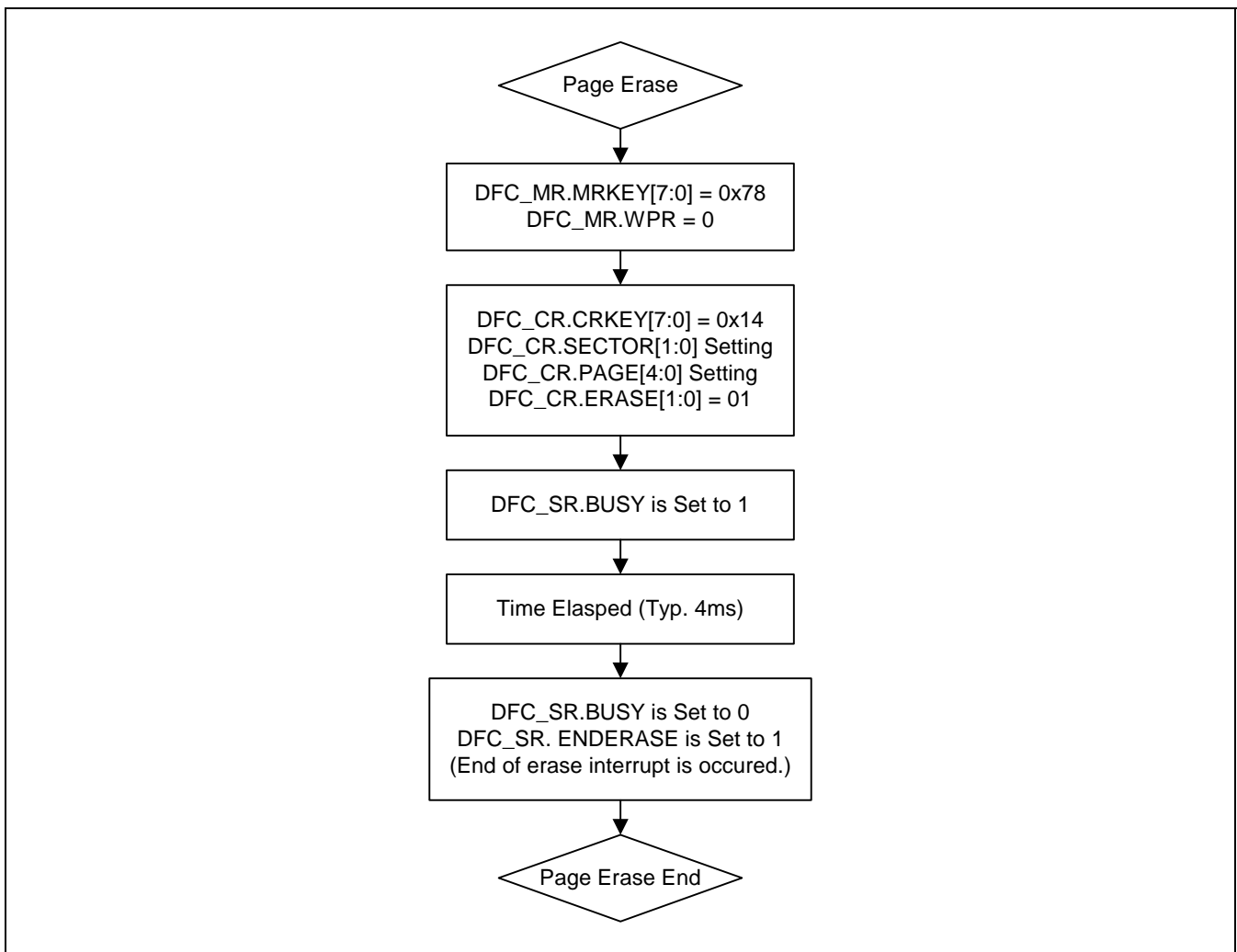


Figure 9-6. Page Erase Flow Chart

2.4 WRITE CYCLE

Data Flash write must be done in the program flash or SRAM only. A write cycle takes typical 41us.

When the write access is done, an end of write interrupt is generated. Write accesses can be done in 32 bits, 16 bits or 8 bits. A write cycle can only be performed on a previously erased flash. If a write is requested on an address whose data has not been erased before, the write is not taken into account by the flash (data does not match 0xFFFFFFFF).

During a write access, if a new read or write access occurs, this new access is not taken into account, a denied interrupt is generated.

During a write cycle, if an erase cycle is programmed, this erase cycle is not taken into account and a denied access interrupt is generated.

When write cycles are not allowed (bit WPR high in the DFC_MR Register), if a write access occurs, an abort is generated on the internal bus and a denied access interrupt is generated.

During a write access, the busy bit in the DFC_SR is set to high and automatically cleared at the end of the write cycle.

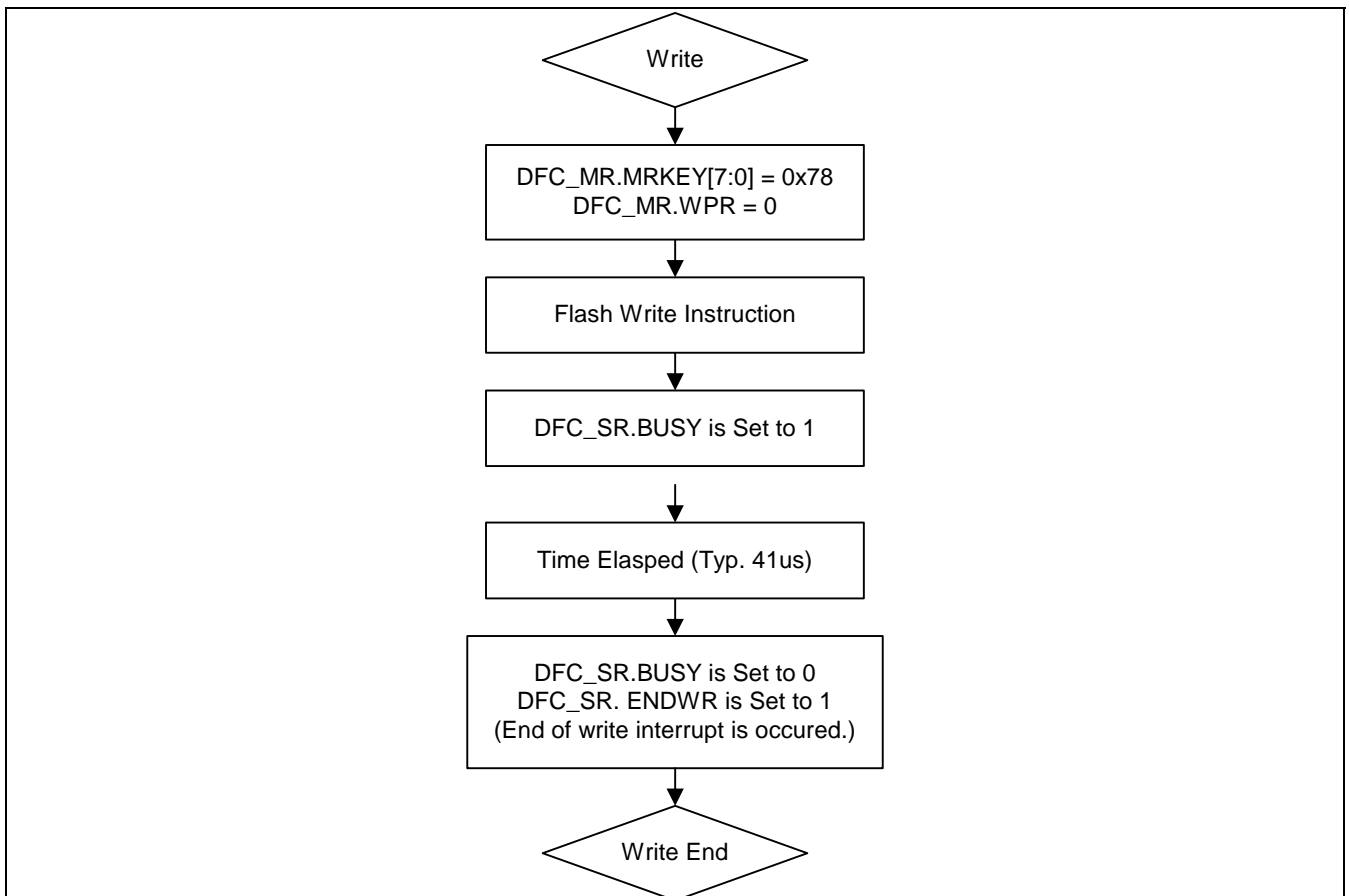


Figure 9-7. Write Flow Chart

2.5 STANDBY MODE

In order to decrease the power consumption, the data flash standby mode is implemented. It is done when the bit STANDBY is high in the DFC_MR register. When the data flash memories are in standby mode, it is not possible to access in the data flash memories until the STANDBY bit is not cleared.

A denied access interrupt is generated when the flash memory is in standby mode and a new erase or a new read or write access is attempted. In addition for the last two mode, an abort is generated.

During an erase or write or read cycle, if the user set the standby mode, the IFC waits the end of erase or write or read cycle to set the data flash memory in standby.

2.6 INTERRUPTS GENERATION

There are 3 sources of interrupts:

- End of write cycle interrupt occurs once the write cycle has ended.
- End of erase cycle interrupt occurs once the chip or sector erase cycle has ended.
- Denied access interrupt occurs when:
 - The flash is busy (during an erase or write cycle) or in standby and a new read, write or erase cycle is attempted
 - A write access or erase cycle are attempted while write access are not allowed (see write protection bit in the DFC_MR Register)

2.7 DATA ABORT GENERATION

Access is aborted when:

- The flash is busy (during an erase or write cycle) or in standby and a new access (read or write) is attempted.
- A write access is attempted while write access are not allowed (see write protection bit in the DFC_MR Register).

3. REGISTERS DESCRIPTION

Base Address – 0xFFE00000

Table 9-1. DFC Special Function Registers

Offset Address	Name	Description	R/W	Reset State
0x000	–	Reserved	–	–
–	–	–	–	–
0x0050	DFC_ECR	Enable clock register	W	–
0x0054	DFC_DCR	Disable clock register	W	–
0x0058	DFC_PMSR	Power management status register	R	0XXXXXXXX0
0x005C	–	Reserved	–	–
0x0060	DFC_CR	Controller register	W	0x00000000
0x0064	DFC_MR	Mode register	R/W	0x80000090
0x0068	–	Reserved	–	–
0x006C	DFC_CSR	Clear status register	W	0x00000000
0x0070	DFC_SR	Status register	R	0x00000000
0x0074	DFC_IER	Interrupt enable register	W	–
0x0078	DFC_IDR	Interrupt disable register	W	–
0x007C	IMR	Interrupt mask register	R	0x00000000

DFC Enable Clock Register				DFC_ECR (0x050)				Access: Write only	
31	30	29	28	27	26	25	24		
–	–	–	–	–	–	–	–		
W	W	W	W	W	W	W	W		W
23	22	21	20	19	18	17	16		
–	–	–	–	–	–	–	–		
W	W	W	W	W	W	W	W		W
15	14	13	12	11	10	9	8		
–	–	–	–	–	–	–	–		
W	W	W	W	W	W	W	W		W
7	6	5	4	3	2	1	0		
–	–	–	–	–	–	DFC	–		
W	W	W	W	W	W	W	W		W

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **DFC: DFC Clock enable**

0: No effect

1: Enable DFC clock

DFC Disable Clock Register

DFC_DCR (0x054)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	DFC	–
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **DFC: DFC Clock disable**

0: No effect

1: Disable DFC clock

DFC Power Management Status			DFC_PMSR (0x058)				Access: Read only	
31	30	29	28	27	26	25	24	
-		IPIDCODE[25:20]						
R-0	R-0	R-U	R-U	R-U	R-U	R-U	R-U	
23	22	21	20	19	18	17	16	
IPIDCODE[19:12]								
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U	
15	14	13	12	11	10	9	8	
IPIDCODE[11:4]								
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U	
7	6	5	4	3	2	1	0	
IPIDCODE[3:0]				-	-	DFC	-	
R-U	R-U	R-U	R-U	R-0	R-0	R-0	R-0	

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- DFC: DFC Power Management Status**

- 0: DFC clock is disabled
- 1: DFC clock is enabled

- IPIDCODE[25:0]**

This field contains the version number of the module, coded on 26 bits.

DFC Control Register **DFC_CR (0x060)** Access: Write only

31	30	29	28	27	26	25	24
SECTOR[1:0]		PAGE[4:0]					-
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
CRKEY[7:0]							
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
-	-	-	-	-	ERASE		-
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- SECTOR[1:0]: Sector Number

SECTOR[1:0]	Sector Number	Sector Size
0x0	0	8 Kbytes
0x1	1	8 Kbytes
0x2	2	8 Kbytes
0x3	3	8 Kbytes

- PAGE[4:0]: Page Number

PAGE[1:0]	Page Number	Page Size
0x00	0	256 bytes
0x01	1	256 bytes
0x02	2	256 bytes
---	---	---
0x1E	30	256 bytes
0x1F	31	256 bytes

- **CRKEY[7:0]: Key for the write access into the DFC_CR Register**

Any write in this register will only be effective if the CRKEY[7:0] is equal to 0x14

- **ERASE[1:0]: Erase cycle**

ERASE[1:0]	Effect	Needed Action
0x0	No Effect	Nothing
0x1	Start a Page Erase	Sector & Page must be programmed
0x2	Start a Sector Erase	Sector must be programmed
0x3	Start a Chip Erase	Nothing

DFC Mode Register				DFC_MR (0x064)				Access: Read/Write
31	30	29	28	27	26	25	24	
BA[7:0]								
R/W-1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
23	22	21	20	19	18	17	16	
–	–	–	–	–	–	–	–	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
15	14	13	12	11	10	9	8	
MRKEY[7:0]								
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	
7	6	5	4	3	2	1	0	
WPR	–	–	STANDEN	–	–	–	–	
R/W-1	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0	

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **BA[7:0]: Data Flash Controller Base Address**

The BA[7:0] contains the MSB of the DFC base address

- **MRKEY[7:0]: Key for the write access into the DFC_MR Register**

Any write in this register will be only effective if the MRKEY[7:0] is equal to 0x78.

- **WPR: Write and Erase Protection**

0: Write and erase are allowed in flash.

1: Write and erase are not allowed (only read is allowed) in flash

NOTES:

1. If the WPR = 1 and a write access occurs, the DFC sends an abort response at the ASB master and a denied access interrupt is generated.
2. If the WPR = 1 and an erase is executed, a denied access interrupt is generated.
3. At reset WPR = 1

- **STANDEN: Standby mode enable**

0: Standby mode is disabled

1: Standby mode is enabled

9-1. IMPORTANT NOTICE

1. If the STANDEN = 1 and a new access occurs, a denied access interrupt is generated and the access is aborted.
2. At reset STANDEN bit value is "1".

DFC Clear Status		DFC_CSR (0x06C)						Access: Write only	
31	30	29	28	27	26	25	24		
–	–	–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16		
–	–	–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8		
–	–	–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0		
–	–	–	–	–	DACCESS	ENDERASE	ENDWR		
W	W	W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **ENDWR: Clear end of write cycle**

0: No effect

1: Clears the DFC end of write status

- **ENDERASE: Clear end of erase cycle**

0: No effect

1: Clears the DFC end of erase cycle status

- **DACCESS: Clear denied access**

0: No effect

1: Clears the DFC denied access status

DFC Status Register				DFC_SR (0x070)				Access: Read only
31	30	29	28	27	26	25	24	
–	–	–	–	–	–	–	–	
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	
23	22	21	20	19	18	17	16	
–	–	–	–	–	–	–	–	
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	
15	14	13	12	11	10	9	8	
–	–	–	–	–	–	–	BUSY	
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	
7	6	5	4	3	2	1	0	
–	–	–	–	–	DACCESS	ENDERASE	ENDWR	
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **ENDWR: End of write cycle**

0: No write cycle has been yet completed since last corresponding ENDWR bit was cleared in DFC_SR.

1: A write cycle has been completed

- **ENDERASE: End of chip or sector erase**

0: No erase cycle has been yet completed since last corresponding ENDERASE bit was cleared in DFC_SR

1: An erase cycle has been completed

- **DACCESS: Denied access**

0: No denied access has occurred since last corresponding DACCESS bit was cleared in DFC_SR

1: An access has occurred during the flash controller was busy; or write or erase cycle are attempted when the write protection is set.

- **BUSY: Flash busy (status bit)**

0: Flash controller is not busy

1: Flash controller is busy, no more read, write or erase accesses are possible until busy bit is high

DFC Interrupt Enable		DFC_IER (0x074)				Access: Write only		
31	30	29	28	27	26	25	24	
–	–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16	
–	–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8	
–	–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0	
–	–	–	–	–	DACCESS	ENDERASE	ENDWR	
W	W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **ENDWR: end of write interrupt enable**

0: No effect

1: Enable the ENDWR interrupt

- **ENDERASE: end of erase interrupt enable**

0: No effect

1: Enable the ENDERASE interrupt

- **DACCESS: denied access interrupt enable**

0: No effect

1: Enable the DACCESS interrupt

DFC Interrupt Disable		DFC_IDR (0x078)				Access: Write only		
31	30	29	28	27	26	25	24	
–	–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16	
–	–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8	
–	–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0	
–	–	–	–	–	DACCESS	ENDERASE	ENDWR	
W	W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **ENDWR: end of write interrupt disable**

0: No effect

1: Disable the ENDWR interrupt

- **ENDERASE: end of erase interrupt disable**

0: No effect

1: Disable the ENDERASE interrupt

- **DACCESS: denied access interrupt disable**

0: No effect

1: Disable the DACCESS interrupt

DFC Interrupt Mask				DFC_IMR (0x07C)				Access: Read only			
31	30	29	28	27	26	25	24				
–	–	–	–	–	–	–	–				
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0				
23	22	21	20	19	18	17	16				
–	–	–	–	–	–	–	–				
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0				
15	14	13	12	11	10	9	8				
–	–	–	–	–	–	–	–				
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0				
7	6	5	4	3	2	1	0				
–	–	–	–	–	DACCESS	ENDERASE	ENDWR				
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0				

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **ENDWR: end of write interrupt mask**

0: The end of write interrupt is disabled

1: The end of write interrupt is enabled

- **ENDERASE: end of erase interrupt mask**

0: The end of erase interrupt is disabled

1: The end of erase interrupt is enabled

- **DACCESS: denied access interrupt mask**

0: The denied access interrupt is disabled

1: The denied access interrupt is enabled

10 GENERAL PURPOSE I/O (PIO)

1. OVERVIEW

1.1 FUNCTIONAL DESCRIPTION

All I/O lines in the micro-controller are unified in the PIO controller. All I/O lines are controlled by this module. The PIO controller also provides an internal interrupt signal to the Generic Interrupt Controller.

1.2 BLOCK DIAGRAM

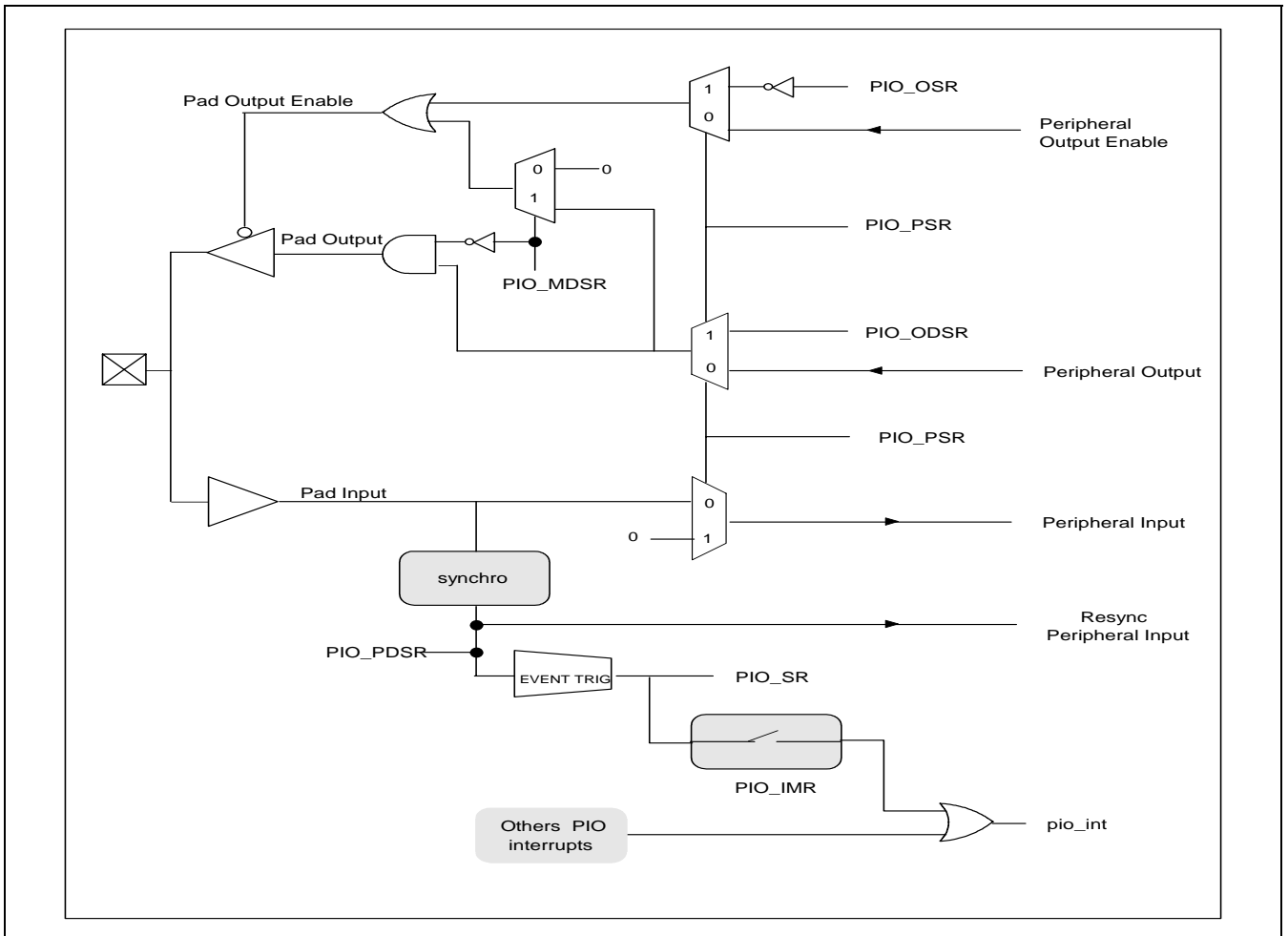


Figure 10-1. PIO Block Diagram

2. FUNCTIONAL OPERATION

2.1 PIO CONTROLLER BLOCK

2.1.1 General Description

To match different applications, peripherals have their own dedicated pins multiplexed with general purpose I/O pins (PIO). Each PIO block in the peripheral is controlled through the peripheral interface.

2.1.2 Multiplexed I/O Lines

All I/O lines are multiplexed with an I/O signal of the peripheral. After reset, the pin is controlled by the PIO controller(See P11-8, PIO_PSR).

Each parallel I/O line is bi-directional, whether the peripheral defines the signal as input or output.

The figure 11-1 shows the multiplexing of the peripheral signals(ex. Peripheral Output Enable, Peripheral Output, etc.) with the PIO controller signal (ex. PIO_OSR, PIO_ODSR, etc.).

Each pin of the peripheral can be independently controlled using the PIO_PER (PIO Enable) and PIO_PDR (PIO Disable) registers.

The PIO_PSR (PIO Status) indicates whether the pin is controlled by the peripheral or by the PIO controller block.

2.1.3 Output Selection

The user can select the direction of each individual I/O signal (input or output) using the PIO_OER (Output Enable) and PIO_ODR (Output Disable) registers. The output status of the I/O signal can be read in the PIO_OSR (Output Status) register. The direction defined has effect only if the pin is configured to be controlled by the PIO controller block.

2.1.4 I/O Levels

Each pin can be configured to be independently driven high or low. The level is defined in different ways, according to the following conditions.

- If a pin is controlled by the PIO controller block and is defined as an output (see Output Selection above), the level is programmed using the PIO_SODR (Set Output Data) and PIO_CODR (Clear Output Data) registers. In this case, the programmed value can be read in the PIO_ODSR (Output Data Status) register. The user can affect all bits (the bus) at the same time with the PIO_WODR register. The programmed value is also read in PIO_ODSR.
- If a pin is controlled by the PIO controller block and is not defined as an output, the real level is determined by the external circuit. If a pin is not controlled by the PIO controller block, the state of the pin is defined by the Peripheral controller.

In all cases, the real level on the pin can be read in the PIO_PDSR (Pin Data Status) register.

2.1.5 Interrupts

Each PIO controller block also provides an internal interrupt signal shared with the peripheral interrupt.

Each PIO can be programmed to generate an interrupt when a level change occurs (rising/falling edge). This is controlled by the PIO_IER (Interrupt Enable) and PIO_IDR (Interrupt Disable) registers which enable/disable the I/O interrupt (and the peripheral interrupts) by setting/clearing the corresponding bit in the PIO_IMR.

When a change in level occurs, the corresponding bit in the PIO_SR (Interrupt Status) register is set whether the pin is used as a PIO or a peripheral signal and whether it is defined as input or output.

If the corresponding interrupt in PIO_IMR (Interrupt Mask) register is enabled, the PIO interrupt is asserted.

The PIO interrupt is cleared when a write access is performed on the PIO_CSR register (with the corresponding bit set at a logical 1)

The name of PIO interrupt is GPIO0, GPIO1, GPIO2 and GPIO3 each by each (See P13-2). Any level change of P0.x can make PC to jump to the GPIO0 interrupt vector. This is same situation for GPIO1, GPIO2 and GPIO3. These interrupt are different from IRQ0 to IRQ19 (external interrupt).

2.1.6 User Interface

Each individual PIO is associated with a bit position in the PIO controller user interface registers.

Each of these registers is 32 bits wide. If a parallel I/O line is not defined, writing to the corresponding bits has no effect.

Undefined bits read zero.

2.1.7 Open Drain / Push Pull Output

The PIO can either be configured as an open drain output (only drives a low level) or as a push pull output (drives high and low levels).

When the PIO is configured as open drain (multi-driver), an external pull-up is necessary to guarantee a logic level (logical one) when the pin is not being driven.

The PIO_MDER (Multi-Driver Enable) and PIO_MDDR (Multi-Driver Disable) registers control this option and respectively configure the I/O as open drain or push pull. The multi-driver option can be selected whether the I/O pin is controlled by the PIO controller or the peripheral controller. Bits at logical one in the PIO_MDSR (Multi-Driver Status) indicate pins configured as open drain.

3. REGISTERS DESCRIPTION

Base Addresses – PIO0: 0xFFE64000
 PIO1: 0xFFE68000
 PIO2: 0xFFE6C000

Table 10-1. PIO Special Function Registers

Offset Address	Name	Description	R/W	Reset State
0x000	PIO_PER	PIO pin enable register	W	–
0x004	PIO_PDR	PIO pin disable register	W	–
0x008	PIO_PSR	PIO pin status register	R	(note)
0x00C	–	Reserved	–	–
0x010	PIO_OER	Output enable register	W	–
0x014	PIO_ODR	Output disable register	W	–
0x018	PIO_OSR	Output status register	R	0x00000000
0x01C	–	Reserved	–	–
0x020	–	Reserved	–	–
0x024	–	Reserved	–	–
0x028	–	Reserved	–	–
0x02C	PIO_WODR	Write output data register	W	–
0x030	PIO_SODR	Set output data register	W	–
0x034	PIO_CODR	Clear output data register	W	–
0x038	PIO_ODSR	Output data status register	R	0x00000000
0x03C	PIO_PDSR	Pin data status register	R	0xFFFFFFFF
0x040	PIO_MDER	Multi-Driver enable register	W	–
0x044	PIO_MDDR	Multi-Driver disable register	W	–
0x048	PIO_MDSR	Multi-Driver status register	R	0x00000000
0x04C	–	Reserved	–	–
0x050	PIO_ECR	Enable clock register	W	–
0x054	PIO_DCR	Disable clock register	W	–
0x058	PIO_PMSR	Power management status register	R	0x00000000
0x05C	–	Reserved	–	–
0x060	PIO_CR	Control register	W	–
0x064	–	Reserved	–	–
0x068	–	Reserved	–	–

NOTE: PIO0 reset state = 0x007FFFFFFF, PIO1 reset state = 0xFFFFFFFF and PIO2 reset state = 0x00FFFFFFF.

Table 10-1. PIO Special Function Registers (Continued)

Offset Address	Name	Description	R/W	Reset State
0x006C	PIO_CSR	Clear status register	W	–
0x0070	PIO_SR	Status register	R	0x00000000
0x0074	PIO_IER	Interrupt enable register	W	–
0x0078	PIO_IDR	Interrupt disable register	W	–
0x007C	PIO_IMR	Interrupt mask register	R	0x00000000

PIO Pin Enable Register **PIO_PER (0x000)** **Access: Write only**

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **Px : Px pin**

0: No effect

1: The corresponding I/O is driven by PIO

PIO Pin Disable Register

PIO_PDR (0x004)

Access: Write only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **Px : Px pin**

0: No effect

1: The corresponding I/O is driven by peripheral

PIO Pin Status Register **PIO_PSR (0x008)** **Access: Read only**

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
R-1	R-1	R-1	R-1	R-1	R-1	R-1	R-1
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
R-1	R-1	R-1	R-1	R-1	R-1	R-1	R-1
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
R-1	R-1	R-1	R-1	R-1	R-1	R-1	R-1
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0
R-1	R-1	R-1	R-1	R-1	R-1	R-1	R-1

W: Write R: Read -0: 0 After reset
-1: 1 After reset
-U: Undefined after reset

- **Px : Px pin**

0: The corresponding I/O is driven by peripheral

1: The corresponding I/O is driven by PIO

PIO Output Enable Register

PIO_OER (0x0010)

Access: Write only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **Px : Px pin**

0: No effect

1: Enable the PIO output on the corresponding pin

PIO Output Disable Register **PIO_ODR (0x0014)** **Access: Write only**

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **Px : Px pin**

0: No effect

1: Disable the PIO output on the corresponding pin

PIO Output Status Register **PIO_OSR (0x0018)** **Access: Read only**

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **Px : Px pin**

- 0: The corresponding PIO is input on this line
- 1: The corresponding PIO is output on this line

PIO Write Output Data Register **PIO_WODR (0x002C)** **Access: Write only**

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **Px : Px pin**

The purpose of this register is same with PIO_SODR and PIO_CODR. But, all output data is affected at the same time. This function is different from that of PIO_SODR and PIO_CODR.

PIO Set Output Data Register **PIO_SODR (0x0030)** **Access: Write only**

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **Px : Px pin**

0: No effect

1: PIO output data on the corresponding pin is set

PIO Clear Output Data Register

PIO_CODR (0x0034)

Access: Write only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0
W	W	W	W	W	W	W	W

*W: Write R: Read -0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **Px : Px pin**

0: No effect

1: PIO output data on the corresponding pin is cleared

PIO Output Data Status Register

PIO_ODSR (0x0038)

Access: Read only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **Px : Px pin**

0: The output data for the corresponding PIO is programmed to 0

1: The output data for the corresponding PIO is programmed to 1

PIO Pin Data Status Register **PIO_PDSR (0x003C)** **Access: Read only**

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **Px : Px pin**

0: The real level of corresponding pin is at logic 0

1: The real level of corresponding pin is at logic 1

PIO Multi-Driver Enable Register

PIO_MDER (0x0040)

Access: Write only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **Px : Px pin**

0: No effect

1: Enable the multi-driver option on the corresponding pin

PIO Multi-Driver Disable Register **PIO_MDDR (0x0044)** **Access: Write only**

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **Px : Px pin**

0: No effect

1: Disable the multi-driver option on the corresponding pin

PIO Multi-Driver Status Register

PIO_MDSR (0x0048)

Access: Read only

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **Px : Px pin**

0: The corresponding PIO is not configured as an open drain

1: The corresponding PIO is configured as an open drain

PIO Enable Clock Register		PIO_ECR (0x0050)						Access: Write only
31	30	29	28	27	26	25	24	
–	–	–	–	–	–	–	–	
W	W	W	W	W	W	W	W	
23	22	21	20	19	18	17	16	
–	–	–	–	–	–	–	–	
W	W	W	W	W	W	W	W	
15	14	13	12	11	10	9	8	
–	–	–	–	–	–	–	–	
W	W	W	W	W	W	W	W	
7	6	5	4	3	2	1	0	
–	–	–	–	–	–	–	PIO	
W	W	W	W	W	W	W	W	

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **PIO : PIO clock enable**

0: No effect

1: Enable the PIO clock

PIO Disable Clock Register

PIO_DCR (0x0054)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	PIO
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **PIO : PIO clock disable**

0: No effect

1: Disable the PIO clock

PIO Power Management Status Register **PIO_PMSR (0x0058)** **Access: Read only**

31	30	29	28	27	26	25	24
–	–	IPIDCODE[25:20]					
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
IPIDCODE[19:12]							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
IPIDCODE[11:4]							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
IPIDCODE[3:0]				–	–	–	PIO
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **PIO : PIO clock status**

0: PIO clock is disabled

1: PIO clock is enabled

- **IPIDCODE[25:20]**

This field contains the version number of the module, coded on 26 bits.

PIO Control Register		PIO_CR (0x0060)						Access: Write only
31	30	29	28	27	26	25	24	
–	–	–	–	–	–	–	–	
W	W	W	W	W	W	W	W	
23	22	21	20	19	18	17	16	
–	–	–	–	–	–	–	–	
W	W	W	W	W	W	W	W	
15	14	13	12	11	10	9	8	
–	–	–	–	–	–	–	–	
W	W	W	W	W	W	W	W	
7	6	5	4	3	2	1	0	
–	–	–	–	–	–	–	SWRST	
W	W	W	W	W	W	W	W	

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **SWRST : PIO software reset**

0: No effect

1: Reset the PIO

A software triggered hardware reset of the PIO is performed. It resets all the registers except the PIO_PMSR register.

PIO Clear Status Register				PIO_CSR (0x006C)				Access: Write only
31	30	29	28	27	26	25	24	
P31	P30	P29	P28	P27	P26	P25	P24	
W	W	W	W	W	W	W	W	
23	22	21	20	19	18	17	16	
P23	P22	P21	P20	P19	P18	P17	P16	
W	W	W	W	W	W	W	W	
15	14	13	12	11	10	9	8	
P15	P14	P13	P12	P11	P10	P9	P8	
W	W	W	W	W	W	W	W	
7	6	5	4	3	2	1	0	
P7	P6	P5	P4	P3	P2	P1	P0	
W	W	W	W	W	W	W	W	

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **Px : PIO clear interrupt status**

0: No effect

1: Clear the corresponding bit in the interrupt status register

PIO Status Register **PIO_SR (0x0070)** **Access: Read only**

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **Px : PIO interrupt status**

These bits indicate for each pin when an input logic value change has been detected (rising or falling edge).

- 0: No input change has been detected on the corresponding pin since the corresponding bit was cleared.
- 1: At least one input change has been detected on the corresponding pin since the corresponding bit was cleared.

PIO Interrupt Enable Register **PIO_IER (0x0074)** **Access: Write only**

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **Px : PIO interrupt enable**

0: No effect

1: Enable an interrupt when a logic level change is detected on the corresponding pin

PIO Interrupt Disable Register **PIO_IDR (0x0078)** **Access: Write only**

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **Px : PIO interrupt disable**

0: No effect

1: Disable the interrupt on the corresponding pin. Logic level changes are still detected.

PIO Interrupt Mask Register **PIO_IMR (0x007C)** **Access: Read only**

31	30	29	28	27	26	25	24
P31	P30	P29	P28	P27	P26	P25	P24
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
P23	P22	P21	P20	P19	P18	P17	P16
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
P15	P14	P13	P12	P11	P10	P9	P8
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **Px : PIO interrupt mask**

0: The interrupt is not enabled on the corresponding input pin

1: The interrupt is enabled on the corresponding input pin

11

GENERAL PURPOSE TIMER (GPT)

1. OVERVIEW

1.1 FUNCTIONAL DESCRIPTION

The General Purpose Timer has a 16-bit timer/counter channel, a Power Management Controller and an optional Parallel I/O Controller. It has 2 operating modes (capture mode or waveform mode), to perform a wide range of functions including frequency measurement, event counting, interval measurement, pulse generation, delay timing, pulse width modulation and interrupt generation.

1.2 BLOCK DIAGRAM

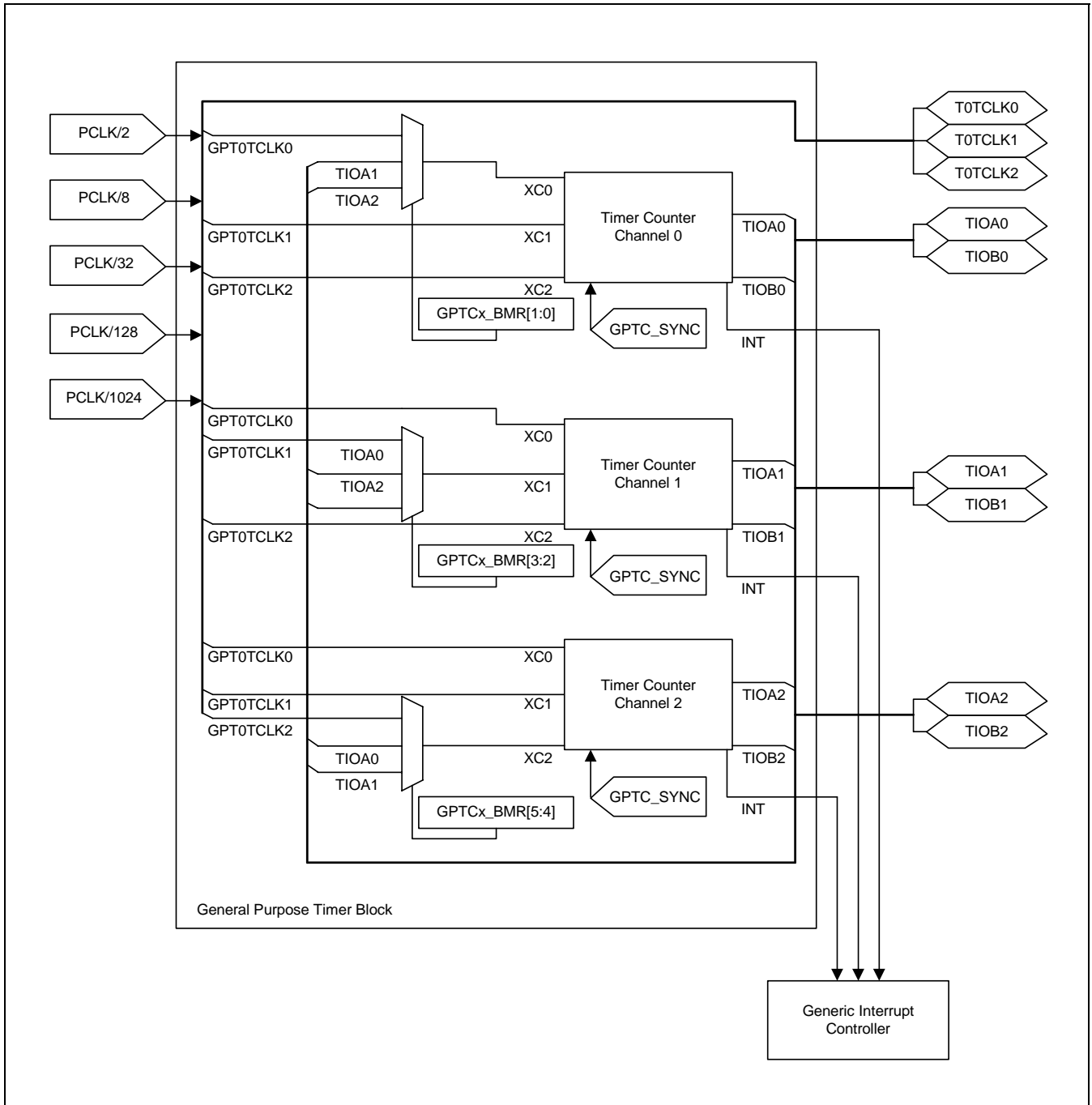


Figure 11-1. General Purpose Timer (3 channels) Block Diagram

2. EXTERNAL PIN DESCRIPTION

The following table shows the pin configurations in the different operating modes.

Table 11-1. Pin Descriptions Corresponding to Operation Modes

Pin Name	Capture Mode	Waveform Mode: Single waveform mode	Waveform Mode: Dual waveform mode
TIOA[2:0]	Input capture or trigger	Output waveform	Output waveform
TIOB[2:0]	Input trigger	Input trigger (GPT_MR.EEVT = 00)	Output waveform (GPT_MR.EEVT = 01, 10, 11)
TCLK[2:0]	Input external clock	Input external clock	Input external clock external trigger

3. FUNCTIONAL OPERATION

3.1 GENERAL DESCRIPTION

3.1.1 Functionality

The one channel General Purpose Timer, shown in its environment in the last figure, has the following components :

- One 16-bit resettable counter (GPT_CV)
- One 16-bit compare register (GPT_RC)
- 2 16-bit capture/compare registers (GPT_RA, GPT_RB)
- One multiplexer allowing the selection of 6 clocks : 5 internal and 1 external. It is possible to combine 2 clocks to generate a burst clock. The external clock can be used as external trigger source.
- An internal interrupt signal which can be programmed to generate processor interrupts via the Generic Interrupt Controller (module GIC). They are generated on the occurrence of:
 - Counter overflow
 - Load Register (A or B)
 - Equal Compare Register (A or B or C)
 - External edge detection
 - Overwrite register
- One software trigger which can be used as a software reset.
- One software reset which resets the channel and its associated registers (except PMC registers). 3 Parallel I/O pins which can be dedicated for multiple counter functions (Operation mode dependent).
 - TIOA, in capture mode, is an event input to load register A or register B or an input trigger. In waveform mode, it is an output to generate a waveform.
 - TIOB, in capture mode, is an input trigger. In waveform mode, it is either an output to generate a waveform (dual waveform mode) or an input trigger (single waveform mode).
 - TCLK, in capture mode and in waveform mode, is an external clock input or external trigger.

3.1.2 Clock Source

3.1.2.1 Description

The General Purpose Timer can use one of 6 different clocks. The 3 bits, CLKS [2:0], of the mode registers GPT_MR, determine whether the counter is clocked by one of the 5 internal clock sources (PCLK/x) generated in the prescaler block or the external clock (TCLK).

The counter clock sources can be any of the following:

- External event input (XC0, XC1, XC2)
- One of 5 internal clocks (PCLK/2, PCLK/8, PCLK/32, PCLK/128, PCLK/1024)
- A burst clock (see “Burst clock” explanations).

The maximal count duration when an internal clock is used, is determined by the internal clock (CLK, See Figure 12-2) and the prescale number.

Maximal Count Duration (seconds) = $2^{16}/\text{CLK}$ where CLK is in Hz.

Counter Resolution = $1/\text{CLK}$

3.1.2.2 Block Diagram

The following figure shows the clock selection block:

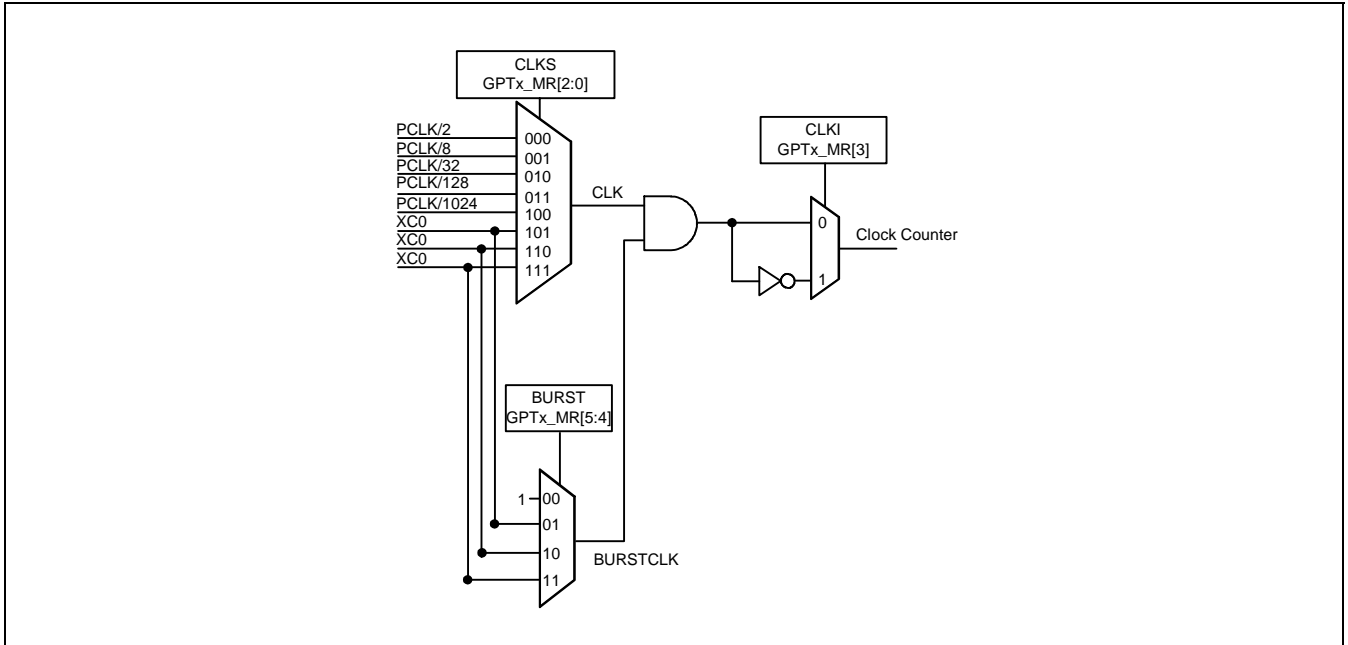


Figure 11-2. Clock Selection Block Diagram

3.1.2.3 External Clock

When an external clock source is used, the 16 bit counter can be programmed as a 16 bit event counter. An external transition (rising or falling following the state of CLKI, bit [3] of the Mode registers) increments the counter.

If an external clock is used, make sure that each of its pulses have a duration strictly higher than the PCLK period.

3.1.2.4 Burst Clock

If BURST (bits [5:4] of the mode register) selects an external clock, this clock is combined with CLK through a logical AND.

So, the considered General Purpose Timer channel is clocked by CLK only when CLKBURST is high as shown hereunder:

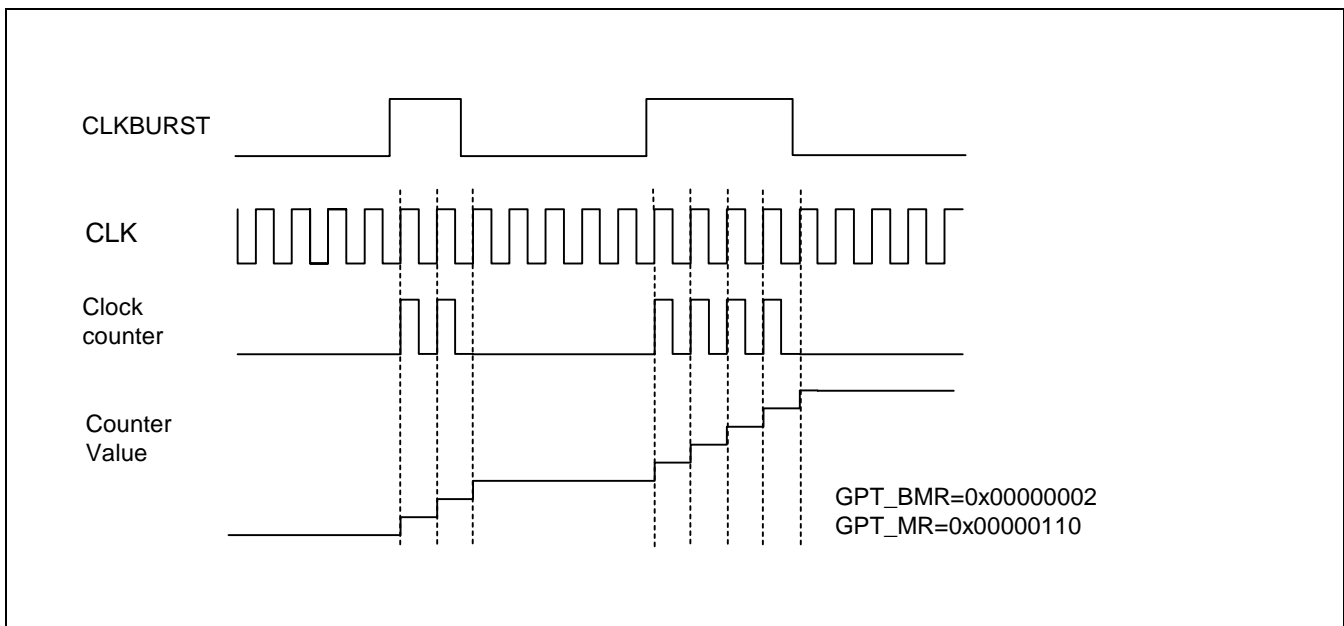


Figure 11-3. Burst Clock Timing Diagram

3.1.3 16–Bit Counter

The 16-bit counter is a free-running counter clocked by 8 sources : 5 internal and 3 external.

The program can access the counter value in real time in read only access with the counter value register GPT_CV.

When counter reset occurs, the counter is loaded with 0x0000 and begins its count if its clock is enabled (the clock is disabled or enabled by CLKDIS and CLKEN, bits [2] and [1] of the control register GPT_CR).

When the maximal value is reached (0xFFFF), the counter rolls over to a count of 0x0000, sets an overflow flag (COVFS, bit [0] of the status register GPT_SR), can generate an interrupt (enabled by COVFS, bit [0] of the interrupt enable register GPT_IER and disabled by COVFS, bit [0] of the interrupt disable register GPT_IDR), and continues to count up.

Counter reset:

During counting, the counter can be reset to 0x0000 following the action of:

- A software Trigger (SWTRG in GPT_CR)
- An external Trigger
- An equality on Compare C
- The synchronous bit TCSYNC, bit [1] of the block control register GPT_BCR.

Each time it is reset, the counter passes to 0x0000 at the next valid counter clock edge, as shown hereunder:

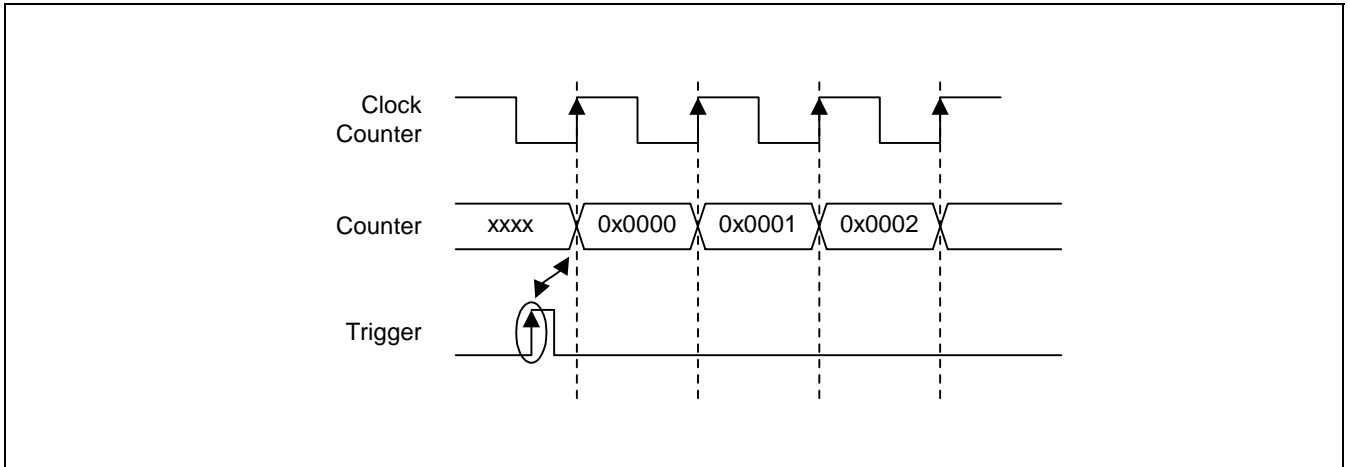


Figure 11-4. Counter Reset Diagram

3.1.4 16–Bit Registers

The one channel General Purpose Timer contains 3 capture/compare 16-bit registers.

The mode determines whether the capture/compare registers are used as capture registers or compare registers.

In capture mode, registers A and B are capture registers and can be loaded by TIOA edges.

In waveform mode, registers A and B are compare registers.

Register C is always a compare register.

The compare registers can generate a counter reset (RC) or a waveform modification (RA, RB and RC) when the counter reaches the value programmed in them.

In an application, the required compare register values must be calculated using the following equation:

$$\text{Compare Value} = (t \times \text{CLK}) - 1$$

where:

- t = desired timer compare period (in seconds)
- CLK = counter clock (in Hertz)

Example:

You want to determine the value needed in a compare register to obtain an equality with the counter after 0.1 second with PCLK = 30 MHz.

1/ Determine the minimal prescale value: You should divide PCLK by the maximal counter value 0xFFFF (65,535) to know the divisor factor.

$$30,000,000 / 65,535 = 457.77$$

The value of the divider greater than or equal to 457.77 is DIVmin = 1,024

⇒ CLK must therefore be at least PCLK/1,024 (29.3 kHz) if you want be to able to obtain an equal condition after 0.1 second.

2/ Calculate the register value with this clock: For this, use the following equation:

$$\text{Compare Value} = 0.1 \times (\text{PCLK}/1,024) - 1 = 0,1 \times (30,000,000/1,24) - 1 = 2928.69$$

Rounding value of 2,929 (0x0B71) generates a 0.01% error.

3.1.5 External Edge Detection

The General Purpose Timer contains many external edge detection options.

The operating mode determines their number:

- Capture mode:
 - TIOA as Load register and external trigger
 - TIOB as external trigger
- Waveform mode : dual waveform mode.
 - XC0, XC1, XC2 as external trigger
- Waveform mode : single waveform mode.
 - TIOB as external trigger

For each Edge detection, it is possible to choose a rising edge, a falling edge or both.

Whereas when a reset is caused, it occurs at the next valid counter clock edge.

If an external trigger is used, make sure that each of its pulses has a duration strictly higher than the PCLK period.

3.1.6 Interrupts

The one channel General Purpose Timer contains a total of 8 timer interrupts.

They can be enabled or disabled from the registers GPT_IER and GPT_IDR.

The programming mode determines which interrupts are available following the table:

Table 11-2. Available Interrupts

Interrupt Name	Capture Mode	Waveform Mode
Counter overflow interrupt COVFS	O	O
Load overrun interrupt LOVRS	O	X
Compare register A interrupt CPAS	X	O
Compare register B interrupt CPBS	X	O
Compare register C interrupt CPCS	O	O
Load capture register A interrupt LDRAS	O	X
Load capture register B interrupt LDRBS	O	X
External trigger interrupt ETRGS	O	O

3.2 TIMER IN CAPTURE MODE

3.2.1 General Description

The capture (wave measurement) mode is entered by setting WAVE (bit [15] in the Mode Register) to 0.

It is the default operating mode after a hardware reset.

It forces TIOA and TIOB pins as input pins.

The capture mode provides the possibility to determine the duration between 2 events.

An event may either be an external input signal on TIOA or TIOB or an internal event (software trigger or equality between the counter and a predefined compare value).

An external event (rising or falling edge) on TIOA can result in capture register A being loaded, capture register B being loaded or a trigger effect (reset and start the counter).

A predefined compare value (16 bits) can be set in the compare register C.

When the capture register B is loaded, it can disable the counter clock and/or stop the counter.

The user may choose an internal clock source (PCLK/2, PCLK/8, PCLK/32, PCLK/128 or PCLK/1024) or the external clock (XC0, XC1, XC2).

A burst mode is available. It generates a burst clock. For more details, refer to "Clock source" in the overview. 6 Interrupts can be produced:

- External trigger detected.
- RA loaded.
- RB loaded.
- Counter overflow (when the counter passes from 0xFFFF to 0x0000).
- Overrun (when RA or RB is reloaded before the old value is read).
- Compare RC (the counter reaches the value stored in register C).

Finally, the synchronize register can be used to cause a software trigger for reset and start the counter at the next valid counter clock edge on all channels at the same time.

The following figures show different applications we can do with the capture mode.

3.2.2 Measure TIOA Pulse and Phase Between TIOB and TIOA

A TIOB rising edge resets and starts the counter.

A rising TIOA edge loads RA and a falling TIOA edge loads RB.

The counter is stopped when RB is loaded.

An external trigger restarts a capture cycle.

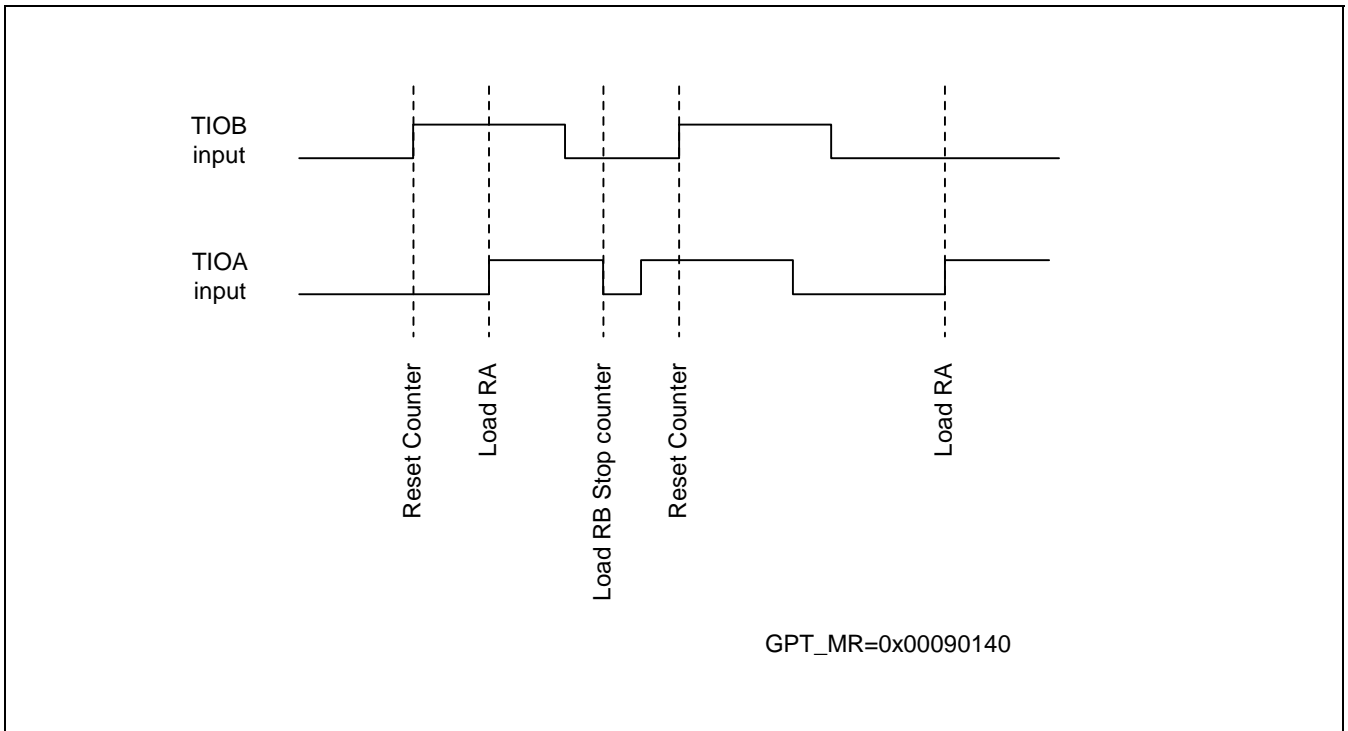


Figure 11-5. TIOA Pulse

RA contains the phase between TIOB and TIOA.

$(RB-RA)$ is the duration of the TIOA pulse.

Note that RB is not loaded when a falling edge occurs on TIOA after the reset of the counter. RA is always loaded first, and RB can not be loaded while RA is not loaded.

3.2.3 Measure Duration Between 2 Successive Rising TIOA Edges

A TIOB rising edge resets and starts the counter.

A rising TIOA edge after the reset loads RA and a second loads RB.

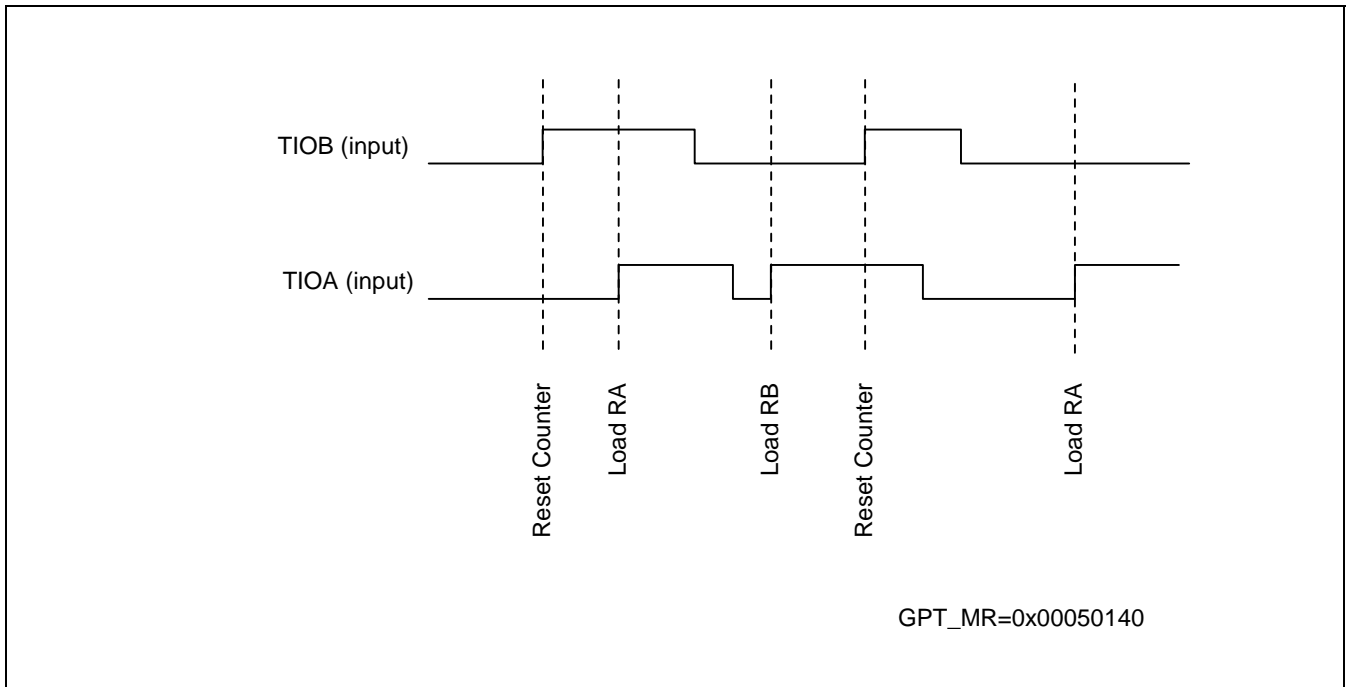


Figure 11-6. TIOA Edges

RA contains the phase between TIOB and TIOA.

(RB-RA) is the period of the TIOA pulse.

3.2.4 Measure the Duration of a TIOA Pulse or Period (TIOB is not used)

A TIOA falling edge resets and starts the counter and loads RB if RA is already loaded.

A TIOA rising edge loads RA.

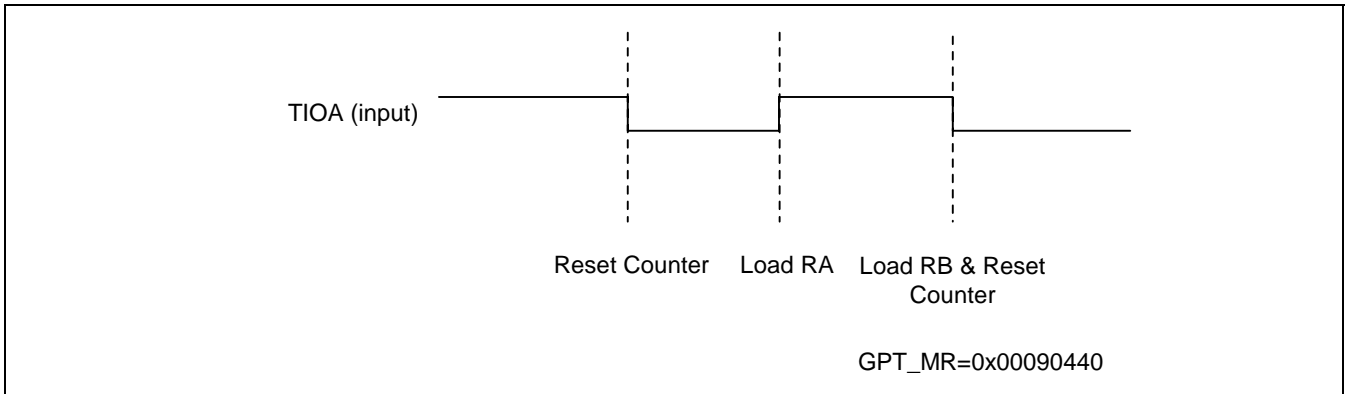


Figure 11-7. TIOA Pulse or Period

RA contains the duration of a TIOA pulse. (low level).
 RB contains the duration of the TIOA period. (high level)

3.2.5 Event Counter on the External Clock TCLK

The counter is incremented with each TCLK rising edge.

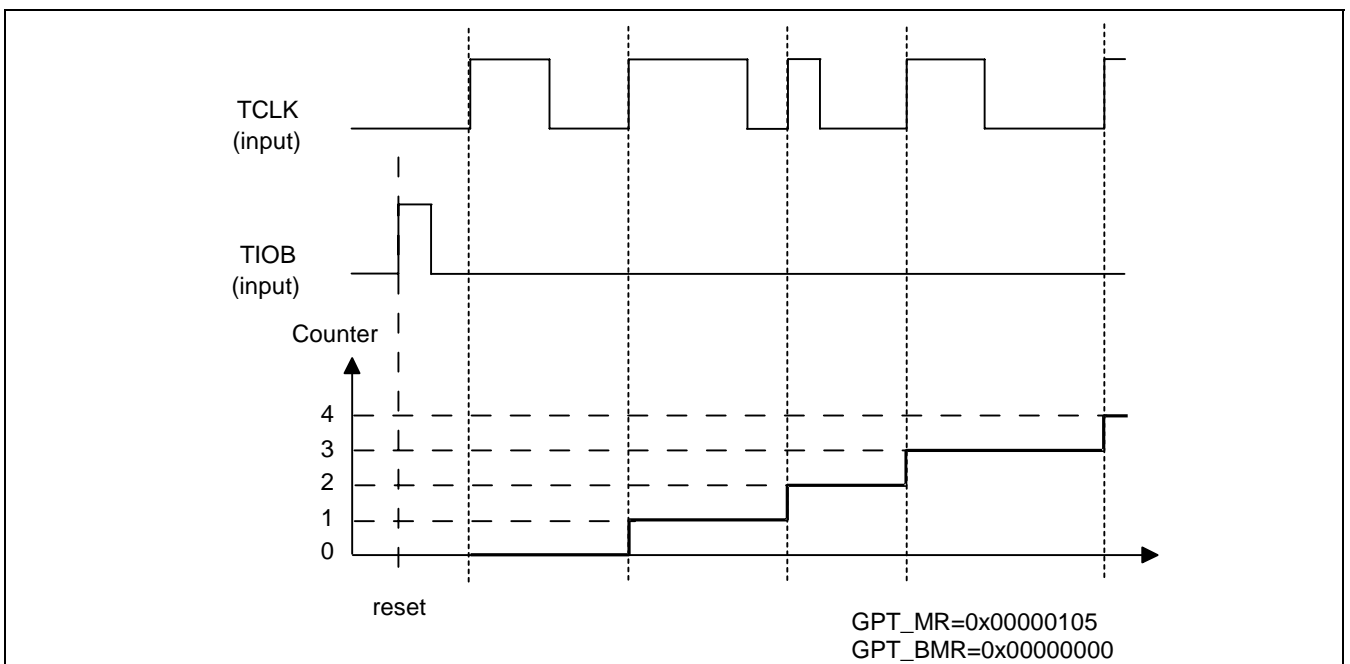


Figure 11-8. Counter on TCLK

The counter value contains the number of detected TCLK rising edge.

3.3 TIMER IN WAVEFORM MODE

3.3.1 General description

The waveform mode is entered by setting WAVE (bit [15] in the GPT_MR) to 1. It forces TIOA as an output pin. TIOB can be used either as an output (dual waveform mode) or as an input (single waveform mode).

The waveform mode provides the possibility to generate either symmetrical or variable duty-cycle waveforms.

TIOA pin can be controlled (set, cleared or toggled) by 4 events:

- A software trigger.
- An external event edge (rising, falling edge or both).
- An equality between the counter and compare register A value.
- An equality between the counter and compare register C value.

As output pin, TIOB pin can be controlled (set, cleared or toggled) by 4 events :

- A software trigger.
- An external event edge (rising, falling edge or both).
- An equality between the counter and compare register B value.
- An equality between the counter and compare register C value.

When TIOB is used as an external trigger source, the compare register B is not used.

When an equal condition on compare register C is detected, one of 3 events can occur :

- The counter can reset and start at the next valid counter clock edge.
- The counter can be stopped.
- The counter can be stopped and the counter clock disabled.

If this condition restarts the counter, the user can generate a continuous wave with a period proportional to compare register C +1 value. If it does not restart the counter, the user can generate a continuous waveform with a period proportional to 0xFFFF (maximal counter value : $2^{16}-1$).

The user may choose an internal clock source (PCLK/2, PCLK/8, PCLK/32, PCLK/128 or PCLK/1024) or the external clock (TCLK). A burst mode is available. It generates a burst clock. For more details, refer to “clock source” in the overview.

5 Interrupts can be produced :

- External trigger detected.
- Counter overflow (when the counter passes from 0xFFFF to 0x0000).
- Compare RA (the counter reaches the value stored in register A).
- Compare RB (the counter reaches the value stored in register B).
- Compare RC (the counter reaches the value stored in register C).

Finally, the synchronize register can be used to cause a software trigger for reset and start the counter at the next valid counter clock edge on all channels at the same time.

The following figures show different applications we can do with the waveform mode.

3.3.2 Dual Pulse Width Modulation (PWM) Generation (Dual waveform mode)

TIOA is toggled by RA and RC, TIOB by RB and RC.

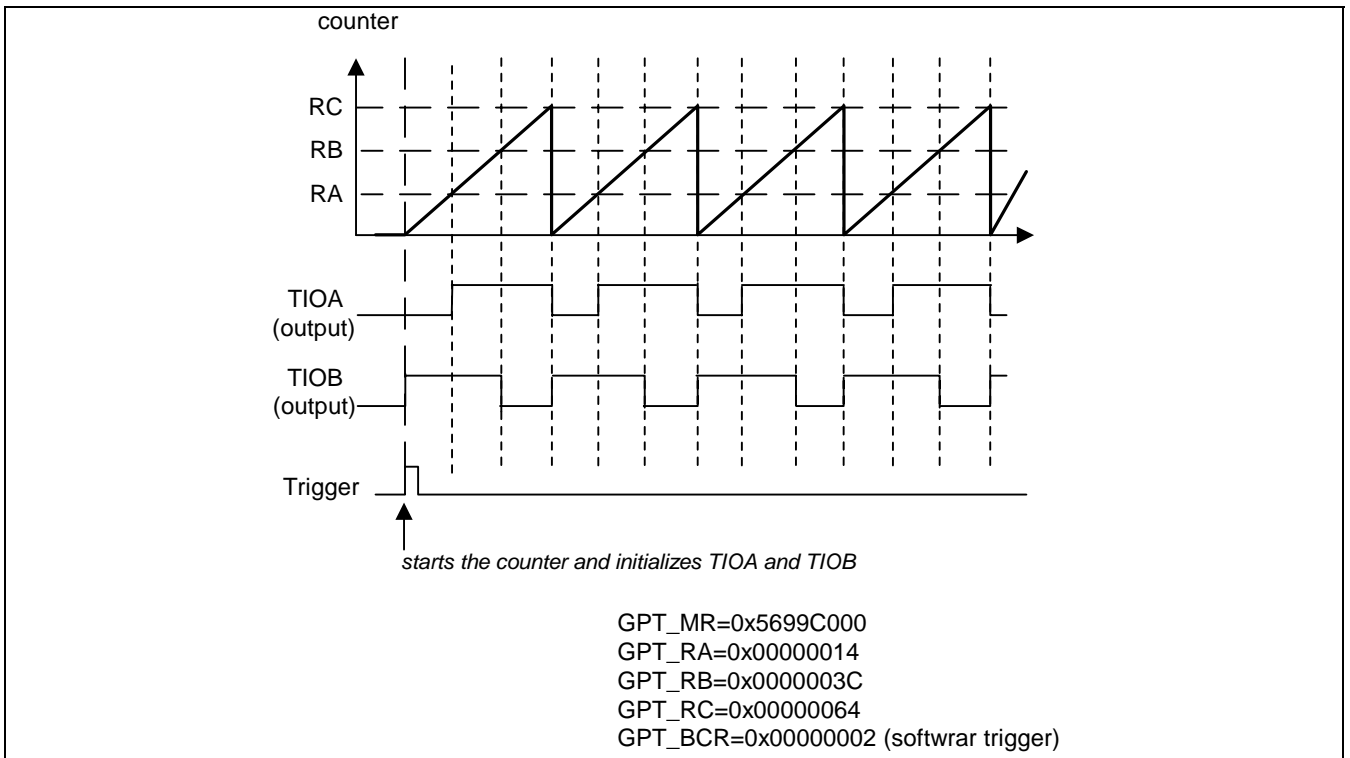


Figure 11-9. Dual Pulse Width Modulation

RC contains the both signals frequency. RA determines the TIOA duty cycle and RB the TIOB duty cycle.

3.3.3 Generation of 2 Identical out of Phase Square-Wave Signals (Dual waveform mode)

TIOA is toggled each time the counter reaches RA value, TIOB each time the counter reaches RC value.

A trigger (external or software) starts the counter and initializes TIOA and TIOB.

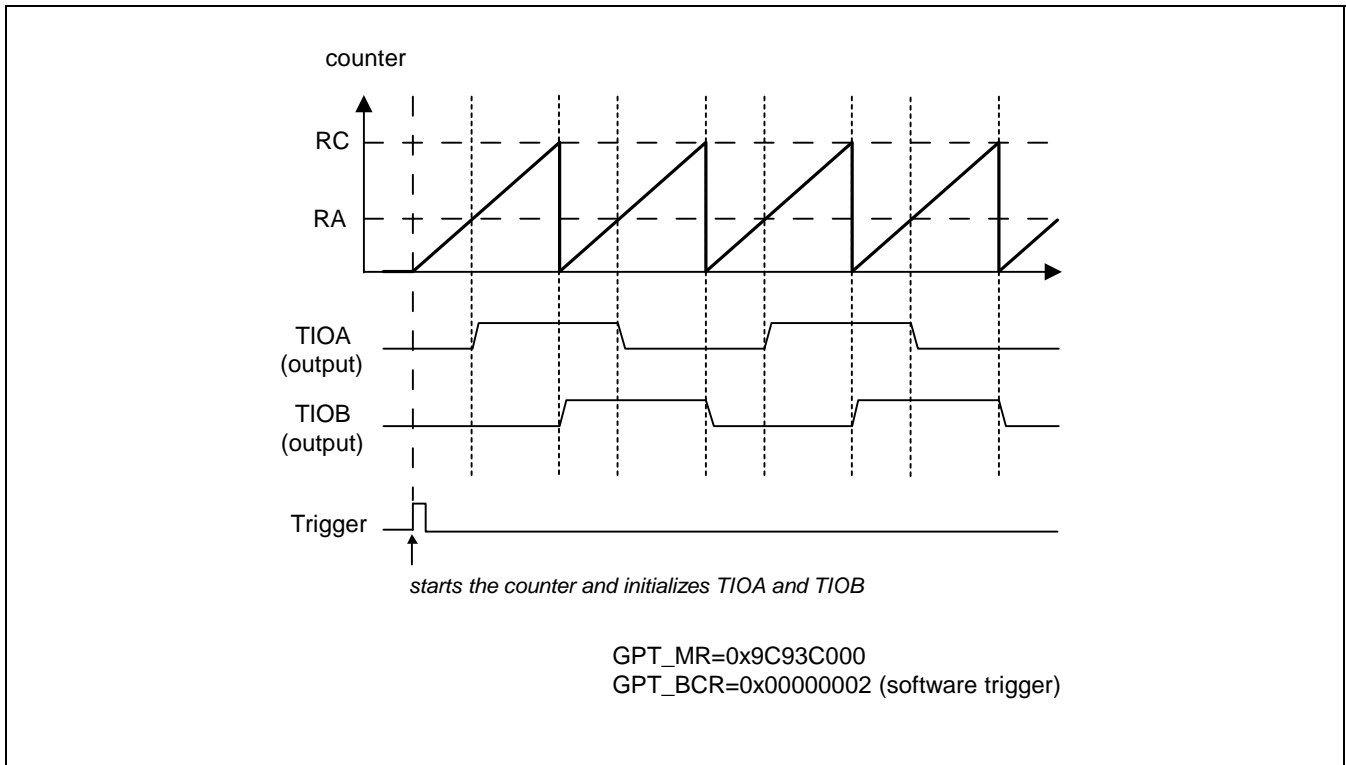


Figure 11-10. 2 Square Signals

RC contains the both signals frequency. RA contains the delay between the signals.

3.3.4 Pulse Generation (Dual waveform mode)

This application generates only one pulse on TIOA and on TIOB after a trigger occurs.

Their durations are given by RA value for TIOA pulse and RB value for TIOB pulse.

After each new trigger, another pulse is generated.

When a trigger occurs, the counter is reset at the next valid counter clock edge when the counter clock is enabled.

If we want to start the pulse exactly when the counter is reset, it is necessary to have RC=0.

So, it is the comparison with RC=0 which starts the pulse and not the trigger.

In this case, the user must disable the counter clock when a compare RB is detected to stop the counter.

To accept a new trigger, the user must then re-enable the counter clock.

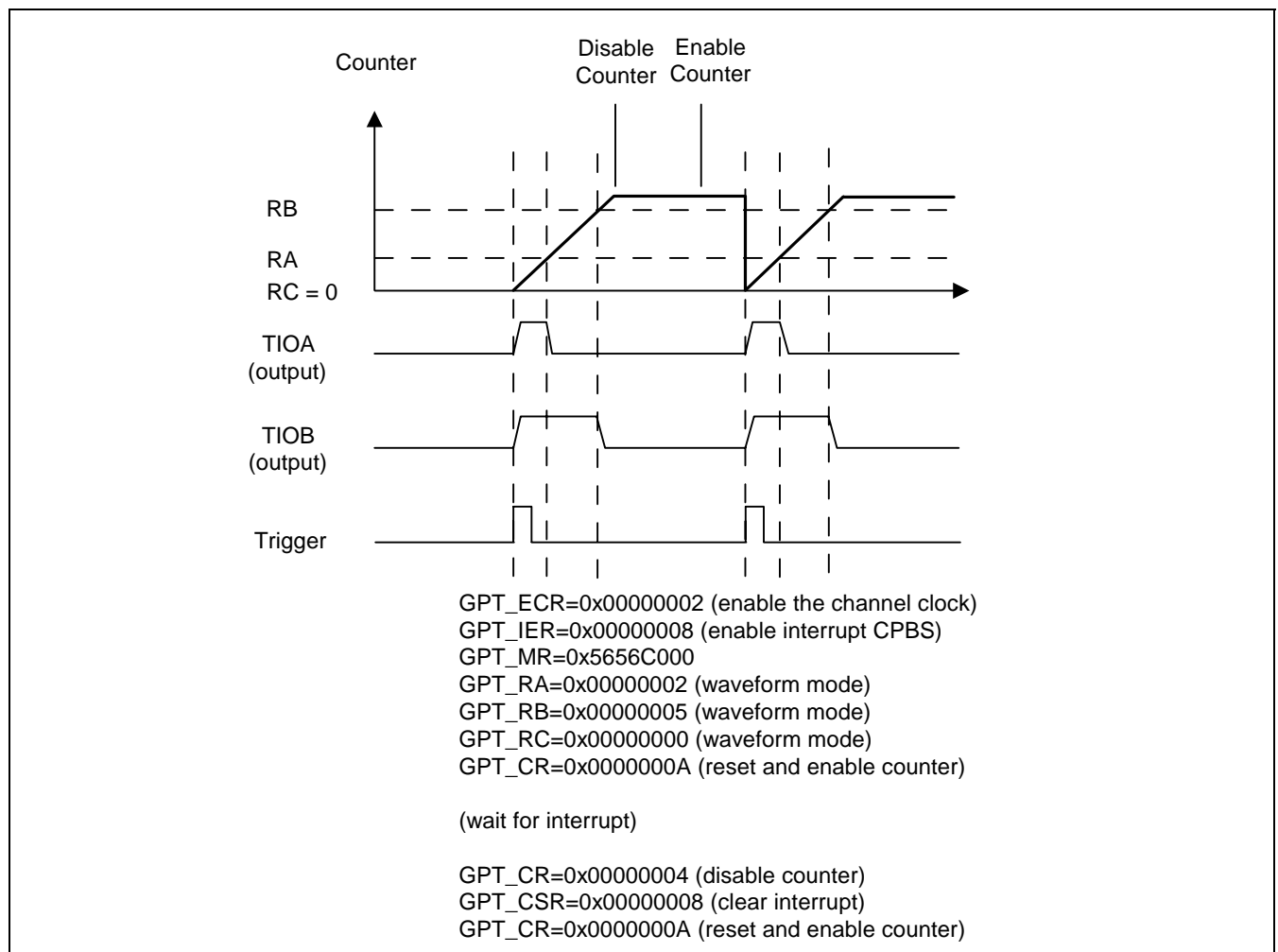


Figure 11-11. Pulse Generation

3.3.5 Trigger on TIOB Input Pin (Single waveform mode)

In each of the precedent examples, TIOB is used as an output but it is possible to use it as a trigger input and generate only TIOA output signal.

The following application is the same as the first one (in chapter Dual Pulse Width Modulation) where TIOB is not used as an output but as an input.

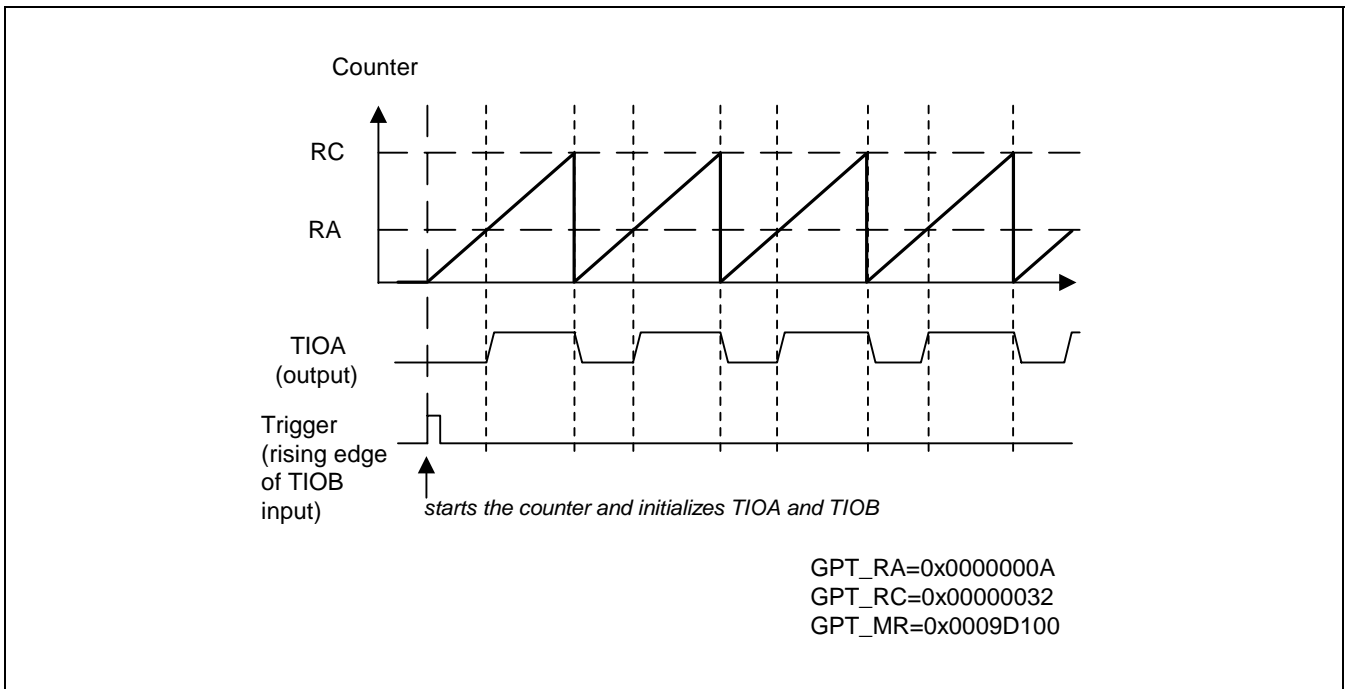


Figure 11-12. Single Waveform with Trigger on TIOB

3.3.6 Event Counter on an External Clock (TCLK0)

The counter is incremented with each TCLK rising edge. (See CLKI bit in the GPT_MR register)

This application can be generated in capture mode.

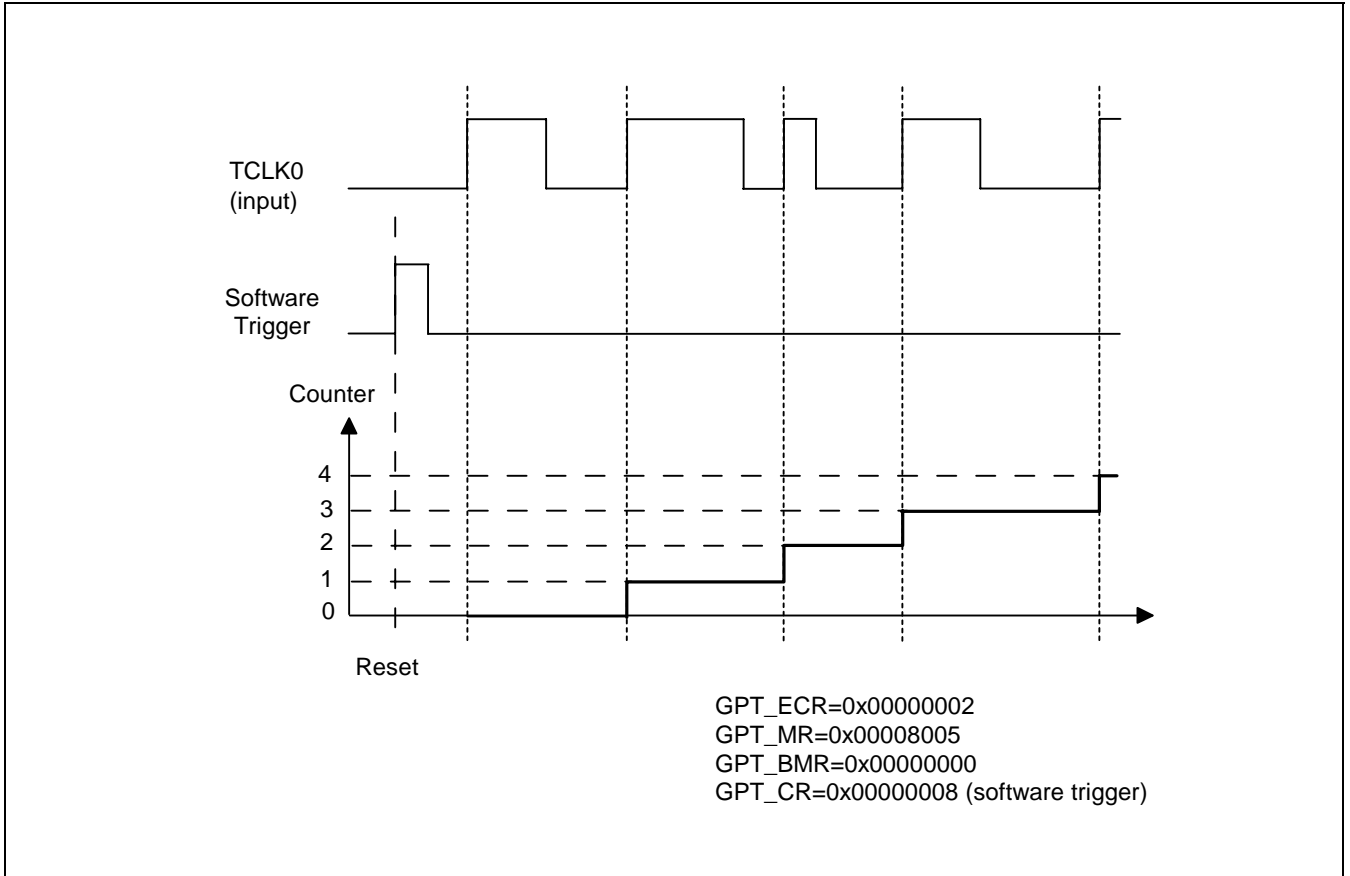


Figure 11-13. Event Counter

3.4 TIMER BLOCK PROGRAMMING

3.4.1 General Description

This module controls the entire general purpose timer with its 3 channels.

