MPC82E52A

8-bit micro-controller

MEGAWIN

Features

- Enhanced 80C51 Central Processing Unit
- Operation voltage range 4.5V ~ 5.5V, built-in Low-Voltage Detector and Reset circuit
- Operation frequency range up to 25MHz
- 8K bytes on-chip flash memory with ISP/IAP capability
- 256 byte scratch-pad RAM
- Two-level code protection for flash memory access
- Two 16-bit timer/counter
- 7 sources, 4-level-priority interrupt capability
- One enhanced UART with automatic address recognition and frame error detection
- 15 bits Watch-Dog-Timer with 8-bit pre-scalar, one-time enabled
- SPI Master/Slave mode
- One Programmable Counter Array (PCA)
- 8-bit Analog-to-Digital Converter (ADC)
- Power control: idle mode and power-down mode, Power-down can be woken-up through /INT0 and /INT1
- 15 programmable I/O ports
- Alternative built-in 6MHz oscillator
- Fully static operation
- Excellent noise immunity
- Very low power consumption
- Two package type: 20L-SDIP and 20L-SOP

General Description

MPC82E52A is a single-chip 8-bit micro-controller with instruction sets fully compatible with industrial-standard 80C51 series micro controller.

There is very excellent MCU kernel built in this device compared to general 80C51 MCUs those take twelve oscillating cycles to finish an instruction, the device could take only one oscillating cycle to finish one instruction.

There is 8K bytes flash memory embedded which could be used as program or data. Also the In-System Programming and In-Application Programming mechanisms are supported. The data endurance of the embedded flash gets over 20,000 times, and 7 years data retention is guaranteed.

The operation frequency reaches at 25MHz. An user can apply a crystal oscillator for the oscillating source, or alternatively uses the built in 6MHz RC oscillator to save system cost.

The built in high performance Analog-to-Digital Converter make it easy to sensing the environment or implement a set of scan keys in low cost.

The UART and SPI interfaces make the device convenient to communicate with the peripheral component, say talking to a personal computer via RS-232 port, or communicating with a serial memory.

The Pulse-Width-Modulator (PWM) and Programmable Counter Array (PCA) make the device to drive the peripheral step motor or LED in least cost.

The MPC82E52A is really the most efficient MCU adapted for simple control, say electronic scales, remote controller, security encoder/decoder, and user interface controller.

Pin Description

Pin Definition

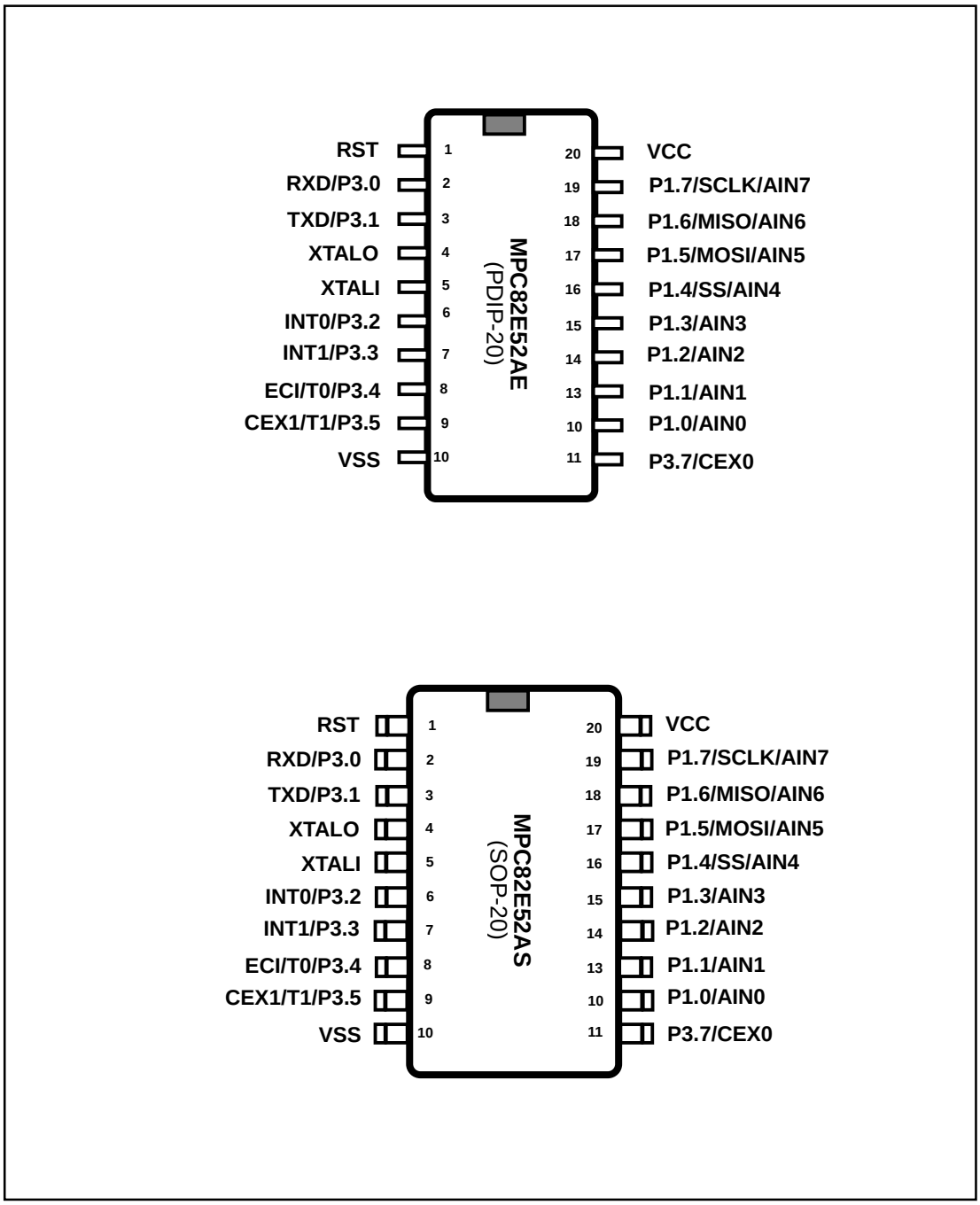
Pin Name	PIN NUMBER		TYPE	DESCRIPTION
	DIP-20	SOP-20		
RST	1	1	ID	**RST** := A high duty on this pin keeps for at least 10us plus 36 oscillation cycles will reset the device.
P3.0 (RXD)	2	2	BU	**P3.0** := General purpose 4-state I/O port with internal pull-up mechanism; can be configured as open-drain output. **RXD** := Data Receiving pin for built-in UART functionality.
P3.1 (TXD)	3	3	BU	**P3.1** := General purpose 4-state I/O port with internal pull-up mechanism; can be configured as open-drain output. **TXD** := Data Transmitting pin for built-in UART functionality.
XTALO XTALI	4 5	4 5	O I	**XTALO** := Output from the inverting oscillator amplifier. **XTALI** := Input to the inverting oscillator amplifier.
P3.2(INT0)	6	6	BU	**P3.2** := General purpose 4-state I/O port with internal pull-up mechanism; can be configured as open-drain output. **INT0** := External interrupt source
P3.3(INT1)	7	7	BU	**P3.3** := General purpose 4-state I/O port with internal pull-up mechanism; can be configured as open-drain output. **INT1** := External interrupt source

P3.4(ECI/T0)	8	8	BU	**P3.4** := General purpose 4-state I/O port with internal pull-up mechanism; can be configured as open-drain output. **ECI** := External Clock Input to Programmable Counter Array(PCA) **T0** := Alternative clock input to timer-0
P3.5(CEX1/T1)	9	9	BU	**P3.5** := General purpose 4-state I/O port with internal pull-up mechanism; can be configured as open-drain output. **CEX1**:= Capture Event trigger to Programmable Counter Array(PCA) module-1 or PWM output **T1** := Alternative clock input to timer-1
VSS	10	10	G	Ground
P3.7(CEX0)	11	11	BU	**P3.6** := General purpose 4-state I/O port with internal pull-up mechanism; can be configured as open-drain output. **CEX0**:= Capture Event trigger to Programmable Counter Array(PCA) module-0 or PWM output
P1.0(AIN0)	12	12	BU	**P1.0** := General purpose 4-state I/O port with internal pull-up mechanism; can be configured as open-drain output. **AIN0**:= Alternative ADC input
P1.1(AIN1)	13	13	BU	**P1.1** := General purpose 4-state I/O port with internal pull-up mechanism; can be configured as open-drain output. **AIN1**:= Alternative ADC input

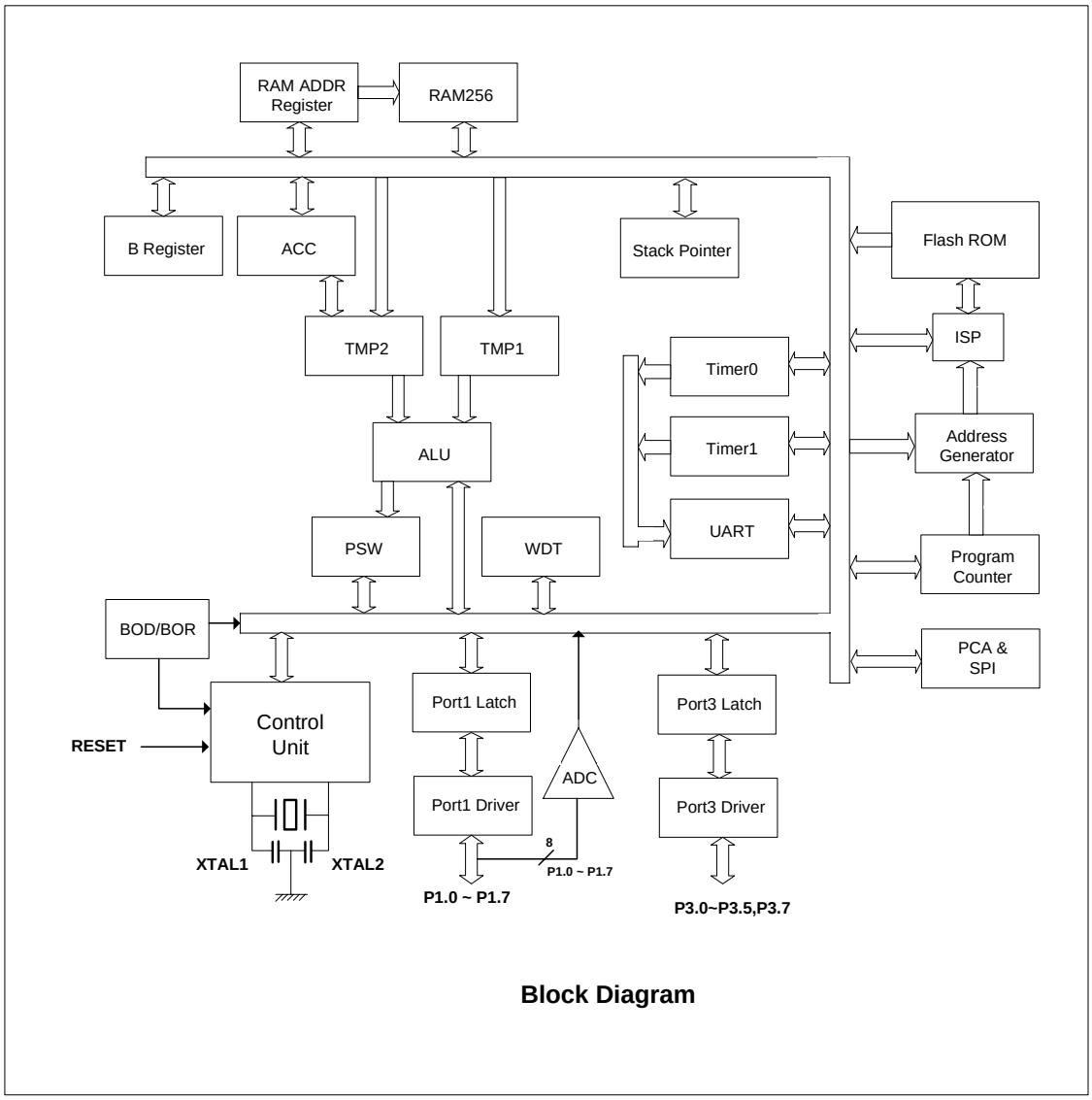
P1.2(AIN2)	14	14	BU	**P1.2** := General purpose 4-state I/O port with internal pull-up mechanism; can be configured as open-drain output. **AIN2**:= Alternative ADC input
P1.3(AIN3)	15	15	BU	**P1.3** := General purpose 4-state I/O port with internal pull-up mechanism; can be configured as open-drain output. **AIN3**:= Alternative ADC input
P1.4(SS/AIN4)	16	16	BU	**P1.3** := General purpose 4-state I/O port with internal pull-up mechanism; can be configured as open-drain output. **SS**:= Serial mode Selector or Chip-Enabling pin for Serial Peripheral Interface(SPI) **AIN4**:= Alternative ADC input
P1.5(MOSI/AIN5)	17	17	BU	**P1.5** := General purpose 4-state I/O port with internal pull-up mechanism; can be configured as open-drain output. **MOSI**:= Master data Output or Slave data Input for Serial Peripheral Interface(SPI) **AIN5**:= Alternative ADC input
P1.6(MOSI/AIN5)	18	18	BU	**P1.6** := General purpose 4-state I/O port with internal pull-up mechanism; can be configured as open-drain output. **MISO**:= Master data Input or Slave data Output for Serial Peripheral Interface(SPI) **AIN6**:= Alternative ADC input

P1.7(MOSI/AIN5)	19	19	BU	**P1.7** := General purpose 4-state I/O port with internal pull-up mechanism; can be configured as open-drain output. **SCLK**:= Serial Clock for Serial Peripheral Interface(SPI) **AIN7**:= Alternative ADC input
VCC	20	20	P	Power supply

Pin Configuration



Block Diagram



Block Diagram

Special Function Register

Address Map

	9	A	B	C	D	E	F
F8		CH	CCAP0H	CCAP1H				
F0	B		PCAPWM0*	PCAPWM1*				
E8		CL	CCAP0L	CCAP1L				
E0	ACC	WDTCR	IFD	IFADRH	IFADRL	IFMT	SCMD	ISPCR
D8	CCON	CMOD	CCAPM0	CCAPM1				
D0	PSW							
C8								
C0						ADCTL	ADCV	PCON2*
B8	IP	SADEN						
B0	P3	P3M0	P3M1					IPH
A8	IE	SADDR						
A0								*reserved*
98	SCON	SBUF						
90	P1	P1M0	P1M1					
88	TCON	TMOD	TL0	TL1	TH0	TH1	AUXR	
80		SP	DPL	DPH	SPISTAT	SPICTL	SPIDAT	PCON

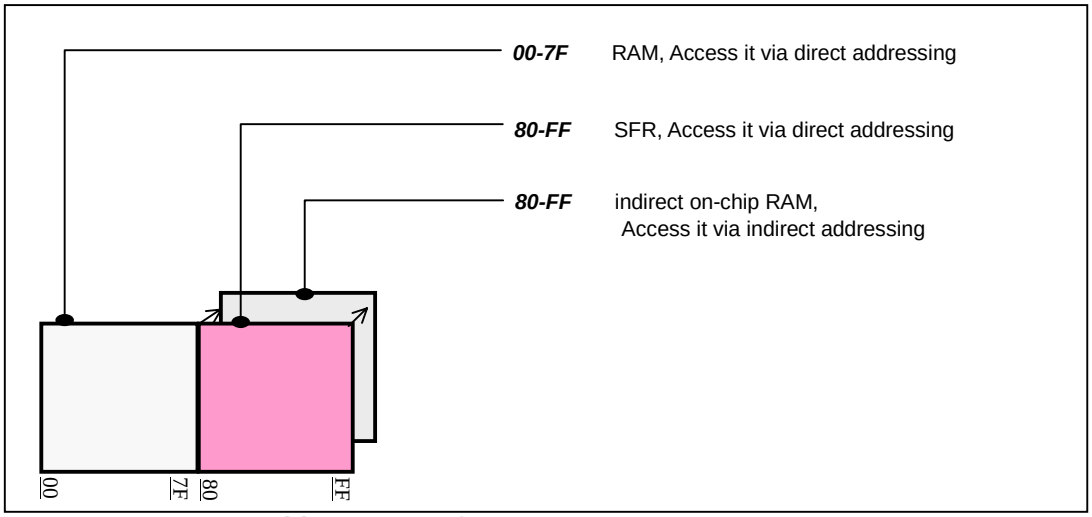
* Write Only

Bits Description

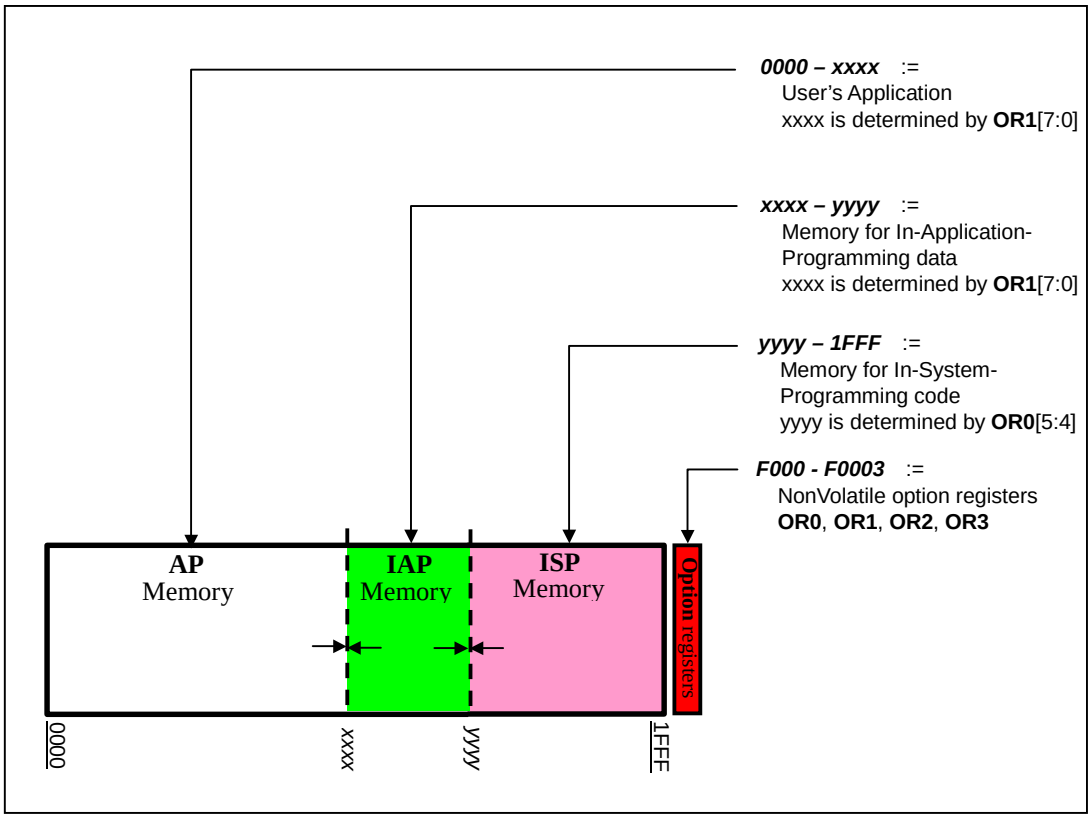
SYMBOL	DESCRIPTION									INITIAL VALUE
SP	Stack Pointer									00000111B
DPL	Data Pointer Low									00000000B
DPH	Data Pointer High									00000000B
SPISTAT	SPI status register	SPIF	WCOL							00xxxxxxB
SPICTL	SPI control register	SSIG	SPEN	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	00000000B
SPIDAT	SPI data register									00000000B
PCON	Power Control	SMOD	*reserved0*	LVF	POF	GF1	GF0	PD	IDL	00110000B
TCON	Timer Control	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	00000000B
TMOD	Timer Mode	GATE	C//T	M1	M0	GATE	C//T	M1	M0	00000000B
TL0	Timer Low 0									00000000B
TL1	Timer Low 1									00000000B
TH0	Timer High 0									00000000B
TH1	Timer High 1									00000000B
AUXR	Auxiliary	T0X12	T1X12	URM0X6	EADCI	ESPI	ENLVFI			000000xxB
P1	Port 1									11111111B
P1M0	Port1 configure register 0									00000000B
P1M1	Port1 configure register 1									00000000B
SCON	Serial Control	SM0	SM1	SM2	REN	TB8	RB8	TI	RI	00000000B
SBUF	Serial Buffer									xxxxxxxxB
IE	Interrupt Enable	EA	EPCA	ESPI	ES	ET1	EX1	ET0	EX0	00000000B
SADDR	Slave Address									00000000B
P3	Port 3	P3.7	-	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0	11111111B
P3M0	Port3 configure register 0									00000000B
P3M1	Port3 configure register 1									00000000B
IPH	Interrupt Priority High	-	PPCAH	PSPIH	PSH	PT1H	PX1H	PT0H	PX0H	x0000000B
IP	Interrupt Priority Low	-	PPCA	PSPI	PS	PT1	PX1	PT0	PX0	x0000000B
SADEN	Slave Address Mask									00000000B
ADCTL	ADC control register	ADCON	SPEED1	SPEED0	ADCI	ADCS	CHS2	CHS1	CHS0	00000000B
ADCV	ADC result									xxxxxxxxB
PCON2	Power Control register 2						CKS2	CKS1	CKS0	xxxxx000B
PSW	Program Status Word	CY	AC	F0	RS1	RS0	OV	-	P	00000000B
CCON	PCA control register	CF	CR	-	-	-	-	CCF1	CCF0	00xxxx00B
CMOD	PCA mode register	CIDL	-	-	-	-	CPS1	CPS0	ECF	0xxxx000B
CCAPM0	PCA module0 mode	-	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0	x0000000B
CCAPM1	PCA module1 mode	-	ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1	x0000000B
WDTCR	Watch-dog-timer Control	WRF	-	ENW	CLW	WIDL	PS2	PS1	PS0	0x000000B
IFD	ISP Flash data									11111111B
IFADRH	ISP Flash Address High									00000000B
IFADRL	ISP Flash Address Low									00000000B
IFMT	ISP Mode Table	-	-	-	-	-	MS2	MS1	MS0	xxxxx000B
SCMD	ISP Serial Command									xxxxxxxxB
ISPCR	ISP Control Register	ISPEN	BS	SWRST	-	-		WAIT		00001000B
CL	Low byte of PCA timer									00000000B
CCAP0L	Low byte of PCA module0 Compare/Capture register									00000000B
CCAP1L	Low byte of PCA module1 Compare/Capture register									00000000B
B	B Register									00000000B
PCAPWM0	PWM mode auxiliary 0							EPC0H	EPC0L	xxxxxx00B
PCAPWM1	PWM mode auxiliary 1							EPC1H	EPC1L	xxxxxx00B
CH	High byte of PCA timer									00000000B
CCAP0H	High byte of PCA module0 Compare/Capture register									00000000B
CCAP1H	High byte of PCA module1 Compare/Capture register									00000000B
ACC	Accumulator									00000000B