It has 2 functionalities :

- The block control register provides the means to synchronize the 3 General Purpose Timers channels. It can generate a software trigger on all 3 channels at the exactly the same time.
- The Block mode register provides the means to daisy chain 2 or 3 channels. In this way, the user can improve the counter capacity.

The following figure shows the block diagram.

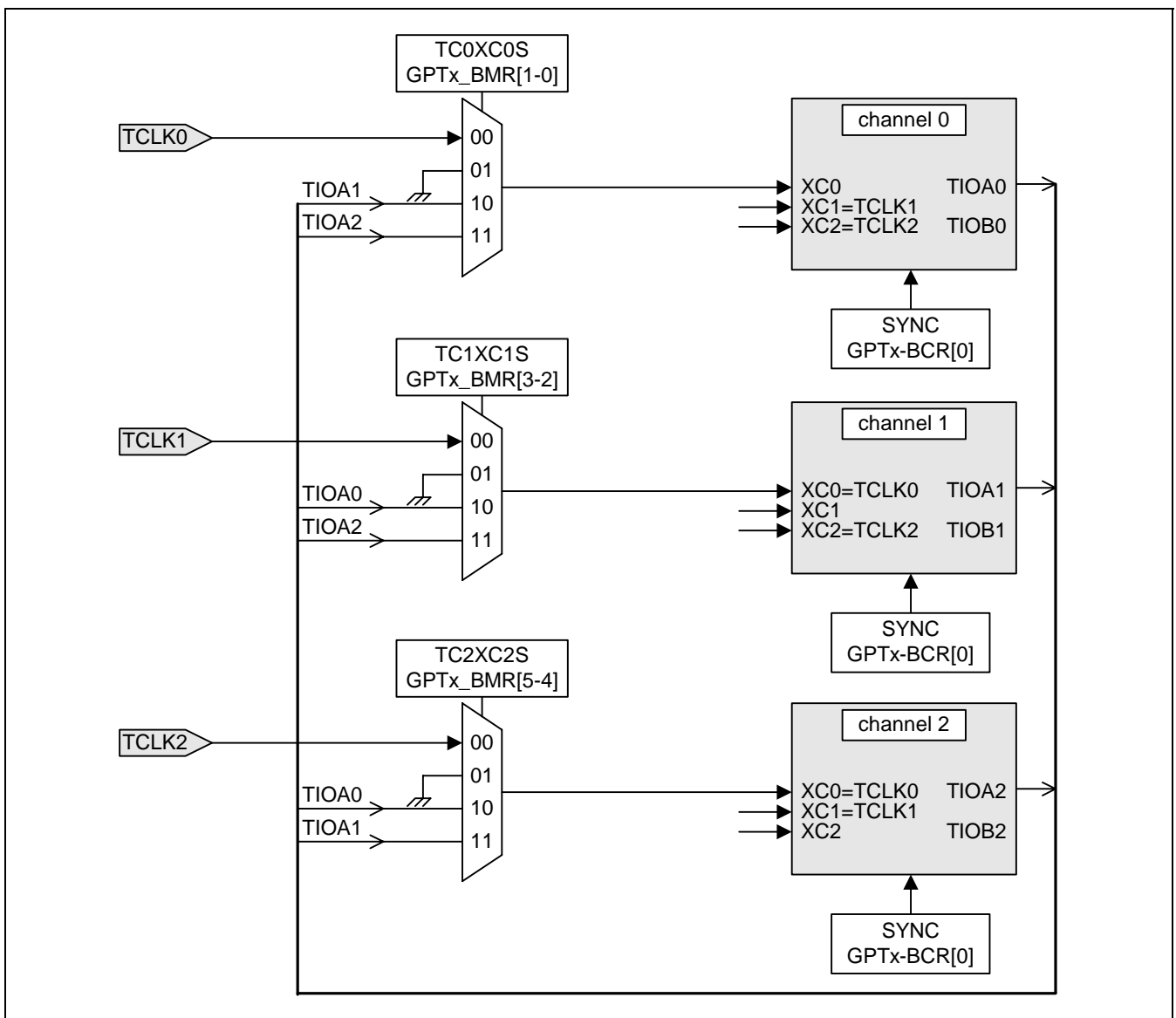


Figure 11-14. General Purpose Timer Block Programming Diagram

3.4.2 With channel 0, generate an external clock for the channel 1

With GPT0_RA and GPT0_RC, channel 0 generates a pulse width modulation output (PWM) in waveform mode which is used as an external clock by the channel 1.

Remark:

If RC is not used, this can generate a 32 bit counter.

Each time the counter 0 passes 0xFFFF (overflow condition), this increments the counter by 1.

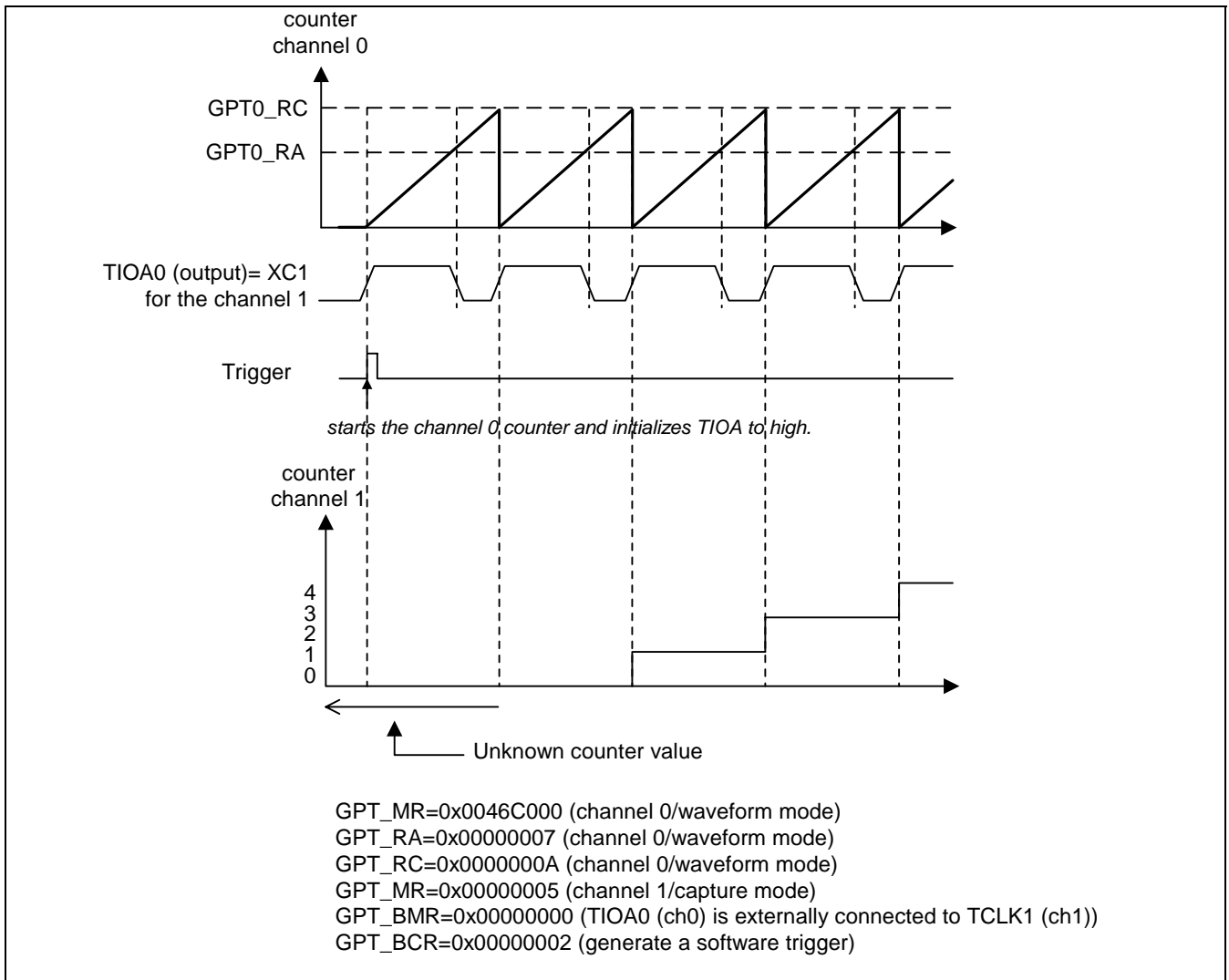


Figure 11-15. Application with General Purpose Timer Block Programming

3.5 PROGRAMMING EXAMPLES

Example of use of the General Purpose Timer :

We want to generate a tick (usually used in RTOS) of 10 ms. Then in the interrupt the counter is restarted. PCLK is 30MHz.

Configuration:

- Enable the clock of GPT peripheral by writing bit GPT in GPT_ECR
- Do a software reset of the GPT peripheral to be in a known state by writing bit SWRST in GPT_CR.
- Configuration of GPT_MR. We choose a Clock divider of 1024, Wave mode selected, CPCTRG is selected when counter is equal to RC register, then it cause a trigger on the counter (restart to 0), CPCSTOP allow the counter to stop when equal to RC.
- Configuration of GPT_RC : This will set the period which is set to 10ms, period = $PCLK / (GPT\ CLOCK\ divider * tick\ frequency)$ gives $30MHz / (1024 * 100)$
- Configuration of GPT_IER : We want to generate an interrupt when counter equal the RC register value by writing bit CPCS in GPT_IER. GIC must be configured.
- Start the counter by writing bit CLKEN and SWTRG in GPT_CR, then 10ms after, the interruption is generated.

Interruption Handling:

- IRQ Entry and call C function.
- Read GPT_SR and verify the source of the interrupt. This register is read and clear, care should be taken to keep status information to be able to proceed all case.
- Interrupt treatment : We restart the counter by writing bit CLKEN and SWTRG in GPT_CR, then 10ms after, another interruption is generated.
- IRQ Exit.

4. REGISTERS DESCRIPTION

Base Addresses – Channel 0 : 0xFFE1C000
 Channel 1 : 0xFFE1C100
 Channel 2 : 0xFFE1C200
 Multi channel control: 0xFFE1C300

Table 11-3. GPT Special Function Registers

Offset Address	Name	Description	R/W	Reset State
0x000 – 0x04C	–	Reserved	–	–
0x050	GPT_ECR	Enable clock register	W	–
0x054	GPT_DCR	Disable clock register	W	–
0x058	GPT_PMSR	Clock status register	R	(note)
0x05C	–	Reserved	–	–
0x060	GPT_CR	Control register	W	–
0x064	GPT_MR	Mode register	R/W	0x00000000
0x068	–	Reserved	–	–
0x06C	GPT_CSR	Clear status register	W	–
0x070	GPT_SR	Status register	R	0x00000X00
0x074	GPT_IER	Interrupt enable register	W	–
0x078	GPT_IDR	Interrupt disable register	W	–
0x07C	GPT_IMR	Interrupt mask register	R	0x00000000
0x080	GPT_CV	Counter value	R	0x00000000
0x084	GPT_RA	Capture–Compare register A	R/W	0x00000000
0x088	GPT_RB	Capture–Compare register B	R/W	0x00000000
0x08C	GPT_RC	Compare register C	R/W	0x00000000

Table 11-4. Multi Channels Control Registers Memory Map

Offset Address	Name	Description	R/W	Reset State
0x300	GPT_BCR	Block control register	W	–
0x304	GPT_BMR	Block mode register	R/W	0x00000000

NOTE: The reset value of the register depends on the level of the external pin after reset.

GPT Enable Clock Register		GPT_ECR (0x050)						Access: Write only	
31	30	29	28	27	26	25	24		
DBGEN	–	–	–	–	–	–	–		
W	W	W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16		
–	–	–	–	–	–	–	–		
W	W	W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8		
–	–	–	–	–	–	–	–		
W	W	W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0		
–	–	–	–	–	–	GPT	–		
W	W	W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **GPT : General Purpose Timer clock enable**

0: No effect

1: General Purpose Timer and clock divider block are enabled.

- **DBGEN : Debug mode enable**

0: No effect

1: DBGACK has influence on the GPT.

GPT Disable Clock Register

GPT_DCR (0x054)

Access: Write only

31	30	29	28	27	26	25	24
DBGEN	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	GPT	–
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **GPT : General Purpose Timer clock disable**

0: No effect

1: General Purpose Timer and clock divider block are disabled.

- **DBGEN : Debug mode disable**

0: No effect

1: DBGACK has no influence on the GPT.

GPT Power Management Status Register **GPT_PMSR (0x058)** **Access: Read only**

31	30	29	28	27	26	25	24
DBGEN	–	IPIDCODE[25:20]					
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
IPIDCODE[19:12]							
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
IPIDCODE[11:4]							
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
IPIDCODE[3:0]				–	–	GPT	–
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **GPT : General Purpose Timer clock status**

0: General Purpose Timer and clock divider block are disabled.

1: General Purpose Timer and clock divider block are enabled.

- **IPIDCODE[25:0]**

This field contains the version number of the module, coded on 26 bits.

- **DBGEN : Debug mode**

0: DBGACK has no influence on the GPT.

1: DBGACK has influence on the GPT.

GPT Control Register **GPT_CR (0x060)** **Access: Write only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	SWTRG	CLKDIS	CLKEN	SWRST
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **SWRST : Software reset**

0: No effect.

1: Generates a software reset. (GPT_PMSR is not reset.)

- **CLKEN : Counter clock enable**

0: No effect.

1: Enables counter clock if CLKDIS = 0.

- **CLKDIS : Counter clock disable**

0: No effect.

1: Disables counter clock.

- **SWTRG : Software trigger**

0: No effect.

1: Generates a software trigger.

This bit generates a software trigger for resetting and starting the counter at the next valid counter clock edge when the counter clock is enabled.

GPT Mode Register in Capture Mode				GPT_MR (0x064)				Access: Read/Write
31	30	29	28	27	26	25	24	
–	–	–	–	–	–	–	–	
W	W	W	W	W	W	W	W	
23	22	21	20	19	18	17	16	
–	–	–	–	LDRB[1:0]		LDRA[1:0]		
W	W	W	W	W	W	W	W	
15	14	13	12	11	10	9	8	
WAVE=0	CPCTRG	–	–	–	ABETRG	ETRGEDG[1:0]		
W	W	W	W	W	W	W	W	
7	6	5	4	3	2	1	0	
LDBDIS	LDBSTOP	BURST[1:0]		CLKI	CLKS[2:0]			
W	W	W	W	W	W	W	W	

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **CLKS[2:0] : Clock select**

PCLK are 5 internal clocks.

XC0, XC1 and XC2 are external clock.

For more details, see the part “clock source” in the chapter “Functional operation”.

Table 11-5. Clock Select

CLKS			Counter Clock Source
0	0	0	PCLK/2
0	0	1	PCLK/8
0	1	0	PCLK/32
0	1	1	PCLK/128
1	0	0	PCLK/1024
1	0	1	XC0
1	1	0	XC1
1	1	1	XC2

- **CLKI : Clock inverter**

- 0: Normal clock (The counter is incremented on a rising edge).
 1: Inverted clock (The counter is incremented on a falling edge).

- **BURST[1:0] : Burst**

This signal is combined with the selected clock through a logical AND.

For more details, see the part “clock source” in the chapter “Functional operation”.

Table 11-6. Burst

Burst		Burst Signal Selected
0	0	None
0	1	XC0
1	0	XC1
1	1	XC2

- **LDBSTOP : Load RB stops counter**

- 0: The counter is not stopped when RB is loaded.
 1: The counter is stopped when RB is loaded.

If the counter is stopped, it can restart (to 0x0000) just with a trigger condition.

If an equal condition on a TIOA edge induces both trigger condition and load capture register B, the trigger has no effect.

- **LDBDIS : Load RB disables clock**

- 0: The counter clock is not disabled when RB is loaded.
 1: The counter clock is disabled and the counter stopped when RB is loaded.

If the counter clock is disabled, it can be enabled only by asserting CLKEN, bit [1] of the control register.

- **ETRGEDG[1:0] : External trigger edge**

The external trigger source is either TIOA or TIOB following ABETRG, bit [10] of the mode register.

When an external trigger is generated, 3 events occur :

- It resets and starts the counter, only if counter clock is running.
- The ETRGS flag is set in the status register.
- If enabled, ETRGS interrupt is generated.

Table 11-7. External Trigger Edge

ETRGEDG		Edge
0	0	None
0	1	Rising edge
1	0	Falling edge
1	1	Each edge

- **ABETRG : TIOA or TIOB as external trigger**

0: Select TIOB as external trigger.

1: Select TIOA as external trigger.

NOTE: The counter can start only if the clock is enabled.

- **CPCTRG : Compare RC trigger**

0: An equal condition on RC does not cause a trigger.

1: An equal condition on RC causes a trigger.

NOTE: The counter can start only if the clock is enabled.

- **WAVE : Waveform**

0: Capture mode.

1: Waveform mode.

NOTE: The capture mode is the default mode after hardware reset.

- **LDRA[1:0] : Load RA**

These 2 bits activate one of 4 possible TIOA edge conditions to load RA, as shown in the table hereunder with x=A.

NOTE: The application must ensure that the event that loads RA will occur after the next counter clock edge following the configuration of LDRA (the counter is reset on the next counter clock edge following the configuration of LDRA).

- **LDRB[1:0] : Load RB**

These 2 bits activate one of 4 possible TIOA edge conditions to load RB, as shown in the table hereunder with x=B:

Table 11-8. Load Rx

LDRx		Edge
0	0	None
0	1	Rising edge
1	0	Falling edge
1	1	Each edge

GPT Mode Register in Waveform Mode **GPT_MR (0x064)** **Access: Read/Write**

31	30	29	28	27	26	25	24
BSWTRG[1:0]		BEEVT[1:0]		BCPC[1:0]		BCPB[1:0]	
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
ASWTRG[1:0]		AEEVT[1:0]		ACPC[1:0]		ACPA[1:0]	
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
WAVE=1	CPCTRG	–	ENETRG	EEVT[1:0]		EEVTEDG[1:0]	
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
CPCDIS	CPCSTOP	BURST[1:0]		CLKI	CLKS[2:0]		
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **CLKS[2:0] : Clock select**

For more details, see the part “clock source” in the chapter “Functional operation”.

Table 11-9. Clock Select

CLKS			Counter Clock Source
0	0	0	PCLK/2
0	0	1	PCLK/8
0	1	0	PCLK/32
0	1	1	PCLK/128
1	0	0	PCLK/1024
1	0	1	XC0
1	1	0	XC1
1	1	1	XC2

- **CLKI : Clock inverter**

- 0: Normal clock (The counter is incremented on a rising edge).
 1: Inverted clock (The counter is incremented on a falling edge).

- **BURST[1:0] : Burst**

This signal is combined with the selected clock through a logical AND.

For more details, see the part “clock source” in the chapter “Functional operation”.

Table 11-10. Burst

Burst		Burst Signal Selected
0	0	None
0	1	XC0
1	0	XC1
1	1	XC2

- **CPCSTOP : Compare RC stops the counter**

- 0: The counter is not stopped when an equal condition on RC is detected.
 1: The counter is stopped when an equal condition on RC is detected. If the counter is stopped, it can restart (to 0x0000) just with a trigger condition.
 If an equal condition on RC induces both trigger condition and stop counter, the trigger will have no effect.

- **CPCDIS : Compare RC disables clock**

- 0: The counter clock is not disabled when an equal condition on RC is detected.
 1: The counter clock is disabled and the counter stopped when an equal condition on RC is detected.
 If the counter clock is disabled, it can be enabled only by asserting CLKEN, bit [1] of the control register.

- **EEVTEG : External event edge**

These 2 bits activate one of 4 possible external event modes. The external event source is selected by EEVT, bits [11:10] of the mode register.

When an external trigger is generated, 5 events occur :

- The ETRGS flag is set in the status register.
- If enabled, ETRGS interrupt is generated.
- It can reset and start the counter at the next valid counter clock edge if ENETRGS (bit [12] of the mode register) is high.
- TIOA pin can be set, clear, toggle or unchanged following AEEVT (bits [21:20] of the mode register).
- TIOB pin can be set, clear, toggle or unchanged following BEEVT (bits [29:28] of the mode register).

Table 11-11. External Event Edge

EEVTEG		Edge
0	0	None
0	1	Rising edge
1	0	Falling edge
1	1	Each edge

- **EEVT : External event**

These bits select an external event source among 4 pins:

Table 11-12. External Event

EEVT		External Trigger
0	0	TIOB
0	1	XC0
1	0	XC1
1	1	XC2

If TIOB is selected, the mode is in the single waveform mode (see the synoptic). TIOA is used as an output and TIOB as an input.

The following bits are disabled :

- BSWTRG, bits [31:30] of the mode register.
- BEEVT, bits [29:28] of the mode register.
- BCPC, bits [27:26] of the mode register.
- BCPB, bits [25:24] of the mode register.
- Compare register B.

If an external clock is selected, the mode is in the dual waveform mode. TIOA and TIOB are used as outputs.

- **ENETRГ : Enable external trigger**

This bit determines whether an external event can be used as a trigger to reset and start the counter at the next valid counter clock edge.

The external event source is selected by EEVT, bits [11:10] of the mode register.

- 0: External event does not reset and start the counter. Selected external trigger can only be used to control TIOA and TIOB.
- 1: External trigger resets and starts the counter.

NOTE: The counter can start only if the clock is enabled.

- **CPCTRG : Compare RC trigger**

This bit determines whether an equal condition on the Compare C register can cause a trigger (reset and start the counter at the next valid counter clock edge).

0: An equal condition on RC does not cause a trigger.

1: An equal condition on RC causes a trigger.

NOTE: The counter can start only if the clock is enabled.

- **WAVE : Waveform**

0: Capture mode.

1: Waveform mode.

- **ACPA[1:0] : TIOA compare A**

These 2 bits determine the effect on the TIOA output pin caused by an equal comparison between the counter and the Compare register A value. See following table with xxx=CPA.

NOTE: If several events which control TIOA output (set, clear or toggle) arrive at the same time, only one will have an action according to the following priority order :

- ASWTRG (highest priority)
- AEEVT
- ACPC
- ACPA

- **ACPC[1:0] : TIOA compare C**

These 2 bits determine the effect on the TIOA output pin caused by an equal comparison between the counter and the Compare register C value. See following table with xxx=CPC.

- **AEEVT[1:0] : TIOA external event**

These 2 bits determine the effect on the TIOA output pin caused by an external event. The external event source is selected by EEVT, bits [11:10] of the mode register. See following table with xxx=EEVT.

- **ASWTRG[1:0] : TIOA software trigger**

These 2 bits determine the effect on the TIOA output pin caused by a software trigger. See following table with xxx=SWTRG :

Table 11-13. Effect on TIOA Event

Axxx		Waveform Pin
0	0	None
0	1	Set
1	0	Clear
1	1	Toggle

- **BCBP[1:0] : TIOB compare B**

These 2 bits determine the effect on the TIOB output pin caused by an equal comparison between the counter and the Compare register B value. These bits are active only if TIOB is not an input (see EEVT, bits [11:10] of the mode register). See following table with xxx=CPB.

NOTE: if several events which control TIOB output (set, clear or toggle) arrive at the same time, only one will have an action according to the following priority order :

- BSWTRG (highest priority)
- BEEVT
- BCPC
- BCPB

- **BCPC[1:0] : TIOB compare C**

These 2 bits determine the effect on the TIOB output pin caused by an equal comparison between the counter and the Compare register C value. These bits are active only if TIOB is not an input (see EEVT, bits [11:10] of the mode register). See following table with xxx=CPC.

- **BEEVT[1:0] : TIOB External event**

These 2 bits determine the effect on the TIOB output pin caused by an external event. The external event source is selected by EEVT, bits [11:10] of the mode register. These bits are active only if TIOB is not an input (see EEVT, bits [11:10] of the mode register). See following table with xxx=EEVT.

- **BSWTRG[1:0] : TIOB software trigger**

These 2 bits determine the effect on the TIOB output pin caused by a software trigger. These bits are active only if TIOB is not an input (see EEVT, bits [11:10] of the mode register). See following table with xxx=SWTRG :

Table 11-14. Effect on TIOB Event

Bxxx		Waveform Pin
0	0	None
0	1	Set
1	0	Clear
1	1	Toggle

GPT Clear Status Register				GPT_CSR (0x070)				Access: Write only
31	30	29	28	27	26	25	24	
–	–	–	–	–	–	–	–	
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	
23	22	21	20	19	18	17	16	
–	–	–	–	–	–	–	–	
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	
15	14	13	12	11	10	9	8	
–	–	–	–	–	–	–	–	
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	
7	6	5	4	3	2	1	0	
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS	
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **COVFS : Clear counter overflow interrupt**

0: No effect.
1: Clear COVFS interrupt.

- **LOVRS : Clear load overrun interrupt**

0: No effect.
1: Clear LOVRS interrupt.

- **CPAS : Clear compare register a interrupt**

0: No effect.
1: Clear CPAS interrupt.

- **CPBS : Clear compare register B interrupt**

0: No effect.
1: Clear CPBS interrupt.

- **CPCS : Clear compare register C interrupt**

0: No effect.
1: Clear CPCS interrupt.

- **LDRAS : Clear load register A interrupt**

0: No effect.
1: Clear LDRAS interrupt.

- **LDRBS : Clear load register B interrupt**

0: No effect.
1: Clear LDRBS interrupt.

- **ETRGS : Clear external trigger interrupt**

0: No effect.
1: Clear ETRGS interrupt.

GPT Status Register				GPT_SR (0x070)				Access: Read only
31	30	29	28	27	26	25	24	
–	–	–	–	–	–	–	–	
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	
23	22	21	20	19	18	17	16	
–	–	–	–	–	TCLKS	TIOAS	TIOBS	
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	
15	14	13	12	11	10	9	8	
–	–	–	–	–	MTIOB	MTIOA	CLKSTA	
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	
7	6	5	4	3	2	1	0	
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS	
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

NOTE: This register is a “read-active” register, which means that reading it can affect the state of some bits. When read GPT_SR register, following bits are cleared if set : COVFS, CPAS, CPBS, CPCS, ETRGS, TIOBS, TIOAS and TCLKS. When debugging, to avoid this behavior, users should use ghost registers.

- **COVFS : Counter overflow status**

This bit is set when a counter overflow is detected. An overflow occurs when the counter reaches its maximal value 0xFFFF ($2^{16}-1$) and passes to 0x0000.

- 0: No overflow detected.
- 1: Overflow detected since the last read of GPT_SR.

- **LOVRS : Load overrun status**

This bit is set when an overrun is detected. An overrun occurs when the capture registers A or B are reloaded before being read.

- 0: No overrun detected.
- 1: An overrun detected since last read of GPT_SR.

- **CPAS : Compare register A status**

This bit is set when the counter reaches the register A value.

- 0: Compare A condition has not occurred since the last read of GPT_SR.
- 1: Compare A condition has occurred since the last read of GPT_SR.

- **CPBS : Compare register B status**

This bit is set when the counter reaches the register B value.

- 0: Compare B condition has not occurred since the last read of GPT_SR.
- 1: Compare B condition has occurred since the last read of GPT_SR.

- **CPCS : Compare register C status**

This bit is set when the counter reaches the register C value.

- 0: Compare C condition has not occurred since the last read of GPT_SR.
- 1: Compare C condition has occurred since the last read of GPT_SR.

- **LDRAS : Load register A status**

- 0: Register A not loaded.
- 1: Register A loaded since last read of GPT_SR.

- **LDRBS : Load register B status**

- 0: Register B not loaded.
- 1: Register B loaded since last read of GPT_SR.

- **ETRGS : External trigger status**

This bit is set when an external trigger is detected. An external trigger occurs with a valid edge on the valid trigger pin.

- 0: External trigger not detected.
- 1: External trigger detected since last read of GPT_SR.

In Capture Mode:

The edge polarity is set by ETRGEDG, bits [9:8] of the mode register, and the valid trigger pin is set by ABETRG, bit [10] of the mode register.

In Waveform Mode:

The edge polarity is set by EEVTEG, bits [9:8] of the mode register, and the valid trigger pin is set by ENETRG, bit [12] of the mode register.

- **CLKSTA : Clock Status**

- 0: Clock disabled.
- 1: Clock enabled.

- **MTIOA : TIOA mirror**

This bit reflects the TIOA pin value.

As TIOA is an input after a hardware reset, its reset value is undefined.

- **MTIOB : TIOB mirror**

This bit reflects the TIOB pin value.

As TIOB is an input after a hardware reset, its reset value is undefined.

- **TIOBS : TIOB status**

- 0: At least one input change has been detected on the pin TIOB since the register was last read.
- 1: No input change has been detected on the TIOB pin since the register was last read.

- **TIOAS : TIOA status**

0: At least one input change has been detected on the TIOA pin since the register was last read.

1: No input change has been detected on the TIOA pin since the register was last read.

- **TCLKS : TCLK status**

0: At least one input change has been detected on the TCLK pin since the register was last read.

1: No input change has been detected on the TCLK pin since the register was last read.

GPT Interrupt Enable Register **GPT_IER (0x074)** **Access: Write only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	TCLKS	TIOAS	TIOBS
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **COVFS : Counter overflow interrupt enable**

0: No effect

1: Enable COVFS interrupt

- **LOVRS : Load overrun interrupt enable**

0: No effect

1: Enable LOVRS interrupt

- **CPAS : Compare register A interrupt enable**

0: No effect

1: Enable Compare A interrupt

- **CPBS : Compare register B interrupt enable**

0: No effect

1: Enable Compare B interrupt

- **CPCS : Compare register C interrupt enable**

0: No effect

1: Enable CPCS interrupt

- **LDRAS : Load register A interrupt enable**

0: No effect

1: Enable LDRAS interrupt

- **LDRBS : Load register B interrupt enable**

0: No effect

1: Enable LDRBS interrupt

- **ETRGS : External trigger interrupt enable**

0: No effect

1: Enable ETRGS interrupt

- **TIOBS : TIOB interrupt enable**

0: No effect

1: Enable TIOBS interrupt

- **TIOAS : TIOA interrupt enable**

0: No effect

1: Enable TIOAS interrupt

- **TCLKS : TCLK interrupt enable**

0: No effect

1: Enable TCLKS interrupt

GPT Interrupt Disable Register				GPT_IDR (0x078)				Access: Write only
31	30	29	28	27	26	25	24	
–	–	–	–	–	–	–	–	
W	W	W	W	W	W	W	W	
23	22	21	20	19	18	17	16	
–	–	–	–	–	TCLKS	TIOAS	TIOBS	
W	W	W	W	W	W	W	W	
15	14	13	12	11	10	9	8	
–	–	–	–	–	–	–	–	
W	W	W	W	W	W	W	W	
7	6	5	4	3	2	1	0	
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS	
W	W	W	W	W	W	W	W	

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **COVFS : Counter overflow interrupt disable**

0: No effect

1: Disable COVFS interrupt

- **LOVRS : Load overrun interrupt disable**

0: No effect

1: Disable LOVRS interrupt

- **CPAS : Compare register A interrupt disable**

0: No effect

1: Disable Compare A interrupt

- **CPBS : Compare register B interrupt disable**

0: No effect

1: Disable Compare B interrupt

- **CPCS : Compare register C interrupt disable**

0: No effect

1: Disable CPCS interrupt

- **LDRAS : Load register A interrupt disable**

0: No effect

1: Disable LDRAS interrupt

- **LDRBS : Load register B interrupt disable**

0: No effect

1: Disable LDRBS interrupt

- **ETRGS : External trigger interrupt disable**

0: No effect

1: Disable ETRGS interrupt

- **TIOBS : TIOB interrupt disable**

0: No effect

1: Disable TIOBS interrupt

- **TIOAS : TIOA interrupt disable**

0: No effect

1: Disable TIOAS interrupt

- **TCLKS : TCLK interrupt disable**

0: No effect

1: Disable TCLKS interrupt

GPT Interrupt Mask Register **GPT_IMR (0x07C)** **Access: Read only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	TCLKS	TIOAS	TIOBS
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **COVFS : Counter overflow status**

0: COVFS interrupt is disabled.

1: COVFS interrupt is enabled.

- **LOVRS : Load overrun status**

0: LOVRS interrupt is disabled.

1: LOVRS interrupt is enabled.

- **CPAS : Compare register A status**

This bit is set when the counter reaches the register A value.

0: Compare A condition has not occurred since the last read of GPT_SR.

1: Compare A condition has occurred since the last read of GPT_SR.

- **CPBS : Compare register B status**

This bit is set when the counter reaches the register B value.

0: Compare B condition has not occurred since the last read of GPT_SR.

1: Compare B condition has occurred since the last read of GPT_SR.

- **CPCS : Compare register C status**

0: CPCS interrupt is disabled.

1: CPCS interrupt is enabled.

- **LDRAS : Load register A status**

0: LDRAS interrupt is disabled.

1: LDRAS interrupt is enabled.

- **LDRBS : Load register B status**

0: LDRBS interrupt is disabled.

1: LDRBS interrupt is enabled.

- **ETRGS : External trigger status**

0: ETRGS interrupt is disabled.

1: ETRGS interrupt is enabled.

- **TIOBS : TIOB status**

0: TIOBS interrupt is disabled.

1: TIOBS interrupt is enabled.

- **TIOAS : TIOA status**

0: TIOAS interrupt is disabled.

1: TIOAS interrupt is enabled.

- **TCLKS : TCLK status**

0: TCLKS interrupt is disabled.

1: TCLKS interrupt is enabled.

GPT Counter Value		GPT_CV (0x080)						Access: Read only
31	30	29	28	27	26	25	24	
–	–	–	–	–	–	–	–	
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	
23	22	21	20	19	18	17	16	
–	–	–	–	–	–	–	–	
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	
15	14	13	12	11	10	9	8	
CV[15:8]								
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	
7	6	5	4	3	2	1	0	
CV[7:0]								
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **CV[15:0] : Counter value**

These 16 bits contain the counter value in real time.

The maximal counter value is 0xFFFF = 65535.

When a trigger occurs, the counter will be reset to 0x0000 at the next valid counter clock edge, if the counter clock is enabled.

GPT Register A in Capture Mode				GPT_RA (0x084)				Access: Read only
31	30	29	28	27	26	25	24	
–	–	–	–	–	–	–	–	
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	
23	22	21	20	19	18	17	16	
–	–	–	–	–	–	–	–	
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	
15	14	13	12	11	10	9	8	
RA[15:8]								
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	
7	6	5	4	3	2	1	0	
RA[7:0]								
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **RA : Register A value**

This register is loaded with the current counter value when a valid edge occurs on TIOA pin.

This valid edge is defined by LDRA (bits [17:16] of the mode register).

When this register is loaded, 2 events occur :

- The LDRAS flag is set in the status register.
- If enabled, LDRAS interrupt is generated.

This register can not be loaded if the counter is stopped or the counter clock disabled.

GPT Register B in Capture Mode				GPT_RB (0x088)				Access: Read only
31	30	29	28	27	26	25	24	
–	–	–	–	–	–	–	–	
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	
23	22	21	20	19	18	17	16	
–	–	–	–	–	–	–	–	
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	
15	14	13	12	11	10	9	8	
RB[15:8]								
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	
7	6	5	4	3	2	1	0	
RB[7:0]								
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **RB : Register B value**

This register is loaded with the current counter value when a valid edge occurs on TIOA pin.

This valid edge is defined by LDRB (bits [19:18] of the mode register).

When this register is loaded, 4 events occur :

- The LDRBS flag is set in the status register.
- If enabled, LDRBS interrupt is generated.
- The counter clock can be disabled according to LDBDIS (bit [7] of the mode register).
- The counter can be stopped according to LDBSTOP (bit [6] of the mode register).

This register can not be loaded if the counter is stopped or the counter clock disabled.

GPT Register C in Capture Mode

GPT_RC (0x08C)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
RC[15:8]							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
RC[7:0]							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **RC : Register C value**

When the counter reaches this value, 3 events occur :

- The CPCS flag is set in the status register.
- If enabled, CPCS interrupt is generated.
- If CPCTRG (bit [14] of the mode register) is high, the counter is reset, and it restarts at 0x0000.

GPT Register A in Waveform Mode				GPT_RA (0x084)				Access: Read/Write
31	30	29	28	27	26	25	24	
–	–	–	–	–	–	–	–	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
23	22	21	20	19	18	17	16	
–	–	–	–	–	–	–	–	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
15	14	13	12	11	10	9	8	
RA[15:8]								
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
7	6	5	4	3	2	1	0	
RA[7:0]								
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **RA[15:0] : Register A value**

When the counter reaches this value, 3 events occur :

- The CPAS flag is set in the status register.
- If enabled, CPAS interrupt is generated.
- TIOA pin can be set, clear, toggle or unchanged following bits ACPA (bits [17:16] of the mode register).

GPT Register B in Waveform Mode **GPT_RB (0x088)** **Access: Read/Write**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
RB[15:8]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
RB[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **RB[15:0] : Register B value**

When the counter reaches this value, 3 events occur :

- The CPBS flag is set in the status register.
- If enabled, CPBS interrupt is generated.
- TIOB pin can be set, clear, toggle or unchanged following bits BCPB (bits [25:24] of the mode register). These bits are active only if TIOB is not an input (see EEVT, bits [11:10] of the mode register).

GPT Register C in Waveform Mode				GPT_RC (0x08C)				Access: Read/Write
31	30	29	28	27	26	25	24	
–	–	–	–	–	–	–	–	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
23	22	21	20	19	18	17	16	
–	–	–	–	–	–	–	–	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
15	14	13	12	11	10	9	8	
RC[15:8]								
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
7	6	5	4	3	2	1	0	
RC[7:0]								
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **RC[15:0] : Register C value**

When the counter reaches this value, 7 events occur :

- The CPCS flag is set in the status register.
- If enabled, CPCS interrupt is generated.
- If bit CPCTRG (bit [14] of the mode register) is high, the counter is reset and restarts at 0x0000 at the next valid counter clock edge.
- The counter clock can be disabled according CPCDIS (bit [7] of the mode register).
- The counter can be stopped according CPCSTOP (bit [6] of the mode register).
- TIOA pin can be set, clear, toggle or unchanged following bits ACPC (bits [19:18] of the mode register).
- TIOB pin can be set, clear, toggle or unchanged following bits BCPC (bits [27:26] of the mode register).

GPT Block Control Register

GPT_BCR (0x300)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	TCSYNC	SWRST
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **SWRST : Software reset**

This bit generates software reset on the 3 timer channels simultaneously.

0: No effect.

1: Generate software reset.

- **TCSYNC : Synchronization bit**

This bit generates a software trigger on the general purpose timer 3 channels simultaneously.

A software trigger resets and starts the counter at the next valid counter clock edge.

0: No effect.

1: Resets and starts all 3 timers channel simultaneously.

GPT Block Mode Register		GPT_BMR (0x304)				Access: Read/Write	
31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
–	–	TC2XC2S[1:0]		TC1XC1S[1:0]		TC0XC0S[1:0]	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- TC0XC0S : TCK0 XC0 selection.**

These bits select the external clock XC0 source for the channel 0 as shown hereunder.

Channel 0 External Clock Select

TC0XC0S[1:0]		Selected Signal
0	0	TCLK0
0	1	none
1	0	TIOA1
1	1	TIOA2

- TC1XC1S : TCK1 XC1 selection**

These bits select the external clock XC1 source for the channel 1 as shown hereunder.

Channel 1 External Clock Select

TC1XC1S[1:0]		Selected Signal
0	0	TCLK1
0	1	none
1	0	TIOA0
1	1	TIOA2

- **TC2XC2S : TCK2 XC2 selection**

These bits select the external clock XC2 source for the channel 2 as shown hereunder.

Channel 2 External Clock Select

TC2XC2S[1:0]		Selected Signal
0	0	TCLK2
0	1	none
1	0	TIOA0
1	1	TIOA1

12

GENERIC INTERRUPT CONTROLLER (GIC)

1. OVERVIEW

1.1 FUNCTIONAL DESCRIPTION

The GIC is an 16–level priority, individually maskable, vectored interrupt controller. It can substantially reduce the software and real time overhead in handling internal and external interrupts.

The interrupt controller is connected to the nFIQ (Fast Interrupt Request) and the nIRQ (standard Interrupt Request) inputs of an ARM processor. The nIRQ and nFIQ lines can be asserted by all interrupt sources from INT[0] to INT[63].

An 16–level priority encoder allows the customer to define the priority between the different interrupt sources. The interrupt sources from IRQ[0] to IRQ[63] can be programmed to be positive or negative edge triggered or high or low level sensitive.

Table 12-1. Interrupt Sources

Interrupt Source	Description	GIC Bit	Interrupt Source	Description	GIC Bit
0	Data flash controller	DFC	12	UART	UART0
1	Stop mode controller	STOP_MODE	13	Inter integrated circuit	I2C0
2	Low voltage detector	LVD	14	UART	UART1
3	Interleave flash controller	IFC	15	Controller area network	CAN1
4	Pulse width modulation	PWM0	16	–	–
5	Analog to digital converter	ADC0	17	–	–
6	Serial peripheral Interface 8 bits	SPI0	18	Capture	CAPT0
7	Watchdog	WD	19	–	–
8	Controller area network	CAN0	20	USART	USART0
9	General purpose timer channel	GPT0CH0	21	Inter Integrated circuit	I2C1
10	Simple timer	ST0	22	Software interrupt	SWIRQ1
11	Simple timer	ST1	23	–	–

Table 12-1. Interrupt Sources (Continued)

Interrupt Source	Description	GIC Bit	Interrupt Source	Description	GIC Bit
24	Serial peripheral Interface 16 bits	SPI1	44	External interrupt	IRQ4
25	Stamp timer	STT	45	External interrupt	IRQ5
26	General purpose timer channel	GPT0CH1	46	External interrupt	IRQ6
27	General purpose timer channel	GPT0CH2	47	External interrupt	IRQ7
28	Lite direct memory access	LDMA	48	External interrupt	IRQ8
29	General purpose I/O 0	GPIO0	49	External interrupt	IRQ9
30	General purpose I/O 1	GPIO1	50	External interrupt	IRQ10
31	General purpose I/O 2	GPIO2	51	External interrupt	IRQ11
32	–	–	52	–	–
33	Stepper motor controller	SMC0	53	–	–
34	Stepper motor controller	SMC1	54	–	–
35	Stepper motor controller	SMC2	55	–	–
36	Stepper motor controller	SMC3	56	–	–
37	–	–	57	–	–
38	–	–	58	–	–
39	Software interrupt	SWIRQ2	59	External interrupt	IRQ19
40	External interrupt	IRQ0	60	Stable interrupt	STABLE
41	External interrupt	IRQ1	61	Software interrupt	SWIRQ4
42	External interrupt	IRQ2	62	Software interrupt	SWIRQ5
43	External interrupt	IRQ3	63	Software interrupt	SWIRQ6

1.2 BLCOK DIAGRAM

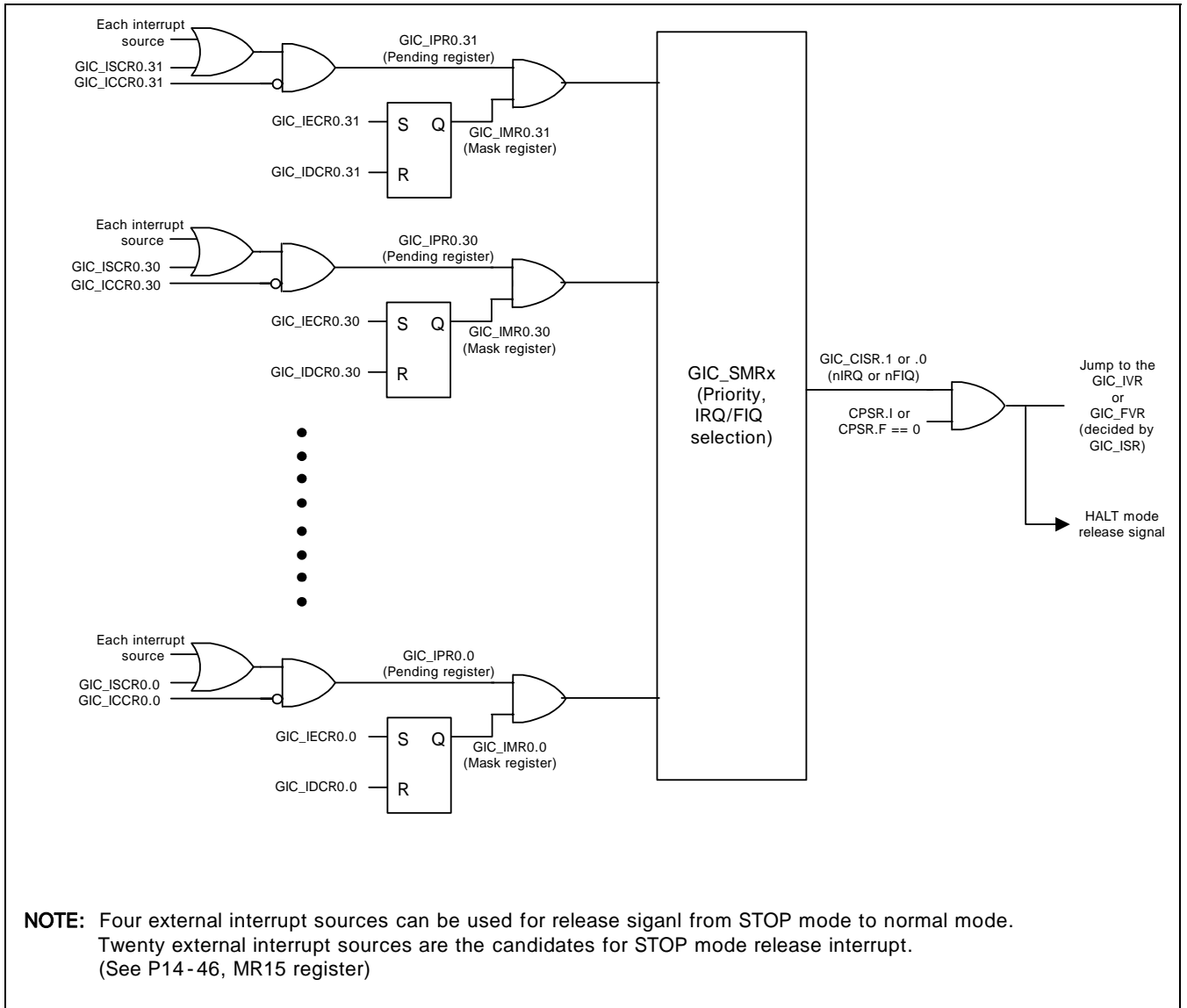


Figure 12-1. Generic Interrupt Controller Block Diagram

1.3 EXTERNAL PIN DESCRIPTION

Table 12-2. Generic Interrupt Controller Pin Description

Pin Name	Function	I/O Type	Active Level	Comment
IRQ[11:0]	External interrupt	I	High or Low	—

2. FUNCTIONAL OPERATION

2.1 INTERRUPT HANDLING

2.1.1 Hardware Interrupt Vectoring

The hardware interrupt vectoring reduces the number of instructions to reach the interrupt handler to only one. By storing the following instruction at address 0x00000018, the processor loads the program counter with the interrupt handler address stored in the GIC_IVR register. Execution is then vectored to the interrupt handler corresponding to the current interrupt.

ldr PC,[PC,# -0xF20]

The current interrupt is the interrupt with the highest priority when the Interrupt Vector Register (GIC_IVR) is read.

The value read in the GIC_IVR corresponds to the address stored in the Source Vector Register (GIC_SVR) of the current interrupt. Each interrupt source has its corresponding GIC_SVR. In order to take advantage of the hardware interrupt vectoring it is necessary to store the address of each interrupt handler in the corresponding GIC_SVR, at system initialization.

2.1.2 Priority Controller

The nIRQ and nFIQ line are controlled by an 16-level priority encoder. Each source has a programmable priority level of 7 to 0. Level 7 is the highest priority and level 0 the lowest. Each source can be redirected to nFIQ or nIRQ line, nFIQ has higher priority related to nIRQ.

So the highest priority is an interrupt source redirected to nFIQ line with a priority level of 7. And the lowest priority is a source interrupt redirected to nIRQ line with a priority level of 0. The table here after shows the priority levels from lowest to highest.

Table 12-3. Priority Levels

Line	Level	Priority
nFIQ	7	Highest
nFIQ	6	
–	–	–
nFIQ	1	
nFIQ	0	
nIRQ	7	
nIRQ	6	
–	–	–
nIRQ	1	
nIRQ	0	Lowest

When the GIC receives more than one unmasked interrupt at a time, the interrupt with the highest priority is serviced first. If both interrupts have equal priority, the interrupt with the lowest interrupt source number is serviced first (See Table 12-1).

The current priority level is defined as the priority level of the current interrupt when registers GIC_IVR or GIC_FVR (depending of the source of the interruption) is read (the interrupt which will be serviced).

2.1.3 Interrupt Nesting

If a higher priority unmasked interrupt occurs while an interrupt already exists, there are two possible outcomes depending on whether the GIC_IVR or GIC_FVR has been read.

- If the nIRQ/nFIQ line has been asserted but the GIC_IVR or GIC_FVR has not been read, then the processor will read the new higher priority interrupt handler address in the GIC_IVR (or GIC_FVR) register and the current interrupt level is updated.
- If the processor has already read the GIC_IVR (or GIC_FVR) then the nIRQ or nFIQ line is reasserted. When the processor has authorized nested interrupts to occur and reads the GIC_IVR (or GIC_FVR) again, it reads the new higher priority interrupt handler address. At the same time the current priority value is pushed onto a first-in last-out stack and the current priority is updated to the higher priority.

When the end of interrupt command register (GIC_EOICR) is written, the current interrupt level is updated with the last stored interrupt level from the stack (if any). Hence at the end of a higher priority interrupt, the GIC returns to the previous state corresponding to the preceding lower priority interrupt which had been interrupted.

2.1.4 Software Interrupt Handling

The hardware interrupt handler must read the GIC_IVR or GIC_FVR as soon as possible. This de-asserts the nIRQ or nFIQ request to the processor and clears the interrupt in case it is programmed to be edge triggered. This permits the GIC to assert the nIRQ or nFIQ line again when a higher priority unmasked interrupt occurs. At the end of the interrupt service routine, the end of interrupt command register (GIC_EOICR) must be written. This allows pending interrupts to be serviced.

2.1.5 Interrupt Masking

Each interrupt source INT[63:0], can be enabled or disabled using the command registers GIC_IECR and GIC_IDCR.

The interrupt mask can be read in the read only register GIC_IMR. A disabled interrupt does not affect the servicing of other interrupts.

2.1.6 Fast Interrupt Request

All external lines INT[63:0] can be the source of the nFIQ request to the processor. It can be programmed to be positive or negative edge triggered or high or low level sensitive in the GIC_SMR[31:0] register.

The fast interrupt handler address can be stored in the GIC_SVR[31:0] register. The value written into these register is available by reading the GIC_FVR register when an FIQ interrupt is raised. By storing the following instruction at address 0x0000001C (if an ARM processor is used as the host processor), the processor will load the program counter with the interrupt handler address stored in the GIC_FVR register.

```
ldr PC,[PC,# -0xF20]
```

Alternatively the interrupt handler can be stored starting from address 0x0000001C as described in ARM datasheet.

2.1.7 Software Interrupt

Any interrupt source of the GIC can be a software interrupt. It must be programmed to be edge triggered in order to set or clear it by writing to the GIC_ISCR and GIC_ICCR. This is totally independent of the SWI instruction of the ARM processor.

2.2 STANDARD INTERRUPT SEQUENCE

For details on the registers mentioned in the steps below, refer to an ARM Datasheet. It is assumed that:

- GIC_SVR must be written with corresponding interrupt vector address and interrupts must be enabled by writing GIC_IECRx register.
- The Instruction at address 0x18(IRQ exception vector address) is : *ldr pc, [pc, #-0xF20]*.

When nIRQ is asserted, if the bit I of CPSR is 0, the sequence is:

- The CPSR is stored in SPSR_irq, the current value of the Program Counter is loaded in the IRQ link register (r14_irq) and the Program Counter (r15) is loaded with 0x18. In the following cycle during fetch at address 0x1C, the ARM core adjusts r14_irq, decrementing it by 4.
- The ARM enters into IRQ mode.
- When the instruction loaded at address 0x18 is executed, the Program Counter is loaded with the value read in GIC_IVR.
- The previous step has effect to branch to the corresponding interrupt service routine. This should start by saving the Link Register(r14_irq) and the SPSR(SPSR_irq). Note that the Link Register must be decremented by 4 when it is saved, if it is to be restored directly into the Program Counter at the end of the interrupt. The instruction *sub pc, lr, #4* may be used, for example.
- Further interrupts can then be unmasked by clearing the I bit in the CPSR, even though it is allowed that re-assertion of the nIRQ can be occurred by another IRQ source.
- The Interrupt Handler can then proceed as required, saving the registers which will be used and restoring them at the end. Note that if the interrupt is programmed to be level sensitive, the source of the interrupt must be disappeared during this phase.
- The End Of Interrupt Command Register (GIC_EOICR) must be written in order to indicate to the GIC that the current interrupt is finished.
- The SPSR (SPSR_irq) is restored. Finally, the saved value of the Link Register is restored directly into the PC.

NOTE

The I bit in the SPSR is significant. If it is set, it indicates that the ARM processor was just about to mask IRQ interrupts when the mask instruction was interrupted. Hence, when the SPSR is restored, the mask instruction is completed (IRQ is masked).

2.3 FAST INTERRUPT SEQUENCE

For details on the registers mentioned in the steps below, refer to an ARM Embedded Core Datasheet. It is assumed that:

- The Generic Interrupt Controller has been programmed, GIC_SVR is loaded with fast interrupt service routine address and the fast interrupt is enabled.
- The Instruction at address 0x1C(FIQ exception vector address) is: `: ldr pc, [pc, #-0xF20]`.
- Nested Fast Interrupts are not needed by the user.

When nFIQ is asserted, if the bit F of CPSR is 0, the sequence is:

- The CPSR is stored in SPSR_fiq, the current value of the Program Counter is loaded in the FIQ link register (r14_fiq) and the Program Counter (r15) is loaded with 0x1C. In the following cycle, during fetch at address 0x20, the ARM adjusts r14_fiq, decrementing it by 4.
- The ARM processor enters FIQ mode.
- When the instruction loaded at address 0x1C is executed, the Program Counter is loaded with the value read in GIC_FVR. Reading the GIC_FVR has effect of automatically clearing the fast interrupt (source 0 connected to the FIQ line), if it has been programmed to be edge triggered. In this case only, it de-asserts the nFIQ line on the processor.
- The previous step has effect to branch to the corresponding interrupt service routine. It is not necessary to save the Link Register(r14_fiq) and the SPSR(SPSR_fiq) if nested fast interrupts are not needed.
- The Interrupt Handler can then proceed as required. It is not necessary to save registers r8 to r13 because FIQ mode has its own dedicated registers and the user r8 to r13 are banked. The other registers, r0 to r7, must be saved before being used, and restored at the end (before the next step). Note that if the fast interrupt is programmed to be level sensitive, the source of the interrupt must be disappeared during this phase in order to de-assert the nFIQ line.
- The End Of Interrupt Command Register (GIC_EOICR) must be written in order to indicate to the GIC that the current interrupt is finished.
- Finally, the Link Register (r14_fiq) is restored into the PC after decrementing it by 4 (with instruction `sub pc, lr, #4` for example).

NOTE

The F bit in the SPSR is significant. If it is set, it indicates that the ARM was just about to mask FIQ interrupts when the mask instruction was interrupted. Hence when the SPSR is restored, the interrupted instruction is completed (FIQ is masked).

2.4 SPURIOUS INTERRUPT SEQUENCE

A spurious interrupt can be occurred only in the interrupt which is programmed to be level sensitive. A spurious interrupt is a very short duration signal on one of the interrupt input lines. It is handled by the following sequence of actions.

- When an interrupt is active, the GIC asserts then nIRQ (or nFIQ) line and the ARM enters IRQ (or FIQ) mode. At this moment, if the interrupt source disappears, the nIRQ (or nFIQ) line is de-asserted but the ARM continues with the hardware interrupt handler.
- If the IRQ Vector Register (GIC_IVR) is read when the nIRQ is not asserted, the GIC_IVR is read with the contents of the Spurious Interrupt Vector Register.
- If the register FIQ Vector Register (GIC_FVR) is read when the nFIQ is not asserted, the GIC_FVR is read with the contents of the Spurious Interrupt Vector Register.
- The Spurious Interrupt Routine must at least write into the GIC_EOICR to perform an end of interrupt command. Until the GIC_EOICR write is received by the interrupt controller, the nIRQ (or nFIQ) line is not re-asserted.
- This causes the ARM to jump into the Spurious Interrupt Routine.
- During a Spurious Interrupt Routine, the Interrupt Status Register GIC_ISR reads 0.

3. REGISTERS DESCRIPTION

Base Address – 0xFFFF0000

Table 12-4. GIC Special Function Registers

Offset Address	Name	Description	R/W	Reset State
0xFFFFEF00	GIC_SMR0	Source mode register 0	R/W	0x00000000
0xFFFFEF04	GIC_SMR1	Source mode register 1	R/W	0x00000000
...
0xFFFFEFC	GIC_SMR63	Source mode register 63	R/W	0x00000000
0xFFFFF000	GIC_SVR0	Source vector register 0	R/W	0XXXXXXXXXX
0xFFFFF004	GIC_SVR1	Source vector register 1	R/W	0XXXXXXXXXX
...
0xFFFFF0FC	GIC_SVR63	Source vector register 63	R/W	0XXXXXXXXXX
0xFFFFF100	GIC_IVR	IRQ vector register	R	0XXXXXXXXXX
0xFFFFF104	GIC_FVR	FIQ vector register	R	0XXXXXXXXXX
0xFFFFF108	GIC_ISR	Interrupt status register	R	0x00000000
0xFFFFF10C	GIC_IPR0	Interrupt pending register 0	R	0xFFFFFFFF
0xFFFFF110	GIC_IPR1	Interrupt pending register 1	R	0xFFFFFFFF
0xFFFFF114	GIC_IMR0	Interrupt mask register 0	R	0x00000000
0xFFFFF118	GIC_IMR1	interrupt mask register 1	R	0x00000000
0xFFFFF11C	GIC_CISR	Core interrupt status register	R	0x00000000
0xFFFFF120	GIC_IECR0	Interrupt enable command register 0	W	–
0xFFFFF124	GIC_IECR1	Interrupt enable command register 1	W	–
0xFFFFF128	GIC_IDCR0	Interrupt disable command register 0	W	–
0xFFFFF12C	GIC_IDCR1	Interrupt disable command register 1	W	–
0xFFFFF130	GIC_ICCR0	Interrupt clear command register 0	W	–
0xFFFFF134	GIC_ICCR1	Interrupt clear command register 1	W	–
0xFFFFF138	GIC_ISCR0	Interrupt set command register 0	W	–
0xFFFFF13C	GIC_ISCR1	Interrupt set command register 1	W	–
0xFFFFF140	GIC_EOICR	End of interrupt command register	W	–
0xFFFFF144	GIC_SPU	Spurious vector register	R/W	0x00000000

GIC Source Mode		GIC_SMRx (0xFFFFFEXX)						Access: Read/Write
31	30	29	28	27	26	25	24	
–	–	–	–	–	–	–	–	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
23	22	21	20	19	18	17	16	
–	–	–	–	–	–	–	–	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
15	14	13	12	11	10	9	8	
–	–	–	–	–	–	–	–	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
7	6	5	4	3	2	1	0	
–	–	SRCTYP[1:0]		SDI	PRIOR[2:0]			
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- PRIOR[2:0] : Priority level**

These bits program the priority level (from 0–lowest to 7–highest) of all the interrupt sources

- SDI : Select destination interrupt nFIQ or nIRQ**

0: nIRQ line

1: nFIQ line

- SRCTYP[1:0] : Interrupt Source Type**

Table 12-5. Interrupt Source Type Field

SRCTYPE[1]	SRCTYPE[0]	Interrupt Detection
0	0	Low level sensitive
0	1	Negative edge triggered
1	0	High level sensitive
1	1	Positive edge triggered

NOTE: The type of all interrupt except external interrupt (IRQx) and software interrupt (SWIRQx) must be set to positive edge triggered. The all type of interrupt is possible for external interrupt (IRQx). The type of software interrupt (SWIRQx) is not available.

GIC Source Vector		GIC_SVRx (0xFFFF0XX)						Access: Read/Write
31	30	29	28	27	26	25	24	
VECT[31:24]								
R/W-U	R/W-U	R/W-U	R/W-U	R/W-U	R/W-U	R/W-U	R/W-U	
23	22	21	20	19	18	17	16	
VECT[23:16]								
R/W-U	R/W-U	R/W-U	R/W-U	R/W-U	R/W-U	R/W-U	R/W-U	
15	14	13	12	11	10	9	8	
VECT[15:8]								
R/W-U	R/W-U	R/W-U	R/W-U	R/W-U	R/W-U	R/W-U	R/W-U	
7	6	5	4	3	2	1	0	
VECT[7:0]								
R/W-U	R/W-U	R/W-U	R/W-U	R/W-U	R/W-U	R/W-U	R/W-U	

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **VECT[31:0] : Interrupt Handler Address**

Address of the corresponding handler for each interrupt source

GIC Interrupt Vector		GIC_IVR (0xFFFFF100)						Access: Read only
31	30	29	28	27	26	25	24	
IRQV[31:24]								
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U	
23	22	21	20	19	18	17	16	
IRQV[23:16]								
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U	
15	14	13	12	11	10	9	8	
IRQV[15:8]								
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U	
7	6	5	4	3	2	1	0	
IRQV[7:0]								
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U	

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- IRQV[31:0] : Interrupt Vector Address**

Address of the currently serviced interrupt vector (user programmed in the GIC_SVR register)

NOTE: GIC_IVR = 0x00000000 when there is no interrupt.

GIC FIQ Vector		GIC_FVR (0xFFFFF104)						Access: Read only
31	30	29	28	27	26	25	24	
FIQV[31:24]								
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U	
23	22	21	20	19	18	17	16	
FIQV[23:16]								
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U	
15	14	13	12	11	10	9	8	
FIQV[15:8]								
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U	
7	6	5	4	3	2	1	0	
FIQV[7:0]								
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U	

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- FIQV[31:0] : FIQ Vector Address**

Address of the currently serviced fast interrupt vector (user programmed in the GIC_SVR register)

GIC Interrupt Status **GIC_ISR (0xFFFFF108)** **Access: Read only**

31	30	29	28	27	26	25	24	
–	–	–	–	–	–	–	–	
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	
23	22	21	20	19	18	17	16	
–	–	–	–	–	–	–	–	
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	
15	14	13	12	11	10	9	8	
–	–	–	–	–	–	–	–	
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	
7	6	5	4	3	2	1	0	
–	–	IRQID[5:0]						–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **IRQID[5:0] : Current IRQ Identifier**
Current interrupt source number

GIC Interrupt Pending 0

GIC_IPR0 (0xFFFFF10C)

Access: Read only

31	30	29	28	27	26	25	24
GPIO2	GPIO1	GPIO0	LDMA	GPT0CH2	GPT0CH1	STT	SPI1
R-1	R-1	R-1	R-1	R-1	R-1	R-1	R-1
23	22	21	20	19	18	17	16
-	SWIRQ1	I2C1	USART0	-	CAPT0	-	-
R-1	R-1	R-1	R-1	R-1	R-1	R-1	R-1
15	14	13	12	11	10	9	8
CAN1	UART1	I2C0	UART0	ST1	ST0	GPT0CH0	CAN0
R-1	R-1	R-1	R-1	R-1	R-1	R-1	R-1
7	6	5	4	3	2	1	0
WD	SPI0	ADC0	PWM0	IFC	LVD	STOP_MODE	DFC
R-1	R-1	R-1	R-1	R-1	R-1	R-1	R-1

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

GIC Interrupt Pending 1

GIC_IPR1 (0xFFFFF110)

Access: Read only

31	30	29	28	27	26	25	24
SWIRQ6	SWIRQ5	SWIRQ4	SWIRQ3	STABLE	-	-	-
R-1	R-1	R-1	R-1	R-1	R-1	R-1	R-1
23	22	21	20	19	18	17	16
-	-	-	-	IRQ11	IRQ10	IRQ9	IRQ8
R-1	R-1	R-1	R-1	R-1	R-1	R-1	R-1
15	14	13	12	11	10	9	8
IRQ7	IRQ6	IRQ5	IRQ4	IRQ3	IRQ2	IRQ1	IRQ0
R-1	R-1	R-1	R-1	R-1	R-1	R-1	R-1
7	6	5	4	3	2	1	0
SWIRQ2	-	-	SMC3	SMC2	SMC1	SMC0	-
R-1	R-1	R-1	R-1	R-1	R-1	R-1	R-1

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **INT[63:0] : Interrupt Pending**

0: Corresponding interrupt is inactive

1: Corresponding interrupt is pending

GIC_IRP0[31:0] is matched from Interrupt source 0 to Interrupt source 31.

GIC_IRP1[31:0] is matched from Interrupt source 32 to Interrupt source 63.

For a level sensitive interrupt, if the active level occurs on the interrupt line, the interrupt is pending as long as the interrupt line is active.

For an edge sensitive interrupt, if the active edge occurs on the interrupt line, the interrupt is pending till the reading of either GIC_IVR or GIC_FVR registers, depending on the SID bit of the corresponding GIC_SMR register.

12-1. IMPORTANT NOTICE

Pending interrupts cannot be cleared by software. If a pending interrupt is masked, nIRQ or nFIQ line is not asserted, but it will be as soon as the pending interrupt is enabled. (GIC_IECR)

GIC Interrupt Mask 0

GIC_IMR0 (0xFFFFF114)

Access: Read only

31	30	29	28	27	26	25	24
GPIO2	GPIO1	GPIO0	LDMA	GPT0CH2	GPT0CH1	STT	SPI1
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	SWIRQ1	I2C1	USART0	–	CAPT0	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
CAN1	UART1	I2C0	UART0	ST1	ST0	GPT0CH0	CAN0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
WD	SPI0	ADC0	PWM0	IFC	LVD	STOP_MODE	DFC
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

GIC Interrupt Mask 1

GIC_IMR1 (0xFFFFF118)

Access: Read only

31	30	29	28	27	26	25	24
SWIRQ6	SWIRQ5	SWIRQ4	SWIRQ3	STABLE	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	IRQ11	IRQ10	IRQ9	IRQ8
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
IRQ7	IRQ6	IRQ5	IRQ4	IRQ3	IRQ2	IRQ1	IRQ0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
SWIRQ2	–	–	SMC3	SMC2	SMC1	SMC0	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- INT[63:0] : Interrupt Mask

0: Corresponding interrupt is disabled

1: Corresponding interrupt is enabled

GIC_IMR0[31:0] is matched from interrupt source 0 to interrupt source 31.

GIC_IMR1[31:0] is matched from interrupt source 32 to interrupt source 63.

GIC Core Interrupt Status				GIC_CISR (0xFFFFF11C)				Access: Read only	
31	30	29	28	27	26	25	24		
-	-	-	-	-	-	-	-		
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0		
23	22	21	20	19	18	17	16		
-	-	-	-	-	-	-	-		
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0		
15	14	13	12	11	10	9	8		
-	-	-	-	-	-	-	-		
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0		
7	6	5	4	3	2	1	0		
-	-	-	-	-	-	NIRQ	NFIQ		
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0		

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

• **NFIQ : nFIQ Status**

0: nFIQ line is inactive

1: nFIQ line is active

• **NIRQ : nIRQ Status**

0: nIRQ line is inactive

1: nIRQ line is active

GIC Interrupt Enable Command 0**GIC_IECR0 (0xFFFF120)****Access: Write only**

31	30	29	28	27	26	25	24
GPIO2	GPIO1	GPIO0	LDMA	GPT0CH2	GPT0CH1	STT	SPI1
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
23	22	21	20	19	18	17	16
–	SWIRQ1	I2C1	USART0	–	CAPT0	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8
CAN1	UART1	I2C0	UART0	ST1	ST0	GPT0CH0	CAN0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
7	6	5	4	3	2	1	0
WD	SPI0	ADC0	PWM0	IFC	LVD	STOP_MODE	DFC
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset***GIC Interrupt Enable Command 1****GIC_IECR1 (0xFFFF124)****Access: Write only**

31	30	29	28	27	26	25	24
SWIRQ6	SWIRQ5	SWIRQ4	SWIRQ3	STABLE	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
23	22	21	20	19	18	17	16
–	–	–	–	IRQ11	IRQ10	IRQ9	IRQ8
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8
IRQ7	IRQ6	IRQ5	IRQ4	IRQ3	IRQ2	IRQ1	IRQ0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
7	6	5	4	3	2	1	0
SWIRQ2	–	–	SMC3	SMC2	SMC1	SMC0	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- INT[63:0] : Interrupt Enable**

0: No effect

1: Enables the corresponding interrupt

GIC_IECR0[31:0] is matched from interrupt source 0 to interrupt source 31.

GIC_IECR1[31:0] is matched from interrupt source 32 to interrupt source 63.

GIC Interrupt Disable Command 0 **GIC_IDCR0 (0xFFFF128)** **Access: Write only**

31	30	29	28	27	26	25	24
GPIO2	GPIO1	GPIO0	LDMA	GPT0CH2	GPT0CH1	STT	SPI1
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
23	22	21	20	19	18	17	16
-	SWIRQ1	I2C1	USART0	-	CAPT0	-	-
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8
CAN1	UART1	I2C0	UART0	ST1	ST0	GPT0CH0	CAN0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
7	6	5	4	3	2	1	0
WD	SPI0	ADC0	PWM0	IFC	LVD	STOP_MODE	DFC
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

GIC Interrupt Disable Command 1 **GIC_IDCR1 (0xFFFF12C)** **Access: Write only**

31	30	29	28	27	26	25	24
SWIRQ6	SWIRQ5	SWIRQ4	SWIRQ3	STABLE	-	-	-
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
23	22	21	20	19	18	17	16
-	-	-	-	IRQ11	IRQ10	IRQ9	IRQ8
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8
IRQ7	IRQ6	IRQ5	IRQ4	IRQ3	IRQ2	IRQ1	IRQ0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
7	6	5	4	3	2	1	0
SWIRQ2	-	-	SMC3	SMC2	SMC1	SMC0	-
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

• **INT[63:0] : Interrupt Disable**

0: No effect

1: Disables the corresponding interrupt

GIC_IDCR0[31:0] is matched from interrupt source 0 to interrupt source 31.

GIC_IDCR1[31:0] is matched from interrupt source 32 to interrupt source 63.

GIC Interrupt Clear Command 0**GIC_ICCR0 (0xFFFFF130)****Access: Write only**

31	30	29	28	27	26	25	24
GPIO2	GPIO1	GPIO0	LDMA	GPT0CH2	GPT0CH1	STT	SPI1
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
23	22	21	20	19	18	17	16
-	SWIRQ1	I2C1	USART0	-	CAPT0	-	-
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8
CAN1	UART1	I2C0	UART0	ST1	ST0	GPT0CH0	CAN0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
7	6	5	4	3	2	1	0
WD	SPI0	ADC0	PWM0	IFC	LVD	STOP_MODE	DFC
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset***GIC Interrupt Clear Command 1****GIC_ICCR1 (0xFFFFF134)****Access: Write only**

31	30	29	28	27	26	25	24
SWIRQ6	SWIRQ5	SWIRQ4	SWIRQ3	STABLE	-	-	-
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
23	22	21	20	19	18	17	16
-	-	-	-	IRQ11	IRQ10	IRQ9	IRQ8
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8
IRQ7	IRQ6	IRQ5	IRQ4	IRQ3	IRQ2	IRQ1	IRQ0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
7	6	5	4	3	2	1	0
SWIRQ2	-	-	SMC3	SMC2	SMC1	SMC0	-
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- INT[63:0] : Interrupt Clear**

0: No effect

1: Clears the corresponding interrupt

GIC_ICCR0[31:0] is matched from interrupt source 0 to interrupt source 31.

GIC_ICCR1[31:0] is matched from interrupt source 32 to interrupt source 63.

GIC Interrupt Set Command 0 **GIC_ISCR0 (0xFFFF138)** **Access: Write only**

31	30	29	28	27	26	25	24
GPIO2	GPIO1	GPIO0	LDMA	GPT0CH2	GPT0CH1	STT	SPI1
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
23	22	21	20	19	18	17	16
-	SWIRQ1	I2C1	USART0	-	CAPT0	-	-
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8
CAN1	UART1	I2C0	UART0	ST1	ST0	GPT0CH0	CAN0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
7	6	5	4	3	2	1	0
WD	SPI0	ADC0	PWM0	IFC	LVD	STOP_MODE	DFC
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

GIC Interrupt Set Command 1 **GIC_ISCR1 (0xFFFF13C)** **Access: Write only**

31	30	29	28	27	26	25	24
SWIRQ6	SWIRQ5	SWIRQ4	SWIRQ3	STABLE	-	-	-
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
23	22	21	20	19	18	17	16
-	-	-	-	IRQ11	IRQ10	IRQ9	IRQ8
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8
IRQ7	IRQ6	IRQ5	IRQ4	IRQ3	IRQ2	IRQ1	IRQ0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
7	6	5	4	3	2	1	0
SWIRQ2	-	-	SMC3	SMC2	SMC1	SMC0	-
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

• **INT[63:0] : Interrupt Set**

0: No effect

1: Sets the corresponding interrupt

GIC_ISCR0[31:0] is matched from interrupt source 0 to interrupt source 31.

GIC_ISCR1[31:0] is matched from interrupt source 32 to interrupt source 63.

GIC End of Interrupt Command				GIC_EOICR (0xFFFFF140)				Access: Write only
31	30	29	28	27	26	25	24	
–	–	–	–	–	–	–	–	
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	
23	22	21	20	19	18	17	16	
–	–	–	–	–	–	–	–	
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	
15	14	13	12	11	10	9	8	
–	–	–	–	–	–	–	–	
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	
7	6	5	4	3	2	1	0	
–	–	–	–	–	–	–	–	
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

A write access to this register (with any value) indicates that the interrupt treatment is completed.

GIC Spurious Vector		GIC_SPU (0xFFFFF144)						Access: Read/Write
31	30	29	28	27	26	25	24	
SPUVECT[31:24]								
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
23	22	21	20	19	18	17	16	
SPUVECT[23:16]								
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
15	14	13	12	11	10	9	8	
SPUVECT[15:8]								
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
7	6	5	4	3	2	1	0	
SPUVECT[7:0]								
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- SPUVECT[31:0] : Spurious Interrupt Vector Handler Address**
 Address of the spurious interrupt handler

13

I/O CONFIGURATION (IOCONF)

1. OVERVIEW

1.1 FUNCTIONAL DESCRIPTION

The I/O configuration module has been designed to handle specific I/O configuration features and device special functions such as :

- Multiple digital features multiplexed with general purpose I/O on a device pin.
- Mixed analog/digital multiplexing function on a device pin
- Software controllable embedded pull-up.
- Software controllable drivability selection on output buffers.
- External wake-up interrupt selection. (MR15)
- Low voltage detection interrupt and reset enable. (MR16)

This part of the circuit datasheet describes only the I/O configuration registers. The effect of each bit of those registers is detailed in pinout description. (chapter 2)

1.2 BLOCK DIAGRAM

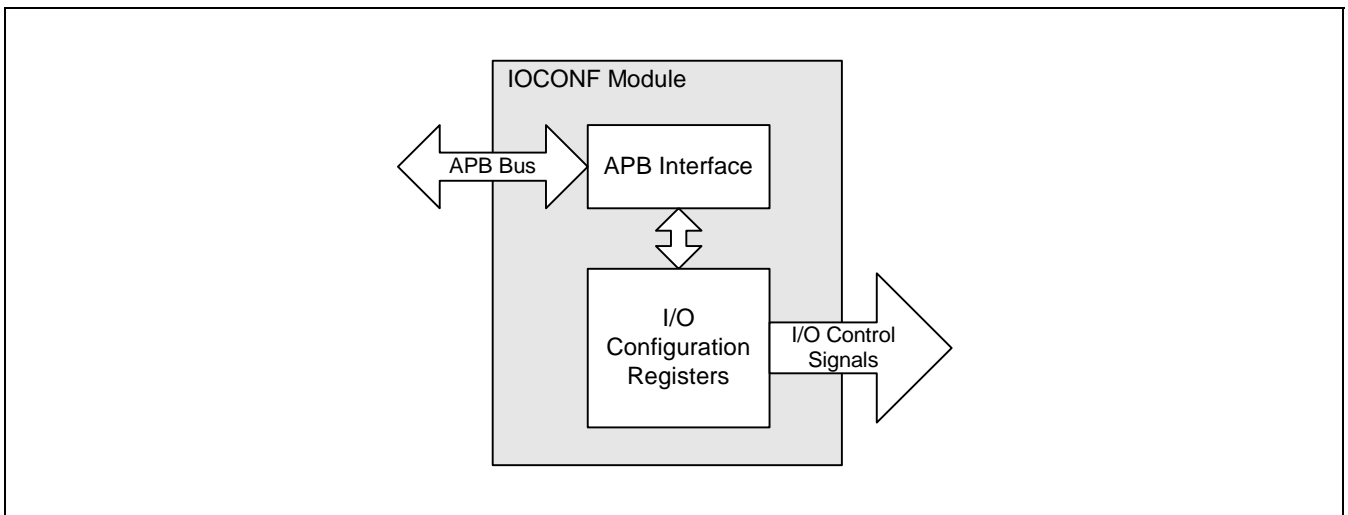


Figure 13-1. I/O Configuration Module Block Diagram

2. FUNCTIONAL OPERATION

2.1 I/O CONFIGURATION

2.1.1 General Description

The IO configuration module is simply a register bank. To have a complete pinout configuration, this module must be configured in accordance with the GPIO register. With this module, the selection between standard I/O and function is done after with the I/O configuration block.

2.1.2 Escaping From Stop Mode

The chip can be escaped from stop mode by four external interrupt. The selection of external interrupt for escaping from stop mode can be done by MR15 register.

2.2 PINOUT CONFIGURATION

Table 13-1. I/O Configuration

Pin	Function	GPIO Configuration	IOCFG Configuration	I/O Type	Activate Pull-Up
2	P0.0	PIO0.PSR[0]=1	X	I/O	IOCFG.MR0[0]=0
	NSS0	PIO0.PSR[0]=0	IOCFG.MR0[3]=0	O	
	SM00	PIO0.PSR[0]=0	IOCFG.MR0[3]=1	O	
3	P0.1	PIO0.PSR[1]=1	X	I/O	IOCFG.MR0[4]=0
	MISO0	PIO0.PSR[1]=0	IOCFG.MR0[7]=0	O	
	SM01	PIO0.PSR[1]=0	IOCFG.MR0[7]=1	I	
4	P0.2	PIO0.PSR[2]=1	IOCFG.MR0[11]=0	I/O	IOCFG.MR0[8]=0
	MOSI0	PIO0.PSR[2]=0	IOCFG.MR0[11]=0	O	
	SM02	PIO0.PSR[2]=0	IOCFG.MR0[11]=1	O	
5	P0.3	PIO0.PSR[3]=1	X	I/O	IOCFG.MR0[12]=0
	SPCK0	PIO0.PSR[3]=0	IOCFG.MR0[15]=0	O	
	SM03	PIO0.PSR[3]=0	IOCFG.MR0[15]=1	I/O	
6	P0.4	PIO0.PSR[4]=1	X	I/O	IOCFG.MR0[16]=0
	IRQ0	PIO0.PSR[4]=0	IOCFG.MR0[19]=0	O	
	SM10	PIO0.PSR[4]=0	IOCFG.MR0[19]=1	I	
7	P0.5	PIO0.PSR[5]=1	X	I/O	IOCFG.MR0[20]=0
	TCLK0	PIO0.PSR[5]=0	IOCFG.MR0[23]=0	O	
	SM11	PIO0.PSR[5]=0	IOCFG.MR0[23]=1	O	
8	P0.6	PIO0.PSR[6]=1	X	I/O	IOCFG.MR0[24]=0
	TCLK1	PIO0.PSR[6]=0	IOCFG.MR0[27]=0	O	
	SM12	PIO0.PSR[6]=0	IOCFG.MR0[27]=1	O	

Table 13-1. I/O Configuration (Continued)

Pin	Function	GPIO Configuration	IOCFG Configuration	I/O Type	Activate Pull-Up
9	P0.7	PIO0.PSR[7]=1	X	I/O	IOCFG.MR0[28]=0
	TCLK2	PIO0.PSR[7]=0	IOCFG.MR0[31]=0	O	
	SM13	PIO0.PSR[7]=0	IOCFG.MR0[31]=1	O	
10	P0.8	PIO0.PSR[8]=1	X	I/O	IOCFG.MR1[0]=0
	UARTRXD1	PIO0.PSR[8]=0	IOCFG.MR1[3]=0	O	
	SM20	PIO0.PSR[8]=0	IOCFG.MR1[3]=1	I	
11	P0.9	PIO0.PSR[9]=1	X	I/O	IOCFG.MR1[4]=0
	UARTTXD1	PIO0.PSR[9]=0	IOCFG.MR1[7]=0	O	
	SM21	PIO0.PSR[9]=0	IOCFG.MR1[7]=1	I	
12	P0.10	PIO0.PSR[10]=1	X	I/O	IOCFG.MR1[8]=0
	TIOA1	PIO0.PSR[10]=0	IOCFG.MR1[11]=0	O	
	SM22	PIO0.PSR[10]=0	IOCFG.MR1[11]=1	I/O	
13	P0.11	PIO0.PSR[11]=1	X	I/O	IOCFG.MR1[12]=0
	TIOB1	PIO0.PSR[11]=0	IOCFG.MR1[15]=0	O	
	SM23	PIO0.PSR[11]=0	IOCFG.MR1[15]=1	I/O	
14	P0.12	PIO0.PSR[12]=1	X	I/O	IOCFG.MR1[16]=0
	IRQ1	PIO0.PSR[12]=0	IOCFG.MR1[19]=0	O	
	SM30	PIO0.PSR[12]=0	IOCFG.MR1[19]=1	I/O	
15	P0.13	PIO0.PSR[13]=1	X	I/O	IOCFG.MR1[20]=0
	USARTCLK0	PIO0.PSR[13]=0	IOCFG.MR1[23]=0	O	
	SM31	PIO0.PSR[13]=0	IOCFG.MR1[23]=1	I/O	
16	P0.14	PIO0.PSR[14]=1	X	I/O	IOCFG.MR1[24]=0
	USARTRXD0	PIO0.PSR[14]=0	IOCFG.MR1[27]=0	O	
	SM32	PIO0.PSR[14]=0	IOCFG.MR1[27]=1	I	
17	P0.15	PIO0.PSR[15]=1	X	I/O	IOCFG.MR1[28]=0
	UARTTXD0	PIO0.PSR[15]=0	IOCFG.MR1[31]=0	O	
	SM33	PIO0.PSR[15]=0	IOCFG.MR1[31]=1	O	
20	P0.16	PIO0.PSR[16]=1	X	I/O	IOCFG.MR2[0]=0
	CANRXD0	PIO0.PSR[16]=0	IOCFG.MR2[3]=0	I	
	IRQ2	PIO0.PSR[16]=0	IOCFG.MR2[3]=1	I	
21	P0.17	PIO0.PSR[17]=1	X	I/O	IOCFG.MR2[4]=0
	CANTXD0	PIO0.PSR[17]=0	IOCFG.MR2[7]=0	O	
	IRQ3	PIO0.PSR[17]=0	IOCFG.MR2[7]=1	I	

Table 13-1. I/O Configuration (Continued)

Pin	Function	GPIO Configuration	IOCFG Configuration	I/O Type	Activate Pull-Up
22	P0.18	PIO0.PSR[18]=1	X	I/O	IOCFG.MR2[16]=0
	UARTRXD0	PIO0.PSR[18]=0	IOCFG.MR2[11]=0	I	
	IRQ4	PIO0.PSR[18]=0	IOCFG.MR2[11]=1	I	
23	P0.19	PIO0.PSR[19]=1	X	I/O	IOCFG.MR2[20]=0
	UARTTXD0	PIO0.PSR[19]=0	IOCFG.MR2[15]=0	O	
	IRQ5	PIO0.PSR[19]=0	IOCFG.MR2[15]=1	I	
24	P0.20	PIO0.PSR[20]=1	X	I/O	IOCFG.MR2[24]=0
	TIOA0	PIO0.PSR[20]=0	IOCFG.MR2[19]=0	I/O	
	SDA1	PIO0.PSR[20]=0	IOCFG.MR2[19]=1	I/O	
25	P0.21	PIO0.PSR[21]=1	X	I/O	IOCFG.MR2[28]=0
	TIOB0	PIO0.PSR[21]=0	IOCFG.MR2[23]=0	I/O	
	SCL1	PIO0.PSR[21]=0	IOCFG.MR2[23]=1	I/O	
34	P0.22	PIO0.PSR[22]=1	IOCFG.MR2[27]=0	I/O	N/A
	COM0	PIO0.PSR[22]=0	IOCFG.MR2[27]=1	ANALOG OUT	
35	P0.23	PIO0.PSR[23]=1	IOCFG.MR2[31]=0	I/O	N/A
	COM1	PIO0.PSR[23]=0	IOCFG.MR2[31]=1	ANALOG OUT	
36	P0.24	PIO0.PSR[24]=1	IOCFG.MR3[3]=0	I/O	N/A
	COM2	PIO0.PSR[24]=0	IOCFG.MR3[3]=1	ANALOG OUT	
37	P0.25	PIO0.PSR[25]=1	IOCFG.MR3[7]=0	I/O	N/A
	COM3	PIO0.PSR[25]=0	IOCFG.MR3[7]=1	ANALOG OUT	
38	P1.0	PIO1.PSR[0]=1	IOCFG.MR4[3]=0	I/O	N/A
	SEG0	PIO1.PSR[0]=0	IOCFG.MR4[3]=1	ANALOG OUT	
39	P1.1	PIO1.PSR[1]=1	IOCFG.MR4[7]=0	I/O	N/A
	SEG1	PIO1.PSR[1]=0	IOCFG.MR4[7]=1	ANALOG OUT	
40	P1.2	PIO1.PSR[2]=1	IOCFG.MR4[11]=0	I/O	N/A
	SEG2	PIO1.PSR[2]=0	IOCFG.MR4[11]=1	ANALOG OUT	
41	P1.3	PIO1.PSR[2]=1	IOCFG.MR4[15]=0	I/O	N/A
	SEG3	PIO1.PSR[2]=0	IOCFG.MR4[15]=1	ANALOG OUT	
42	P1.4	PIO1.PSR[4]=1	IOCFG.MR4[19]=0	I/O	N/A
	SEG4	PIO1.PSR[4]=0	IOCFG.MR4[19]=1	ANALOG OUT	
43	P1.5	PIO1.PSR[5]=1	IOCFG.MR4[23]=0	I/O	N/A
	SEG5	PIO1.PSR[5]=0	IOCFG.MR4[23]=1	ANALOG OUT	

Table 13-1. I/O Configuration (Continued)

Pin	Function	GPIO Configuration	IOCFG Configuration	I/O Type	Activate Pull-Up
44	P1.6	PIO1.PSR[6]=1	IOCFG.MR4[27]=0	I/O	N/A
	SEG6	PIO1.PSR[6]=0	IOCFG.MR4[27]=1	ANALOG OUT	
45	P1.7	PIO1.PSR[7]=1	IOCFG.MR4[31]=0	I/O	N/A
	SEG7	PIO1.PSR[7]=0	IOCFG.MR4[31]=1	ANALOG OUT	
46	P1.8	PIO1.PSR[8]=1	IOCFG.MR5[3]=0	I/O	N/A
	SEG8	PIO1.PSR[8]=0	IOCFG.MR5[3]=1	ANALOG OUT	
47	P1.9	PIO1.PSR[9]=1	IOCFG.MR5[7]=0	I/O	N/A
	SEG9	PIO1.PSR[9]=0	IOCFG.MR5[7]=1	ANALOG OUT	
48	P1.10	PIO1.PSR[10]=1	IOCFG.MR5[11]=0	I/O	N/A
	SEG10	PIO1.PSR[10]=0	IOCFG.MR5[11]=1	ANALOG OUT	
49	P1.11	PIO1.PSR[11]=1	IOCFG.MR5[15]=0	I/O	N/A
	SEG11	PIO1.PSR[11]=0	IOCFG.MR5[15]=1	ANALOG OUT	
50	P1.12	PIO1.PSR[12]=1	IOCFG.MR5[19]=0	I/O	N/A
	SEG12	PIO1.PSR[12]=0	IOCFG.MR5[19]=1	ANALOG OUT	
51	P1.13	PIO1.PSR[13]=1	IOCFG.MR5[23]=0	I/O	N/A
	SEG13	PIO1.PSR[13]=0	IOCFG.MR5[23]=1	ANALOG OUT	
52	P1.14	PIO1.PSR[14]=1	IOCFG.MR5[27]=0	I/O	N/A
	SEG14	PIO1.PSR[14]=0	IOCFG.MR5[27]=1	ANALOG OUT	
53	P1.15	PIO1.PSR[15]=1	IOCFG.MR5[31]=0	I/O	N/A
	SEG15	PIO1.PSR[15]=0	IOCFG.MR5[31]=1	ANALOG OUT	
54	P1.16	PIO1.PSR[16]=1	IOCFG.MR6[3]=0	I/O	N/A
	IRQ6	PIO1.PSR[16]=0	IOCFG.MR6[3]=0	I	
	SEG16	PIO1.PSR[16]=0	IOCFG.MR6[3]=1	ANALOG OUT	
55	P1.17	PIO1.PSR[17]=1	IOCFG.MR6[7]=0	I/O	N/A
	IRQ7	PIO1.PSR[17]=0	IOCFG.MR6[7]=0	I	
	SEG17	PIO1.PSR[17]=0	IOCFG.MR6[7]=1	ANALOG OUT	
56	P1.18	PIO1.PSR[18]=1	IOCFG.MR6[11]=0	I/O	N/A
	IRQ8	PIO0.PSR[18]=0	IOCFG.MR6[11]=0	I	
	SEG18	PIO1.PSR[18]=0	IOCFG.MR6[11]=1	ANALOG OUT	
57	P1.19	PIO1.PSR[19]=1	IOCFG.MR6[15]=0	I/O	N/A
	NPCS3	PIO1.PSR[19]=0	IOCFG.MR6[15]=0	I	
	SEG19	PIO1.PSR[19]=0	IOCFG.MR6[15]=1	ANALOG OUT	

Table 13-1. I/O Configuration (Continued)

Pin	Function	GPIO Configuration	IOCFG Configuration	I/O Type	Activate Pull-Up
58	P1.20	PIO1.PSR[20]=1	IOCFG.MR6[19]=0	I/O	N/A
	NPCS2	PIO1.PSR[20]=0	IOCFG.MR6[19]=0	I	
	SEG20	PIO1.PSR[20]=0	IOCFG.MR6[19]=1	ANALOG OUT	
59	P1.21	PIO1.PSR[21]=1	IOCFG.MR6[23]=0	I/O	N/A
	NPCS1	PIO1.PSR[21]=0	IOCFG.MR6[23]=0	I	
	SEG21	PIO1.PSR[21]=0	IOCFG.MR6[23]=1	ANALOG OUT	
60	P1.22	PIO1.PSR[22]=1	IOCFG.MR6[27]=0	I/O	N/A
	NPCS0	PIO1.PSR[22]=0	IOCFG.MR6[27]=0	I	
	SEG22	PIO1.PSR[22]=0	IOCFG.MR6[27]=1	ANALOG OUT	
61	P1.23	PIO1.PSR[23]=1	IOCFG.MR6[31]=0	I/O	N/A
	MISO1	PIO1.PSR[23]=0	IOCFG.MR6[31]=0	I	
	SEG23	PIO1.PSR[23]=0	IOCFG.MR6[31]=1	ANALOG OUT	
62	P1.24	PIO1.PSR[24]=1	IOCFG.MR7[3]=0	I/O	N/A
	MOSI1	PIO1.PSR[24]=0	IOCFG.MR7[3]=0	O	
	SEG24	PIO1.PSR[24]=0	IOCFG.MR7[3]=1	ANALOG OUT	
63	P1.25	PIO1.PSR[25]=1	IOCFG.MR7[7]=0	I/O	N/A
	SPCK1	PIO1.PSR[25]=0	IOCFG.MR7[7]=0	I/O	
	SEG25	PIO1.PSR[25]=0	IOCFG.MR7[7]=1	ANALOG OUT	
64	P1.26	PIO1.PSR[26]=1	IOCFG.MR7[11]=0	I/O	N/A
	PWM0	PIO1.PSR[26]=0	IOCFG.MR7[11]=0	O	
	SEG26	PIO1.PSR[26]=0	IOCFG.MR7[11]=1	ANALOG OUT	
65	P1.27	PIO1.PSR[27]=1	IOCFG.MR7[15]=0	I/O	N/A
	PWM1	PIO1.PSR[27]=0	IOCFG.MR7[15]=0	O	
	SEG27	PIO1.PSR[27]=0	IOCFG.MR7[15]=1	ANALOG OUT	
66	P1.28	PIO1.PSR[28]=1	IOCFG.MR7[19]=0	I/O	N/A
	CANRXD1	PIO1.PSR[28]=0	IOCFG.MR7[19]=0	I	
	SEG28	PIO1.PSR[28]=0	IOCFG.MR7[19]=1	ANALOG OUT	
67	P1.29	PIO1.PSR[29]=1	IOCFG.MR7[23]=0	I/O	N/A
	CANTXD1	PIO1.PSR[29]=0	IOCFG.MR7[23]=0	O	
	SEG29	PIO1.PSR[29]=0	IOCFG.MR7[23]=1	ANALOG OUT	
68	P1.30	PIO1.PSR[30]=1	X	I/O	IOCFG.MR7[24]= 0
	IRQ9	PIO1.PSR[30]=0	IOCFG.MR7[27]=0	I	
	SDA0	PIO1.PSR[30]=0	IOCFG.MR7[27]=1	I/O	

Table 13-1. I/O Configuration (Continued)

Pin	Function	GPIO Configuration	IOCFG Configuration	I/O Type	Activate Pull-Up
69	P1.31	PIO1.PSR[31]=1	X	I/O	IOCFG.MR7[28]=0
	ADTRG	PIO1.PSR[31]=0	IOCFG.MR7[31]=0	I	
	SCL0	PIO1.PSR[31]=0	IOCFG.MR7[31]=1	I/O	
70	P2.0	PIO2.PSR[0]=1	IOCFG.MR8[3]=0	I/O	N/A
	AIN0	PIO2.PSR[0]=0	IOCFG.MR8[3]=1	ANALOG IN	
71	P2.1	PIO2.PSR[1]=1	IOCFG.MR8[7]=0	I/O	N/A
	AIN1	PIO2.PSR[1]=0	IOCFG.MR8[7]=1	ANALOG IN	
72	P2.2	PIO2.PSR[2]=1	IOCFG.MR8[11]=0	I/O	N/A
	AIN2	PIO2.PSR[2]=0	IOCFG.MR8[11]=1	ANALOG IN	
73	P2.3	PIO2.PSR[3]=1	IOCFG.MR8[15]=0	I/O	N/A
	AIN3	PIO2.PSR[3]=0	IOCFG.MR8[15]=1	ANALOG IN	
74	P2.4	PIO2.PSR[4]=1	IOCFG.MR8[19]=0	I/O	N/A
	AIN4	PIO2.PSR[4]=0	IOCFG.MR8[19]=1	ANALOG IN	
75	P2.5	PIO2.PSR[5]=1	IOCFG.MR8[23]=0	I/O	N/A
	AIN5	PIO2.PSR[5]=0	IOCFG.MR8[23]=1	ANALOG IN	
76	P2.6	PIO2.PSR[6]=1	IOCFG.MR8[27]=0	I/O	N/A
	AIN6	PIO2.PSR[6]=0	IOCFG.MR8[27]=1	ANALOG IN	
77	P2.7	PIO2.PSR[7]=1	IOCFG.MR8[31]=0	I/O	N/A
	AIN7	PIO2.PSR[7]=0	IOCFG.MR8[31]=1	ANALOG IN	
78	P2.8	PIO2.PSR[8]=1	IOCFG.MR9[3]=0	I/O	N/A
	AIN8	PIO2.PSR[8]=0	IOCFG.MR9[3]=1	ANALOG IN	
79	P2.9	PIO2.PSR[9]=1	IOCFG.MR9[7]=0	I/O	N/A
	AIN9	PIO2.PSR[9]=0	IOCFG.MR9[7]=1	ANALOG IN	
80	P2.10	PIO2.PSR[10]=1	IOCFG.MR9[11]=0	I/O	N/A
	AIN10	PIO2.PSR[10]=0	IOCFG.MR9[11]=1	ANALOG IN	
81	P2.11	PIO2.PSR[11]=1	IOCFG.MR9[15]=0	I/O	N/A
	IRQ10	PIO2.PSR[11]=0	IOCFG.MR9[15]=0	I	
	AIN11	PIO2.PSR[11]=1	IOCFG.MR9[15]=1	ANALOG IN	
82	P2.12	PIO2.PSR[12]=1	IOCFG.MR9[19]=0	I/O	N/A
	IRQ11	PIO2.PSR[12]=0	IOCFG.MR9[19]=0	I	
	AIN12	PIO2.PSR[12]=0	IOCFG.MR9[19]=1	ANALOG IN	

Table 13-1. I/O Configuration (Continued)

Pin	Function	GPIO Configuration	IOCFG Configuration	I/O Type	Activate Pull-Up
83	P2.13	PIO2.PSR[13]=1	IOCFG.MR9[23]=0	I/O	N/A
	CAPT0	PIO2.PSR[13]=0	IOCFG.MR9[23]=0	I	
	AIN13	PIO2.PSR[13]=0	IOCFG.MR9[23]=1	ANALOG IN	
84	P2.14	PIO2.PSR[14]=1	IOCFG.MR9[27]=0	I/O	N/A
	TIOA2	PIO2.PSR[14]=0	IOCFG.MR9[27]=0	I/O	
	AIN14	PIO2.PSR[14]=0	IOCFG.MR9[27]=1	ANALOG IN	
85	P2.15	PIO2.PSR[15]=1	IOCFG.MR9[31]=0	I/O	N/A
	TIOB2	PIO2.PSR[15]=0	IOCFG.MR9[31]=0	I/O	
	AIN15	PIO2.PSR[15]=0	IOCFG.MR9[31]=1	ANALOG IN	

NOTE: 'X' means 'don't care'.

3. REGISTERS DESCRIPTION

Base Address – 0xFFE2C000

Table 13-2. IOCONF Special Function Registers

Offset Address	Name	Description	R/W	Reset State
0x00000000	MR0	IO configuration mode register 0	R/W	0x11111111
0x00000004	MR1	IO configuration mode register 1	R/W	0x11111111
0x00000008	MR2	IO configuration mode register 2	R/W	0x00111100
0x0000000C	MR3	IO configuration mode register 3	R/W	0x00000000
0x00000010	MR4	IO configuration mode register 4	R/W	0x00000000
0x00000014	MR5	IO configuration mode register 5	R/W	0x00000000
0x00000018	MR6	IO configuration mode register 6	R/W	0x00000000
0x0000001C	MR7	IO configuration mode register 7	R/W	0x11000000
0x00000020	MR8	IO configuration mode register 8	R/W	0x00000000
0x00000024	MR9	IO configuration mode register 9	R/W	0x00000000
0x00000028	–	–	–	–
0x0000002C	–	–	–	–
0x00000030	–	–	–	–
0x00000034	–	–	–	–
0x00000038	–	–	–	–
0x0000003C	MR15	Wake-up interrupt selection register	R/W	0x00000000
0x00000040	MR16	LVD reset and interrupt enable register	R/W	0x00000003

IO Configuration Mode Register 0

MR0 (0x000)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	PIO0_7_PUENB	–	–	–	PIO0_6_PUENB
R-0	R-0	R-0	R/W-1	R-0	R-0	R-0	R/W-1
23	22	21	20	19	18	17	16
–	–	–	PIO0_5_PUENB	–	–	–	PIO0_4_PUENB
R-0	R-0	R-0	R/W-1	R-0	R-0	R-0	R/W-1
15	14	13	12	11	10	9	8
PIO0_3_F3EN	–	–	PIO0_3_PUENB	PIO0_2_F3EN	–	–	PIO0_2_PUENB
R/W-0	R-0	R-0	R/W-0	R/W-0	R-0	R-0	R/W-0
7	6	5	4	3	2	1	0
–	–	–	PIO0_1_PUENB	–	–	–	PIO0_0_PUENB
R-0	R-0	R-0	R/W-0	R-0	R-0	R-0	R/W-0

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **PIO0_0_PUENB**

0: Pull-up enable on pin PIO0_0

1: Pull-up disable on pin PIO0_0

- **PIO0_1_PUENB**

0: Pull-up enable on pin PIO0_1

1: Pull-up disable on pin PIO0_1

- **PIO0_2_PUENB**

0: Pull-up enable on pin PIO0_2

1: Pull-up disable on pin PIO0_2

- **PIO0_2_F3EN**

0: Third function disable on pin PIO0_2 (CANRXD1)

1: Third function enable on pin PIO0_2 (IRQ0)

- **PIO0_3_PUENB**

0: Pull-up enable on pin PIO0_3

1: Pull-up disable on pin PIO0_3

- **PIO0_3_F3EN**

0: Third function disable on pin PIO0_3 (CANTXD1)

1: Third function enable on pin PIO0_3 (IRQ1)

- **PIO0_4_PUENB**

0: Pull-up enable on pin PIO0_4

1: Pull-up disable on pin PIO0_4

- **PIO0_5_PUENB**

0: Pull-up enable on pin PIO0_5

1: Pull-up disable on pin PIO0_5

- **PIO0_6_PUENB**

0: Pull-up enable on pin PIO0_6

1: Pull-up disable on pin PIO0_6

- **PIO0_7_PUENB**

0: Pull-up enable on pin PIO0_7

1: Pull-up disable on pin PIO0_7

IO Configuration Mode Register 1 MR1 (0x004) Access: Read/Write

31	30	29	28	27	26	25	24
PIO0_15_F3EN	–	–	PIO0_15_PUENB	PIO0_14_F3EN	–	–	PIO0_14_PUENB
R/W-0	R-0	R-0	R/W-1	R/W-0	R-0	R-0	R/W-1
23	22	21	20	19	18	17	16
PIO0_13_F3EN	–	–	PIO0_13_PUENB	PIO0_12_F3EN	–	–	PIO0_12_PUENB
R/W-0	R-0	R-0	R/W-1	R/W-0	R-0	R-0	R/W-1
15	14	13	12	11	10	9	8
PIO0_11_F3EN	–	–	PIO0_11_PUENB	PIO0_10_F3EN	–	–	PIO0_10_PUENB
R/W-0	R-0	R-0	R/W-1	R/W-0	R-0	R-0	R/W-1
7	6	5	4	3	2	1	0
PIO0_9_F3EN	–	–	PIO0_9_PUENB	PIO0_8_F3EN	–	–	PIO0_8_PUENB
R/W-0	R-0	R-0	R/W-1	R/W-0	R-0	R-0	R/W-1

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **PIO0_8_PUENB**

- 0: Pull-up enable on pin PIO0_8
- 1: Pull-up disable on pin PIO0_8

- **PIO0_8_F3EN**

- 0: Third function disable on pin PIO0_8 (NPCS0)
- 1: Third function enable on pin PIO0_8 (SCL0)

- **PIO0_9_PUENB**

- 0: Pull-up enable on pin PIO0_9
- 1: Pull-up disable on pin PIO0_9

- **PIO0_9_F3EN**

- 0: Third function disable on pin PIO0_9 (NPCS1)
- 1: Third function enable on pin PIO0_9 (SDA0)

- **PIO0_10_PUENB**

- 0: Pull-up enable on pin PIO0_10
- 1: Pull-up disable on pin PIO0_10

- **PIO0_10_F3EN**

- 0: Third function disable on pin PIO0_10 (NPCS2)
- 1: Third function enable on pin PIO0_10 (SCL1)

- **PIO0_11_PUENB**

- 0: Pull-up enable on pin PIO0_11
- 1: Pull-up disable on pin PIO0_11

- **PIO0_11_F3EN**

- 0: Third function disable on pin PIO0_11 (NPCS3)
- 1: Third function enable on pin PIO0_11 (SDA1)

- **PIO0_12_PUENB**

- 0: Pull-up enable on pin PIO0_12
- 1: Pull-up disable on pin PIO0_12

- **PIO0_12_F3EN**

- 0: Third function disable on pin PIO0_12 (MISO1)
- 1: Third function enable on pin PIO0_12 (IRQ11)

- **PIO0_13_PUENB**

- 0: Pull-up enable on pin PIO0_13
- 1: Pull-up disable on pin PIO0_13

- **PIO0_13_F3EN**

- 0: Third function disable on pin PIO0_13 (MOSI1)
- 1: Third function enable on pin PIO0_13 (IRQ12)

- **PIO0_14_PUENB**

- 0: Pull-up enable on pin PIO0_14
- 1: Pull-up disable on pin PIO0_14

- **PIO0_14_F3EN**

- 0: Third function disable on pin PIO0_14 (SPCK1)
- 1: Third function enable on pin PIO0_14 (IRQ13)

- **PIO0_15_PUENB**

- 0: Pull-up enable on pin PIO0_15
- 1: Pull-up disable on pin PIO0_15

- **PIO0_15_F3EN**

- 0: Third function disable on pin PIO0_15 (NSS0)
- 1: Third function enable on pin PIO0_15 (IRQ14)

IO Configuration Mode Register 2				MR2 (0x008)				Access: Read/Write
31	30	29	28	27	26	25	24	
–	–	–	–	PIO0_22_F3EN	–	–	PIO0_22_PUENB	
R-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R/W-1	
23	22	21	20	19	18	17	16	
PIO0_21_F3EN	–	–	PIO0_21_PUENB	PIO0_20_F3EN	–	–	PIO0_20_PUENB	
R/W-0	R-0	R-0	R/W-1	R/W-0	R-0	R-0	R/W-1	
15	14	13	12	11	10	9	8	
PIO0_19_F3EN	–	–	PIO0_19_PUENB	PIO0_18_F3EN	–	–	PIO0_18_PUENB	
R/W-0	R-0	R-0	R/W-1	R/W-0	R-0	R-0	R/W-1	
7	6	5	4	3	2	1	0	
PIO0_17_F3EN	–	–	PIO0_17_PUENB	PIO0_16_F3EN	–	–	PIO0_16_PUENB	
R/W-0	R-0	R-0	R/W-1	R/W-0	R-0	R-0	R/W-1	

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **PIO0_16_PUENB**

0: Pull-up enable on pin PIO0_16

1: Pull-up disable on pin PIO0_16

- **PIO0_16_F3EN**

0: Third function disable on pin PIO0_16 (MISO0)

1: Third function enable on pin PIO0_16 (IRQ15)

- **PIO0_17_PUENB**

0: Pull-up enable on pin PIO0_17

1: Pull-up disable on pin PIO0_17

- **PIO0_17_F3EN**

0: Third function disable on pin PIO0_17 (MOSI0)

1: Third function enable on pin PIO0_17 (IRQ16)

- **PIO0_18_PUENB**

0: Pull-up enable on pin PIO0_18

1: Pull-up disable on pin PIO0_18

- **PIO0_18_F3EN**

0: Third function disable on pin PIO0_18 (SPCK0)

1: Third function enable on pin PIO0_18 (IRQ17)

- **PIO0_19_PUENB**

0: Pull-up enable on pin PIO0_19

1: Pull-up disable on pin PIO0_19

- **PIO0_19_F3EN**

0: Third function disable on pin PIO0_19 (SM40)

1: Third function enable on pin PIO0_19 (USARTRXD0)

- **PIO0_20_PUENB**

0: Pull-up enable on pin PIO0_20

1: Pull-up disable on pin PIO0_20

- **PIO0_20_F3EN**

0: Third function disable on pin PIO0_20 (SM41)

1: Third function enable on pin PIO0_20 (USARTTXD0)

- **PIO0_21_PUENB**

0: Pull-up enable on pin PIO0_21

1: Pull-up disable on pin PIO0_21

- **PIO0_21_F3EN**

0: Third function disable on pin PIO0_21 (SM42)

1: Third function enable on pin PIO0_21 (USARTCLK0)

- **PIO0_22_PUENB**

0: Pull-up enable on pin PIO0_22

1: Pull-up disable on pin PIO0_22

- **PIO0_22_F3EN**

0: Third function disable on pin PIO0_22 (SM43)

1: Third function enable on pin PIO0_22 (XCS1)

IO Configuration Mode Register 3			MR3 (0x00C)				Access: Read/Write	
31	30	29	28	27	26	25	24	
–	–	–	PIO1_7_PUENB	–	–	–	PIO1_6_PUENB	
R-0	R-0	R-0	R/W-1	R-0	R-0	R-0	R/W-1	
23	22	21	20	19	18	17	16	
PIO1_5_F3EN	–	–	PIO1_5_PUENB	PIO1_4_F3EN	–	–	PIO1_4_PUENB	
R/W-0	R-0	R-0	R/W-1	R/W-0	R-0	R-0	R/W-1	
15	14	13	12	11	10	9	8	
PIO1_3_F3EN	–	–	PIO1_3_PUENB	PIO1_2_F3EN	–	–	PIO1_2_PUENB	
R/W-0	R-0	R-0	R/W-0	R/W-0	R-0	R-0	R/W-0	
7	6	5	4	3	2	1	0	
PIO1_1_F3EN	–	–	PIO1_1_PUENB	PIO1_0_F3EN	–	–	PIO1_0_PUENB	
R/W-0	R-0	R-0	R/W-1	R/W-0	R-0	R-0	R/W-1	

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **PIO1_0_PUENB**

0: Pull-up enable on pin PIO1_0

1: Pull-up disable on pin PIO1_0

- **PIO1_0_F3EN**

0: Third function disable on pin PIO1_0 (SM50)

1: Third function enable on pin PIO1_0 (IRQ9)

- **PIO1_1_PUENB**

0: Pull-up enable on pin PIO1_1

1: Pull-up disable on pin PIO1_1

- **PIO1_1_F3EN**

0: Third function disable on pin PIO1_1 (SM51)

1: Third function enable on pin PIO1_1 (IRQ10)

- **PIO1_2_PUENB**

0: Pull-up enable on pin PIO1_2

1: Pull-up disable on pin PIO1_2

- **PIO1_2_F3EN**

- 0: Third function disable on pin PIO1_2 (SM52)
- 1: Third function enable on pin PIO1_2 (CANRXD3)

- **PIO1_3_PUENB**

- 0: Pull-up enable on pin PIO1_3
- 1: Pull-up disable on pin PIO1_3

- **PIO1_3_F3EN**

- 0: Third function disable on pin PIO1_3 (SM53)
- 1: Third function enable on pin PIO1_3 (CANTXD3)

- **PIO1_4_PUENB**

- 0: Pull-up enable on pin PIO1_4
- 1: Pull-up disable on pin PIO1_4

- **PIO1_4_F3EN**

- 0: Third function disable on pin PIO1_4 (PWM0)
- 1: Third function enable on pin PIO1_4 (IRQ2)

- **PIO1_5_PUENB**

- 0: Pull-up enable on pin PIO1_5
- 1: Pull-up disable on pin PIO1_5

- **PIO1_5_F3EN**

- 0: Third function disable on pin PIO1_5 (PWM1)
- 1: Third function enable on pin PIO1_5 (IRQ3)

- **PIO1_6_PUENB**

- 0: Pull-up enable on pin PIO1_6
- 1: Pull-up disable on pin PIO1_6

- **PIO1_7_PUENB**

- 0: Pull-up enable on pin PIO1_7
- 1: Pull-up disable on pin PIO1_7

IO Configuration Mode Register 4				MR4 (0x010)				Access: Read/Write
31	30	29	28	27	26	25	24	
–	–	–	–	–	–	–	–	
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	
23	22	21	20	19	18	17	16	
–	–	–	–	–	–	–	–	
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	
15	14	13	12	11	10	9	8	
PIO1_11_F3EN	–	–	PIO1_11_PUENB	PIO1_10_F3EN	–	–	PIO1_10_PUENB	
R/W-0	R-0	R-0	R/W-1	R/W-0	R-0	R-0	R/W-1	
7	6	5	4	3	2	1	0	
–	–	–	PIO1_9_PUENB	–	–	–	PIO1_8_PUENB	
R-0	R-0	R-0	R/W-1	R-0	R-0	R-0	R/W-1	

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **PIO1_8_PUENB**

0: Pull-up enable on pin PIO1_8

1: Pull-up disable on pin PIO1_8

- **PIO1_9_PUENB**

0: Pull-up enable on pin PIO1_9

1: Pull-up disable on pin PIO1_9

- **PIO1_10_PUENB**

0: Pull-up enable on pin PIO1_10

1: Pull-up disable on pin PIO1_10

- **PIO1_10_F3EN**

0: Third function disable on pin PIO1_10 (TIOA2)

1: Third function enable on pin PIO1_10 (IRQ18)

- **PIO1_11_PUENB**

0: Pull-up enable on pin PIO1_11

1: Pull-up disable on pin PIO1_11

- **PIO1_11_F3EN**

0: Third function disable on pin PIO1_11 (TIOB2)

1: Third function enable on pin PIO1_11 (IRQ19)

IO Configuration Mode Register 5				MR5 (0x014)				Access: Read/Write							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIO1_19_F3EN	–	–	–	PIO1_18_F3EN	–	–	–	PIO1_17_F3EN	–	–	–	PIO1_16_F3EN	–	–	–
R/W-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIO1_15_F3EN	–	–	–	PIO1_14_F3EN	–	–	–	PIO1_13_F3EN	–	–	–	PIO1_12_F3EN	–	–	–
R/W-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0
R/W-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **PIO1_12_F3EN**

- 0: Third function disable on pin PIO1_12
- 1: Third function enable on pin PIO1_12 (COM0)

- **PIO1_13_F3EN**

- 0: Third function disable on pin PIO1_13 (A23)
- 1: Third function enable on pin PIO1_13 (COM1)

- **PIO1_14_F3EN**

- 0: Third function disable on pin PIO1_14 (NWAIT)
- 1: Third function enable on pin PIO1_14 (COM2)

- **PIO1_15_F3EN**

- 0: Third function disable on pin PIO1_15 (EPCCLK)
- 1: Third function enable on pin PIO1_15 (COM3)

- **PIO1_16_F3EN**

- 0: Third function disable on pin PIO1_16 (A22)
- 1: Third function enable on pin PIO1_16 (SEG0)

- **PIO1_17_F3EN**

0: Third function disable on pin PIO1_17 (A21)

1: Third function enable on pin PIO1_17 (SEG1)

- **PIO1_18_F3EN**

0: Third function disable on pin PIO1_18 (A20)

1: Third function enable on pin PIO1_18 (SEG2)

- **PIO1_19_F3EN**

0: Third function disable on pin PIO1_19 (A19)

1: Third function enable on pin PIO1_19 (SEG3)

IO Configuration Mode Register 6 **MR6 (0x018)** **Access: Read/Write**

31	30	29	28	27	26	25	24
PIO1_27_F3EN	–	–	–	PIO1_26_F3EN	–	–	–
R/W-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
PIO1_25_F3EN	–	–	–	PIO1_24_F3EN	–	–	–
R/W-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
PIO1_23_F3EN	–	–	–	PIO1_22_F3EN	–	–	–
R/W-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
PIO1_21_F3EN	–	–	–	PIO1_20_F3EN	–	–	–
R/W-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **PIO1_20_F3EN**

- 0: Third function disable on pin PIO1_20 (A18)
- 1: Third function enable on pin PIO1_20 (SEG4)

- **PIO1_21_F3EN**

- 0: Third function disable on pin PIO1_21 (A17)
- 1: Third function enable on pin PIO1_21 (SEG5)

- **PIO1_22_F3EN**

- 0: Third function disable on pin PIO1_22 (A16)
- 1: Third function enable on pin PIO1_22 (SEG6)

- **PIO1_23_F3EN**

- 0: Third function disable on pin PIO1_23 (A15)
- 1: Third function enable on pin PIO1_23 (SEG7)

- **PIO1_24_F3EN**

- 0: Third function disable on pin PIO1_24 (A14)
- 1: Third function enable on pin PIO1_24 (SEG8)

- **PIO1_25_F3EN**

0: Third function disable on pin PIO1_25 (A13)

1: Third function enable on pin PIO1_25 (SEG9)

- **PIO1_26_F3EN**

0: Third function disable on pin PIO1_26 (A12)

1: Third function enable on pin PIO1_26 (SEG10)

- **PIO1_27_F3EN**

0: Third function disable on pin PIO1_27 (A11)

1: Third function enable on pin PIO1_27 (SEG11)

IO Configuration Mode Register 7				MR7 (0x01C)				Access: Read/Write
31	30	29	28	27	26	25	24	
–	–	–	–	–	–	–	–	
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	
23	22	21	20	19	18	17	16	
–	–	–	–	–	–	–	–	
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	
15	14	13	12	11	10	9	8	
PIO1_31_F3EN	–	–	–	PIO1_30_F3EN	–	–	–	
R/W-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0	
7	6	5	4	3	2	1	0	
PIO1_29_F3EN	–	–	–	PIO1_28_F3EN	–	–	–	
R/W-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0	

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **PIO1_28_F3EN**

- 0: Third function disable on pin PIO1_28 (A10)
- 1: Third function enable on pin PIO1_28 (SEG12)

- **PIO1_29_F3EN**

- 0: Third function disable on pin PIO1_29 (A9)
- 1: Third function enable on pin PIO1_29 (SEG13)

- **PIO1_30_F3EN**

- 0: Third function disable on pin PIO1_30 (A8)
- 1: Third function enable on pin PIO1_30 (SEG14)

- **PIO1_31_F3EN**

- 0: Third function disable on pin PIO1_31 (A7)
- 1: Third function enable on pin PIO1_31 (SEG15)

IO Configuration Mode Register 8				MR8 (0x020)				Access: Read/Write
31	30	29	28	27	26	25	24	
PIO2_7_F3EN	–	–	–	PIO2_6_F3EN	–	–	–	
R/W-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0	
23	22	21	20	19	18	17	16	
PIO2_5_F3EN	–	–	–	PIO2_4_F3EN	–	–	–	
R/W-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0	
15	14	13	12	11	10	9	8	
PIO2_3_F3EN	–	–	–	PIO2_2_F3EN	–	–	–	
R/W-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0	
7	6	5	4	3	2	1	0	
PIO2_1_F3EN	–	–	–	PIO2_0_F3EN	–	–	–	
R/W-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0	

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **PIO2_0_F3EN**

0: Third function disable on pin PIO2_0 (A6)

1: Third function enable on pin PIO2_0 (SEG16)

- **PIO2_1_F3EN**

0: Third function disable on pin PIO2_1 (A5)

1: Third function enable on pin PIO2_1 (SEG17)

- **PIO2_2_F3EN**

0: Third function disable on pin PIO2_2 (A4)

1: Third function enable on pin PIO2_2 (SEG18)

- **PIO2_3_F3EN**

0: Third function disable on pin PIO2_3 (A3)

1: Third function enable on pin PIO2_3 (SEG19)

- **PIO2_4_F3EN**

0: Third function disable on pin PIO2_4 (A2)

1: Third function enable on pin PIO2_4 (SEG20)

- **PIO2_5_F3EN**

0: Third function disable on pin PIO2_5 (A1)

1: Third function enable on pin PIO2_5 (SEG21)

- **PIO2_6_F3EN**

0: Third function disable on pin PIO2_6 (A0)

1: Third function enable on pin PIO2_6 (SEG22)

- **PIO2_7_F3EN**

0: Third function disable on pin PIO2_7 (CTRL2)

1: Third function enable on pin PIO2_7 (SEG23)

IO Configuration Mode Register 9				MR9 (0x024)				Access: Read/Write
31	30	29	28	27	26	25	24	
PIO2_15_F3EN	–	–	–	PIO2_14_F3EN	–	–	–	
R/W-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0	
23	22	21	20	19	18	17	16	
PIO2_13_F3EN	–	–	–	PIO2_12_F3EN	–	–	–	
R/W-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0	
15	14	13	12	11	10	9	8	
PIO2_11_F3EN	–	–	–	PIO2_10_F3EN	–	–	–	
R/W-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0	
7	6	5	4	3	2	1	0	
PIO2_9_F3EN	–	–	–	PIO2_8_F3EN	–	–	–	
R/W-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0	

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **PIO2_8_F3EN**

0: Third function disable on pin PIO2_8 (CTRL1)

1: Third function enable on pin PIO2_8 (SEG24)

- **PIO2_9_F3EN**

0: Third function disable on pin PIO2_9 (CTRL0)

1: Third function enable on pin PIO2_9 (SEG25)

- **PIO2_10_F3EN**

0: Third function disable on pin PIO2_10 (XCS0)

1: Third function enable on pin PIO2_10 (SEG26)

- **PIO2_11_F3EN**

0: Third function disable on pin PIO2_11 (D7)

1: Third function enable on pin PIO2_11 (SEG27)

- **PIO2_12_F3EN**

0: Third function disable on pin PIO2_12 (D6)

1: Third function enable on pin PIO2_12 (SEG28)

- **PIO2_13_F3EN**

0: Third function disable on pin PIO2_13 (D5)

1: Third function enable on pin PIO2_13 (SEG29)

- **PIO2_14_F3EN**

0: Third function disable on pin PIO2_14 (D4)

1: Third function enable on pin PIO2_14 (SEG30)

- **PIO2_15_F3EN**

0: Third function disable on pin PIO2_15 (D3)

1: Third function enable on pin PIO2_15 (SEG31)

IO Configuration Mode Register 10				MR10 (0x028)				Access: Read/Write							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIO2_23_F3EN	–	–	–	PIO2_22_F3EN	–	–	–	PIO2_21_F3EN	–	–	–	PIO2_20_F3EN	–	–	–
R/W-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PIO2_19_F3EN	–	–	–	PIO2_18_F3EN	–	–	–	PIO2_17_F3EN	–	–	–	PIO2_16_F3EN	–	–	–
R/W-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0
R/W-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0	R/W-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **PIO2_16_F3EN**

- 0: Third function disable on pin PIO2_16 (D2)
- 1: Third function enable on pin PIO2_16 (SEG32)

- **PIO2_17_F3EN**

- 0: Third function disable on pin PIO2_17 (D1)
- 1: Third function enable on pin PIO2_17 (SEG33)

- **PIO2_18_F3EN**

- 0: Third function disable on pin PIO2_18 (D0)
- 1: Third function enable on pin PIO2_18 (SEG34)

- **PIO2_19_F3EN**

- 0: Third function disable on pin PIO2_19 (CAPT1)
- 1: Third function enable on pin PIO2_19 (SEG35)

- **PIO2_20_F3EN**

- 0: Third function disable on pin PIO2_20 (CAPT0)
- 1: Third function enable on pin PIO2_20 (SEG36)

- **PIO2_21_F3EN**

- 0: Third function disable on pin PIO2_21 (CANRXD2)
- 1: Third function enable on pin PIO2_21 (SEG37)

- **PIO2_22_F3EN**

0: Third function disable on pin PIO2_22 (CANTXD2)

1: Third function enable on pin PIO2_22 (SEG38)

- **PIO2_23_F3EN**

0: Third function disable on pin PIO2_23 (ADTRG)

1: Third function enable on pin PIO2_23 (SEG39)

Wake-Up Interrupt Selection Register			MR15 (0x03C)				Access: Read/Write	
31	30	29	28	27	26	25	24	
–	–	–	STOP_WU3[4:0]					
R-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
23	22	21	20	19	18	17	16	
–	–	–	STOP_WU2[4:0]					
R-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
15	14	13	12	11	10	9	8	
–	–	–	STOP_WU1[4:0]					
R-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
7	6	5	4	3	2	1	0	
–	–	–	STOP_WU0[4:0]					
R-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

4 wake up interrupt sources are available. They are used in the clock manager to switch from the STOP mode back to the NORMAL mode.

- **STOP_WU0[4:0] : External wake up interrupt 0 selection**

This field is used to select one of the 20 available external interrupt sources to be used as wake up interrupt 0. 0x1 to 0x14 : The corresponding external interrupt source is used as wake up interrupt 0. 0x1 is IRQ0 and 0x14 is IRQ19.

others : No external interrupt is selected.

- **STOP_WU1[4:0] : External wake up interrupt 1 selection**

This field is used to select one of the 20 available external interrupt sources to be used as wake up interrupt 1. 0x1 to 0x14 : The corresponding external interrupt source is used as wake up interrupt 1. 0x1 is IRQ0 and 0x14 is IRQ19.

others : No external interrupt is selected.

- **STOP_WU2[4:0] : External wake up interrupt 2 selection**

This field is used to select one of the 20 available external interrupt sources to be used as wake up interrupt 2. 0x1 to 0x14 : The corresponding external interrupt source is used as wake up interrupt 2. 0x1 is IRQ0 and 0x14 is IRQ19.

others : No external interrupt is selected.

- **STOP_WU3[4:0] : External wake up interrupt 3 selection**

This field is used to select one of the 20 available external interrupt sources to be used as wake up interrupt 3. 0x1 to 0x14 : The corresponding external interrupt source is used as wake up interrupt 3. 0x1 is IRQ0 and 0x14 is IRQ19.

others : No external interrupt is selected.

NOTE: The name of external wake up interrupt is STOP_MODE(See P12-2).

LVD Reset and Interrupt Enable Register

MR16 (0x040)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	LVD_INT_EN	LVD_RST_EN
R-0	R-0	R-0	R-0	R-0	R-0	R/W-1	R/W-1

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **LVD_RST_EN**

0: Low Voltage Detector reset is disabled

1: Low Voltage Detector reset is enabled

- **LVD_INT_EN**

0: Low Voltage Detector interrupt is disabled

1: Low Voltage Detector interrupt is enabled

NOTES:

1. The name of LVD interrupt is LVD(See P12-2).
2. The reset functionality of LVD must be turned off (MR16.0 == 0) before entering stop mode.

14

I2C

1. OVERVIEW

1.1 FUNCTIONAL DESCRIPTION

The I2C (Inter-Integrated Circuit) bus is a two-wire synchronous serial interface consisting of one data (SDA) and one clock (SCL). Each device connected to the bus is software addressable by a unique address and simple relationships exist at all times. The lines SDA and SCL are bi-directional lines connected to a positive supply voltage via a pull-up resistor. The output stages of devices connected to the bus must have an open-drain in order to perform the wired-AND function.

It is a true multi-master bus including collision detection and arbitration to prevent data corruption if two or more masters simultaneously initiate data transfer. Clock synchronization is performed using the wired-AND connection of I2C interfaces to the SCL lines.

The I2C interface can operate in fast mode or in normal mode. This allows baud rates from 0 to 400 Kbit/s (fast mode) or from 0 to 100 Kbit/s (normal mode). The device supports four modes : Master Transmitter, Master Receiver, Slave Transmitter, Slave Receiver. The device allows 7-bits addressing or 10-bits addressing and detection of its own address and General Call address (slave mode).

1.2 BLOCK DIAGRAM

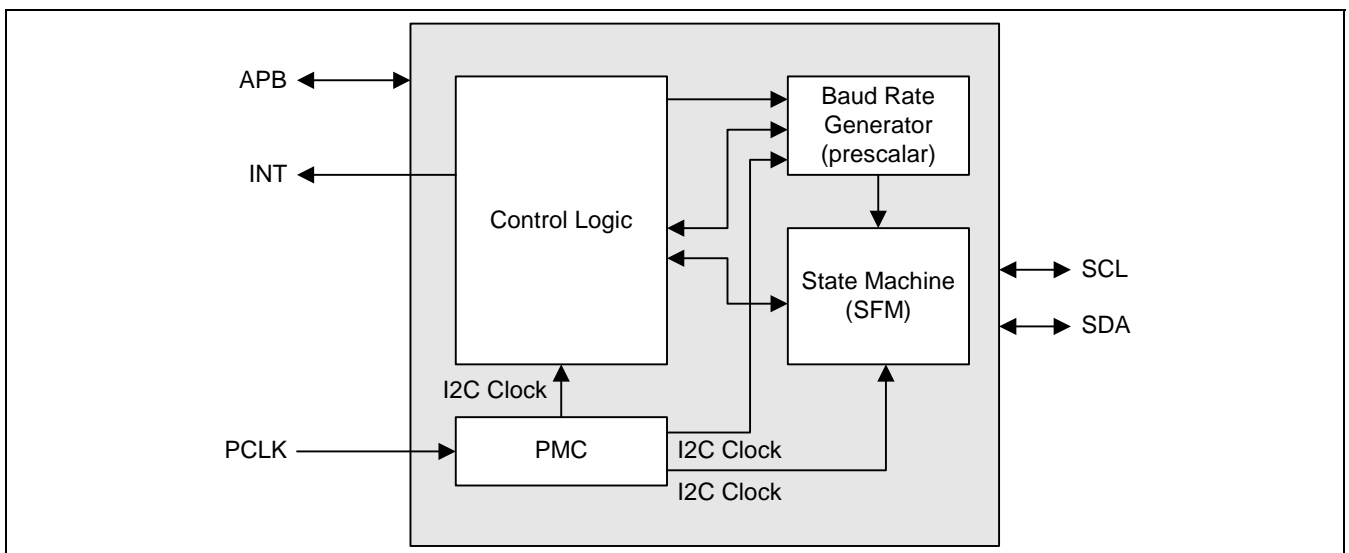


Figure 14-1. I2C Block Diagram

2. EXTERNAL PIN DESCRIPTION

Table 14-1. I2C Pin Description

Pin Name	Function	I/O Type	Active Level	Comment
SDA[1:0]	Data line	I/O	H	–
SCL[1:0]	Clock line	I/O	H	–

3. FUNCTIONAL OPERATION

3.1 I2C BUS CONCEPT

3.1.1 General Description

Two wires, serial data (SDA) and serial clock (SCL), carry information between the devices connected to the bus. Each device is recognized by a unique address — whether it's a microcontroller, LCD driver, memory or keyboard interface - and can operate as either a transmitter or receiver, depending on the function of the device. Obviously an LCD driver is only a receiver, whereas a memory can both receive and transmit data. In addition to transmitters and receivers, devices can also be considered as masters or slaves when performing data transfers. A master is the device which initiates a data transfer on the bus and generates the clock signals to permit that transfer. At that time, any device addressed is considered a slave.

The I2C bus is a multi-master bus. This means that more than one device capable of controlling the bus can be connected to it. As masters are usually micro-controllers, let's consider the case of a data transfer between two microcontrollers connected to the I2C bus. This highlights the master-slave and receiver-transmitter relationships to be found on the I2C bus. It should be noted that these relationships are not permanent, but only depend on the direction of data transfer at that time. The transfer of data would proceed as follows :

1: Suppose microcontroller A wants to send information to microcontroller B

- Microcontroller A (master) addresses microcontroller B(slave).
- Microcontroller A (master-transmitter) sends data to microcontroller B (slave).
- Microcontroller A terminates the transfer.

2: If microcontroller A wants to receive information from microcontroller B

- Microcontroller A (master) addresses microcontroller B (slave).
- Microcontroller A (master-receiver) receives data from microcontroller B(slave-transmitter).
- Microcontroller A terminates the transfer.

Even in this case, the master (microcontroller A) generates the timing and terminates the transfer.

The possibility of connecting more than one microcontroller to the I2C bus means that more than one master could try to initiate a data transfer at the same time. To avoid the chaos that might ensue from such an event —an arbitration procedure has been developed. This procedure relies on the wired-AND connection of all I2C interfaces to the I2C bus.

If two or more masters try to put information onto the bus, the first to produce a 'one' when the other produces a 'zero' will lose the arbitration. The clock signals during arbitration are a synchronized combination of the clocks generated by the masters using the wired-AND connection to the SCL.

Generation of clock signals on the I2C bus is always the responsibility of master devices; each master generates its own clock signals when transferring data on the bus. Bus clock signals from a master can only be altered when they are stretched by a slow-slave device holding-down the clock line or by another master when arbitration occurs.

3.1.2 General Characteristics

Both SDA and SCL are bi-directional lines, connected to a positive supply voltage via a pull-up. When the bus is free, both lines are HIGH. The output stages of devices connected to the bus must have an open-drain in order to perform the wired-AND function. Data on the I2C bus can be transferred at a rate up to 100Kbit/s in the standard mode, or up to 400Kbit/s in the fast mode. The number of interfaces connected to the bus is solely dependent on the bus capacitance limit of 400pF.

The table below shows some examples of configuration for some pre-defined baud rates, depending on the I2C clock, FAST mode and PRV field (in I2C_MR register):

Table 14-2. Examples of Baud Rate Configuration

I2C clock	Baudrate	FAST	PRV	% error
4 MHz	9600	0	0x196	0.16
	125K	1	0x01C	0.00
	192K	1	0x010	4.00
20 MHz	9600	0	0x821	0.03
	125K	1	0x09C	0.00
	192K	1	0.100	0.15
30 MHz	9600	0	0xC31	0.00
	125K	1	0X0EC	0.00
	192K	1	0x098	0.15

3.1.3 Bit Transfer

Due to the variety of different technology devices (CMOS, NMOS, bipolar) which can be connected to the I2C bus, the levels of the logical '0' (LOW) and '1' (HIGH) are not fixed and depend on the associated level of VDD. One clock pulse is generated for each data bit transferred.

3.1.3.1 Data Validity

The data on the SDA line must be stable during the HIGH period of the clock. The HIGH or LOW state of the data line can only change when the clock signal on the SCL line is LOW.

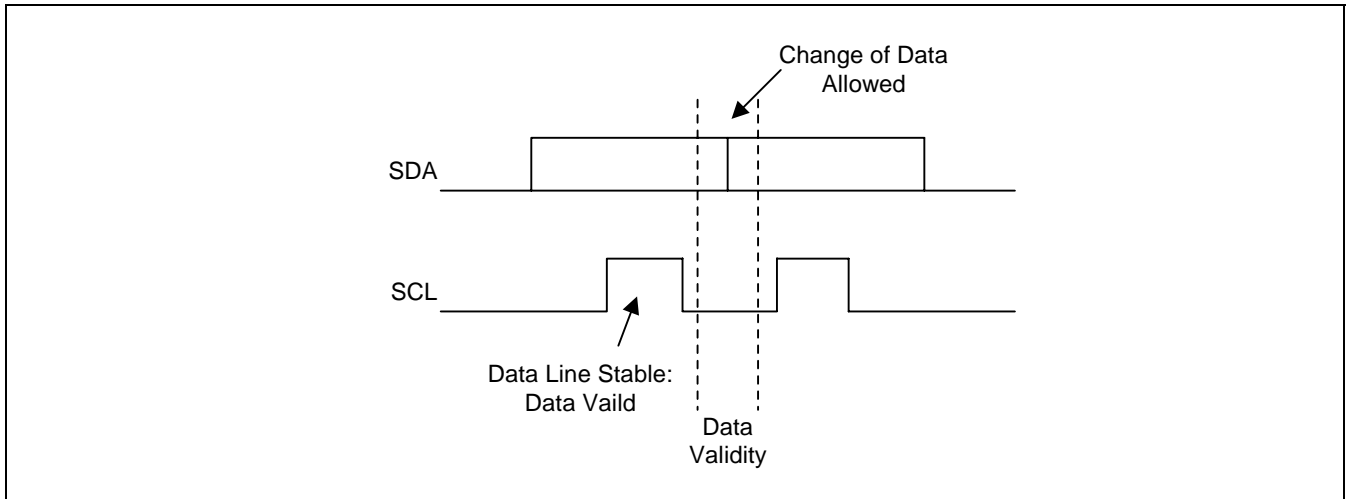


Figure 14-2. Data Validity

3.1.3.2 START and STOP Conditions

Within the procedure of the I2C bus, unique situations arise which are defined as START and STOP conditions. A HIGH to LOW transition on the SDA line while SCL is HIGH is one such unique case. This situation indicates a START condition.

A LOW to HIGH transition on the SDA line while SCL is HIGH defines a STOP condition. START and STOP conditions are always generated by the master.

The bus is considered to be busy after the START condition. The bus is considered to be free again a certain time after the STOP condition.

Detection of START and STOP conditions by devices connected to the bus is easy if they incorporate the necessary interfacing hardware. However, microcontrollers with no such interface have to sample the SDA line at least twice per clock period in order to sense the transition.

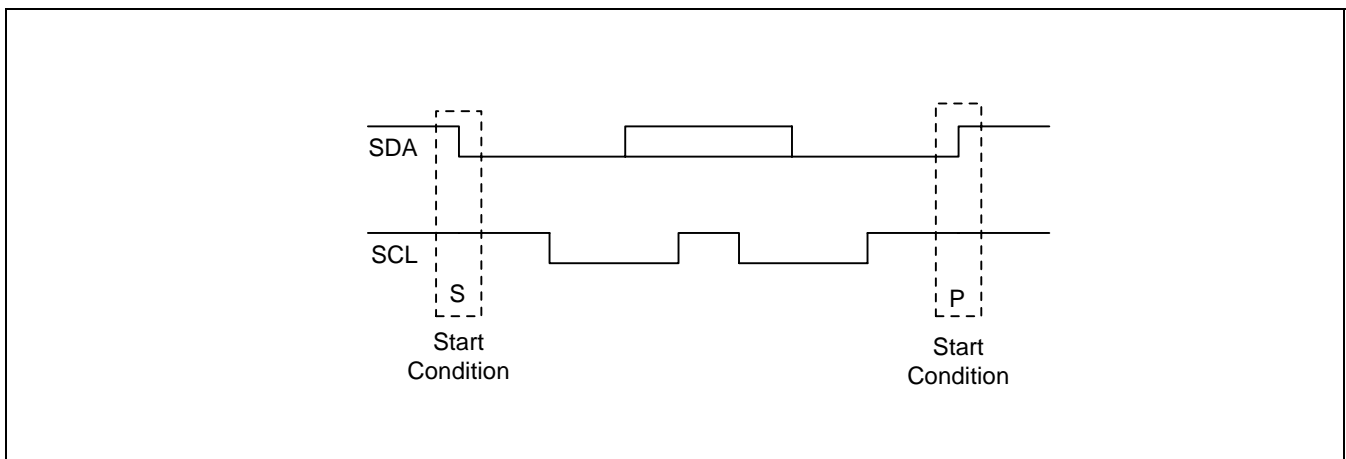


Figure 14-3. Start and Stop Conditions

3.1.4 Transferring Data

3.1.4.1 Byte Format

Every byte put on the SDA line must be 8-bits long. The number of bytes that can be transmitted per transfer is unrestricted. Each byte has to be followed by an acknowledge bit. Data is transferred with the most significant bit (MSB) first. If a receiver can't receive another complete byte of data until it has performed some other function, for example servicing an internal interrupt, it can hold the clock line SCL LOW to force the transmitter into a wait state. Data transfer then continues when the receiver is ready for another byte of data and releases clock line SCL.

In some cases, it's permitted to use a different format from the I2C-bus format (for CBUS compatible devices for example). A message which starts with such an address can be terminated by generation of a STOP condition, even during the transmission of a byte. In this case, no acknowledge is generated.

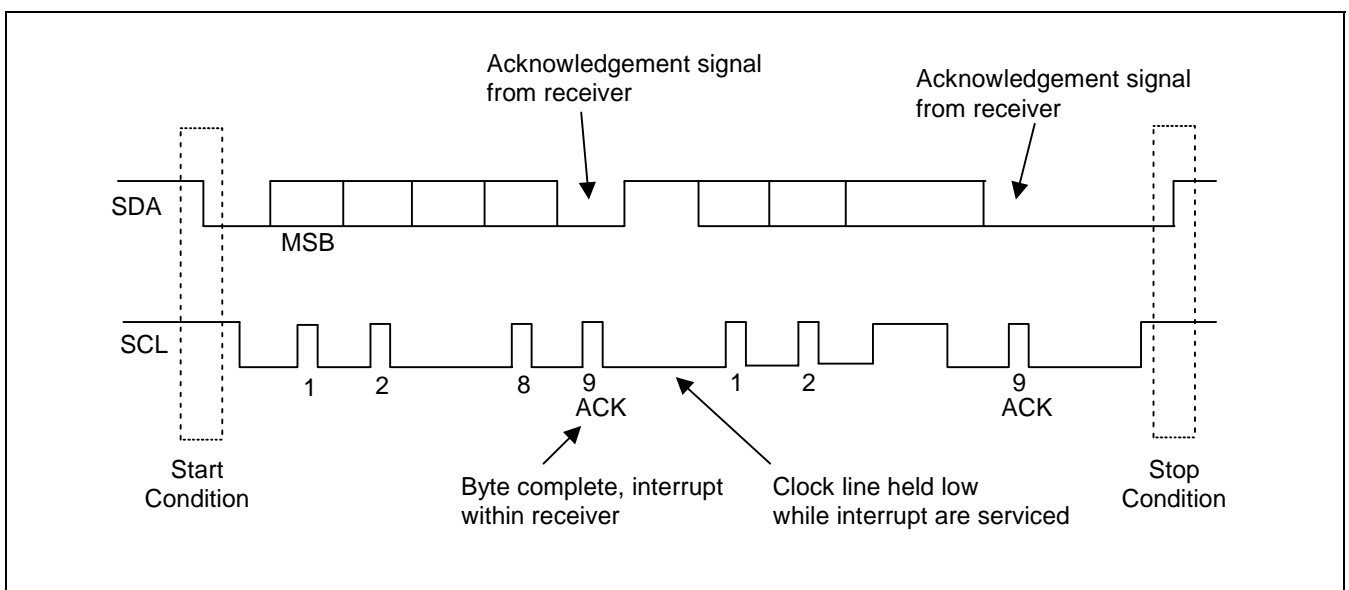


Figure 14-4. Data Transfer on the I2C Bus

3.1.4.2 Acknowledge

Data transfer with acknowledge is mandatory. The acknowledge-related clock pulse is generated by the master. The transmitter releases the SDA line (HIGH) during the acknowledge clock pulse.

The receiver must pull down the SDA line during the acknowledge clock pulse so that it remains stable LOW during the HIGH period of this clock pulse. Of course, set-up and hold times must also be taken into account.

Usually, a receiver which has been addressed is obliged to generate an acknowledge after each byte has been received, except when the message starts with a CBUS address.

When a slave-receiver doesn't acknowledge the slave address (for example, it's unable to receive because it's performing some real-time function), the data line must be left HIGH by the slave. The master can then generate a STOP condition to abort the transfer.

If a slave-receiver does acknowledge the slave address but, some time later in the transfer cannot receive any more data bytes, the master must again abort the transfer. This is indicated by the slave generating the not acknowledge on the first byte to follow. The slave leaves the data line HIGH and the master generates the STOP condition.

If a master-receiver is involved in a transfer, it must signal the end of data to the slave-transmitter by not generating an acknowledge on the last byte that was clocked out of the slave. The slave-transmitter must release the data line to allow the master to generate a STOP or repeated START condition.

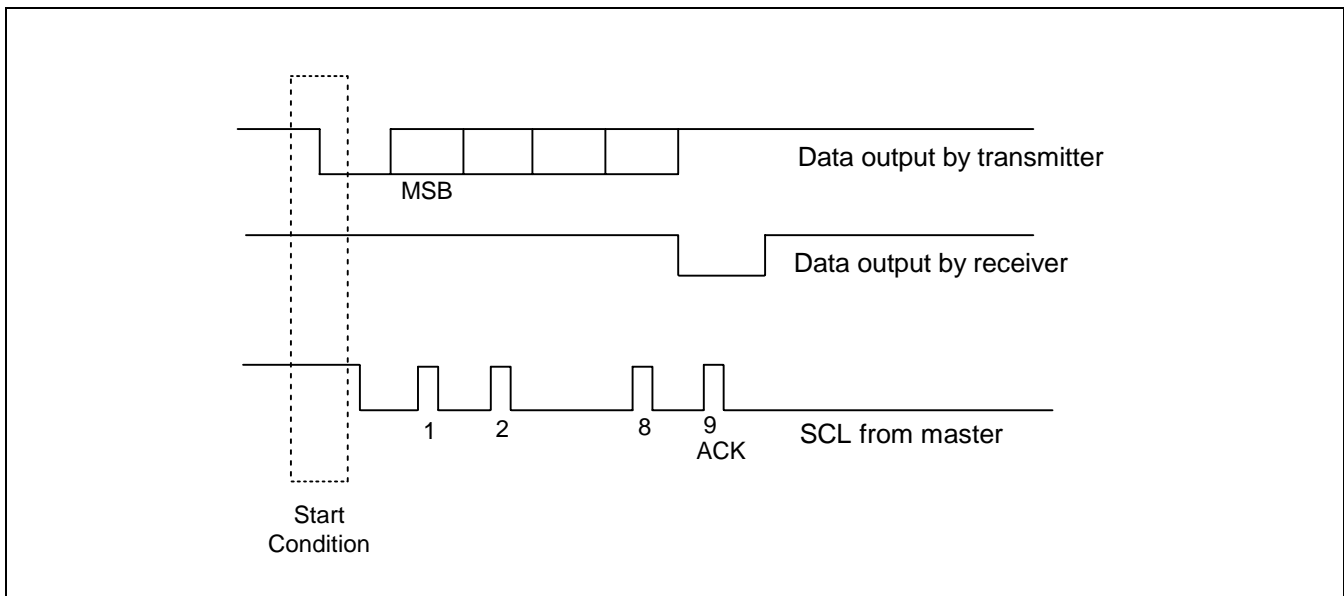


Figure 14-5. Acknowledge

3.1.5 Arbitration on Clock Generation

3.1.5.1 Synchronization

All masters generate their own clock on the SCL line to transfer messages on the I2C-bus. Data is only valid during the HIGH period of the clock. A defined clock is therefore needed for the bit-by-bit arbitration procedure to take place.

Clock synchronization is performed using the wired-AND connection of I2C interfaces to the SCL line. This means that a HIGH to LOW transition on the SCL line will cause the devices concerned to start counting off their LOW period and, once a device clock has gone LOW, it will hold the SCL line in that state until the clock HIGH state is reached. However, the LOW to HIGH transition of this clock may not change the state of the SCL line if another clock is still within its LOW period. The SCL line will therefore be held LOW by the device with the longest LOW period. Devices with shorter LOW periods enter a HIGH wait-state during this time. When all devices concerned have counted off their LOW period, the clock line will be released and go HIGH. There will then be no difference between the device clocks and the state of the SCL line, and all the devices will start counting their HIGH periods. The first device to complete its HIGH period will again pull the SCL line LOW.

In this way, a synchronized SCL clock is generated with its LOW period determined by the device with the longest clock LOW period, and its HIGH period determined by the one with the shortest clock HIGH period.

3.1.5.2 Arbitration

A master may start a transfer only if the bus is free. Two or more masters may generate a START condition within the minimum hold time (THD;STA) of the START condition which results in a defined START condition to the bus.

Arbitration takes place on the SDA line, while the SCL line is at the HIGH level, in such a way that the master which transmits a HIGH level, while another master is transmitting a LOW level will switch off its DATA output stage because the level on the bus doesn't correspond to its own level.

Arbitration can continue for many bits. Its first stage is comparison of the address bits. If the masters are each trying to address the same device, arbitration continues with comparison of the data. Because address and data information on the I2C-bus is used for arbitration, no information is lost during this process.

A master which loses the arbitration can generate clock pulses until the end of the byte in which it loses the arbitration.

If a master also incorporates a slave function and it loses arbitration during the addressing stage, it's possible that the winning master is trying to address it. The losing master must therefore switch over immediately to its slave-receiver mode.

Since control of the I2C -bus is decided solely on the address and data sent by competing masters, there is no central master, nor any order of priority on the bus.

Special attention must be paid if, during a serial transfer, the arbitration procedure is still in progress at the moment when a repeated START condition or a STOP condition is transmitted to the I2C -bus. If it's possible for such a situation to occur, the masters involved must send this repeated START condition or STOP condition at the same position in the format frame. In other words, arbitration isn't allowed between:

- A repeated START condition and a data bit
- A STOP condition and a data bit
- A repeated START condition and a STOP condition.

3.1.5.3 Use of the Clock Synchronizing Mechanism as a Handshake

In addition to being used during the arbitration procedure, the clock synchronization mechanism can be used to enable receivers to cope with fast data transfers, on either a byte level or a bit level.

On the byte level, a device may be able to receive bytes of data at a fast rate, but needs more time to store a received byte or prepare another byte to be transmitted. Slaves can then hold the SCL line LOW after reception and acknowledgement of a byte to force the master into a wait state until the slave is ready for the next byte transfer in a type of handshake procedure.

On the bit level, a device such as a microcontroller without, or with only a limited hardware I2C interface on-chip can slow down the bus clock by extending each clock LOW period. The speed of any master is thereby adapted to the internal operating rate of this device.

3.1.6 Formats with 7-bits Addresses

After the START condition (S), a slave address is sent. This address is 7 bits long followed by an eighth bit which is a data direction bit (R/W) - a 'zero' indicates a transmission (WRITE), a 'one' indicates a request for data (READ). A data transfer is always terminated by a STOP condition (P) generated by the master. However, if a master still wishes to communicate on the bus, it can generate a repeated START condition (Sr) and address another slave without first generating a STOP condition. Various combinations of read/write formats are then possible within such a transfer.

Possible data transfer formats are:

- Master-transmitter transmits to slave-receiver. The transfer direction is not changed
- Master reads slave immediately after first byte.

At the moment of the first acknowledge, the master-transmitter becomes a master-receiver and the slave-receiver becomes a slave-transmitter. This acknowledge is still generated by the slave.

The STOP condition is generated by the master.

- Combined formats. During a change of direction within a transfer, the START condition and the slave address are both repeated, but with the R/W bit reversed. If a master receiver sends a repeated START condition, it has previously sent a not acknowledge (A).

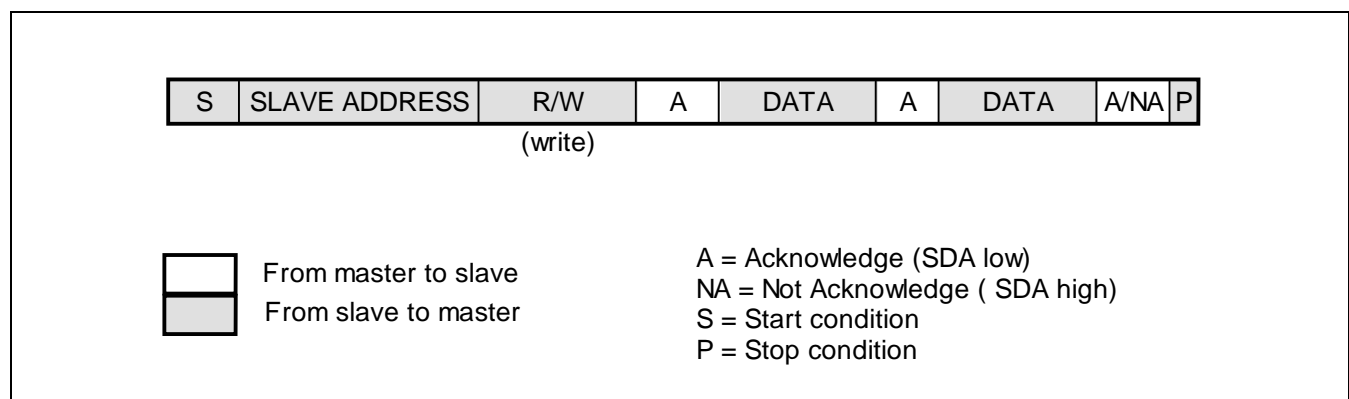


Figure 14-6. A master-Transmitter Addresses a Slave

3.1.7 7-Bits Addressing

The addressing procedure for the I2C-bus is such that the first byte after the START condition usually determines which slave will be selected by the master. The exception is the 'general call' address which can address all devices. When this address is used, all devices should, in theory, respond with an acknowledge. However, devices can be made to ignore this address. The second byte of the general call address then defines the action to be taken.

3.1.7.1 Definition of Bits in the First Byte

The first seven bits of the first byte make up the slave address. The eighth bit is the LSB (least significant bit). It determines the direction of the message. A 'zero' in the least significant position of the first byte means that the master will write information to a selected slave. A 'one' in this position means that the master will read information from the slave.

When an address is sent, each device in a system compares the first seven bits after the START condition with its address. If they match, the device considers itself addressed by the master as a slave-receiver or slave-transmitter, depending on the R/W bit.

A slave address can be made-up of a fixed and a programmable part. Since it's likely that there will be several identical devices in a system, the programmable part of the slave address enables the maximum possible number of such devices to be connected to the I2C -bus. The number of programmable address bits of a device depends on the number of pins available. For example, if a device has 4 fixed and 3 programmable address bits, a total of 8 identical devices can be connected to the same bus.

The I2C-bus committee coordinates allocation of I2C addresses.

Two groups of eight addresses (0000XXX and 1111XXX) are reserved for the purposes shown in Table 1. The bit combination 11110XX of the slave address is reserved for 10-bit addressing.

Table 14-3. Definition of Bytes in First Byte

Slave Address	RNW bit	Description
0000 000	0	General call address
0000 000	1	START byte (note 1)
0000 001	X	CBUS address (note 2)
0000 010	X	Reserved for different bus format (note 3)
0000 011	X	Reserved for future purposes
0000 1XX	X	HS-mode master code
1111 1XX	X	Reserved for future purposes
1111 0XX	X	10-bit slave addressing

NOTES:

1. No device is allowed to acknowledge at the reception of the START byte.
2. The CBUS address has been reserved to enable the inter-mixing of CBUS compatible and I2C -bus compatible devices in the same system. I2C -bus compatible devices are not allowed to respond on reception of this address.
3. The address reserved for a different bus format is included to enable I2C and other protocols to be mixed. Only I2C -bus compatible devices that can work with such formats and protocols are allowed to respond to this address.

The general call address is for addressing every device connected to the I2C -bus. However, if a device doesn't need any of the data supplied within the general call structure, it can ignore this address by not issuing an acknowledgement. If a device does require data from a general call address, it will acknowledge this address and behave as a slave-receiver.

The second and following bytes will be acknowledged by every slave-receiver capable of handling this data.

A slave which cannot process one of these bytes must ignore it by not acknowledging. The meaning of the general call address is always specified in the second byte.

There are two cases to consider:

- When the least significant bit B is a 'zero'
- When the least significant bit B is a 'one'.

When bit B is a 'zero'; the second byte has the following definition:

- 00000110 (H'06'). Reset and write programmable part of slave address by hardware. On receiving this 2-byte sequence, all devices designed to respond to the general call address will reset and take in the programmable part of their address. Precautions have to be taken to ensure that a device is not pulling down the SDA or SCL line after applying the supply voltage, since these low levels would block the bus.
- 00000100 (H'04'). Write programmable part of slave address by hardware. All devices which define the programmable part of their address by hardware (and which respond to the general call address) will latch this programmable part at the reception of this two byte sequence. The device will not reset.
- 00000000 (H'00'). This code is not allowed to be used as the second byte.

The remaining codes have not been fixed and devices must ignore them.

When bit B is a 'one'; the 2-byte sequence is a 'hardware general call'. This means that the sequence is transmitted by a hardware master device, such as a keyboard scanner, which cannot be programmed to transmit a desired slave address. Since a hardware master doesn't know in advance to which device the message has to be transferred, it can only generate this hardware general call and its own address - identifying itself to the system.

The seven bits remaining in the second byte contain the address of the hardware master. This address is recognized by an intelligent device (e.g. a microcontroller) connected to the bus which will then direct the information from the hardware master. If the hardware master can also act as a slave, the slave address is identical to the master address.

In some systems, an alternative could be that the hardware master transmitter is set in the slave-receiver mode after the system reset.

In this way, a system configuring master can tell the hardware master-transmitter (which is now in slave-receiver mode) to which address data must be sent. After this programming procedure, the hardware master remains in the master-transmitter mode.

3.1.7.2 Start Byte

Microcontrollers can be connected to the I2C -bus in two ways. A microcontroller with an on-chip hardware I2C – bus interface can be programmed to be only interrupted by requests from the bus. When the device doesn't have such an interface, it must constantly monitor the bus via software. Obviously, the more times the microcontroller monitors, or polls the bus, the less time it can spend carrying out its intended function. There is therefore a speed difference between fast hardware devices and a relatively slow microcontroller which relies on software polling.

In this case, data transfer can be preceded by a start procedure which is much longer than normal.

The start procedure consists of :

- A START condition (S)
- A START byte (00000001)
- An acknowledge clock pulse (ACK)
- A repeated START condition (Sr).

After the START condition S has been transmitted by a master which requires bus access, the START byte (00000001) is transmitted. Another microcontroller can therefore sample the SDA line at a low sampling rate until one of the seven zeros in the START byte is detected. After detection of this LOW level on the SDA line, the microcontroller can switch to a higher sampling rate to find the repeated START condition Sr which is then used for synchronization.

A hardware receiver will reset on receipt of the repeated START condition Sr and will therefore ignore the START byte.

An acknowledge-related clock pulse is generated after the START byte. This is present only to conform with the byte handling format used on the bus. No device is allowed to acknowledge the START byte.

3.1.8 Extensions to the I2C-bus Specification

The I2C -bus with a data transfer rate of up to 100 kbit/s and 7-bit addressing has now been in existence for more than ten years with an unchanged specification. The concept is accepted world-wide as a de facto standard and hundreds of different types of I2C -bus compatible ICs are available. The I2C -bus specification is now extended with the following two features:

- A fast-mode which allows a fourfold increase of the bit rate to 0 to 400 kbit/s
- 10-bit addressing which allows the use of up to 1024 additional addresses.

There are two reasons for these extensions to the I2C-bus specification:

- New applications will need to transfer a larger amount of serial data and will therefore demand a higher bit rate than 100 kbit/s. Improved IC manufacturing technology now allows a fourfold speed increase without increasing the manufacturing cost of the interface circuitry.
- Most of the 112 addresses available with the 7-bit addressing scheme have been issued more than once. To prevent problems with the allocation of slave addresses for new devices, it is desirable to have more address combinations. About a tenfold increase of the number of available addresses is obtained with the new 10-bit addressing.

All new devices with an I2C-bus interface are provided with the fast-mode. Preferably, they should be able to receive and/or transmit at 400 kbit/s. The minimum requirement is that they can synchronize with a 400 kbit/s transfer; they can then prolong the LOW period of the SCL signal to slow down the transfer. Fast-mode devices must be downward-compatible which means that they must still be able to communicate with 0 to 100 kbit/s devices in a 0 to 100 kbit/s I2C -bus system.

Obviously, devices with a 0 to 100 kbit/s I2C-bus interface cannot be incorporated in a fast-mode I2C-bus system because, since they cannot follow the higher transfer rate, unpredictable states of these devices would occur.

Slave devices with a fast-mode I2C -bus interface can have a 7-bit or a 10-bit slave address. However, a 7-bit address is preferred because it is the cheapest solution in hardware and it results in the shortest message length. Devices with 7-bit and 10-bit addresses can be mixed in the same I2C-bus system regardless of whether it is a 0 to 100 kbit/s standard-mode system or a 0 to 400 kbit/s fast-mode system. Both existing and future masters can generate either 7-bit or 10-bit addresses.

3.1.8.1 Fast Mode

In the fast-mode of the I2C-bus, the protocol, format, logic levels and maximum capacitive load for the SDA and SCL lines quoted in the previous I2C-bus specification are unchanged. Changes to the previous I2C-bus specification are:

- The maximum bit rate is increased to 400 kbit/s
- Timing of the serial data (SDA) and serial clock (SCL) signals has been adapted. There is no need for compatibility with other bus systems such as CBUS because they cannot operate at the increased bit rate.
- The inputs of fast-mode devices must incorporate spike suppression and a Schmitt trigger at the SDA and SCL inputs.
- The output buffers of fast-mode devices must incorporate slope control of the falling edges of the SDA and SCL signals.
- If the power supply to a fast-mode device is switched off, the SDA and SCL I/O pins must be floating so that they don't obstruct the bus lines.
- The external pull-up devices connected to the bus lines must be adapted to accommodate the shorter maximum permissible rise time for the fast-mode I2C-bus. For bus loads up to 200pF, the pull-up device for each bus line can be a resistor; for bus loads between 200pF and 400pF, the pull-up device can be a current source (3mA max.) or a switched resistor.

3.1.8.2 10-Bits Addressing

The 10-bit addressing does not change the format in the I2C-bus specification. Using 10 bits for addressing exploits the reserved combination 1111XXX for the first seven bits of the first byte following a START (S) or repeated START (Sr) condition.

The 10-bit addressing does not affect the existing 7-bit addressing. Devices with 7-bit and 10-bit addresses can be connected to the same I2C-bus, and both 7-bit and 10-bit addressing can be used in a standard-mode system (up to 100 kbit/s) or a fast-mode system (up to 400 kbit/s).

Although there are eight possible combinations of the reserved address bits 1111XXX, only the four combinations 11110XX are used for 10-bit addressing. The remaining four combinations 11111XX are reserved for future I2C-bus enhancements.

A - Definition of bits in the first two bytes

The 10-bit slave address is formed from the first two bytes following a START condition (S) or a repeated START condition (Sr).

The first seven bits of the first byte are the combination 11110XX of which the last two bits (XX) are the two most-significant bits (MSBs) of the 10-bit address; the eighth bit of the first byte is the R/W bit that determines the direction of the message. A 'zero' in the least significant position of the first byte means that the master will write information to a selected slave. A 'one' in this position means that the master will read information from the slave.

If the R/W bit is 'zero', then the second byte contains the remaining 8 bits (XXXXXXXX) of the 10-bit address. If the R/W bit is 'one', then the next byte contains data transmitted from a slave to a master.

B - Formats with 10-bit addresses

Various combinations of read/write formats are possible within a transfer that includes 10-bit addressing. Possible data transfer formats are:

- Master-transmitter transmits to slave-receiver with a 10-bit slave address. The transfer direction is not changed. When a 10-bit address follows a START condition, each slave compares the first seven bits of the first byte of the slave address (11110XX) with its own address and tests if the eighth bit (R/Wdirection bit) is 0. It is possible that more than one device will find a match and generate an acknowledge (A1). All slaves that found a match will compare the eight bits of the second byte of the slave address (XXXXXXXX) with their own addresses, but only one slave will find a match and generate an acknowledge (A2). The matching slave will remain addressed by the master until it receives a STOP condition (P) or a repeated START condition (Sr) followed by a different slave address.
- Master-receiver reads slave-transmitter with a 10-bit slave address. The transfer direction is changed after the second R/W bit. Up to and including acknowledge bit A2, the procedure is the same as that described for a master-transmitter addressing a slave-receiver. After the repeated START condition (Sr), a matching slave remembers that it was addressed before. This slave then checks if the first seven bits of the first byte of the slave address following Sr are the same as they were after the START condition (S), and tests if the eighth (R/W) bit is 1. If there is a match, the slave considers that it has been addressed as a transmitter and generates acknowledge A3.

The slave-transmitter remains addressed until it receives a STOP condition (P) or until it receives another repeated START condition (Sr) followed by a different slave address. After a repeated START condition (Sr), all the other slave devices will also compare the first seven bits of the first byte of the slave address (11110XX) with their own addresses and test the eighth (R/W) bit. However, none of them will be addressed because R/W = 1 (for 10-bit devices), or the 11110XX slave address (for 7-bit devices) does not match.

- Combined format. A master transmits data to a slave and then reads data from the same slave. The same master occupies the bus all the time. The transfer direction is changed after the second R/W bit – Combined format. A master transmits data to one slave and then transmits data to another slave. The same master occupies the bus all the time.
- Combined format. 10-bit and 7-bit addressing combined in one serial transfer. After each START condition (S), or each repeated START condition (Sr), a 10-bit or 7-bit slave address can be transmitted.

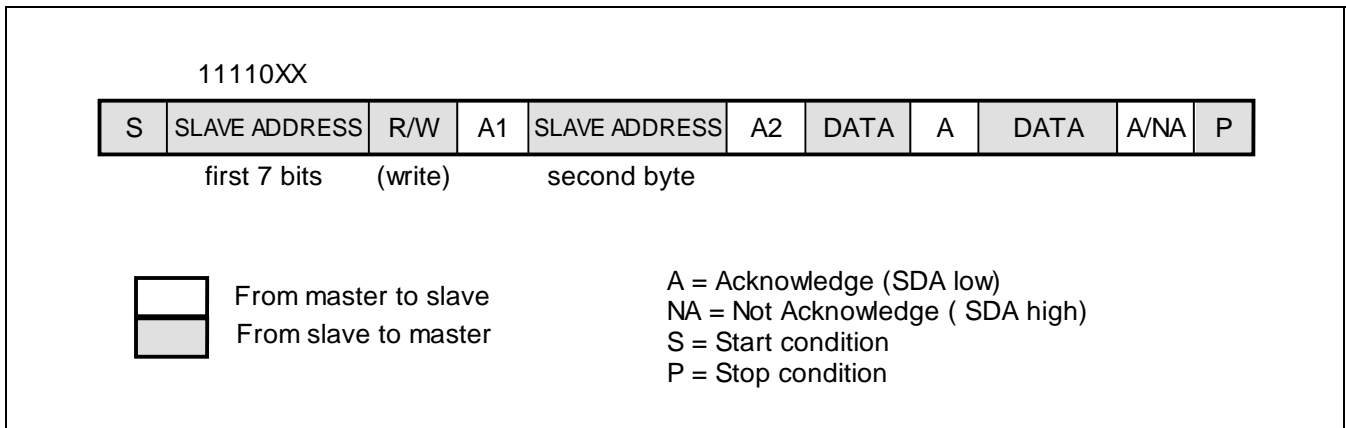


Figure 14-7. A master-Transmitter Addresses a Slave-Receiver with a 10–bits Address

3.1.9 General Call Address and Start Byte

The 10-bit addressing procedure for the I2C -bus is such that the first two bytes after the START condition (S) usually determine which slave will be selected by the master. The exception is the 'general call' address 00000000 (H'00').

Slave devices with 10-bit addressing will react to a 'general call' in the same way as slave devices with 7-bit addressing.

Hardware masters can transmit their 10-bit address after a 'general call'. In this case, the 'general call' address byte is followed by two successive bytes containing the 10-bit address of the master-transmitter.

The START byte 00000001 (H'01') can precede the 10-bit addressing in the same way as for 7-bit addressing.

4. REGISTERS DESCRIPTION

Base Addresses – I2C0 : 0xFFE50000
I2C1 : 0xFFE54000

Table 14-4. I2C Special Function Registers

Offset Address	Name	Description	R/W	Reset State
0x000 – 0x04C	–	Reserved	–	–
0x050	I2C_ECR	Enable clock register	W	–
0x054	I2C_DCR	Disable clock register	W	–
0x058	I2C_PMSR	Power management status register	R	0xFFFFFFFF0
0x05C	–	Reserved	–	–
0x060	I2C_CR	Control register	R/W	0x00000000
0x064	I2C_MR	Mode register	R/W	0x000001F4
0x068	–	Reserved	–	–
0x06C	–	Reserved	–	–
0x070	I2C_SR	Status register	R	0x000000F8
0x074	I2C_IER	Interrupt enable register	W	–
0x078	I2C_IDR	Interrupt disable register	W	–
0x07C	I2C_IMR	Interrupt mask register	R	0x00000000
0x080	I2C_DAT	Serial data register	R/W	0x00000000
0x084	I2C_ADR	Serial slave address register	R/W	0x00000000
0x088	I2C_THOLD	Hold/Setup delay register	R/W	0x00000001

I2C Enable Clock Register

I2C_ECR (0x050)

Access: Write only

31	30	29	28	27	26	25	24
DBGEN	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	I2C	–
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- I2C : I2C clock enable**

0: No effect.

1: Enables the I2C clock.

- DBGEN : debug mode enable**

0: No effect.

1: Enables the debug feature on the I2C block.

I2C Disable Clock Register

I2C_DCR (0x054)

Access: Write only

31	30	29	28	27	26	25	24
DBGEN	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	I2C	–
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **I2C : I2C clock disable**

0: No effect.

1: Disables the I2C clock.

- **DBGEN : debug mode disable**

0: No effect.

1: Disables the debug feature on the I2C block.

I2C Power Management Status Register **I2C_PMSR (0x058)** **Access: Read only**

31	30	29	28	27	26	25	24
DBGEN	–	IPIDCODE[25:20]					
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
IPIDCODE[19:12]							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
IPIDCODE[11:4]							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
IPIDCODE[3:0]				–	–	I2C	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **I2C clock status**

0: I2C clock disabled.

1: I2C clock enabled.

- **IPIDCODE[25:0]**

This field contains the version number of the module, coded on 26 bits.

- **DBGEN : debug mode**

0: dbgack_sclk input has no influence on I2C function.

1: dbgack_sclk is enabled. When this signal is low, the I2C function is left unchanged, and when this signal is high, the I2C function is frozen. However, full read/write access to internal register is kept for the debug purpose.

I2C Control Register			I2C_CR (0x060)				Access: Read/Write	
31	30	29	28	27	26	25	24	
–	–	–	–	–	–	–	–	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
23	22	21	20	19	18	17	16	
–	–	–	–	–	–	–	–	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
15	14	13	12	11	10	9	8	
–	–	–	–	–	–	–	ENA	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
7	6	5	4	3	2	1	0	
–	–	–	SI	STA	STO	AA	SWRST	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **SWRST : I2C software reset**

0: No effect.

1: Generate a software reset (I2C_PMSR register is not resettled).

- **AA : I2C acknowledge.**

0: No acknowledge is returned (SDA remains high during the acknowledge SCL clock pulses).

1: Acknowledge is returned when the “own slave address” is received (or the general call address has been received while bit GC of I2C_ADR is set) or when a data byte has been received while being in the receiver mode.

- **STO : I2C stop.**

0: Clear not to send a STOP condition on the bus.

1: Generates a stop condition. When the STOP condition is detected on bus, the STO bit is automatically cleared. In slave mode, this bit is used to recover from a bus error. In such case, no STOP condition is transmitted, but the interface do as if a STOP condition has been received and switch to the 'not addressed' slave mode (the STO bit is cleared by the interface).

- **STA : I2C start**

0: Set to '0' put the Interface in slave mode.

1: When set to '1', the interface enters in master mode, check the status of the I2C bus and generates a START condition if the bus is free. If the bus is not free, the Interface will wait until a STOP condition to generate a START condition (after a minimum time). Set to '0' put the Interface in slave mode.

- **SI : I2C interrupt**

0: Clear SI.

1: An interrupt has to be serviced. When the SI is '1', the i2c_int line is high, and the SCL line is pulled low. The transfer is suspend until the SI bit is cleared.

- **ENA : I2C enable**

0: Disables the I2C Interface (When the I2C interface is disabled, the SCL and SDA lines are disabled. No output is generated and no input is taken account).

1: Enables the I2C Interface.

I2C Mode Register **I2C_MR (0x064)** **Access: Read/Write**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
–	–	–	FAST	PRV[11:8]			
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1
7	6	5	4	3	2	1	0
PRV[7:0]							
R/W-1	R/W-1	R/W-1	R/W-1	R/W-0	R/W-1	R/W-0	R/W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **PRV[11:0] : pre-scalar value**

These bits select the speed of the bus (FSCL). The value of PRV (pre-scalar Value) is used to generate the FSCL with the formula :

$$FSCL = PCLK / (PRV+4)$$

- **FAST : Fast mode**

0: Disables FAST mode. Enables STANDARD mode. In this mode the high to low ratio is 1:1 and the maximum baud rate is 100 kHz.

1: Enables FAST mode. In this mode the high to low ratio is 2:3 and the maximum baud rate is 400 kHz.

Table 14-5. Values of FSCL in kHz Depending of PRV, FAST and PCLK

PRV (decimal)	FAST	PCLK (MHz)				
		10	20	25	40	50
500	0	19.8	39.7	49.6	80	99.2
400	0	24.7	49.5	61.8	99	–
250	0	39.3	78.7	98.4	–	–
200	0	49	98	–	–	–
125	0	77.5	–	–	–	–
125	1	77.5	155	194	310	387
100	1	96	192	240	400	–
62.5	1	150	300	375	–	–
50	1	185	370	–	–	–
25	1	344	–	–	–	–

NOTES:

1. The PCLK frequency must be at least 6 times greater than the faster FSCL frequency used on the I2C bus (SCL re-synchronization + state machine).
2. The PCLK frequency must be 6 times greater than the desired FSCL value programmed in PRV.
3. PRV value after reset is '500' (decimal).
4. It is not possible to write '000' (hexadecimal) to PRV.

I2C Status Register

I2C_SR (0x070)

Access: Read only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
SR[4:0]					–	–	–
R-1	R-1	R-1	R-1	R-1	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- SR[4:0] : I2C status code**

Interface Status code. There are 27 possible status codes.

I2C_SR reset value is 0x000000F8. When the I2C_SR contains this status code, no relevant status information is available. All other status codes correspond to a defined state of the interface. When each of these states is entered the corresponding status code appears in this register and the bit SI of I2C_CR is set.

All of these status codes, their meaning, the next action to be performed by the sequencer/controller (software) driving the interface and the next action the interface will perform, are described next page.

Table 14-6. Master/Transmitter Mode Status Codes

Status Code	Meaning	Next action to be Done by Controller (Software)				Next Action Interface will Perform	
		STA	STO	AA	SI		
0x00000008	START condition has been transmitted	x	0	x	0	Load I2C_DAT with slave address and R/W bit. Reset SI of I2C_CR.	Slave address and R/W will be transmitted. Wait ACK.
0x00000010	REPEAT START condition has been transmitted	x	0	x	0	Load I2C_DAT with slave address and R/W bit. Reset SI of I2C_CR.	Slave address and R/W will be transmitted. Wait ACK. If R/W is Read, interface switch into receiver mode.
0x00000018	Slave address and WRITE has been sent, ACK received	0	0	x	0	Load I2C_DAT with data byte. Reset SI of I2C_CR.	Data byte will be transmitted. Wait for ACK.
		1	0	x	0	Set STA of I2C_CR. Reset SI of I2C_CR.	Generates repeated START condition.
		0	1	x	0	Set STO of I2C_CR Reset SI of I2C_CR.	Generates STOP condition.
		1	1	x	0	Set STA and STO of I2C_CR. Reset SI of I2C_CR.	Generates STOP condition, then generates SART condition
0x00000020	Slave address and WRITE has been sent, No ACK received	As above.			As above.	As above.	
0x00000028	Data byte transmitted, ACK received	As above.			As above.	As above.	
0x00000030	Data byte transmitted, No ACK received	As above.			As above.	As above.	
0x00000038	Arbitration lost in Slave address and R/W bit transmission or in data byte transmission	0	0	x	0	Reset SI of I2C_CR.	Release I2C bus, switch to slave mode.
		1	0	x	0	Set STA of I2C_CR. Reset SI of I2C_CR.	Wait until the I2C is free to generate a START condition.

Table 14-7. Master/Receiver Mode Status Codes

Status Code	Meaning	Next action to be Done by Controller (Software)					Next Action Interface will Perform
		STA	STO	AA	SI		
0x00000008	START condition has been transmitted	x	0	x	0	Load I2C_DAT with slave address and R/W bit. Reset SI of I2C_CR.	Slave address and R/W will be transmitted. Wait ACK.
0x00000010	REPEAT START condition has been transmitted	x	0	x	0	Load I2C_DAT with slave address and R/W bit. Reset SI of I2C_CR.	Slave address and R/W will be transmitted. Wait ACK. If R/W is Read, interface switch into receiver mode.
0x00000038	Arbitration lost in Slave address and R/W bit transmission	0	0	x	0	Reset SI of I2C_CR.	Release I2C bus, switch to slave mode.
		1	0	x	0	Set STA of I2C_CR. Reset SI of I2C_CR.	Wait until the I2C is free to generate a START condition.
0x00000040	Slave address and Read has been sent, ACK received	0	0	1	0	Reset SI of I2C_CR. Set AA of I2C_CR.	Data byte will be received, ACK will be returned.
		0	0	0	0	Reset SI of I2C_CR. Reset AA of I2C_CR.	Data byte will be received, No ACK will be returned.
0x00000048	Slave address and Read has been sent, No ACK received	1	0	x	0	Set STA of I2C_CR Reset SI of I2C_CR.	Generates repeated START condition.
		0	1	x	0	Set STO of I2C_CR. Reset SI of I2C_CR.	Generates STOP condition.
		1	1	x	0	Set STA and STO of I2C_CR. Reset SI of I2C_CR.	Generates STOP condition, then generates START condition
0x00000050	Data byte received, ACK returned	0	0	1	0	Read data byte, reset SI of I2C_CR, set AA of I2C_CR.	New data byte will be received and ACK will be returned.
		0	0	0	0	Read data byte, reset SI of I2C_CR, reset AA of I2C_CR.	New data byte will be received and no ACK will be returned.

Table 14-7. Master/Receiver Mode Status Codes (Continued)

Status Code	Meaning	Next action to be Done by Controller (Software)				Next Action Interface will Perform	
		STA	STO	AA	SI		
0x00000058	Data byte received, No ACK returned.	1	0	x	0	Read data byte, set STA of I2C_CR, reset SI of I2C_CR.	Generates repeated START condition.
		0	1	x	0	Read data byte, set STO of I2C_CR, reset SI of I2C_CR.	Generates STOP condition.
		1	1	x	0	Read data byte, set STO of I2C_CR, set STA of I2C_CR, reset SI of I2C_CR.	Generates STOP condition, then generates START condition

Table 14-8. Slave/Receiver Mode Status Codes

Status Code	Meaning	Next action to be Done by Controller (Software)					Next Action Interface will Perform
		STA	STO	AA	SI		
0x00000060	Own Slave address +W received, ACK returned.	x	0	1	0	Reset SI of I2C_CR. Set AA of I2C_CR.	Data byte will be received, ACK will be returned.
		x	0	0	0	Reset SI of I2C_CR. Reset AA of I2C_CR.	Data byte will be received, No ACK will be returned.
0x00000068	Arbitration lost in Slave address + R/W (in master mode); switch to slave mode, own slave address received; ACK has been returned	x	0	1	0	Reset SI of I2C_CR. Set AA of I2C_CR.	Data byte will be received, ACK will be returned.
		x	0	0	0	Reset SI of I2C_CR. Reset AA of I2C_CR.	Data byte will be received, No ACK will be returned.
0x00000070	General Call Address has been received; ACK has been returned	x	0	1	0	Reset SI of I2C_CR. Set AA of I2C_CR.	Data byte will be received, ACK will be returned.
		x	0	0	0	Reset SI of I2C_CR. Reset AA of I2C_CR.	Data byte will be received, No ACK will be returned.
0x00000078	Arbitration lost in Slave address + R/W (in master mode); switch to slave mode, general call address received; ACK has been returned	x	0	1	0	Reset SI of I2C_CR. Set AA of I2C_CR.	Data byte will be received, ACK will be returned.
		x	0	0	0	Reset SI of I2C_CR. Reset AA of I2C_CR.	Data byte will be received, No ACK will be returned.