Memory

Organization



Address Space for MPC82E52A RAM



Address Space for MPC82E52A embedded Flash memory

RAM

There are 256 bytes RAM built in MPC82E52A.

The user can visit the leading 128-byte RAM via direct addressing instructions, we name those RAM as *direct RAM* that occupies address space 00h to 7Fh.

Followed 128-byte RAM can be visited via indirect addressing instructions, we name those RAM as *indirect RAM* that occupied address space 80h to *FF*h.

Since the MPC82E52A has 256 bytes RAM only, any instruction to access RAM addressing over FF_H is inhibited. Furthermore, since the MPC82E52A is lack of PORT0 and PORT2 to address chip-external memory, so all MOVX instructions are inhibited.

Nonvolatile Registers:

There are four Nonvolatile Registers named **OR0**, **OR1**, **OR2**, and **OR3** individually. They are designed to configure the MPC82E52A, say to decide to use internal RC oscillator or use crystal oscillator as oscillating source, or allocate the built-in flash for application program, application data and In-System-Program code.

Generally, the only way to program those four nonvolatile registers is making use of a popular NVM writer, say Hi-Lo System All-11, Leaper-48 and Megawin-Provided MCU writer. The user's program and the ISP program never can change those option registers.

NVM register: OR0 *(*Option Register 0*):*

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
LVFWP	**ENLVR**	**ISPAS1**	**ISPAS0**	**HWBS**	*reserved**1***	**SB**	**LOCK**

LVFWP := Low-Voltage-Flag-Write-Protecting bit.

0 :=
inhibit the flash read/write action via ISP/IAP mechanism while the power supply drops under a specific voltage level. Typically, the Low-Voltage is identified to 3.7V associated with 12MHz oscillator.

1 := (default)
No inhibition on the flash-writing action.

ENLVR := Enable-Low-Voltage-Reset

0 :=
Clearing the bit will reset the device while the power supply drops under a specific voltage level. Typically, the Low-Voltage is identified to 3.7V associated with 12MHz oscillator.

1 := (default)
Setting the bit implies never reset the device in spite of voltage dropping.

{**ISPAS1**, **ISPAS0**} := ISP-Address-Start

{0,0} :=

Set the ISP start address 1400_H. (ISP code could take 3K bytes)

{0,1} :=

Set the ISP start address 1800_H. (ISP code could take 2K bytes)

{1,0} :=

Set the ISP start address $1C00_H$. (ISP code could take 1K bytes)

{1,1} := (default)

Express no ISP code.

HWBS := HardWare-Boot-Selector

0 :=

Clearing the bit is to configure the device to boot from ISP program after power-up.

1 := (default)

Setting the bit is to configure the device to boot normally from user's application program after power-up.

In fact, the boot entrance is determined by register **SWBS** *from SFR* **ISPCR** *ignoring of the boot comes from RST-pin press, software-trigger, or power-up. However, if a boot happens and that boot comes from power-up action, the device will first load the complement of the* **HWBS** *to* **SWBS***, and after that decides the boot entrance according to the state of bit* **SWBS***, so the HWBS named HardWare Boot Selector. It influence on power-up boot, but does not influence on boot from RST-pin or software-trigger.*

*reserved**1*** := The bit is reserved for further user, and should be left set.

The user must not clear the bit, or there could be inadvertent effect impacted on the device.

SB := Used to decide if the program code will be ScramBled while it is dumped.

0 :=

Code dump from Writer is scrambled.

1 := (default)

Code dump from Writer is transparent.

LOCK := Used to decide if the program code will be LOCKed against the popular writer.

0 :=

Code dumping from Writer is locked.

1 := (default)

Permit code dumping from general Writers.

NVM register: OR1 *(*Option Register 1*):*

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
							-

OR1[7:1] := Used to set the boundary of IAP memory

The user's application program can change only the IAP flash memory, while neither of AP flash memory itself and the ISP flash memory. The IAP memory is defined between address scope **OR1**[7:1]*512 and ISP-Address-Start. Setting the OR1[7:1] 1111111_B means no IAP memory.

NVM register: OR2 *(*Option Register 2*):*

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
*reserved**1***	**OSCDN**	**-**	*reserved**1***	**-**	*reserved**1***	**ENROSC**	*reserved**1***

*reserved**1*** := The bit is reserved for further user, and should be left set.

The user must not clear the bit, or there could be inadvertent effect impacted on the device.

OSCDN := Used to adjust the behavior of crystal oscillator.

0 :=

The current gain of crystal oscillator amplifier is reduced. It will bring help to EMI reducing and improve the power consumption. Dealing with application does not need high frequency clock (under 12MHz). It is recommended to do so.

1 := (default)

The current gain of crystal oscillator is enough for oscillator to start oscillating up to 40MHz.

ENROSC := Used to determined if to enable the built-in RC oscillator.

0 :=

Clearing the bit will enable the built-in RC oscillator, and set that oscillator as the oscillating source

1 := (default)

Setting the bit means to disable the built-in RC oscillator.

NVM register: OR3 *(*Option Register 3*):*

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
*reserved**1***	*reserved**1***	**HWENW**	**-**	**HWWIDL**	**HWPS2**	**HWPS1**	**HWPS0**

*reserved**1*** := The bit is reserved for further user, and should be left set.

The user must not clear the bit, or there could be inadvertent effect impacted on the device.

HWENW := HardWare ENable Watch-dog-timer

0 :=

Clearing the bit will automatically enable the watch-dog-timer after power-up immediately. If the bit is set to *0*, also those other bits **HWWIDL**, **HWPS2**, **HWPS1** and **HWPS0** will be loaded Into SFR WDTCR after power-up.

1 := (default)

Setting the bit means to disable the built-in RC oscillator.

HWWIDL := HardWare enables reset from Watch-dog-timer in spite of the MCU lies idle.

0 :=

Enable watch-dog-timer to keep working in spite of the MCU has been put into idle mode.

1 := (default)

Watch-dog-timer is also suspended while the MCU lies idle.

If the bit **HWENW** is left *1*, the bit **HWWIDL** makes no sense.

{ HWPS2, HWPS1, HWPS0} := HardWare Watch-dog-timer Pre-Scalar

If the bit **HWENW** is cleared to *0*, those bits will be loaded into SFR **WDTCR** after power-up. Those three bits set the pre-scalar of the watch-dog-timer.

If the bit **HWENW** is left *1*, those three bits makes no sense.

{0,0,0} :=

The frequency of the clock source for the watch-dog-timer is divided by 2.

{0,0,1} :=

The frequency of the clock source for the watch-dog-timer is divided by 4.

{0,1,0} :=

The frequency of the clock source for the watch-dog-timer is divided by 8.

{0,1,1} :=

The frequency of the clock source for the watch-dog-timer is divided by 16

{1,0,0} :=

The frequency of the clock source for the watch-dog-timer is divided by 32

{1,0,1} :=

The frequency of the clock source for the watch-dog-timer is divided by 64

{1,1,0} :=

The frequency of the clock source for the watch-dog-timer is divided by 128

{1,1,1} :=

The frequency of the clock source for the watch-dog-timer is divided by 256

Embedded Flash

There is totally 8K byte flash embedded in the MPC82E52A.

The user can configure the whole flash to store his application program, or he can configure the flash for both storage of application(AP) program and In-System-Program(ISP) code, even he can configure the flash for storage of AP, ISP, and In-Application-Program(IAP) memory.

While the program counter of MPC82E52A is spanning over $1FFF_H$, the device will do nothing.

The user can develop his ISP program and put it into the embedded flash that addressed from 1400_H, 1800_H, or $1C00_H$, meanwhile configure **OR0[5:4]**, and set **OR0[3]** to 0, so to direct the device to boot from his ISP code.

If there is requirement from the user's application program to store nonvolatile parameters, the user can allocate part of the embedded flash as IAP memory by configure **OR1[7:1]**.

Functional Description

I/O Port Configuration

All 15 port pins on MPC82E52A may be independently configured to one of four modes : quasi-bidirectional(standard 8051 port output), push-pull output, open-drain output or input-only. All port pins default to quasi-bidirectional after reset. Each port pin has a Schmitt-triggered input for improved input noise rejection. During power-down, all the schmitt-triggered inputs are disabled with the exception of P3.2 and P3.3, which may be used to wake-up the device. Therefore P3.2 and P3.3 should not be left floating during power-down.