Table 14-8. Slave/Receiver Mode Status Codes (Continued)

Status Code	Meaning	Next action to be Done by Controller (Software)					Next Action Interface will Perform
		STA	STO	AA	SI		
0x00000080	Addressed with own address and W, data byte received, ACK returned	x	0	1	0	Reset SI of I2C_CR. Set AA of I2C_CR.	Data byte will be received, ACK will be returned.
		x	0	0	0	Reset SI of I2C_CR. Reset AA of I2C_CR.	Data byte will be received, No ACK will be returned.
0x00000088	Addressed with own address, data byte received, no ACK returned	0	0	0	0	Read data byte, reset SI of I2C_CR. Reset AA of I2C_CR.	Switch to 'not addressed' slave mode. Own address and general call recognition disabled.
		0	0	1	0	Read data byte, reset SI of I2C_CR. Set AA of I2C_CR.	Switch to 'not addressed' slave mode. Acknowledge own slave address and general call (if GC='1' of I2C_ADR).
		1	0	0	0	Read data byte, reset SI of I2C_CR. Reset AA of I2C_CR. Set STA of I2C_CR.	Switch to 'not addressed' slave mode. Own address and general call recognition disabled. Start will be transmitted as soon as the bus becomes free.
		1	0	1	0	Read data byte, reset SI of I2C_CR. Set AA of I2C_CR. Set STA of I2C_CR.	Switch to 'not addressed' slave mode. Acknowledge own slave address and general call (if GC='1' of I2C_ADR). Start will be transmitted as soon as the bus becomes free.

Table 14-8. Slave/Receiver Mode Status Codes (Continued)

Status Code	Meaning	Next action to be Done by Controller (Software)					Next Action Interface will Perform
		STA	STO	AA	SI		
0x00000090	Addressed by general call address, data byte received, ACK returned	x	0	1	0	Read data byte, reset SI of I2C_CR. Set AA of I2C_CR.	Data byte will be received, ACK returned.
		x	0	0	0	Read data byte, reset SI of I2C_CR. Reset AA of I2C_CR.	Data byte will be received, no ACK returned.
0x00000098	Addressed by general call address, data byte received, no ACK returned	0	0	0	0	Read data byte, reset SI of I2C_CR. Reset AA of I2C_CR.	Switch to 'not addressed' slave mode. Own address and general call recognition disabled.
		0	0	1	0	Read data byte, reset SI of I2C_CR. Set AA of I2C_CR.	Switch to 'not addressed' slave mode. Acknowledge own slave address and general call (if GC='1' of I2C_ADR).
		1	0	0	0	Read data byte, reset SI of I2C_CR. Reset AA of I2C_CR. Set STA of I2C_CR	Switch to 'not addressed' slave mode. Own address and general call recognition disabled. Start will be transmitted as soon as the bus becomes free.
		1	0	1	0	Read data byte, reset SI of I2C_CR. Set AA of I2C_CR. Set STA of I2C_CR.	Switch to 'not addressed' slave mode. Acknowledge own slave address and general call (if GC='1' of I2C_ADR). Start will be transmitted as soon as the bus becomes free.

Table 14-8. Slave/Receiver Mode Status Codes (Continued)

Status Code	Meaning	Next action to be Done by Controller (Software)					Next Action Interface will Perform
		STA	STO	AA	SI		
0x000000A0	A stop condition or repeated start condition has been received while still addressed as slave	0	0	0	0	Reset SI of I2C_CR. Reset AA of I2C_CR.	As Above
		0	0	1	0	Reset SI of I2C_CR. Set AA of I2C_CR.	
		1	0	0	0	Reset SI of I2C_CR. Reset AA of I2C_CR. Set STA of I2C_CR.	
		1	0	1	0	Reset SI of I2C_CR. Set AA of I2C_CR. Set STA of I2C_CR.	

Table 14-9. Slave/Transmitter Mode Status Codes

Status Code	Meaning	Next action to be Done by Controller (Software)					Next Action Interface will Perform
		STA	STO	AA	SI		
0x000000A8	Own Slave address + R received, ACK returned.	x	0	1	0	Write data byte. Reset SI of I2C_CR. Set AA of I2C_CR.	Data byte will be transmitted
		x	0	0	0	Write data byte. Reset SI of I2C_CR. Reset AA of I2C_CR.	Last data byte will be transmitted and slave will not further respond.
0x000000B0	Arbitration lost in slave address + R/W as master; own slave address has been received; ACK has been returned.	x	0	1	0	Write data byte. Reset SI of I2C_CR. Set AA of I2C_CR.	Data byte will be transmitted
		x	0	0	0	Write data byte. Reset SI of I2C_CR. Reset AA of I2C_CR.	Last data byte will be transmitted and slave will not further respond.
0x000000B8	Data has been transmitted; ACK has been received.	x	0	1	0	Write data byte. Reset SI of I2C_CR. Set AA of I2C_CR.	Data byte will be transmitted
		x	0	0	0	Write data byte. Reset SI of I2C_CR. Reset AA of I2C_CR.	Last data byte will be transmitted and slave will not further respond.

Table 14-9. Slave/Transmitter Mode Status Codes (Continued)

Status Code	Meaning	Next action to be Done by Controller (Software)					Next Action Interface will Perform
		STA	STO	AA	SI		
0x000000C0	Data has been transmitted; No ACK has been received	0	0	0	0	Reset SI of I2C_CR. Reset AA of I2C_CR.	Switch to 'not addressed' slave mode. Own address and general call recognition disabled.
		0	0	1	0	Reset SI of I2C_CR. Set AA of I2C_CR.	Switch to 'not addressed' slave mode. Acknowledge own slave address and general call (if GC='1' of I2C_ADR).
		1	0	0	0	Reset SI of I2C_CR. Reset AA of I2C_CR. Set STA of I2C_CR.	Switch to 'not addressed' slave mode. Own address and general call recognition disabled. Start will be transmitted as soon as the bus becomes free.
		1	0	1	0	Reset SI of I2C_CR. Set AA of I2C_CR. Set STA of I2C_CR.	Switch to 'not addressed' slave mode. Acknowledge own slave address and general call (if GC='1' of I2C_ADR). Start will be transmitted as soon as the bus becomes free.

Table 14-9. Slave/Transmitter Mode Status Codes (Continued)

Status Code	Meaning	Next action to be Done by Controller (Software)					Next Action Interface will Perform
		STA	STO	AA	SI		
0x000000C8	Last data has been transmitted, ACK received.	0	0	0	0	Reset SI of I2C_CR. Reset AA of I2C_CR.	As Above
		0	0	1	0	Reset SI of I2C_CR. Set AA of I2C_CR.	
		1	0	0	0	Reset SI of I2C_CR. Reset AA of I2C_CR. Set STA of I2C_CR.	
		1	0	1	0	Reset SI of I2C_CR. Set AA of I2C_CR. Set STA of I2C_CR.	

Table 14-10. Miscellaneous Status Codes

Status Code	Meaning	Next action to be Done by Controller (Software)					Next Action Interface will Perform
		STA	STO	AA	SI		
0x000000F8	No relevant state information; SI='0' in I2C_CR	–	–	–	–	No action.	Wait or proceed current transfer.
0x00000000	Bus error due to an illegal START or STOP condition.	0	1	x	0	No action.	Only internal hardware is affected. In all cases, the bus is released and STO is reset.

I2C Interrupt Enable Register

I2C_IER (0x074)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	SI	–	–	–	–
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **SI : SI interrupt enable**

0: No effect

1: Enables SI interrupt

I2C Interrupt Disable Register

I2C_IDR (0x078)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	SI	–	–	–	–
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **SI : SI interrupt disable**

0: No effect

1: Disables SI interrupt

I2C Interrupt Mask Register

I2C_IMR (0x07C)

Access: Read only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
–	–	–	SI	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write R: Read -0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **SI : SI interrupt mask**

0: Interrupt SI is not enabled

1: Interrupt SI is enabled.

I2C Serial Data Register				I2C_DAT (0x080)				Access: Read/Write
31	30	29	28	27	26	25	24	
–	–	–	–	–	–	–	–	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
23	22	21	20	19	18	17	16	
–	–	–	–	–	–	–	–	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
15	14	13	12	11	10	9	8	
–	–	–	–	–	–	–	–	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
7	6	5	4	3	2	1	0	
DAT[7:0]								
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **DAT[7:0] : I2C data**

Contains the incoming byte (just received from the I2C-bus) in the receiver mode and the outgoing data (to be send to the I2C-bus) in transmitter mode.

DAT is always shifted from right to left. It means, the first bit transmitted to the I2C-bus is the MSB in transmitter mode, and the MSB is the first bit received from the I2C-bus in the receiver mode.

During a transmission (when the interface send data to the I2C-bus) while the data is shifted out, the value on the SDA line is shifted in. So in case of an arbitration lost, DAT will contains the correct data (the value read from the bus).

I2C Serial Slave Address Register				I2C_ADR (0x084)				Access: Read/Write
31	30	29	28	27	26	25	24	
–	–	–	–	–	–	–	–	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
23	22	21	20	19	18	17	16	
–	–	–	–	–	–	–	–	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
15	14	13	12	11	10	9	8	
–	–	–	–	–	–	–	–	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
7	6	5	4	3	2	1	0	
ADR[6:0]							GC	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **GC : General call**

Enable the recognition of 'general call address'. When GC is set to '1', the interface will generate an interrupt when the 'general call address' is recognized.

- **ADR[6:0] : I2C address**

Contains the 7-bit I2C address to which the interface will respond when programmed as a slave (transmitter or receiver).

I2C Hold/Setup Delay Register				I2C_THOLD (0x088)				Access: Read/Write
31	30	29	28	27	26	25	24	
–	–	–	–	–	–	–	–	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
23	22	21	20	19	18	17	16	
–	–	–	–	–	–	–	–	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
15	14	13	12	11	10	9	8	
–	–	–	–	–	–	–	–	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
7	6	5	4	3	2	1	0	
DL[7:0]								
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **DL[7:0] : Hold/setup delay**

Contains the value of the Hold/Setup delay that is calculated with the formula:

$$\text{THOLD} = \text{DL}[7:0] * \text{PCLK}$$

$$\text{TSETUP} = \text{DL}[7:0] * \text{PCLK}$$

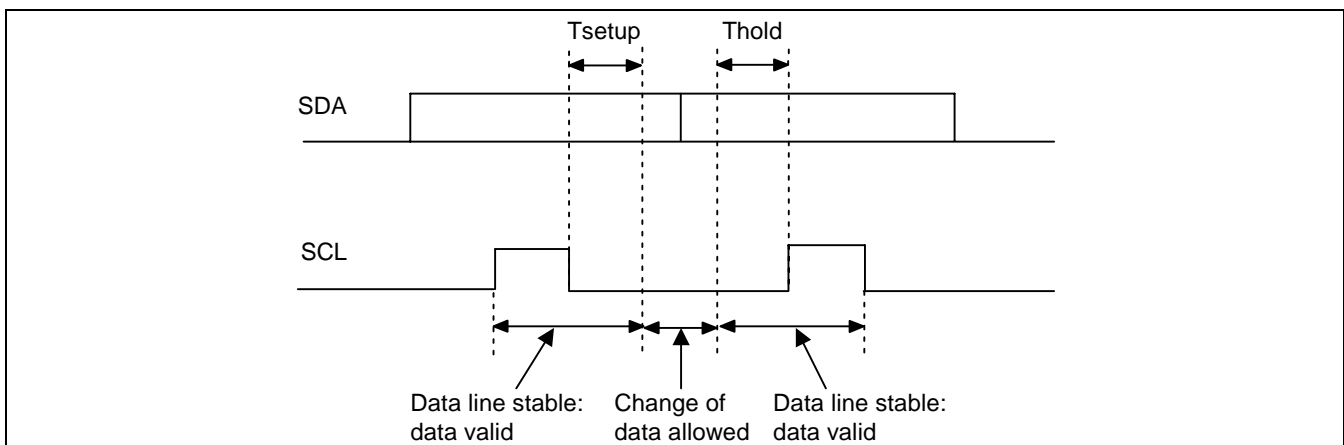


Figure 14-8. Hold and Setup Delays

NOTES:

1. DL value after reset is '1' (hexadecimal).
2. Setup delay (TSETUP) must be 250 ns (min in standard mode) or 100 ns (min in fast mode).
3. An I2C device must internally provide a hold time of at least 300 ns for the SDA signal. It is the user's responsibility to program the correct hold value to meet timing requirements of slow devices.
4. It is not possible to write '0' (hexadecimal) to DL.

5. TIMINGS

Table 14-11. Timing Requirements

Parameter	Symbol	Standard-Mode I2C Bus		Fast-Mode I2C Bus		Unit
		Min	Max	Min	Max	
SCL clock frequency	FSCL	0	100	0	400	kHz
Bus free time between a STOP and START condition	TBUF	4.7	–	1.3	–	μF
Hold time (repeated) START condition. After this period, the first clock pulse is generated	THD;STA	4.0	–	0.6	–	μF
Low period of the SCL clock	TLOW	4.7	–	1.3	–	μF
High period of the SCL clock	THIGH	4.0	–	0.6	–	μF
Set-up time for a repeated START condition	TSU;STA	4.7	–	0.6	–	μF
Data hold time	THD;DAT	0	–	0	0.9	μF
Data set-up time	TSU;DAT	250	–	100	–	ns
Rise time for both SDA and SCL signals	Tr	–	1000	20+01Cb	300	ns
Fall time for both SDA and SCL signals	Tf	–	300	20+01Cb	300	ns
Set-up time for STOP condition	TSU;STO	4.0	–	0.6	–	μF
Capacitive load for each bus line	Cb	–	400	–	400	μF

15

INTERLEAVE PROGRAM FLASH MEMORY

1. OVERVIEW

1.1 FUNCTIONAL DESCRIPTION

The Interleaved Program Flash Memory Controller (IFC) is a generic module used to access synchronous flash.

The memory flash size is 512Kbytes. To improve read access performance, the memory is composed of two interleaved flash memories.

The IFC has 2 different working modes:

- Normal mode: the flash memories are able to work at the system clock frequency. In this case, all read accesses are executed with no wait state.
- High speed mode: the flash memories work at the system clock frequency divided by 2. In this case, non sequential read accesses require one wait state and sequential read accesses are executed with no wait state. Thanks to a cache buffer, fetch accesses at consecutive address are performed with no wait state.

The working mode is configured by access in the Mode Register (IFC_MR).

The user can change the working mode or the write access protection through the IFC_MR register.

Write access and flash erase are supported.

Interrupts are generated to indicate end of memory erase cycle, end of write cycle and denied access.

1.2 BLOCK DIAGRAM

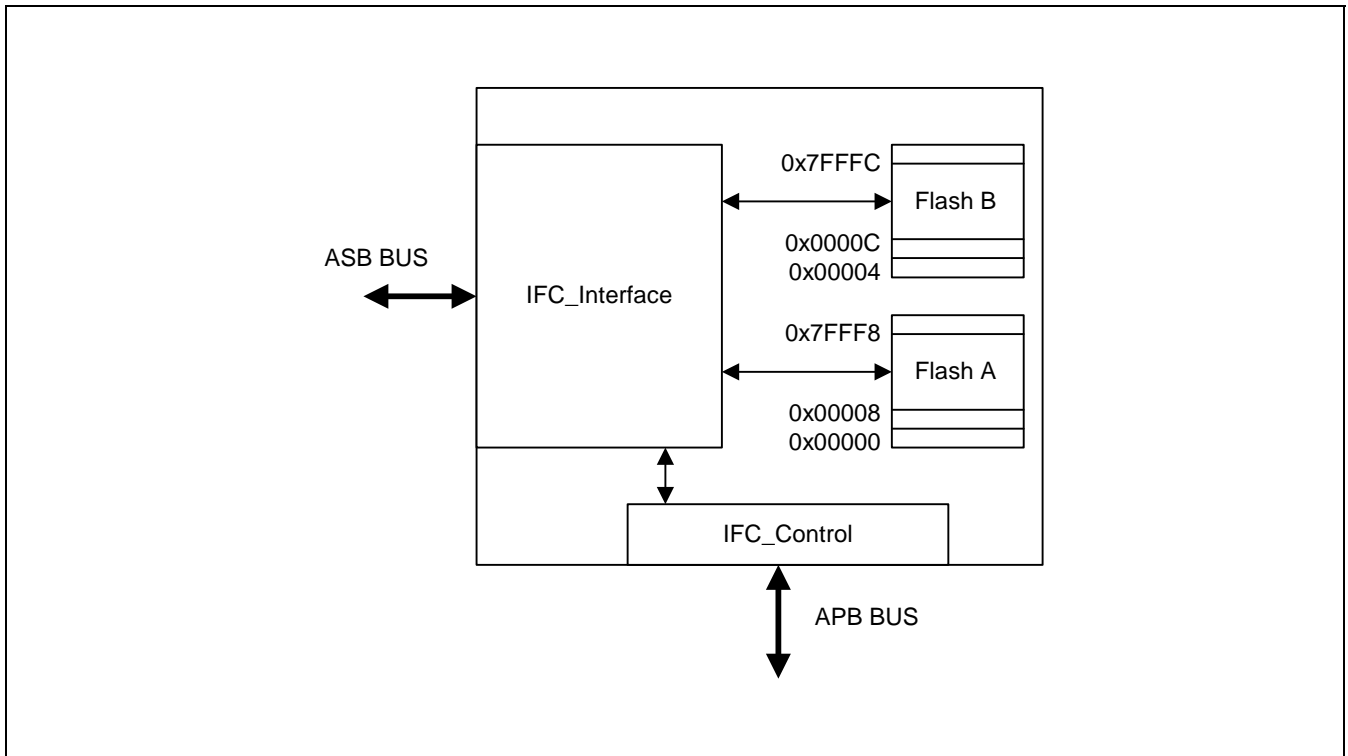


Figure 15-1. Interleaved Program Flash Memory Controller Block Diagram

2. FUNCTIONAL OPERATION

2.1 INTERLEAVED FLASH

2.1.1 General Description

The Interleaved Program Flash controller is designed to control synchronous flash. The memory is composed of 2 interleaved flash memories. This interleaved feature allows to anticipate read access, to reduce the number of wait states on the bus and to have a flash clock frequency equal to the system frequency divided by 2.

The IFC has 2 working modes: normal mode and high speed mode. The normal mode shall be used when the flash memory works at the system clock frequency. When the flash memory is not able to work at the system clock frequency, the high speed mode must be used.

2.1.2 Interleaved Memories

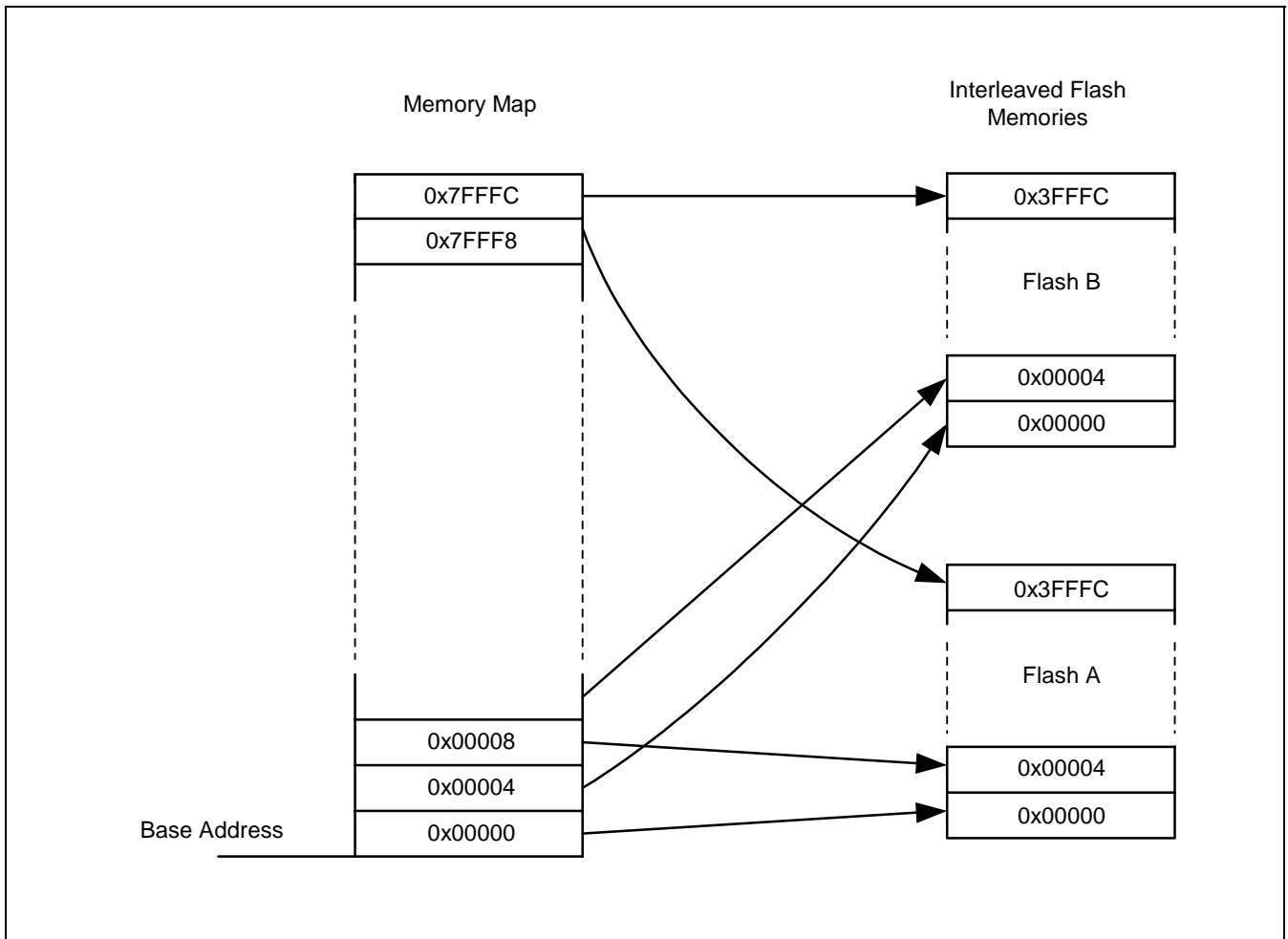


Figure 15-2. Interleaved Memory

2.1.3 Speed Mode

The speed mode selection (normal mode or high speed mode) depends on the system frequency.

Flash memory has a critical time to make data available during a read access. This time: $T_{dataout}$ limits the system frequency.

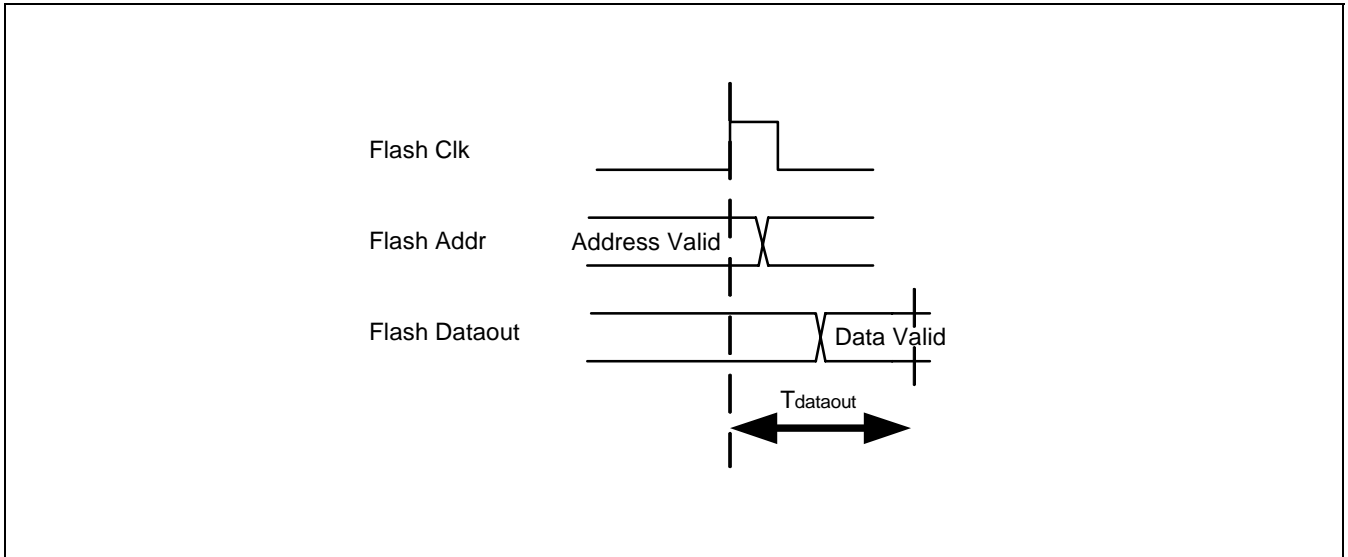


Figure 15-3. Flash Data Out Timing

The IFC can work in normal mode when the system frequency is lower than the maximum read flash frequency ($1 / T_{dataout}$). In this mode, each flash runs at the system clock.

The IFC must work in high speed mode when the system frequency is greater than the maximum read access flash frequency ($1 / T_{dataout}$). In this mode, each flash run at the system frequency divided by 2.

2.2 READ ACCESS

2.2.1 Read Access in Normal Mode

In normal mode, the flash is able to run at the system clock frequency. A read access takes one cycle of system clock. All read accesses (non sequential or sequential) are done without wait state.

2.2.2 Read Access in High Speed Mode

In high speed mode, the flash memories run at the system clock frequency divided by 2. A read access takes at least 2 cycles of system clock. A non sequential read access takes one wait state. All sequential read accesses are done without wait state. The sequential read accesses are anticipated.

2.2.2.1 Non Sequential Access

One non sequential access is done with one wait state. Typically, non sequential read accesses are done by the LDMA module or during a read data from the ARM7.

2.2.2.2 Sequential Access

During a sequential access, read accesses are consecutive. The first non sequential access takes one wait state and the following sequential accesses are done without wait state. Sequential accesses are only done by the ARM7 to fetch code and to load multiple data (load multiple instruction).

- Sequential 32.bits Read Access
- Sequential 16.bits Read Access

2.2.2.3 Consecutive Sequential Access

During a sequential accesses, read addresses are anticipated. At the end of sequential access, one access is done in excess. This access is stored in a cache buffer and is not erased while a new sequential read access has not done.

Thanks to this cache buffer, fetch accesses from ARM7 at consecutive addresses are without wait state.

2.3 ERASE CYCLE

Program Flash erase must be done in the data flash or SRAM only.

There are two types of erase cycle:

- Chip erase: the chip erase cycle erases the whole program flash memory. It is done when the CE bit is set to high in the IFC_CR (IFC Control Register). The chip erase takes typical 32ms. When the chip erase is finished, the IFC generates an end of erase interrupt.

Lower 16KB is the boot ROM area (0x0 – 0x3FFF, sector 0). Boot ROM area is not erased by chip erase operation.

- Sector erase: the flash memory is divided in several sectors of 16 Kbytes. The sector erase cycle erases one sector of flash memory. It is done by programming the sector number (SECTOR[4:0]) and setting to high the SE bit in the IFC_CR (IFC Control Register). The sector erase takes typical 32ms. When the sector erase cycle is finished, the IFC generates an end of erase interrupt.

During one (sector or chip) erase cycle, if a read or write access occurs, this access is not taken into account, a denied access interrupt is generated.

During one (sector or chip) erase cycle, if a new erase cycle is programmed, this erase cycle is not taken into account and a denied access interrupt is generated.

If erase (sector or chip) cycle is not allowed (bit WPR high in the IFC_MR Register) and an erase cycle is programmed, the erase cycle is ignored and an denied access interrupt is generated.

During one (sector or chip) erase cycle, the busy bit in the IFC_SR (IFC Status Register) is set to high and automatically cleared at the end of the erase cycle.

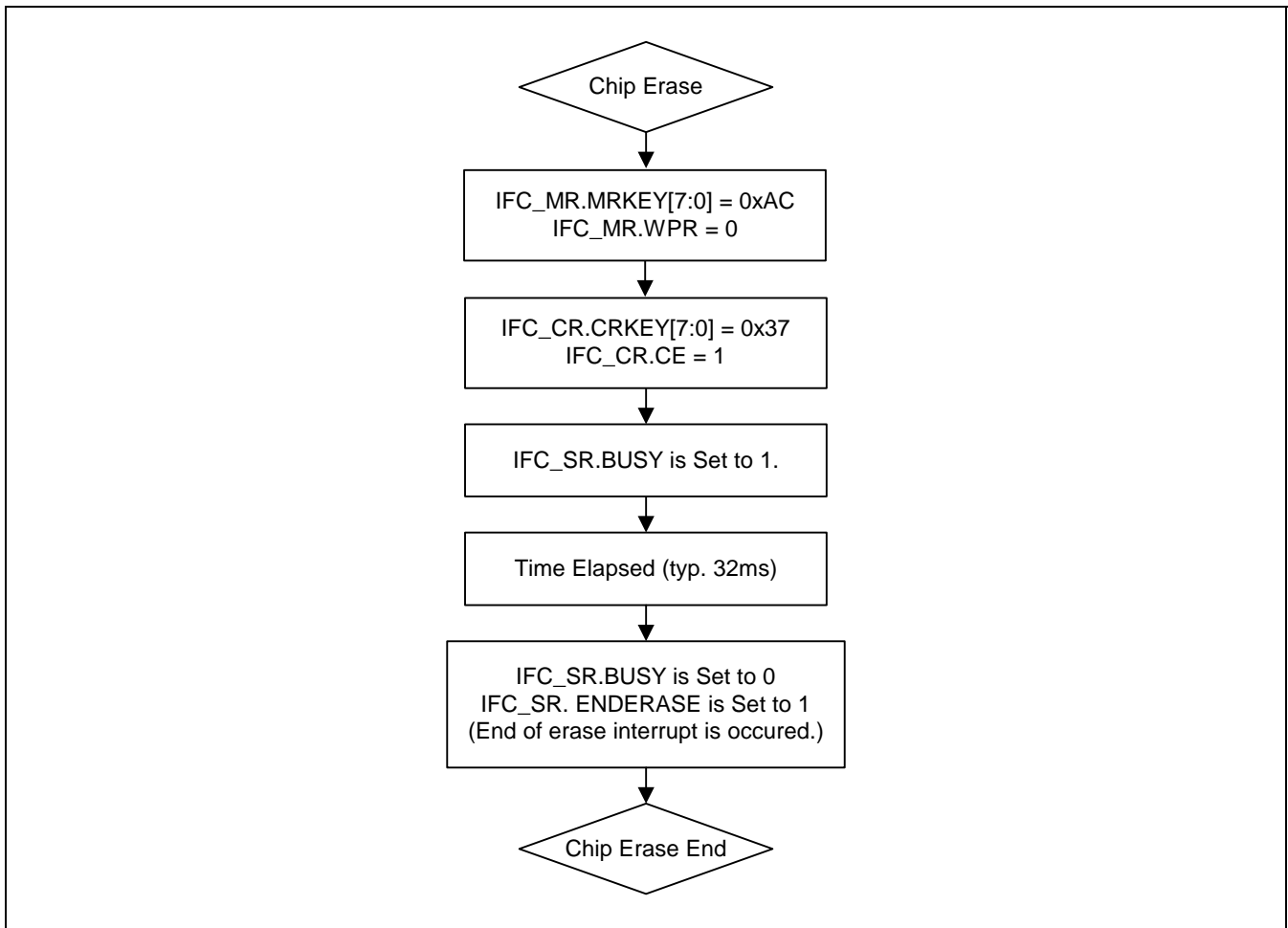


Figure 15-4. Chip Erase Flow Chart

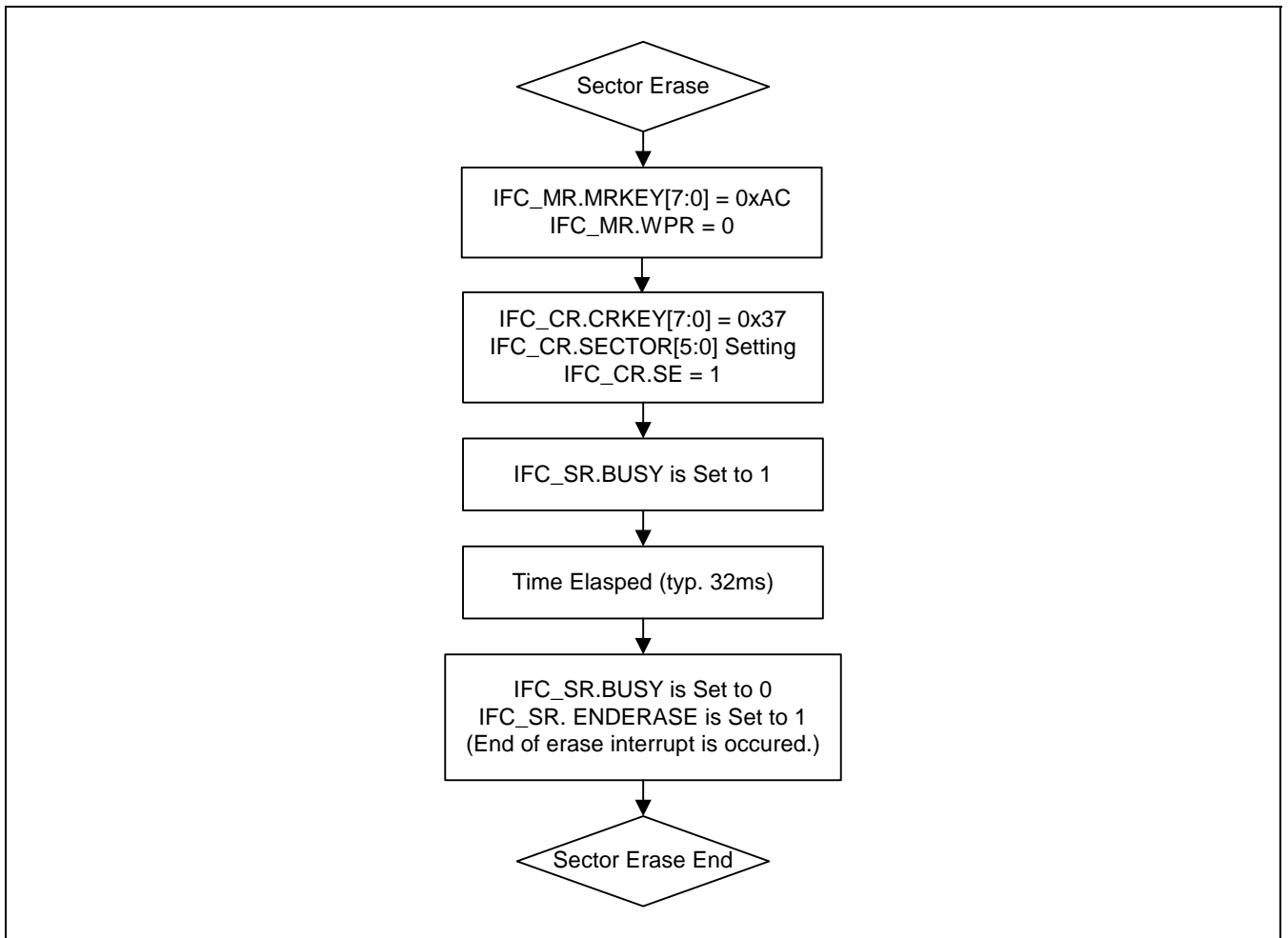


Figure 15-5. Sector Erase Flow Chart

2.4 WRITE CYCLE

Program Flash write must be done in the data Flash or SRAM only. A write cycle takes typical 41us. When the write access is done, an end of write interrupt is generated. All write accesses must be done in 32 bits; if a write access is done in 8 or 16 bits, this access is not performed.

A write cycle can only performed on a previously erased flash. If a write is requested on an address whose data has not been erased before, the write is not taken into account by the flash (data does not match 0xFFFFFFFF).

During a write access, if a new read or write access occurs, this new access is not taken into account, a denied interrupt is generated.

During a write cycle , if an erase cycle is programmed, this erase cycle is not taken into account and a denied access interrupt is generated.

When write cycles are not allowed (bit WPR high in the IFC_MR Register), if a write access occurs, a denied access interrupt is generated.

During a write access, the busy bit in the IFC_SR is set to high and automatically cleared at the end of the write cycle.

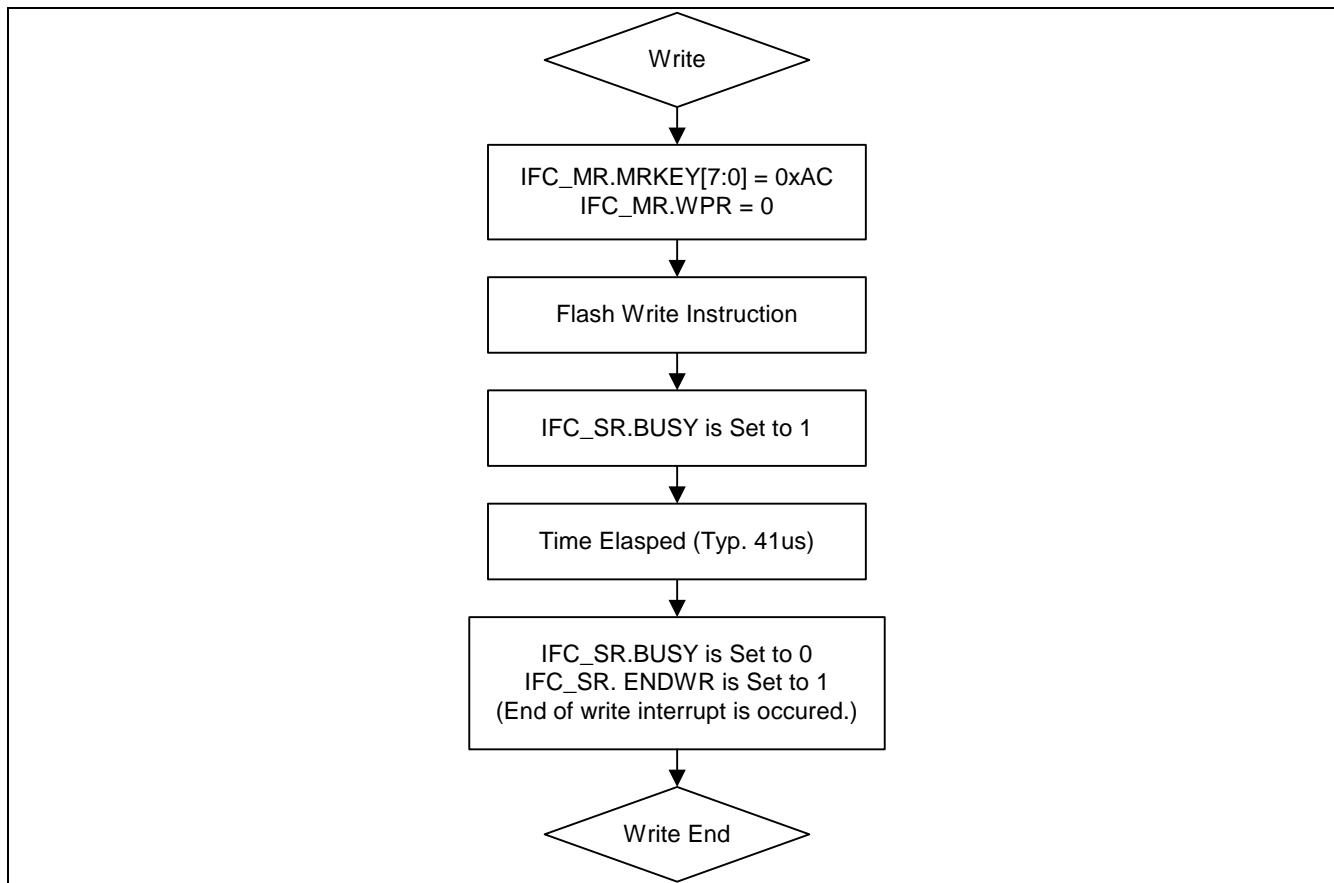


Figure 15-6. Write Flow Chart

2.5 INTERRUPTS GENERATION

There are 3 sources of interrupts:

- End of write cycle interrupt occurs once the write cycle has ended.
- End of erase cycle interrupt occurs once the chip or sector erase cycle has ended.
- Denied access interrupt occurs when:
 - The flash is busy (during an erase or write cycle) and a new read, write or erase cycle is attempted
 - A write access occurs while write access are not allowed (see write protection bit in the IFC_MR Register),
 - A write access occurs in 8–bits or 16–bits. (only 32–bits write access are allowed).

2.6 DATA ABORT GENERATION

Access is aborted when:

- The flash is busy (during an erase or write cycle) and a new access (read or write) is attempted.
- A write access is attempted while write access are not allowed (see write protection bit in the ASB_MR Register),
- A write access is attempted in 8–bits or 16–bits. (only 32–bits write access are allowed).

3. REGISTERS DESCRIPTION

Base Address – 0xFFE04000

Table 15-1. IFC Special Function Registers

Offset Address	Name	Description	R/W	Reset State
0x000 – 0x054	–	Reserved	–	–
0x058	IFC_PMSR	Power management status register	R	0XXXXXXXX0
0x05C	–	Reserved	–	–
0x060	IFC_CR	Controller register	W	0x00000000
0x064	IFC_MR	Mode register	R/W	0x00000080
0x068	–	Reserved	–	–
0x06C	IFC_CSR	Clear status register	W	0x00000000
0x070	IFC_SR	Status register	R	0x00000000
0x074	IFC_IER	Interrupt enable register	W	0x00000000
0x078	IFC_IDR	Interrupt disable register	W	0x00000000
0x07C	IFC_IMR	Interrupt mask register	R	0x00000000

IFC Power Management Status Register

IFC_PMSR (0x058)

Access: Read only

31	30	29	28	27	26	25	24
-		IPICODE[25:20]					
R-0	R-0	R-U	R-U	R-U	R-U	R-U	R-U
23	22	21	20	19	18	17	16
IPICODE[19:12]							
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U
15	14	13	12	11	10	9	8
IPICODE[11:4]							
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U
7	6	5	4	3	2	1	0
IPICODE[3:0]				-	-	-	-
R-U	R-U	R-U	R-U	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- IPICODE[25:0] : IP identifier code**

This field contains the version number of the module, coded on 26bits.

IFC Control Register **IFC_CR (0x060)** **Access: Write only**

31	30	29	28	27	26	25	24
–	SECTOR[4:0]					–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
CRKEY[7:0]							
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	CE	SE	–
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

• **SECTOR[4:0]: Sector Number**

Table 15-2. Sector Number

SECTOR[4:0]	Sector Number	Sector Size
0x00	0	16 Kbytes
0x01	1	16 Kbytes
0x02	2	16 Kbytes
...
0x1E	30	16 Kbytes
0x1F	31	16 Kbytes

• **CRKEY[7:0]: Key for the write access into the IFC_CR Register**

Any write in this register will only be effective if the CRKEY[7:0] is equal to 0x37

• **CE: Chip Erase**

0: No effect

1: Starts a chip erase (the whole memory is erased)

NOTE: During a chip erase, writing SECTOR[4:0] and SE shall have no effect

• **SE: Sector Erase**

0: No effect

1: Starts a sector erase. The number of the sector to erase must be programmed in SECTOR[4:0].

IFC Mode Register				IFC_MR (0x064)				Access: Read/Write							
31	30	29	28	27	26	25	24								
BA[7:0]															
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0								
23	22	21	20	19	18	17	16								
–	–	–	–	–	–	–	–								
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0								
15	14	13	12	11	10	9	8								
MRKEY[7:0]															
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0								
7	6	5	4	3	2	1	0								
WPR	–	–	STANDEN	–	SPEEDMODE	–	–								
R/W-1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0								

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **BA[7:0]: Interleave Flash controller Base Address**

The BA[7:0] contains the MSB of the IFC base address. BA[7:0] should be 0x00.

NOTE: BA[7:0] should be 0x00.

- **MRKEY[7:0]: Key for the write access into the IFC_MR Register**

Any write in this register will be only effective if the MRKEY[7:0] is equal at 0xAC

- **WPR: Write and Erase Protection**

0: Write and erase cycle are allowed in the program flash

1: Write and erase cycle are not allowed (only read access is allowed) in the program flash

NOTE: If the WPR = 1 and a write or erase access occurs, a denied access interrupt is generated.

- **STANDEN: Standby mode enable**

0: Standby mode is disabled.

1: Standby mode is enabled.

NOTE: If the STANDEN = 1 and a new access occurs, a denied access interrupt is generated.

- **SPEEDMODE: Speed mode**

0: Normal mode : can be used when the system clock frequency is lower than 20 MHz

1: High speed mode: must be used when the system clock frequency is greater than 20 MHz

15-1. IMPORTANT NOTICE

1. When switching from High Speed mode to normal, slow and low power mode, software has to ensure to wait the STABLE interrupt before clearing the SPEEDMODE bit.
2. When the SPEEDMODE bit is set to one, the speed mode feature of the FLASH controller is dynamically disabled during the LOWPOWER, SLOW or HALT microcontroller modes. (See clock manager chapter for more details on LOWPOWER, SLOW or HALT modes.)
3. SPEEDMODE bit must be set to 0 before stop mode entering because of current consumption.
4. At any time, if the system clock frequency is more than 20MHz and the SPEEDMODE bit is not set to one, the microcontroller may become in unknown state.

IFC Clear Status		IFC_CSR (0x06C)						Access: Write only	
31	30	29	28	27	26	25	24		
–	–	–	–	–	–	–	–		
W	W	W	W	W	W	W	W		W
23	22	21	20	19	18	17	16		
–	–	–	–	–	–	–	–		
W	W	W	W	W	W	W	W		W
15	14	13	12	11	10	9	8		
–	–	–	–	–	–	–	–		
W	W	W	W	W	W	W	W		W
7	6	5	4	3	2	1	0		
–	–	–	–	–	DACCESS	ENDERASE	ENDWR		
W	W	W	W	W	W	W	W		W

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **ENDWR: Clear end of write cycle**

0: No effect

1: Clears the ASB_IFC end of write status

- **ENDERASE: Clear end of erase cycle**

0: No effect

1: Clears the ASB_IFC end of erase cycle status

- **DACCESS: Clear denied access**

0: No effect

1: Clears the ASB_IFC denied access status

IFC Status Register **IFC_SR (0x070)** **Access: Read only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	BUSY
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
–	–	–	–	–	DACCESS	ENDERASE	ENDWR
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

● **ENDWR: End of write cycle**

0: No write cycle has been yet completed since last corresponding ENDWR bit was cleared in IFC_SR.

1: A write cycle has been completed

● **ENDERASE: End of chip or sector erase**

0: No erase cycle has been yet completed since last corresponding ENDERASE bit was cleared in IFC_SR

1: A erase cycle has been completed

● **DACCESS: Denied access**

0: No denied access has occurred in the last corresponding DACCESS bit was cleared in SR

1: An access has occurred during the flash controller was busy; or a write access occurs in 8 or 16 bits ; or write or erase cycle are attempted when the write protection is set.

● **BUSY: Flash busy (status bit)**

0: Flash controller is not busy

1: Flash controller is busy, no more read, write or erase accesses are possible until busy bit is high

IFC Interrupt Enable		IFC_IER (0x074)				Access: Write only	
31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	DACCESS	ENDERASE	ENDWR
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **ENDWR: End of Write Interrupt Enable**

0: No effect

1: Enable the ENDWR interrupt

- **ENDERASE: End of Erase Interrupt Enable**

0: No effect

1: Enable the ENDERASE interrupt

- **DACCESS: Denied Access Interrupt Enable**

0: No effect

1: Enable the DACCESS interrupt

IFC Interrupt Disable		IFC_IDR (0x078)				Access: Write only	
31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	DACCESS	ENDERASE	ENDWR
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **ENDWR: End of write interrupt disable**

0: No effect

1: Disable the ENDWR interrupt

- **ENDERASE: End of erase interrupt disable**

0: No effect

1: Disable the ENDERASE interrupt

- **DACCESS: Denied access interrupt disable**

0: No effect

1: Disable the DACCESS interrupt

IFC Interrupt Mask

IFC_IMR (0x07C)

Access: Read only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
–	–	–	–	–	DACCESS	ENDERASE	ENDWR
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **ENDWR: End of write interrupt mask**

0: The end of write interrupt is disabled

1: The end of write interrupt is enabled

- **ENDERASE: End of erase interrupt mask**

0: The end of erase interrupt is disabled

1: The end of erase interrupt is enabled

- **DACCESS: Denied access interrupt mask**

0: The denied access interrupt is disabled

1: The denied access interrupt is enabled

16

INTERNAL RAM CONTROLLER (IRC)

1. OVERVIEW

1.1 FUNCTIONAL DESCRIPTION

The IRC allows to read and write in the internal memory. It is connected to the ASB bus and it is programmable by the APB bus.

The main features are :

- 16 Kbytes of SRAM
- 32 bits memory with byte, half word and word accesses
- Base Address programmable
- Access with zero wait states

2. REGISTERS DESCRIPTION

Base Address – 0xFFFF0000

Table 16-1. IRC Special Function Registers

Offset Address	Name	Description	R/W	Reset State
0x000	IRC_MR	IRC Mode Register	R/W	0x00300000

IRC Mode Register **IRC_MR (0x000)** **Access: Read/Write**

31	30	29	28	27	26	25	24
BA[11:4]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
BA[3:0]				-	-	-	-
R/W-0	R/W-0	R/W-1	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **BA[11:0] : Base Address**

Base address where internal SRAM must be accessed.

Base address can be changed to other memory space.

Be careful that abort doesn't occur by overlapping other valid memory space.

17

LCD CONTROLLER (LCDC)

1. OVERVIEW

1.1 FUNCTIONAL DESCRIPTION

The LCD Controller module has 4 common drivers and 30 segment drivers, so it can control up to 120 LCD segments.

Each LCD segment is configured by a corresponding bit in the LCD display memory. 4 duty modes (static, 1/2, 1/3, 1/4) and 3 bias (static, 1/2, 1/3) modes are provided. LCD bias voltage can be supplied from either internal VDD or external bias pin. Furthermore, the LCD Controller module has internal voltage dividing circuit which provides 3 different voltage levels.

- Supports 4 x 30 LCD segments (total 120 LCD segments panel)
- Four common output pins COM[3:0]
- Forty segments output pins SEG[29:0]
- Provides one external LCD bias pin (VLCD)
- Supports 3 bias and 4 duty operating modes
- Equips internal voltage dividing registers
- LCD display memory (32-bit x 4) for display information

2. EXTERNAL PIN DESCRIPTION

Table 17-1. LCD Pin Description

Pin Name	Function	I/O Type	Active Level	Comments
VLCD	External LCD bias pin	I	–	–
COM[3:0]	Appropriate voltage level for COM signal	O	–	–
SEG[29:0]	Appropriate voltage level for SEG signal	O	–	–

17-1. IMPORTANT NOTICE

The I/O pin configuration has to be done corresponding pins used read LCD signals.

If used the 3 COMS and 20 SEGS, Pin 34, Pin 35, Pin 36 and Pin 38 ~ Pin 59 should be configured as COM 0, COM 1, COM 2 and SEG 0 ~ 19.

That means COM 3 and SEG 20 ~ 29 should to be configured as normal I/O Pin or the other function.

3. FUNCTIONAL OPERATION

3.1 LCD CONTROLLER

3.1.1 Introduction

The LCD controller is designed to support up to 120 segments LCD panel.

Many functions as LCD Display RAM, Clock Generator, Timing Controller, COM Driver, SEG Driver and Bias Divider will allow the user to configure the controller for a specific application.

3.1.2 Block Diagram

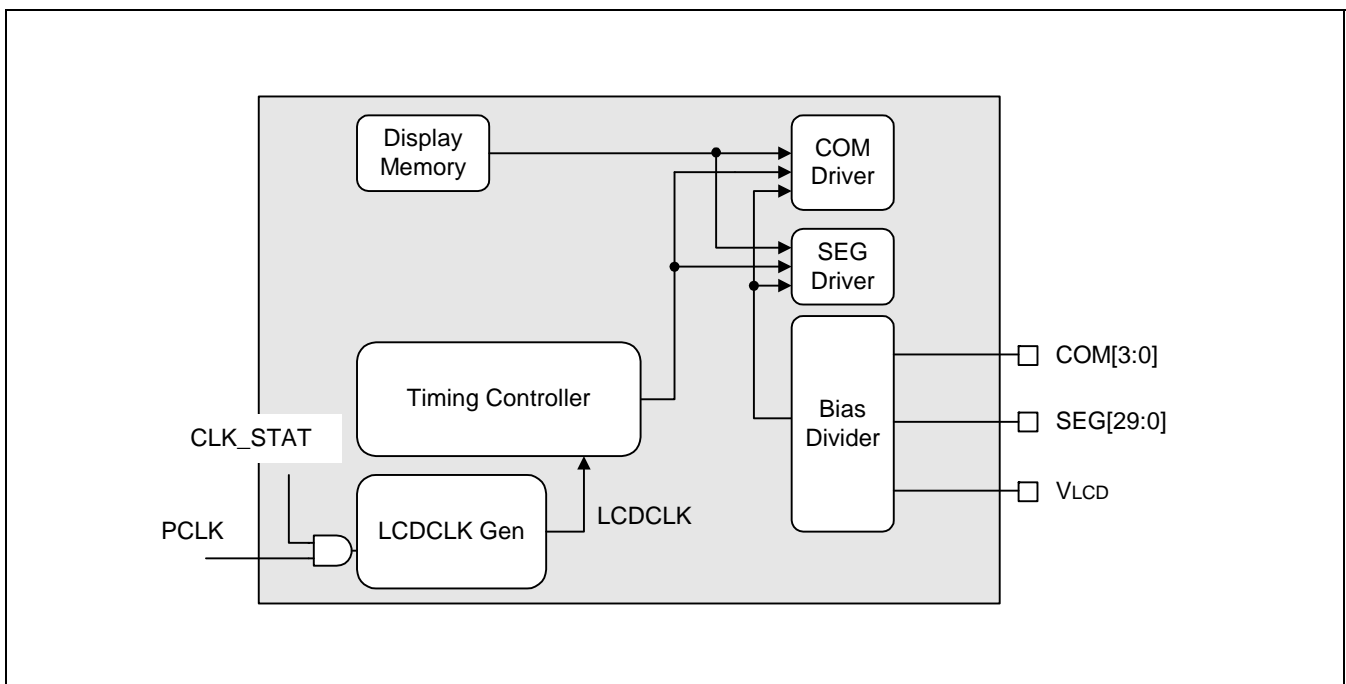


Figure 17-1. LCD Controller Block Diagram

3.1.2.1 LCD Display RAM

The LCD Display RAM consists of five 32-bit registers and indicates which LCD segment is in on or off state.

Logical 1 means the corresponding segment is on state and logical 0 means vice versa.

3.1.2.2 LCD Clock Generator

The LCD controller needs 1 kHz clock signal for proper operation. This module generates the 1 kHz clock signal from the PCLK regardless of the frequency of the system clock.

It is necessary to set the appropriate value to the LCD clock scaling register (LCD_SCR).

3.1.2.3 Timing Controller

This module generates pre-selection signal according to the pre-defined frame frequency and the bias-duty operating mode.

3.1.2.4 COM Driver

This module generates the control signals for COM[3:0] signals based on the pre-defined frame frequency and the bias-duty operating mode.

3.1.2.5 SEG Driver

This module generates the SEG[29:0] signals based on the pre-defined frame frequency, the bias-duty operating mode and the contents of the LCD display RAM.

3.1.2.6 LCD Bias Divider

This module provides three voltage levels (IVLC0, IVLC1 and IVLC2) by using the internal voltage dividing circuit.

External VDD is used as an LCD bias in default. However, the LCD bias can be supplied from the internal VLCD pin. (LCD_SR.VLCD_SEL == 1)

Furthermore, the voltage dividing circuit can be deactivated.

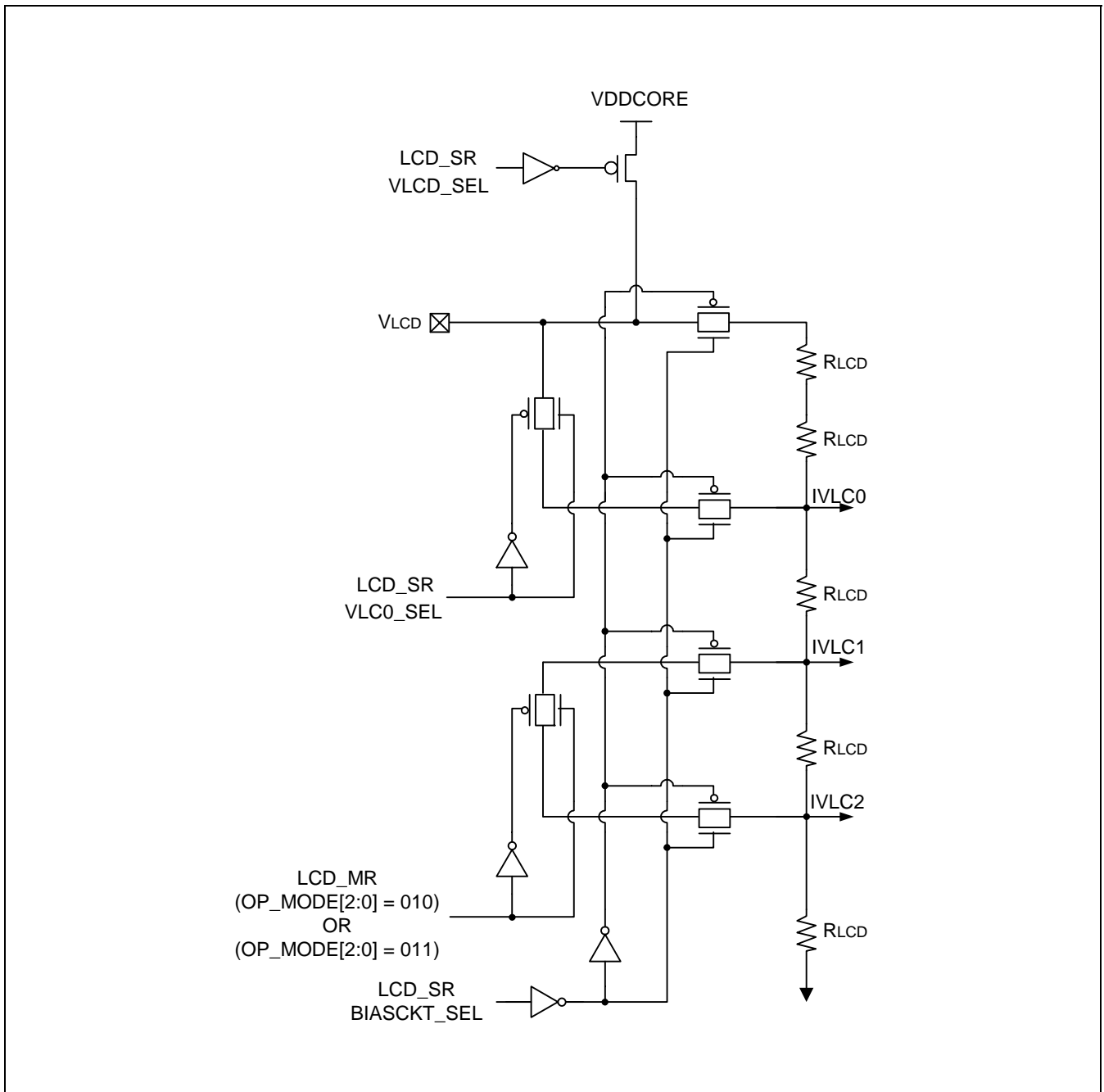


Figure 17-2. LCD Bias Driver

4. TIMINGS

4.1 STATIC MODE

The AC timing diagram of COM/SEG signals under static mode is described below:

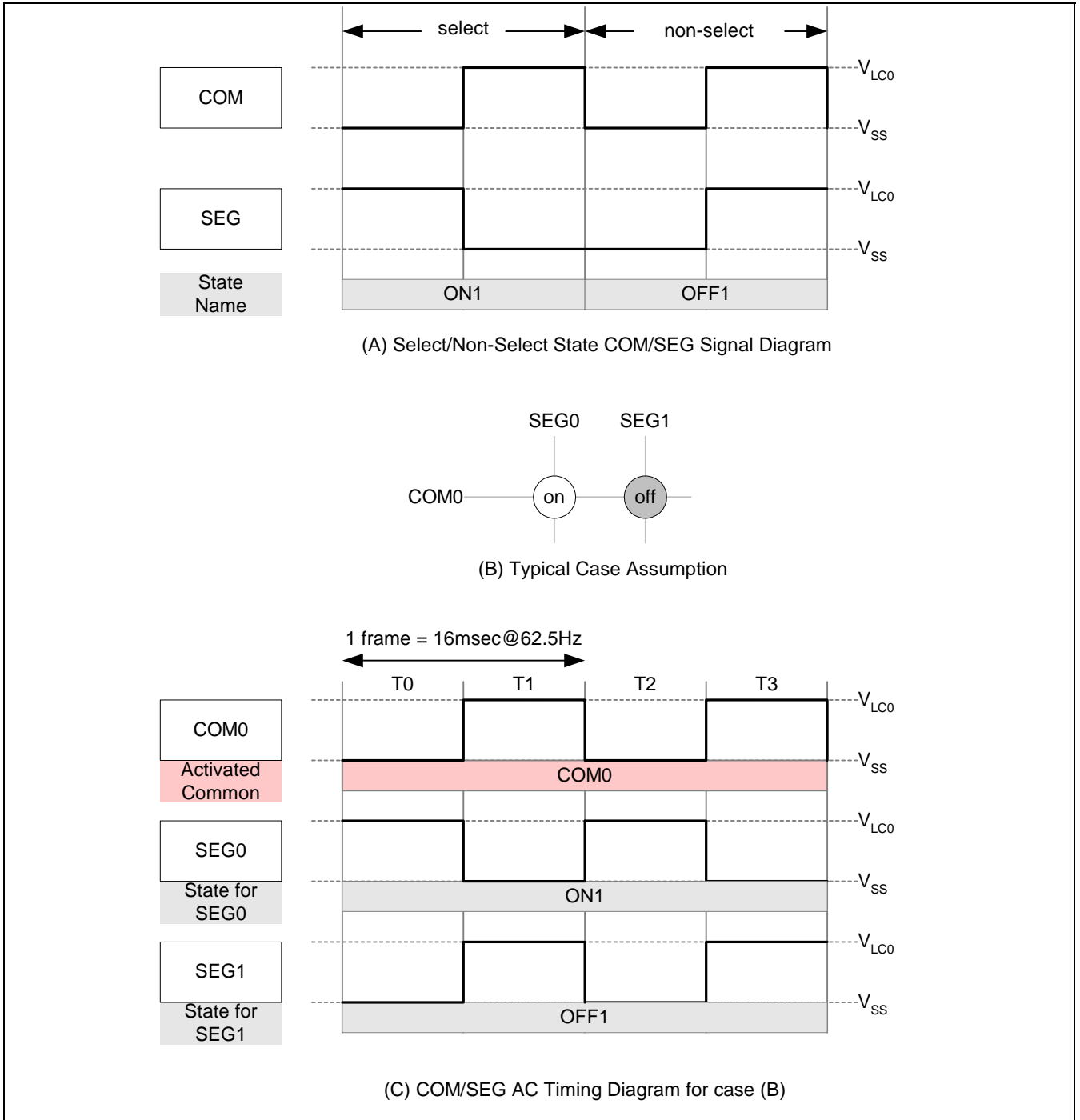
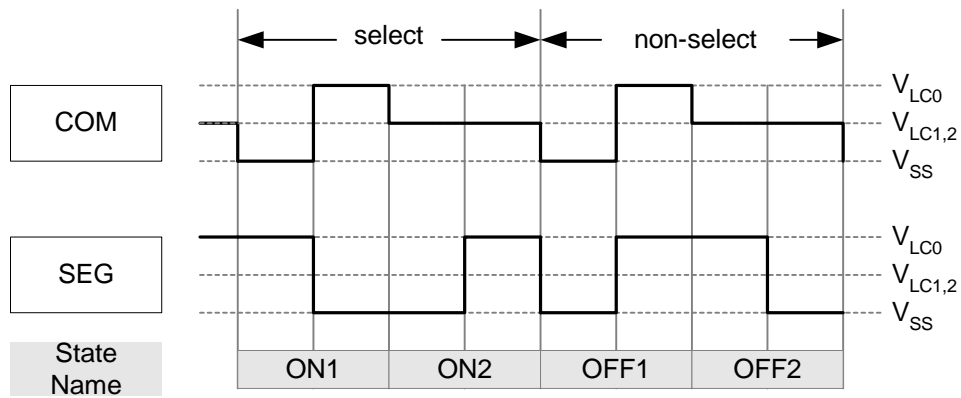


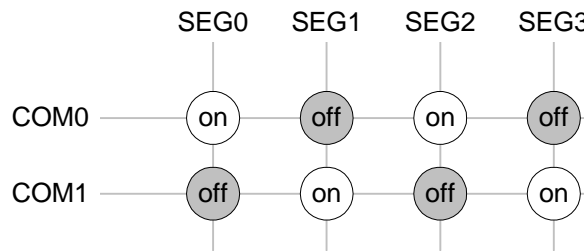
Figure 17-3. COM/SEG Signals Under Static Mode

4.2 1/2 DUTY, 1/2 BIAS MODE

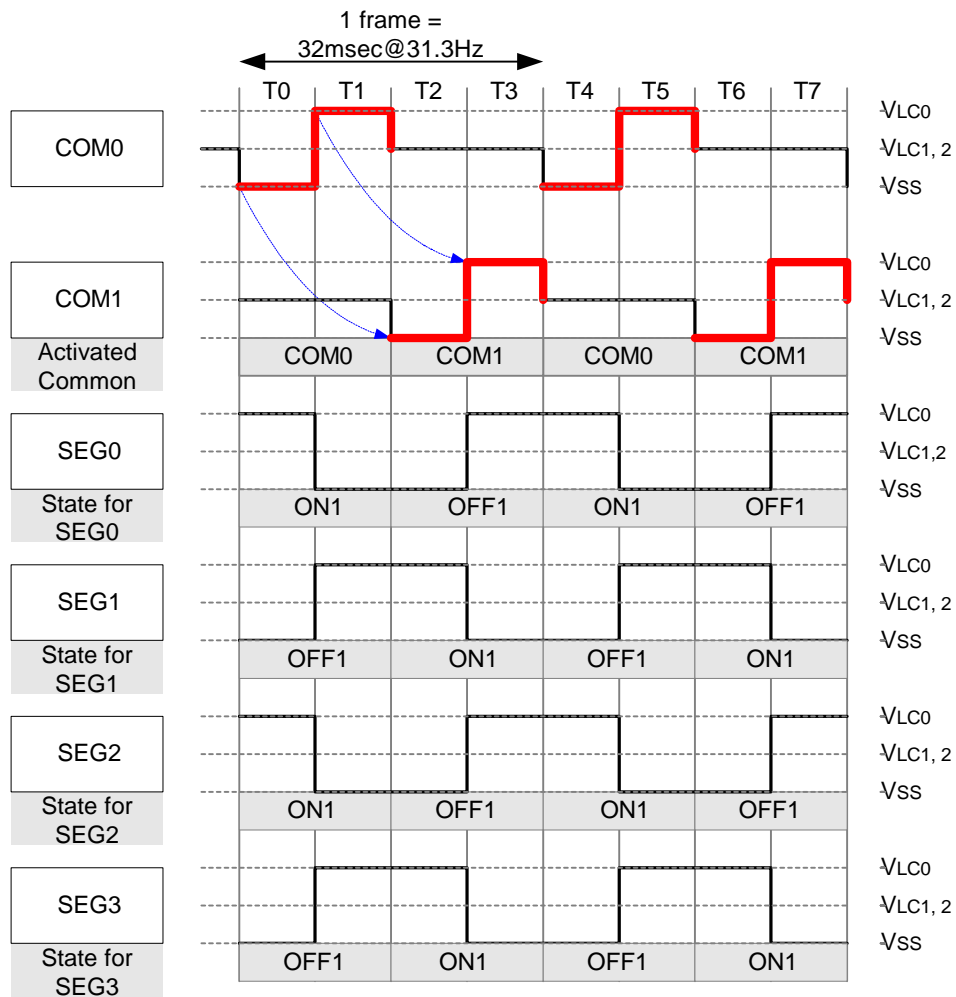
The AC Timing diagram of COM/SEG signals under 1/2 duty 1/2 bias mode is described below:



(A) Select/Non-Select State COM/SEG Signal Diagram



(B) Typical case Assumption

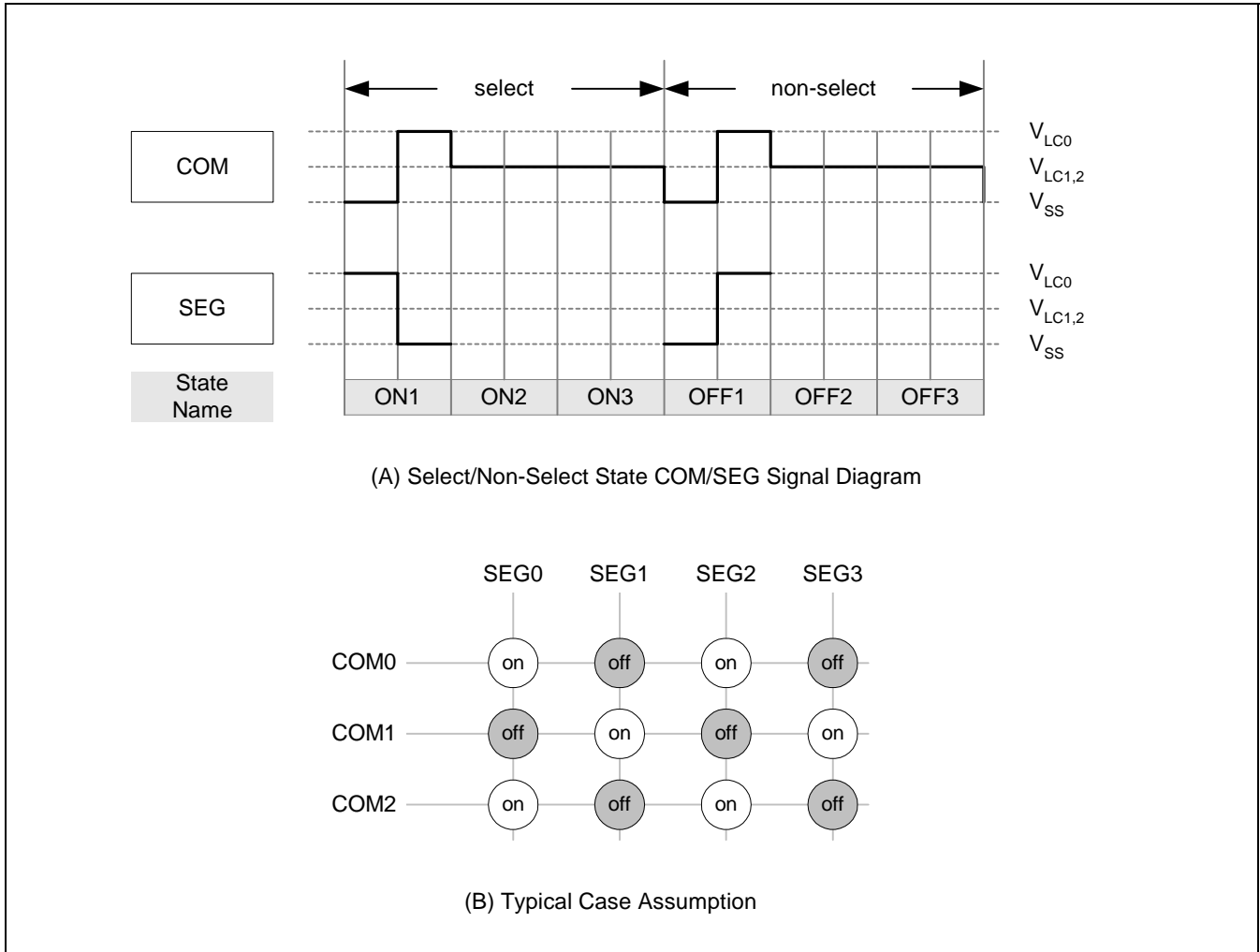


(C) COM/SEG AC Timing Diagram for Case (B)

Figure 17-4. COM/SEG Signals Under 1/2 Duty 1/2 Bias Mode

4.3 1/3 DUTY, 1/2 BIAS MODE

The AC timing diagram of COM/SEG signals under 1/3 duty 1/2 bias mode is described in Figure 17-5.



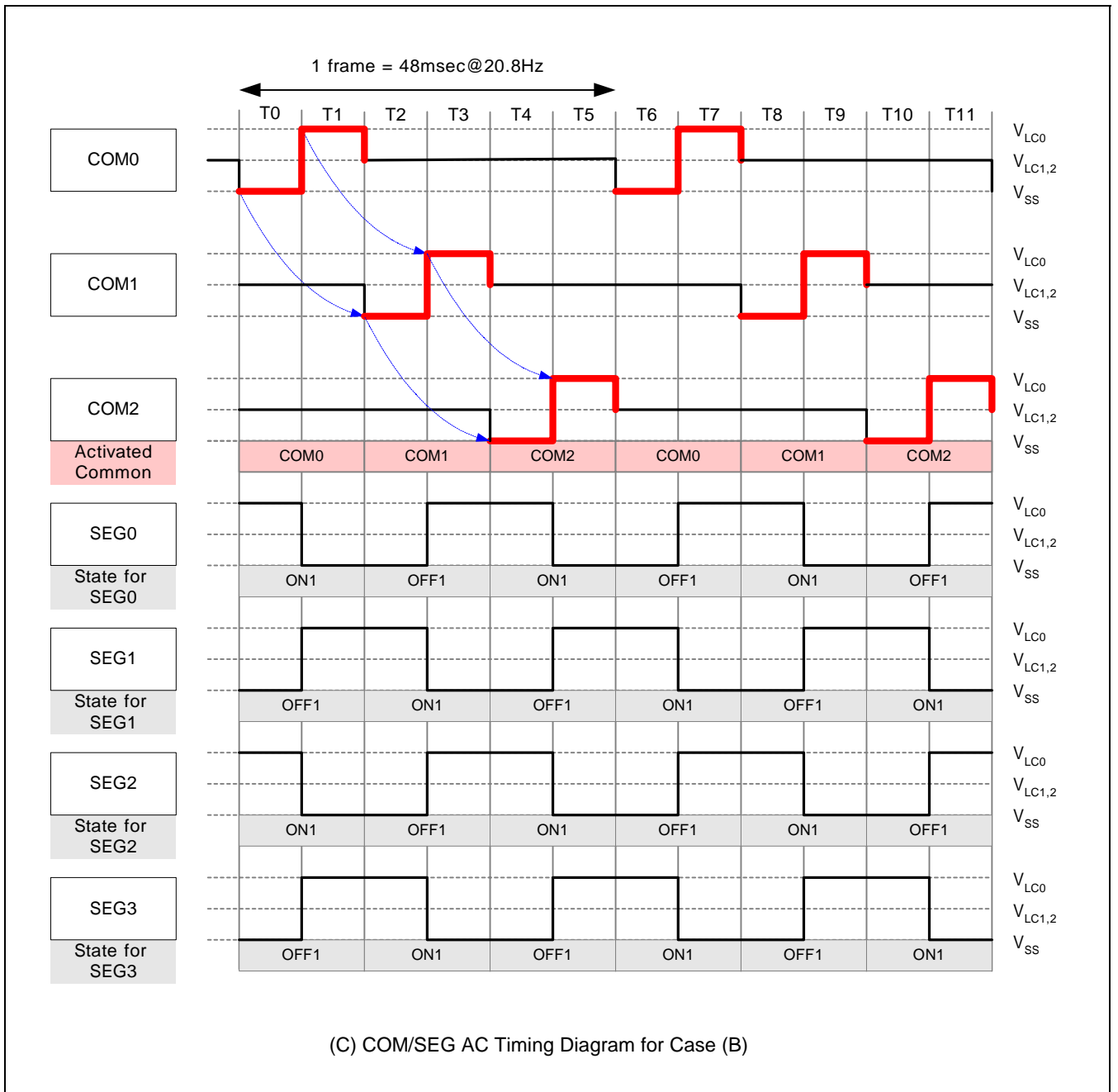
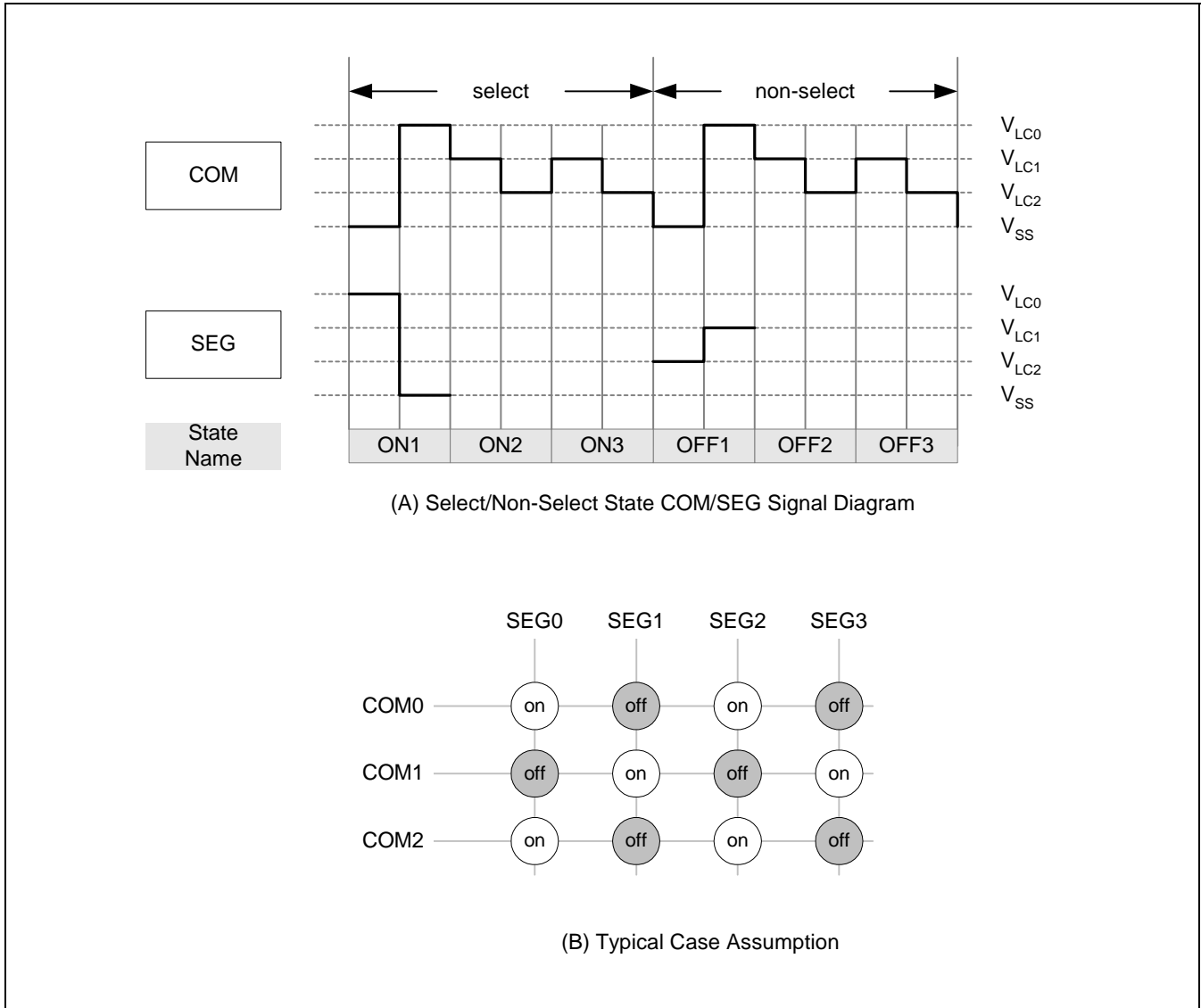


Figure 17-5. COM/SEG Signals Under 1/3 Duty 1/2 Bias Mode

4.4 1/3 DUTY, 1/3 BIAS MODE

The AC timing diagram of COM/SEG signals under 1/3 duty 1/3 bias mode is described in Figure 17-6.



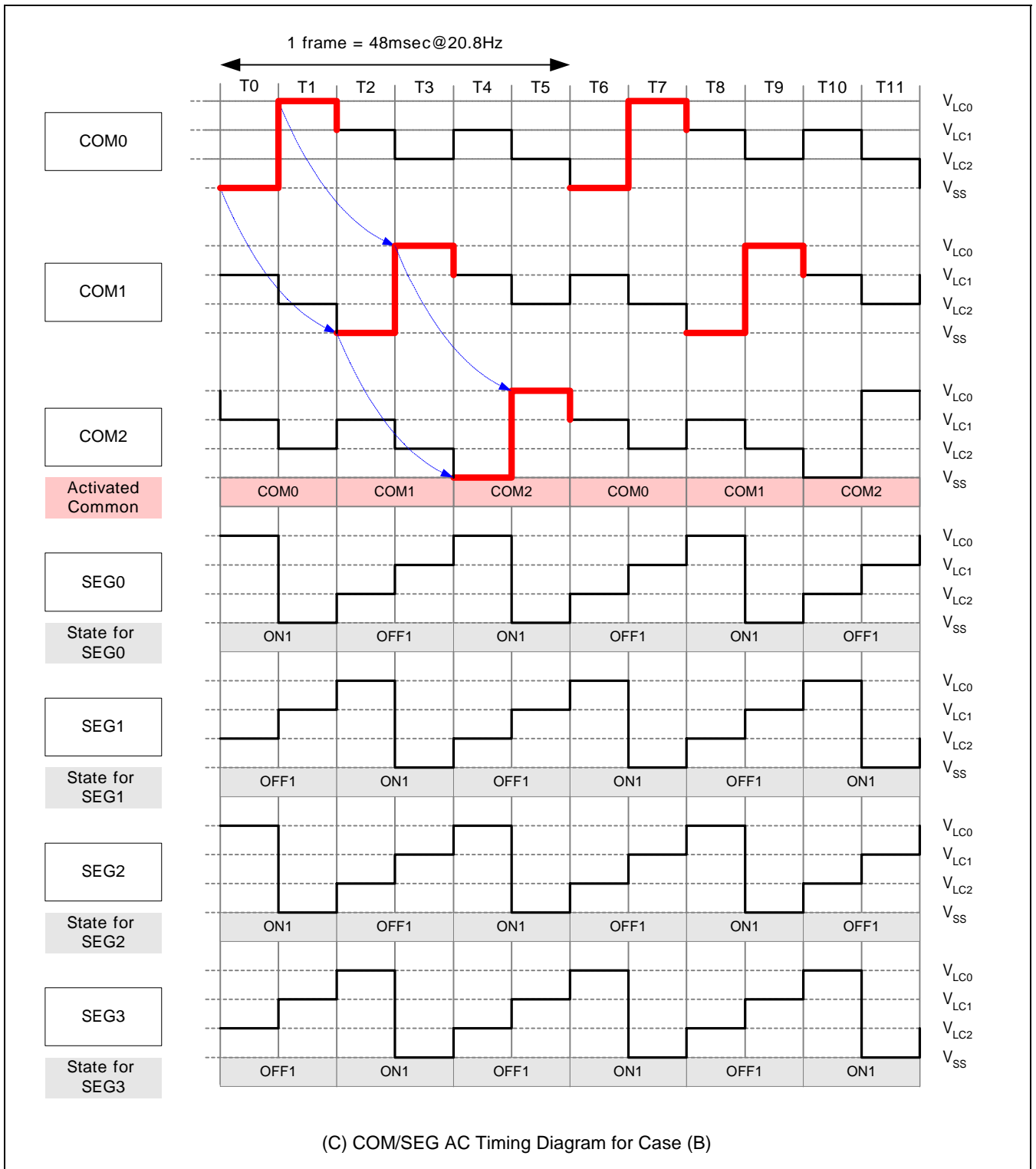
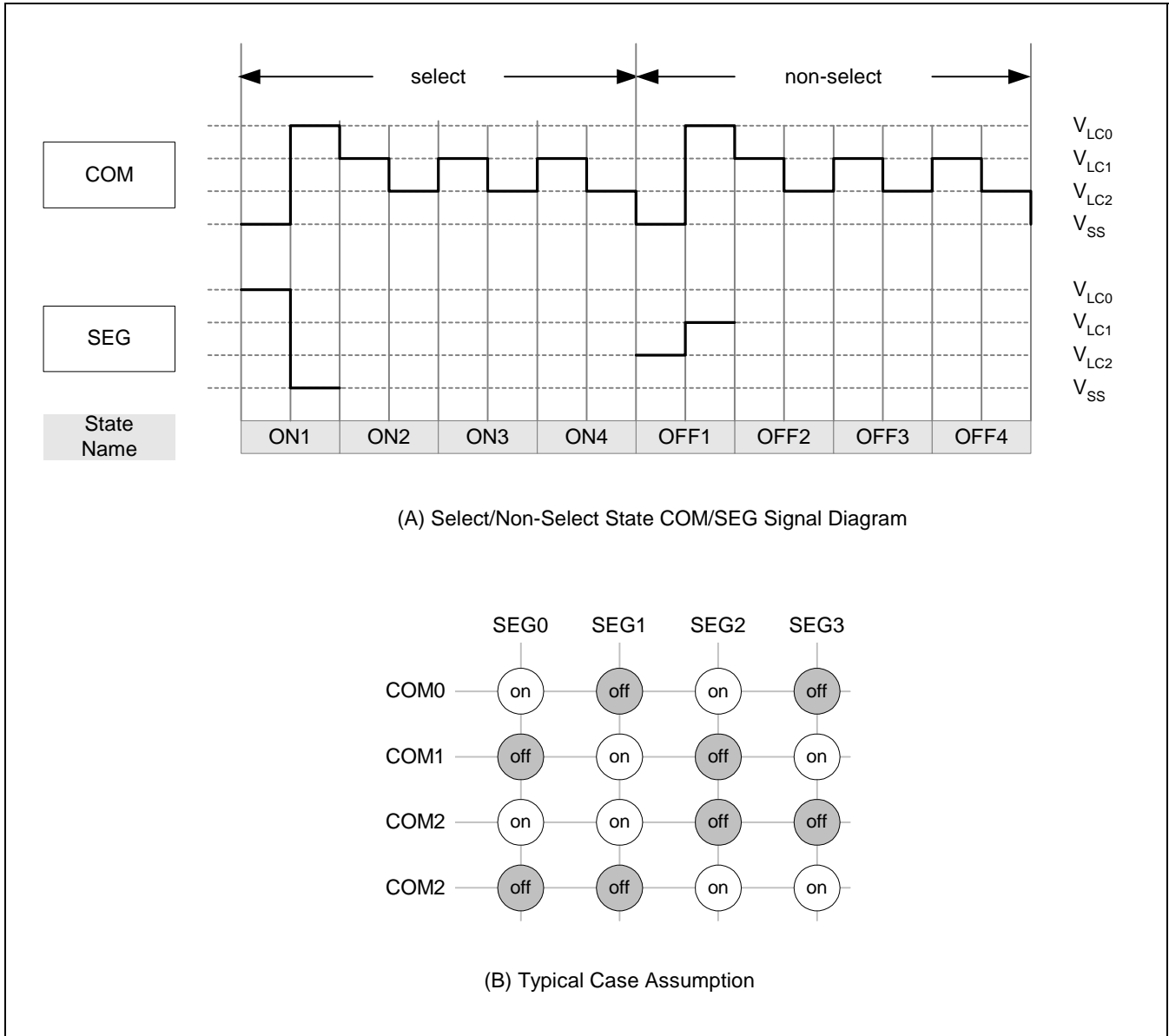


Figure 17-6. COM/SEG Signals Under 1/3 Duty 1/3 Bias Mode

4.5. 1/4 DUTY, 1/3 BIAS MODE

The AC timing diagram of COM/SEG signals under 1/4 duty 1/3 bias mode is described in Figure 17-7.