There are several special function registers designed to configure those I/O ports.

SFR: **P1M0**

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
P1M07	**P1M06**	**P1M05**	**P1M04**	**P1M03**	**P1M02**	**P1M01**	**P1M00**

SFR: **P1M1**

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
P1M17	**P1M16**	**P1M15**	**P1M14**	**P1M13**	**P1M12**	**P1M11**	**P1M10**

SFR: **P3M0**

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
P3M07	**P3M06**	**P3M05**	**P3M04**	**P3M03**	**P3M02**	**P3M01**	**P3M00**

SFR: **P3M1**

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
P3M17	**P3M16**	**P3M15**	**P3M14**	**P3M13**	**P3M12**	**P3M11**	**P3M10**

Configuration of I/O port

P*x*M0*n*	P*x*M1*n*	Port Mode
0	*0*	Quasi-bidirectional
0	*1*	Push-Pull output
1	*0*	Input Only (High-impedance)
1	*1*	Open-Drain Output

(***x*** = *1* or *3* ***n*** = *7, 6, 5, 4, 3, 2, 1* or *0*)

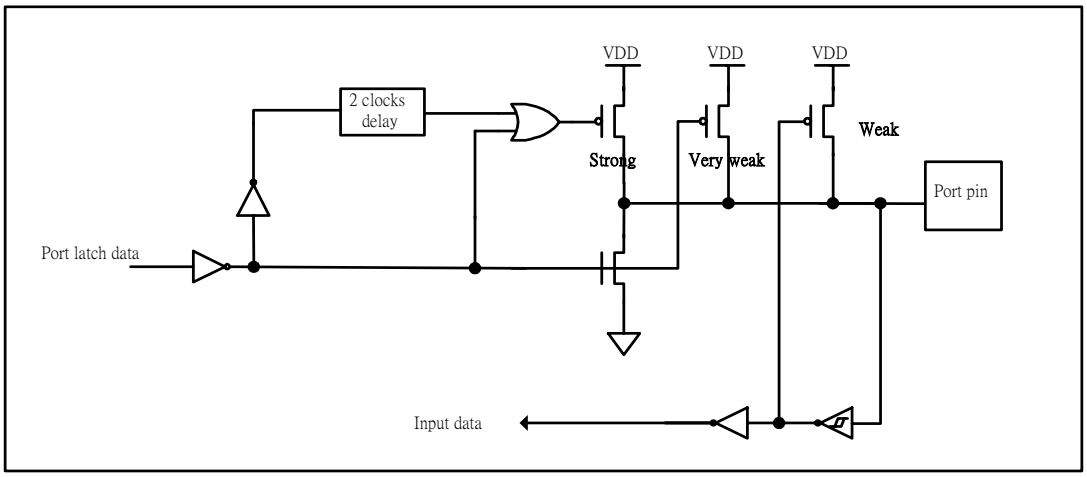
Quasi-bidirectional Mode

Port pins in quasi-bidirectional output mode function similar to the standard 8051 port pins. A quasi-bidirectional port can be used as an input and output without the need to reconfigure the port. This is possible because when the port outputs a logic high, it is weakly driven, allowing an external device to pull the pin low. When the pin outputs low, it is driven strongly and able to sink a large current. There are three pull-up transistors in the quasi-bidirectional output that serve different purposes.

One of these pull-ups, called the “very weak” pull-up, is turned on whenever the port register for the pin contains a logic “1”. This very weak pull-up sources a very small current that will pull the pin high if it is left floating.

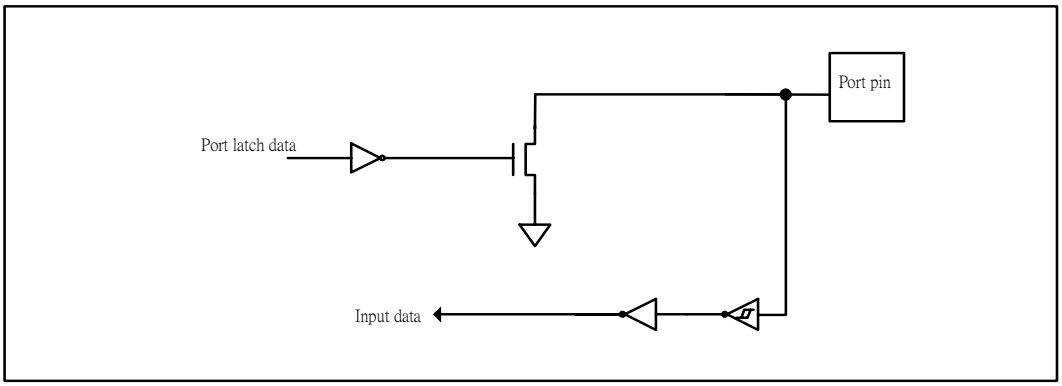
A second pull-up, called the “weak” pull-up, is turned on when the port register for the pin contains a logic “1” and the pin itself is also at a logic “1” level. This pull-up provides the primary source current for a quasi-bidirectional pin that is outputting a 1. If this pin is pulled low by the external device, this weak pull-up turns off, and only the very weak pull-up remains on. In order to pull the pin low under these conditions, the external device has to sink enough current to over-power the weak pull-up and pull the port pin below its input threshold voltage.

The third pull-up is referred to as the “strong” pull-up. This pull-up is used to speed up low-to-high transitions on a quasi-bidirectional port pin when the port register changes from a logic “0” to a logic “1”. When this occurs, the strong pull-up turns on for two CPU clocks, quickly pulling the port pin high.



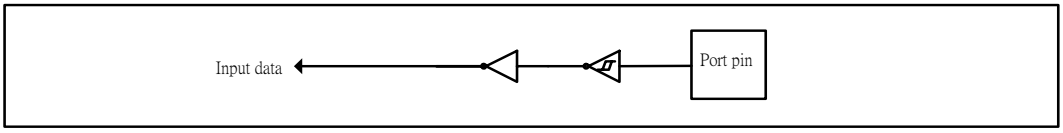
Open-drain Output

The open-drain output configuration turns off all pull-ups and only drives the pull-down transistor of the port pin when the port register contains a logic “0”. To use this configuration in application, a port pin must have an external pull-up, typically tied to VDD. The input path of the port pin in this configuration is the same as quasi-bidirection mode.



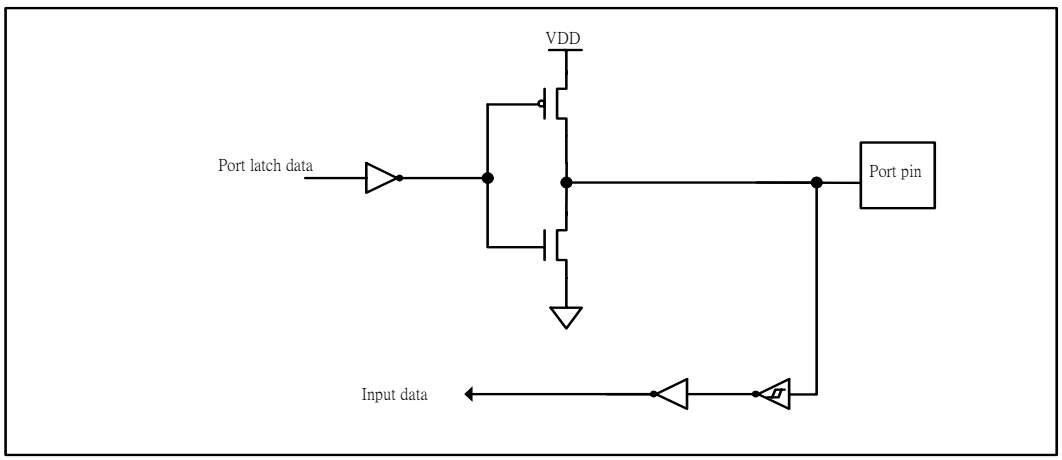
Input-only Mode

The input-only configuration is a Schmitt-triggered input without any pull-up resistors on the pin.



Push-pull Output

The push-pull output configuration has the same pull-down structure as both the open-drain and the quasi-bidirectional output modes, but provides a continuous strong pull-up when the port register contains a logic “1”. The push-pull mode may be used when more source current is needed from a port output.



Timer/Counter

MPC82E52A has two 16-bit timers, and they are named **T0** and **T1**. Each of them can also be used as a general event counter, which counts the transition from 1 to 0.

Since the MPC82E52A is a RISC-like MCU which execute faster than traditional 80C51 MCU from other providers. Based on consideration of compatibility with traditional 80C51 MCUs, the frequency of the clock source for **T0** and **T1** is designed to be selectable between oscillator frequency divided-by-12(default) or oscillator frequency.

The user can configure T0/T1 to work under mode-0, mode-1, mode-2 and mode-3. It is fully the same to a traditional 80C51 MCU.

There are two SFR designed to configure timers **T0** and **T1**. They are **TMOD**, **TCON**.

The user also should take a glace of SFR **AUXR** which decide the frequency of the clock source driving the **T0** and **T1**.

SFR: **TMOD**

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
GATE	**C//T**	**M1**	**M0**	**GATE**	**C//T**	**M1**	**M0**
(for timer1 use)				(for timer0 use)			

GATE := Gating control

0 :=
Timer *x* is enabled whenever "TR*x*" control bit is set.

1 := (default)
Timer/Counter *x* is enabled only while "/INT*x*" pin is high and **"**TR*x*" control bit is set.

C//T := Timer or Counter function selector. *0*: =timer, *1*: =counter

0 :=
Configure **T*x*** as Timer use

1 := (default)
Configure **T*x*** as Counter use

{M1, M0}: mode select

{0, 0} : =
Configure **T*x*** as 13-bit timer/counter

{0, 1} : =
Configure **T*x*** as 16-bit timer/counter

{1, 0} : =
Configure **T*x*** as 8-bit timer/counter with automatic reload capability

{1, 1} : =
for **T0,** set **TL0** as 8-bit timer/counter, **TH0** is locked into 8-bit timer
for **T1,** set Timer/Counter1 Stopped

SFR: **TCON**

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
TF1	**TR1**	**TF0**	**TR0**	**IE1**	**IT1**	**IE0**	**IT0**

TF1: = Timer1 overflow flag.
This bit is automatically set by hardware on **T1** overflow, and will be automatically cleared by hardware when the processor vectors to the interrupt routine.

TR1: = Timer1 run control bit.
0 :=
Stop **T1** counting
1 := (default)
Start **T1** counting

TF0: = Timer0 overflow flag.
This bit is automatically set by hardware on **T0** overflow, and will be automatically cleared by hardware when the processor vectors to the interrupt routine.