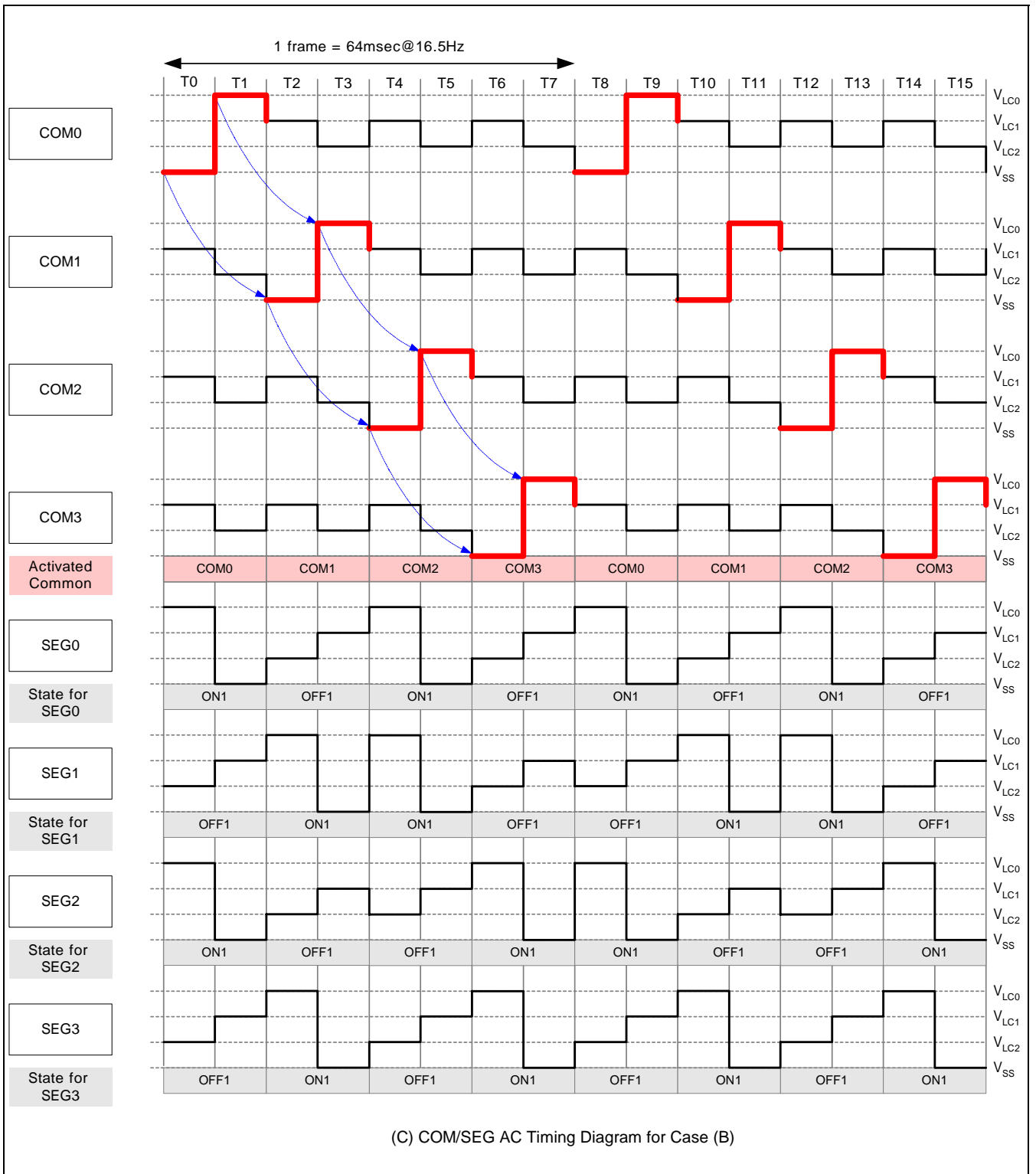


Figure 17-7. COM/SEG Signals Under 1/4 Duty 1/3 Bias Mode

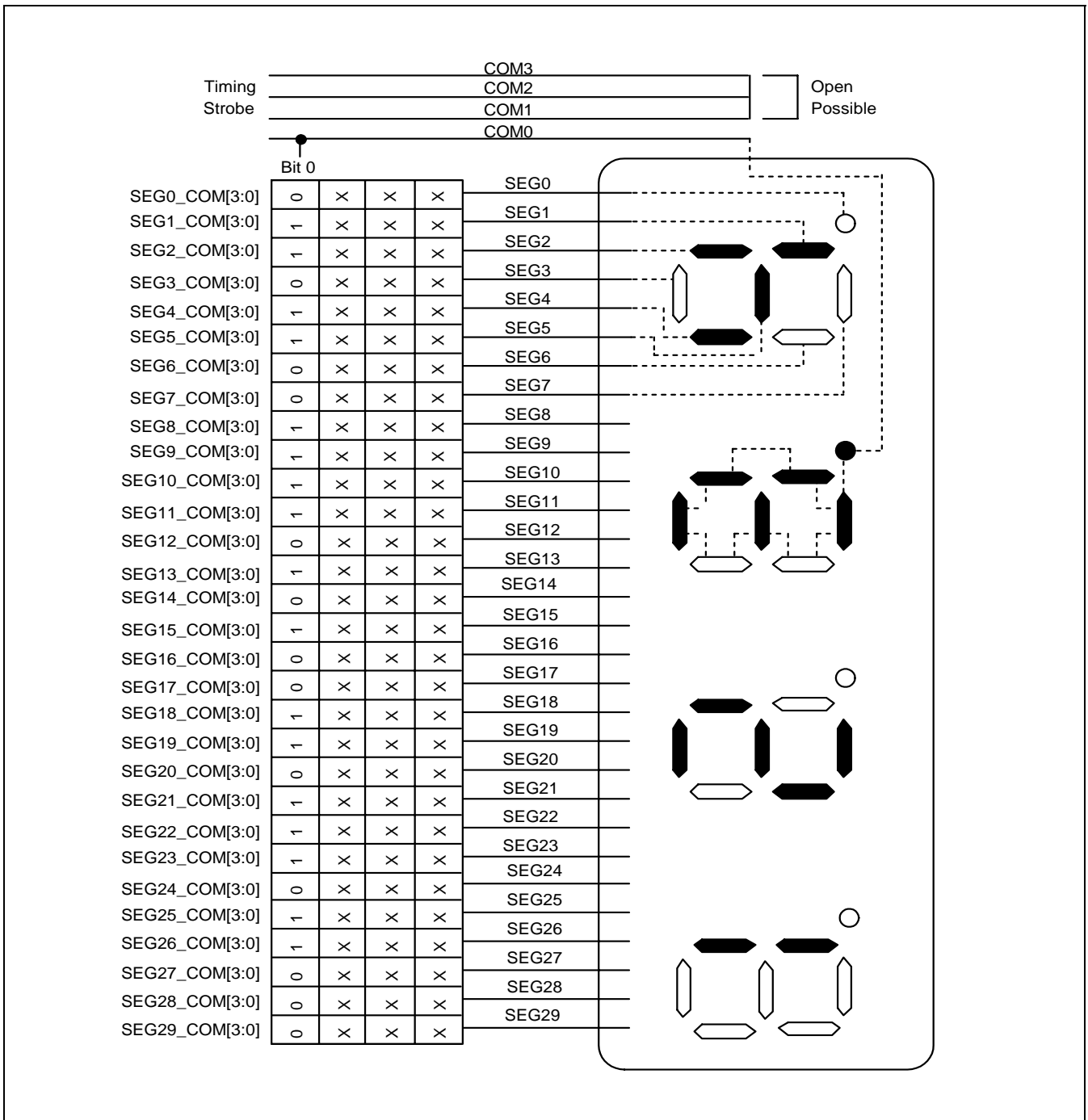


Figure 17-8. LCD connection example in static mode

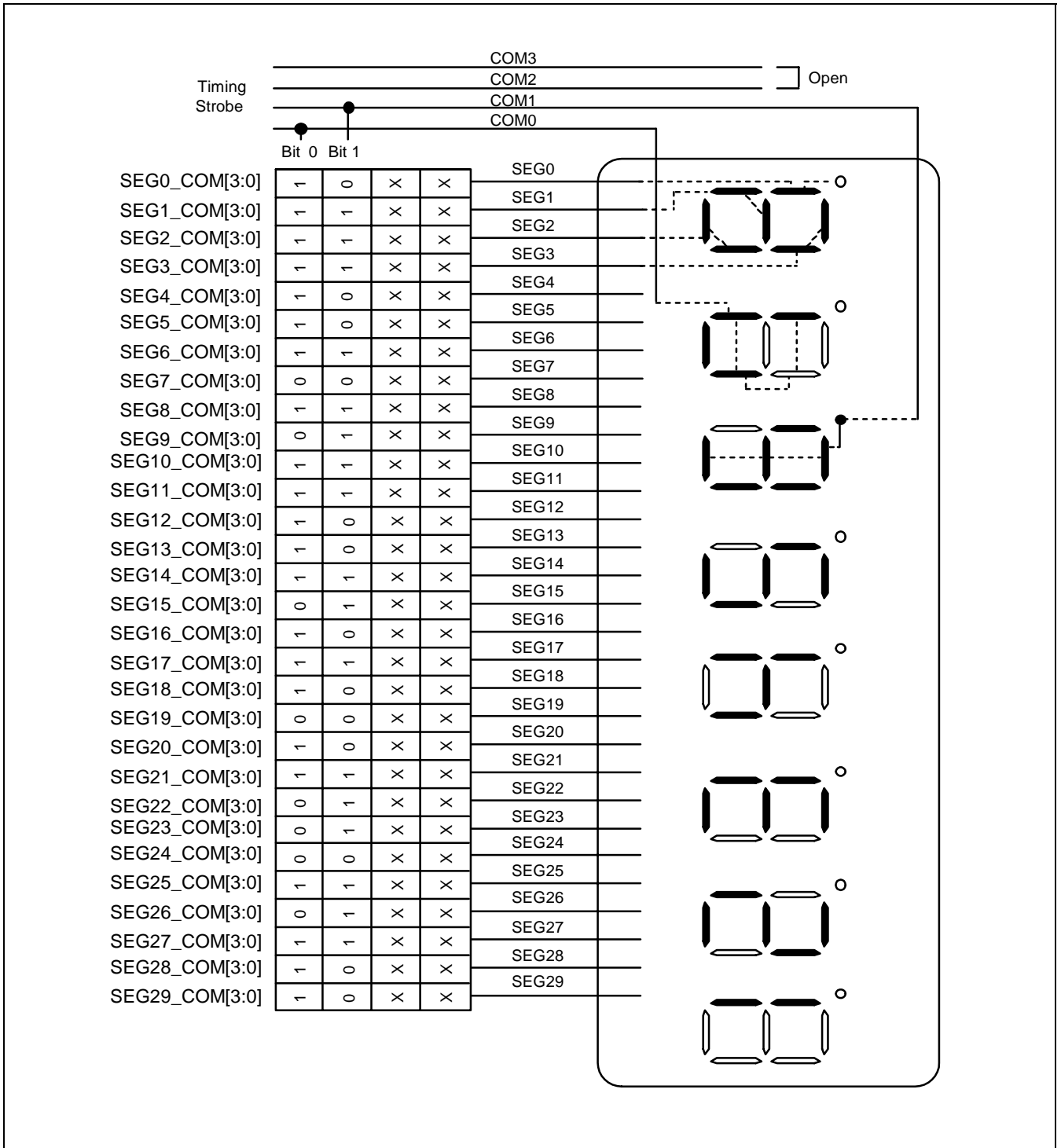


Figure 17-9. LCD connection example at 1/2 Duty, 1/2 Bias

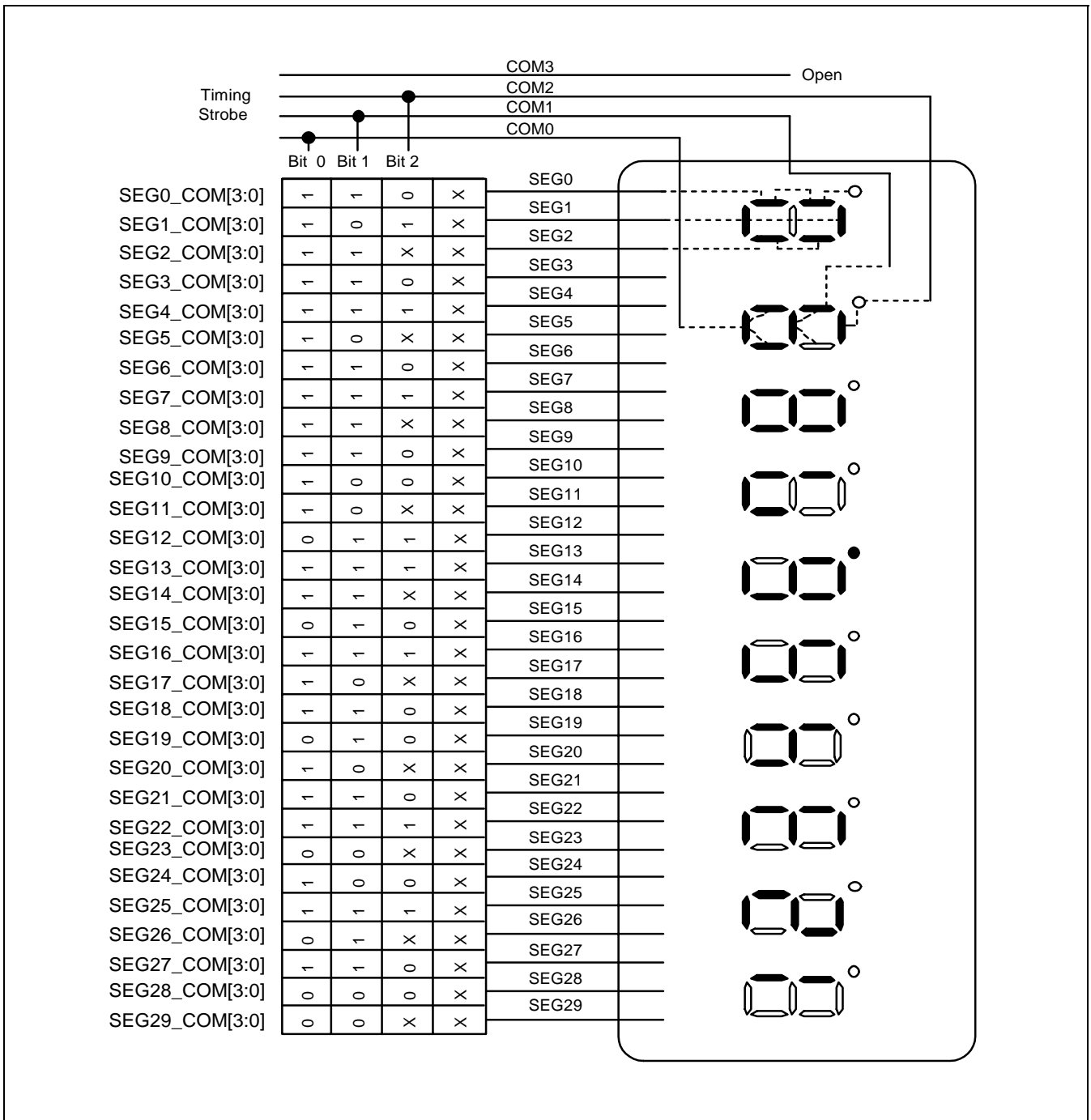


Figure 17-10. LCD connection example at 1/3 Duty, 1/2 Bias

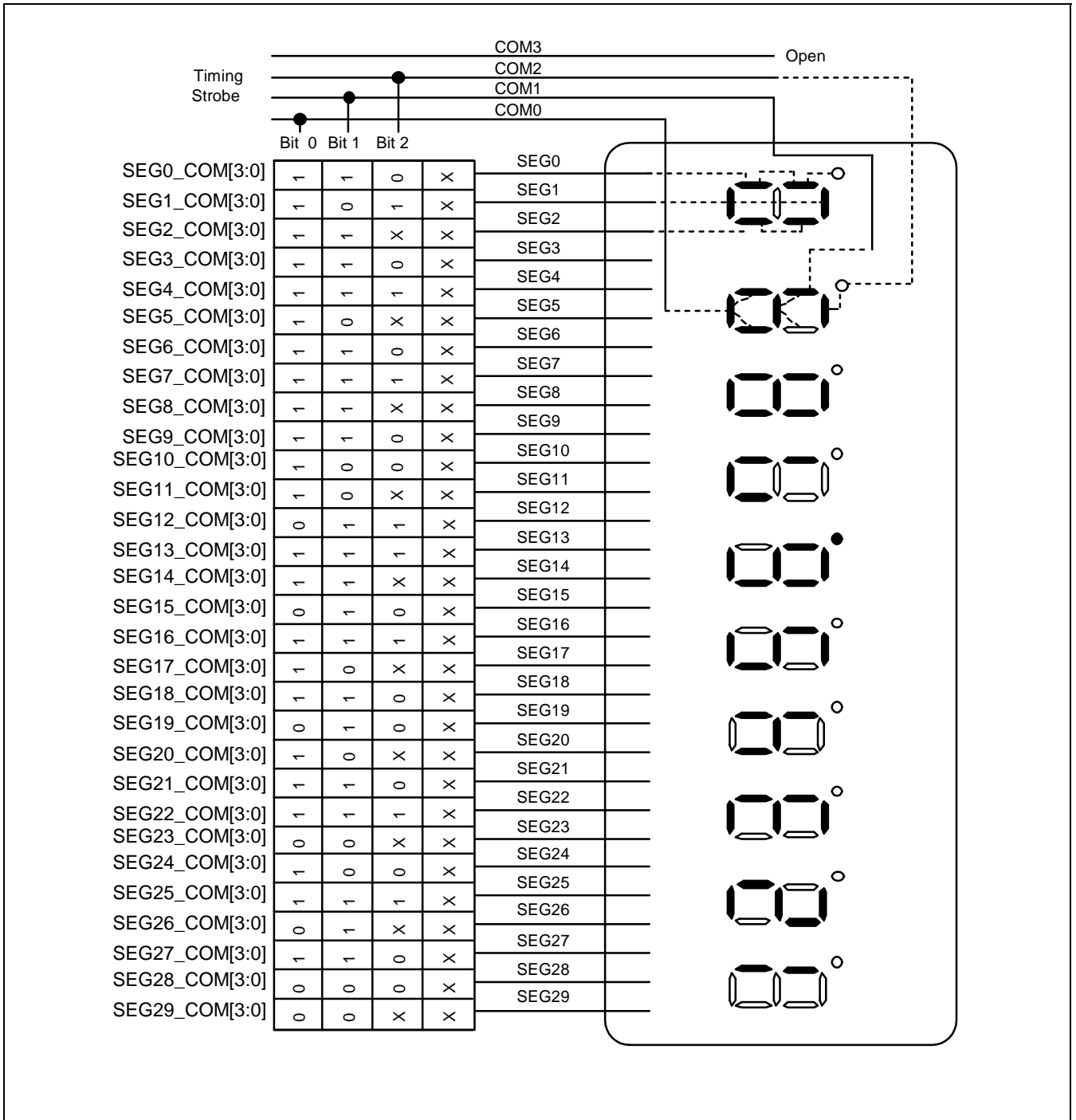


Figure 17-11. LCD connection example at 1/3 Duty, 1/3 Bias

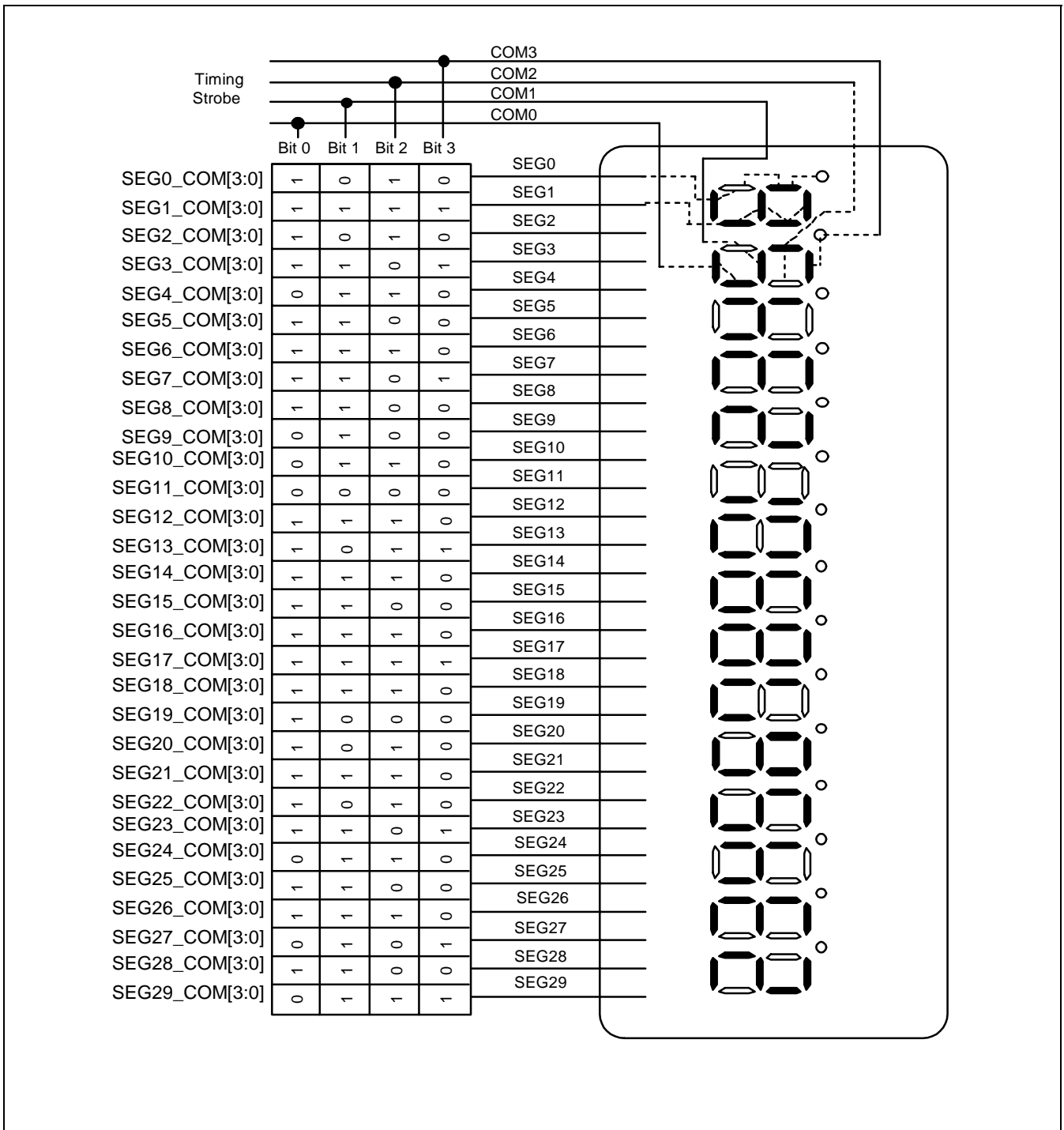


Figure 17-12. LCD connection example at 1/4 Duty, 1/3 Bias

5. REGISTERS DESCRIPTION

Base Address – 0xFFE58000

Table 17-2. LCDC Special Function Registers

Offset Address	Name	Description	R/W	Reset State
0x0000 – 0x004C	Reserved	–	–	–
0x0050	LCD_ECR	Enable clock register	W	–
0x0054	LCD_DCR	Disable clock register	W	–
0x0058	LCD_PMSR	Power management status register	R	0x00000000
0x005C	Reserved	–	–	–
0x0060	LCD_CR	Control register	W	–
0x0064	LCD_MR	Mode register	R/W	0x00000000
0x0068 – 0x006C	Reserved	–	–	–
0x0070	LCD_SR	Status register	R	0x00000000
0x0074 – 0x007C	Reserved	–	–	–
0x0080	LCD_CLKDIVR	Clock dividing register	R/W	0x00009C3F
0x0084 – 0x008C	Reserved	–	–	–
0x0090	LCD_DM0	LCD display memory 0	R/W	0x00000000
0x0094	LCD_DM1	LCD display memory 1	R/W	0x00000000
0x0098	LCD_DM2	LCD display memory 2	R/W	0x00000000
0x009C	LCD_DM3	LCD display memory 3	R/W	0x00000000
–	Reserved	–	–	–

LCD Enable Clock Register

LCD_ECR (0x050)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	LCD	–
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **LCD : LCD Clock enable**

0: No effect

1: Enable LCD Clock

LCD Disable Clock Register

LCD_DCR (0x054)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	LCD	–
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **LCD : LCD Clock disable**

0: No effect

1: Disable LCD Clock

LCD Power Management Status Register LCD_PMSR (0x058) Access: Read only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	LCD	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **LCD : LCD Clock status**

0: LCD Clock is disabled.

1: LCD Clock is enabled.

LCD Control Register **LCD_CR (0x060)** **Access: Write only**

31	30	29	28	27	26	25	24
LCD_EN	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
BIASCKT_SEL	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
VLCD_SEL	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
IVLC0_SEL	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **LCD_EN : LCD Enable**

0: Disable LCD Controller.

1: Enable LCD Controller.

NOTE: When the LCD_EN bit is 0, LCD output signals (COM and SEG signals) will have logic value 0. In other words, LCD Controller is in a display turn-off state.

- **BIASCKT_SEL : BIAS Circuit Select**

0: Use internal bias circuit.

1: Use external bias circuit.

NOTES:

1. When the BIASCKT_SEL bit is set to 1, user should prepare an appropriate bias division circuit and supply the divided voltage to the IVLC[2:0]. But, there are no pins for external bias division circuit in the S3F4A0K.
2. This bit can be used to reduce the power consumption. When this bit is set to 1, VDD power will not supply to the internal bias circuit.

- **VLCD_SEL : VLCD Select**

0: Use the external VDD as the VLCD.

1: Use the internal VDD as the VLCD.

NOTE: If user wants to use external VLCD, user should supply an appropriate voltage level to the VLCD pin, unless LCD Controller will not operate properly.

- **IVLC0_SEL : VLC0 Level Select**

0: IVLC0 is determined by the internal bias dividing circuit. ($IV_{LC0} = \frac{3}{5} V_{LCD}$)1: IVLC0 will have same voltage level with V_{LCD} . ($IV_{LC0} = V_{LCD}$)

LCD Mode Register

LCD_MR (0x064)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
–	–	LCD_FREQ[1:0]		–	OP_MODE[2:0]		
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- OP_MODE[2:0] : Operating Bias and Duty Mode

OP_MODE[2:0]			Duty and Bias Mode
0	0	0	1/4 Duty, 1/3 Bias Mode
0	0	1	1/3 Duty, 1/3 Bias Mode
0	1	0	1/3 Duty, 1/2 Bias Mode
0	1	1	1/2 Duty, 1/2 Bias Mode
1	0	0	Static Mode
Others			Reserved

- LCD_FREQ[1:0] : LCD Frequency

LCD_FREQ[1:0]		LCD Frequency
0	0	62.5 Hz
0	1	125 Hz
1	0	250 Hz
1	1	500 Hz

NOTE: The LCD display frame frequency is tightly affected by a LCD operating clock which is configured by LCD_SCR register. The appropriate clock scaling value must be set in the LCD_SCR register for 1 kHz LCD clock.

LCD Status Register

LCD_SR (0x070)

Access: Read only

31	30	29	28	27	26	25	24
LCD_EN	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
BIASCKT_SEL	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
VLCD_SEL	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
IVLC0_SEL	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **LCD_EN : LCD Enable**

0: LCD Controller is disabled.

1: LCD Controller is enabled.

- **BIASCKT_SEL : BIAS Circuit Select**

0: Internal bias circuit is used.

1: External bias circuit is used.

- **VLCD_SEL : VLCD Select**

0: External VDD is used as the VLCD.

1: Internal VDD is used as the VLCD.

- **IVLC0_SEL : IVLC0 Level Select**

0: IVLC0 is determined by the internal bias dividing circuit.

1: IVLC0 will have same voltage level with bias voltage level.

LCD Display Memory 0 Register

LCD_DM0 (0x090)

Access: Read/Write

31	30	29	28	27	26	25	24
SEG7_COM[3:0]				SEG6_COM[3:0]			
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
SEG5_COM[3:0]				SEG4_COM[3:0]			
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
SEG3_COM[3:0]				SEG2_COM[3:0]			
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
SEG1_COM[3:0]				SEG0_COM[3:0]			
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

LCD Display Memory 1 Register

LCD_DM1 (0x094)

Access: Read/Write

31	30	29	28	27	26	25	24
SEG15_COM[3:0]				SEG14_COM[3:0]			
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
SEG13_COM[3:0]				SEG12_COM[3:0]			
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
SEG11_COM[3:0]				SEG10_COM[3:0]			
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
SEG9_COM[3:0]				SEG8_COM[3:0]			
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

LCD Display Memory 2 Register

LCD_DM2 (0x098)

Access: Read/Write

31	30	29	28	27	26	25	24
SEG23_COM[3:0]				SEG22_COM[3:0]			
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
SEG21_COM[3:0]				SEG20_COM[3:0]			
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
SEG19_COM[3:0]				SEG18_COM[3:0]			
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
SEG17_COM[3:0]				SEG16_COM[3:0]			
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

LCD Display Memory 3 Register

LCD_DM3 (0x09C)

Access: Read/Write

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
SEG29_COM[3:0]				SEG28_COM[3:0]			
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
SEG27_COM[3:0]				SEG26_COM[3:0]			
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
SEG25_COM[3:0]				SEG24_COM[3:0]			
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **SEG[29:0]_COM[3:0] : LCD Segment [29:0] for COM[3:0]**

0: LCD Segment is off state.

1: LCD Segment is on state.

18

LITE DIRECT MEMORY ACCESS (LDMA)

1. OVERVIEW

1.1 FUNCTIONAL DESCRIPTION

- The LDMA (Lite DMA) provides an optimized solution for automatic transfers between peripheral and memory or between memory and memory. It is possible to use EPC and internal SRAM as both source and destination. But, IFC and DFC can only be source.
- Without processor help, LDMA can transfer a certain amount of data from one block(peripheral or memory) to another block(peripheral or memory).
- Number of transfer(1 to 65535), unit of data (bytes, half-words or words) and address of source and destination are configurable.
- After desired number of transfer(counter) is programmed, each transfer decreases this counter until zero(readable by user) automatically.
- Source and target addresses are also updated in user interface to show next accessed address.
- An interrupt can be generated at the end of each transfer (when counter is zero).
- An interrupt can also be generated on source or destination access error.
- LDMA has 8 channels that work in parallel.
- To improve DMA performance, there is an AHB bus dedicated to data transfer between LDMA and peripheral on APB bus.

1.2 BLOCK DIAGRAM

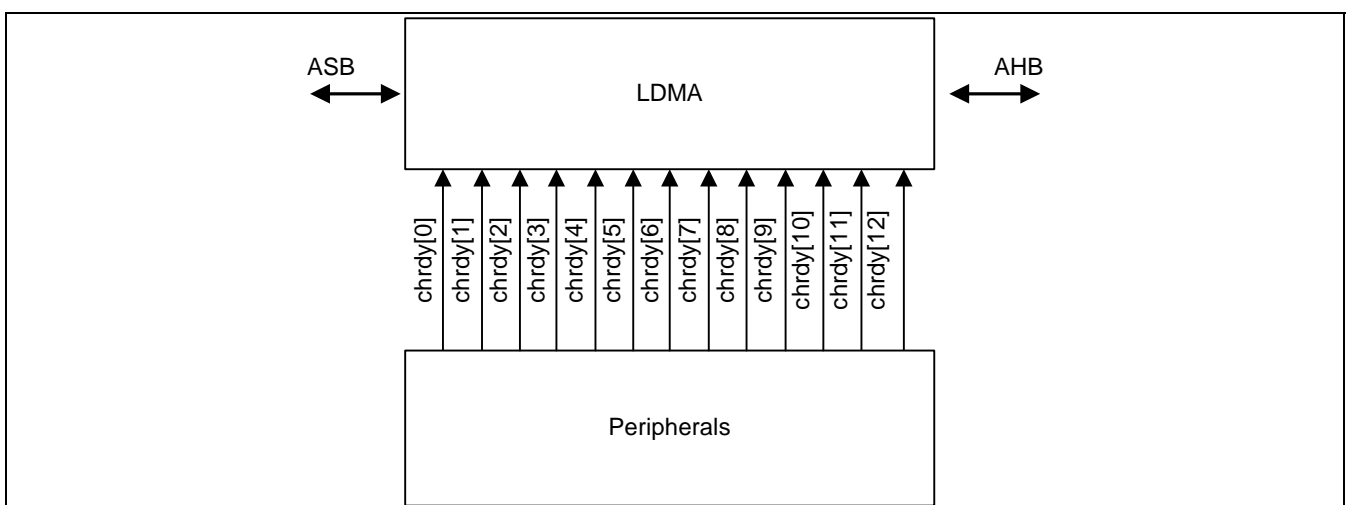


Figure 18-1. LDMA Block Diagram

2. FUNCTIONAL OPERATION

2.1 LITE DIRECT MEMORY ACCESS

2.1.1 General Description

LDMA can manage automatic transfers between peripheral and memory (both directions) or between memory and memory.

For this, it is composed of 8 channels that work in parallel, however only one transfer can be occurred at the same time (see priority management for more details).

Each channel can be configured to transfer up to 65535 data (by the unit of byte, half-word or word).

An interrupt can be generated at the end of each transfer (when counter is zero)

User can set number of transfers (1 to 65535), unit of data(byte, half-word or word), source and target addresses and choose between software and hardware trigger. Address can be fixed or automatically increased depending of size of each transfer (+1 for a byte, +2 for a half-word, +4 for a word). State of counter and current addresses are readable by the user at any time.

2.1.2 Block Diagram

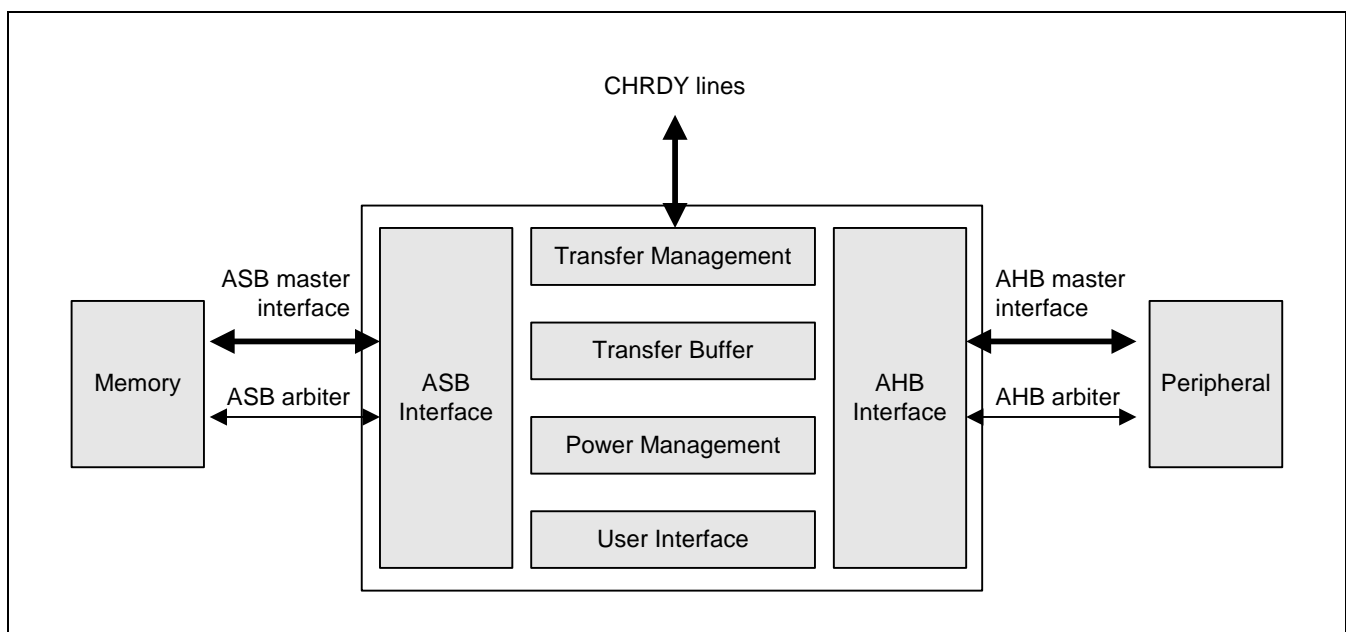


Figure 18-2. LDMA with Peripherals

2.2 LDMA WITH HARDWARE TRIGGER

2.2.1 General Description

The LDMA can transfers 1 to 65535 data (by the unit of byte, half-word or word) between peripheral and memory (both directions) or between memory and memory without processor help.

Each transfer can be triggered on the rising edge of an external signal (from a peripheral for example), except when enabling a channel where trigger occurs immediately if external signal is high. There is 13 trigger signals (CHRDY[0] ~ CHRDY[12]).

Peripheral	CHRDY Signal	
	Tx	Rx
ADC 10-bit	–	12
SPI0 (8-bit)	0	1
UART0	4	5
UART1	6	7
USART0	8	9
CAPT0	–	10
SPI1 (16-bit)	2	3

If trigger signal is generated by the source of the transfer, CHRDY signal means (when high) that a new data is ready to be transferred from source to destination. If trigger signal is generated by the recipient of the transfer, CHRDY signal means (when high) that the recipient is ready to receive a new data from the source. By this mechanism, the LDMA transfers are done at the right moment, without missing or overwriting data for specific devices like peripheral.

The user has to program :

- The unit of transfer (byte, half word, word)
- Source and destination addresses
- Number of transfers (1 to 65535)
- Whether source and destination address have to be increased or not
- Whether trigger signals is hardware or software

2.2.2 Trigger Signals

Trigger signals are the CHRDY[x] signals, allowing LDMA to do a new transfer when high.

2.2.3 ASB(memory) and AHB(peripheral) Buses Management

LDMA will first read data on source (ASB or AHB) before to write on recipient (ASB or AHB) without delay between the two accesses.

No new transfer will start before these two accesses achieved.

2.2.4 End of LDMA Transfers Interrupt

The end of LDMA transfer event is the end of the last transfer programmed by user on a given channel.

When counter of transfers of a channel (programmed by user) reach zero, an interrupt can be occurred in LDMA. End of transfer flag(LDMA_END of LDMA_SRx, See P18-24) is set at the end of the last transfer.

LDMA channel is automatically disabled at the end of transfers.

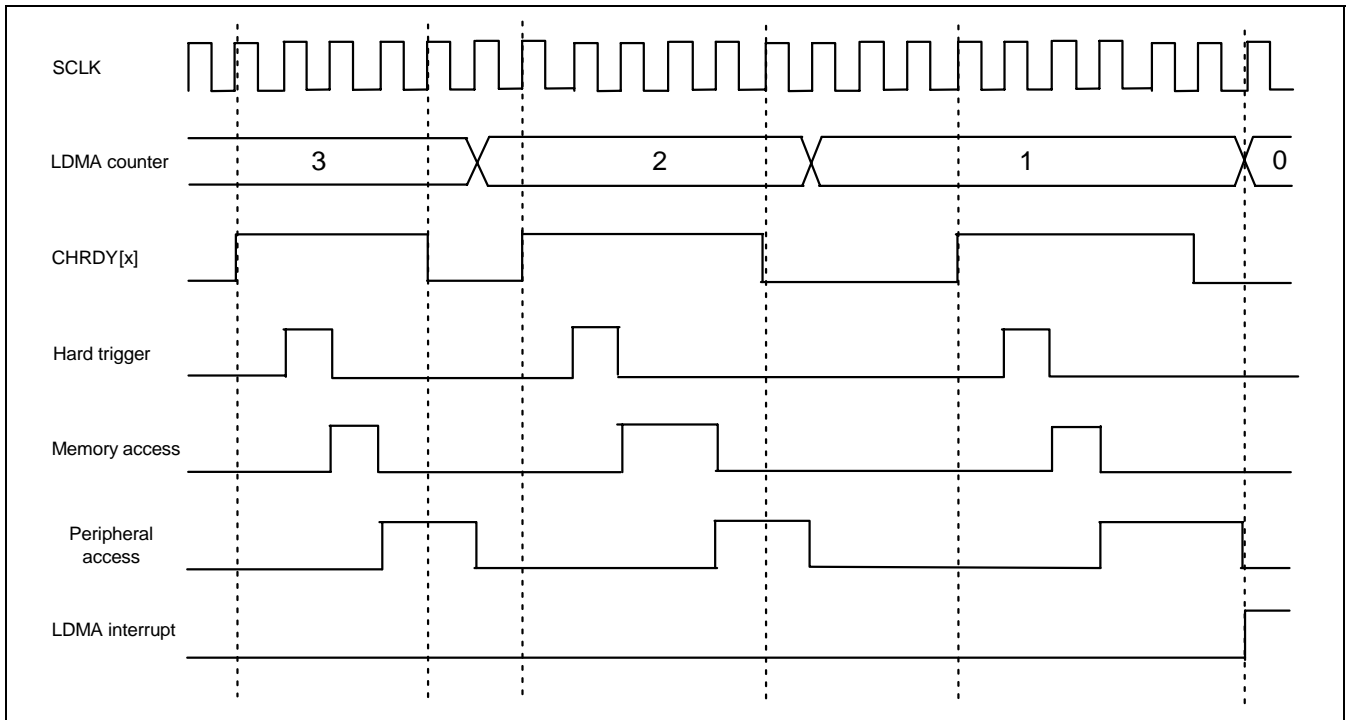


Figure 18-3. Example : Transfers From a Memory to a Peripheral

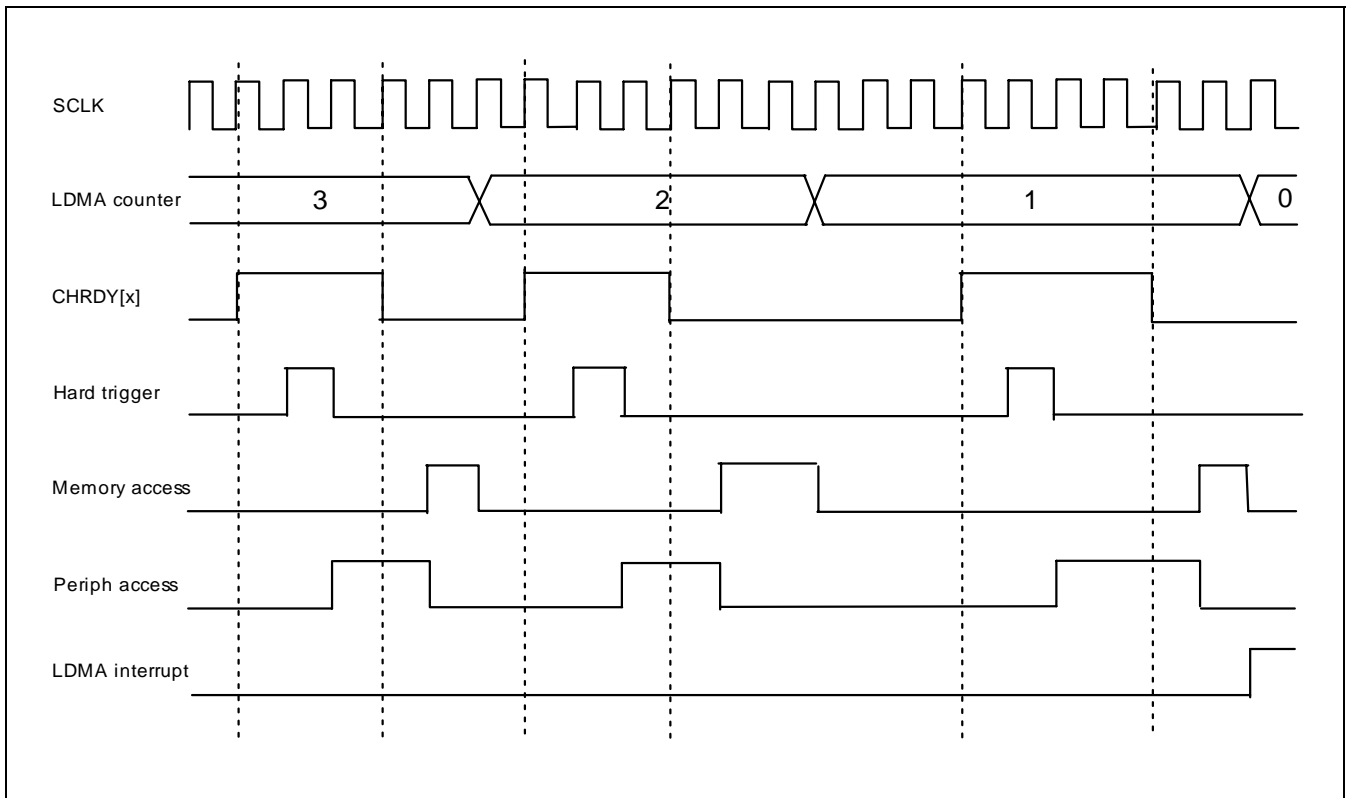


Figure 18-4. Example : Transfers From a Peripheral to a Memory

2.3 LDMA WITH SOFTWARE TRIGGER

2.3.1 General Description

Any LDMA channel can be configured to make software triggered transfer between any addresses of memory/peripheral (even an address linked to a peripheral register, however it is not generally recommended in this configuration).

The configuration and characteristics are the same (number of transfers, unit, direction, addresses), except for trigger that is software (The LCHEN bit of LDMA_CRx register, See P18-19).

User must launch transfers by software, and all transfers are completed in a row without other delay between transfers than those due to bus and/or LDMA load. No further intervention is needed until the end of all the transfers of the channel.

18-1. IMPORTANT NOTICE

Huge number of data can monopolize bus during a long time, preventing any access of other masters (that can freeze or slow down the processor during a long time).

2.3.2 ASB(memory) and AHB(peripheral) Buses Management

For each LDMA channel used with software trigger, user must specify on which bus the source is connected, and on which bus the recipient is connected (it can be the same bus for both).

LDMA will first read data on source(ASB or AHB) before to write on recipient(ASB or AHB) with no delay between the two accesses.

No new transfer will start before these two accesses achieved.

2.3.3 End of LDMA Transfers Interrupt

The end of LDMA transfer event is the end of the last transfer programmed by user on a given channel.

When counter of transfers (programmed by user) reach zero, an interrupt can be occurred in LDMA. End of transfer flag(LDMA_END of LDMA_SRx, See P18-24) is set at the end of the last transfer.

LDMA channel is automatically disabled at the end of transfers.

2.4 PRIORITY MANAGEMENT

LDMA manages priority in a very simple way : the channel with the smallest number has the highest priority, whatever configuration of each channel.

When several channels have pending transfers, the LDMA will always choose the channel having the highest priority.

When a LDMA channel is configured with a software trigger, when granted, all transfers programmed are done until the end.

When a LDMA channel is configured with a hardware trigger, a transfer starts on each rising edge of its associated CHRDY line. If another CHRDY line rising edge occurs, the corresponding channel performs a transfer. If two CHRDY rising edge occur at the same time, the associated channel with the highest priority (smallest number) performs a transfer.

It is recommended to choose first channels for hardware triggered transfers, and followers for software triggered transfers.

18-2. IMPORTANT NOTICE

If a channel with a low number is configured to make software triggered transfers, it will keep priority until the end of all its transfers. All the other channels will have to wait the end of transfers of this channel.

2.5 LDMA USER INTERFACE

2.5.1 General Description

The interface is exactly the same for all channels. Hereunder is the description of registers of one channel. Before enabling a channel, user must completely configure it. Change must not be made in channel configuration when channel is enabled.

2.5.2 Channel Configuration

In the Mode Register(LDMA_MR_x, See P18-21) of the channel, user must set :

- The source and destination buses (ASB(memory) or AHB(peripheral)) for the data transfer. This is done by SRC and DEST bits.
- The type of transfer (increased address or not) for both source and destination. This is done by SRC_INCR and DEST_INCR bits.
- The unit of transfer(byte, half word or word). This is done by LDMA_SIZE[1:0] bits.
- The trigger type (hardware or software). This is done by TRIG bit.
- The CHRDY lines number used for hardware trigger. This is done by CHREADY[4:0] bits.

In the Source Address Register and in the Destination Address Register, user has to configure each memory start address (source and destination). Depending on the SRC_INCR and DEST_INCR bits of Mode Register, those addresses are incremented or not.

When addresses are configured to be increased, those registers will be automatically updated to give the address where following transfer will be done (read or write). It means that user can read at any time source and recipient addresses where following transfer will be done. Addresses are increased at each transfer by 1 for a byte transfer size, by 2 for a half word transfer size and by 4 for a word transfer size.

In the Counter Register, user has to configure the number of transfers. This 16 bits register allows to configure 1 to 65535 transfers. Here again, the register will be updated at each transfer, so as user can read the number of remaining transfers.

The end of transfer occurred when this register reach 0.

18-3. IMPORTANT NOTICE

As APB accesses are 32 bits (byte write capability is not supported), the LDMA transfer size to APB location must always be set to 32 bits. Otherwise, bits are overwritten.

2.5.3 Start of Transfers

Once the channel configuration is done, user can start transfer by enabling the channel in its Control Register. This is done by LCHEN bit (See P18-19). User can check if a channel is enabled or not by reading the bit CHEN (See P18-23) in the corresponding Channel Status Register.

If the trigger is hardware, the LDMA channel waits for the rising edge of its associated CHRDY line before each transfer (first trigger is detected if CHRDY[x] is high or on rising edge of CHRDY[x], further trigger are detected only on rising edge of CHRDY[x]).

If the trigger is software, all the programmed transfers of the LDMA channel will be executed as quickly as possible (depending on bus load and channel priority).

2.5.4 Interrupts Management

Interrupt Enable Register, Interrupt Disable Register and Interrupt Mask Register manage the interrupts of all the channels. A LDMA channel can generate interrupts only if its associated channel interrupt mask is set in the Interrupt Mask Register. User can choose the interrupt of a channel with Interrupt Enable Register. Status Register indicates which LDMA channel has an interrupt pending.

Interrupts associated to a channel are managed by the Channel Mask Register, Channel Clear Status Register and Channel Status Register.

Each channel can generate three different interrupts :

- Transfer finished
- Source error
- Destination error

The end of transfer interrupt occurred when the channel counter reaches zero. When counter is zero, bit LDMA_END is set in the corresponding channel interrupt status register.

Source error and destination error occur when any transfer is completed with an ERROR response from the selected slave. When a source error occurs, bit SRC_ERROR is set in the corresponding channel interrupt status register. When a destination error occurs, bit DEST_ERROR is set in the corresponding channel interrupt status register. These two errors indicate that current transfer couldn't be achieved. In this case, the channel is disabled, and the counter and the source and destination addresses are not updated (keeping values of current transfer).

An interrupt can be generated by the LDMA when one or several of these bits are set. The interrupt is generated if the channel interrupts(LDMA_IMR) are enabled and if the specific channel interrupt(LDMA_IMRx) is enabled too.

When user treats a LDMA interrupt, it has to :

- First, read the LDMA Status Register(LDMA_SR), to know which channel(s) is(are) source(s) of the LDMA interrupt
- Then, read the LDMA corresponding channel Status Register(LDMA_SRx), to know exactly which event has produced the interrupt (end of transfer, source or destination error)

2.6 POWER MANAGEMENT

The power management is done automatically in the LDMA. The LDMA clocks are automatically disabled when no transfer is pending. But, reads and writes are always possible in APB registers. This ensures a minimum power consumption, even in a full speed system, and without user management.

2.7 DEBUG MODE

When the processor of the micro-controller enters in debug mode, the LDMA can be stopped, depending on bit DBGEN of the PMSR register.

If bit DBGEN is low, the LDMA continues to transfer at full speed, until end of all pending transfers.

If bit DBGEN is high, the LDMA stops immediately when the processor enters in debug mode. This stops all the clocks, but APB accesses can still be done (with the debugger for example).

If processor enters debug mode during a LDMA transfer, the current transfer and access (ASB or AHB access) is hold.

During the debug mode, no hardware trigger will be taken in account.

When processor resumes to normal mode, the LDMA restarts exactly where it has been stopped (except if the user has produced a software reset during debug mode).

18-4. IMPORTANT NOTICE

During debug state, if the user modifies the LDMA registers and/or the memory spaces involved in the halted transfer, it is recommended to reset the LDMA before leaving debug mode if transfers are pending, this to avoid unexpected effects.

3. REGISTERS DESCRIPTION

Base Address– 0xFFFF8000

Table 18-1. LDMA Special Function Registers

Offset Address	Name	Description	R/W	Reset State
0x000 – 0x04C	–	Reserved	–	–
0x050	LDMA_ECR	LDMA enable clock register	W	–
0x054	LDMA_DCR	LDMA disable clock register	W	–
0x058	LDMA_PMSR	LDMA power management status register	R	0x00000000
0x05C	–	Reserved	–	–
0x060	LDMA_CR	LDMA control register	W	–
0x064 – 0x06C	–	Reserved	–	–
0x070	LDMA_SR	LDMA status register	R	0x00000000
0x074	LDMA_IER	LDMA interrupt enable register	W	–
0x078	LDMA_IDR	LDMA interrupt disable register	W	–
0x07C	LDMA_IMR	LDMA interrupt mask register	R	0x00000000
0x080 – 0x0FC	–	Reserved	–	–
0x100 + 0x80*x (with x = 0 to 7)	LDMA_CRx	LDMA channel x control register	W	–
0x104 + 0x80*x (with x = 0 to 7)	LDMA_MRx	LDMA channel x mode register	R/W	0x00000000
0x108 + 0x80*x (with x = 0 to 7)	LDMA_CSRx	LDMA channel x interrupt clear status register	W	–
0x10C + 0x80*x (with x = 0 to 7)	LDMA_SRx	LDMA channel x status register	R	0x00000000
0x110 + 0x80*x (with x = 0 to 7)	LDMA_IERx	LDMA channel x interrupt enable register	W	–
0x114 + 0x80*x (with x = 0 to 7)	LDMA_IDRx	LDMA channel x interrupt disable register	W	–
0x118 + 0x80*x (with x = 0 to 7)	LDMA_IMRx	LDMA channel x interrupt mask register	R	0x00000000
0x11C + 0x80*x (with x = 0 to 7)	LDMA_ASRCRx	LDMA channel x source address register	R/W	0x00000000
0x120 + 0x80*x (with x = 0 to 7)	LDMA_ADSTRx	LDMA channel x destination address register	R/W	0x00000000
0x124 + 0x80*x (with x = 0 to 7)	LDMA_CNTRx	LDMA channel x transfer counter	R/W	0x00000000

LDMA Enable Clock Register

LDMA_ECR (0x50)

Access: Write only

31	30	29	28	27	26	25	24
DBGEN	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **DBGEN : Debug mode enable**

0: No effect

1: LDMA is halted during processor debug mode.

LDMA Disable Clock Register

LDMA_DCR (0x54)

Access: Write only

31	30	29	28	27	26	25	24
DBGEN	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **DBGEN : Debug mode disable**

0: No effect.

1: LDMA is not halted during processor debug mode.

LDMA Power Management Status Register LDMA_PMSR (0x58) Access: Read only

31	30	29	28	27	26	25	24
DBGEN	–	IPIDCODE[25:20]					
R-0	R-0	R-U	R-U	R-U	R-U	R-U	R-U
23	22	21	20	19	18	17	16
IPIDCODE[19:12]							
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U
15	14	13	12	11	10	9	8
IPIDCODE[11:4]							
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U
7	6	5	4	3	2	1	0
IPIDCODE[3:0]				–	–	–	–
R-U	R-U	R-U	R-U	R-0	R-0	R-0	R-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **IPIDCODE[25:0]**

This field contains the version number of the module, coded on 26 bits.

- **DBGEN : Debug mode**

0: LDMA is not halted during processor debug mode.

1: LDMA is halted during processor debug mode.

LDMA Control Register **LDMA_CR (0x60)** **Access: Write only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	SWRST
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **SWRST : software reset**

0: No effect

1: Global software reset (resets all registers except PMSR register)

LDMA Status Register

LDMA_SR (0x70)

Access: Read only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
CH7_IT	CH6_IT	CH5_IT	CH4_IT	CH3_IT	CH2_IT	CH1_IT	CH0_IT
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **CHx_IT : channel x interrupt**

0: No interrupt on channel x

1: Channel x made an interrupt

LDMA Interrupt Enable Register

LDMA_IER (0x74)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
7	6	5	4	3	2	1	0
CH7_IT	CH6_IT	CH5_IT	CH4_IT	CH3_IT	CH2_IT	CH1_IT	CH0_IT
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **CHx_IT : channel x interrupt enable**

0: No effect

1: Enable channel x interrupt

LDMA Interrupt Disable Register

LDMA_IDR (0x78)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
7	6	5	4	3	2	1	0
CH7_IT	CH6_IT	CH5_IT	CH4_IT	CH3_IT	CH2_IT	CH1_IT	CH0_IT
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **CHx_IT : channel x interrupt disable**

0: No effect

1: Disable channel x interrupt

LDMA Interrupt Mask Register

LDMA_IMR (0x7C)

Access: Read only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
CH7_IT	CH6_IT	CH5_IT	CH4_IT	CH3_IT	CH2_IT	CH1_IT	CH0_IT
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- CHx_IT : channel x interrupt mask**

0: Channel x interrupt is disabled.

1: Channel x interrupt is enabled.

LDMA Channel x Control Register **LDMA_CRx (0x100 + 80*x)** **Access: Write only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
7	6	5	4	3	2	1	0
–	–	–	–	–	LCHDIS	LCHEN	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **LCHEN : enable LDMA channel x**

0: No effect

1: Enable LDMA channel x

NOTE: When a software trigger is programmed, writing 1 in this register start transfers.

- **LCHDIS : disable LDMA channel x**

0: No effect

1: Disable LDMA channel x

LDMA Channel x Mode Register			LDMA_MR _x (0x104 + 80*x)				Access: Read/Write	
31	30	29	28	27	26	25	24	
–	–	–	–	–	–	–	–	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
23	22	21	20	19	18	17	16	
–	–	–	–	–	–	–	–	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
15	14	13	12	11	10	9	8	
–	–	–	CHREADY[4:0]					–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
7	6	5	4	3	2	1	0	
–	LDMA_SIZE[1:0]		TRIG	DEST_INCR	SRC_INCR	DEST	SRC	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **SRC : Source bus for data transfer**

0: AHB bus is the source bus (peripheral)

1: ASB bus is the source bus (memory)

- **DEST : Destination bus for data transfer**

0: AHB bus is the destination bus (peripheral)

1: ASB bus is the destination bus (memory)

- **SRC_INCR : Increase source address**

0: Source address will not be increased.

1: Source address will be increased.

- **DEST_INCR : Increase destination address**

0: Destination address will not be increased.

1: Destination address will be increased.

- **TRIG : Trigger type**

0: Software trigger (The LCHEN of LDMA_CR_x is the start bit for software trigger.)

1: Hardware trigger

- **LDMA_SIZE[1:0]** : unit of the transfer

Table 19-2. LDMA Transfers Size

LDMA_SIZE[1:0]	Size	Description
00	8 bits	byte
01	16 bits	halfword
10	32 bits	word
11	reserved	

- **CHREADY[4:0]** : defines which CHRDY signal is valid for channel x.

18-5. IMPORTANT NOTICE

1. This register can be written only when channel X is disabled.
2. CHREADY is don't care condition in the software trigger.

LDMA Channel x Interrupt Clear Status Register LDMA_CSRx (0x108 + 80*x)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
7	6	5	4	3	2	1	0
–	–	–	–	–	DEST_ERROR	SRC_ERROR	LDMA_END
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **LDMA_END : Clear LDMA channel x transfer finished**

0: No effect

1: Clear channel x status register.

- **SRC_ERROR : Clear LDMA channel x source error**

0: No effect

1: Clear channel x status register.

- **DEST_ERROR : Clear LDMA channel x destination error**

0: No effect

1: Clear channel x status register.

LDMA Channel x Status Register

LDMA_SRx (0x108 + 80*x)

Access: Read only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	CHEN
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
7	6	5	4	3	2	1	0
–	–	–	–	–	DEST_ERROR	SRC_ERROR	LDMA_END
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **LDMA_END : LDMA channel x transfer finish**

0: LDMA channel x transfer is not finished.

1: LDMA channel x transfer is finished.

- **SRC_ERROR : LDMA channel x source error**

0: No source error

1: Source error

- **DEST_ERROR : LDMA channel x destination error**

0: No destination error

1: Destination error

- **CHEN : LDMA channel x enable**

0: LDMA channel x is disabled.

1: LDMA channel x is enabled.

18-6. IMPORTANT NOTICE

Because the channel is enabled by the user, this CHEN bit can not be used for interrupt source.

LDMA Channel x Interrupt Enable Register LDMA_IERx (0x110 + 80*x) Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
7	6	5	4	3	2	1	0
–	–	–	–	–	DEST_ERROR	SRC_ERROR	LDMA_END
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **LDMA_END : LDMA channel x transfer finished**

0: No effect

1: Enable interrupt

- **SRC_ERROR : LDMA channel x source error**

0: No effect

1: Enable interrupt

- **DEST_ERROR : LDMA channel x destination error**

0: No effect

1: Enable interrupt

LDMA Channel x Interrupt Disable Register LDMA_IDRx (0x114 + 80*x) Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
7	6	5	4	3	2	1	0
–	–	–	–	–	DEST_ERROR	SRC_ERROR	LDMA_END
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **LDMA_END : LDMA channel x transfer finished**

0: No effect

1: Disable interrupt

- **SRC_ERROR : LDMA channel x source error**

0: No effect

1: Disable interrupt

- **DEST_ERROR : LDMA channel x destination error**

0: No effect

1: Disable interrupt

LDMA Channel x Interrupt Mask Register **LDMA_IMRx (0x118 + 80*x)** **Access: Read only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
–	–	–	–	–	DEST_ERROR	SRC_ERROR	LDMA_END
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **LDMA_END : LDMA channel x transfer finished**

0: Interrupt is not masked

1: Interrupt is masked

- **SRC_ERROR : LDMA channel x source error**

0: Interrupt is not masked

1: Interrupt is masked

- **DEST_ERROR : LDMA channel x destination error**

0: Interrupt is not masked

1: Interrupt is masked

LDMA Channel x Source Address Register LDMA_ASRCRx (0x11C + 80*x) Access: Read/Write

31	30	29	28	27	26	25	24
SRC_ADD[31:24]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
SRC_ADD[23:16]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
SRC_ADD[15:8]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
SRC_ADD[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **SRC_ADD[31:0] : source address**

18-7. IMPORTANT NOTICE

This register can be written only when channel X is disabled.

LDMA Channel x Destination Address Register LDMA_ADSTRx (0x120 + 80*x) Access: Read/Write

31	30	29	28	27	26	25	24
DEST_ADD[31:24]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
DEST_ADD[23:16]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
DEST_ADD[15:8]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
DEST_ADD[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **DEST_ADD[31:0] : destination address**

18-8. IMPORTANT NOTICE

This register can be written only when channel X is disabled.

LDMA Channel x Transfer Counter		LDMA_CNTRx (0x124 + 80*x)				Access: Read/Write	
31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
DATA_CNT[15:8]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
DATA_CNT[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **DATA_CNT[15:0]** : number of data to be transferred (number of byte, half word or word, depending on bits LDMA_SIZE[1:0] of Mode register)

18-9. IMPORTANT NOTICE

This register can be written only when channel X is disabled.

19

PULSE WIDTH MODULATION (PWM)

1. OVERVIEW

1.1 FUNCTIONAL DESCRIPTION

The PWM block provides 2 channels which generate pulses. The width and the frequency of the pulses can be controlled independently on each channel. Then with the PWM it is possible to generate 2 different waveforms on the same time.

1.2 BLOCK DIAGRAM

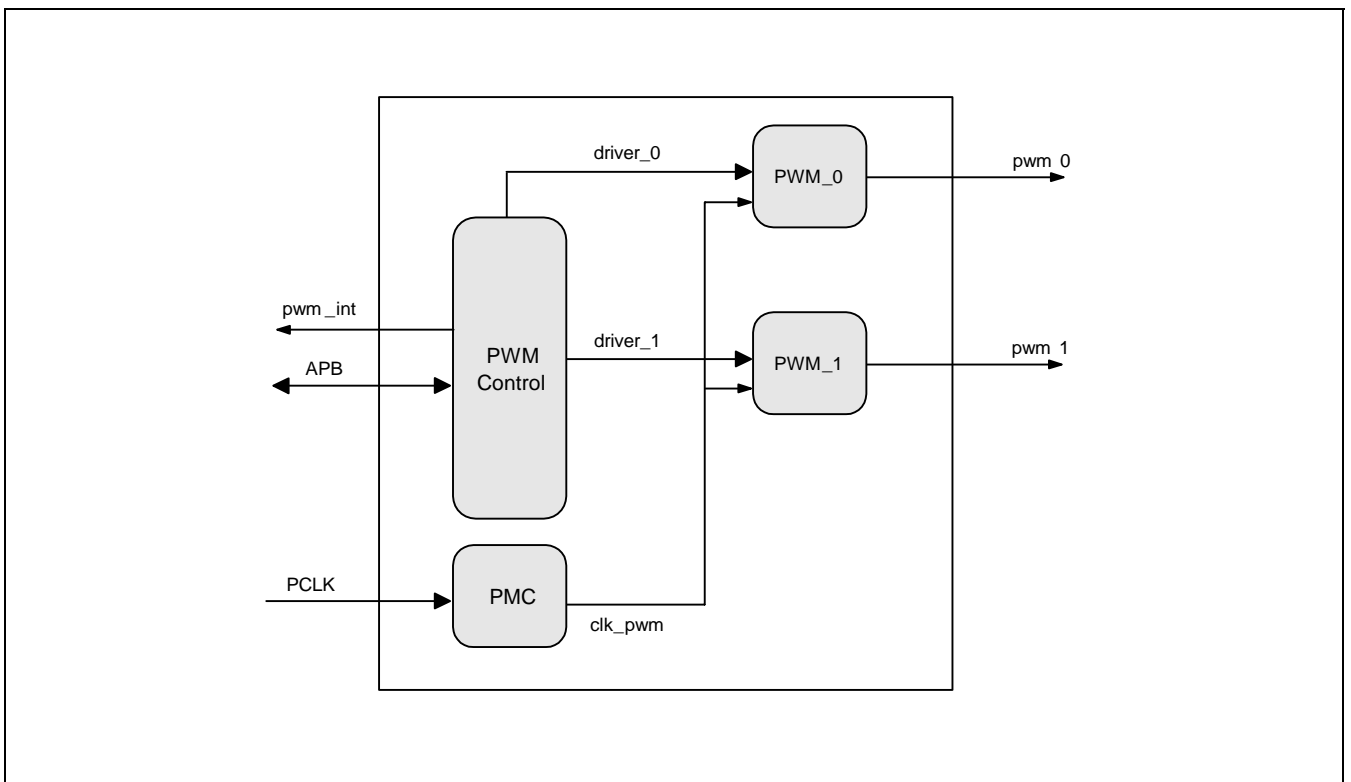


Figure 19-1. PWM Block Diagram

2. EXTERNAL PIN DESCRIPTION

Table 19-1. PWM Pin Description

Pin Name	Function	I/O Type	Active Level	Comments
PWM[1:0]	Pulse width modulation output	O	–	–

3. FUNCTIONAL OPERATION

3.1 PWM PARAMETERS

3.1.1 General Description

There are 3 parameters for each PWM : counter frequency, delay and pulse width.
A fourth parameter allows users to change polarity of the pulse and the delay levels. By default, the pulse is at a logical '1' level and the delay is at a logical '0' level.

3.1.2 Counter Frequency

Basically, the PWM module is a counter. It counts "delay width" cycles, setting the PWMX output inactive, then it counts "pulse width" cycles, setting the PWMx output active. This operation is repeated until the PWM channel is stopped. User can control the frequency of this counter with a pre-scalar.

3.1.3 Delay Width

This is the number of counter cycles during which the output is inactive. PWM starts with a delay when enabled.

3.1.4 Pulse Width

This is the number of counter cycles during which the output is active.

3.1.5 PWM Output Level When Disabled

When the PWM module goes from enable to disable state, its outputs are not fixed the specific level.

3.2 PROGRAMMING EXAMPLES

If the counter frequency is set to PCLK/2, delay width is equal to 4 cycles and pulse width to 2 cycles:

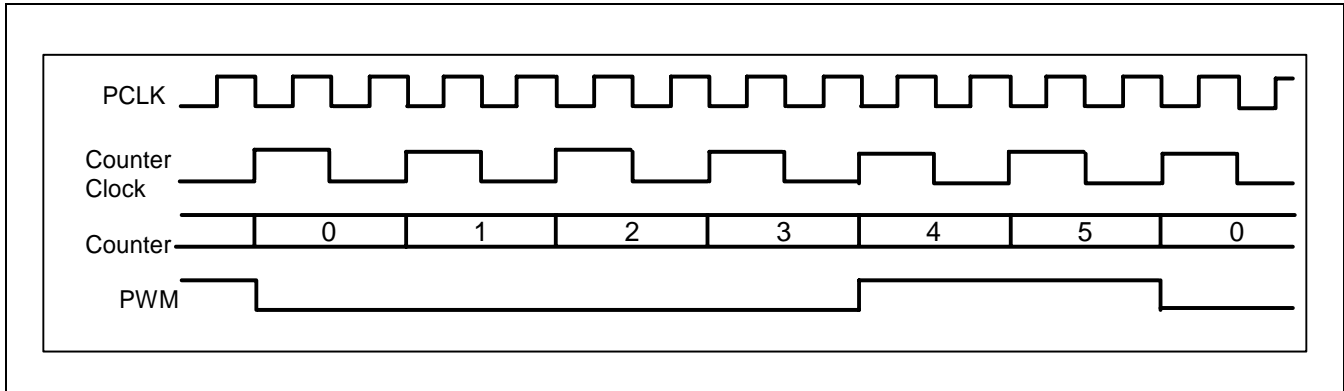


Figure 19-2. Example

Example of use of the PWM :

We want to generate a pulse with a frequency of 5 KHz with a duty cycle of 50%, PCLK = 30MHz.

Configuration :

- Enable the clock on PWM peripheral by writing bit PWM in PWM_ECR.
- Do a software reset of the PWM peripheral to be in a known state by writing bit SWRST in PWM_CR.
- Configuration of PWM_MR : We use a PWM clock of 15 MHz with a PRESCAL of 0 and the pulse is at a logical state 1.
- Configuration of PWM_DLYx : This register indicate the number of PWM clock needed to form the delay (level 0), 1500 is suitable for half period of a 0.1 ms.
- Configuration of PWM_PULx : This register indicate the number of PWM clock needed to form the pulse (level 1), 1500 is suitable for half period of a 0.1ms.
- Start the chosen PWM by writing the bit PWMENx in PWM_CR. This should supply a pulse of 5 kHz on the chosen PWM with a duty cycle of 50%

4. REGISTERS DESCRIPTION

Base Address – PWM0 : 0xFFE08000

Table 19-2. PWM Special Function Registers

Offset Address	Name	Description	R/W	Reset State
0x000 – 0x04C	–	Reserved	–	–
0x050	PWM_ECR	Enable clock register	W	–
0x054	PWM_DCR	Disable clock register	W	–
0x058	PWM_PMSR	Power management status register	R	0XXXXXXXX0
0x05C	–	Reserved	–	–
0x060	PWM_CR	Control register	W	–
0x064	PWM_MR	Mode register	R/W	0x20202020
0x068	–	Reserved	–	–
0x06C	PWM_CSR	Clear status register	W	–
0x070	PWM_SR	Status register	R	0x00000000
0x074	PWM_IER	Interrupt enable register	W	–
0x078	PWM_IDR	Interrupt disable register	W	–
0x07C	PWM_IMR	Interrupt mask register	R	0x00000000
0x080	PWM_DLY0	Delay register 0	R/W	0x00000000
0x084	PWM_PUL0	Pulse register 0	R/W	0x00000000
0x088	PWM_DLY1	Delay register 1	R/W	0x00000000
0x08C	PWM_PUL1	Pulse register 1	R/W	0x00000000

PWM Enable Clock Register

PWM_ECR (0x050)

Access: Write only

31	30	29	28	27	26	25	24
DBGEN	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	PWM	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **PWM : PWM clock enable**

0: No effect

1: Enable PWM clock

- **DBGEN : Debug mode enable**

0: No effect

1: Enable debug mode

PWM Disable Clock Register

PWM_DCR (0x054)

Access: Write only

31	30	29	28	27	26	25	24
DBGEN	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	PWM	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **PWM : PWM clock disable**

0: No effect

1: Disable PWM clock

- **DBGEN : Debug mode disable**

0: No effect

1: Disable debug mode

PWM Power Management Status Register **PWM_PMSR (0x058)** **Access: Read only**

31	30	29	28	27	26	25	24
DBGEN	–	IPIDCODE[25:20]					
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
23	22	21	20	19	18	17	16
IPIDCODE[19:12]							
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8
IPIDCODE[11:4]							
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
7	6	5	4	3	2	1	0
IPIDCODE[3:0]				–	–	PWM	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **PWM : PWM clock status**

- 0: PWM clock disabled
- 1: PWM clock enabled

- **IPIDCODE[25:0]**

This field contains the version number of the module, coded on 26 bits.

- **DBGEN : Debug mode**

- 0: The DEBUG ACKNOWLEDGE generated by the ICE interface has no influence on the PWM function.
- 1: The DEBUG ACKNOWLEDGE freezes the PMW function when the ICE interface is activated. However full read/write access to internal register is kept for the debug purpose.

NOTE: The PWM_PMSR register is not reset by software reset.

PWM Control Register **PWM_CR (0x060)** **Access: Write only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8
–	–	–	–	–	PWMDIS1	PWMEN1	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
7	6	5	4	3	2	1	0
–	–	–	–	–	PWMDIS0	PWMEN0	SWRST
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **SWRST : PWM software reset**

0: No effect.

1: Reset the PWM

A software triggered hardware reset of the PWM is performed. It resets all the registers except the PWM_PMSR register.

- **PWMENx : PWM enables channel number x**

0: No effect.

1: Enables the PWM.

- **PWMDISx : PWM disables channel number x**

0: No effect.

1: Disables the PWM.

In case both PWMENx and PWNDISx are equal to one the corresponding PWM channel will be disabled.

PWM Mode Register **PWM_MR (0x064)** **Access: Read/Write**

31	30	29	28	27	26	25	24
–	–	–	–				
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
–	–	–	–				
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
–	–	PL1	PRESCAL1[4:0]				
R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
–	–	PL0	PRESCAL0[4:0]				
R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **PRESCALx[4:0]**

Counter Clock frequency = $PCLK / 2^{\text{PRESCALAR}+1}$

- **PLx**

0: The pulses are at logic level 0. During the delay the output is at logic level 1.

1: The pulses are at logic level 1. During the delay the output is at logic level 0.

NOTES:

1. When the Pulse Level changes and a pulse start or a pulse end is detected, the corresponding bit is set in the status register (generating an interrupt if enabled)
2. Each PWM cycle starts with a delay.

PWM Clear Status Register

PWM_CSR (0x06C)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	PEND1	PSTA1
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
7	6	5	5	3	2	1	0
–	–	–	–	–	–	PEND0	PSTA0
W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **PSTAx : Clear pulse start interrupt**

0: No effect.

1: Clear PSTA interrupt.

- **PENDx : Clear pulse end interrupt**

0: No effect.

1: Clear PENDING interrupt.

NOTE: “x” indicates that the bit concerns the PWMx.

PWM Status Register **PWM_SR (0x070)** **Access: Read only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	PWENS1	PEND1	PSTA1
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	5	3	2	1	0
–	–	–	–	–	PWENS0	PEND0	PSTA0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **PSTAx**

0: No pulse start since the last read of PWM_SR.

1: A pulse started since the last read of PWM_SR.

- **PENDx**

0: No pulse end since the last read of PWM_SR.

1: A pulse ended since the last read of PWM_SR.

- **PWENSx**

0: PWM is disabled.

1: PWM is enabled.

NOTE: “x” indicates that the bit concerns the PWMx.

PWM Interrupt Enable Register

PWM_IER (0x074)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	PEND1	PSTA1
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	5	3	2	1	0
–	–	–	–	–	–	PEND0	PSTA0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **PSTAx: Pulse start interrupt enable**

0: No effect.

1: Enable Pulse Start Interrupt

- **PENDx: Pulse end interrupt enable**

0: No effect.

1: Enable Pulse End Interrupt

NOTE: “x” indicates that the bit concerns the PWMx.

PWM Interrupt Disable Register

PWM_IDR (0x078)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	PEND1	PSTA1
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	5	3	2	1	0
–	–	–	–	–	–	PEND0	PSTA0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **PSTAx: Pulse start interrupt disable**

0: No effect

1: Disable Pulse Start Interrupt

- **PENDx: Pulse end interrupt disable**

0: No effect

1: Disable Pulse End Interrupt

NOTE: “x” indicates that the bit concerns the PWMx.

PWM Interrupt Mask Register

PWM_IMR (0x07C)

Access: Read only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	PEND1	PSTA1
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	5	3	2	1	0
–	–	–	–	–	–	PEND0	PSTA0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **PSTAx: Pulse start interrupt mask**

0: Pulse Start Interrupt is disabled.

1: Pulse Start Interrupt is enabled.

- **PENDx: Pulse end interrupt mask**

0: Pulse End Interrupt is disabled.

1: Pulse End Interrupt is enabled.

NOTE: "x" indicates that the bit concerns the PWMx.

PWM Delay Register 0

PWM_DLY0 (0x080)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
DELAY[15:8]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
DELAY[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- DELAY[15:0]**

Numbers of counter cycles during which the output is inactive.

NOTE: “x” indicates that the bit concerns the PWMx.

Change of this value is immediately taken into account. This may cause a bad delay on the current pulse.

The DELAY field should not be left equal to zero when the considered PWM is used.

PWM Delay Register 1

PWM_DLY1 (0x088)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
DELAY[15:8]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
DELAY[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- DELAY[15:0]**

Numbers of counter cycles during which the output is inactive.

NOTE: “x” indicates that the bit concerns the PWMx.

Change of this value is immediately taken into account. This may cause a bad delay on the current pulse.

The DELAY field should not be left equal to zero when the considered PWM is used.

PWM Pulse Register 0

PWM_PUL0 (0x084)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
PULSE[15:8]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
PULSE[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- PULSE[15:0] : Pulse Width on channel x**

Numbers of counter cycles during which the output is active.

NOTE: “x” indicates that the bit concerns the PWMx.

Change of this value is immediately taken into account. This may cause a bad delay on the current pulse.

PWM Pulse Register 1

PWM_PUL1 (0x08C)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
PULSE[15:8]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
PULSE[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- PULSE[15:0] : Pulse Width on channel x**

Numbers of counter cycles during which the output is active.

NOTE: "x" indicates that the bit concerns the PWMx.

Change of this value is immediately taken into account. This may cause a bad delay on the current pulse.

20 SERIAL PERIPHERAL INTERFACE 16-BIT (SPI16)

1. OVERVIEW

1.1 FUNCTIONAL DESCRIPTION

The SPI16 provides communication with external devices in master or slave mode. It also allows communication between processors if an external processor is connected to the system. 7 pins are associated with the SPI16 Interface.

The main features are :

- 8 to 16 bits transfer
- 7 pins : NPCS[3:0], MISO1, MOSI1 and SPCK1
- Master / Slave mode
- Programmable baud rate generator
- Serial clock with programmable polarity and phase
- Local loop back mode
- Interrupt generation
- 2 dedicated LDMA channels

The NPCS0 pin can be used for peripheral chip select output or slave select input.

2. EXTERNAL PIN DESCRIPTION

Table 20-1. SPI16 Pin Description

Pin Name	Function	I/O Type	Active Level	Comments
SPCK1	SPI clock	I/O	–	–
MISO1	Master in slave out	I/O	–	–
MOSI1	Master out slave in	I/O	–	–
NPCS[3:1]	Chip select	I/O	Low	–
NPCS0	Chip select (slave input)	I/O	Low	–

2.1 PIN DESCRIPTION

Seven pins are associated with the SPI16 interface. When not needed for the SPI16 function, each of these pins can be configured as a PIO, using PIO Controller functions.

NOTES:

1. When PIO block is included, after a hardware reset the pins are controlled by the PIO. Even the NPCS0 Parallel I/O Controller for multi-master operation is provided by the PIO Controller. Multi-Driver option : to configure a SPI8 pin as open-drain to support external drivers, set the corresponding bits in the SPI_MDSR register or in convenient register of external PIO block. The NSS pin can function as a peripheral chip select output or slave select input.
2. A pull up resistor must be connected on the corresponding pin configured in open-drain mode.

3. FUNCTIONAL OPERATION

3.1 MASTER MODE

3.1.1 General description

In master mode, the SPI16 controls data transfers to and from the slave(s). The SPI16 drives the chip select(s) to the slave(s) and the serial clock (SPCK). After enabling the SPI16, a data transfer begins when the ARM core writes to the Transmit Data Register (SPI_TDR).

Transmit and Receive buffers maintain the data flow at a constant rate with a reduced requirement for high priority interrupt servicing. As long as new data is available in the Transmit Data Register (SPI_TDR), the SPI16 continues to transfer data. If the Receive Data Register (SPI_RDR) has not been read before new data is received, the overrun error (SPIOVRE) flag is set.

The delay between the activation of the chip select and the start of the data transfer (DLYBS) as well as the delay between each data transfer (DLYBCT) can be programmed for each of the 4 external chip selects. All data transfer characteristics including the 2 timing values are programmed in registers SPI_SSR0 to SPI_SSR3 (Slave Select Registers, See P21-22).

In master mode the peripheral selection can be defined in 2 different ways by the PS bit of SPI_MR register(See P21-14). :

- Fixed peripheral select : SPI16 exchanges data with only one peripheral.
- Variable peripheral select : Data can be exchanged with more than one peripheral.

NOTE: The SPI16 master should never be configured as described below :
SPI16 master clock equals PCLK by setting the bit DIV32 to a logical 0 in the SPI16 mode register (SPI_MR). Pin NPCS[3:0] are used as an open drain. These 2 options set in the same configuration may cause the mode fault due to the frequency of the PCLK which is used to sample the state of the NPCS[3:0] signal after each transmission.

Fixed Peripheral Select

This mode is ideal for transferring memory blocks without the extra overhead in the transmit data register. Fixed peripheral select is activated by setting bit PS to a logical 0 in the SPI16 Mode Register (SPI_MR). The peripheral is defined by the PCS[3:0] field in SPI_MR. This option is only available when the SPI16 is programmed in master mode.

Variable Peripheral Select

Variable peripheral select is activated by setting bit PS to a logical 1 in the SPI16 Mode Register (SPI_MR). The PCS field in Transmit Data Register (SPI_TDR) is used to select the destination peripheral. The data transfer characteristics are changed when the selected peripheral changes, according to the associated chip select register. The PCS field in the SPI_MR has no effect. This option is only available when the SPI16 is programmed in master mode.

Chip Select

The Chip Select lines are driven by the SPI16 only if it is programmed in Master Mode. These lines are used to select the destination peripheral.

The PCSDEC field in SPI_MR (Mode Register) is used to select 1 to 4 peripherals or 1 to 16 peripherals :

- PCSDEC = 0, each NPCSx acts as a chip select line so up to 4 devices can be selected.
- PCSDEC = 1, the NPCS[3:0] signals act as an address bus selecting up to 16 peripherals.

If variable peripheral select is active (PS = 1 in SPI_MR), the chip select signals are defined for each transfer in the PCS field in SPI_TDR. Chip select signals can thus be defined independently for each transfer.

If fixed peripheral select is active (PS = 0 in SPI_MR), chip select signals are defined for all transfers by the field PCS in SPI_MR. If a transfer with a new peripheral is necessary, the software must wait until the current transfer is completed, then change the value of PCS in SPI_MR before writing new data in SPI_TDR.

The value on the NPCS pins at the end of each transfer can be read in the Receive Data Register (SPI_RDR). By default, all NPCS signals are high (equal to one) before and after each transfer.

Mode Fault Detection

A mode fault is detected when the SPI16 is programmed in Master Mode and a low level is driven by an external master on the NPCS0 signal.

When a mode fault is detected, the MODF bit in the SPI_SR is set until the SPI_SR is read and the SPI16 is disabled until re-enabled by bit SPIEN in the SPI_CR (Control Register).

A mode fault is not detected when the master NPCS0 signal is configured in PIO and driven low.

3.1.2 Block Diagram

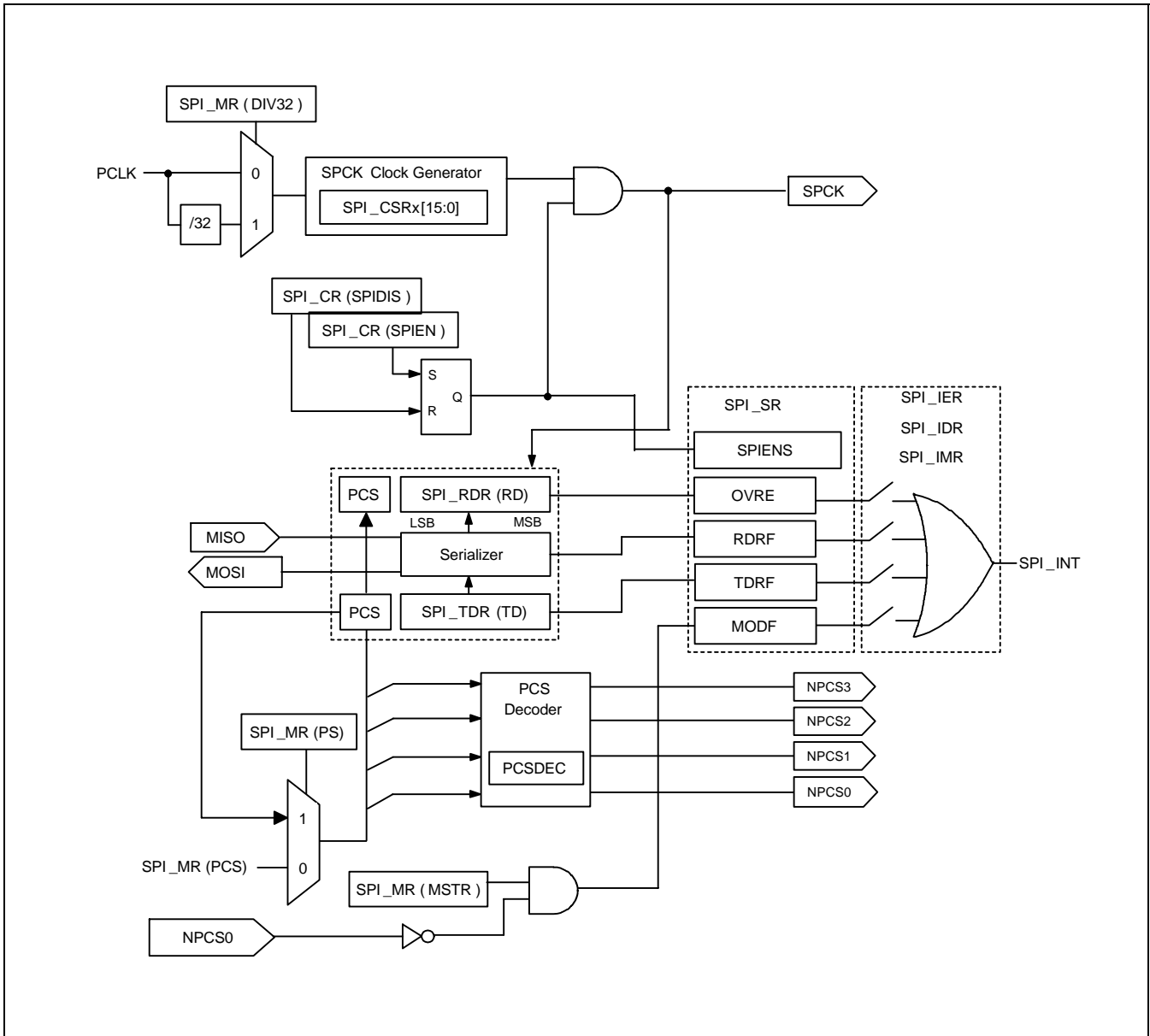


Figure 20-1. SPI16 Block Diagram in Master Mode

3.1.3 Flow Chart

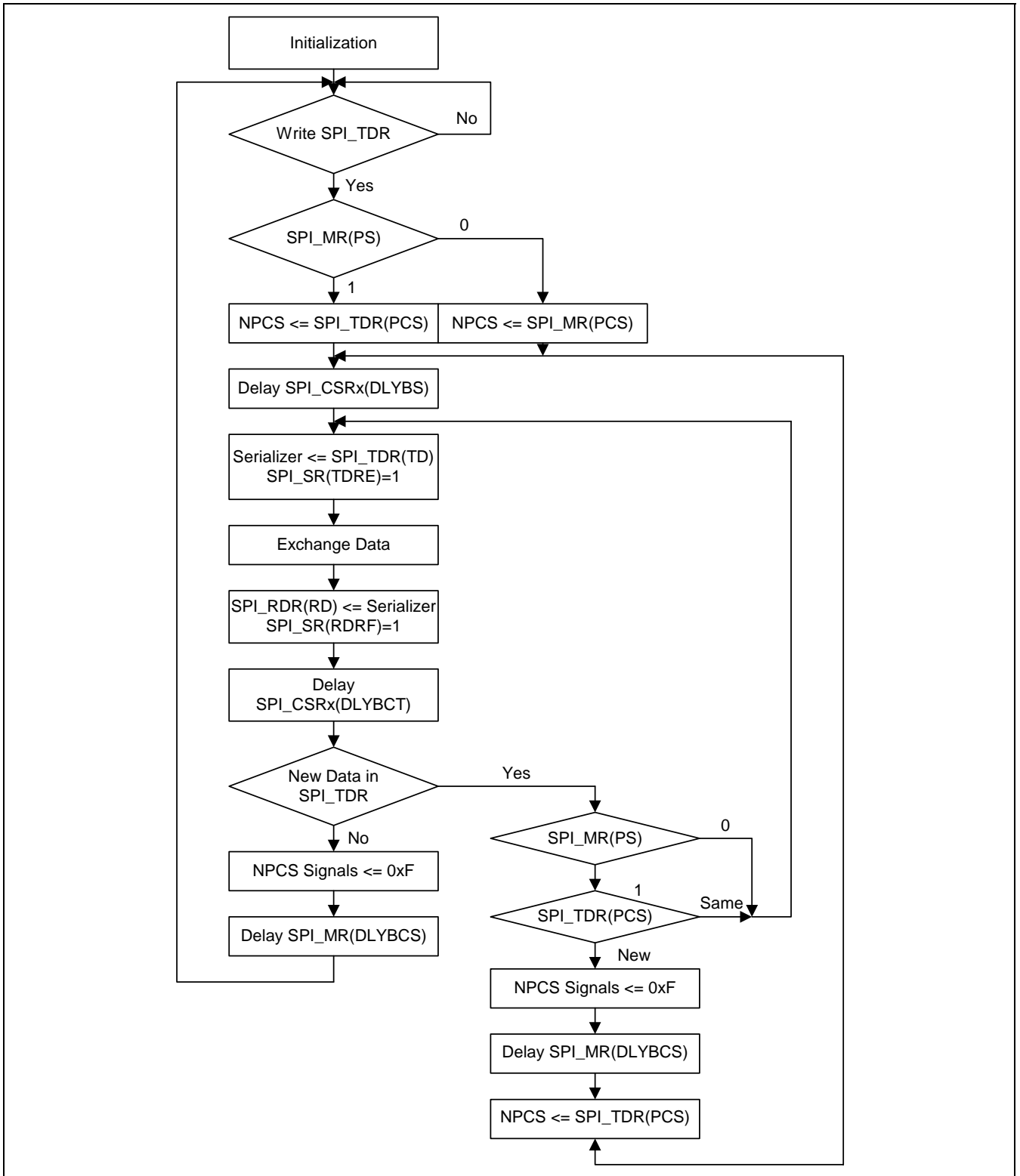


Figure 20-2. SPI16 Flow Diagram in Master Mode

3.2 SLAVE MODE

3.2.1 General Description

In Slave Mode, the SPI16 waits for NPCS0 to go active low before receiving the serial clock from an external master. In slave mode CPOL, NCPHA and BITS fields of SPI_SSR0 are used to define the transfer characteristics. The other fields in SPI_SSR0 and the other Chip Select Registers are not used in slave mode.