TR0: = Timer0 run control bit.
0 :=
Stop **T0** counting
1 := (default)
Start **T0** counting

IE1: = External Interrupt-1 flag.
This bit is automatically set by hardware on interrupt from the external interrupt-1, and will be automatically cleared by hardware when the processor vectors to the interrupt routine.

IT1: = Interrupt-1 type control bit.
0 :=
Set the interrupt-1 triggered by low duty from pin EX1
1 := (default)
Set the interrupt-1 triggered by negative rising edge from pin EX1

IE0: = External Interrupt-0 flag.
This bit is automatically set by hardware on interrupt from the external interrupt-0, and will be automatically cleared by hardware when the processor vectors to the interrupt routine.

IT0: = Interrupt-0 type control bit.
0 :=
Set the interrupt-0 triggered by low duty from pin EX1
1 := (default)
Set the interrupt-0 triggered by negative rising edge from pin EX1

SFR: **AUXR**

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
T0X12	**T1X12**	**URM0X6**	**EADCI**	**ESPI**	**ENLVFI**	**-**	**-**

T0X12: = **T0** clock source selector

0 := (default)

Set the frequency of the clock source for **T0** as the oscillator frequency divided-by-12.
It will compatible to the traditional 80C51 MCU.

1 :=

Set the frequency of the clock source for **T0** as the oscillator frequency.
It will drive the **T0** faster than a traditional 80C51 MCU.

T1X12: = **T1** clock source selector

0 := (default)

Set the frequency of the clock source for **T1** as the oscillator frequency divided-by-12.
It will compatible to the traditional 80C51 MCU.

1 :=

Set the frequency of the clock source for **T1** as the oscillator frequency.
It will drive the **T1** faster than a traditional 80C51 MCU.

URM0X6: = Baud rate selector of UART while it is working under Mode-0

0 := (default)

Set the baud rate of the UART functional block as oscillator frequency divided-by-12.
It will compatible to the traditional 80C51 MCU.

1 :=

Set the baud rate of the UART functional block as oscillator frequency divided-by-2.
It will transmit/receive data faster than a traditional 80C51 MCU.

EADCI: = Enable/Disable interrupt from A/D converter

0 := (default)

Inhibit the ADC functional block to generate interrupt to the MCU

1 :=

Enable the ADC functional block to generate interrupt to the MCU

ESPI: = Enable/Disable interrupt from Serial Peripheral Interface(SPI)

0 := (default)

Inhibit the SPI functional block to generate interrupt to the MCU

1 :=

Enable the SPI functional block to generate interrupt to the MCU

ENLVFI: = Enable/Disable interrupt from low-voltage sensor

0 := (default)

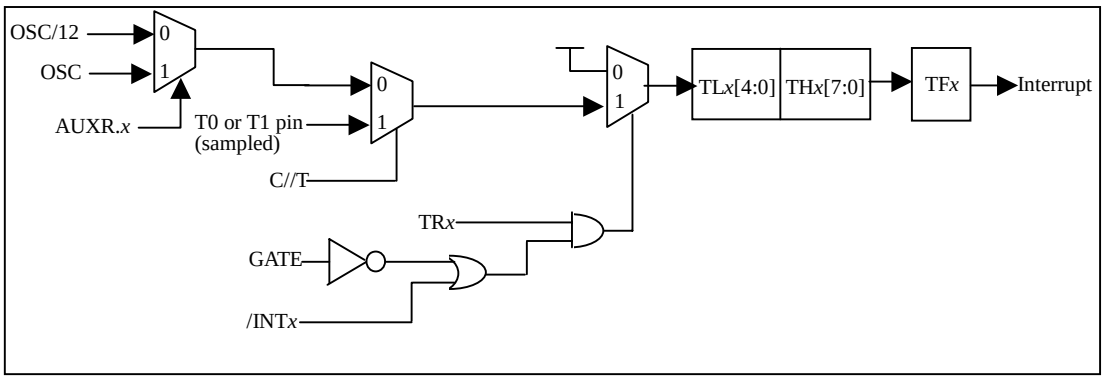
Inhibit the low-voltage sensor functional block to generate interrupt to the MCU

1 :=

Enable the low-voltage sensor functional block to generate interrupt to the MCU

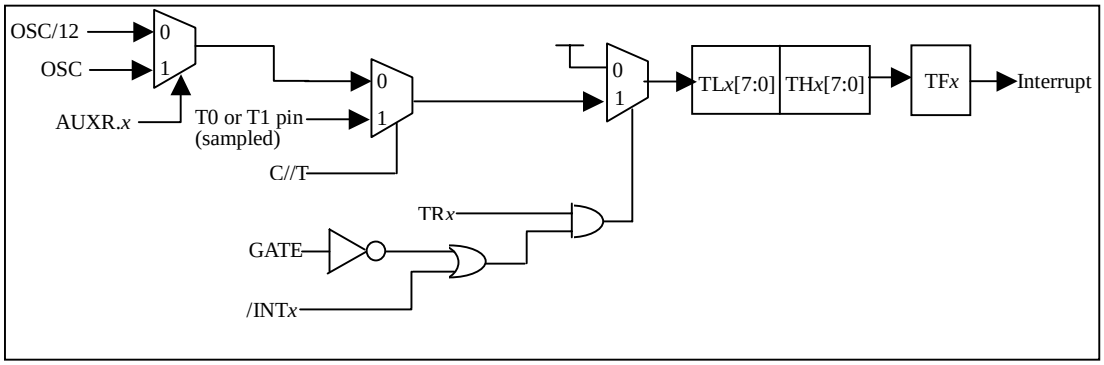
Mode 0

The timer register is configured as a 13-bit register. As the count rolls over from all 1s to all 0s, it sets the timer interrupt flag **TFx**. The counted input is enabled to the timer when **TRx** = 1 and either GATE=0 or INTx = 1. Mode 0 operation is the same for Timer0 and Timer1.



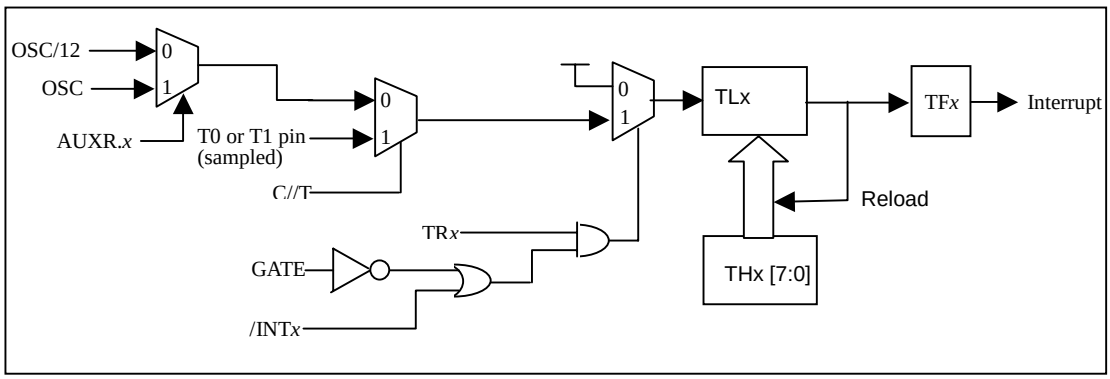
Mode 1

Mode1 is the same as Mode0, except that the timer register is being run with all 16 bits.



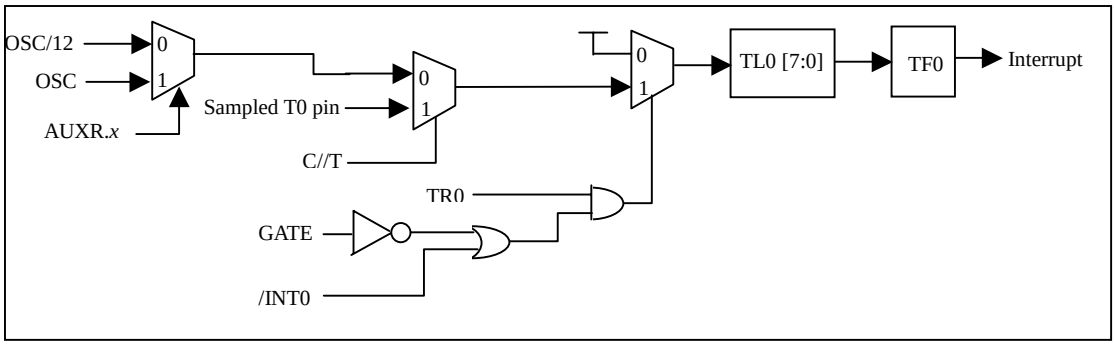
Mode 2

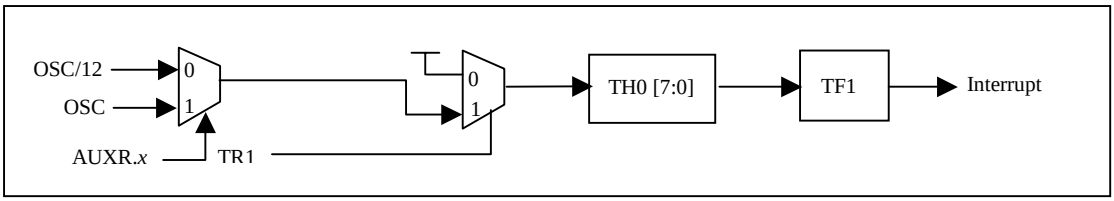
Mode 2 configures the timer register as an 8-bit counter (TL*x*) with automatic reload. Overflow from TL*x* does not only set TF*x*, but also reloads TL*x* with the content of TH*x*, which is determined by user's program. The reload leaves TH*x* unchanged. Mode 2 operation is the same for Timer0 and Timer1.



Mode 3

Timer1 in Mode3 simply holds its count, the effect is the same as setting TR1 = 1. Timer0 in Mode 3 enables TL0 and TH0 as two separate 8-bit counters. TL0 uses the Timer0 control bits such like C/T, GATE, TR0, INT0 and TF0. TH0 is locked into a timer function (can not be external event counter) and take over the use of TR1, TF1 from Timer1. TH0 now controls the Timer1 interrupt.





Interrupt

There are seven interrupt sources available in MPC82E52A. Each interrupt source can be individually enabled or disabled by setting or clearing a bit in the SFR named **IE**. This register also contains a global disable bit (**EA**), which can be cleared to disable all interrupts at once.