NOTE

The SPI16 in slave mode can not be used with the LDMA for data transmission or reception.

3.2.2 Block Diagram

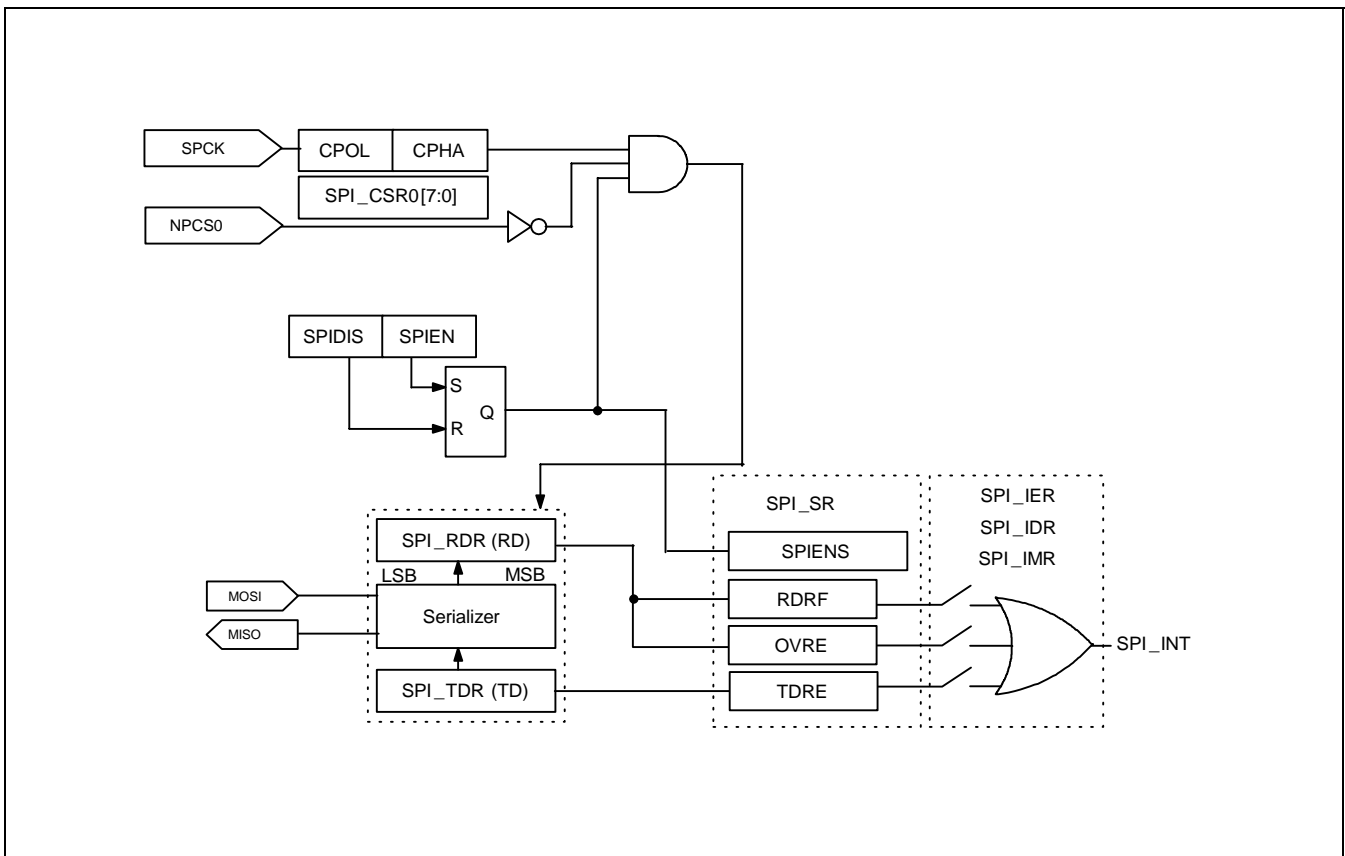


Figure 20-3. SIP16 Block Diagram in Slave Mode

4. TIMINGS

4.1 TIMINGS MODULE

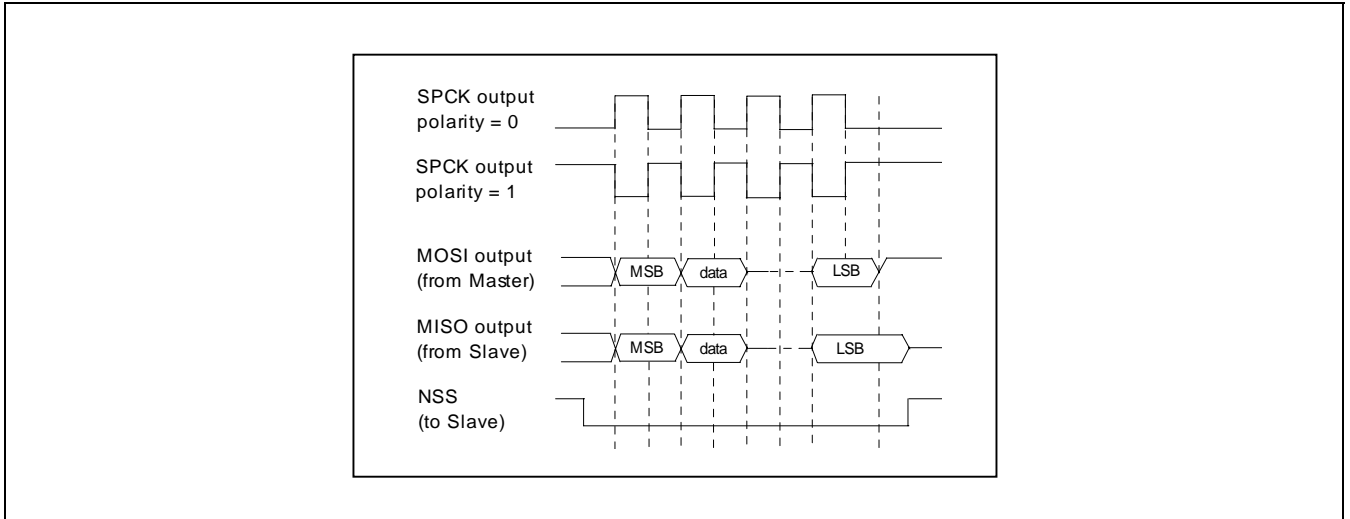


Figure 20-4. SPI16 in Master Mode, phase=0

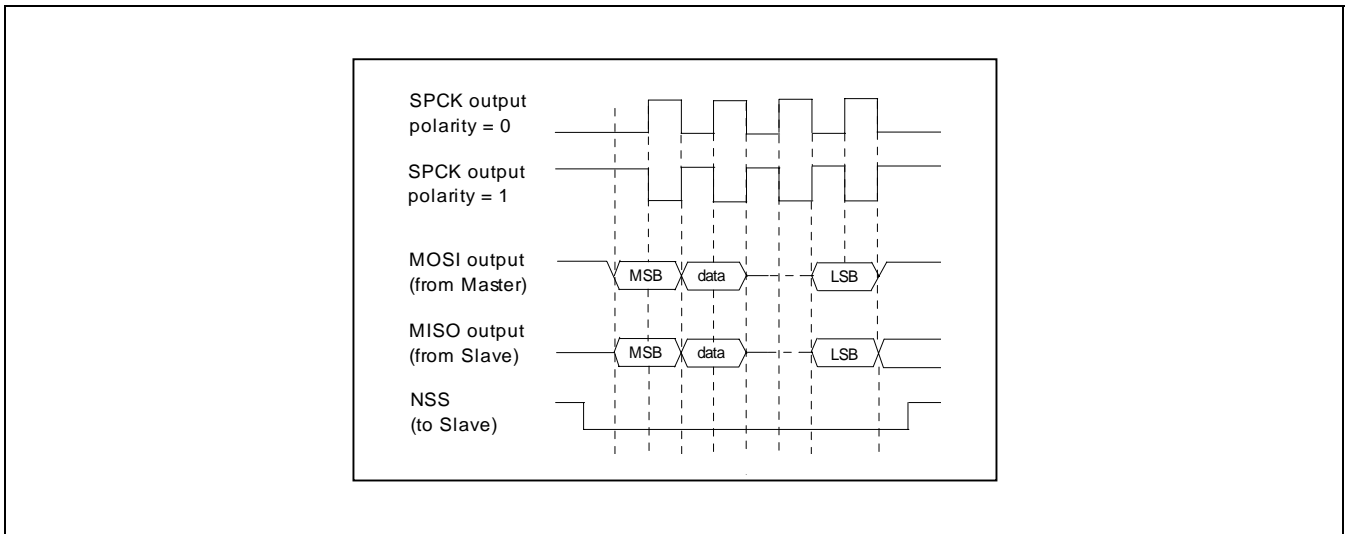


Figure 20-5. SPI16 in Master Mode, phase=1

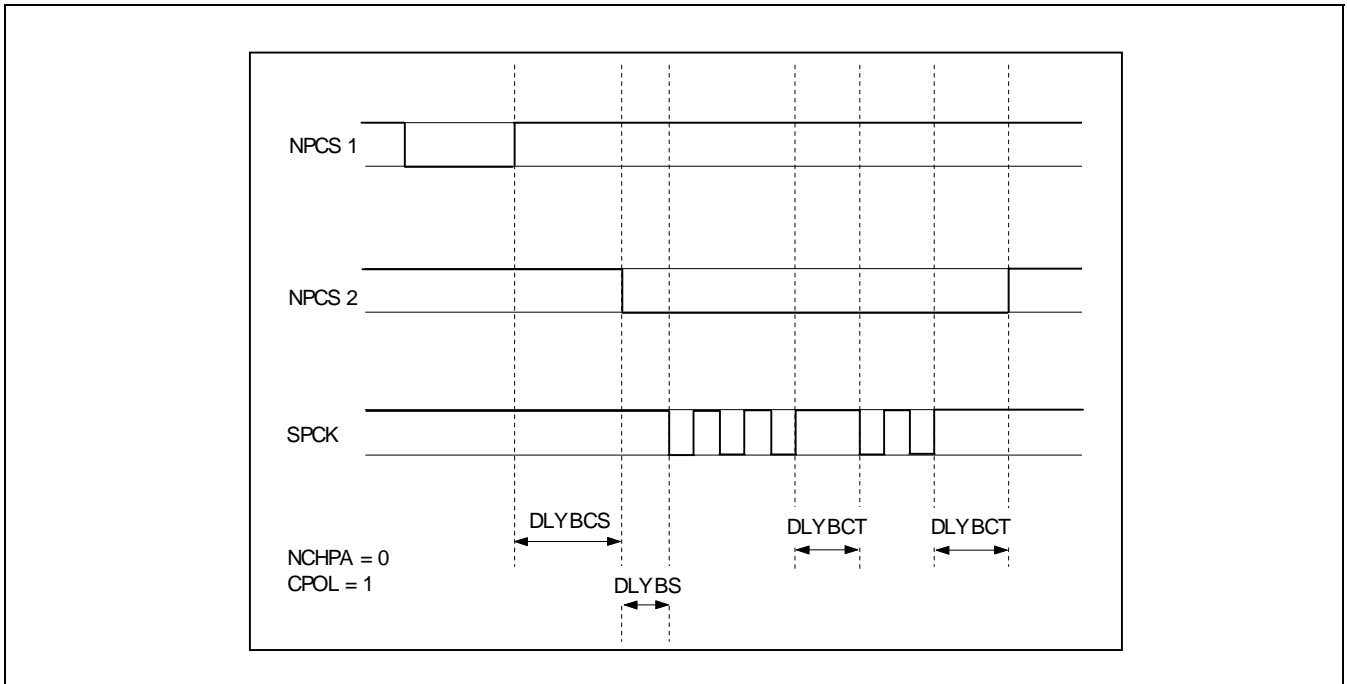


Figure 20-6. DLYBCS, DLYBS and DLYBCT

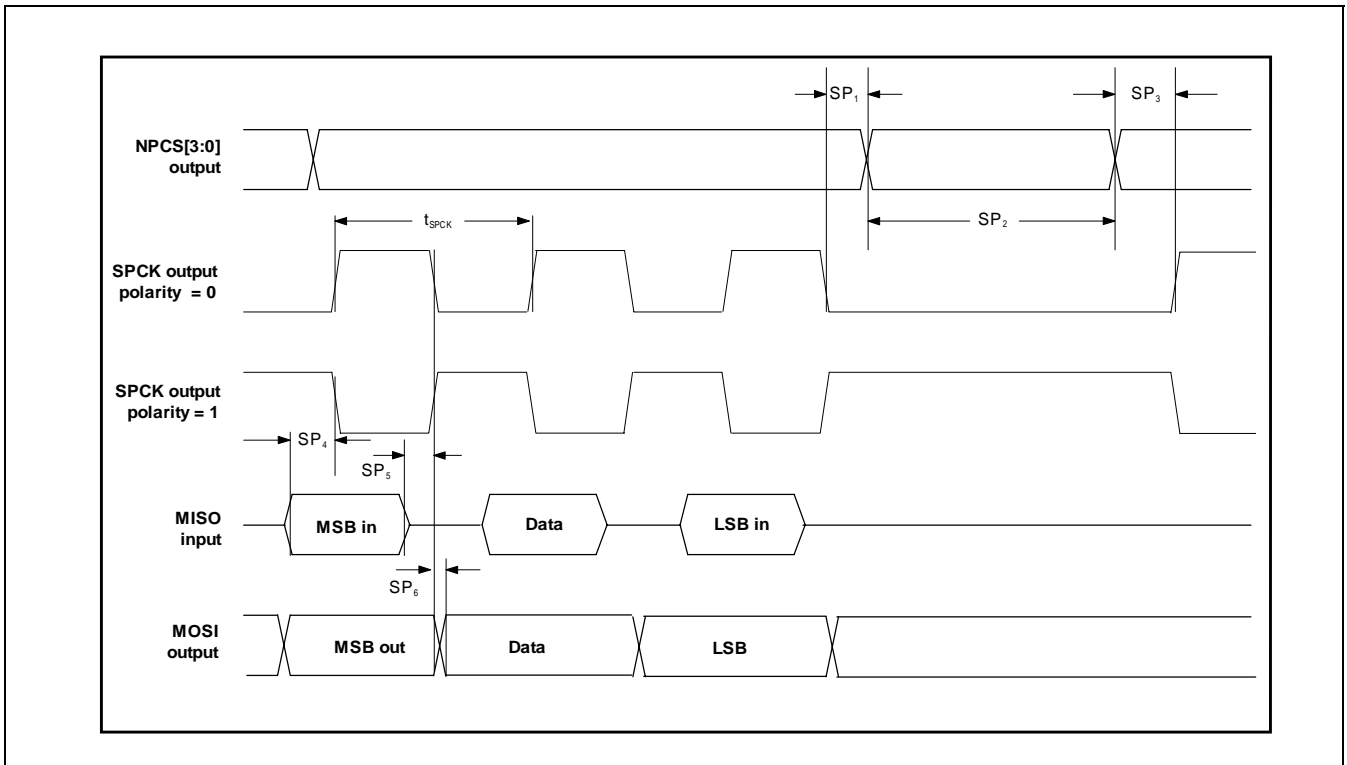


Figure 20-7. SPI16 Timings

4.2 TIMINGS VALUES

Table 20-2. SPI16 Timing Values

Symbol	Parameter	Minimum	Maximum	Unit
T_{SPCK}	SPI16 operating period	$4 \times (t_{CP})$	$16320 \times (t_{CP})$	ns
f_{SPCK}	SPI16 operating frequency	$1/16320 \times (t_{CP})$	$1/4 \times (t_{CP})$	MHz
SP_1	Delay before NPCS[3:0]	$4 \times (t_{CP})$	$261120 \times (t_{CP})$	ns
SP_2	Delay between chip select	$6 \times (t_{CP})$	$8160 \times (t_{CP})$	ns
SP_3	Delay before SPCK	$2 \times (t_{CP})$	$8160 \times (t_{CP})$	ns
SP_4	MISO/SPCK setup time	–	18	ns
SP_5	MOSI/SPCK hold time	0	–	ns
SP_6	MOSI valid after SPCK edge	–	7	ns

5. REGISTERS DESCRIPTION

Base Address – 0xFFE60000

Table 20-3. SPI16 Special Function Registers

Offset Address	Name	Description	R/W	Reset State
0x000 – 0x04C	–	Reserved	–	–
0x050	SPI_ECR	Enable clock register	W	–
0x054	SPI_DCR	Disable clock register	W	–
0x058	SPI_PMSR	Power management status register	R	0XXXXXXXX0
0x05C	–	Reserved	–	–
0x060	SPI_CR	Control register	W	0x00000000
0x064	SPI_MR	Mode register	R/W	0x00000000
0x068	–	Reserved	–	–
0x06C	SPI_CSR	Clear status register	W	0x00000000
0x070	SPI_SR	Status register	R	0x00000000
0x074	SPI_IER	Interrupt enable register	W	–
0x078	SPI_IDR	Interrupt disable register	W	–
0x07C	SPI_IMR	Interrupt mask register	R	0x00000000
0x080	SPI_RDR	Receive data register	R	–
0x084	SPI_TDR	Transmit data register	W	0x00000000
0x088	–	Reserved	–	–
0x08C	–	Reserved	–	–
0x090	SPI_SSR0	Slave select register 0	R/W	0x00000000
0x094	SPI_SSR1	Slave select register 1	R/W	0x00000000
0x098	SPI_SSR2	Slave select register 2	R/W	0x00000000
0x09C	SPI_SSR3	Slave select register 3	R/W	0x00000000

SPI Enable Clock Register

SPI_ECR (0x050)

Access: Write only

31	30	29	28	27	26	25	24
DBGEN	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	SPI	–
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **SPI : SPI clock enable**

0: No effect

1: Enable SPI clock

- **DBGEN : debug mode enable**

0: No effect

1: Enable debug mode

SPI Disable Clock Register

SPI_DCR (0x054)

Access: Write only

31	30	29	28	27	26	25	24
DBGEN	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	SPI	–
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **SPI : SPI clock disable**

0: No effect

1: Disable SPI clock

- **DBGEN : debug mode disable**

0: No effect

1: Disable debug mode

SPI Power Management Status Register

SPI_PMSR (0x058)

Access: Read only

31	30	29	28	27	26	25	24
DBGEN	–	IPIDCODE[25:20]					
R-0	R-0	R-U	R-U	R-U	R-U	R-U	R-U
23	22	21	20	19	18	17	16
IPIDCODE[19:12]							
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U
15	14	13	12	11	10	9	8
IPIDCODE[11:4]							
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U
7	6	5	4	3	2	1	0
IPIDCODE[3:0]				–	–	SPI	–
R-U	R-U	R-U	R-U	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **SPI : SPI clock status**

0: SPI clock disabled.

1: SPI clock enabled.

NOTE: The SPI_PMSR register is not reset by software reset.

- **IPIDCODE[25:0]**

This field contains the version number of the module, coded on 26 bits.

- **DBGEN : debug mode**

0: dbgack_sclk input has no influence on I2C function.

1: dbgack_sclk is enabled. When this signal is low, the I2C function is left unchanged, and when this signal is high, the I2C function is frozen. However, full read/write access to internal register is kept for the debug purpose.

SPI Control Register **SPI_CR (0x060)** **Access: Write only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	SPIDIS	SPIEN	SWRST
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **SWRST : SPI software reset**

0: No effect

1: Reset the SPI

A Software triggered hardware reset of the SPI is performed. It reset all the registers (except SPI_PMSR).

- **SPIEN : SPI enable**

0: No effect

1: Enable the SPI

- **SPIDIS : SPI disable**

0: No effect

1: Disable the SPI

All pins will be set in input mode and no data will be received or transmitted.

In case a transfer is in progress, the transfer will be finished before the SPI is disabled.

The SPI will be disabled in case both SPIEN and SPIDIS are equal to one.

SPI Mode Register **SPI_MR (0x064)** **Access: Read/Write**

31	30	29	28	27	26	25	24
DLYBCS[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
-				PCS[3:0]			
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
-							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
LLB	-	-	MODFEN	DIV32	PCSDEC	PS	MSTR
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

NOTE: When SPI16 is set to fixed mode and is communicated by CS1, CS2 or CS3, MODFEN bit must not be set.

- **MSTR : Master/Slave selection**

- 0: Slave Mode. (The SPI in slave mode can not be used with the LDMA for data transmission or reception.)
- 1: Master Mode.

- **PS : Peripheral selection** (This bit is only used in master mode.)

- 0: Fixed peripheral selection
- 1: Variable peripheral selection

This bit is only used in master mode. In case of fixed peripheral select, the PCS[3:0] is defined in the mode register(SPI_MR). For variable peripheral select, PCS[3:0] is defined in the transmit data register(SPI_TDR).

- **PCSDEC : Chip select decode** (This bit is only used in master mode.)

- 0: The chip selects are directly connected to a peripheral device.
- 1: The 4 chip select lines are connected to a 4 to 16 decoder.

This bit is only used in master mode. In case PCSDEC is one, up to 16 Chip Select signals can be generated with the 4 lines using an external 4 to 16 decoder.

The Chip Select Registers define the characteristics of the 16 chip selects according to the following rules :

- ⇒ SPI_CSR0 defines peripheral chip select signals 0 to 3
- ⇒ SPI_CSR1 defines peripheral chip select signals 4 to 7
- ⇒ SPI_CSR2 defines peripheral chip select signals 8 to 11
- ⇒ SPI_CSR3 defines peripheral chip select signals 12 to 15

- **DIV32 : Clock selection** (This bit is only used in master mode.)
 - 0: SPI Master Clock equals PCLK.
 - 1: SPI Master Clock equals PCLK/32.

- **MODFEN : Mode Fault detection enable / disable** (This bit is only used in master mode.)
 - 0: Disable Mode Fault detection
 - 1: Enable Mode Fault detection

- **LLB : Local loop back**
 - 0: Disable Local loop back path
 - 1: Enable Local loop back path

LLB controls the local loop back on the data serializer for testing.

- **PCS[3:0] : Peripheral chip select** (These bits are only used in master mode.)

This field is only used when peripheral selection(PS of SPI_MR) is 0.

In case PCSDEC = 0 :

PCS[3:0] = xxx0 => NPCS[3:0] = 1110

PCS[3:0] = xx01 => NPCS[3:0] = 1101

PCS[3:0] = x011 => NPCS[3:0] = 1011

PCS[3:0] = 0111 => NPCS[3:0] = 0111

PCS[3:0] = 1111 => forbidden (no peripheral is selected)

(x = don't care)

In case PCSDEC = 1 :

NPCS[3:0] => output signals = PCS[3:0]

- **DLYBCS[7:0] : Delay between chip selects** (These bits are only used in master mode.)

This field defines the delay from one NPCS inactive to the activation of another NPCS. The DLYBCS[7:0] time will guarantee non-overlapping chip selects and resolves bus contentions in case of peripherals with long data float times.

When DLYBCS[7:0] is less than 6, 6 SPI Master Clock period will be inserted by default.

Else, the following equation determines the delay :

$NPCS_to_SPCK_Delay = DLYBCS[7:0] * SPI_Master_Clock_period.$

SPI Clear Status Register

SPI_CSR (0x06C)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	ENDTRANS	–	–	OVRE	MODF	–	–
W	W	W	W	W	W	W	W

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **MODF : Clear mode fault error** (This bit is only used in master mode.)

0: No effect.

1: Clear the MODF interrupt.

- **OVRE : Clear overrun error**

0: No effect.

1: Clear the OVRE interrupt.

- **ENDTRANS : Clear End of Transfer(chip select is inactive) Only available in Master mode.**

0: No effect.

1: Clear the ENDTRANS bit in the interrupt status register

SPI Status Register **SPI_SR (0x070)** **Access: Read only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	BUSY	ENS
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
–	ENDTRANS	–	–	OVRE	MODF	TDRE	RDRF
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

NOTE: This register is a “read-active” register, which means that reading it can affect the state of some bits. When reading SPI_SR register, following bits are cleared if set : MODF, OVRE, REND, TEND, SPCK, MISO, MOSI, NPCS0, NPCS1, NPCS2 and NPCS3. When debugging, to avoid this behavior, users should use ghost registers.

- **RDRF : Receive data register full**

0: No data has been received since the last read of SPI_RDR.

1: A data has been received and the receive data has been transferred from the serializer in SPI_RDR since the last read of SPI_RDR.

NOTE: This bit is cleared when SPI_RDR is read.

- **TDRE : Transmit data register empty**

0: A data has been written in SPI_TDR and not yet transferred to the serializer.

1: The last data written in the Transmit Data Register has been transferred to the serializer.

TDRE equals zero when the SPI is disabled or at reset. The SPI enable command sets this bit to one.

NOTE: This bit is cleared when SPI_TDR is written.

- **MODF : Mode fault error** (This bit is only used in master mode.)

0: No Mode Fault has been detected since the last read of SPI_SR.

1: A Mode Fault occurred since the last read of SPI_SR.

- **OVRE : Overrun error**

0: No overrun has been detected since the last read of SPI_SR.

1: An overrun has occurred since the last read of SPI_SR.

An overrun occurs when SPI_RDR is loaded at least twice from the serializer since the last read of the SPI_RDR.

- **ENDTRANS : End of Transfer(chip select is inactive) Only available in Master mode.**

0: A transfer is in progress or no transfer since the last end transfer interrupt.

1: End of transfer, the chip select has just become inactive.

- **ENS : SPI Enable**

0: SPI is disabled.

1: SPI is enabled.

- **BUSY : SPI Busy in Master mode**

0: SPI is not busy.

1: SPI is busy in master mode.

SPI Interrupt Enable Register **SPI_IER (0x074)** **Access: Write only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	ENDTRANS	–	–	OVRE	MODF	TDRE	RDRF
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **RDRF : Receive Data Register Full interrupt enable**

0: No effect.

1: Enable the RDRF interrupt.

- **TDRE : Transmit Data Register Empty interrupt enable**

0: No effect.

1: Enable the TDRE interrupt.

- **MODF : Mode fault error interrupt enable** (This bit is only used in master mode.)

0: No effect.

1: Enable the MODF interrupt.

- **OVRE : Overrun error interrupt enable**

0: No effect.

1: Enable the OVRE interrupt.

- **ENDTRANS : End of Transfer(chip select is inactive) Only available in Master mode interrupt enable**

0: No effect.

1: Enable the ENDTRANS interrupt.

SPI Interrupt Disable Register

SPI_IDR (0x078)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	ENDTRANS	–	–	OVRE	MODF	TDRE	RDRF
W	W	W	W	W	W	W	W

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **RDRF : Receive Data Register Full interrupt disable**

0: No effect.

1: Disable the RDRF interrupt.

- **TDRE : Transmit Data Register Empty interrupt disable**

0: No effect.

1: Disable the TDRE interrupt.

- **MODF : Mode fault error interrupt disable** (This bit is only used in master mode.)

0: No effect.

1: Disable the MODF interrupt.

- **OVRE : Overrun error interrupt disable**

0: No effect.

1: Disable the OVRE interrupt.

- **ENDTRANS : End of Transfer(chip select is inactive) Only available in Master mode interrupt disable**

0: No effect.

1: Disable the ENDTRANS interrupt.

SPI Interrupt Mask Register

SPI_IMR (0x07C)

Access: Read only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
–	ENDTRANS	–	–	OVRE	MODF	TDRE	RDRF
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **RDRF : Receive data register full interrupt mask**

0: Receive Data Register Full Interrupt is disabled.

1: Receive Data Register Full Interrupt is enabled.

- **TDRE : Transmit data register empty interrupt mask**

0: Transmit Data Register Empty Interrupt is disabled.

1: Transmit Data Register Empty Interrupt is enabled.

- **MODF : Mode fault error interrupt mask** (This bit is only used in master mode.)

0: Mode Fault Interrupt is disabled.

1: Mode Fault Interrupt is enabled.

Only used in Master Mode.

- **OVRE : Overrun error interrupt mask**

0: Overrun Error Interrupt is disabled.

1: Overrun Error Interrupt is enabled.

- **ENDTRANS : End of Transfer(chip select is inactive) Only available in Master mode interrupt mask**

0: Data Transfer End interrupt is disabled.

1: Data Transfer End interrupt is enabled.

SPI Receive Data Register

SPI_RDR (0x080)

Access: Read only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	PCS[3:0]			
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
RD[15:8]							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
RD[7:0]							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset***NOTES:**

1. When reading this register, RDRF bit is clear in the SPI_RDR.
2. When debugging, to avoid clearing RDRF bit, users should use ghost registers.

- **RD[15:0] : Receive data**

Data received by the SPI interface is stored in this register. Data stored is right justified. Unused bits are read at zero.

- **PCS[3:0] : Peripheral chip select status** (These bits are only used in master mode.)

In Master Mode only, these bits indicate the value on the NPCS pins at the end of a transfer.

SPI Transmit Data Register **SPI_TDR (0x084)** **Access: Write only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	PCS[3:0]			
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
TD[15:8]							
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
TD[7:0]							
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **TD[15:0] : Transmit data**

Data that is to be transmitted by the SPI Interface is stored in this register. Information to be transmitted must be written to the transmit data register in a right-justified format.

- **PCS[3:0] : Peripheral chip select** (These bits are only used in master mode.)

This field is only used when peripheral selection(PS of SPI_MR) is 1.

In case PCSDEC = 0 :

PCS[3:0] = xxx0 => NPCS[3:0] = 1110

PCS[3:0] = xx01 => NPCS[3:0] = 1101

PCS[3:0] = x011 => NPCS[3:0] = 1011

PCS[3:0] = 0111 => NPCS[3:0] = 0111

PCS[3:0] = 1111 => forbidden (no peripheral is selected)

(x = don't care)

In case PCSDEC = 1 :

NPCS[3:0] => output signals = PCS[3:0]

SPI Slave Select Register 0
 SPI Slave Select Register 1
 SPI Slave Select Register 2
 SPI Slave Select Register 3

SPI_SSR0 (0x080)
 SPI_SSR1 (0x088)
 SPI_SSR2 (0x090)
 SPI_SSR3 (0x098)

Access: Read/Write

31	30	29	28	27	26	25	24
DLYBCT[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
DLYBS[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
SCBR[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
BITS[3:0]				-	-	NCPHA	CPOL
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **CPOL : Clock Polarity**

0: The inactive state value of SPCK is logic level zero.

1 : The inactive state value of SPCK is logic level one.

CPOL is used to determine the inactive state value of the serial clock (SPCK). It is used with NCPHA to produce a desired clock/data relationship between master and the slave devices.

- **NCPHA : Clock Phase**

0: Data is changed on the leading edge of SPCK and captured on the following edge of SPCK.

1: Data is captured on the leading edge of SPCK and changed on the following edge of SPCK.

NCPHA determines which edge of SPCK causes data to change and which edge causes data to be captured.

NCPHA is used with CPOL to produce a desired clock/data relationship between master and slave devices.

NOTE: The shape of clock related to CPOL, NCPHA is depicted in the Figure 20-4, 21-5. (Page 20-7)

- **BITS[3:0] : Bits Per Transfer**

Table 20-4. SPI Bits Per Transfer

BITS[3:0]	Bits Per Transfer
0000	8
0001	9
0010	10
0011	11
0100	12
0101	13
0110	14
0111	15
1000	16
1001	Reserved
1010	Reserved
1011	Reserved
11XX	Reserved

The BITS field determines the number of data bits transferred. Reserved values default to 8 bits.

- **SCBR[7:0] : Serial Clock Baud Rate** (These bits are only used in master mode.)

The SPI interface uses a modulus counter to derive SPCK baud rate from the SPI Master Clock, selected between PCLK and PCLK/32. Baud rate is selected by writing a value from 2 to 255 into SCBR[7:0]. The following equation determines the SPCK baud rate :

$$\text{SPCK_Baud_Rate} = \text{SPI_Master_Clock_frequency} / (2 \times \text{SCBR}[7:0])$$

NOTE: Giving SCBR[7:0] a value of zero or one disables the baud rate generator. SPCK is disabled and assumes its inactive state value. No serial transfers occur. At reset, baud rate is disabled.

Table 20-5. Baudrate

SPI_master_clock	Baudrate	SCBR[7:0]	Error
4 MHz	125K	0x10	0%
	250K	0x08	0%
	500K	0x04	0%
	1M	0x02	0%
20 MHz	125K	0x50	0%
	250K	0x28	0%
	500K	0x14	0%
	1 MHz	0x0A	0%
	4 MHz	0x03	16.6%
40 MHz	125K	0xA0	0%
	250K	0x50	0%
	500K	0x28	0%
	1 MHz	0x14	0%
	4 MHz	0x05	0%

- **DLYBS[7:0] : Delay Before SPCK** (These bits are only used in master mode.)

This field defines the length of delay from NPCS valid to the first valid SPCK transition.

When DLYBS[7:0] equals zero, the NPCS valid to SPCK transition is one-half SPCK clock period.

Else, the following equation determines the delay :

$$\text{NPCS_to_SPCK_Delay} = \text{DLYBS}[7:0] \times \text{SPI_Master_Clock_period.}$$

- **DLYBCT[7:0] : Delay Between Consecutive Transfers** (These bits are only used in master mode.)

This field defines the delay between 2 consecutive transfers with the same peripheral, without removing the slave select.

When DLYBCT[7:0] equals zero, the delay between consecutive transfers is one-half SPCK clock period.

Else, the following equation determines the delay :

$$\text{Delay_between_consecutive_transfers} = \text{DLYBCT}[7:0] \times \text{SPI_Master_Clock_period.}$$

21 SERIAL PERIPHERAL INTERFACE 8-BIT (SPI8)

1. OVERVIEW

1.1 FUNCTIONAL DESCRIPTION

The SPI8 module provides a serial synchronous duplex communication with external devices such as microcontroller or peripheral. This module has one slave select and can transfer from 1 to 8 bits data.

The main features are :

- 1 to 8 bits transfer
- 4 pins : NSS0, MISO0, MOSI0 and SPCK0
- Master / Slave mode
- Programmable baud rate generator
- Serial clock with programmable polarity and phase
- Local loop back mode
- Interrupt generation
- 2 dedicated LDMA channels

1.1.1 Pin Description

Four pins are associated with the SPI8 interface. When not needed for the SPI8 function, each of these pins can be configured as a PIO, using PIO Controller functions.

NOTES:

1. When PIO block is included, after a hardware reset the pins are controlled by the PIO. Even the NSS Parallel I/O Controller for multi-master operation is provided by the PIO Controller. Multi-Driver option : to configure a SPI8 pin as open-drain to support external drivers, set the corresponding bits in the PIO_MDSR register. The NSS pin can function as a peripheral chip select output or slave select input.
2. A pull up resistor must be connected on the corresponding pin configured in open-drain mode.

1.2 BLOCK DIAGRAM

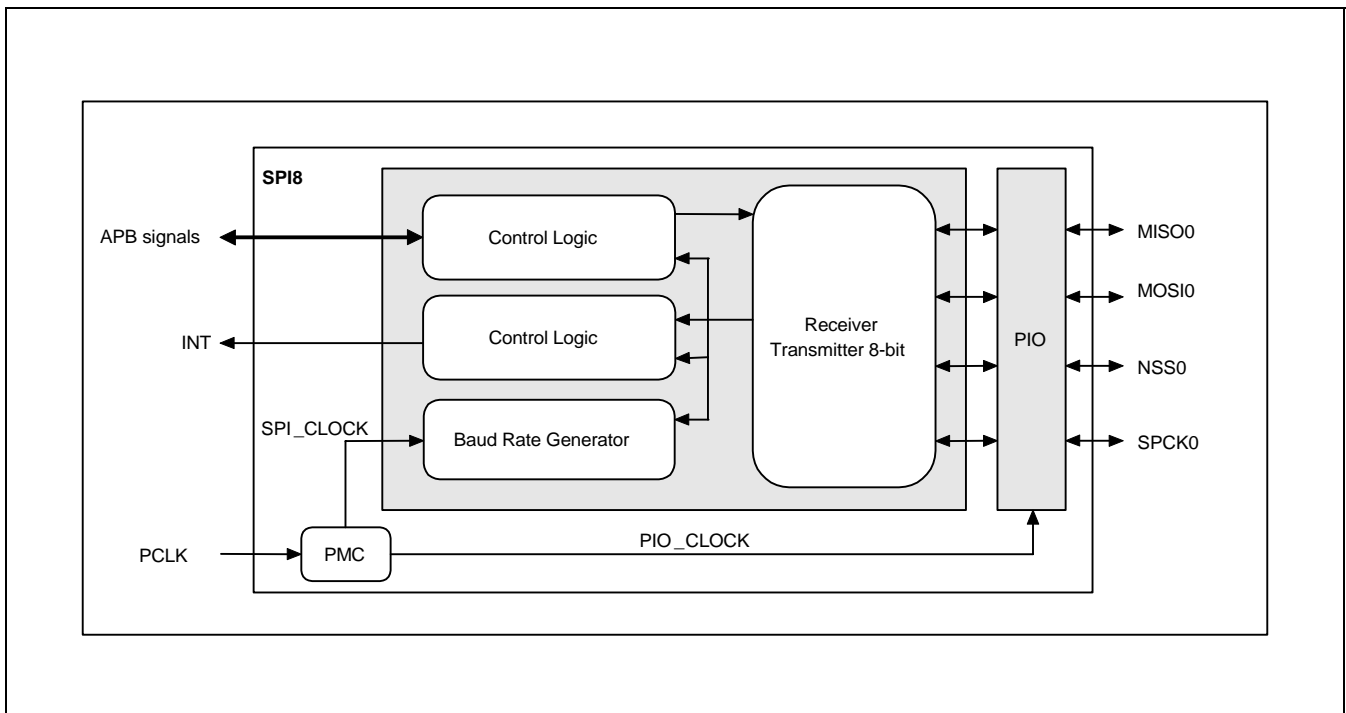


Figure 21-1. SPI8 Block Diagram

2. EXTERNAL PIN DESCRIPTION

Table 21-1. SPI8 Pin Description

Pin Name	Function	I/O Type	Active Level	Comments
SPCK0	SPI clock	I/O	–	–
MISO0	Master in slave out	I/O	–	–
MOSI0	Master out slave in	I/O	–	–
NSS0	Peripheral chip select	I/O	Low	–

3. FUNCTIONAL OPERATION

3.1 MASTER MODE

3.1.1 General Description

In Master Mode, the SPI8 controls data transfers to and from the slave connected to the SPI8 bus. The SPI8 drives the chip select (NSS) to the slave and the serial clock (SPCK). After enabling the SPI8, a data transfer begins when the ARM core writes to the Transmit Data Register (SPI_TDR).

Transmit and Receive buffers maintain the data flow at a constant rate with a reduced requirement for high priority interrupt servicing. As long as new data is available in the Transmit Data Register (SP_TDR) the SPI8 continues to transfer data. If the Receive Data Register (SPI_RDR) has not been read before new data is received, the Overrun Error (SPIOVRE) flag is set.

The delay between the activation of the NSS chip select and the start of the data transfer (DLYBS) as well as the delay between each data transfer (DLYBCT) can be programmed. All data transfer characteristics including the two timing values are programmed in registers SPI_SSR (Slave Select Registers).

NOTE

When multi-masters are used, it is necessary to configure SPI8 NSS pin in open drain and set a pull up resistor on it. This configuration is needed to allow the SPI8 configured in master mode to detect a mode fault when a low level is driven by an external master on the NSS signal.

3.1.2 Slave Select

The Slave Select (NSS) is driven by the SPI8 only if it is programmed in Master Mode. This line is used to select the external device.

Once the transmission is enabled, the Slave Select (NSS) is driven low by the SPI8 and the length of the signal will take :

One data transmitted :

$DLYBS + \text{Number of bit per transfer} \times Tbit + DLYBCT$

Several data transmitted :

$DLYBS + \text{Number of data transmitted} \times (\text{Number of bit per transfer} \times Tbit + DLYBCT)$

3.1.3 Mode Fault Detection

By default, the mode fault detection is disabled.

This option can be enabled by setting the bit MODFEN in the SPI_MR register for multi master operation.

A mode fault is detected when the SPI8 is programmed in Master Mode and a low level is driven by an external master on the NSS signal.

DLYBCT is also used to specify the sampling rate of the mode fault detection signal after each end of transfer.

According to DLYBCT value, the sampling of the Mode fault detection signal starts (0.5, 1, 2 or 4)Tbit after the rising edge of the chip select.

When a mode fault is detected, the MODF bit in the SPI_SR is set until the SPI_CSR is programmed to clear it and the SPI8 is disabled until a re-enabled is made by setting the bit SPIEN in the SPI_CR (Control Register).

3.1.4 Functional Flow Chart in Master Mode

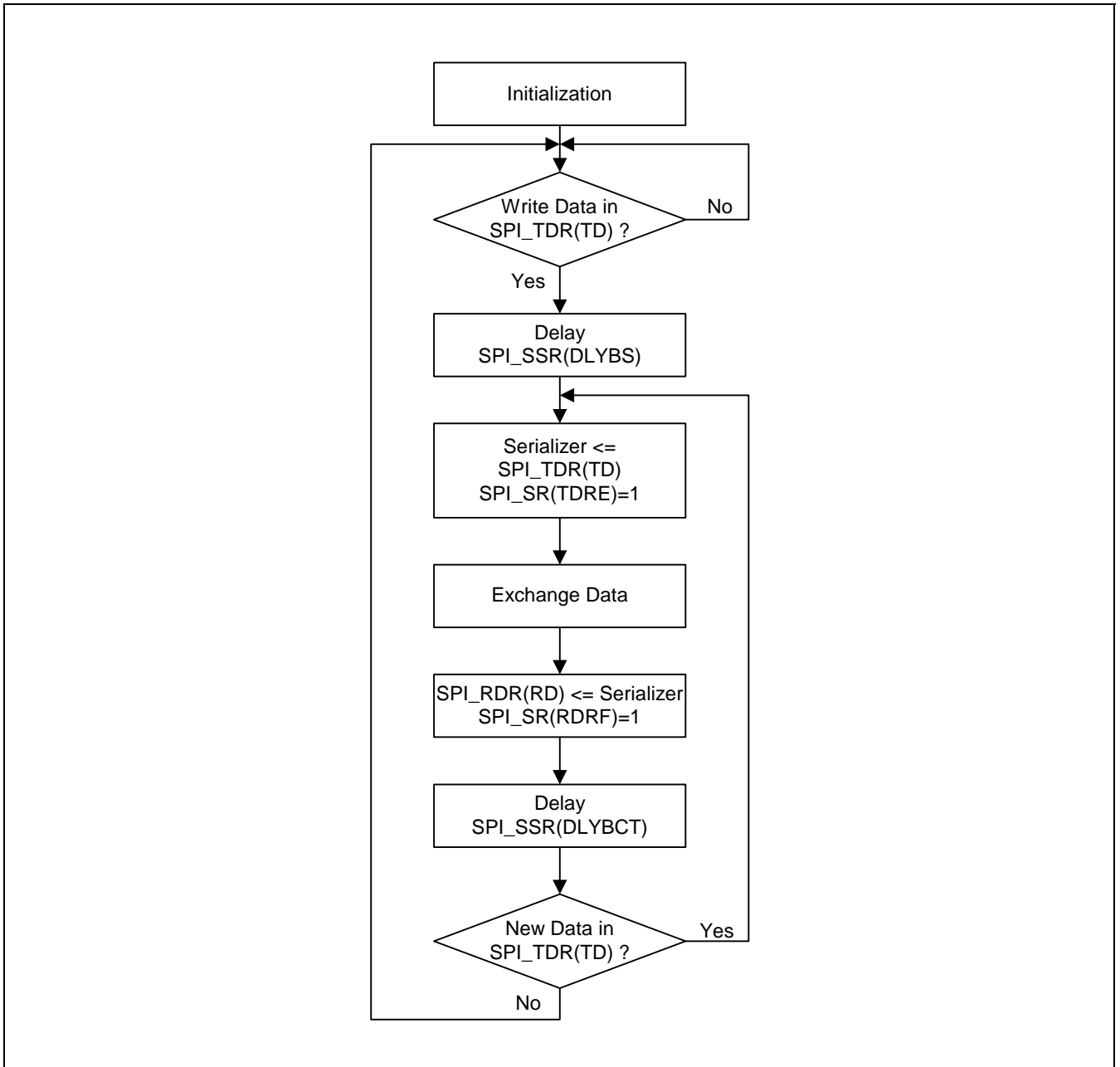


Figure 21-2. SPI8 Flow Chart in Master Mode

3.1.5 SPI8 Configured as a Master

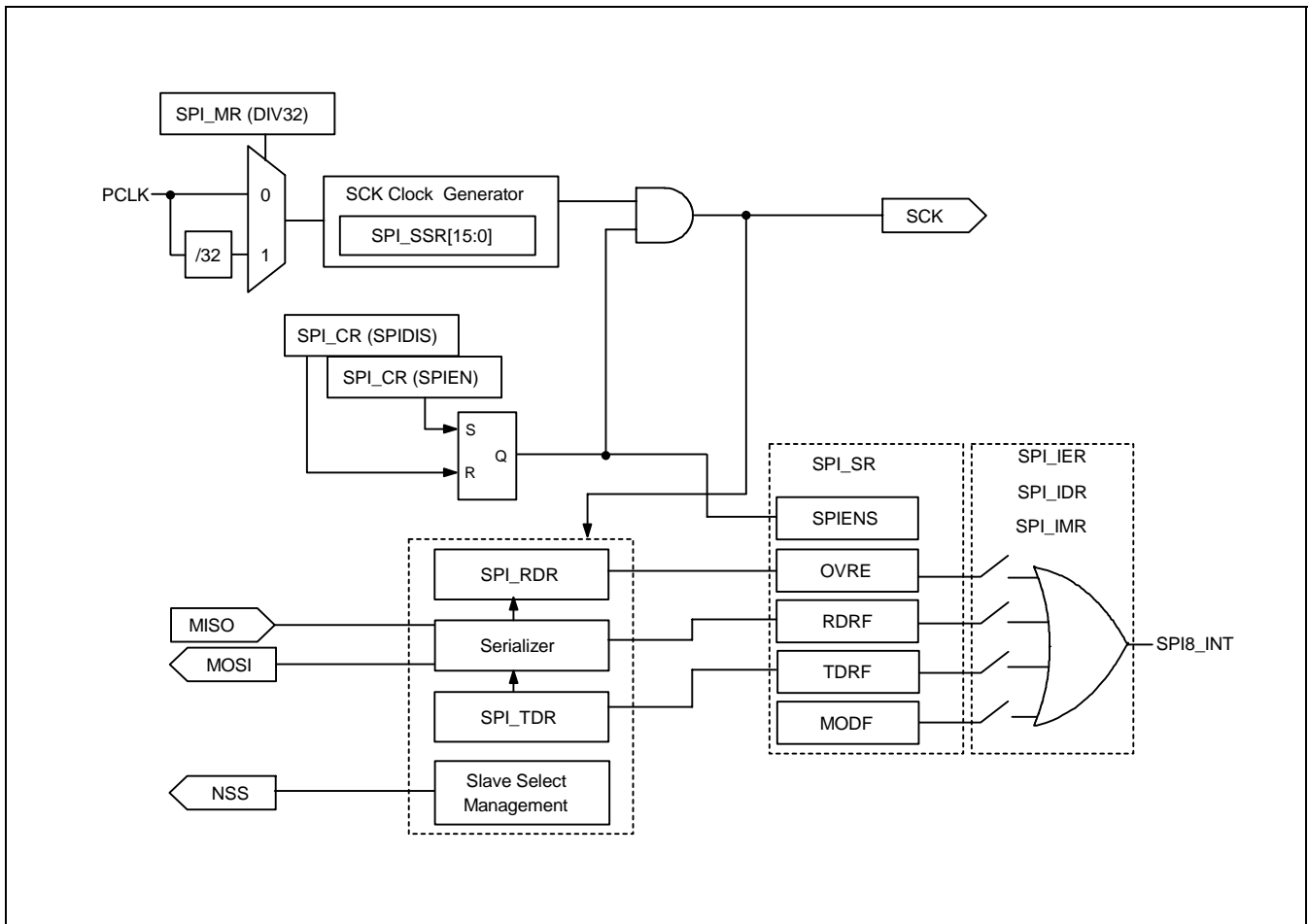


Figure 21-3. SPI8 in Master Mode

3.2 SLAVE MODE

3.2.1 General Description

In slave mode CPOL, NCPHA fields of SPI_SSR are used to define the transfer characteristics. The other fields in SPI_SSR Register are not used in slave mode.

In Slave Mode, the SPI8 waits for NSS to go active low before receiving the serial clock from an external master to be able to receive or send data.

The slave transmission data length will depend on the master transmission data length. If the SPI8 is configured in slave mode and wants to respond 3 characters, it is necessary that the master send 3 characters to the SPI8 in order to receive the full length.

3.2.2 SPI8 Configured as a Slave

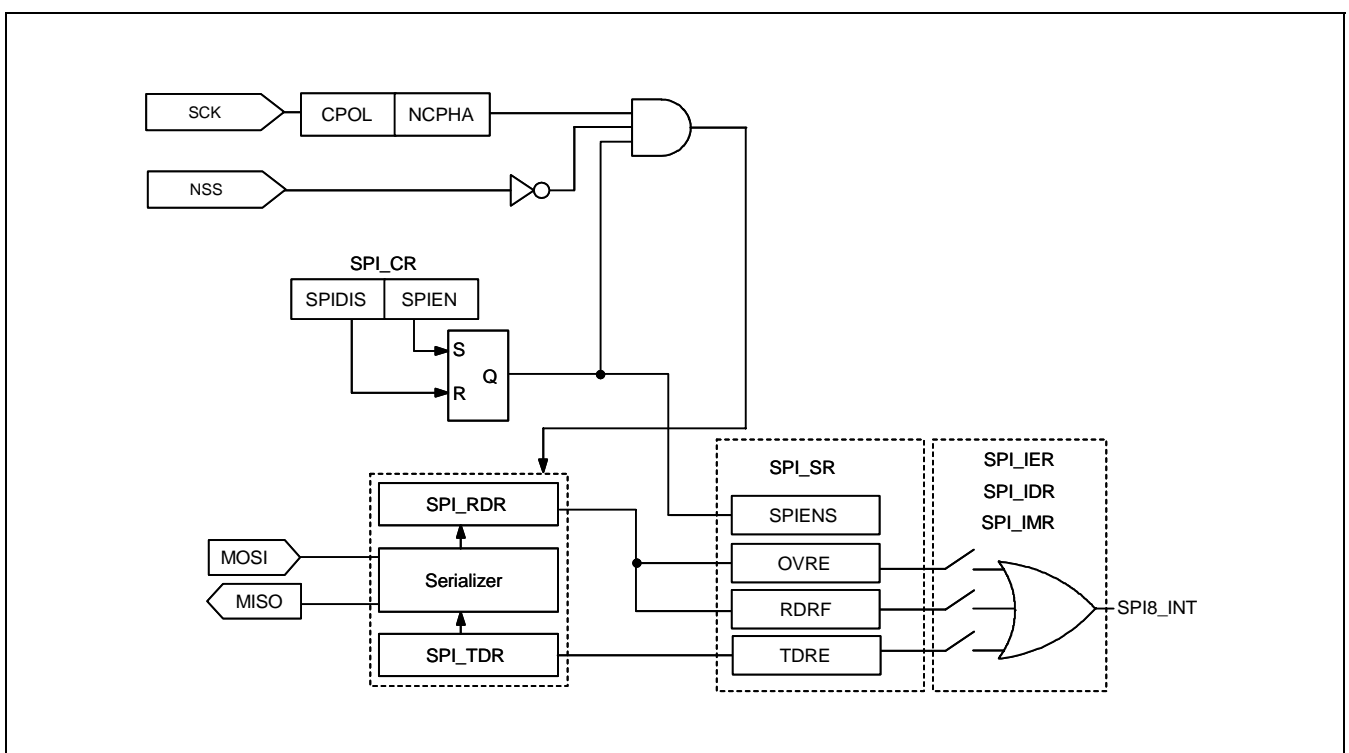


Figure 21-4. SPI8 in Slave Mode

4. TIMINGS

4.1 TIMINGS MODULE

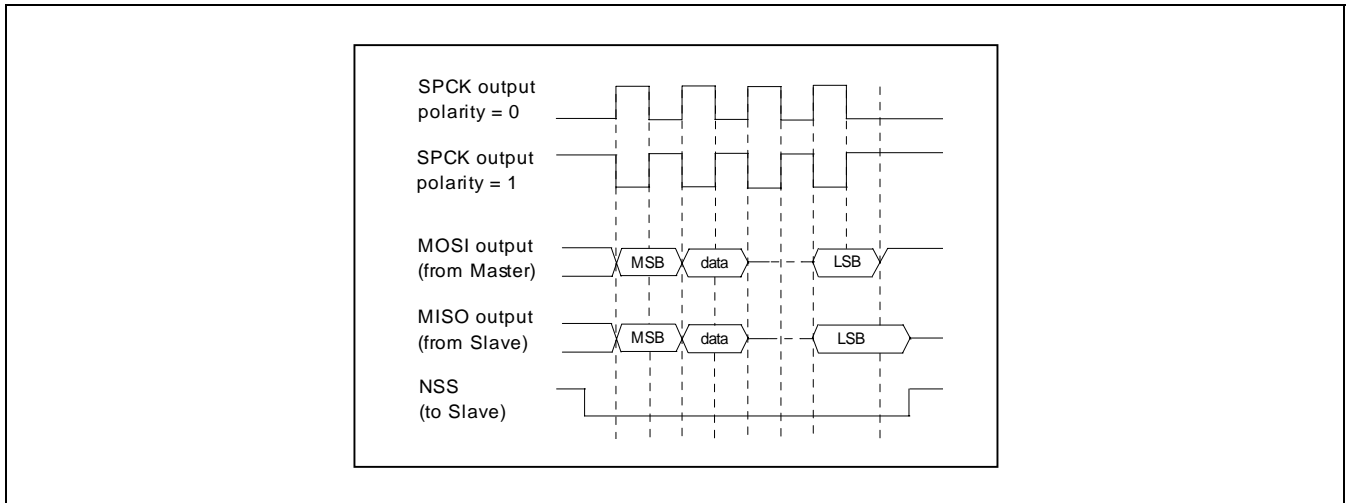


Figure 21-5. SPI8 in Master Mode, phase=0

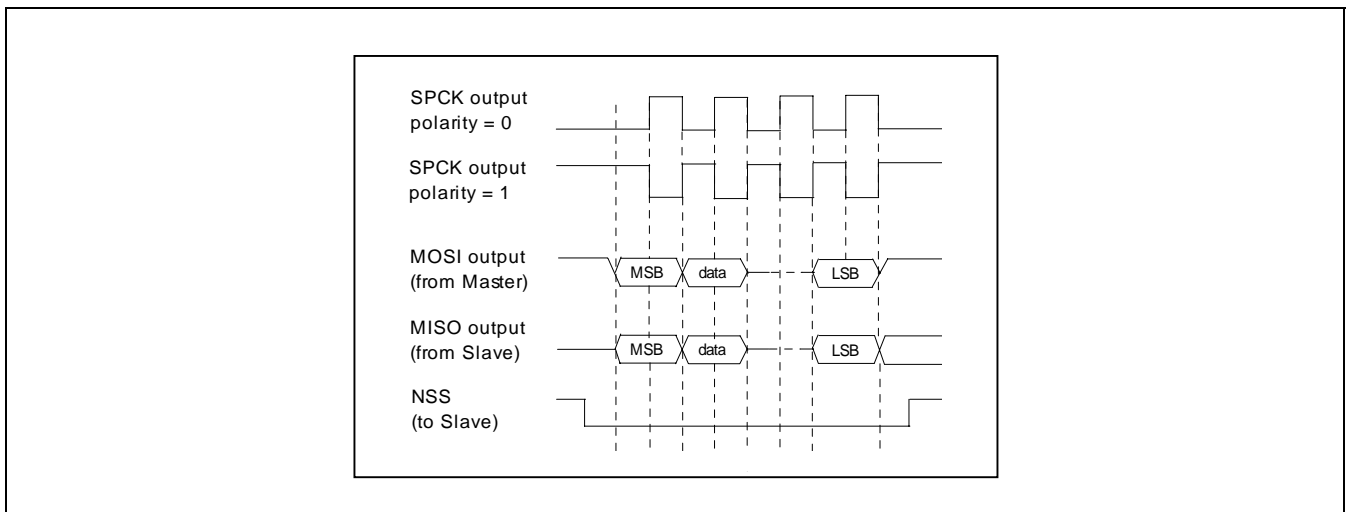


Figure 21-6. SPI8 in Master Mode, phase=1

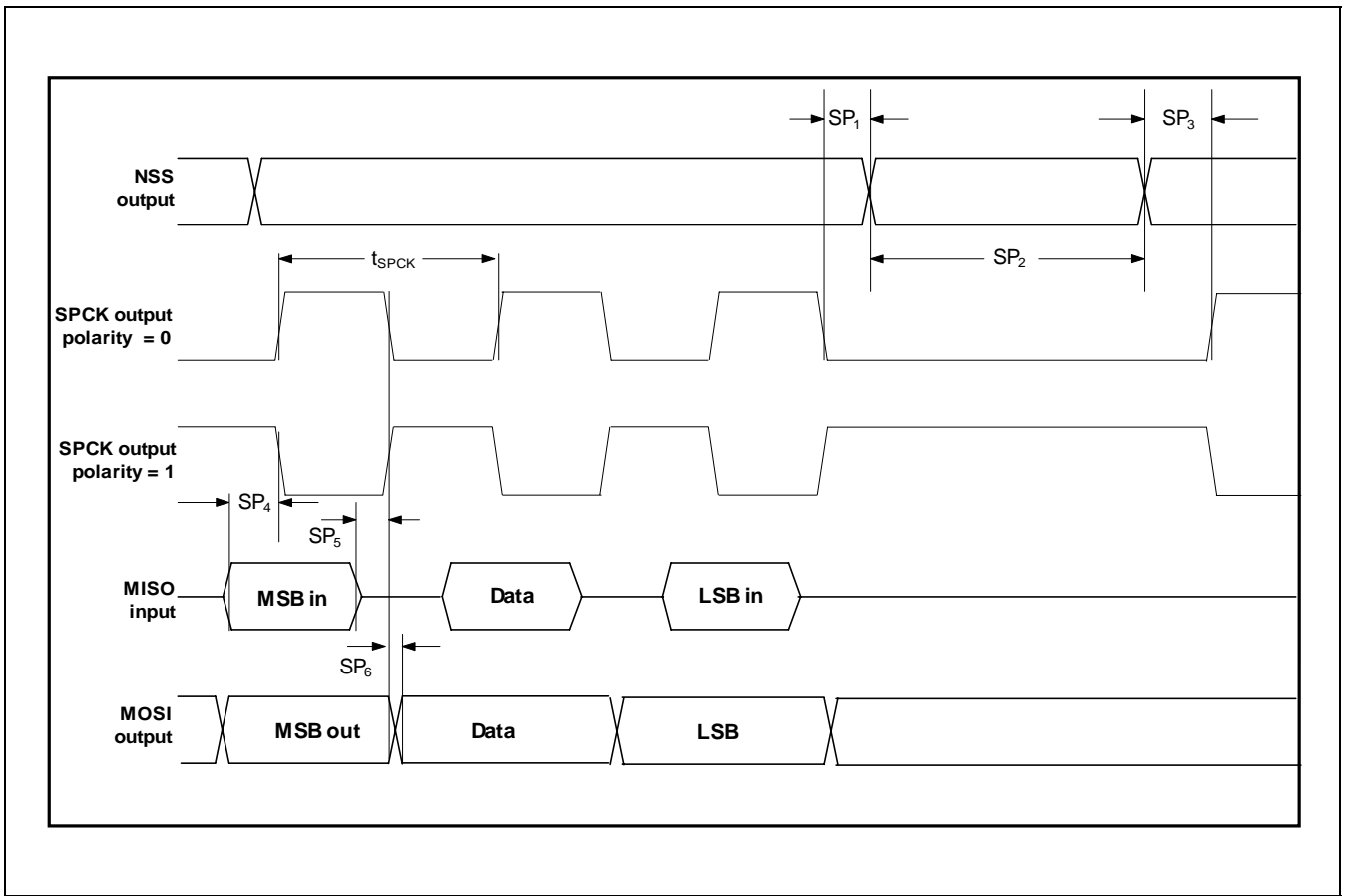


Figure 21-7. SPI8 Timings

5. REGISTERS DESCRIPTION

Base Address – 0xFFE10000

Table 21-3. SPI8 Special Function Registers

Offset Address	Name	Description	R/W	Reset State
0x000 – 0x04C	–	Reserved	–	–
0x050	SPI_ECR	Enable clock register	W	–
0x054	SPI_DCR	Disable clock register	W	–
0x058	SPI_PMSR	Power management status register	R	0XXXXXXXX0
0x05C	–	Reserved	–	–
0x060	SPI_CR	Control register	W	0x00000000
0x064	SPI_MR	Mode register	R/W	0x00000000
0x068	–	Reserved	–	–
0x06C	SPI_CSR	Clear status register	W	0x00000000
0x070	SPI_SR	Status register	R	0x00000000
0x074	SPI_IER	Interrupt enable register	W	–
0x078	SPI_IDR	Interrupt disable register	W	–
0x07C	SPI_IMR	Interrupt mask register	R	0x00000000
0x080	SPI_RDR	Receive data register	R	–
0x084	SPI_TDR	Transmit data register	W	0x00000000
0x088	–	Reserved	–	–
0x08C	–	Reserved	–	–
0x090	SPI_SSR	Slave select register 0	R/W	0x00000000

SPI Enable Clock Register

SPI_ECR (0x050)

Access: Write only

31	30	29	28	27	26	25	24
DBGEN	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	SPI	–
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **SPI : SPI clock enable**

0: No effect

1: Enable SPI clock

- **DBGEN : Debug mode enable**

0: No effect

1: Enable debug mode

SPI Disable Clock Register

SPI_DCR (0x054)

Access: Write only

31	30	29	28	27	26	25	24
DBGEN	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	SPI	–
W	W	W	W	W	W	W	W

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **SPI : SPI clock disable**

0: No effect

1: Disable SPI clock

- **DBGEN : Debug mode disable**

0: No effect

1: Disable debug mode

SPI Power Management Status Register

SPI_PMSR (0x058)

Access: Read only

31	30	29	28	27	26	25	24
DBGEN	–	IPIDCODE[25:20]					
R-0	R-0	R-U	R-U	R-U	R-U	R-U	R-U
23	22	21	20	19	18	17	16
IPIDCODE[19:12]							
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U
15	14	13	12	11	10	9	8
IPIDCODE[11:4]							
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U
7	6	5	4	3	2	1	0
IPIDCODE[3:0]				–	–	SPI	–
R-U	R-U	R-U	R-U	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **SPI : SPI clock status**

0: SPI clock disabled.

1: SPI clock enabled.

NOTE: The SPI_PMSR register is not reset by software reset.

- **IPIDCODE[25:0]**

This field contains the version number of the module, coded on 26 bits.

- **DBGEN : Debug mode**

0: Debug mode disabled.

1: Debug mode enabled.

NOTE: When enable, the SPI will enter in debug mode on core debug acknowledge.

SPI Control Register **SPI_CR (0x060)** **Access: Write only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	SPIDIS	SPIEN	SWRST
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **SWRST : SPI software reset**

0: No effect

1: Reset the SPI

A Software triggered hardware reset of the SPI is performed. It reset all the registers (except SPI_PMSR).

- **SPIEN : SPI enable**

0: No effect

1: Enable the SPI

- **SPIDIS : SPI disable**

0: No effect

1: Disable the SPI

All pins will be set in input mode and no data will be received or transmitted.

In case a transfer is in progress, the transfer will be finished before the SPI is disabled.

In case both SPIEN and SPIDIS are equal to one when the control register is written the SPI will be disabled.

NOTE: The bit ENS of the SPI_SR register indicates if the SPI is enabled or disabled.

SPI Mode Register

SPI_MR (0x064)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
LLB	–	–	MODFEN	DIV32	–	–	MSTR
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **MSTR : Master/Slave mode selection**

0: Slave mode (The SPI in slave mode can not be used with the LDMA for data transmission or reception.)

1: Master mode

- **DIV32 : Clock selection** (This bit is only used in master mode.)

0: SPI Master Clock = PCLK

1: SPI Master Clock = PCLK/32

- **MODFEN : Mode Fault detection enable / disable** (This bit is only used in master mode.)

0: Disable Mode Fault detection

1: Enable Mode Fault detection

- **LLB : Local Loop Back**

0: Disable Local Loop Back path

1: Enable Local Loop Back path

LLB controls the local loop back on the data serializer for testing.

SPI Clear Status Register

SPI_CSR (0x06C)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	ENDTRANS	–	–	OVRE	MODF	–	–
W	W	W	W	W	W	W	W

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **MODF : Clear mode fault error** (This bit is only used in master mode.)

0: No effect.

1: Clear the MODF interrupt.

- **OVRE : Clear overrun error**

0: No effect.

1: Clear the OVRE interrupt.

- **ENDTRANS : Clear end of transfer(chip select is inactive) only available in master mode.**

0: No effect.

1: Clear the ENDTRANS bit in the interrupt status register

SPI Status Register

SPI_SR (0x070)

Access: Read only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	BUSY	ENS
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
–	ENDTRANS	–	–	OVRE	MODF	TDRE	RDRF
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

NOTE: This register is a “read-active” register, which means that reading it can affect the state of some bits. When reading SPI_SR register, following bits are cleared if set : MODF, OVRE, REND, TEND, SPCK, MISO, MOSI, NPCS0, NPCS1, NPCS2 and NPCS3. When debugging, to avoid this behavior, users should use ghost registers.

- **RDRF : Receive data register full**

0: No data has been received since the last read of SPI_RDR.

1: A data has been received and the receive data has been transferred from the serializer in SPI_RDR since the last read of SPI_RDR.

NOTE: This bit is cleared when SPI_RDR is read.

- **TDRE : Transmit data register empty**

0: A data has been written in SPI_TDR and not yet transferred to the serializer.

1: The last data written in the Transmit Data Register has been transferred to the serializer.

TDRE equals zero when the SPI is disabled or at reset. The SPI enable command sets this bit to one.

NOTE: This bit is cleared when SPI_TDR is written.

- **MODF : Mode fault error** (This bit is only used in master mode.)

0: No Mode Fault has been detected since the last read of SPI_SR.

1: A Mode Fault occurred since the last read of SPI_SR.

- **OVRE : Overrun error**

0: No overrun has been detected since the last read of SPI_SR.

1: An overrun has occurred since the last read of SPI_SR.

An overrun occurs when SPI_RDR is loaded at least twice from the serializer since the last read of the SPI_RDR.

- **ENDTRANS : End of transfer(chip select is inactive) only available in master mode**

0: A transfer is in progress or no transfer since the last end transfer interrupt

1: End of transfer, the chip select has just become inactive.

- **ENS : SPI status**

0: SPI is disabled.

1: SPI is enabled.

- **BUSY : SPI Busy in Master mode**

0: SPI is not busy.

1: SPI is busy in master mode.

SPI Interrupt Enable Register

SPI_IER (0x074)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	ENDTRANS	–	–	OVRE	MODF	TDRE	RDRF
W	W	W	W	W	W	W	W

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **RDRF : Receive data register full interrupt enable**

0: No effect

1: Enable the RDRF interrupt

- **TDRE : Transmit data register empty interrupt enable**

0: No effect

1: Enable the TDRE interrupt

- **MODF : Mode fault error interrupt enable** (This bit is only used in master mode.)

0: No effect

1: Enable the MODF interrupt

- **OVRE : Overrun error interrupt enable**

0: No effect

1: Enable the OVRE interrupt

- **ENDTRANS : End of Transfer(chip select is inactive) Only available in Master mode enable**

0: No effect

1: Enable the ENDTRANS interrupt

SPI Interrupt Disable Register

SPI_IDR (0x078)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	ENDTRANS	–	–	OVRE	MODF	TDRE	RDRF
W	W	W	W	W	W	W	W

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **RDRF : Receive data register full interrupt disable**

0: No effect

1: Disable the RDRF interrupt

- **TDRE : Transmit data register empty interrupt disable**

0: No effect

1: Disable the TDRE interrupt

- **MODF : Mode fault error interrupt disable** (This bit is only used in master mode.)

0: No effect

1: Disable the MODF interrupt

- **OVRE : Overrun error interrupt disable**

0: No effect

1: Disable the OVRE interrupt

- **ENDTRANS : End of Transfer(chip select is inactive) Only available in Master mode disable**

0: No effect

1: Disable the ENDTRANS interrupt

SPI Interrupt Mask Register

SPI_IMR (0x07C)

Access: Read only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
–	ENDTRANS	–	–	OVRE	MODF	TDRE	RDRF
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **RDRF : Receive data register full interrupt mask**

0: Receive Data Register Full Interrupt is disabled.

1: Receive Data Register Full Interrupt is enabled.

- **TDRE : Transmit data register empty interrupt mask**

0: Transmit Data Register Empty Interrupt is disabled.

1: Transmit Data Register Empty Interrupt is enabled.

- **MODF : Mode fault error interrupt mask** (This bit is only used in master mode.)

0: Mode Fault Interrupt is disabled.

1: Mode Fault Interrupt is enabled.

Only used in Master Mode.

- **OVRE : Overrun error interrupt mask**

0: Overrun Error Interrupt is disabled.

1: Overrun Error Interrupt is enabled.

- **ENDTRANS : Clear End of Transfer(chip select is inactive) Only available in Master mode.**

0: Data Transfer End interrupt is disabled.

1: Data Transfer End interrupt is enabled.

SPI Receive Data Register

SPI_RDR (0x080)

Access: Read only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
RD[7:0]							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- RD[7:0] : Received data**

Data received by the SPI8 interface are stored in this register. Data stored are right justified.

SPI Transmit Data Register

SPI_TDR (0x084)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
TD[7:0]							
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- TD[7:0] : Transmit data**

Data that is to be transmitted by the SPI8 interface is stored in this register. Information to be transmitted must be written to the transmit data register in a right-justified format.

SPI Slave Select Register **SPI_SSR (0x090)** **Access: Read/Write**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	DLYBCT[1:0]	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	DLYBS[1:0]	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
SCBR[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
–	BITS[2:0]			–	–	NCPHA	CPOL
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **CPOL : Clock Polarity**

0: The inactive state value of SPCK is logic level zero.

1 : The inactive state value of SPCK is logic level one.

CPOL is used to determine the inactive state value of the serial clock (SPCK). CPOL is used with NCPHA to produce a desired clock/data relationship between master and the slave devices.

- **NCPHA : Clock Phase**

0: Data is changed on the leading edge of SPCK and captured on the following edge of SPCK.

1: Data is captured on the leading edge of SPCK and changed on the following edge of SPCK.

NCPHA determines which edge of SPCK causes data to change and which edge causes data to be captured.

NCPHA is used with CPOL to produce a desired clock/data relationship between master and slave devices.

- **BITS[2:0] : Bits Per Transfer**

Table 21-4. SPI Bits Per Transfer

BITS[2:0]	Bits Per Transfer
001	1
010	2
011	3
100	4
101	5
110	6
111	7
000	8

- **SCBR[7:0] : Serial Clock Baud Rate** (These bits are only used in master mode.)

The SPI interface uses a modulus counter to derive SPCK baud rate from the SPI Master Clock, selected between PCLK and PCLK/32. Baud rate is selected by writing a value from 2 to 255 into SCBR[7:0]. The following equation determines the SPCK baud rate :

$$\text{SPCK_Baud_Rate} = \text{SPI_Master_Clock_frequency} / (2 \times \text{SCBR}[7:0])$$

NOTE: Giving SCBR[7:0] a value of zero or one disables the baud rate generator. SPCK is disabled and assumes its inactive state value. No serial transfers occur. At reset, baud rate is disabled.

Table 21-5. Baudrate

SPI_master_clock	Baudrate	SCBR[7:0]	Error
4 MHz	125K	0x10	0%
	250K	0x08	0%
	500K	0x04	0%
	1M	0x02	0%
20 MHz	125K	0x50	0%
	250K	0x28	0%
	500K	0x14	0%
	1 MHz	0x0A	0%
	4 MHz	0x03	16.6%
40 MHz	125K	0xA0	0%
	250K	0x50	0%
	500K	0x28	0%
	1 MHz	0x14	0%
	4 MHz	0x05	0%

- **DLYBS[1:0] : Delay Before SPCK** (These bits are only used in master mode.)
This field defines the length of delay from NSS valid to the first SPCK transition.

Table 21-6. Delay Before SPCK

DLYBS[1:0]	Delay (SPCK Periods)
00	0.5
01	1
10	2
11	4

- **DLYBCT[1:0] : Delay between Consecutive Transfers** (These bits are only used in master mode.)
This field defines the delay between 2 consecutive transfers with the same peripheral without removing the slave select.

Table 21-7. Delay between Consecutive Transfers

DLYBCT[1:0]	Delay (SPCK Periods)
00	0.5
01	1
10	2
11	4

22

SIMPLE TIMER (ST)

1. OVERVIEW

1.1 FUNCTIONAL DESCRIPTION

The simple timer channel provides basic functions for timing calculation including 2 cascaded divider and a 16-bit counter.

Asynchronous ST is clocked on the low frequency clock. (ST0)

Synchronous ST is clocked on the core clock. (ST1)

The pre-scalar defines the clock frequency of the down counter of each channel.

The 16-bit counter starts down-counting when a value different of zero is loaded. An interrupt is generated when the counter reaches 0x0000.

This module includes :

- 1 asynchronous ST, with 2 channels
- 1 synchronous ST, with 2 channels

2. FUNCTIONAL OPERATION

2.1 GENERAL DESCRIPTION

2.1.1 Functional Description

When a value is loaded in the LOAD[15:0] bits of the ST_CT register and the channel is started by CHENx of ST_CR, the counter starts down-counting at channel clock frequency until the counter reaches zero. The delay between the load and the interrupt is : Counter value x Clock period.

The counter clock frequency is given by the relations :

Divider_clock = module clock / (2 x SYSCAL[10:0] + 1)

Counter_clock = Divider clock / $2^{\text{PRESCAL}[3:0]}$, where :

- module clock is the clock which supplies the ST; module clock is PCLK if the ST is synchronous, LFCLK if the ST is asynchronous
- SYSCAL[10:0] is the first prescaler, written in the ST_PR register
- PRESCAL[3:0] is the second prescaler, written in the ST_PR register

The current counter value can be read in the corresponding ST_CCV register.

When the counter reaches 0x0, the bit CHEND is set to 1 in the ST_SR register and generates an interrupt if the corresponding bit is enabled in ST_IMR register. This bit is cleared in writing the corresponding bit in ST_CSR register. The maximum timer error is one counter_clock period.

The simple timer integrates also an automatic reload function if the AUTOREL bit is set in the ST_PR register. When the counter reaches 0x0, it is automatically reloaded with the value written in LOAD[15:0] field of the ST_CT.

NOTE

In the case of asynchronous ST – meaning when the ST is supplied by LFCLK – any write in the ST_CR and ST_CSR registers is not traduced instantaneously. To verify the real ST state, the user must read the ST_SR register.

It is not possible to change the counter register(ST_CT) contents when the Simple Timer is enabled.

If a channel is disabled before it reaches the end of the down-counting, the current counter value(ST_CCV) is held. If no write has occurred in the counter register(ST_CT) before it is re-enabled, the down-counting restarts from value held in the current counter value register.

2.1.2 Block Diagram

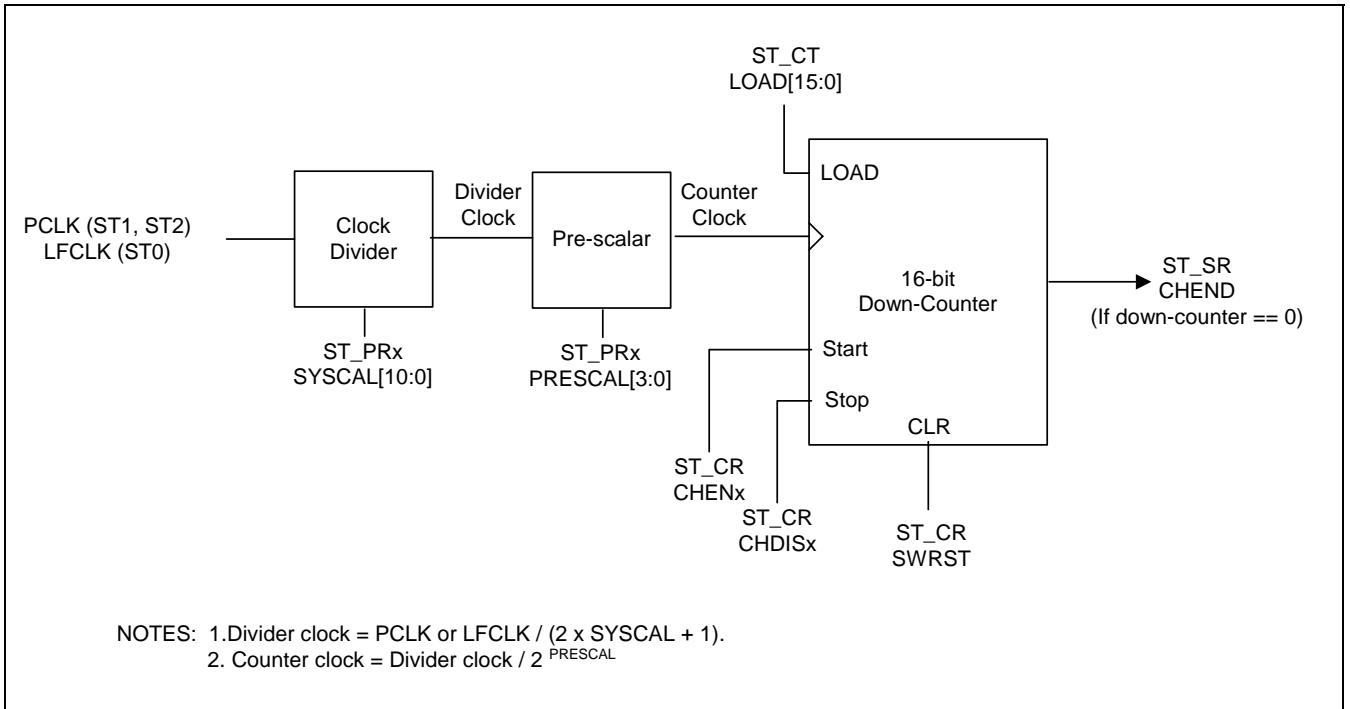


Figure 22-1. Simple Timer Block Diagram

2.1.3 Interrupt Description (Refer ST_IMR in the P19-11)

There are 3 types of interrupt, for the simple timer :

- A CHDIS interrupt which indicates that the channel has been disabled.
- A CHLD interrupt which indicates that the channel has started to down count and loaded the data.
- A CHEND interrupt which indicates that the channel has reached the end of the down-counting. This interrupt occurs after the after the down counter reaches the value 0x0000.
 - 2 PCLK cycles in case of a synchronous simple timer
 - 2 PCLK cycles in case of a asynchronous simple timer in low power mode
 - 3 PCLK cycles in case of a asynchronous simple timer in normal mode

2.1.4 Power Management

The Simple Timer is provided with a power management block allowing optimization of power consumption. When the timer is stopped, the clock is immediately stopped. When the clock is re-enabled, the timer resumes action where it left off.

2.2 EXAMPLE

Example of use of the Simple Timer :

We want to generate a tick (usually used in RTOS) of 10 ms with auto-reload and an interrupt.

The ST is supplied by LFCLK = 1MHz.

Configuration :

- Enable the clock on ST peripheral by writing bit ST in ST_ECR.
- Do a software reset of the ST peripheral to be in a known state by writing bit SWRST in ST_CR and wait about four LFCLK period for the circuitry to be stabilized.
- By setting SYSCAL=49 and PRESCAL=0 in the ST_PR register, the counter clock frequency is 10000Hz, meaning 0.1ms period. With LOAD=100 in the ST_CT register, the 10ms tick delay is assured. The user must verify that the counter load is completed by checking the CHLD bit in ST_SR register(or by using the corresponding interrupt). To activate the auto-reload functionality, the AUTOREL bit must be set in the ST_PR register.
- Interrupt configuration : By writing the CHEND bit in ST_IER register, an interrupt will occurs each time the counter reaches 0x0. GIC must be configured. When the interrupt occurs, the CHEND bit must be cleared by writing the corresponding bit in the ST_CSR register.
- Writing the CHEN bit in the ST_CR register will start the ST. User can verify that CHENSx bit in ST_SR is set.

Interruption Handling :

- IRQ Entry and call C function.
- Read ST_SR and verify the source of the interrupt.
- Clear the corresponding interrupt at peripheral level by writing in the ST_CSR.
- Interrupt treatment.
- IRQ Exit.

3. REGISTERS DESCRIPTION

Base Addresses – ST0 (asynchronous): 0xFFE20000
 ST1 (synchronous) : 0xFFE24000

Table 22-1. Simple Timer Special Function Registers

Offset Address	Name	Description	R/W	Reset State
0x000 – 0x04C	–	Reserved	–	–
0x050	ST_ECR	Enable clock register	W	–
0x054	ST_DCR	Disable clock register	W	–
0x058	ST_PMSR	Power management status register	R	0XXXXXXXX0
0x05C	–	Reserved	–	–
0x060	ST_CR	Control register	W	–
0x064	–	Reserved	–	–
0x068	–	Reserved	–	–
0x06C	ST_CSR	Clear status register	W	–
0x070	ST_SR	Status register	R	–
0x074	ST_IER	Interrupt enable register	W	–
0x078	ST_IDR	Interrupt disable register	W	–
0x07C	ST_IMR	Interrupt mask register	R	0x00000000
0x080	ST_PR0	Channel 0 pre-scalar	R/W	0x00000000
0x084	ST_CT0	Channel 0 counter	R/W	0x00000000
0x088	ST_PR1	Channel 1 pre-scalar	R/W	0x00000000
0x08C	ST_CT1	Channel 1 counter	R/W	0x00000000
0x090 – 0x1FC	–	Reserved	–	–
0x200	ST_CCV0	Current counter value	R	0x00000000
0x204	ST_CCV1	Current counter value	R	0x00000000

ST Enable Clock Register

ST_ECR (0x050)

Access: Write only

31	30	29	28	27	26	25	24
DBGEN	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	ST	–
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **ST : Simple Timer clock enable**

0: No effect

1: Enable Timer clock

- **DBGEN : Debug mode enable**

0: No effect

1: Enable debug mode

ST Disable Clock Register

ST_DCR (0x054)

Access: Write only

31	30	29	28	27	26	25	24
DBGEN	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	ST	–
W	W	W	W	W	W	W	W

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **ST : Simple Timer clock disable**

0: No effect

1: Disable Simple Timer clock

- **DBGEN : Debug mode disable**

0: No effect

1: Disable debug mode

ST Power Management Status Register **ST_PMSR (0x058)** **Access: Read only**

31	30	29	28	27	26	25	24
DBGEN	–	IPIDCODE[25:20]					
R-0	R-0	R-U	R-U	R-U	R-U	R-U	R-U
23	22	21	20	19	18	17	16
IPIDCODE[19:12]							
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U
15	14	13	12	11	10	9	8
IPIDCODE[11:4]							
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U
7	6	5	4	3	2	1	0
IPIDCODE[3:0]				–	–	ST	–
R-U	R-U	R-U	R-U	R-0	R-0	R-0	R-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **ST : Simple Timer clock status**

0: Simple Timer clock disabled.

1: Simple Timer clock enabled.

- **IPIDCODE[25:0]**

This field contains the version number of the module, coded on 26 bits.

- **DBGEN : Debug mode**

0: DBGACK has no influence on the Simple Timer.

1: DBGACK has influence on the Simple Timer.

NOTE: The ST_PMSR register is not reset by software reset.

ST Control Register **ST_CR (0x060)** **Access: Write only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	CHDIS1	CHDIS0
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	CHEN1	CHEN0
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	SWRST
W	W	W	W	W	W	W	W

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **SWRST : ST software reset**

0: No effect.

1: Reset the Simple Timer.

A software reset of the ST is performed. It resets all the registers.

22-1. IMPORTANT NOTICE

For the ST0(Asynchronous ST), user must ensure that $LFCLK < PCLK/8$ before writing 0x1 in the ST_CR register. Otherwise, the module is stuck in reset.

- **CHENx : Simple Timer channel enable (Start signal)**

0: No effect.

1: Enables the Channel x.

- **CHDISx : Simple Timer channel disable (Stop signal)**

0: No effect.

1: Disables the Channel x.

In case both CHENx and CHDISx are equal to one when the control register is written, the channel will be disabled.

22-2. IMPORTANT NOTICE

After software reset, software must ensure that no channel is disabled before being enabled, otherwise, the corresponding channel is frozen. It is forbidden to disable a channel after software reset. After software reset, once a channel has been enabled, it can be disabled and re-enabled as many times as needed.

ST Clear Status Register

ST_CSR (0x06C)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	CHLD1	CHLD0
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	CHDIS1	CHDIS0
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	CHEND1	CHEND0
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **CHENDx : Clear channel end interrupt**

0: No effect.

1: Clear CHENDx interrupt.

- **CHDISx : Clear channel disable interrupt**

0: No effect.

1: Clear CHDISx interrupt.

- **CHLDx : Clear channel load interrupt**

0: No effect.

1: Clear CHLDx interrupt.

ST Status Register

ST_SR (0x070)

Access: Read only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	CHLD1	CHLD0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	CHDIS1	CHDIS0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	CHENS1	CHENS0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	CHEND1	CHEND0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **CHENDx : Channel end status**

0: No end of counting on Channel x.

1: End of counting on Channel x.

22-3. IMPORTANT NOTICE

When CHENDx occurs and auto-reload mode is not set, before setting CHENx bit of ST_CR, to restart the timer, software must configured register ST_CTx, otherwise channel x does not restart.

- **CHENSx : Channel enable status**

0: Channel x is disabled.

1: Channel x is enabled.

- **CHDISx : Channel disable status**

0: No effect.

1: Divider has been reset.

CHDISx indicates that the divider module has been reset (a delay due to asynchronous clock is introduced).

- **CHLDx : Channel load status**

0: No effect.

1: Down-counter is loaded.

CHLDx indicates also that the counter (divider) has been enabled (a delay due to asynchronous clock is introduced).

ST Interrupt Enable Register

ST_IER (0x074)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	CHLD1	CHLD0
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	CHDIS1	CHDIS0
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	CHEND1	CHEND0
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **CHENDx : Channel end interrupt enable**

0: No effect.

1: Enable the CHENDx interrupt

- **CHDISx : Channel disable interrupt enable**

0: No effect.

1: Enable the CHDISx interrupt

- **CHLDx : Channel load interrupt enable**

0: No effect.

1: Enable the CHLDx interrupt

ST Interrupt Disable Register

ST_IDR (0x078)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	CHLD1	CHLD0
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	CHDIS1	CHDIS0
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	CHEND1	CHEND0
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **CHENDx : Channel end interrupt disable**

0: No effect.

1: Disable the CHENDx interrupt

- **CHDISx : Channel disable interrupt disable**

0: No effect.

1: Disable the CHDISx interrupt

- **CHLDx : Channel load interrupt disable**

0: No effect.

1: Disable the CHLDx interrupt

ST Interrupt Mask Register

ST_IMR (0x07C)

Access: Read only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	CHLD1	CHLD0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	CHDIS1	CHDIS0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	CHEND1	CHEND0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **CHENDx : Channel end interrupt mask**

0: Channel x end interrupt is disabled.

1: Channel x end interrupt is enabled.

- **CHDISx : Channel disable interrupt mask**

0: Channel x disable interrupt is disabled.

1: Channel x disable interrupt is enabled.

- **CHLDx : Channel load interrupt mask**

0: Channel x load interrupt is disabled.

1: Channel x load interrupt is enabled.

ST Channel 0 Prescaler Register

ST_PR0 (0x080)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	SYSCAL[10:8]		
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
SYSCAL[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
–	–	–	AUTOREL	PRESCAL[3:0]			
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **PRESCAL[3:0] : Channel 0 prescaler**

Counter clock = divider clock / 2^{PRESCAL} .

- **AUTOREL : Auto reload**

0: The counter is not automatically reloaded with the COUNTER[15:0] value when it reaches 0x0000.

1: The counter is automatically reloaded with the COUNTER[15:0] value when it reaches 0x0000.

- **SYSCAL[10:0] : Module clock prescaler value**

This prescaler is used to divide the module clock.

Divider clock = PCLK or LFCLK / (2xSYSCAL + 1).

if SYSCAL = 0, divider clock = PCLK or LFCLK

22-4. IMPORTANT NOTICE

It is forbidden to write this register unless one of the following conditions is met :

- After hardware or software reset and before channel 0 enable
- After channel 0 disable as soon as the following condition arises : (CHENS0==0) and (CHDIS0==1)

ST Channel 0 Counter Register **ST_CT0 (0x084)** **Access: Read/Write**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
LOAD[15:8]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
LOAD[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **LOAD[15:0] : Channel 0 preload value**

Gives the consign which is asked to the timer. In other word from which counter value the Simple Timer has to decrement.

22-5. IMPORTANT NOTICE

1. In auto-reload mode, LOAD[15:0] must be higher or equal than 0x2.
2. When auto-reload mode is not set, software must configure LOAD[15:0] before setting the CHEN0 bit in register ST_CR, in order to restart the timer.
3. It is forbidden to write this register unless one of the following conditions is met :
 - After hardware or software reset and before channel 0 enable
 - After channel 0 disable as soon as the following condition arises : (CHENS0==0) and CHDIS0==1)

ST Channel 1 Prescaler Register

ST_PR1 (0x080)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	SYSCAL[10:8]		
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
SYSCAL[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
–	–	–	AUTOREL	ST_PRESCAL[3:0]			
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- PRESCAL[3:0] : Channel 1 prescaler**

Counter clock = divider clock / 2^{PRESCAL} .

- AUTOREL : Auto reload**

0: The counter is not automatically reloaded with the COUNTER[15:0] value when it reaches 0x0000.

1: The counter is automatically reloaded with the COUNTER[15:0] value when it reaches 0x0000.

- SYSCAL[10:0] : Module clock prescaler value**

This prescaler is used to divide the module clock.

Divider clock = PCLK or LFCLK / (2xSYSCAL + 1).

if SYSCAL = 0, divider clock = PCLK or LFCLK

22-6. IMPORTANT NOTICE

It is forbidden to write this register unless one of the following conditions is met :

- After hardware or software reset and before channel 0 enable
- After channel 0 disable as soon as the following condition arises : (CHENS0==0) and (CHDIS0==1)

ST Channel 1 Counter Register

ST_CT1 (0x084)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
LOAD[15:8]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
LOAD[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- LOAD[15:0] : Channel 1 preload value**

Gives the consign which is asked to the timer. In other word from which counter value the Simple Timer has to decrement.

22-7. IMPORTANT NOTICE

- In auto-reload mode, LOAD[15:0] must be higher or equal than 0x2.
- When auto-reload mode is not set, software must configure LOAD[15:0] before setting the CHEN0 bit in register ST_CR, in order to restart the timer.
- It is forbidden to write this register unless one of the following conditions is met :
 - After hardware or software reset and before channel 0 enable
 - After channel 0 disable as soon as the following condition arises: (CHENS0==0) and (CHDIS0==1)

ST Current Counter 0 Value Register

ST_CC0 (0x200)

Access: Read only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
COUNT[15:8]							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
COUNT[7:0]							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- COUNT[15:0] : Current Counter Value Register**

This register gives the current value of the down counter.

For asynchronous ST only, to be sure the counter value read is good, the counter must be read identical two consecutive time. This is due to the asynchronism between PCLK and LFCLK.

ST Current Counter 1 Value Register **ST_CCV1 (0x204)** **Access: Read only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
COUNT[15:8]							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
COUNT[7:0]							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **COUNT[15:0] : Current Counter Value Register**

This register gives the current value of the down counter.

For asynchronous ST only, to be sure the counter value read is good, the counter must be read identical two consecutive time. This is due to the asynchronism between PCLK and LFCLK.

23

SPECIAL FUNCTION MODULE (SFM)

1. OVERVIEW

1.1 FUNCTIONAL DESCRIPTION

The SFM provides registers which implement the following special functions :

- Chip identification
- Architecture information
- Memory information

2. REGISTERS DESCRIPTION

Base Address – 0xFFFE4000

Table 23-1. SFM Special Function Registers

Offset Address	Name	Description	R/W	Reset State
0x000	SFM_CIDR	SFM chip ID register	R	0x01047271
0x004	SFM_ARCR	SFM architecture register	R	0x0003AA00
0x008	SFM_MSR	SFM memory size register	R	0x000F033F

SFM Chip Identifier Register

SFM_CIDR (0x000)

Access: Read only

31	30	29	28	27	26	25	24
VER[3:0]				PN[15:12]			
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-1
23	22	21	20	19	18	17	16
PN[11:4]							
R-0	R-0	R-0	R-0	R-0	R-1	R-0	R-0
15	14	13	12	11	10	9	8
PN[3:0]				MC[10:7]			
R-0	R-1	R-1	R-1	R-0	R-0	R-1	R-0
7	6	5	4	3	2	1	0
MC[6:0]							-
R-0	R-1	R-1	R-1	R-0	R-0	R-0	R-1

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **MC[10:0] : Manufacturer Code**

JTAG reference

- **PN[15:0] : Part Number**

This is the Part number.

⇒ The upper 4 bits define the type of chip as shown in the table hereunder :

Table 23-2. Type of Chip

PN[15:12]	CODE	Comment
0000	MET	Metering, Smart Access
0001	AUT	Automotive
0010	IND	Industry
0011	TXT	Teletext
0100	MUL	Multimedia
Others	XXX	Reserved

⇒ The 12 lower bits define the chip number coded in DCB.

- **VER[3:0] : Version**

Chip version, beginning at 0.



SFM Architecture Register **SFM_ARCR (0x004)** **Access: Read only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	IRAME	BOOT
R-0	R-0	R-0	R-0	R-0	R-0	R-1	R-1
15	14	13	12	11	10	9	8
NVDE	NVDT[2:0]			NVPE	NVPT[2:0]		
R-1	R-0	R-1	R-0	R-1	R-0	R-1	R-0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	ARC[1:0]	
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- ARC[1:0] : Architecture**

Chip architecture

Table 23-3. Chip Architecture

ARC[1:0]	Architecture
00	ARM7TDMI
01	ARM926EJS
10	ARM946ES
11	Reserved

- **NVPT[2:0] : Non Volatile Program Memory Type**

Type of program non volatile memory

Table 23-4. Memory type

NVPT[2:0]	Architecture
000	ROM
001	EEPROM
010	FLASH
011	FRAM
Others	Reserved

- **NVPE : Non volatile Program Memory Exists**

0: No non volatile program memory on the chip

1: Non volatile program memory available on the chip

- **NVDT[2:0] : Non Volatile Data Memory Type**

Type of data non volatile memory.

Table 23-5. Memory type

NVPT[2:0]	Architecture
000	ROM
001	EEPROM
010	FLASH
011	FRAM
Others	Reserved

- **NVDE : Non Volatile Data Memory Exists**

0: No non volatile data memory on the chip

1: Non volatile data memory available on the chip

- **BOOT :**

0: Boot on external memory

1: Boot on internal non volatile memory

- **IRAME : Internal RAM exists**

0: No internal RAM

1: Internal RAM exists

SFM Memory Size Register **SFM_MSR (0x008)** **Access: Read only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
IRAMS[7:0]							
R-0	R-0	R-0	R-0	R-1	R-1	R-1	R-1
15	14	13	12	11	10	9	8
NVDMS[7:0]							
R-0	R-0	R-0	R-0	R-0	R-0	R-1	R-1
7	6	5	4	3	2	1	0
NVPMS[7:0]							
R-0	R-0	R-1	R-1	R-1	R-1	R-1	R-1

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **NVPMS[7:0] : Non Volatile Program Memory Size**

The non volatile program memory size = (NVPMS[7:0]+1) × 8 kbytes.

- **NVDMS[7:0] : Non Volatile Data Memory Size**

The non volatile data memory size = (NVDMS[7:0]+1) × 8 kbytes.

- **IRAMS[7:0] : Internal RAM Size**

The internal RAM size = (IRAMS[7:0]+1) × 1 kbytes.

24 STAMP TIMER (STT)

1. OVERVIEW

1.1 FUNCTIONAL DESCRIPTION

The Stamp Timer provides a counter and an alarm function. It is also used to stamp the CAN messages.

1.2 BLOCK DIAGRAM

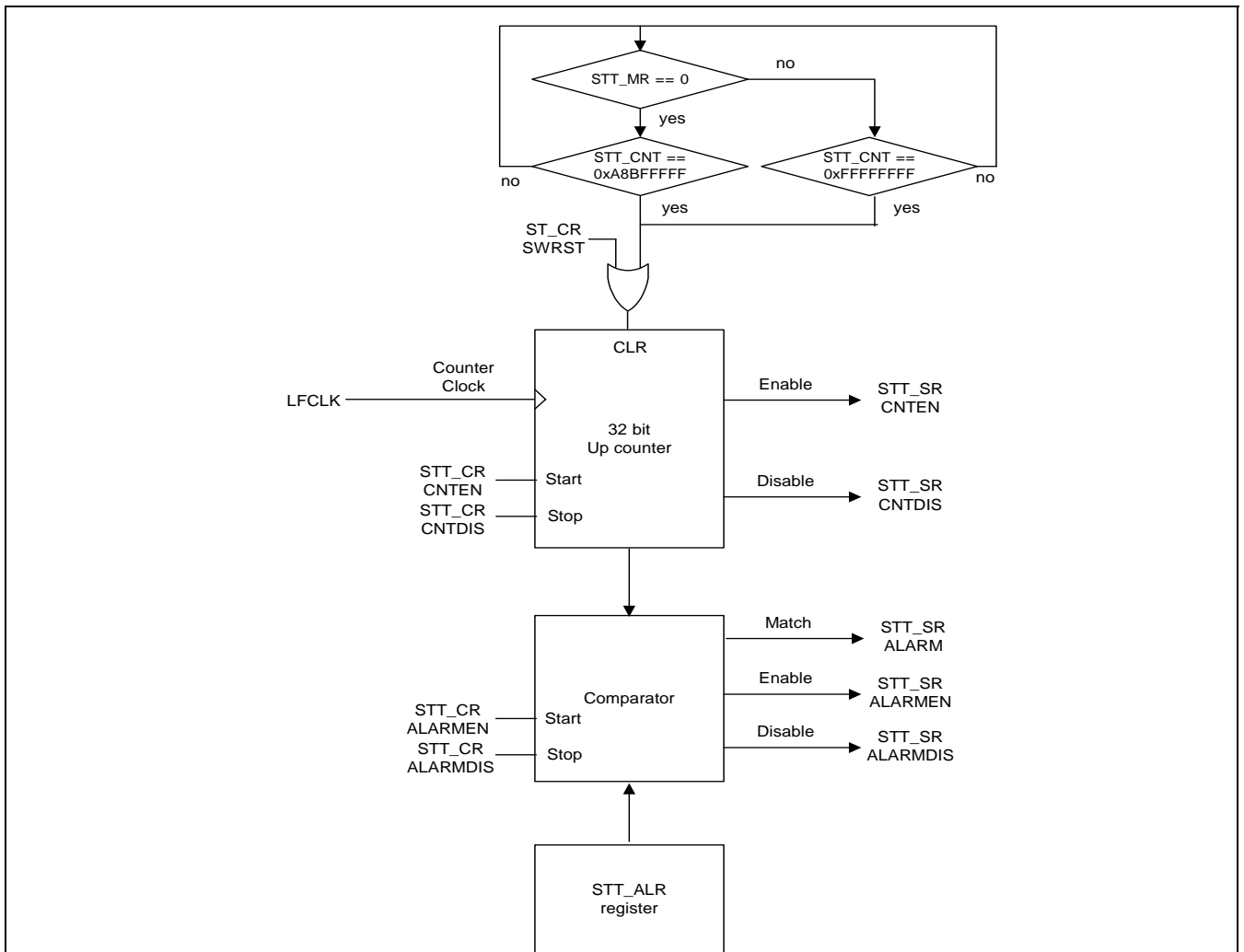


Figure 24-1. Simple Timer Block Diagram

2. FUNCTIONAL OPERATION

2.1 TIMER FUNCTIONALITY

2.1.1 Counter Register

The counter is a 32-bit counter that indicates the number of low frequency clock (LFCLK) pulses elapsed since the last time it was reset to zero.

The counter is incremented every period of the low frequency clock.

The counter is reset to 0x00000000 when the counter reaches 0xA8C00000 or 0xFFFFFFFF (it is configurable on the Mode Register).

A write access can only be performed when the counter is disabled because of an asynchronous interface (see "Asynchronous interface" below).

2.1.2 Alarm Register

The alarm register has the same resolution as the counter.

An interrupt is generated at the end of the period at which the value in the counter register equals the value in the alarm register.

A write access can only be performed when the alarm counter is disabled because of an asynchronous interface. An invalid data (i.e. value greater or equal to 0xA8C00000) will not be written into the alarm register if CNTRST = 0.

2.1.3 Asynchronous Interface

As the counter is an asynchronous counter (can used the LFCLK clock), some precautions must be taken with it.

When enabling or disabling alarm or counter, software must wait for enabled or disabled interrupt to be sure that the alarm or the counter is really enabled or disabled.

2.1.4 CAN Time Stamp

The 32 bits register forming the counter is provided to the CAN module. After each transmission or reception of a CAN frame, the value of the current counter will be automatically written in the corresponding CAN channel CAN_STPx register.

2.2 PROGRAMMING EXAMPLES

Example of use of the timer :

Use of the timer to generate an interrupt after 1 second according that the low frequency clock is equal to 1MHz.
(1 second = $0xF4240 / 1\text{MHz}$)

Configuration :

- Do a software reset of the timer to be in a known state by writing bit SWRST in STT_CR and wait about four LFCLK period for the circuitry to be stabilized
- Configuration of STT_ALARM : When the counter will be equal to this value, an interrupt can be generated. For 1 second at 1MHz, Alarm value should be $0xF423F$.
- Configuration of STT_CNT : Starting value from which counter will start to increase. Should be left to 0.
- Configuration of STT_IER : bit ALARM enables interrupt at the peripheral level when counter is equal to the programmed value in STT_ALARM. Other interrupts can be activated to indicate when the counter or alarm functionality are really enabled or disabled as stamp timer is clocked on the LFCLK. GIC has to be configured.
- Configuration of STT_CR : Starts the counter and enables the alarm (bit ALARMEN and CNTEN). Counter will start to increment when bit CNTEN will be set in STT_SR, same for alarm functionality bit ALARMEN, an interrupt can be programmed with those this events.

Interruption Handling :

- IRQ Entry and call C function.
- Read STT_SR and verify the source of the interrupt.
- Clear the corresponding interrupt at peripheral level by writing in the STT_CSR.
- Interrupt treatment, counter will restart automatically counting from 0 when it reaches $0xF423F$ (STT_MR). If users want to set a lower value in STT_ALARM, and if they want to restart counter, they must disable counter and set it to 0 and enable it again.
- IRQ Exit.

24-1. IMPORTANT NOTICE

If internal oscillator is used for clock source of STT, STT cannot be used for application which needs accurate timing measurement because internal oscillator(1MHz) has not good accuracy.

3. REGISTERS DESCRIPTION

Base Address – 0xFFE30000

Table 24-1. Stamp Timer Special Function Registers

Offset Address	Name	Description	R/W	Reset Status
0x000 – 0x05C	–	Reserved	–	–
0x050	STT_ECR	Enable clock register	W	–
0x054	STT_DCR	Disable clock register	W	–
0x058	STT_PMSR	Power management status register	R	0XXXXXXXX0
0x05C	–	Reserved	–	–
0x060	STT_CR	Control register	W	–
0x064	STT_MR	Mode register	R/W	0XXXXXXXX0
0x068	–	Reserved	–	–
0x06C	STT_CSR	Clear status register	W	–
0x070	STT_SR	Status register	R	0x00000000
0x074	STT_IER	Interrupt enable register	W	–
0x078	STT_IDR	Interrupt disable register	W	–
0x07C	STT_IMR	Interrupt mask register	R	0x00000000
0x080	STT_CNT	Counter register	R/W	0x00000000
0x084	STT_ALR	Alarm register	R/W	0x00000000

STT Enable Clock Register

STT_ECR (0x050)

Access: Write only

31	30	29	28	27	26	25	24
DBGEN	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **DBGEN : Debug mode enable**

0: No effect

1: Enable debug mode

STT Disable Clock Register

STT_DCR (0x054)

Access: Write only

31	30	29	28	27	26	25	24
DBGEN	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **DBGEN : Debug mode disable**

0: No effect

1: Disable debug mode

STT Power Management Status Register

STT_PMSR (0x058)

Access: Read only

31	30	29	28	27	26	25	24
DBGEN	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **DBGEN : Debug mode**

0: STT is not halted during ARM core debug mode

1: STT is halted during ARM core debug mode

STT Control Register **STT_CR (0x060)** **Access: Write only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	ALARMDIS	ALARMEN	CNTDIS	CNTEN	SWRST
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **SWRST : STT software reset**

- 0: No effect.
1: Resets the STT.

24-2. IMPORTANT NOTICE

User must ensure that $LFCLK < PCLK/8$ before writing 0x1 in the STT_CR register. Otherwise, the module is stuck in reset.

A software triggered hardware reset of the STT is performed. It reset all the registers. The software must wait 4 LFCLK to set up properly before using other registers.

- **CNTEN : STT counter enable**

- 0: No effect.
1: Enables the STT seconds counter.

- **CNTDIS : STT counter disable**

- 0: No effect.
1: Disables the STT seconds counter.

In case both CNTEN and CNTDIS are equal to one when the control register is written, the STT counter will be disabled.

- **ALARMEN : STT alarm enable**

- 0: No effect.
1: Enables the STT alarm.

- **ALARMDIS : STT alarm disable**

- 0: No effect.
1: Disables the STT alarm.

In case both ALARMEN and ALARMDIS are equal to one when the control register is written the STT alarm will be disabled.

STT Mode Register

STT_MR (0x064)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	CNTRST
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **CNTRST : Counter Reset**

0: The counter is reset to 0x00000000 at the end of the period when it reaches 0xA8BFFFFFF.

1: The counter is reset to 0x00000000 at the end of the period when it reaches 0xFFFFFFFF.

24-3. IMPORTANT NOTICE

CNTRST bit can only be written when CNTRST=0.

STT Clear Status Register

STT_CSR (0x06C)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	ALARMDIS	ALARMEN	CNTDIS	CNTEN	ALARM
W	W	W	W	W	W	W	W

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **ALARM : Clear alarm interrupt**

0: No effect.

1: Clears ALARM interrupt.

- **CNTEN : Clear counter enabled**

0: No effect.

1: Clears the counter enabled interrupt.

- **CNTDIS : Clear counter disabled**

0: No effect.

1: Clears the counter disabled interrupt.

- **ALARMEN : Clear alarm enabled**

0: No effect.

1: Clears the alarm enabled interrupt.

- **ALARMDIS : Clear alarm disabled**

0: No effect.

1: Clears the alarm disabled interrupt.

STT Status Register

STT_SR (0x06C)

Access: Read only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	ALARMENS	CNTENS
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
–	–	WSEC	ALARMDIS	ALARMEN	CNTDIS	CNTEN	ALARM
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **ALARM : Alarm interrupt**

0: No alarm occurred.

1: An alarm occurred since last clear of the status register.

- **CNTEN : counter enabled interrupt**

0: No counter enabled interrupt.

1: A counter enabled interrupt occurred since last clear of the status register.

- **CNTDIS : counter disabled interrupt**

0: No counter disabled interrupt.

1: A counter disabled interrupt occurred since last clear of the status register.

- **ALARMEN : Alarm enabled interrupt**

0: No alarm enabled interrupt.

1: An alarm enabled interrupt occurred since last clear of the status register.

- **ALARMDIS : Alarm disabled interrupt**

0: No alarm disabled interrupt.

1: An alarm disabled interrupt occurred since last clear of the status register.

- **WSEC : Write Counter**

0: No effect.

1: A write is occurring on the second counter register.

- **CNTENS : Counter enable status**

0 : Counter is disabled.

1: Counter is enabled.

- **ALARMENS : Alarm enable status**

0: Alarm is disabled.

1: Alarm is enabled.

Precaution:

1. When an interrupt occurs, be sure to wait at least a low frequency clock (LFCLK) period before reading the STT_SR register (this is due to the asynchronous interface).
2. When a write access is performed in STT_CR(or STT_CSR) register in order to set (or clear) a bit in the STT_SR register, software must wait a minimum of a low frequency clock (LFCLK) period before reading the STT_SR (this is due to the asynchronous interface).

STT Interrupt Enable Register

STT_IER (0x074)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	ALARMDIS	ALARMEN	CNTDIS	CNTEN	ALARM
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **ALARM : Alarm interrupt enable**

0: No effect

1: Enable the Alarm interrupt

- **CNTEN : Counter interrupt enable**

0: No effect

1: Enable the CNTEN interrupt

- **CNTDIS : Counter interrupt enable**

0: No effect

1: Enable the CNTDIS interrupt

- **ALARMEN : Alarm interrupt enable**

0: No effect

1: Enable the ALARMEN interrupt

- **ALARMDIS : Alarm interrupt enable**

0: No effect

1: Enable the ALARMDIS interrupt

STT Interrupt Disable Register

STT_IDR (0x078)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	ALARMDIS	ALARMEN	CNTDIS	CNTEN	ALARM
W	W	W	W	W	W	W	W

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **ALARM : Alarm interrupt disable**

0: No effect

1: Disable the Alarm interrupt

- **CNTEN : Counter interrupt disable**

0: No effect

1: Disable the CNTEN interrupt

- **CNTDIS : Counter interrupt disable**

0: No effect

1: Disable the CNTDIS interrupt

- **ALARMEN : Alarm interrupt disable**

0: No effect

1: Disable the ALARMEN interrupt

- **ALARMDIS : Alarm interrupt disable**

0: No effect

1: Disable the ALARMDIS interrupt

STT Interrupt Mask Register

STT_IMR (0x07C)

Access: Read only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
–	–	–	ALARMDIS	ALARMEN	CNTDIS	CNTEN	ALARM
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **ALARM : Alarm interrupt mask**

0: ALARM interrupt is disabled.

1: ALARM interrupt is enabled.

- **CNTEN : Counter enabled interrupt mask**

0: CNTEN interrupt is disabled.

1: CNTEN interrupt is enabled.

- **CNTDIS : Counter disabled interrupt mask**

0: CNTDIS interrupt is disabled.

1: CNTDIS interrupt is enabled.

- **ALARMEN : Alarm enabled interrupt mask**

0: ALARMEN interrupt is disabled.

1: ALARMEN interrupt is enabled.

- **ALARMDIS : Alarm disabled interrupt mask**

0: ALARMDIS interrupt is disabled.

1: ALARMDIS interrupt is enabled.

STT Seconds Register		STT_CNT (0x080)				Access: Read /Write	
31	30	29	28	27	26	25	24
COUNT[31:24]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
COUNT[23:16]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
COUNT[15:8]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
COUNT[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **COUNT[31:0] : Counter registers**

Number of LFCLK clock cycles periods elapsed since last reset to zero.

24-4. IMPORTANT NOTICE

This register can only be written when CNTENS = 0.

An invalid data (i.e. value greater or equal to 0xA8C00000) will not be written into the counter register if CNTRST=0.

STT Alarm Register

STT_ALR (0x084)

Access: Read/Write

31	30	29	28	27	26	25	24
ALARMREG[31:24]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
ALARMREG[23:16]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
ALARMREG[15:8]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
ALARMREG[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **ALARMREG[31:0] : Alarm register**

An interrupt can be generated when the counter register reaches this value.

24-5. IMPORTANT NOTICE

This register can only be written when ALARMENS = 0.

An invalid data (i.e. value greater or equal to 0xA8C00000) will not be written into the alarm register if CNTRST=0.

25

STEPPER MOTOR CONTROLLER (SMC)

1. OVERVIEW

1.1 FUNCTIONAL DESCRIPTION

The stepper motor controller is a module capable of generating signals used to spin a stepper motor.

Two functional mode are supported :

- PWM mode(2 channels) : the stepper motor controller generates periodic signals. The duty cycle and the frequency are configurable
- Stepper motor controller : the stepper motor controller is able to produce full stop, half step and micro-step switching waveforms. It permits unipolar and bipolar stepper motor drive.

The main features are :

- The motor rotation speed is configurable : a clock divider allows to fine tune the frequency of the generated waveform,
- 'One phase on' and 'two phase on' full stepping driving methods are supported.
- Half stepping driving method supported.
- 'Cosinus/sinus' and 'high torque' microstepping driving methods are supported. The number of microsteps by full step is configurable.
- Two running modes are offered : continuous mode and target mode. In continuous mode, the waveform is permanently repeated until a stop command. In target mode, the waveform is produced until the requested position is reached. Then, once the number of step is reached, the motor locks in its targeted position.
- Friendly user's interface in target mode : the motor is controlled with a single parameter : the position to reach.
- Readable motor position : the current motor position is updated permanently and is readable on the fly.
- PWM mode is offered. In this mode, two outputs can generate periodic and configurable signals.

1.2 BLOCK DIAGRAM

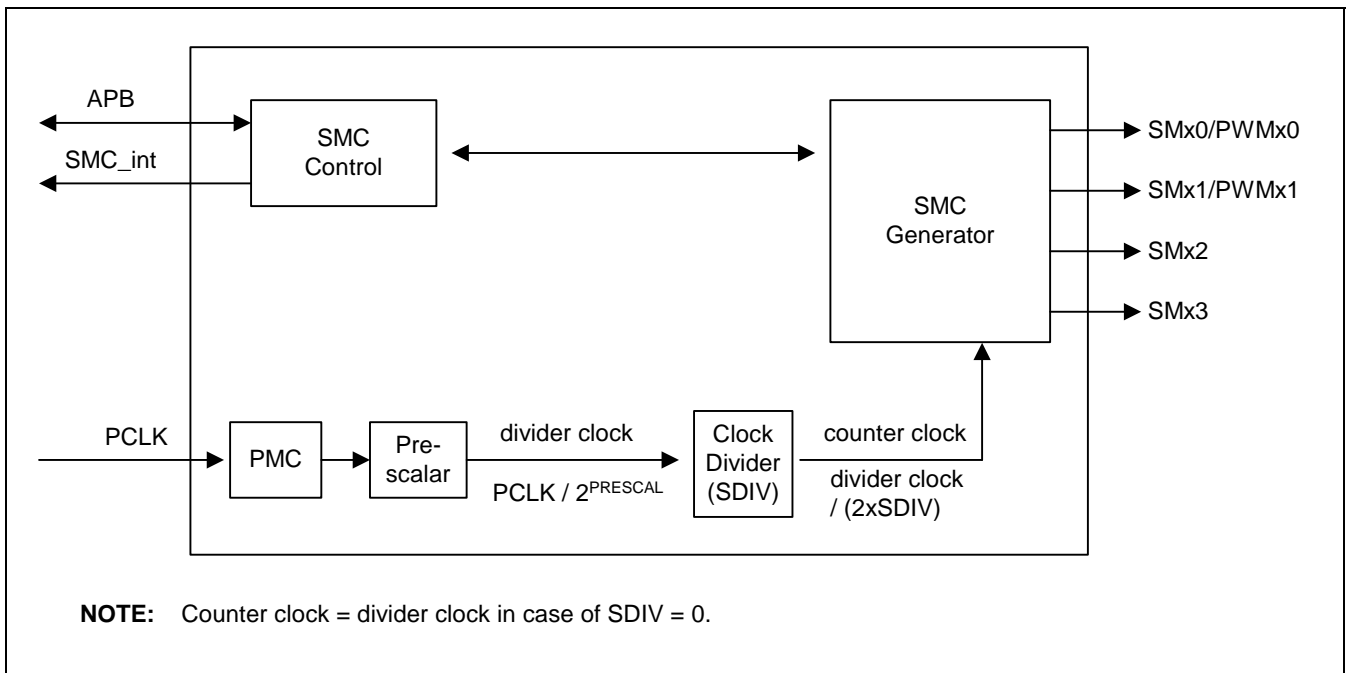


Figure 25-1. SMC Block Diagram

2. EXTERNAL PIN DESCRIPTION

Table 25-1. PWM Pin Description

Pin Name	Function	I/O Type	Active Level	Comments
SMC[3:0][3:0]	Stepper motor controller output	O	–	SMC[3:0][1:0] can be used as PWM output

3. FUNCTIONAL OPERATION

3.1 OPERATION OVERVIEW

3.1.1 General Description

The stepper motor controller offers two exclusives functional modes :

- The pulse width modulation mode (PWMM) allowing to generate two periodic signals with a defined frequency and duty cycle. In this mode, output signals are generated on SMx0 and SMx1 pins.
- The stepper motor controller mode (SMCM) allowing to generate signals to spin a stepper motor. In this mode the SMC[x3:x0] pins generate waveform according to the driving method.

The SMC FM (stepper motor controller functional mode) bit within the SMC_MR register allow to choose between both functional modes. By default, the SMC mode is selected (SMCFM bit set to '1' within the SMC_MR register).

3.2 PWM MODE

3.2.1 General Description

When the SMC module is used as a PWM generator, it is able to drive two PWM outputs : SMx0 and SMx1.

There are 3 parameters defining each signal to generate : counter frequency, delay width and pulse width. The counter frequency is the same for both channels, but delay and pulse widths are configurable per channel.

A fourth parameter allows users to change polarity of the pulse and the delay levels which is also configurable per channel. By default, the pulse is at a logical '1' level and the delay is at a logical '0' level. (See P26-21, PLx bit)

The pulse and the delay width are configurable with two 8-bit registers.

3.2.2 Counter Frequency

Basically, the PWM module is a down-counter. It counts "delay width" cycles, setting the output at delay level, then it counts "pulse width" cycles, setting the output at pulse level. This operation is repeated until the module is disabled.

The frequency of the clock uses to cadence this counter is configurable with a pre-scalar and a sub clock divider.

3.2.3 Delay and Pulse Widths

The delay width is the number of counter cycles during which the output generate the delay level.

The pulse width is the number of counter cycles during which the output generate the pulse level.

When the channel is enabled, it starts by generating a delay following by a pulse.

3.2.4 Enabling and Disabling the SMC Module in PWM Mode

When the SMC module is enabled (SMCENS is set to '1' in the SMC_SR register), writing PWMSTTx in the SMC_CR register starts the PWM generation on channel x : SM[x1:x0] starts generating a delay followed by a pulse and repeats this until a stop command. The stop command (writing PWMSTPx in the SMC_CR register) freezes the channel output in its current state. By starting again the channel, the delay-pulse cycle is restarted from the beginning : a complete delay-pulse cycle is generated whether the stop command was requested.

It is allowed to enabled the SMC module simultaneously to the start command (a single write in the SMC_CR register to enable and start the PWM generation).

When the SMC module configured as PWM generator goes from enable to disable state, PWM generations are stopped and SMx0 and SMx1 output states remain to the delay level defined in the mode register by the PLx bits. If the module is then re-enabled, the start command should be used to generate PWM.

While the SMC is disabled, the SMx0 and SMx1 output levels are fixed and do not change even if the PLx bits are modified. In case the PLx bits are toggled while the SMC module is disabled, the SMx0 and SMx1 outputs will be affected by this change when the SMC module will be enabled.

3.2.5 Min/Max Frequency Configuration Values

Table 25-2. PWM Limits Frequency

PCLK	Min PWM frequency	Max PWM frequency
4 MHz	7e-9Hz	2 MHz
20 MHz	3.5e-8Hz	10 MHz
40 MHz	7e-8Hz	20 MHz

3.3 STEPPER MOTOR MODE

3.3.1 General Description

The stepper motor controller is responsible for providing synchronous control signals which will toggle at the appropriate time in order to spin the rotor.

The stepper motor controller drives simultaneously 4 output signals. The waveform generated is defined by setting the driving method.

3.3.2 Clock Division

The speed of the stepper motor is directly related to the rate of output signal pulses. The frequency of the clock used to trig the output signal pulses is easily fine tunable by using the PRESCAL and SDIV fields in the SMC_MR register.

The new clock generated (a sub division of the system clock) is then used to sequence the output signals.

User should notice that if the output signals frequency is faster than the rotor can move, the rotor will slip until the rate is slowed enough for the rotor to again lock-in to the sequence.

3.3.3 Connecting the Stepper Motor

Bipolar stepper motor can be directly driven by the SMC module in microstep mode. The SMx0 and SMx1 pins are connected to each end of the first winding. The SMx2 and SMx3 pins are connected to each end of the second winding.

Please refers to the following figure :

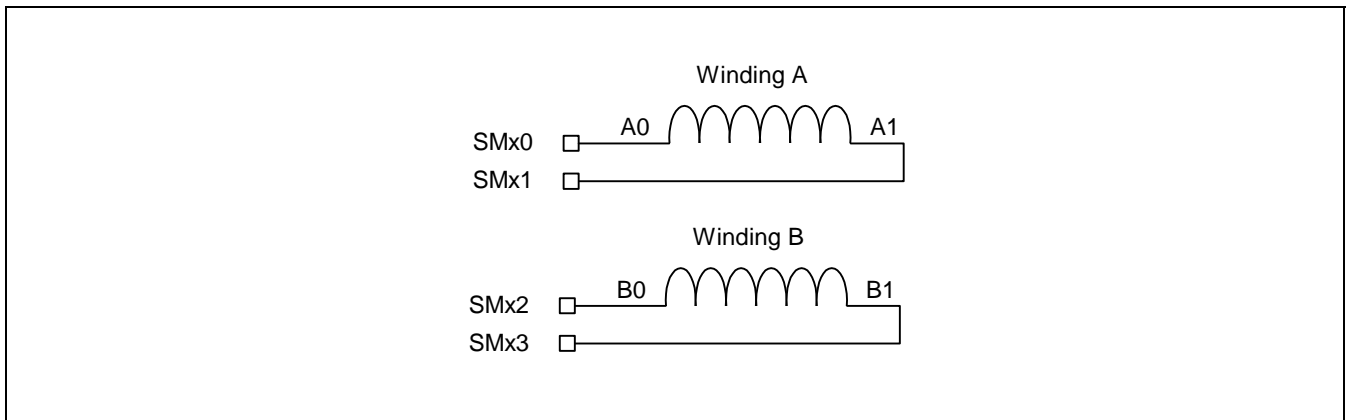


Figure 25-2. Stepper Motor Connection

3.3.4 Driving Methods

Five driving methods are supported by the stepper motor controller. Two methods are full stepping methods, allowing four positions by revolution. A half stepping method is a combination of both full stepping methods and it allows eight steps per revolution. Two microstepping methods are supported.

3.3.4.1 One phase on Full Stepping

This is the simplest way to drive stepper motors. The current is driven through only one winding at a time, so it is termed 'one phase on' method. The rotor aligns with the stator poles. It allows four full step per rotor revolution. The torque generated in this method is less, as only one winding at a time is used.

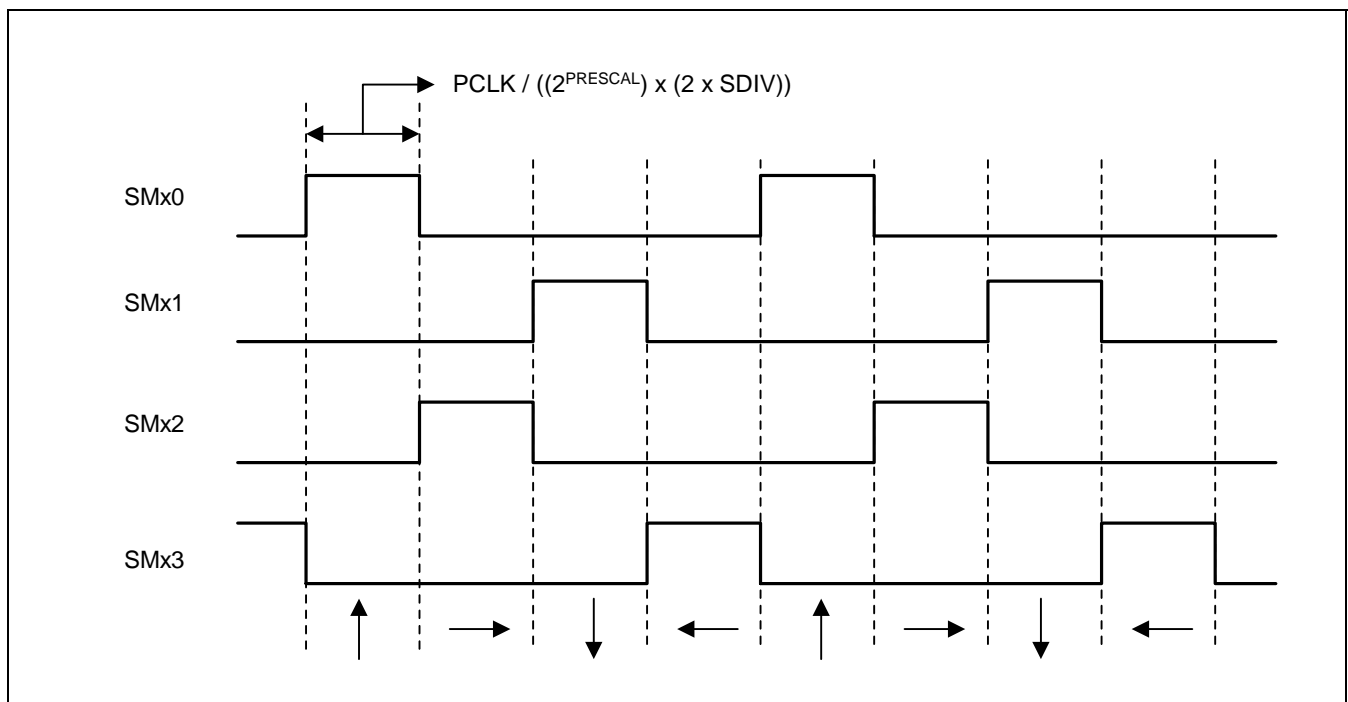


Figure 25-3. One Phase on Full Stepping

3.3.4.2 Two phase on Full Stepping

In this method, both coils of the motor are always energized, so it is termed 'two phase on' method. The rotor aligns with the half-between two stator poles. It allows four full step per rotor revolution. The torque generated in this method is improved, as two winding at a time are used.

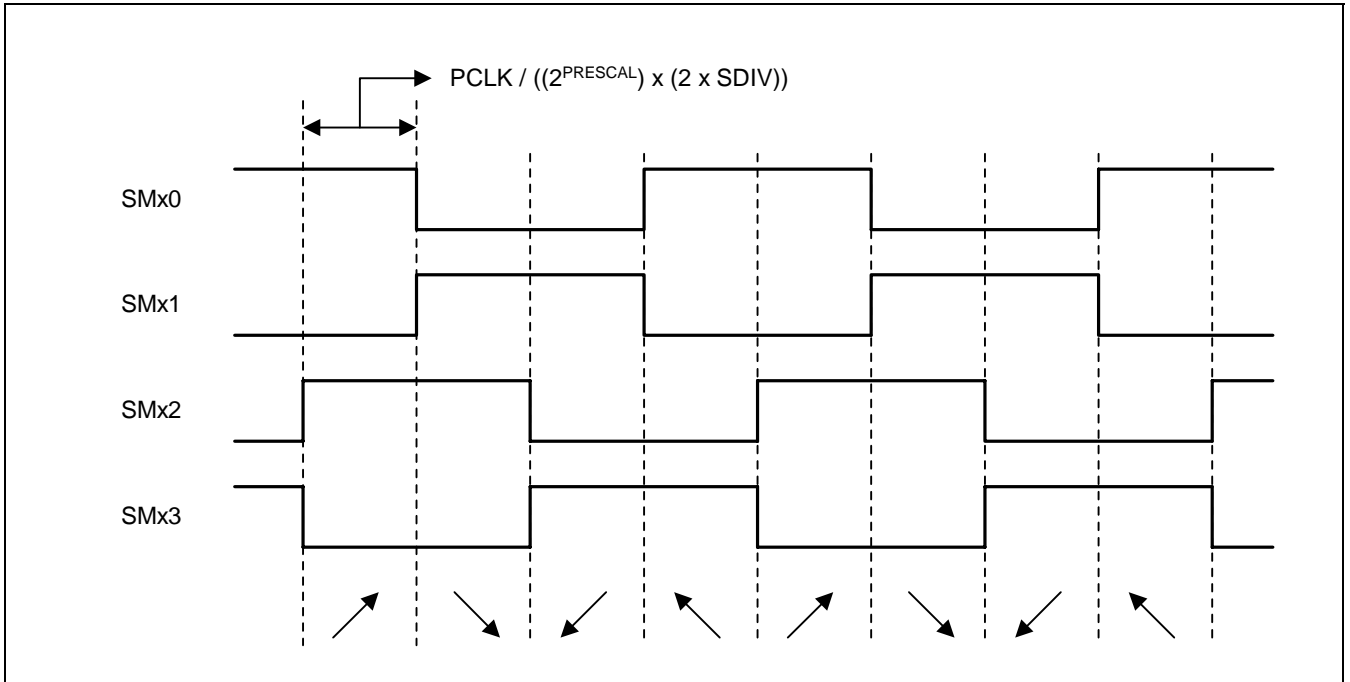


Figure 25-4. Two Phase on Full Stepping

3.3.4.3 Half Stepping

This driving method is a combination of the one phase on and two phase on full stepping driving method. Compare to a full stepping method, the number of step per revolution are doubled from four to eight. The rotor aligns with stator poles and half-between two stator poles.

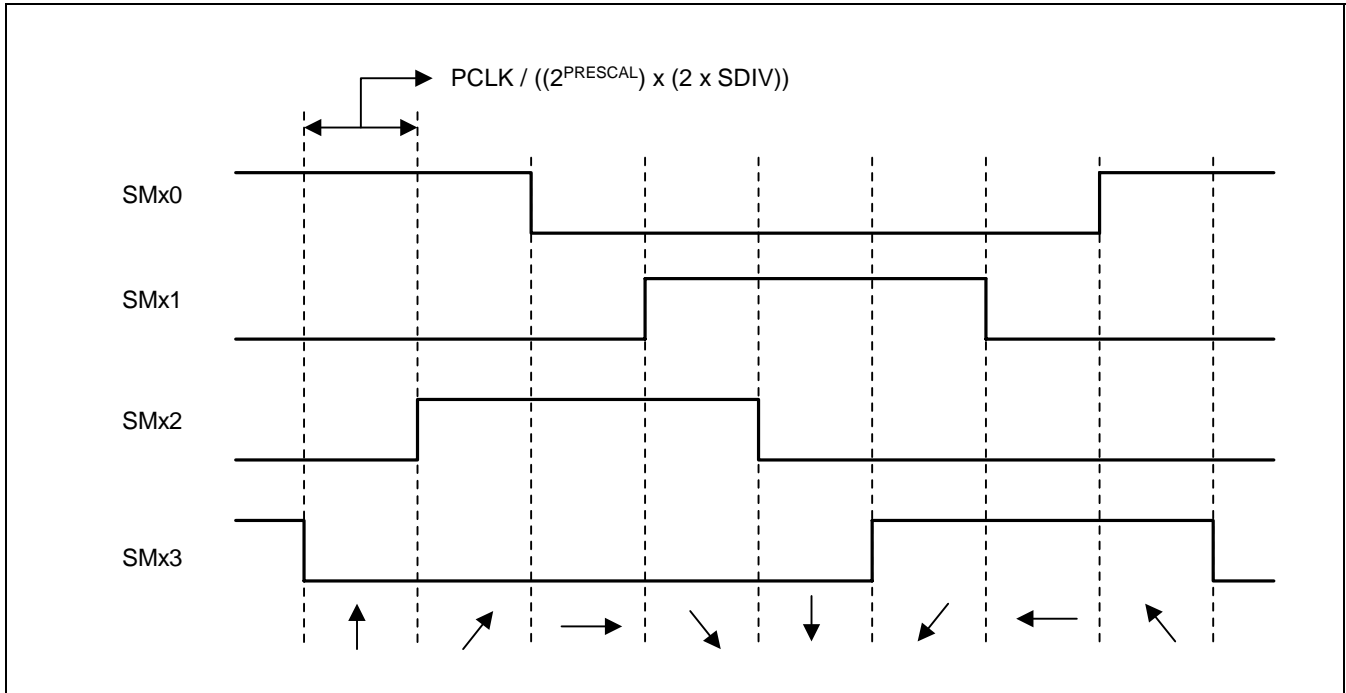


Figure 25-5. Half Stepping

3.3.4.4 Microstep Mode

In this mode the rotor moves a fraction of the full step angle. Full stepping a motor results in jerky movements of the motor, especially at lower speeds. Microstepping is used to achieve increased step resolution and smoother transitions between steps. In most applications, microstepping increases system performance while limiting noise and resonance problems. The microstep mode is selected by setting to '0x4' or '0x5' the DM[2:0] field (driving method) within the SMC_MR register.

3.3.4.4.1 Microstep Achieves with PWM

When the value of the current in a particular coil is either 'no current' or 'a rated current', the rotor aligns with the stator poles (one phase on driving method) or it aligns itself half-between two stator poles (two phase on driving method). In order to create intermediate rotor position, the magnitude of the current in both coils is controlled. This is achieved by pulse-width modulating the voltage across the windings of a motor. It results in a current that is proportional to the duty cycle of the modulated signal.

The transition between a coil being energized in one direction and then energized in the other direction has a sinusoidal shape. This shape gives the smoothest transition between the motor's full step increments.

Rather than calculating the duty cycle for a particular microstep on the fly, a duty cycle look-up table is stored in a ROM.

The PWM cycle is composed of delay (output driven at .0.) and pulse (output driven to .1.). In order to close up at maximum the sinusoidal shape, generated PWM cycles are started by delay during falling phase (output current decreasing from 100% to 0%). But, generated PWM cycles are started by pulse for the rising phase (output current increasing from 0% to 100%).

3.3.4.4.2 Cosinus / Sinus Microstepping Technique

A microstepping technique known as cosinus/sinus microstepping adjusts the current in each winding so the resultant torque is constant. In an ideal motor, the torque produced by each winding is proportional to the current in that winding, and the torques add linearly.

This microstepping technique is selected by setting to '0x4' the DM[2:0] field (driving method) within the SMC_MR register.

Please refer to the following graph showing the current in windings versus the angular position.

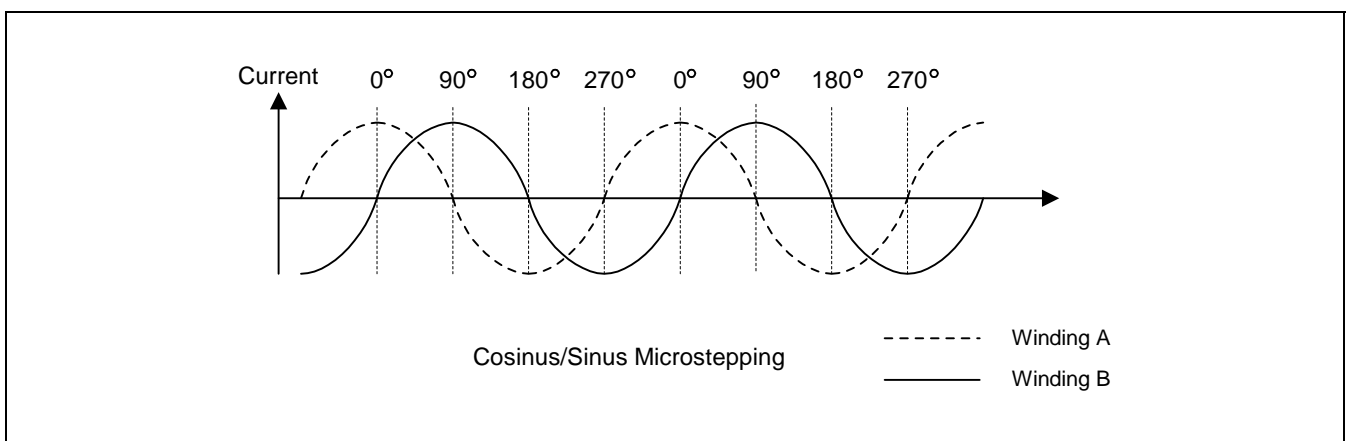


Figure 25-6. Cosinus/Sinus Microstepping

Note that current polarity depends only of the current direction in the winding. A positive current flows from one winding end to the other end and a negative current flows in the reverse direction.

3.3.4.4.3 High Torque Microstepping Technique

A second way to implement microstepping maximizes torque in a bipolar stepping motor, though the torque is not constant while the motor turns. In this method, known as high torque microstepping, one winding is energized while the current flow in the other winding is ramped down, reversed and then ramped up again. The second winding then remains energized while the first winding undergoes the polarity reversal.

This microstepping technique is selected by setting to '0x5' the DM[2:0] field (driving method) within the SMC_MR register.

Please refer to the following graph showing the current in the winding versus the angular position.

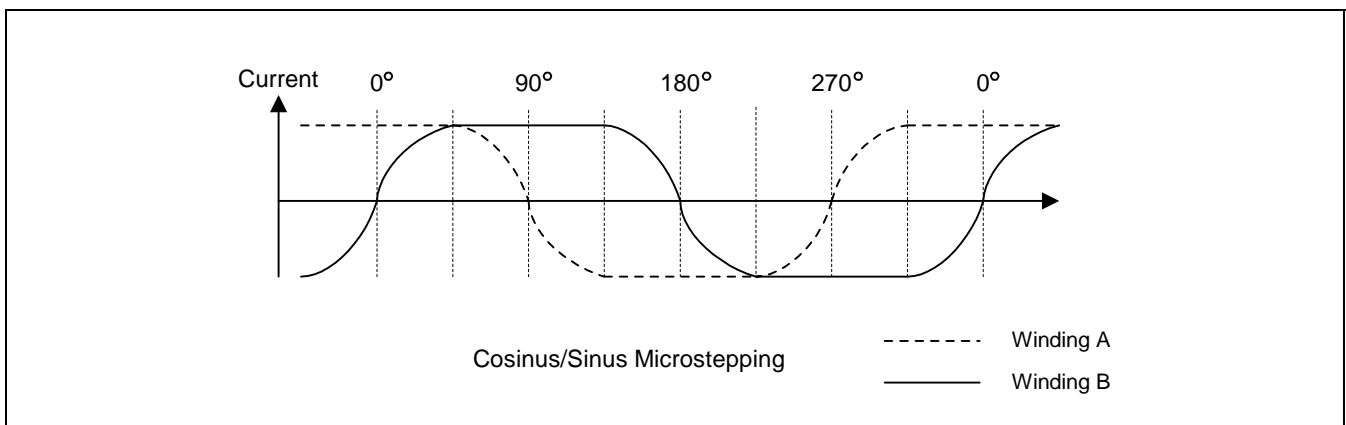


Figure 25-7. High Torque Microstepping

Note that current polarity depends only of the current direction in the winding. A positive current flows from one winding end to the other end and a negative current flows in the reverse direction.

3.3.4.4.4 Number of Microsteps

The number of microsteps by sinusoid quarter is configurable through the NMSQ[1:0] field within the SMC_MR register and can be selected among 4, 8, 16 or 32 microsteps.

It is important to note that 4 microsteps by sinusoid quarter will result in 16 microsteps by revolution for the cosinus/sinus driving method, and in 32 microsteps per revolution for the high torque driving method.

The number of pulse width modulation cycle is generated for a particular microstep can be configured using the NCM field (number of cycle by microstep) within the SMC_MR register. By repeating a lot short PWM cycle for a microstep, the average current flowing in the winding will be smoother avoiding irregularity due to the winding charge followed by winding discharge.

Rather than calculating the duty cycle for a particular microstep on the fly, a duty cycle look-up table is stored in a ROM.

3.3.5 Two Running Modes : Continuous or Target

The stepper motor controller can generate output signals continuously or during a given period according to the two distinct running modes : continuous mode and target mode.

In continuous mode, once the stepper motor controller is started by writing the MSTP bit in the SMC_CR register, it generates output signals continuously according to the driving method. The motor can be locked by stopping the stepper motor controller (MSTP bit in the SMC_CR register). If it is started again, the generated waveform resumes from where it was left. By setting to a logical 1 the DIR (direction) bit, the waveform are generated in reverse causing the stepper motor to rotate backward.

In target mode, the stepper motor controller generates a fixed number of steps which are successive states of the waveform, and they can be generated forward or reverse. Once the stepper motor has been started, by writing the target position to the SMC_TPR (Target Position Register) the stepping motor controller generates output signals according to the waveform which will spin the motor. When the number of rotate steps allowing to reach the target position has been executed, the PR (Position Reached) bit rises in the SMC_SR register and the output signals are frozen involving to lock the motor in its current position. This position will not change until the next write in the SMC_TPR register, and when this new request is performed by writing this register, the waveform generation resumes from where it was left. If a new command is made before the current is finished, the current command is interrupted and does not end, and the new command is started from this point. The DIR_SIGN (direction) bit of the SMC_TPR register has no effect in target mode.

The direction (forward or reverse) is computed automatically by the SMC logic according to the current position and the target position.

3.3.6 Current Motor Position

The current motor position is permanently readable in the SMC_CPR (Current Position) register. This register is initialized to 0 at power up or consequently to a software reset, and then its display the number of step already done by the stepper motor. In case the SMC module is currently generating pulses in order to spin the motor, the value of the SMC_CPR register is continuously updated to reflect the current number of steps.

The NEG (Negative) bit is used to indicate whether the number of step has rotate the motor forward or reverse. NEG equals to '1' means that the motor has rotated reverse.

In case the current motor position overflows, it restarts counting from 0 and the CPO (current position overflow) flag within the SMC_SR register is set.

3.3.7 SMC Interrupts

The stepper motor controller can generate an interrupt on several events :

- Requested position is reached,
- The current position has overflowed.

3.3.8 Enabling and Disabling the SMC

When the SMC module used as stepper motor controller goes from enable to disable state, the internal state machine sets all outputs to '0'. When the SMC module is re-enabled, it starts again from the reset position.

When the SMC module configured as stepper motor controller is enabled for the first time, it drives on the SMC[3:0] outputs :

- 0001b if configured in one phase on, half stepping or microstepping driving modes;
- 0101b if configured in two phase on driving mode.

It is allowed to request simultaneously to enable the SMC and to start the motor configured in continuous mode.

3.4 PROGRAMMING EXAMPLES

3.4.1 PWM Mode

If the counter frequency is set to $PCLK/2$, delay width is equal to 4 cycles and pulse width to 2 cycles :

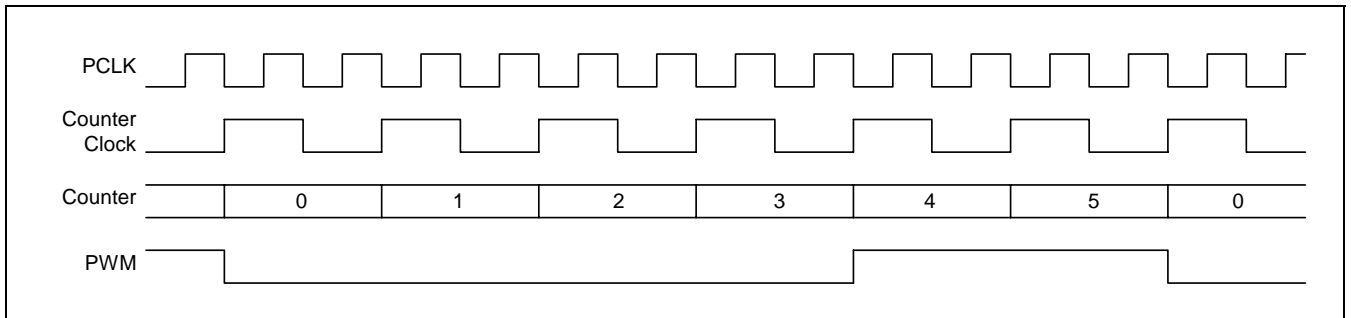


Figure 25-8. PWM Example

3.4.2 Example of use of the PWM :

We want to generate a pulse with a frequency of 5 kHz with a duty cycle of 50%, $PCLK = 30\text{MHz}$.

Configuration :

- Enable the clock on PWM peripheral by writing bit SMC in SMC_ECR.
- Do a software reset of the PWM peripheral to be in a known state by writing bit SWRST in SMC_CR.
- Configuration of SMC_MR : Let say we use a SMC clock of 15 MHz with a PRESCAL of 0 and the pulse is at a logical state 1. Users can choose one of the four pwm available.
- Configuration of SMC_DLYX : This register indicate the number of SMC clock needed to form the delay (level 0), let say 1500 which should give us half a period of a 0.1 ms.
- Configuration of SMC_PULX : This register indicate the number of SMC clock needed to form the pulse (level 1), let say 1500 which should give us half a period of a 0.1ms.
- Start the chosen PWM by writing the SMCEN and PWMSTTx in SMC_CR. This should supply a pulse of 5 kHz on the chosen PWM with a duty cycle of 50%

4. REGISTERS DESCRIPTION

Base Addresses – SMC0 : 0xFFE74000
 SMC1 : 0xFFE78000
 SMC2 : 0xFFE7C000
 SMC3 : 0xFFE80000

Table 25-3. SMC Special Function Registers

Offset Address	Name	Description	R/W	Reset Status
0x000 – 0x04C	–	Reserved	–	–
0x050	SMC_ECR	Enable clock register	W	–
0x054	SMC_DCR	Disable clock register	W	–
0x058	SMC_PMSR	Power management status register	R	0XXXXXXXX0
0x05C	–	Reserved	–	–
0x060	SMC_CR	Control register	W	–
0x064	SMC_MR	Mode register	R/W	0x000000A0
0x068	–	Reserved	–	–
0x06C	SMC_CSR	Clear status register	W	–
0x070	SMC_SR	Status register	R	0x00000000
0x074	SMC_IER	Interrupt enable register	W	–
0x078	SMC_IDR	Interrupt disable register	W	–
0x07C	SMC_IMR	Interrupt mask register	R	0x00000000
0x080	SMC_DLY0	Delay register 0	R/W	0x00000000
0x084	SMC_PUL0	Pulse register 0	R/W	0x00000000
0x088	SMC_DLY1	Delay register 1	R/W	0x00000000
0x08C	SMC_PUL1	Pulse register 1	R/W	0x00000000
0x090 – 0x094	–	Reserved	–	–
0x098	SMC_TPR	Target position register	R/W	0x00000000
0x09C	SMC_CPR	Current position register	R	0x00000000

SMC Enable Clock Register

SMC_ECR (0x050)

Access: Write only

31	30	29	28	27	26	25	24
DBGEN	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	SMC	–
W	W	W	W	W	W	W	W

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **SMC : SMC clock enable**

0: No effect

1: Enable SMC clock

- **DBGEN : Debug enable**

0: No effect

1: Enable debug mode

SMC Power Management Status Register

SMC_PMSR (0x058)

Access: Read only

31	30	29	28	27	26	25	24
DBGEN	–	IPIDCODE[25:20]					
R-0	R-0	R-U	R-U	R-U	R-U	R-U	R-U
23	22	21	20	19	18	17	16
IPIDCODE[19:12]							
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U
15	14	13	12	11	10	9	8
IPIDCODE[11:4]							
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U
7	6	5	4	3	2	1	0
IPIDCODE[3:0]				–	–	SMC	–
R-U	R-U	R-U	R-U	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **SMC : SMC clock status**

0: SMC clock disabled

1: SMC clock enabled

- **IPIDCODE[25:0]**

This field contains the version number of the module, coded on 26 bits.

- **DBGEN : Debug mode**

0: The DEBUG ACKNOWLEDGE generated by the ICE interface has no influence on the SMC function.

1: The DEBUG ACKNOWLEDGE freezes the PMW function when the ICE interface is activated. However full read/write access to internal register is kept for the debug purpose.

NOTE: The SMC_PMSR register is not reset by software reset.

SMC Control Register **SMC_CR (0x060)** **Access: Write only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	PWMSTP1	PWMSTT1	PWMSTP0	PWMSTT0
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	MSTP	MSTT	–	SMCDIS	SMCEN	SWRST
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

• **SWRST : SMC software reset**

0: No effect.

1: Reset the SMC

A software triggered hardware reset of the SMC is performed. It resets all the registers except the SMC_PMSR register.

• **SMCEN : Enable SMC**

0: No effect.

1: Enables the SMC.

• **SMCDIS : Disable SMC**

0: No effect.

1: Disables the SMC.

In case both SMCEN and SMCDIS are equal to one the SMC will be disabled.

• **MSTT : Motor start**

0: No effect.

1: Start the generation of the waveform according to the driving method.

- **MSTP : Motor stop**

0: No effect.

1: Stop the generation of the waveform. The state machine is freezes in its current state involving the motor to keep its current position.

In case both MSTT and MSTP are equal to one the motor is stopped.

NOTE: MSTT and MSTP bits are usable only when the SMC module is used as stepper motor controller in continuous mode.

- **PWMSTTx : Start PWM channel x**

0: No effect.

1: Start PWM generation on SMCx.

- **PWMSTPx : Stop PWM channel x**

0: No effect.

1: Stop PWM generation on SMCx. The output is frozen.

In case both PWMSTTx and PWMSTPx are equal to one the PWM channel x will be stopped.

NOTE: PWMSTTx and PWMSTPx bits have no effect when the SMC module is not used in PWM functional mode.

SMC Mode Register **SMC_MR (0x064)** **Access: Read/Write**

31	30	29	28	27	26	25	24
PL1	PL0	–	–	–	–	–	–
R/W-1	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
DM[2:0]			NMSQ[1:0]		NCM[2:0]		
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
SDIV[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
SMCFM	CONT	–	PRESCAL0[4:0]				
R/W-1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

• **PRESCAL[4:0] : Clock prescaler**

- 0: Intermediate clock frequency = PCLK
- 1–31: Intermediate clock frequency = PCLK / 2^{PRESCALAR}

• **CONT : Continuous mode**

- 0: Target mode. The SMC is set to target mode.
- 1: Continuous mode. The SMC is set to continuous mode.

NOTES:

1. This bit has no effect when the SMC module is used in PWM functional mode.
2. Toggle this bit is forbidden when the module is enabled (SMENS = 1 within the SMC_SR)

• **SMCFM : SMC functional mode**

- 0: The SMC module is used as PWM generator.
- 1: The SMC module is used as stepper motor controller.

NOTE: Toggle this bit is forbidden when the module is enabled (SMENS = 1 within the SMC_SR).

• **SDIV[7:0] : Sub clock divider**

- 0: Counter Clock frequency = Intermediate clock frequency
- 1–255: Counter Clock frequency = Intermediate clock frequency / (2 x SDIV)

• **NCM[2:0] : Number of PWM cycle by microstep**

This field defines the number of PWM cycle to generate for each microstep.

Table 25-4. Relation Between NCM[2:0] and the Number of PWM Cycle by microstep

NCM[2:0]	Number of PWM Cycle by microstep
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128

NOTES:

1. This bit has no effect when the SMC module is used in PWM functional mode.
2. Toggle this bit is forbidden when the module is enabled (SMENS = 1 within the SMC_SR).

- **NMSQ[1:0] : Number of microstep by sinusoid quarter**

This field defines the number of microstep by sinusoid quarter. According to the table below, the number of microstep can be selected between 4, 8, 16 or 32.

Table 25-5. Relation Between NMSQ[1:0] and the Number microstep by Sinusoid Quarter

NMSQ[1:0]	Number microstep by sinusoid quarter
0	2
1	4
2	8
3	16

NOTES:

1. This bit has no effect when the SMC module is used in PWM functional mode.
2. Toggle this bit is forbidden when the module is enabled (SMENS = 1 within the SMC_SR).

- **DM[2:0] : Driving method**

This field defines the driving method to be used.

Table 25-6. Relation Between DM[2:0] and the Driving Method

DM[2:0]	Driving Method
000	One phase on
001	Two phase on
010	Half stepping
100	Cosinus/Sinus microstepping
101	High torque microstepping
Others	Reserved

NOTES:

1. This bit has no effect when the SMC module is used in PWM functional mode.
2. Toggle this bit is forbidden when the module is enabled (SMENS = 1 within the SMC_SR).

- **PLx : Pulse on channel x**

- 0: The pulses are at logic level 0. During the delay the output is at logic level 1.
 1: The pulses are at logic level 1. During the delay the output is at logic level 0.

NOTES:

1. Each PWM cycle starts with a delay.
2. This bit has no effect in SMC mode.
3. Toggle this bit is forbidden when the module is enabled (SMENS = 1 within the SMC_SR).

SMC Clear Status Register

SMC_CSR (0x06C)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	CPO	PR	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	PEND1	PSTA1	PEND0	PSTA0
W	W	W	W	W	W	W	W

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **PSTAx : Clear pulse start interrupt on channel x**

0: No effect.

1: Clear PSTAx interrupt.

NOTE: Setting this bit to '1' is forbidden when the SMC module is used as a stepper motor controller (SMCFM bit is at logical level '1').

- **PENDx : Clear pulse end interrupt on channel x**

0: No effect.

1: Clear PENDx interrupt.

NOTE: Setting this bit to '1' is forbidden when the SMC module is used as a stepper motor controller (SMCFM bit is at logical level '1').

- **PR : Position Reached**

0: No effect.

1: Clear position reached interrupt.

NOTE: Setting this bit to '1' is forbidden when the SMC module is used as a PWM (SMCFM bit is at logical level '0').

- **CPO : Current Position Overflow**

0: No effect.

1: Clear current position overflow interrupt.

NOTE: Setting this bit to '1' is forbidden when the SMC module is used as a PWM (SMCFM bit is at logical level '0').

SMC Status Register **SMC_SR (0x070)** **Access: Read only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	CPO	PR	SMCENS
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
–	–	–	–	PEND1	PSTA1	PEND0	PSTA0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

• **PSTAx**

- 0: No pulse started on channel x since the last time this bit was cleared.
- 1: At least a pulse started on channel x since the last time this bit was cleared.

• **PENDx**

- 0: No pulse ended on channel x since the last time this bit was cleared.
- 1: A pulse ended on channel x since the last time this bit was cleared

• **SMCENS**

- 0: SMC module is disabled.
- 1: SMC module is enabled.

• **PR : Position Reached**

- 0: The motor position requested has not been yet reached since the last time this bit was cleared.
- 1: A motor position requested has been reached at least one time since the last time this bit was cleared.

• **CPO : Current Position Overflow**

- 0: The current position counter has not overflow since the last time this bit was cleared.
- 1: The current position counter has overflow at least one time since the last time this bit was cleared.

SMC Interrupt Enable Register

SMC_IER (0x074)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	CPO	PR	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	PEND1	PSTA1	PEND0	PSTA0
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **PSTAx: Pulse start interrupt on channel x enable**

0: No effect

1: Enable the PSTAx interrupt

- **PENDx: Pulse end interrupt on channel x enable**

0: No effect

1: Enable the PENDx interrupt

- **PR: Position Reached interrupt enable**

0: No effect

1: Enable the PR interrupt

- **CPO: Current Position Overflow interrupt enable**

0: No effect

1: Enable the CPO interrupt

SMC Interrupt Disable Register **SMC_IDR (0x078)** **Access: Write only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	CPO	PR	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	PEND1	PSTA1	PEND0	PSTA0
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **PSTAx: Pulse start interrupt on channel x disable**

- 0: No effect
- 1: Disable the PSTAx interrupt

- **PENDx: Pulse end interrupt on channel x disable**

- 0: No effect
- 1: Disable the PENDx interrupt

- **PR: Position Reached interrupt disable**

- 0: No effect
- 1: Disable the PR interrupt

- **CPO: Current Position Overflow interrupt disable**

- 0: No effect
- 1: Disable the CPO interrupt

SMC Interrupt Mask Register

SMC_IMR (0x07C)

Access: Read only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	CPO	PR	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
–	–	–	–	PEND1	PSTA1	PEND0	PSTA0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **PSTAx: Pulse start interrupt on channel x mask**

0: Pulse Start Interrupt is disabled on channel x.

1: Pulse Start Interrupt is enabled on channel x.

NOTE: Setting this bit to '1' is forbidden when the SMC module is used as a stepper motor controller (SMCFM bit is at logical level '1').

- **PENDx: Pulse end interrupt on channel x mask**

0: Pulse End Interrupt is disabled on channel x.

1: Pulse End Interrupt is enabled on channel x.

NOTE: Setting this bit to '1' is forbidden when the SMC module is used as a stepper motor controller (SMCFM bit is at logical level '1').

- **PR : Position Reached interrupt mask**

0: Position reached interrupt is disabled.

1: Position reached interrupt is enabled.

NOTE: Setting this bit to '1' is forbidden when the SMC module is used as a PWM (SMCFM bit is at logical level '0').

- **CPO : Current Position Overflow interrupt mask**

0: Current position overflow interrupt is disabled.

1: Current position overflow interrupt is enabled.

NOTE: Setting this bit to '1' is forbidden when the SMC module is used as a PWM (SMCFM bit is at logical level '0').

SMC Delay Register 0
SMC Delay Register 1

SMC_DLY0 (0x080)
SMC_DLY1 (0x088)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
DELAY[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

25-1. IMPORTANT NOTICE

Write access in this register is forbidden if the module is configured in SMC mode

- **DELAY[7:0] : delay width**

Numbers of counter cycles during which the output is inactive.

NOTE: Change of this value is immediately taken into account. This may cause a bad width of the current delay.

SMC Pulse Register 0
SMC Pulse Register 1

SMC_PUL0 (0x084)
SMC_PUL1 (0x08C)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
PULSE[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

25-2. IMPORTANT NOTICE

Write access in this register is forbidden if the module is configured in SMC mode

- **PULSE[7:0] : Pulse Width**

Numbers of counter cycles during which the output is active.

NOTE: Change of this value is immediately taken into account. This may cause a bad width of the current pulse.

SMC Target Position Register **SMC_TPR (0x098)** **Access: Read/Write**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
DIR_SIGN	RP[14:8]						
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
RP[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

25-3. IMPORTANT NOTICE

Write access in this register is forbidden if the module is configured in PWM mode

• **RP[14:0] : Requested position**

Command the stepper motor to generate appropriate signal to reach the requested motor position.

• **DIR_SIGN : Direction or sign**

- 0: When configured in continuous mode, the motor moves in forward direction. When configured in target mode, the motor position to reach is positive.
- 1: When configured in continuous mode, the motor moves in reverse direction. When configured in target mode, the motor position to reach is negative.

NOTES:

- 1. The scope of reachable position in target mode ranges from minus 0x7FFF to plus 0x7FFF.
- 2. The DIR_SIGN bit has no effect in target mode. The direction (forward or reverse) is computed automatically by the SMC logic according to the current position and target position. (See P26-11).

SMC Current Position Register

SMC_CPR (0x09C)

Access: Read only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
NEG	CP[14:8]						
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
CP[7:0]							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **CP[14:0] : Current position**

This field gives the number of steps executed since the last reset, power up or counter overflow.

- **NEG : Negative position**

0: The number of steps executed since power up or since the last reset or since the last overflow is positive.

1: The number of steps executed since power up or since the last reset or since the last overflow is negative.

NOTE: In case the current position has overflowed while counting negative steps, if the counter reverses and counts forward then when it reaches '0' the neg flag is cleared. Example : it counts backward and overflows : -32766, -32767, -0, -1, -2, ... and then it reverses and counts forward : -2, -1, 0, 1, 2, 3 ... The negative flag before the overflow is not memorized.

26

UNIVERSAL SYNC/ASYNC RECEIVER/TRANSMITTER (USART)

1. OVERVIEW

1.1 FUNCTIONAL DESCRIPTION

The Universal Asynchronous/Synchronous Receiver/Transmitter (USART) controller is used for communications between microcontrollers. The USART takes bytes of data and transmits the individual bits sequentially (Low Significant Bit first). At the destination, a second USART re-assembles the bits into complete bytes.

Serial transmission is commonly used with modems and for non-networked communication between computers, terminals and other devices.

There are two primary forms of serial transmission : Synchronous and Asynchronous. The asynchronous mode uses two lines for data transfer : RX for the reception and TX for the transmission or of the bits. The synchronous mode used an additional clock signal used to strobe input and output data.

The transmitter and receiver module of UART/USART are connected to the Lite Direct Memory Access Controller. The Lite Direct Memory Access Controller allows the transmission or reception of data bytes without the microcontroller need. See Lite Direct Memory Access Controller chapter for more information.

The main features are :

- Programmable baud rate generator
- Parity, framing and overrun error detection
- Idle flag for J1587 protocol
- Line break generation and detection
- Automatic echo, local loopback and remote loopback channel modes
- Multi-drop mode : address detection and generation
- Interrupt generation
- 2 dedicated LDMA channels per USART
- 5 to 9 bit character length
- Support the LIN protocol : LIN1.2 or LIN2.0 configurable release
- Smart-Card protocol : error signaling and re-transmission
- Asynchronous mode maximum baud rate : $PCLK / 16$
- Synchronous mode maximum baud rate when providing SCK clock : $PCLK / 2$
- Synchronous mode maximum baud rate when receiving SCK clock : $PCLK / 4$

1.2 BLOCK DIAGRAM

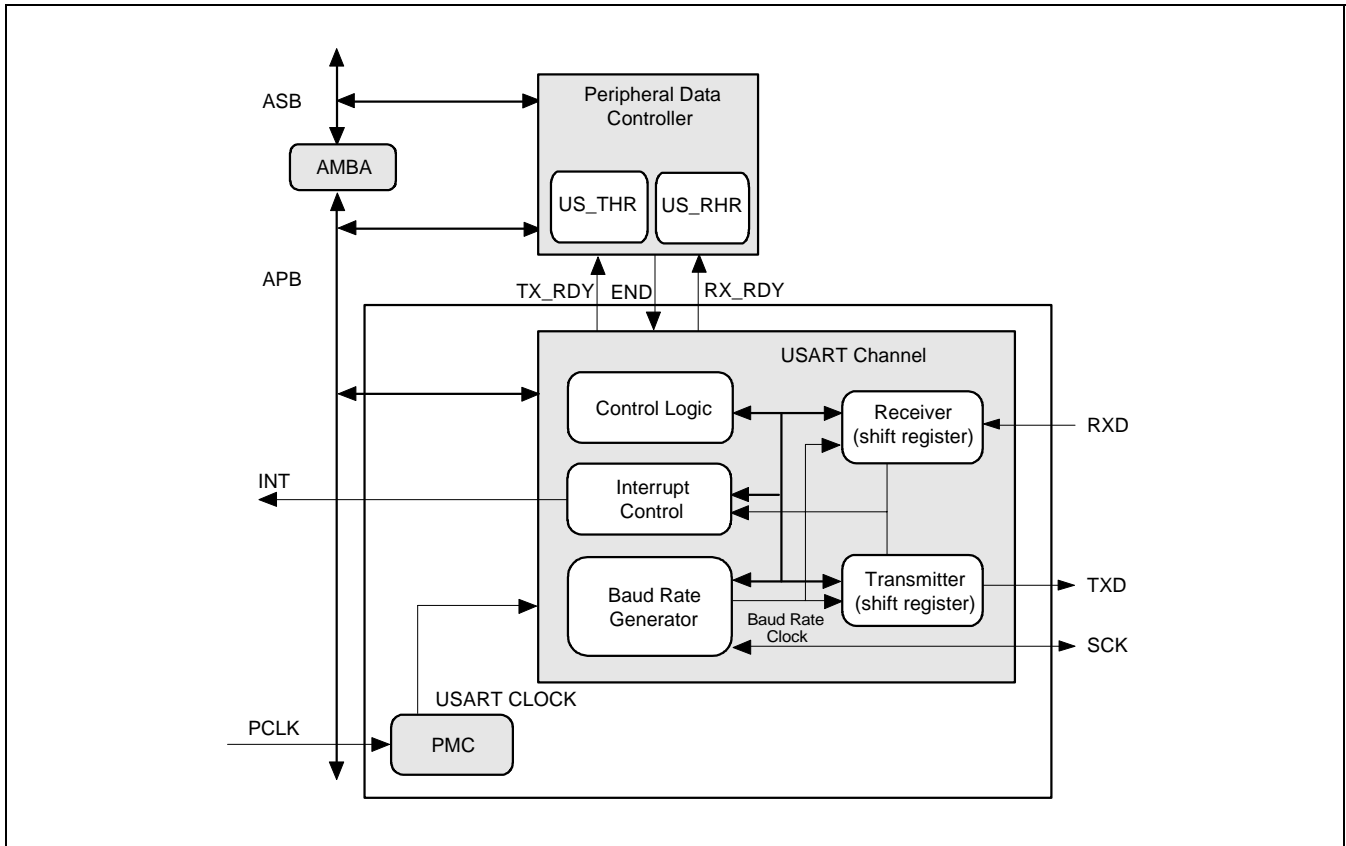


Figure 26-1. USART Block Diagram

2. EXTERNAL PIN DESCRIPTION

Table 26-1. USART Pin Description

Pin Name	Function	I/O Type	Active Level	Comments
UARTTXD0, UARTTXD1, USARTTXD0	USART transmit line	O	–	–
UARTRXD0, UARTRXD1, USARTRXD0	USART reception line	I	–	–
USARTCLK0	USART transmission clock	Bi-Direction	–	–

3. FUNCTIONAL OPERATION

3.1 BAUD RATE GENERATOR

3.1.1 General Description

The baud rate generator provides the bit period clock named the baud rate clock to both the receiver and the transmitter. The baud rate generator can select between external and internal clock sources. The external clock source is USARTCLK0. The internal clock sources can be either PCLK or PCLK divided by 8 (PCLK/8).

NOTE

In all cases, if an external clock is used, the duration of each of its levels must be longer than the PCLK period. The external clock frequency must be less than 40% of PCLK frequency.

Asynchronous mode

When the USART is programmed to operate in asynchronous mode (SYNC = 0 in the Mode Register US_MR), the selected clock is divided by 16 times the value (CD) written in US_BRGR (Baud Rate Generator Register). If US_BRGR is set to 0, the baud rate clock is disabled.

Baud rate = Selected Clock / 16 x CD where selected clock is either PCLK, PCLK/8 or USARTCLK0

Synchronous mode

When the USART is programmed to operate in synchronous mode (SYNC = 1 in the Mode Register US_MR) and the selected clock is internal (CLKS[1] = 0 in the Mode Register US_MR), the Baud Rate Clock is the internal selected clock divided by the value written in US_BRGR. If US_BRGR is set to 0, the Baud Rate Clock is disabled.

Baud Rate = Selected Clock / CD where selected clock is either PCLK, PCLK/8 or USARTCLK0

In synchronous mode with external clock selected (CLKS[1] = 1 in the Mode Register US_MR), the clock is provided directly by the signal on the USARTCLK0 pin. No division is active. The value written in US_BRGR has no effect.

3.1.2 Block Diagram

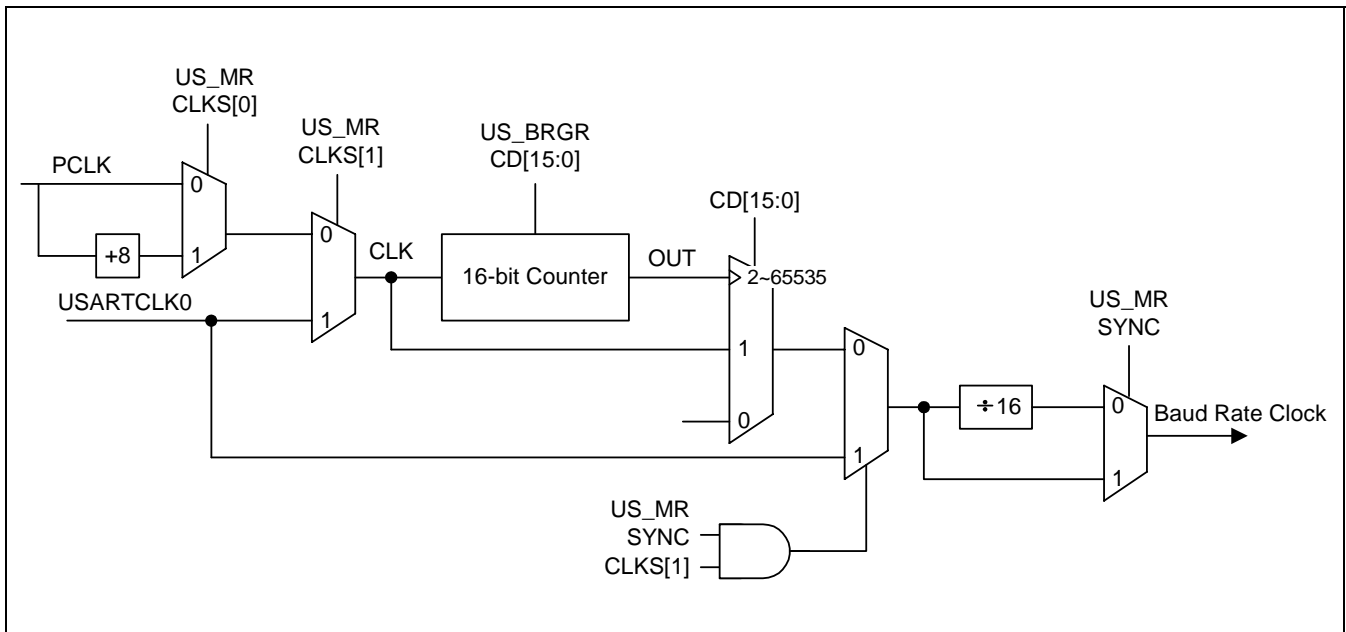


Figure 26-2. USART Baud Rate Generator Block Diagram

3.1.3 Baud Rate Configuration Example

The following table shows different registers configuration of the US_BRGR register for different core frequency. For each case, the calculated error shows the difference between the real baud rate and the expected one.

In the following table, CLKS[1:0]=00 (PCLK selected as USART clock USARTCLK) and SYNC = 0 (asynchronous mode) in the US_MR register.

Table 26-2. Asynchronous Mode (SYNC = 0)

PCLK	Baud Rate	US_BRGR CD[15:0]	% Error
4 MHz	9600	0x001A	0.16%
	125K	0x0002	0.00%
	192K	0x0001	30.2%
20 MHz	9600	0x0082	0.16%
	125K	0x000A	0.00%
	192K	0x0007	6.99%
40 MHz	9600	0x0104	0.16%
	125K	0x0014	0.00%
	192K	0x000D	0.16%

Table 26-3. Synchronous Mode (SYNC = 1)

PCLK	Baud Rate	US_BRGR CD[15:0]	% Error
4 MHz	9600	0x01A0	0.16%
	125K	0x0020	0.00%
	192K	0x0014	4.16%
20 MHz	9600	0x0823	0.016%
	125K	0x00A0	0.00%
	192K	0x0068	0.16%
40 MHz	9600	0x1046	0.016%
	125K	0x0140	0.00%
	192K	0x00D0	0.16%

3.2 RECEIVER

3.2.1 Asynchronous Receiver

The USART is configured for asynchronous operation when SYNC = 0 (bit 8 of US_MR). In asynchronous mode, the USART detects the start of a received character by sampling the RXD signal until it detects a valid start bit. A low level (space) on RXD is interpreted as a valid start bit if it is detected for more than 7 cycles of the sampling clock, which is 16 times the baud rate. Hence a space which is longer than 7/16 of the bit period is detected as a valid start bit. A space that is 7/16 of a bit period or shorter is ignored and the receiver continues to wait for a valid start bit.

When a valid start bit has been detected, the receiver samples the RXD at the theoretical mid-point of each bit. It is assumed that each bit lasts 16 cycles of the sampling clock (one bit period) so the sampling point is 8 cycles (0.5 bit periods) after the start of the bit. The first sampling point is therefore 24 cycles (1.5 bit periods) after the falling edge of the start bit was detected. Each subsequent bit is sampled 16 cycles (1 bit period) after the previous one.

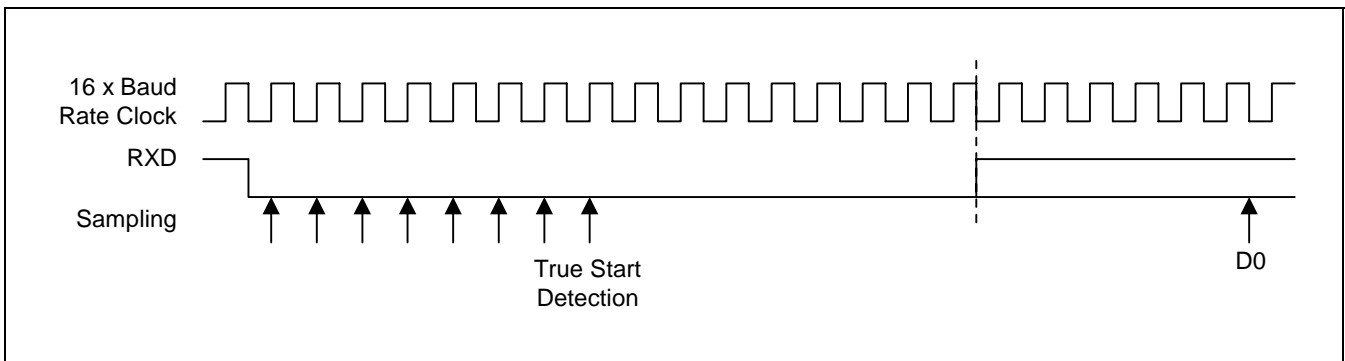


Figure 26-3. Asynchronous Mode, Start Bit Detection

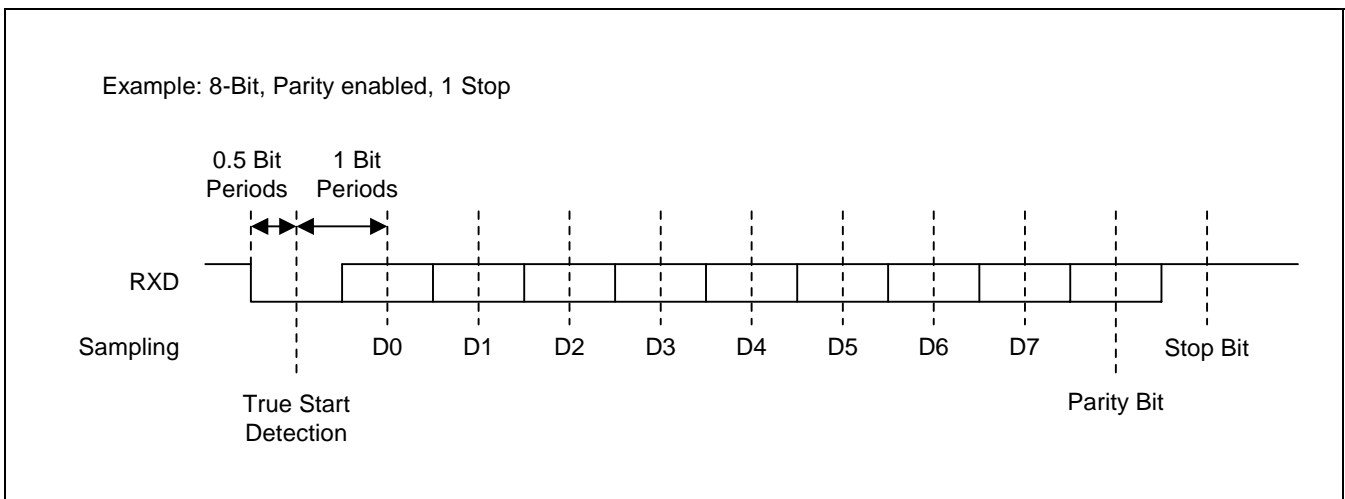


Figure 26-4. Asynchronous Mode, Character Reception

3.2.2 Synchronous Receiver

When configured for synchronous operation ($SYNC = 1$), the receiver samples the RXD signal on each rising edge of USARTCLK0. If a low level is detected, it is considered as a start. Data bits, parity bit and stop bit are sampled and the receiver waits for the next start bit. See example below.