Each interrupt source has two corresponding bits to represent its priority. One is located in SFR named **IPH** and the other in **IP** register. Higher-priority interrupt will be not interrupted by lower-priority interrupt request. If two interrupt requests of different priority levels are received simultaneously, the request of higher priority is serviced. If interrupt requests of the same priority level are received simultaneously, an internal polling sequence determine which request is serviced. The following table shows the internal polling sequence in the same priority level and the interrupt vector address.

Source	Vector address	Priority within level
External interrupt 0	03H	1 (highest)
Timer 0	0BH	2
External interrupt 1	13H	3
Timer1	1BH	4
Serial Port	23H	5
SPI/ADC	2BH	6
PCA/LVF	33H	7

The external interrupt /INT0, /INT1 can each be either level-activated or transition-activated, depending on bits **IT0** and **IT1** in register **TCON**. The flags that actually generate these interrupts are bits **IE0** and **IE1** in **TCON**. When an external interrupt is generated, the flag that generated it is cleared by the hardware when the service routine is vectored to *only if the interrupt was transition –activated*, then the external requesting source is what controls the request flag, rather than the on-chip hardware.

The Timer0 and Timer1 interrupts are generated by TF0 and TF1, which are set by a rollover in their respective Timer/Counter registers in most cases. When a timer interrupt is generated, the flag that generated it is cleared by the on-chip hardware when the service routine is vectored to.

The serial port interrupt is generated by the logical OR of RI and TI. Neither of these flags is cleared by hardware when the service routine is vectored to. The service routine should poll RI and TI to determine which one to request service and it will be cleared by software.

The $2B_H$ interrupt is shared by the logical OR of SPI interrupt and ADC interrupt. Neither of these flags is cleared by hardware when the service routine is vectored to. The service routine should poll them to determine which one to request service and it will be cleared by software.

The 33_H interrupt is shared by the logical OR of PCA interrupt and LVD(Low-Voltage Detector) interrupt. Neither of these flags is cleared by hardware when the service routine is vectored to. The service routine should poll them to determine which one to request service and it will be cleared by software.

All of the bits that generate interrupts can be set or cleared by software, with the same result as though it had been set or cleared by hardware. In other words, interrupts can be generated or pending interrupts can be canceled in software.

The following content describes several SFR related to interrupt mechanism.

SFR: **IE**

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
EA	**EPCA**	**ESPI**	**ES**	**ET1**	**EX1**	**ET0**	**EX0**

EA := Global interrupt controller.
0 := (default)
Disable all interrupts
1 :=
Release interrupt control to all individual interrupt controllers.

EPCA := Interrupt controller of Programmable Counter Array(PCA) and Low-Voltage Detector
0 := (default)
Disable
1 :=
Enable

ESPI := Interrupt controller of Serial Peripheral Interface(SPI) and A/D Converter(ADC).
0 := (default)
Disable
1 :=
Enable

ES := Interrupt controller of Universal Asynchronous Receiver/Transmitter (UART).
0 := (default)
Disable
1 :=
Enable

ET1 := Interrupt controller of Timer-1 interrupt.
0 := (default)
Disable
1 :=
Enable

EX1 := Interrupt controller of external interrupt-1.
0 := (default)
Disable
1 :=
Enable

ET0 := Interrupt controller of Timer-0 interrupt.
0 := (default)
Disable
1 :=
Enable

EX0 := Interrupt controller of external interrupt-0.
0 := (default)
Disable
1 :=
Enable

SFR: **IP**

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
-	**PPCA**	**PSPI**	**PS**	**PT1**	**PX1**	**PT0**	**PX0**

PPCA := If set, Set priority for PCA /LVF interrupt higher
PSPI := If set, Set priority for SPI/ADC interrupt higher
PS := If set, Set priority for serial port interrupt higher(UART)
PT1 := If set, Set priority for timer1 interrupt higher
PX1 := If set, Set priority for external interrupt 1 higher
PT0 := If set, Set priority for timer0 interrupt higher
PX0 := If set, Set priority for external interrupt 0 higher

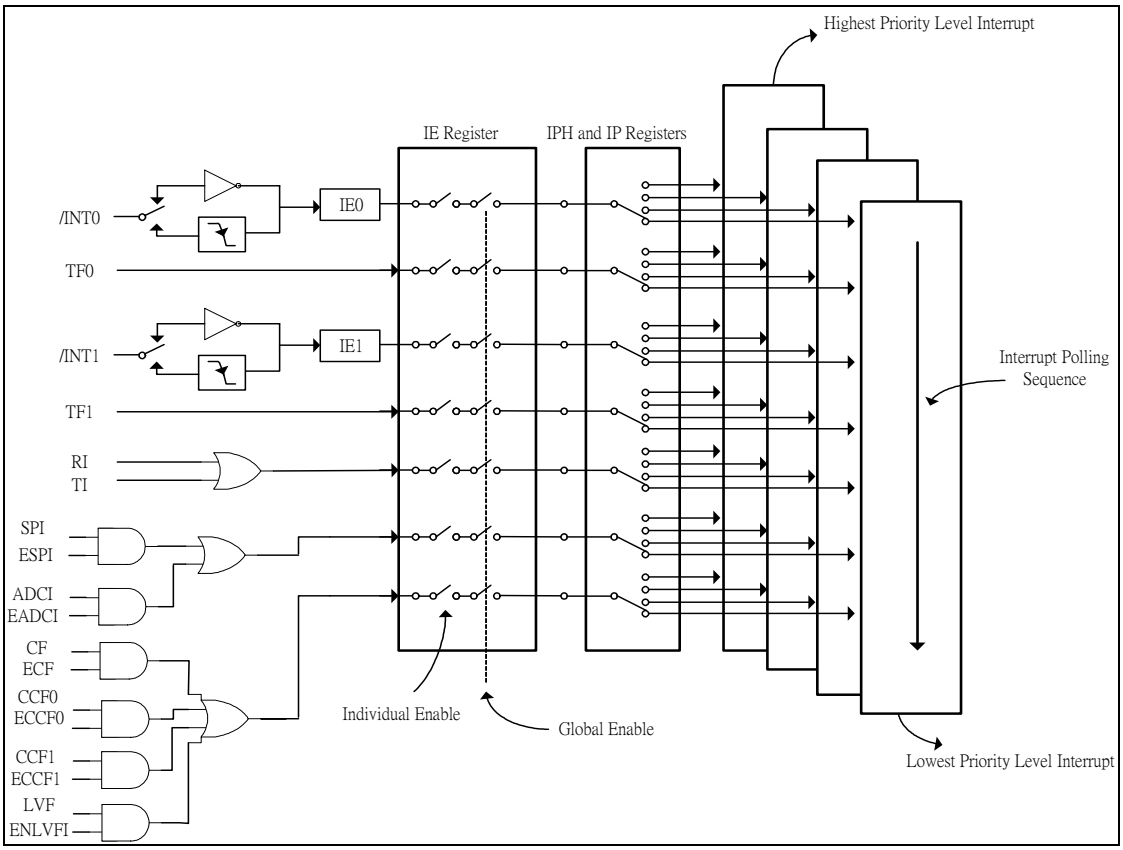
SFR: **IPH**

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
-	**PPCAH**	**PSPIH**	**PSH**	**PT1H**	**PX1H**	**PT0H**	**PX0H**

PPCAH := If set, Set priority for PCA /LVF interrupt higher
PSPIH := If set, Set priority for SPI/ADC interrupt higher
PSH := If set, Set priority for serial port interrupt higher(UART)
PT1H := If set, Set priority for timer1 interrupt higher
PX1H := If set, Set priority for external interrupt 1 higher
PT0H := If set, Set priority for timer0 interrupt higher
PX0H := If set, Set priority for external interrupt 0 higher

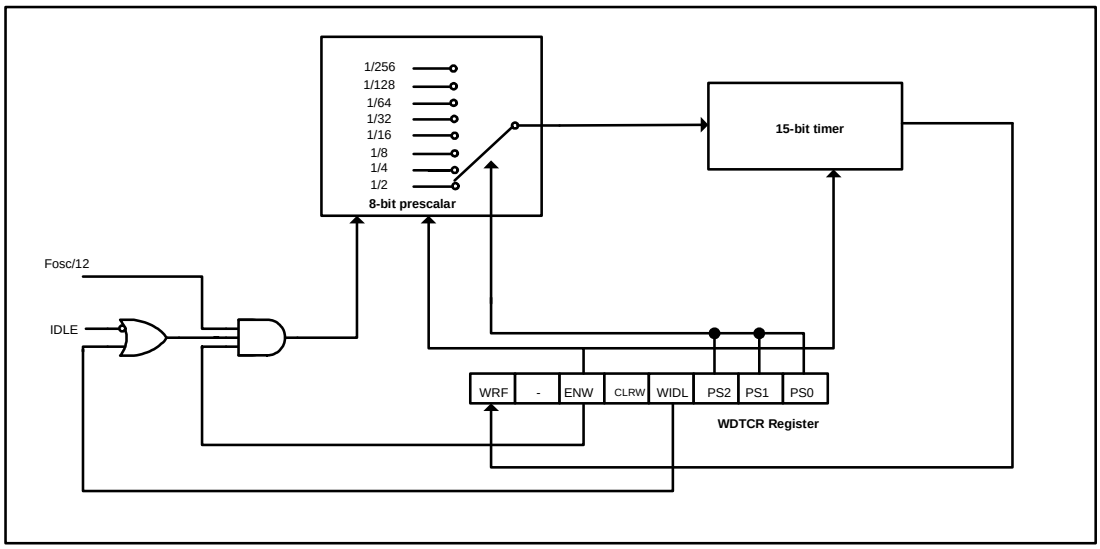
IP and **IPH** are combined to form 4-level priority interrupt as the following table.

{IPH.x , IP.x}	Priority Level
11	1 (highest)
10	2
01	3
00	4


Highest Priority Level Interrupt
IE Register
IPH and IP Registers
/INT0
IE0
TF0
/INT1
IE1
TF1
RI
TI
SPI
ESPI
ADCI
EADCI
CF
ECF
CCF0
ECCF0
CCF1
ECCF1
LVF
ENLVFI
Interrupt Polling Sequence
Individual Enable
Global Enable
Lowest Priority Level Interrupt

Watch Dog Timer

The watch dog timer in MPC82E52A consists of an 8-bit pre-scalar timer and a 15-bit timer. The timer is one-time enabled by setting ENW. Clearing ENW can not stop WDT counting. When the WDT is enabled, software should always reset the timer by writing 1 to CLRW bit before the WDT overflows. If MPC82E52A is out of control by any disturbance, that means the CPU can not run the software normally, then WDT may miss the "writing 1 to CLRW" and overflow will come. WDT overflow reset the CPU to restart. Associated with the WDTCR SFR, a NVM option register bytes name **OR3** are designed to enable WDT and initiate WDTCR with initial states. See Option Register description to know in more details.



To make good use of the watch-dog-timer, the user should take notice on SFR **WDTCR**.

SFR: **WDTCR**

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
WRF	**-**	**ENW**	**CLRW**	**WIDL**	**PS2**	**PS1**	**PS0**

WRF := When WDT overflows, this bit is set. It can be cleared by software.

ENW := Control bit to enable Watch-Dog-Timer
0 := (default)
Disable Watch Dog Timer
1 :=
Enable Watch Dog Timer start counting

CLRW := Set this bit to recount WDT. Hardware will automatically clear this bit.

WIDL := Behavior controller of the WDT while the device is put under idle

0 := (default)

Stop Watch Dog Timer counting

1 :=

Keep Watch Dog Timer counting(so further reset could happen)

{PS2, PS1, PS0}: selector of the WDT pre-scalar output.

{0, 0, 0}: = set the pre-scaling value 2

{0, 0, 1}: = set the pre-scaling value 4

{0, 1, 0}: = set the pre-scaling value 8

{0, 1, 1}: = set the pre-scaling value 16

{1, *0, 0}: = set the pre-scaling value 32

{1, 0, 1}: = set the pre-scaling value 64

{1, 1, 0}: = set the pre-scaling value 128

{1, 1, 1}: = set the pre-scaling value 256

Universal Asynchronous Serial Port (UART)

The serial port of MPC82E52A is duplex. It can transmit and receive simultaneously. The receiving and transmitting of the serial port share the same SFR **SBUF**, but actually there are two SBUF registers implemented in the chip, one is for transmitting and the other is for receiving. The serial port can be operated in 4 different modes.

Mode 0

Generally, this mode purely is used to extend the I/O features of this device.

Operating under this mode, the device receives the serial data or transmits the serial data via pin RXD, while there is a clock stream shifted via pin TXD which makes convenient for external synchronization. An 8-bit data is serially transmitted/received with LSB first. The baud rate is fixed at 1/12 the oscillator frequency. If **AUXR**.5(**URM0X6**) is set, the baud rate is 1/2 oscillator frequency.

Mode1

A 10-bits data is serially transmitted through pin TXD or received through pin RXD. The frame data includes a start bit (*0*), 8 data bits and a stop bit (*1*). After finishing a receiving, the device will keep the stop bit in **RB8** which from SRF **SCON**.

$$\text{Baud Rate (for Mode 1)} \quad \frac{2^{\text{SMOD}}}{32} \times \text{(Timer-1 overflow rate)}$$

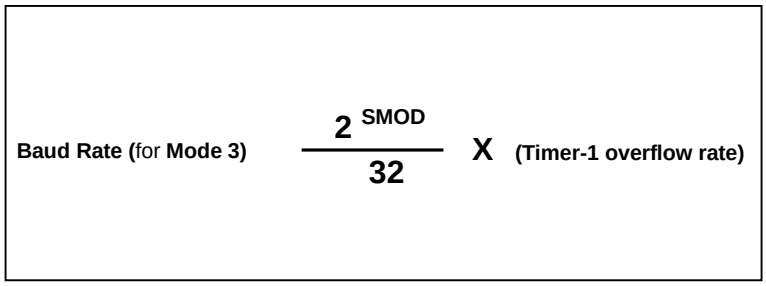
Mode2

An 11-bit data is serially transmitted through **TXD** or received through **RXD**. The frame data includes a start bit (*0*), 8 data bits, a programmable 9th bit and a stop bit (1). On transmit, the 9th data bit comes from **TB8** in SFR **SCON**. On receive, the 9th data bit goes into **RB8** in **SCON**. The baud rate is programmable, and permitted to be set either 1/32 or 1/64 the oscillator frequency.

$$\text{Baud Rate (for Mode 2)} = \frac{2^{\text{SMOD}}}{64} \times \text{Fosc}$$

Mode3

Mode 3 is the same as mode 2 except the baud rate is variable.



In all four modes, transmission is initiated by any instruction that uses SBUF as a destination register. Reception is initiated in mode 0 by the condition **RI** = *0* and **REN** = *1*. Reception is initiated in the other modes by the incoming start bit with *1*-to-*0* transition if **REN**=*1*.

There are several SFRs related to serial port configuration described as following.

SFR: **SCON**

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
SM0	**SM1**	**SM2**	**REN**	**TB8**	**RB8**	**TI**	**RI**

{ SM0, **SM1 }** := Used to set operating mode of the serial port.
- **{ *0, 0* }** := set the serial port operate under Mode 0
- **{ *0, 1* }** := set the serial port operate under Mode 1
- **{ *1, 0* }** := set the serial port operate under Mode 2
- **{ *1, 1* }** := set the serial port operate under Mode 3

SM2 := Enable the *automatic address recognition* feature in mode 2 and 3.
If **SM2**=*1*, **RI** will not be set unless the received 9th data bit is 1, indicating an address, and the received byte is a Given or Broadcast address. In mode1, if SM2=1 then RI will not be set unless a valid stop Bit was received, and the received byte is a Given or Broadcast address.

REN := Enable the serial port reception.
- **0** := (default)
  Disable Watch Dog Timer
- **1** :=
  Enable Watch Dog Timer start counting

TB8 := The 9th data bit, which will be transmitted in Mode 2 and Mode 3.

RB8 := In mode 2 and 3, the received 9th data bit will be put into this bit.

TI := Transmitting done flag. After a transmitting has been finished, the hardware will set this bit.

RI := Receive done flag. After reception has been finished, the hardware will set this bit.

SFR: **SBUF**

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
(data to be transmitted or received data)							

Automatic Address Recognition

There is an extra feature makes the device convenient to act as a master, which communicates to multiple slaves simultaneously. It is really *Automatic Address Recognition*.

There are two SFR **SADDR** and **SADEN** implemented in the device. The user can read or write both of them. Finally, the hardware will make use of these two SFR to "generate" a "compared byte". The formula specifies as following.

Bit[*i*] of **Compared Byte** = (**SADEN[** *i* **] == 1**)? **SADDR**[*i*] **:** ***x***

For example:

Set **SADDR** = 11000000b
Set **SADEN** = 11111101b
⇨ The achieved "Compared Byte" will be "110000x0" (*x* means don't care)

For another example:

Set **SADDR** = 11100000b
Set **SADEN** = 11111010b
⇨ The achieved "Compared Byte" will be "11100x0x"

After the generic "Compared Byte" has been worked out, the MPC82E52A will make use of this byte to determine how to set the bit **RI** in SFR **SCON**.

Normally, an UART will set bit **RI** whenever it has done a byte reception; but for the UART in the MPC82E52A, if the bit **SM2** is set, it will set **RI** according to the following formula.

RI = (SM2 == *1*) && (SBUF == *Compared Byte*) && (RB8 == *1*)

In other words, not all data reception will respond to RI, while specific data does.

By setting the SADDR and the SADEN, the user can filter out those data byte that he doesn't like to care. This feature brings great help to reduce software overhead.

The above feature adapts to the serial port when operated in Mode1, Mode2, and Mode3. Dealing with Mode 0, the user can ignore it.

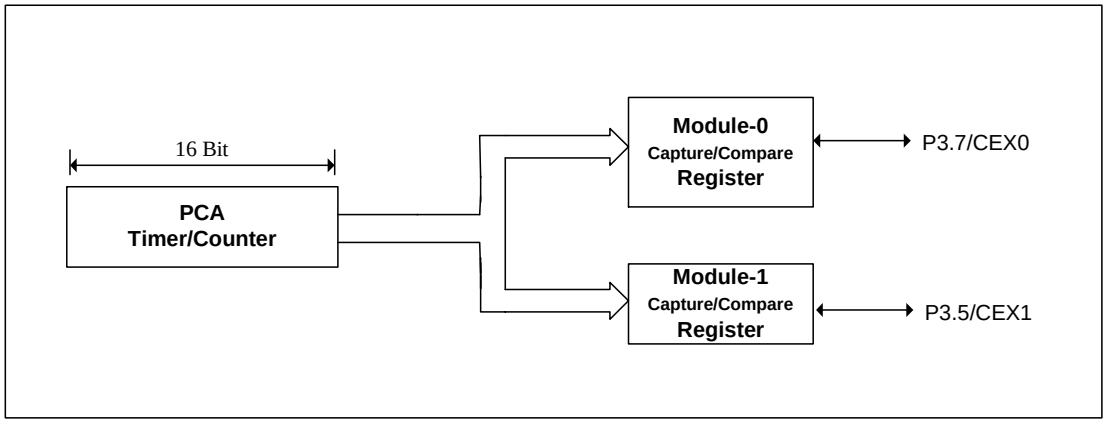
Programmable Counter Array(PCA)

The Programmable Counter Array is a special 16-bit Timer that has two 16-bit capture/compare modules associated with it. Each of the modules can be programmed to operate in one of four modes:

- rising and/or falling edge capture (calculator of duty length for high/low pulse)
- software timer
- high-speed output
- pulse width modulator

Each module has a pin associated with it in port 3. Module-0 is connected to pin P3.7, module-1 to pin P3.5.

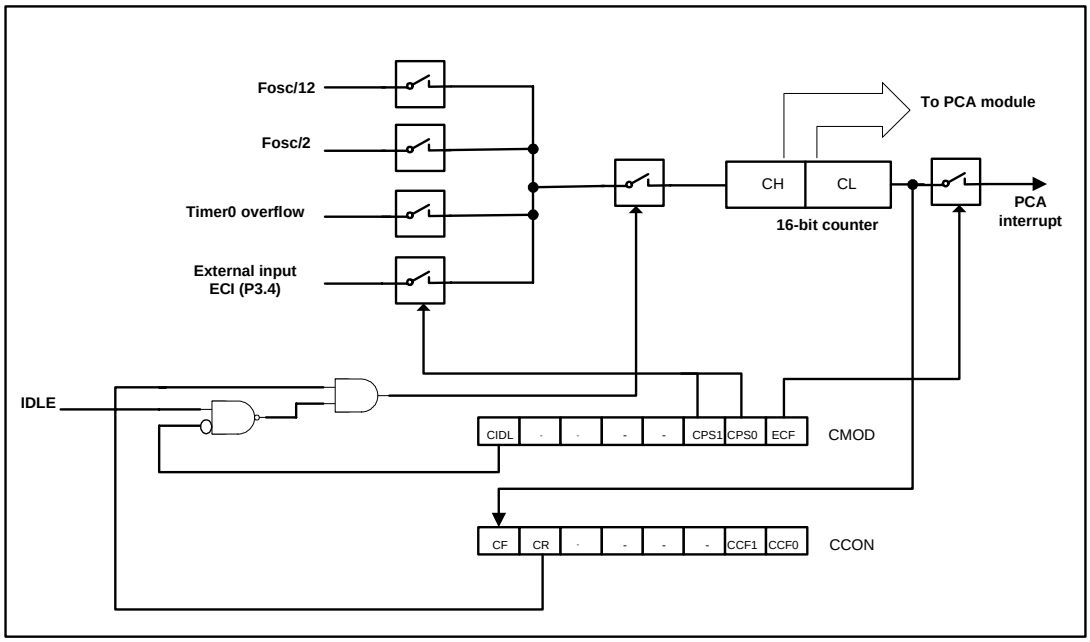
The PCA timer is a common time base for all two modules and can be programmed to run at 1/12 the oscillator frequency, 1/2 the oscillator frequency, the Timer-0 overflow or the input on pin ECI (P3.4). The timer count source is determined from **CPS1** and **CPS0** bits in the SFR **CMOD**.