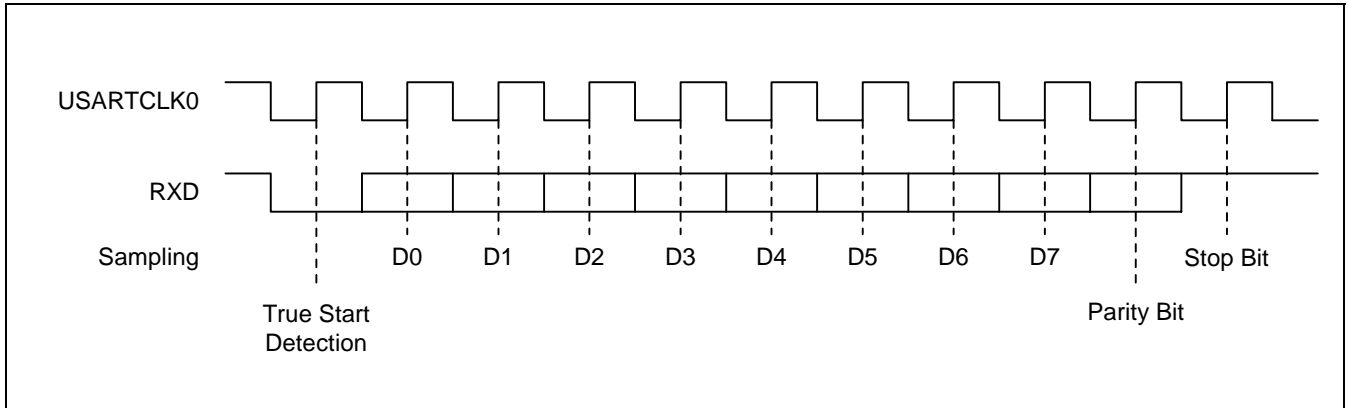


Figure 26-5. Synchronous Mode, Character Reception

3.2.3 Receiver Ready

When a complete character is received, it is transferred to the US_RHR and the RXRDY status bit in US_SR is set. The RXRDY is set after the last stop bit.

If the US_RHR register has not been read since the last transfer and before a transfer to the US_RHR register takes place, the OVRE status bit is set in US_SR register.

3.2.4 Overrun Error

If US_RHR has not been read since the last transfer, the OVRE status bit in US_SR is set.

3.2.5 Parity Error

Each time a character is received, the receiver calculates the parity of the received data bits, in accordance with the field PAR[2:0] in US_MR (Page 27-21). It then compares the result with the received parity bit. If different, the parity error bit PARE in US_SR is set.

3.2.6 Framing Error

If a character is received with a stop bit at low level and with at least one data bit at high level, a framing error is generated. This sets FRAME in US_SR.

3.2.7 Idle Flag

The idle flag turns low when USART receive a start bit and turn high at the end of a J1587 protocol frame (after 10 stop bits). An interrupt can be generated on the rising edge of the idle flag.

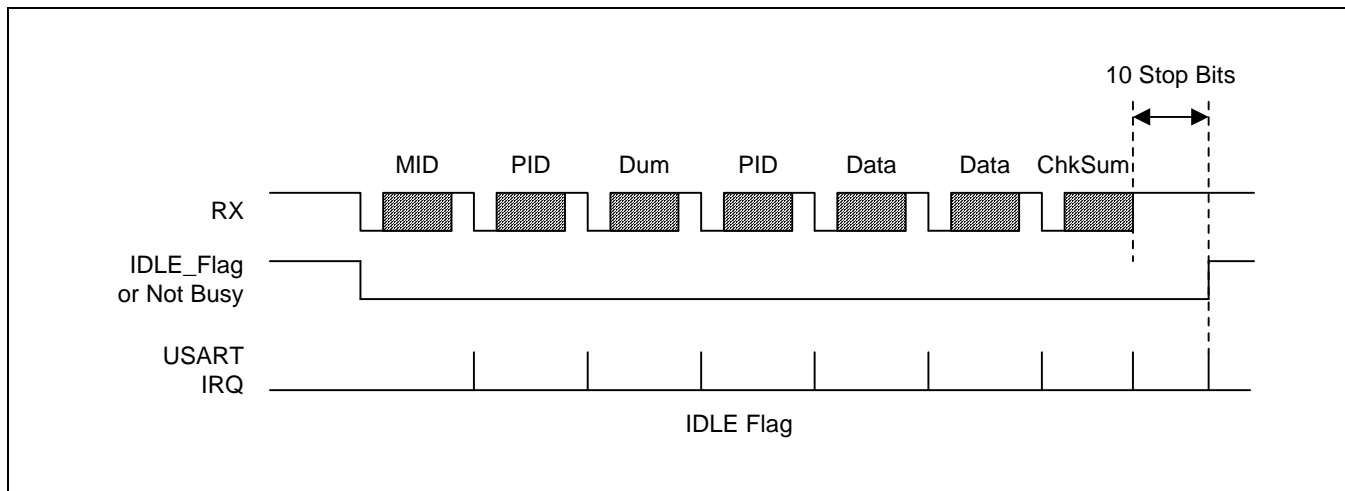


Figure 26-6. IDLE Flag

3.2.8 Time-Out

This function allows an idle condition on the RXD line to be detected. The maximum delay for which the USART should wait for a new character to arrive while the RXD line is inactive (high level) is programmed in TO[15:0] of US_RTOR (Receiver Time-out). When this register is set to 0, no time-out is detected.

Otherwise, the receiver waits for a first character and then initializes a counter which is decremented at each bit period and reloaded at each byte reception. When the counter reaches 0, the TIMEOUT bit in US_SR is set. The user starts (or restarts) the wait for a first character by setting the STTTO (Start Time-Out) bit in US_CR register.

In other words, to start a time-out, the following conditions must be met :

- US_RTOR must not be equal to 0
- The time-out must be started by setting to a logical 1 the STTTO (start time-out) in the US_CR register
- One character must be received

Calculation of time-out duration :

Duration = Value x Bit period in asynchronous mode

Duration = Value x 16 x period in synchronous mode

3.3 TRANSMITTER

3.3.1 General Description

The transmitter has the same behavior in both synchronous and asynchronous operating modes. Start bit, data bits, parity bit and stop bits are serially shifted, least significant bit first.

The number of data bits is selected in the CHRL[1:0] field in US_MR.

The parity bit is set according to the PAR[2:0] field in US_MR register. If the parity type is even then the parity bit depends on the one bit sum of all data bits. For odd parity, the parity bit is the inverted sum of all data bits.

The number of stop bits is selected in the NBSTOP[1:0] field in US_MR.

When a character is written to US_THR (Transmit Holding), it is transferred to the Shift Register as soon as it is empty.

When the transfer occurs, the TXRDY bit in US_SR is set until a new character is written to US_THR. If Transmit Shift Register and US_THR are both empty, the TXEMPTY bit in US_SR is set (after the last stop bit of the last transfer).

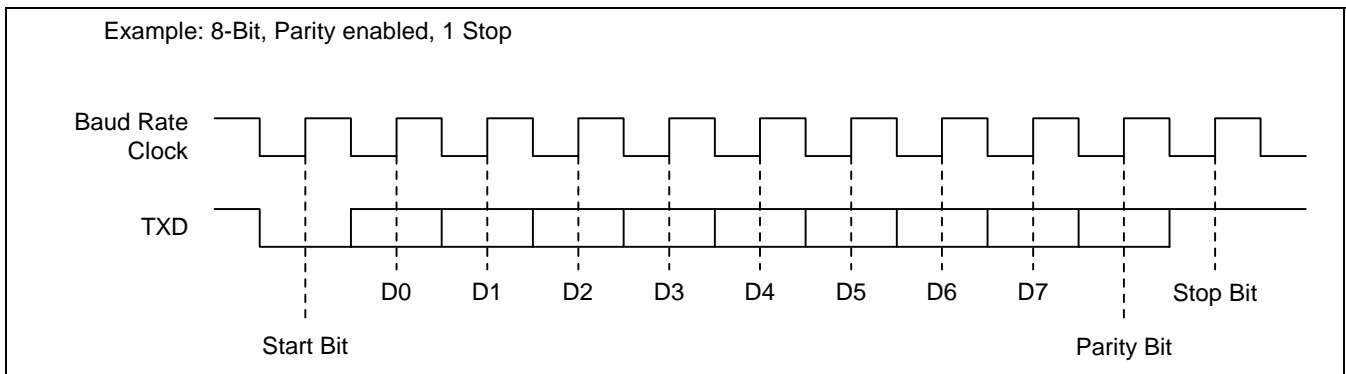


Figure 26-7. Synchronous and Asynchronous Modes, Character Transmission

3.3.2 Time-Guard

The Time-guard function allows the transmitter to insert an idle state on the TXD line between 2 characters. The duration of the idle state is programmed in US_TTGR (Transmitter Time-Guard). When this register is set to zero, no time-guard is generated. Otherwise, the transmitter holds a high level on TXD after each transmitted byte during the number of bit periods programmed in US_TTGR.

3.3.3 Multi-Drop Mode

When the field PAR field in US_MR equals 11Xb, the USART is configured to run in multi-drop mode for automatic address/data detection and the PARE (parity error bit in US_SR register) is used to identify a data byte (parity bit is detected low) or an address byte (parity bit is high). So, in this mode, the parity error bit (PARE in US_SR) is set when data is identified as an address byte. The PARE status bit is cleared with the PARE bit in the US_CSR (Clear Status Register). If the parity bit is detected low, identifying a data byte, PARE is not set.

The transmitter sends an address byte (parity bit set) when a Send Address Command (SENDA) is written to US_CR.

In this case, the next byte written to US_THR will be transmitted as an address. After this any byte transmitted will have the parity bit cleared.

3.4 BREAK

3.4.1 Transmit Break

The transmitter can generate a break condition on the TXD line when the STTBRK(Start break) command is set in US_CR (Control Register). In this case, the characters present in the Transmit Shift Register are completed before the line is held low.

To remove this break condition on the TXD line, the STPBRK(Stop break) command in US_CR must be set. The USART generates a minimum break duration of one character length.

The TXD line then returns to high level (idle state) for at least 12 bit periods to ensure that the end of break is correctly detected. Then the transmitter resumes normal operation.

3.4.2 Receive Break

The break condition is detected by the receiver when all data, parity and stop bits are low. At the moment of the low stop bit detection, the receiver asserts the RXBRK(Break received) bit in US_SR.

End of receive break is detected by a high level for at least 2/16 of the bit period in asynchronous operating mode or at least one sample in synchronous operating mode. RXBRK is also set after end of break has been detected.

3.5 INTERRUPTS

Most of status bit in US_SR has a corresponding bit in US_IER (Interrupt Enable) and US_IDR (Interrupt Disable) which controls the generation of interrupts by asserting the USART interrupt line connected to the GIC. US_IMR (Interrupt Mask Register) indicates the status of the corresponding bits.

When a bit is set in US_SR and the same bit is set in US_IMR, the interrupt line is asserted.

3.6 TEST MODES

The USART can be programmed to operate in 3 different test modes, using the field CHMODE[1:0] in US_MR.

Automatic echo mode allows bit by bit retransmission.

When a bit is received on the RXD line, it is sent to the TXD line. Programming the transmitter has no effect.

Local loopback mode allows the transmitted characters to be received. TXD and RXD pins are not used and the output of the transmitter is internally connected to the input of the receiver. The RXD pin level has no effect and the TXD pin is held high, as in idle state.

Remote loopback mode directly connects the RXD pin to the TXD pin. The Transmitter and the Receiver are disabled and have no effect. This mode allows bit by bit retransmission.

3.7 LIN PROTOCOL (COMPLIANT WITH LIN1.2 AND LIN2.0 RELEASES)

This section describes the LIN protocol supported by the USART.

The USART works as a "Master Control Unit" (as it is defined on the LIN Protocol Specification) when the LIN bit is set in the US_MR register.

The LIN2_0 bit in the US_MR Mode Register selects the LIN protocol release. By default, USART supports the LIN 1.2 protocol. The software user can not modified this bit during a Message Transfer.

It generates the "HEADER FRAME" automatically if the STHEADER is set in the US_CR Controller Register and the end of header is signaled through the ENDHEADER bit on the Status Register. The header content concerning the "IDENTIFIER" is defined by the Identifier Register (US_LIR on Read/Write access). The parity is automatically calculated. In the header part, the SYNC_BREAK length is configurable through the Sync Break Length Register (US_SBLR) (the SYNC_BREAK could be set from 8Tbit up to 31Tbit).

Setting the STRESP bit in the US_CR register has the effect of sending a "RESPONSE MESSAGE" (see the LIN Protocol Specification). The data to be transmitted are defined in the US_DFWR[1:0] register in the LIN1.2 release, the number of data bytes to be transmitted depends on the IDENTIFIER[5:4] value configured in the US_LIR register. In the LIN2.0 release, the number of data bytes to be transmitted depends on the NDATA[2:0] value configured in US_LIR register. The CHECKSUM field is automatically calculated and sent (For the LIN1.2 release, the checksum is classic. For the LIN2.0 release, the checksum can be enhanced or classic, it depends on the CHK_SEL bit configured in the US_LIR register).

It is also possible to fill the message response to the THR register. In this case, user has to wait the END HEADER interrupt before starting to fill the THR register(LDMAC can be used the data bytes.)

The "RESPONSE MESSAGE" received by the USART is available in the US_DFRR[1:0] register even if the USAT has field the "RESPONSE MESSAGE" in the US_DFWR[1:0] register.

The USART is able to detect and to signal the end of "LIN MESSAGE" through the ENDMESS bit on the Status Register. Moreover the USART detects the Bit-Error, the Identifier-Parity-Error, Not-Responding-Error, the Checksum-Error and the Wakeup.

Write accesses in the US_LIR and US_DFWR[1:0] registers, LIN2_0 bit in the US_MR register, STHEADER in the US_CR register and SYNC_BRK[4:0] field in the US_SBLR register have no effect during a message transfer.

It is also advised not to write on the Transmitter Holding Register (US_THR) during the Header.

The inter-byte space is configurable through the US_TTGR register (transmit time guard), the default inter-byte space is one Tbit.

3.7.1 USART configuration for LIN

To work in LIN mode, the USART has to be set in normal mode (see the MODE register) with the transmitter and its receiver enabled.

3.7.2 Message Characteristics

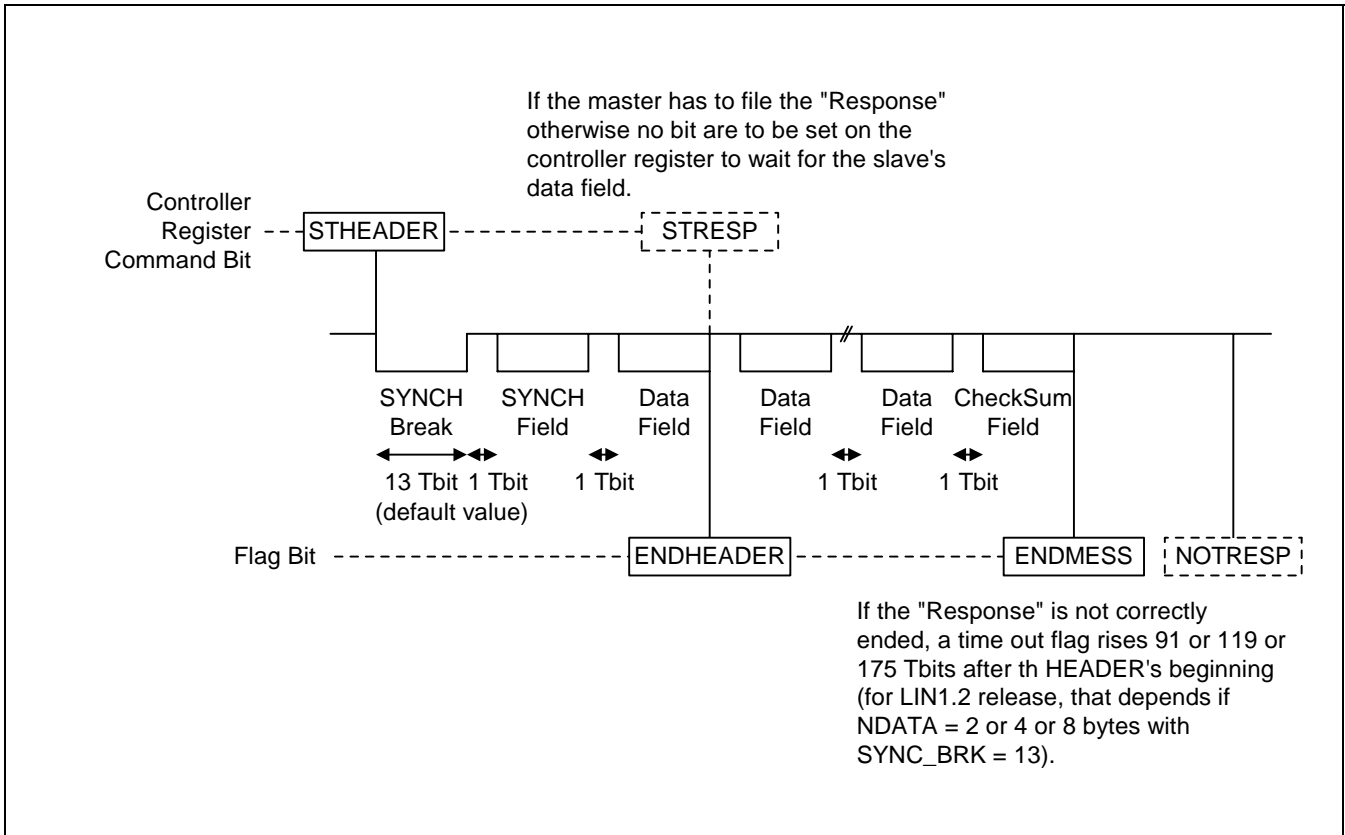


Figure 26-8. Message Characteristics

3.7.3 Wake-up

The USART can wake-up from the sleep mode by sending a Wake-up signal. For the LIN1.2 protocol, this signal consists of the character 0xC0, 0x80 or 0x00. Theses characters will be detected by the master as a valid data byte. For the LIN2.0 protocol, the wake-up request is issued by forcing the line to the dominant state for at least 250us. This will asserts the WAKEUP bit in the US_SR Register.

3.8 SMART-CARD PROTOCOL

The USARTs are ISO7816-3 compliant and allow character repetition or error signaling on parity errors.

The following lines describes the functions which are available if the SMCARDPT bit is set on the US_MR register.

The USART smart card protocol requires that both the transmitter and the receiver are enabled.

If PIO block allows it, TXD can be configured as an open drain output and connected to the RXD pin with an additional external pull-up resistor resulting in the smart card COMMS line.

3.8.1 Character Transmission to Smart Card

The USART is able to detect that the smart card has not received correctly the last transmitted byte by checking the parity error signal generated by the smart card.

When the card generates the error signal, the last transmitted byte is re-transmitted again as many times as determined in the SENDTIME[2:0] bits of the US_MR register until the error signal is no longer generated by the smart card.

When the error signal is detected by the USART, the FRAME error flag is set in the US_SR register.

The error signal is checked by the USART at $t_0 + 11$ bits where t_0 is the falling edge of the start bit (i.e. between the 2 stop bits).

In the following example, a parity error is detected by the smart card. The smart card generates the error signal on the COMMS line. The error signal is detected by the USART which re-transmit the last character.

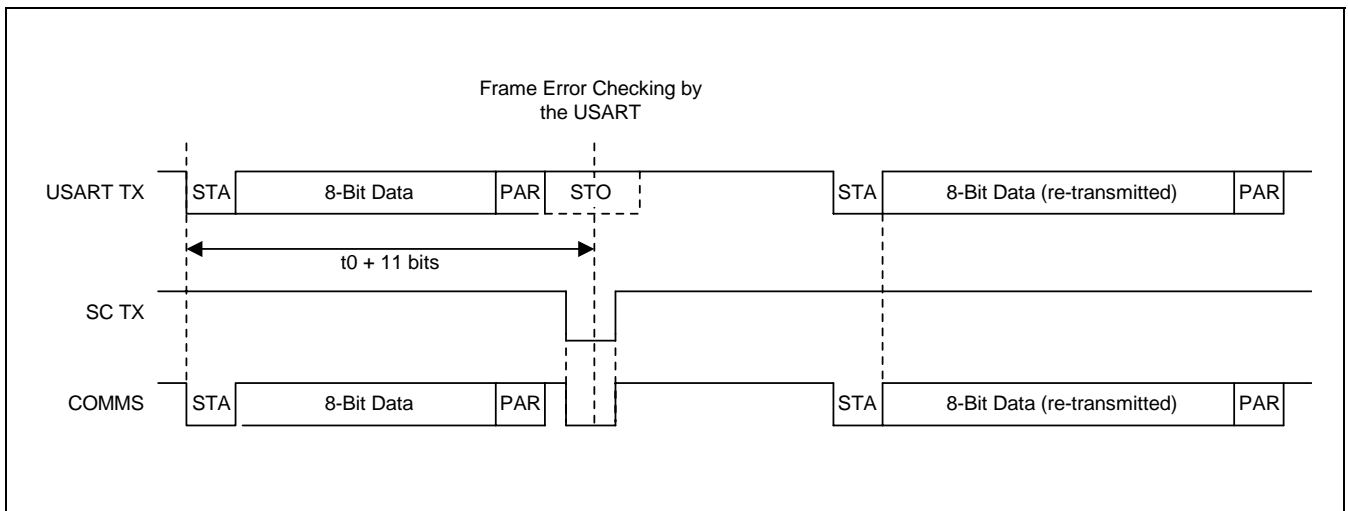


Figure 26-9. Smart-Card Transmission Error

If a LDMA transfer is used to send data byte to the card, the LDMA counter will not be decremented and the LDMA memory pointer will not be incremented until the card has received a correct byte or the maximum repetition time has been reached.

3.8.2 Character Reception from Smart Card

The USART is able to generate the error signal (see ISO7816-3 protocol) when last byte received has a parity error.

When a parity error is detected by the USART, the USART transmission line is driven low for 1.0625 bit period starting at $t_0 + 10.625 + [0:0.0625]$ (i.e. during the 2 stop bits) to indicate to the smart card that a bad reception has occurred on the USART. With T=0 protocol type smart cards, the smart card must re-send the last character.

In that case, the USART signals only a parity error by setting the PARE bit in the US_SR register.

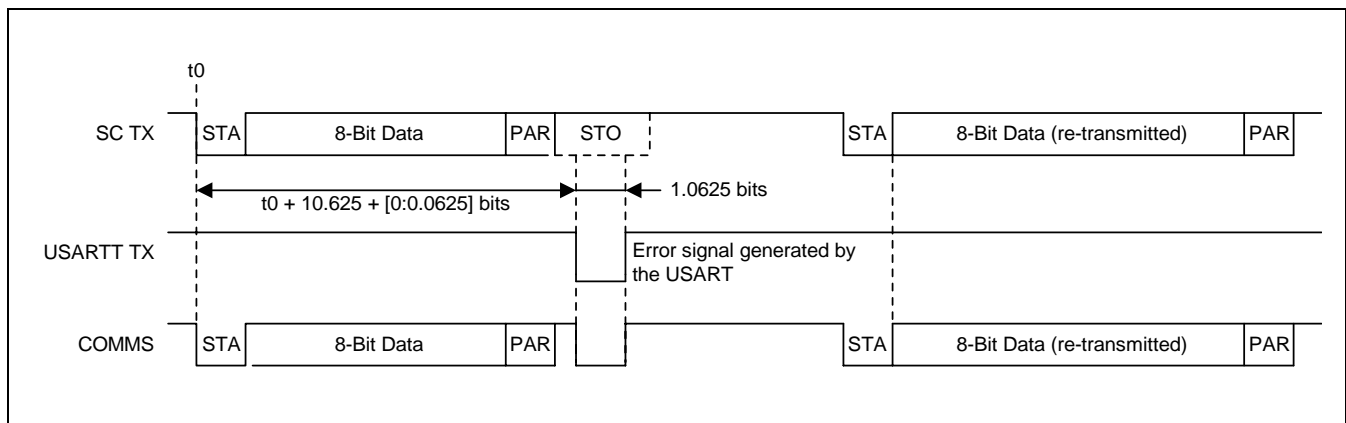


Figure 26-10. Error Signaling on Reception

If a LDMA transfer is used to receive data byte from the card, the LDMA counter will be decremented and the LDMA memory pointer will be incremented even when a parity error is detected. The user must reconfigure the LDMA memory pointer (decrement by 1) and counter (increment by 1) in order to receive all the remaining bytes.

3.8.3 USART Configuration in Smart Card Mode

To work in Smart Card mode, the USART has to be set in normal mode and the number of stop bits must be programmed at 2.(see the MODE register).

3.9 PROGRAMMING EXAMPLES

Example of use of the USART with LIN functionality

Send a Header Frame and then send the Message Frame with an identification of 0x30 with a message of 2x16bits at 19600 bps using interrupt.

Configuration:

Enable the clock on USART by writing bit USART in US_ECR.

- Do a software reset of the USART peripheral to be in a known state by writing bit SWRST in US_CR.
- Configuration of US_BRGR : For 19600bps CD field in the US_BRGR= $PCLK/(16 \times 19600)$ (take nearest integer value)
- Enable Transmitter and Receiver bit RXEN and TXEN in the US_CR register
- Configuration of US_MR register : normal mode, no parity, 8 data bits, 1 stop bit, normal mode, LIN supported, use PCLK clock for USART baud rate generation
- Configuration of US_SBLR register : Set the syncbreak to 0x13
- Enter the data to send in DRWR0 and DFWR1
- Configuration of US_IER register : generate an interrupt at the end of the header and message transmission. This is done by setting the ENDHEADER, ENDMESS in the US_IMR register. Associate interrupt to error like BIT error, Parity error, checksum error ... GIC must be configured.
- Configure the identifier by writing 0x30 in the US_LIR register
- Enable the LIN header transmission by writing bit STHEADER in the US_CR register, this will send the header. Before going to the next step, software should be informed by the interrupt that the header has been sent.
- Enable the LIN response transmission by setting the bit STRESP in the US_CR register, this will send the message response. An interrupt is generated at the end of the transmission.

Interruption Handling :

- IRQ Entry and call C function.
- Read the US_SR and verify the source of the interrupt. This register is read and clear for some status field. Care should be taken to keep status information to be able to proceed with all cases. Bits present in the US_CSR are not read and clear and then should be cleared in the next step.
- Clear the corresponding interrupt at peripheral level by writing in the US_CSR register.
- Interrupt treatment : inform background software that header or message has been transmitted.
- IRQ Exit.

4. REGISTERS DESCRIPTION

Base Addresses –	UART0	: 0xFFE28000
	UART1	: 0xFFE34000
	USART0	: 0xFFE38000

Table 26-4. USART Special Function Registers

Offset Address	Name	Description	R/W	Reset State
0x000 – 0x04C	–	Reserved	–	–
0x050	US_ECR	Enable clock register	W	–
0x054	US_DCR	Disable clock register	W	–
0x058	US_PMSR	Power management status register	R	0XXXXXXXX0
0x05C	–	Reserved	–	–
0x060	US_CR	Control register	W	0x00000000
0x064	US_MR	Mode register	R/W	0x00000000
0x068	–	Reserved	–	–
0x06C	US_CSR	Clear status register	W	0x00000000
0x070	US_SR	Status register	R	0x00000800
0x074	US_IER	Interrupt enable register	W	–
0x078	US_IDR	Interrupt disable register	W	–
0x07C	US_IMR	Interrupt mask register	R	0x00000000
0x080	US_RHR	Receiver holding register	R	0x00000000
0x084	US_THR	Transmitter holding register	W	–
0x088	US_BRGR	Baud rate generator register	R/W	0x00000000
0x08C	US_RTOR	Receiver time-out register	R/W	0x00000000
0x090	US_TTGR	Transmitter time-guard register	R/W	0x00000000
0x094	US_LIR	LIN Identifier register	R/W	0x3AD40000
0x098	US_DFWR0	Data field write 0 register	R/W	0x00000000
0x09C	US_DFWR1	Data field write 1 register	R/W	0x00000000
0x0A0	US_DFRR0	Data field read 0 register	R	0x00000000
0x0A4	US_DFRR1	Data field read 1 register	R	0x00000000
0x0A8	US_SBLR	Synchronous break length register	R/W	0x0000000D

NOTE: The reset value of US_PMSR depends on the version used in the final chip.

USART Enable Clock Register

US_ECR (0x050)

Access: Write only

31	30	29	28	27	26	25	24
DBGEN	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	USART	–
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **USART : USART Clock enable**

0: No effect

1: Enable USART clock

- **DBGEN : Debug mode enable**

0: No effect

1: Enable debug mode

USART Disable Clock Register

US_DCR (0x054)

Access: Write only

31	30	29	28	27	26	25	24
DBGEN	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	USART	–
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **USART : USART Clock disable**

0: No effect

1: Disable USART clock

- **DBGEN : Debug mode disable**

0: No effect

1: Disable debug mode

USART Power Management Status Register US_PMSR (0x058) Access: Read only

31	30	29	28	27	26	25	24
DBGEN	–	IPIDCODE[25:20]					
R-0	R-0	R-U	R-U	R-U	R-U	R-U	R-U
23	22	21	20	19	18	17	16
IPIDCODE[19:12]							
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U
15	14	13	12	11	10	9	8
IPIDCODE[11:4]							
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U
7	6	5	4	3	2	1	0
IPIDCODE[3:0]				–	–	USART	–
R-U	R-U	R-U	R-U	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset***26-1. IMPORTANT NOTICE**

The US_PMSR register is not reset by software reset.

- **USART : USART Clock Status**

0: The USART clock is disabled.

1: The USART clock is enabled.

- **IPIDCODE[25:0] : IP identifier code**

The field contains the version number of the module, coded on 26 bits.

- **DBGEN : Debug mode**

0: The debug acknowledge generated by the ICE interface (dbgack_sclk input signal) has no influence on the USART function.

1: The debug acknowledge freezes the USART function when the ICE interface is activated (High level on input pin). However full read/write access to internal register is kept for the debug purpose.

USART Control Register **US_CR (0x060)** **Access: Write only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	STREPS	STHEADER
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	SENDA	STTTO	STPBRK	STTBRK	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
TXDIS	TXEN	RXDIS	RXEN	RSTTX	RSTRX	–	SWRST
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **SWRST : Software reset**

0: No reset.

1: Reset the USART.

A software triggered hardware reset of the USART is performed. It resets all the registers (except US_PMSR) only if USART clock is enabled by US_ECR.

- **RSTRX : Reset receiver**

0: No effect.

1: The receiver logic is reset.

- **RSTTX : Reset transmitter**

0: No effect.

1: The transmitter logic is reset.

- **RXEN : Receiver enable**

0: No effect.

1: The receiver is enabled if RXDIS is 0.

- **RXDIS : Receiver disable**

0: No effect.

1: The receiver is disabled.

- **TXEN : Transmitter enable**

0: No effect.

1: The transmitter is enabled if TXDIS is 0.

- **TXDIS : Transmitter disable**

0: No effect.

1: The transmitter is disabled.

- **STTBRK : Start break**

0: No effect.

1: If break is not being transmitted, start transmission of a break after the characters present in the Transmit Shift Register have been transmitted.

- **STPBRK : Stop break**

0: No effect.

1: If a break is being transmitted, stop transmission of the break after a minimum of one character length and transmit a high level during 12 bit periods.

- **STTT0 : Start time-out**

0: No effect.

1: Start waiting for a character before clocking the time-out counter.

- **SENDA : Send address**

0: No effect.

1: In Multi-drop Mode only, the next character written to the US_THR is sent with the address bit set.

- **STHEADER : Start header**

0: No effect.

1: If the LIN bit is set on the Mode Register, send the LIN 's Header Frame.

- **STRESP : Start response**

0: No effect

1: Send a part or the Data Field 0 Register content and Data Field 1 Register content.

USART Mode Register **US_MR (0x064)** **Access: Read/Write**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
–	–	–	–	LIN2_0	CLKO	MODE9	SMCARDPT
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
CHMODE[1:0]		NBSTOP[1:0]		PAR[2:0]			SYNC
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
CHRL[1:0]		CLKS[1:0]		SENDDTIME[2:0]			LIN
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

26-2. IMPORTANT NOTICE

It is not possible to set the LIN2_0 bit during a Message Transfer.

- **LIN : Local Interconnect Network mode**

- 0: Disable LIN protocol on USART
- 1: Enable LIN protocol on USART

- **SENDDTIME[2:0] :**

Indicates the maximum number of repetitions a character has to be transmitted by the USART when configured in smart card protocol.

Table 26-5. SENDDTIME Configuration Field

SENDDTIME[2:0]			Time number
0	0	0	0
0	0	1	1
–			
1	1	1	7

- **CLKS[1:0] : Clock selection (baud rate generator input clock)**

Table 26-6. CLKS Clock Selection Field

CLKS[1:0]		Selected clock
0	0	PCLK
0	1	PCLK/8
1	x	External clock (USARTCLK0)

- **CHRL[1:0] : Character length**

Start, stop and parity bits are added to the character length.

Table 26-7. Character Length Field

CHRL[1:0]		Character length
0	0	5 bits
0	1	6 bits
1	0	7 bits
1	1	8 bits

- **SYNC : Synchronous mode select**

0: USART operates in Asynchronous Mode.

1: USART operates in Synchronous Mode.

- **PAR[2:0] : Parity type**

Table 26-8. Parity Type Field

PAR[2:0]			Parity type
0	0	0	Even Parity
0	0	1	Odd Parity
0	1	0	Parity forced to 0 (Space)
0	1	1	Parity forced to 1 (Mark)
1	0	X	No parity
1	1	X	Multi-drop mode

NOTE: For LIN, PAR[2:0] must be set to '10X'.

- **NBSTOP[1:0] : Number of stop bits**

The interpretation of the number of stop bits depends on SYNC.

Table 26-9. NBSTOP Configuration Field

NBSTOP[1:0]		Asynchronous (SYNC = 0)	Synchronous (SYNC = 1)
0	0	1 stop bit	1 stop bit
0	1	1.5 stop bits	Reserved
1	0	2 stop bits	2 stop bits
1	1	Reserved	Reserved

- **CHMODE[1:0] : Channel mode**

Table 26-10. Channel Mode Field

CHMODE[1:0]		Mode Description
0	0	Normal Mode The USART channel operates as a Rx/Tx USART
0	1	Automatic Echo Receiver data input is connected to TXD pin
1	0	Local Loopback Transmitter output signal is connected to receiver input signal
1	1	Remote Loopback RXD pin is internally connected to TXD pin

- **SMCARDPT : Smart card protocol**

0: Disable smart card protocol on USART

1: Enable LIN protocol on USART

- **MODE9 : 9-bit character length**

0: The CHRL field defines the character length.

1: 9-Bit character length.

- **CLKO : Clock output select**

0: The USART does not drive the SCK pin.

1: The USART drives the SCK pin if CLKS[1] is 0.

- **LIN2_0 : Select the LIN protocol release**

This bit is significant when the LIN bit is set. The software user can not changed this value during a Message Transfer (i.e. LIN is busy).

0: USART supports the LIN1.2 protocol.

1: USART supports the LIN2.0 protocol.

USART Clear Status Register

US_CSR (0x06C)

Access: Write only

31	30	29	28	27	26	25	24
–	WAKEUP	CHECKSUM	IPERROR	BITERROR	NOTRESP	ENDMESS	ENDHEADER
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	IDLE	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	ENDTX	ENDRX	RXBRK	–	–
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **RXBRK : Clear break received**

0: No effect.

1: Clears RXBRK interrupt.

- **ENDRX : Clear end of reception**

0: No effect.

1: Clears ENDRX interrupt.

- **ENDTX : Clear end of transmission**

0: No effect.

1: Clears ENDTX interrupt.

- **OVRE : Clear overrun error**

0: No effect.

1: Clears OVRE interrupt.

- **FRAME : Clear framing error**

0: No effect.

1: Clears FRAME interrupt.

- **PARE : Clear parity error**

0: No effect.

1: Clears PARE interrupt.

- **IDLE : Clear idle interrupt**

0: No effect.

1: Clears IDLE interrupt.

- **ENDHEADER : Clear end of Header**

0: No effect.

1: Clears ENDHEADER interrupt.

- **ENDMESS : Clear end of Message**

0: No effect.

1: Clears ENDMESS interrupt.

- **NOTRESP : Clear not responding error**

0: No effect.

1: Clears NOTRESP interrupt.

- **BITERROR : Clear bit error**

0: No effect.

1: Clears BITERROR interrupt.

- **IPERROR : Clear identity parity error**

0: No effect.

1: Clears IPERROR interrupt.

- **CHECKSUM : Clear check sum**

0: No effect.

1: Clears CHECKSUM interrupt.

- **WAKEUP : Clear wake up**

0: No effect.

1: Clears WAKEUP interrupt.

USART Status Register

US_SR (0x070)

Access: Read only

31	30	29	28	27	26	25	24
–	WAKEUP	CHECKSUM	IPERROR	BITERROR	NOTRESP	ENDMESS	ENDHEADER
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	IDLEFLAG	IDLE	TXEMPTY	TIMEOUT
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	RXBRK	TXRDY	RXRDY
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **RXRDY : Receiver ready**

0: No complete character has been received since the last read of the US_RHR or the receiver is disabled.

1: At least one complete character has been received and the US_RHR has not yet been read.

- **TXRDY : Transmitter ready**

0: A character is in US_THR waiting to be transferred to the Transmit Shift Register or the transmitter is disabled.

1: There is no character in the US_THR.

Equal to zero when the USART is disabled or at reset. Transmitter Enable command (in US_CR) sets this bit to one.

- **RXBRK : Break received/end**

0: No Break Received and no End of Break has been detected since the last “Reset Status Bits” command in the Control Register.

1: Break Received or End of Break has been detected since the last “Reset Status Bits” command in the Control Register.

- **OVRE : Overrun error**

- 0: No byte has been transferred from the Receive Shift Register to the US_RHR when RxRDY was asserted since the last "Reset Status Bits" command.
- 1: At least one byte has been transferred from the Receive Shift Register to the US_RHR when RxRDY was asserted since the last "Reset Status Bits" command.

- **FRAME : Framing error**

- 0: No stop bit has been detected low since the last "Reset Status Bits" command.
- 1: At least one stop bit has been detected low since the last "Reset Status Bits" command.

- **PARE : Parity error**

- 0: No parity bit has been detected false (or a parity bit high in multi-drop mode) since the last "Reset Status Bits" command.
- 1: At least one parity bit has been detected false (or a parity bit high in multi-drop mode) since the last "Reset Status Bits" command.

- **TIMEOUT : Receiver time-out**

- 0: There has not been a time-out since the last "Start Time-out" command or the Time-out Register is 0.
- 1: There has been a time-out since the last "Start Time-out" command.

- **TXEMPTY : Transmitter empty**

- 0: There are characters in either US_THR or the Transmit Shift Register.
- 1: There are no characters in both US_THR and the Transmit Shift Register.
Equal to zero when the USART is disabled or after reset. Transmitter Enable command (in US_CR) sets this bit to one.

- **IDLE : Idle interrupt**

- 0: No end of J1587 protocol frame.
- 1: An end of J1587 protocol frame occurred.

- **IDLEFLAG : Idle flag**

- 0: A frame is being received by the USART.
- 1: No frame is being received by the USART.

This bit indicates a frame transmission in J1587 protocol. It's turn low when a reception start and turn high when a reception is followed by at least 10 stop bits (10 bits at high level).

- **ENDHEADER : End of Header**

0: No end of Header has occurred on a LIN Frame.

1: An end of Header has occurred on a LIN Frame.

- **ENDMESS : End of Message**

0: No end of Message occurred on a LIN Frame.

1: An end of Message has occurred on a LIN Frame.

- **NOTREPS : Not Responding**

0: No Slave-not-responding-error has been detected LIN Frame.

1: A Slave-not-responding-error has been detected LIN Frame.

- **BITERROR : Bit Error**

0: No Bit-error has been detected on a LIN Frame.

1: A Bit-error has been detected on a LIN Frame.

- **IPERROR : Identity Parity Error**

0: No Identity-Parity-Error has been detected on a LIN Frame.

1: A Identity-Parity-Error has been detected on a LIN Frame.

- **CHECKSUM : Check Sum**

0: No Checksum-error has been detected LIN Frame.

1: A Checksum-error has been detected LIN Frame.

- **WAKEUP : Wake Up**

0: No Wakeup has been detected.

1: A Wakeup has been detected

USART Interrupt Enable Register

US_IER (0x074)

Access: Write only

31	30	29	28	27	26	25	24
–	WAKEUP	CHECKSUM	IPERROR	BITERROR	NOTRESP	ENDMESS	ENDHEADER
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	IDLE	TXEMPTY	TIMEOUT
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	RXBRK	TXRDY	RXRDY
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **RXRDY : Receiver ready interrupt enable**

0: No effect

1: Enable the RXRDY interrupt

- **TXRDY : Transmitter ready interrupt enable**

0: No effect

1: Enable the TXRDY interrupt

- **RXBRK : Receiver break interrupt enable**

0: No effect

1: Enable the RXBRK interrupt

- **OVRE : Overrun error interrupt enable**

0: No effect

1: Enable the OVRE interrupt

- **FRAME : Framing error interrupt enable**

0: No effect

1: Enable the FRAME interrupt

- **PARE : Parity error interrupt enable**

0: No effect

1: Enable the PARE interrupt

- **TIMEOUT : Time-out interrupt enable**
 - 0: No effect
 - 1: Enable the TIMEOUT interrupt

- **TXEMPTY : Transmitter empty interrupt enable**
 - 0: No effect
 - 1: Enable the TXEMPTY interrupt

- **IDLE : IDLE interrupt enable**
 - 0: No effect
 - 1: Enable the IDLE interrupt

- **ENDHEADER : Ended header interrupt enable**
 - 0: No effect
 - 1: Enable the ENDHEADER interrupt

- **ENDMESS : Ended message interrupt enable**
 - 0: No effect
 - 1: Enable the ENDMESS interrupt

- **NOTREPS : Not responding error interrupt enable**
 - 0: No effect
 - 1: Enable the NOTREPS interrupt

- **BITERROR : Bit error interrupt enable**
 - 0: No effect
 - 1: Enable the BITERROR interrupt

- **IPERROR : Identity parity error interrupt enable**
 - 0: No effect
 - 1: Enable the IPERROR interrupt

- **CHECKSUM : Checksum error interrupt enable**
 - 0: No effect
 - 1: Enable the CHECKSUM interrupt

- **WAKEUP : Wake up interrupt enable**
 - 0: No effect
 - 1: Enable the WAKEUP interrupt

USART Interrupt Disable Register

US_IDR (0x078)

Access: Write only

31	30	29	28	27	26	25	24
–	WAKEUP	CHECKSUM	IPERROR	BITERROR	NOTRESP	ENDMESS	ENDHEADER
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	IDLE	TXEMPTY	TIMEOUT
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	RXBRK	TXRDY	RXRDY
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **RXRDY : Receiver ready interrupt disable**

0: No effect

1: Disable the RXRDY interrupt

- **TXRDY : Transmitter ready interrupt disable**

0: No effect

1: Disable the TXRDY interrupt

- **RXBRK : Receiver break interrupt disable**

0: No effect

1: Disable the RXBRK interrupt

- **OVRE : Overrun error interrupt disable**

0: No effect

1: Disable the OVRE interrupt

- **FRAME : Framing error interrupt disable**

0: No effect

1: Disable the FRAME interrupt

- **PARE : Parity error interrupt disable**

0: No effect

1: Disable the PARE interrupt

- **TIMEOUT : Time-out interrupt disable**
 - 0: No effect
 - 1: Disable the TIMEOUT interrupt

- **TXEMPTY : Transmitter empty interrupt disable**
 - 0: No effect
 - 1: Disable the TXEMPTY interrupt

- **IDLE : IDLE interrupt disable**
 - 0: No effect
 - 1: Disable the IDLE interrupt

- **ENDHEADER : Ended header interrupt disable**
 - 0: No effect
 - 1: Disable the ENDHEADER interrupt

- **ENDMESS : Ended message interrupt disable**
 - 0: No effect
 - 1: Disable the ENDMESS interrupt

- **NOTREPS : Not responding error interrupt disable**
 - 0: No effect
 - 1: Disable the NOTREPS interrupt

- **BITERROR : Bit error interrupt disable**
 - 0: No effect
 - 1: Disable the BITERROR interrupt

- **IPERROR : Identity parity error interrupt disable**
 - 0: No effect
 - 1: Disable the IPERROR interrupt

- **CHECKSUM : Checksum error interrupt disable**
 - 0: No effect
 - 1: Disable the CHECKSUM interrupt

- **WAKEUP : Wake up interrupt disable**
 - 0: No effect
 - 1: Disable the WAKEUP interrupt

USART Interrupt Mask Register

US_IMR (0x07C)

Access: Read only

31	30	29	28	27	26	25	24
–	WAKEUP	CHECKSUM	IPERROR	BITERROR	NOTRESP	ENDMESS	ENDHEADER
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	IDLE	TXEMPTY	TIMEOUT
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
PARE	FRAME	OVRE	–	–	RXBRK	TXRDY	RXRDY
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

W: Write R: Read -0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **RXRDY : Receiver ready interrupt mask**

0: The RXRDY interrupt is disabled.

1: The RXRDY interrupt is enabled.

- **TXRDY : Transmitter ready interrupt mask**

0: The TXRDY interrupt is disabled.

1: The TXRDY interrupt is enabled.

- **RXBRK : Receiver break interrupt mask**

0: The RXBRK interrupt is disabled.

1: The RXBRK interrupt is enabled.

- **OVRE : Overrun error interrupt mask**

0: The OVRE interrupt is disabled.

1: The OVRE interrupt is enabled.

- **FRAME : Framing error interrupt mask**

0: The FRAME interrupt is disabled.

1: The FRAME interrupt is enabled.

- **PARE : Parity error interrupt mask**

0: The PARE interrupt is disabled.

1: The PARE interrupt is enabled.

- **TIMEOUT : Time-out interrupt mask**
 - 0: The TIMEOUT interrupt is disabled.
 - 1: The TIMEOUT interrupt is enabled.

- **TXEMPTY : Transmitter empty interrupt mask**
 - 0: The TXEMPTY interrupt is disabled.
 - 1: The TXEMPTY interrupt is enabled.

- **IDLE : IDLE interrupt mask**
 - 0: The IDLE interrupt is disabled.
 - 1: The IDLE interrupt is enabled.

- **ENDHEADER : Ended header interrupt mask**
 - 0: The ENDHEADER interrupt is disabled.
 - 1: The ENDHEADER interrupt is enabled.

- **ENDMESS : Ended message interrupt mask**
 - 0: The ENDMESS interrupt is disabled.
 - 1: The ENDMESS interrupt is enabled.

- **NOTREPS : Not responding error interrupt mask**
 - 0: The NOTRESP interrupt is disabled.
 - 1: The NOTRESP interrupt is enabled.

- **BITERROR : Bit error interrupt mask**
 - 0: The BITERROR interrupt is disabled.
 - 1: The BITERROR interrupt is enabled.

- **IPERROR : Identity parity error interrupt mask**
 - 0: The IPERROR interrupt is disabled.
 - 1: The IPERROR interrupt is enabled.

- **CHECKSUM : Checksum error interrupt mask**
 - 0: The CHECKSUM interrupt is disabled.
 - 1: The CHECKSUM interrupt is enabled.

- **WAKEUP : Wake up interrupt mask**
 - 0: The WAKEUP interrupt is disabled.
 - 1: The WAKEUP interrupt is enabled.

USART Receiver Holding Register

US_RHR (0x080)

Access: Read only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	RXCHR[8]
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
RXCHR[7:0]							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset***26-3. IMPORTANT NOTICE**

When reading this register, the RXRDY bit is clear in the US_RHR register.

During the debug mode, users should use the ghost registers to avoid clearing RXRDY bit..

- RXCHR[8:0] : Received character**

Last character received if RXRDY is set. The value is maintained in the register until a new byte is received. When number of data bits is less than 9 bits, the bits are right aligned.

USART Transmit Holding Register

US_THR (0x084)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	TXCHR[8]
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
TXCHR[7:0]							
W	W	W	W	W	W	W	W

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **TXCHR[8:0] : Character to be transmitted**

If TXRDY is at a logical 1, this will be the next character to be transmitted (after current one if already a character is present in the transmit shift register). If TXRDY is at a logical 0, the current character in the US_THR register will be overwritten. When number of data bits is less than 9 bits, the bits are right aligned.

USART Baud Rate Generator Register **US_BRGR (0x088)** **Access: Read/Write**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
CD[15:8]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
CD[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

26-4. IMPORTANT NOTICE

In the synchronous mode, the programmed value must be even to ensure a 50:50 mark/space ratio and the software user must enable the clock (i.e. CD is different to zero) after configuring the Baud Rate clock via the US_MR register.

CD = 1 must not be used when internal clock (PCLK) is selected (i.e USCLKS[1:0] = 0).

- **CD[15:0] : Clock Divider**

This register has no effect if the synchronous mode is selected with an external clock.

Table 26-11. Clock Divisor Field

CD[15:0]	Action
0	Disables clock
1	Clock divider bypassed
2 to 65535	Baud Rate (Asynchronous Mode) = Selected clock / (16 x CD) Baud Rate (Synchronous Mode) = Selected clock / CD

USART Receiver Time-Out Register

US_RTOR (0x08C)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
TO[15:8]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
TO[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

26-5. IMPORTANT NOTICE

When the receiver is disabled by setting the RXDIS bit in the US_CR register, the time-out is stopped. If the receiver is re-enabled by setting the RXEN bit in the US_CR register, the timeout restarts where it left off (i.e. it is not reset).

- **TO[15:0] : Time-out value**

When a value is written to this register, a time-out start command is automatically performed.

Table 26-12. Time-Out Configuration Field

TO[15:0]	Action
0	Disables the receiver time-out function.
1–65565	The time-out counter is loaded with TO[15:0] when the time-out start command is given or when each new data character is received (after reception has started).

In asynchronous mode : Time-out duration = TO[15:0] x Bit period.

In synchronous mode : Time-out duration = TO[15:0] x 16 x Bit period.

USART Transmit Time-Guard Register

US_TTGR (0x090)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
TG[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **TG[7:0] : Time-guard Value**

Table 26-13. Time-Guard Configuration Field

TO[15:0]	Action
0	Disables the transmitter time-guard function.
1–255	TXD is inactive high after the transmission of each character for the time-guard duration.

Time-guard duration = TG[7:0] x Bit period.

USART LIN Identifier Register

US_LIR (0x094)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	WAKE_UP_TIME[13:8]					
R/W-0	R/W-0	R/W-1	R/W-1	R/W-1	R/W-0	R/W-1	R/W-0
23	22	21	20	19	18	17	16
WAKE_UP_TIME[7:0]							
R/W-1	R/W-1	R/W-0	R/W-1	R/W-0	R/W-1	R/W-0	R/W-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	CHK_SEL	NDATA[2]
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
NDATA[1:0]		IDENTIFIER[5:0]					
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

26-6. IMPORTANT NOTICE

It is not possible to write on this register during a Message Transfer.

- IDENTIFIER[5:0] : LIN identifier**

Indicates the LIN's identifier content to be transmitted on the next "HEADER MESSAGE". For the LIN1.2 release, the IDENTIFIER[5:4] field specifies the number of data field.

Table 26-14. Number of Data Field for the LIN1.2 Release

IDENTIFIER[5:4]	Number of Data Field
0	2
1	2
2	4
3	8

- NDATA[2:0] : Number of data field for the LIN2.0 release**

Specifies the number of data field to be sent or received. The range is from 0 up to 7 which corresponds to 1 up to 8 data field.

- CHK_SEL : checksum selection**

0: Classic checksum.

1: Enhanced checksum (compliant to LIN2.0 release)

- WAKE_UP_TIME[13:0] : Wake up time for the LIN2.0 release**

Sets the wake up time counter. After reset, the counter is loaded with the 0x3AD4 value which corresponds at least 753us for PCLK=20 MHz.

USART Data Field Write 0 Register **US_DFWR0 (0x098)** **Access: Read/Write**

31	30	29	28	27	26	25	24
DATA3[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
DATA2[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
DATA1[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
DATA0[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

26-7. IMPORTANT NOTICE

It is not possible to write on this register during a Message Transfer.

- **DATA[7:0] : LIN's byte field to be transmitted**

Data to be transmitted on the "LIN's RESPONSE MESSAGE" when the STRESP will be set on the Controller Register.

USART Data Field Write 1 Register

US_DFWR1 (0x09C)

Access: Read/Write

31	30	29	28	27	26	25	24
DATA7[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
DATA6[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
DATA5[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
DATA4[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset***26-8. IMPORTANT NOTICE**

It is not possible to write on this register during a Message Transfer.

- **DATA7[7:0] : LIN's byte field to be transmitted**

Data to be transmitted on the "LIN RESPONSE MESSAGE" when the STRESP will be set on the Controller Register.

USART Data Field Read 0 Register **US_DFRR0 (0x0A0)** **Access: Read only**

31	30	29	28	27	26	25	24
DATA3[7:0]							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
DATA2[7:0]							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
DATA1[7:0]							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
DATA0[7:0]							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **DATAx[7:0] : LIN's byte field to be received**

Data received in the "LIN RESPONSE MESSAGE" when the ENDMESS is set in the US_SR register.

USART Data Field Read 1 Register

US_DFRR1 (0x0A4)

Access: Read only

31	30	29	28	27	26	25	24
DATA7[7:0]							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
DATA6[7:0]							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
DATA5[7:0]							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
DATA4[7:0]							
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- DATA[7:0] : LIN's byte field to be received**

Data received in the "LIN RESPONSE MESSAGE" when the ENDMESS is set in the US_SR register.

USART Synchronous Break Length Register US_SBLR (0x0A8) Access: Read/Write

31	30	29	28	27	26	25	24	
–	–	–	–	–	–	–	–	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
23	22	21	20	19	18	17	16	
–	–	–	–	–	–	–	–	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
15	14	13	12	11	10	9	8	
–	–	–	–	–	–	–	–	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	
7	6	5	4	3	2	1	0	
–	–	–	SYNC_BRK[4:0]					–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1	R/W-0	R/W-1	

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset***26-9. IMPORTANT NOTICE**

It is not possible to write on this register during a Message Transfer.

- **SYNC_BRK[4:0] : Synchronous break length**

Values from 0 to 7 can not be written in the register (register keeps pervious value).

27

WATCHDOG (WD)

1. OVERVIEW

1.1 FUNCTIONAL DESCRIPTION

The watchdog timer is used to prevent the system from locking-up (for example in infinite software loops). If the software does not write to the watchdog during the programmed time, then it can generate an interrupt (WDOVF) or an internal reset.

The watchdog timer has a programmable 16-bit down counter in WD_MR register.

The software can control the action to perform when the WD counter overflows (i.e. reaches 0) :

- If the RSTEN bit is set in the WD_OMR register, an internal reset is generated.
- If the WDOVF bit is set in the WD_IMR register, an interrupt is generated on the Generic Interrupt Controller (GIC).

The low frequency clock from the clock manager supplies the watchdog counter through the programmable divider WDPDIV.

There is a possibility to set a programmable pending window where users can restart the watchdog counter only within this window. This protection is set with the RSTALW bit, otherwise users can restart the watchdog counter whenever. When the pending window is reached, the WDPEND bit is set followed by the PENDING bit.

The LFCLK is divided by the WDPDIV[2:0] divider and provided to the down-counter input WDCLK.

All write accesses are protected by control access keys to help prevent corruption of the watchdog.

To update the contents of the mode and control registers, it is necessary to write the correct bit pattern to the control access key bits at the same time as the control bits are written (the same write access).

2. FUNCTIONAL OPERATION

2.1 WATCHDOG FUNCTIONALITY

2.1.1 General Description

The WD contains a programmable length down-counter. The down-counter input clock is a subdivision of the low frequency clock (LFCLK) from the clock manager :

$$\text{WDCLK} = \text{LFCLK} / \text{WDPDIV}$$

The count length determines the timeout period, and is controlled by loading PCV field of WD_MR register. The time out period (in seconds) is :

$$(\text{PCV}[15:0]+1) / \text{WDCLK}$$

When the counter reaches the value programmed in the pending windows PWL[15:0] of WD_PWR register, the watchdog can generate a watchdog pending interrupt.

The pending interrupt occurs after :

$$\{(\text{PCV}[15:0]) - (\text{PWL}[15:0])\} / \text{WDCLK}$$

If the previous time is negative, the WD pending interrupt should not be used.

In order to prevent an internal reset (if RSTEN bit is set in the WD_OMR) or interrupt (if bit WDOVF is set in the WD_IMR), the software must reset the counter before it reaches 0 by writing the correct key in the WD_CR register (0xC071). The time (in seconds) between the WD pending interrupt and the WD overflow is :

$$(\text{PWL}[15:0]) / \text{WDCLKfreq}$$

When the counter reaches 0, it triggers the programmed action (internal reset or interrupt).

If no WD reset is programmed (i.e. RSTEN is at a logical 0) when the WD reaches 0, it is reset to the programmed value and continues to down count, unless it is disabled. This is to be used to generate periodic interrupts.

2.1.2 Block Diagram

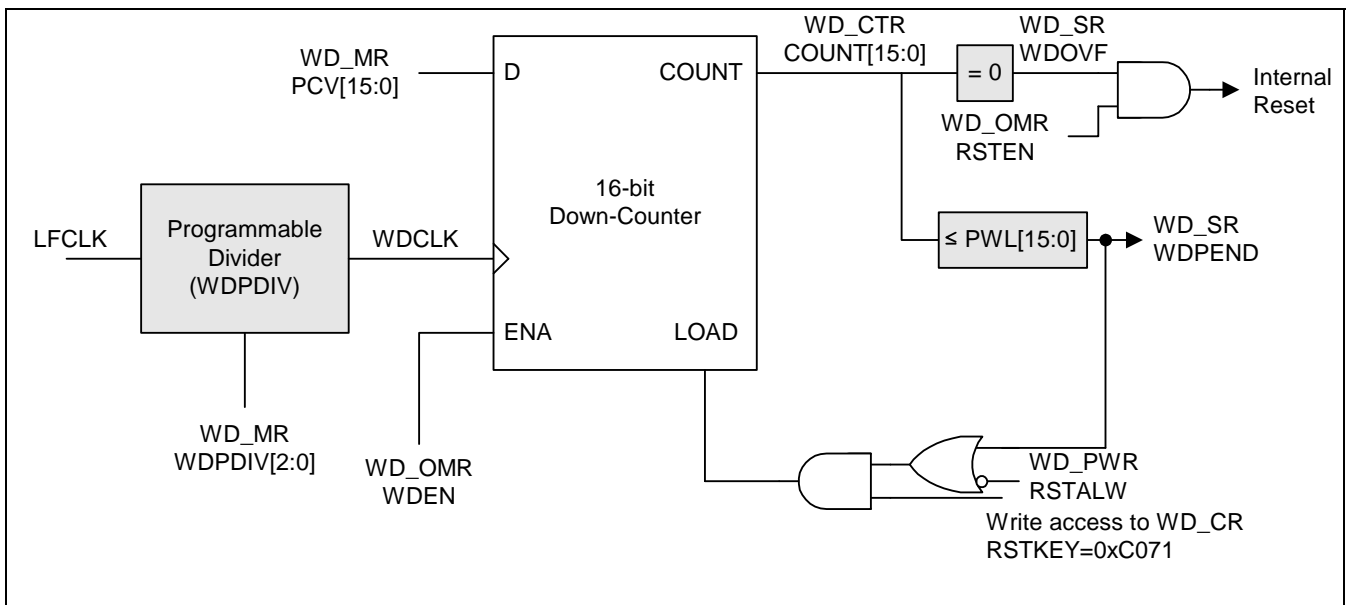


Figure 27-1. Watchdog Block Diagram

2.1.3 Watchdog Events

2.1.3.1 Internal reset pulse generation

If the RSTEN bit is set in the WD_OMR register, an internal reset pulse is generated when the overflow occurs. This pulse is 8 clock cycles long (CORECLK) and is not affected by hardware reset.

After a reset (hardware or WD), the clock selected by the watchdog is LFCLK/2. (See P28-8)

2.1.3.2 Internal interrupt request

The watchdog can generate an internal interrupt request when the overflow occurs. The software can enable or disable this interrupt either in the WD module (WD_IMR register) or in the GIC registers.

2.2 EXAMPLE

Example of use of the watchdog :

Use of the windows to generate an interrupt and reload the watchdog counter within the windows only. If the interruption is not called due to a bug, a reset occurs when the watchdog counter reaches 0.

Configuration :

- Configuration of WD_MR : Choice of the clock to decrease counter, Preload value from which counter starts to decrease.
- Configuration of WD_PWR : Upper limit of the window from which it generates an interrupt when reached and the bit which allows to restart counter only within this window.
- Configuration of WD_IER : Enable Interrupt at the peripheral level when the windows is reached (bit WDPEND) or when the counter overflow (bit WDOVF if watchdog reset is not enabled), GIC must be configured.
- Configuration of WD_OMR : Enable the watchdog (start decrement the counter) and enable the watchdog reset if counter overflow.

Interruption Handling :

- IRQ Entry and call C function.
- Read WD_SR and verify the source of the interrupt.
- Clear the corresponding interrupt at peripheral level by writing in the WD_CSR.
- Interrupt treatment : If this is a windows pending interrupt, restart the watchdog by writing in WD_CR.
- IRQ Exit.

3. REGISTERS DESCRIPTION

Base Address – 0xFFE14000

Table 27-1. Watchdog Special Function Registers

Offset Address	Name	Description	R/W	Reset Status
0x000 – 0x05C	–	Reserved	–	–
0x050	WD_ECR	Enable clock register	W	–
0x054	WD_DCR	Disable clock register	W	–
0x058	WD_PMSR	Power management status register	R	0XXXXXXXX0
0x05C	–	Reserved	–	–
0x060	WD_CR	Control register	W	–
0x064	WD_MR	Mode register	R/W	0x0007FF00
0x068	WD_OMR	Overflow mode register	R/W	0x00000000
0x06C	WD_CSR	Clear status register	W	–
0x070	WD_SR	Status register	R	0x00000101
0x074	WD_IER	Interrupt enable register	W	–
0x078	WD_IDR	Interrupt disable register	W	–
0x07C	WD_IMR	Interrupt mask register	R	0x00000000
0x080	WD_PWR	Pending window register	R	0x0007FF00
0x084	WD_CTR	Counter test register	R	0x000007FF

WD Enable Clock Register

WD_ECR (0x050)

Access: Write only

31	30	29	28	27	26	25	24
DBGEN	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **DBGEN : Debug mode enable**

0: No effect

1: Enable debug mode

WD Disable Clock Register

WD_DCR (0x054)

Access: Write only

31	30	29	28	27	26	25	24
DBGEN	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **DBGEN : Debug mode disable**

0: No effect

1: Disable debug mode

WD Power Management Status Register

WD_PMSR (0x058)

Access: Read only

31	30	29	28	27	26	25	24
DBGEN	–	IPIDCODE[25:20]					
R-0	R-0	R-U	R-U	R-U	R-U	R-U	R-U
23	22	21	20	19	18	17	16
IPIDCODE[19:12]							
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U
15	14	13	12	11	10	9	8
IPIDCODE[11:4]							
R-U	R-U	R-U	R-U	R-U	R-U	R-U	R-U
7	6	5	4	3	2	1	0
IPIDCODE[3:0]				–	–	–	–
R-U	R-U	R-U	R-U	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **DBGEN : Debug mode**

0: DBGACK has no influence on Watchdog.

1: DBGACK has influence on Watchdog.

NOTE: There is no WD bit because the watchdog module has no power management (always running).

- **IPIDCODE[25:0] : IP identifier code**

The field contains the version number of the module, coded on 26 bits.

WD Control Register **WD_CR (0x060)** **Access: Write only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
RSTKEY[15:8]							
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
RSTKEY[7:0]							
W	W	W	W	W	W	W	W

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- **RSTKEY[15:0] : Restart key**

0xC071 : Watchdog counter is restarted if its value is equal or less than pending window length or if pending window is disabled.

Other value : No effect.

NOTE: A restart command (write the restart key in WD_CR register) will not be effective if it occurs less than 2 WDCLK + 1/2 LFCLK periods after a previous start command.

WD Mode Register **WD_MR (0x064)** **Access: Read/Write**

31	30	29	28	27	26	25	24
CKEY[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
PCV[15:8]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1	R/W-1
15	14	13	12	11	10	9	8
PCV[7:0]							
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
7	6	5	4	3	2	1	0
–	–	–	–	–	WDPDIV[2:0]		
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

W: Write R: Read -0: 0 After reset -1: 1 After reset -U: Undefined after reset

- WDPDIV[2:0] : WD clock divider**

Table 27-2. Watchdog Clock Divider

WDPDIV[2:0]			WDCLK
0	0	0	LFCLK/2
0	0	1	LFCLK/4
0	1	0	LFCLK/8
0	1	1	LFCLK/16
1	0	0	LFCLK/32
1	0	1	LFCLK/128
1	1	0	LFCLK/256
1	1	1	LFCLK/512

- PCV[15:0] : Preload counter value**

Counter is preloaded when watchdog counter is restarted. Time to overflow = PCV[15:0] / WDCLK

- CKEY[7:0] : Clock access key**

Used only when writing in WD_MR. CKEY is read as 0.

Write access in WD_MR is allowed only if CKEY[7:0] = 0x37.

WD Overflow Mode Register

WD_OMR (0x068)

Access: Read/Write

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
15	14	13	12	11	10	9	8
OKEY[11:4]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
OKEY[3:0]				–	–	RSTEN	WDEN
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **WDEN : Watchdog enable**

0: Watchdog is disabled.

1: Watchdog is enabled.

- **RSTEN : Reset enable**

0: Generation of an internal reset by the Watchdog is disabled.

1: When overflow occurs, the Watchdog generates an internal reset.

- **OKEY[11:0] : Overflow access key**

Used only when writing WD_OMR. OKEY is read as 0.

0x234 : Write access in WD_OMR is allowed.

Other value : Write access in WD_OMR is prohibited.

WD Clear Status Register **WD_CSR (0x06C)** **Access: Write only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	WDOVF	WDPEND
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **WDPEND : Clear Watchdog Pending**

0: No effect.

1: Clear Watchdog pending interrupt.

- **WDOVF : Clear Watchdog Overflow**

0: No effect.

1: Clear Watchdog overflow interrupt.

WD Status Register

WD_SR (0x070)

Access: Read only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	CLEAR_STATUS	PENDING
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-1
7	6	5	4	3	2	1	0
–	–	–	–	–	–	WDOVF	WDPEND
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-1

W: Write

R: Read

-0: 0 After reset

-1: 1 After reset

-U: Undefined after reset

- **WDPEND : Watchdog Pending**

0: No Watchdog pending.

1: A Watchdog pending has occurred.

- **WDOVF : Watchdog Overflow**

0: No Watchdog overflow.

1: A Watchdog overflow has occurred.

- **PENDING : Watchdog Pending status**

0: Watchdog counter is over pending window length.

1: Watchdog counter is equal or less than pending window length.

- **CLEAR_STATUS : Clear Status**

0: Watchdog has ended its reset.

1: Watchdog resets on process.

WD Interrupt Enable Register **WD_IER (0x074)** **Access: Write only**

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	WDOVF	WDPEND
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **WDPEND : Watchdog Pending interrupt enable**

0: No effect

1: Enable the WDPEND interrupt

- **WDOVF : Watchdog Overflow interrupt enable**

0: No effect

1: Enable the WDOVF interrupt

WD Interrupt Disable Register

WD_IDR (0x078)

Access: Write only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
W	W	W	W	W	W	W	W
7	6	5	4	3	2	1	0
–	–	–	–	–	–	WDOVF	WDPEND
W	W	W	W	W	W	W	W

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **WDPEND : Watchdog Pending interrupt disable**

0: No effect

1: Disable the WDPEND interrupt

- **WDOVF : Watchdog Overflow interrupt disable**

0: No effect

1: Disable the WDOVF interrupt

WD Interrupt Mask Register

WD_IMR (0x07C)

Access: Read only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
–	–	–	–	–	–	WDOVF	WDPEND
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **WDPEND : Watchdog Pending interrupt mask**

0: The WDPEND interrupt is disabled.

1: The WDPEND interrupt is enabled.

- **WDOVF : Watchdog Overflow interrupt mask**

0: The WDOVF interrupt is disabled.

1: The WDOVF interrupt is enabled.

WD Pending Windows Register **WD_PWR (0x080)** **Access: Read/Write**

31	30	29	28	27	26	25	24
PWKEY[7:0]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
23	22	21	20	19	18	17	16
PWL[15:8]							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1	R/W-1
15	14	13	12	11	10	9	8
PWL[7:0]							
R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	RSTALW
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **RSTALW : Restart allowed**

0: Restart allowed every time.

1: Restart allowed only within pending window.

This bit does not disable the bit WDPEND in the interrupt register.

- **PWL[15:0] : Pending Window Length**

Length of the window. Time from preload value of watchdog pending = $(PCV[15:0] - PWL[15:0]) / WDCLK$

- **PWKEY[7:0] : Pending window access key**

Used only when writing in WD_PWR. PWKEY is read as 0.

Write access in WD_PWR is allowed only if PWKEY[7:0] = 0x91.

WD Counter Test Register

WD_CTR (0x084)

Access: Read only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
23	22	21	20	19	18	17	16
–	–	–	–	–	–	–	RESET
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
15	14	13	12	11	10	9	8
COUNT[15:8]							
R-0	R-0	R-0	R-0	R-0	R-1	R-1	R-1
7	6	5	4	3	2	1	0
COUNT[7:0]							
R-1	R-1	R-1	R-1	R-1	R-1	R-1	R-1

*W: Write**R: Read**-0: 0 After reset**-1: 1 After reset**-U: Undefined after reset*

- **COUNT[15:0] : Counter**

Value of Watchdog counter.

- **RESET : Watchdog counter resetting**

0: No watchdog counter reset occurred or watchdog counter reset is achieved.

1: A watchdog counter reset is pending.

As watchdog counter is asynchronous, the write of the right key in WD_CR will not immediately set the watchdog value to initial value.

During this delay this bit is set to one (between reset command and real reset of the counter).

For the same reason, this register must be read twice to be sure of the value.

28

ELECTRICAL DATA

1. ABSOLUTE MAXIMUM RATINGS

Table 28-1. Absolute Maximum Ratings

Parameter	Symbol	Conditions	Rating	Unit
DC supply voltage for VDD core	V_{DD_CORE}	–	–0.3 to 6.0	V
DC supply voltage for I/O block	V_{DD_IO}	–	–0.3 to 6.0	V
DC supply voltage for motor	V_{DD_MOTOR}	–	–0.3 to 6.0	V
DC supply voltage for AVDD	AV_{DD}	–	–0.3 to 6.0	V
DC supply voltage for AVref	AV_{REF}	–	–0.3 to 6.0	V
DC supply voltage for VLCD	V_{LCD}	–	–0.3 to 6.0	V
Digital I/O input voltage	V_{IN}	–	–0.3 to $V_{DD_IO}+0.3$	V
Analog I/O input voltage	AV_{IN}	–	–0.3 to $AV_{DD}+0.3$	V
Output voltage	V_O	All output pins	–0.3 to $V_{DD_IO}+0.3$	V
Operating temperature	T_A	–	–40 to 105	°C
Storage temperature	T_{STG}	–	–65 to 155	°C

2. RECOMMENDED OPERATING CONDITIONS

Table 28-2. Recommended Operating Conditions

Parameter	Symbol	Condition	Rating	Unit
DC supply voltage for V_{DD} core	V_{DD_CORE}	–	3.0 to 5.5	V
DC supply voltage for I/O block	V_{DD_IO}	–	3.0 to 5.5	
DC supply voltage for motor	V_{DD_MOTOR}	–	4.5 to 5.5	
DC supply voltage for AV_{DD}	AV_{DD}	–	4.5 to 5.5	
DC supply voltage for AV_{REF}	AV_{REF}	–	4.5 to 5.5	
DC supply voltage for V_{LCD}	V_{LCD}	–	3.0 to 5.5	
Operating temperature	T_A	–	–40 to 105 (not including stepper motor) –40 to 85 (including stepper motor)	°C

3. D.C. ELECTRICAL CHARACTERISTICS

Table 28-3. D.C. Electrical Characteristics (5V I/O)

($T_A = -40$ to 105 °C, $V_{DD} = 4.5V$ to $5.5V$)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Input high voltage	V_{IH1}	All input pins except V_{IH2} , V_{IH3} and V_{IH4}	$0.8 V_{DD}$	–	V_{DD}	V
	V_{IH2}	SCL0/1, SDA0/1, UARTRXD0/1, USARTRXD0	$0.7 V_{DD}$	–	V_{DD}	
	V_{IH3}	MD0, MD1	$V_{DD} - 0.3$	–	V_{DD}	
	V_{IH4}	X_{IN}	$V_{DD} - 0.3$	–	V_{DD}	
Input low voltage	V_{IL1}	All input pins except V_{IL2} , V_{IL3} and V_{IL4}	0	–	$0.4 V_{DD}$	V
	V_{IL2}	SCL0/1, SDA0/1, UARTRXD0/1, USARTRXD0	–	–	$0.3 V_{DD}$	
	V_{IL3}	MD0, MD1	–	–	0.3	
	V_{IL4}	X_{IN}	–	–	0.3	
Output high voltage	V_{OH1}	$I_{OH} = -2mA$	$V_{DD} - 0.8$	–	–	V
Output low voltage	V_{OL1}	$I_{OL} = 2mA$	–	–	0.8	V

NOTE: When stepper motor is used, the operating temperature range is -40 to 85 °C.

Table 28-3. D.C. Electrical Characteristics (5V I/O. Continued)

 $(T_A = -40$ to $105\text{ }^\circ\text{C}$, $V_{DD} = 4.5\text{V}$ to 5.5V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Input high leakage current	I_{LIH1}	$V_{IN} = V_{DD}$ All input pins except I_{LIH2} , I_{LIH3} and X_{OUT}	–	–	1	μA
	I_{LIH2}	$V_{IN} = V_{DD}$ All SMxx pins			2.5	μA
	I_{LIH3}	$V_{IN} = V_{DD}$, X_{IN} pin			20	
Input low leakage current	I_{LIL1}	$V_{IN} = 0\text{ V}$ All input pins except I_{LIH2} , I_{LIH3} and X_{OUT}	–	–	– 1	μA
	I_{LIL2}	$V_{IN} = V_{DD}$ All SMxx pins			– 2.5	μA
	I_{LIL3}	$V_{IN} = 0\text{ V}$, X_{IN} pin			– 20	
Output high leakage current	I_{LOH1}	$V_{OUT} = 0\text{ V}$ All output pins except I_{LOH2}	–	–	1	μA
	I_{LOH2}	$V_{OUT} = V_{DD}$ All SMxx pins	–	–	2.5	μA
Output low leakage current	I_{LOL1}	$V_{OUT} = 0\text{ V}$ All output pins except I_{LOH2}	–	–	– 1	μA
	I_{LOL2}	$V_{OUT} = 0\text{ V}$ All SMxx pins	–	–	– 2.5	μA
Output rise time	T_R	SMxx pins in output mode with slew control enabled, $V_{DD} = 5\text{ V}$, $R_{load} = 100\text{pF}$, 10% to 90% of V_{OH}	70	100	130	ns
Output fall time	T_F	SMxx pins in output mode with slew control enabled, $V_{DD} = 5\text{ V}$, $R_{load} = 100\text{pF}$, 10% to 90% of V_{OH}	70	100	130	ns
Pull-up resistors	R_{L1}	$V_{DD} = 5\text{V}$, $V_{IN} = 0\text{ V}$, All pull-up controllable GPIO port and nRESET	25	50	100	$\text{k}\Omega$
Feedback resistor	R_{FD}	$V_{IN} = V_{DD}$, $V_{DD} = 5\text{V}$, X_{IN}	500	1000	1500	$\text{k}\Omega$

NOTE: When stepper motor is used, the operating temperature range is -40 to $85\text{ }^\circ\text{C}$.

Table 28-4. D.C. Electrical Characteristics (3.3V I/O)

(T_A = -40 to 105 °C, V_{DD} = 3.0V to 3.6V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Input high voltage	V _{IH1}	All input pins except V _{IH2} , V _{IH3} and V _{IH4}	0.8 V _{DD}	-	V _{DD}	V
	V _{IH2}	SCL0/1, SDA0/1, UARTRXD0/1, USARTRXD0	0.7 V _{DD}		V _{DD}	
	V _{IH3}	MD0, MD1	V _{DD} - 0.3		V _{DD}	
	V _{IH4}	X _{IN}	V _{DD} - 0.3		V _{DD}	
Input low voltage	V _{IL1}	All input pins except V _{IL2} , V _{IL3} and V _{IL4}	0	-	0.3 V _{DD}	V
	V _{IL2}	SCL0/1, SDA0/1, UARTRXD0/1, USARTRXD0			0.3 V _{DD}	
	V _{IL3}	MD0, MD1			0.3	
	V _{IL4}	X _{IN}			0.3	
Output high voltage	V _{OH1}	I _{OH} = -2mA	V _{DD} - 0.8	-	-	V
Output low voltage	V _{OL1}	I _{OL} = 2mA	-	-	0.8	V

Table 28-4. D.C. Electrical Characteristics (3.3V I/O Continued)

(T_A = -40 to 105 °C, V_{DD} = 3.0V to 3.6V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Input high leakage current	I _{LIH1}	V _{IN} = V _{DD} All input pins except I _{LIH2} , I _{LIH3} and X _{OUT}	-	-	1	μA
	I _{LIH2}	V _{IN} = V _{DD} All SMxx pins			2.5	μA
	I _{LIH3}	V _{IN} = V _{DD} , X _{IN} pin			20	
Input low leakage current	I _{LIL1}	V _{IN} = 0 V All input pins except I _{LIL2} , I _{LIL3} and X _{OUT}	-	-	-1	μA
	I _{LIL2}	V _{IN} = 0 V All SMxx pins			-2.5	μA
	I _{LIL3}	V _{IN} = 0 V, X _{IN} pin			-20	
Output high leakage current	I _{LOH1}	V _{OUT} = V _{DD} All output pins except I _{LOH2}	-	-	1	μA
	I _{LOH2}	V _{OUT} = V _{DD} All SMxx pins			2.5	μA
Output low leakage current	I _{LOL1}	V _{OUT} = 0 V All output pins except I _{LOH2}	-	-	-1	μA
	I _{LOL2}	V _{OUT} = 0 V All SMxx pins			-2.5	μA
Pull-up resistors	R _{L1}	V _{IN} = 0 V, V _{DD} = 3.3V, All pull-up controllable port except nRESET	25	50	100	kΩ
Feedback resistor	R _{FD}	V _{IN} = V _{DD} , V _{DD} = 3.3V, X _{IN}	800	1300	2500	kΩ

Table 28-5. Current Consumption

(T_A = -40 to 105 °C, V_{DD} = 3.0V to 5.5V)

Parameter	Symbol	Min	Typ	Max	Unit
Normal mode current consumption (PLL disable, f _{OSC} = 6MHz)	I _{DD1}	-	-	35	mA
Normal mode current consumption (PLL disable, f _{OSC} = 4MHz)	I _{DD2}	-	-	30	
High speed mode current consumption (PLL = 40MHz, CORECLK=40MHz, PCLK=20MHz)	I _{DD3}	-	-	110	
Halt mode current consumption (Only master clock and timer, f _{OSC} = 6MHz)	I _{DD41}	-	-	3.1	
Halt mode current consumption (Only internal oscillator and timer)	I _{DD42}	-	-	1.5	
Stop mode current consumption	I _{DD5}	-	-	200	uA

Table 28-6. Oscillation Characteristics

(T_A = -40 to 105 °C, V_{DD} = 3.0V to 5.5V)

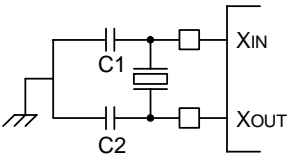
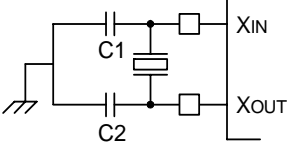
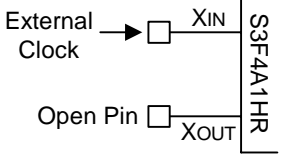
Oscillator	Clock Circuit	Conditions	Min	Typ	Max	Unit
Crystal		CPU clock oscillation frequency	4	-	6	MHz
Ceramic		CPU clock oscillation frequency	4	-	6	MHz
PLL	-	-	12	-	40	MHz
External clock		X _{IN} input frequency	4	-	6	MHz

Table 28-7. Oscillation Stabilization Time(T_A = -40 to 105 °C, V_{DD} = 3.0V to 5.5V)

Oscillator	Test Condition	Min	Typ	Max	Unit
Master crystal	Oscillation stabilization occurs when V _{DD} is equal to the minimum oscillator voltage range.	–	–	10	ms
Master ceramic		–	–	10	ms
PLL stabilization time	PLL stabilization occurs when V _{DD} is equal to the minimum oscillator voltage range.	–	–	150	us
Internal oscillator stabilization time	Internal oscillator stabilization occurs when V _{DD} is equal to the minimum oscillator voltage range.	–	–	100	us
Oscillator stabilization wait time	t _{WAIT} when released by a reset	–	2 ¹⁶ /F _{osc}	–	ms
	t _{WAIT} when released by an interrupt	–	2 ¹⁶ /F _{osc}	–	ms

Table 28-8. Internal RC Oscillation Characteristics(T_A = -40 to 105 °C, V_{DD} = 3.0V to 5.5V)

Oscillator	Symbol	Condition	Min	Typ	Max	Unit
Internal oscillator frequency	F _{IOSC}	–	0.8	1	1.2	MHz
Output clock duty ratio	T _{OD}	–	40	–	60	%

Table 28-9. LVD characteristics(T_A = -40 to 105 °C)

Parameter	Symbol	Conditions	Min	Typ	Max	Units
LVD detection voltage (Reset)	V _{LVD_RESET}	–	2.2	2.4	2.6	V
LVD detection voltage (Interrupt)	V _{LVD_F_INT}	–	3.9	4.2	4.5	V

Table 28-10. Input/Output Capacitance

 $(T_A = -40 \text{ to } 105 \text{ }^\circ\text{C})$

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Input capacitance	C _{IN}	f = 1 MHz V _{DD} = 0 V, unmeasured pins are connected to V _{SS}	–	–	10	pF
Output capacitance	C _{OUT}	–				
I/O capacitance	C _{IO}	–				

Table 28-11. A.C. Electrical Characteristics

 $(T_A = -40 \text{ to } 105 \text{ }^\circ\text{C})$

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Interrupt input high, low width	t _{INTH} , t _{INTL}	–	10	–	–	us
nRESET input low width	t _{RSL}	–	10	–	–	

Table 28-12. A.C. Electrical Characteristics for Internal Flash ROM

 $(T_A = -40 \text{ to } 105 \text{ }^\circ\text{C})$

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Programming time for byte / half-word / word	Ftp	–	28	41	54	μs
Page erasing time	Ftp1	–	2.8	4	5.2	ms
Sector / Chip erasing time	Ftp2	–	22	32	42	ms
Data access time	Ft _{RS}	–	–	–	50	ns
Number of writing/erasing for program flash	Fnwep	$T_A = 25 \text{ }^\circ\text{C}$	10,000	–	–	Times
		$T_A = -40 \text{ to } 105 \text{ }^\circ\text{C}$	1,000	–	–	
Number of writing/erasing for data flash	Fnwed	–	40,000	–	–	Times
Data retention	Ftdr	–	10	–	–	Years

Table 28-13. ADC Electrical Characteristics

 $(T_A = -40 \text{ to } 105 \text{ }^\circ\text{C}, AV_{DD} = 5V \pm 10\%, AV_{REF} = AV_{DD})$

Parameter	Condition	Min	Typ	Max	Unit
Resolution	–	–	10	–	Bit
Integral Non-Linearity	$AV_{REF} = 5V,$ $AV_{SS} = GND$	–	± 1	± 2	LSB
Differential Non-Linearity	$AV_{REF} = 5V,$ $AV_{SS} = GND$	–	± 0.8	± 1	LSB
Bottom offset voltage	$AV_{REF} = 5V,$ $AV_{SS} = GND$	–	10	20	LSB
Top offset voltage	$AV_{REF} = 5V,$ $AV_{SS} = GND$	–	15	30	LSB
Maximum conversion rating	5 ADC_CLK	–	–	500	KSps
ADC clock frequency	@50% duty cycle	–	–	2.5	MHz
Operating supply voltage	–	4.5	5.0	5.5	V

Table 28-14. D.C. Electrical Characteristics for LCD controller

(T_A = -40 to 105 °C, V_{DD} = 3.0V to 5.5V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
LCD voltage dividing resistor	R _{LCD}	T _A = 25 °C	70	110	150	kΩ
V _{LCD} - COM _i voltage drop (i = 0-3)	V _{DC}	-15 μA per common pin	-	45	120	mV
V _{LCD} - SEG _x voltage drop (x = 0-29)	V _{DS}	-15 μA per common pin	-	45	120	mV
Middle output voltage	V _{LC0}	V _{DD} = 3.0 V to 5.5 V	0.6V _{DD} - 0.2	0.6V _{DD}	0.6V _{DD} + 0.2	V
	V _{LC1}		0.4V _{DD} - 0.2	0.4V _{DD}	0.4V _{DD} + 0.2	
	V _{LC2}		0.2V _{DD} - 0.2	0.2V _{DD}	0.2V _{DD} + 0.2	

29 PACKAGE SPECIFICATIONS

1. PACKAGE SPECIFICATIONS

The S3F4A1HR is available in a 100-TQFP-1414 package.

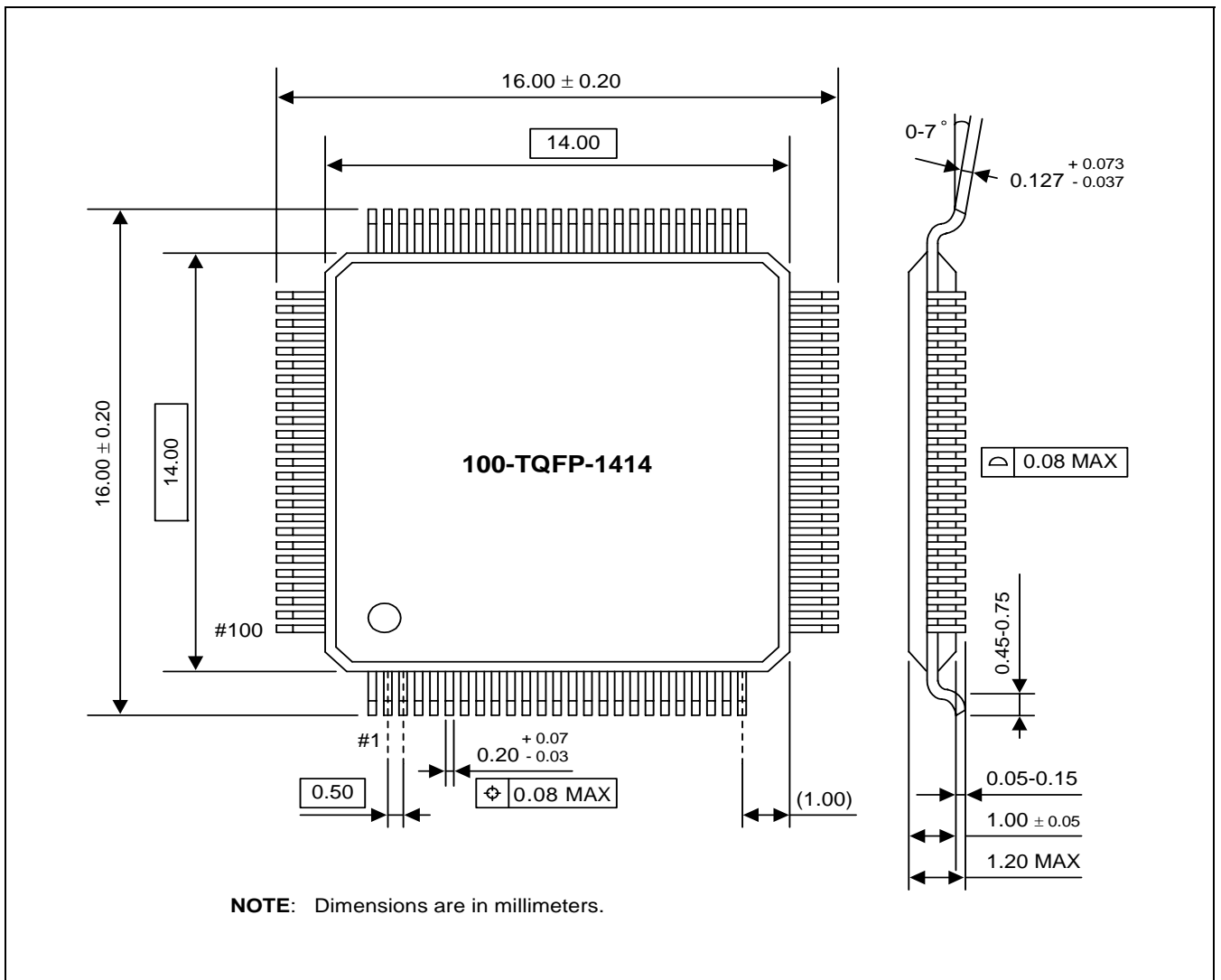


Figure 29-1. 100 Pin Package Dimensions