Programmable Counter Array

In the **CMOD** SFR, there are two additional bits associated with the PCA. On of them is **CIDL** which determines if to stop the PCA while the MCU is put under idle, the other bit is **ECF** which controls if to pass the interrupt from PCA into the MCU.

The **CCON** SFR contains the run control bit for PCA and several flags for the PCA timer and each module. To start the PCA counting, the **CR** bit(**CCON**.6) must be set by software; oppositely clearing bit CR will shut off the PCA. There is a bit named **CF** in SFR **CCON**. The **CF** bit(**CCON**.7) will be set when the PCA timer overflows, and an interrupt will be generated if the **ECF**(**CMOD**.0) is set. The **CF** bit can only be cleared by software. There are two bits named **CCF0** and **CCF1** in SFR **CCON**. The **CCF0** and **CCF1** serve as flags for module-0 and module-1 respectively. They are set by hardware when either a match or a capture occurs. These flags also can only be cleared by software.



PCA Timer/Counter

SFR: **CMOD** *(PCA Counter MODe control register)*

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
CIDL	**-**	**-**	**-**	**-**	**CPS1**	**CPS0**	**ECF**

CIDL := Behavior control of the PCA.
0 := (default)
Disable counting of the PCA counter while the MCU is put under idle state.
1 :=
Enable counting of the PCA counter while the MCU is put under idle state.

{ CPS1, **CPS0 }** := Used to select the clocking source for PCA counter
{ *0, 0* } := set the frequency of the PCA counter clock source as oscillator's frequency over 12
{ *0, 1* } := set the frequency of the PCA counter clock source as oscillator's frequency over 2
{ *1, 0* } := set the PCA counter clock source as Timer-0 overflow
{ *1, 1* } := set the PCA counter clock source as pin ECI(pin P3.4)

ECF := Control bit of deciding if to pass interrupt from PCA timer overflow to the MCU
0 := (default)
Inhibit the interrupt from PCA timer to the MCU
1 :=
Permit the interrupt from PCA timer to the MCU

SFR: **CCON** *(PCA Counter CONfiguration register)*

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
CF	**CR**	**-**	**-**	**-**	**-**	**CCF1**	**CCF0**

CF := PCA Counter overflow Flag
This bit must be set by hardware itself. It can be cleared by software program.

CR := PCA Run control bit
0 := (default)
Disable counting of the PCA counter
1 :=
Start counting of the PCA counter

CCF1 := Module-1 interrupt Flag
This bit must be set by hardware itself when a match or capture from module-1 occurs. It can be cleared by software program.
A match means the value of the PCA counter equals the value of the Capture/Compare Register in the module-1.
A capture means a specific edge from CEX1 happens, so the Capture/Compare register latches the value of the PCA counter, and the CCF1 is set.

CCF0 := Module-0 interrupt Flag
This bit must be set by hardware itself when a match or capture from module-0 occurs. It can be cleared by software program.

Each module in the PCA has a special function register associated with it, **CCAPM0** for module0 and **CCAPM1** for module-1. The register contains the bits that control the mode in which each module will operate. The **ECCF*n*** bit controls if to pass the interrupt from **CCF*n*** flag in the **CCON** SFR to the MCU when a match or compare occurs in the associated module. **PWM*n*** enables the pulse width modulation mode. The **TOG*n*** bit when set causes the pin **CEX*n*** output associated with the module to toggle when there is a match between the PCA counter and the module's *Capture/Compare register*. The match bit(**MAT*n***) when set will cause the **CCF*n*** bit in the CCON register to be set when there is a match between the PCA counter and the module's *Capture/Compare register*.

The next two bits **CAPN*n*** and **CAPP*n*** determine the edge type that a capture input will be active on. The **CAPN*n*** bit enables the negative edge, and the **CAPP*n*** bit enables the positive edge. If both bits are set, both edges will be enabled and a capture will occur for either transition. The bit **ECOM*n*** when set enables the comparator function.

SFR: **CL** *(PCA Counter Low Byte)*

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0

SFR: **CH** *(PCA Counter High Byte)*

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0

SFR: **CCAP0L** *(Compare/CAPture register Low byte in Module-0)*

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
Low Byte of the Compare/ Capture register in PCA Module 0							

SFR: **CCAP0H** *(Compare/CAPture register High byte in Module-0)*

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
High Byte of the Compare/ Capture register in PCA Module 0							

SFR: **CCAP1L** *(Compare/CAPture register Low byte in Module-1)*

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
Low Byte of the Compare/ Capture register in PCA Module 1							

SFR: **CCAP1H** *(Compare/CAPture register High byte in Module-1)*

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
High Byte of the Compare/ Capture register in PCA Module 1							

SFR: **CCAPM0** *(Compare/CAPture register configuration register for module-0)*

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
-	**ECOM0**	**CAPP0**	**CAPN0**	**MAT0**	**TOG0**	**PWM0**	**ECCF0**

SFR: **CCAPM1** *(Compare/CAPture register configuration register for module-1)*

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
-	**ECOM1**	**CAPP1**	**CAPN1**	**MAT1**	**TOG1**	**PWM1**	**ECCF1**

ECOM*n* := used to determine if Enable COMparator

0 := (default)

Disable the comparator function

1 :=

Enable the comparator function

CCAPP*n* := configure the module-*n*'s register to latch the PCA counter on Positive edge of EXI*n* or not

0 := (default)

configure the module-*n*'s register not to latch the PCA counter on CEX*n* posedge.

1 :=

configure the module-*n*'s register to latch the PCA counter on CEX*n* posedge.

CCAPN*n* := configure the module-*n*'s register to latch the PCA counter on Negative edge of EXI*n* or not

0 := (default)

configure the module-*n*'s register not to latch the PCA counter on pin CEX*n* negedge.

1 :=

configure the module-*n*'s register to latch the PCA counter on pin CEX*n* negedge.

MAT*n* := used to determine if set the bit **CCF*n*** in SFR **CCON** while a match from module-*n* occurs..

0 := (default)

Don't set the bit **CCF*n*** while a match occurs between the PCA counter and module-*n*'s register.

1 :=

Set the bit **CCF*n*** while a match occurs between the PCA counter and module-*n*'s register.

TOG*n* := TOGgle the output pin

0 := (default)

Don't toggle the pin CEX*n* while a match occurs between the PCA counter and module-*n*'s register.

1 :=

Toggle the pin CEX*n* while a match occurs between the PCA counter and module-*n*'s register.

PWM*n* := TOGgle the output pin

0 := (default)

Inhibit the PWM functionality from module-n output to pin CEX*n*

1 :=
Enable the pin CEX*n* as the output of the PWM functionality from module-*n*

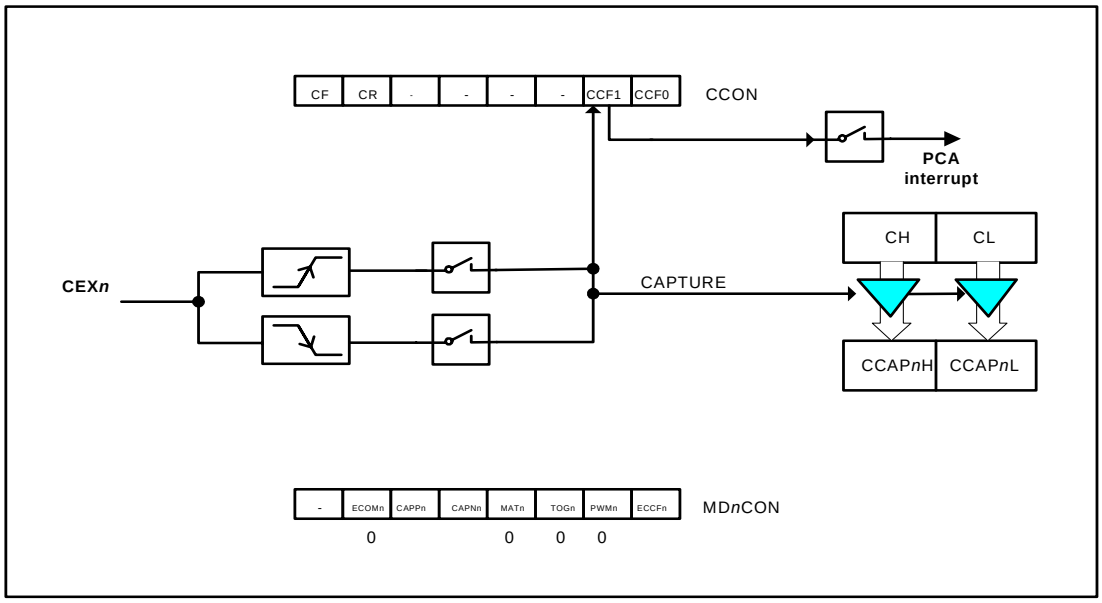
ECCF*n* := TOGgle the output pin
0 := (default)
Inhibit the interrupt(**CCF*n***) from module-*n* to the MCU
1 :=
Permit the interrupt(**CCF*n***) from module-*n* to the MCU

Configure PCA Module

ECOM*n*	CAPP*n*	CAPN*n*	MAT*n*	TOG*n*	PWM*n*	ECCF*n*	Module function
0	0	0	0	0	0	0	No operation
X	1	0	0	0	0	X	16-bit capture by a positive-edge trigger on CEX*n*
X	0	1	0	0	0	X	16-bit capture by a negative trigger on CEXn
X	1	1	0	0	0	X	16-bit capture by a transition on CEXn
1	0	0	1	0	0	X	16-bit Software Timer
1	0	0	1	1	0	X	16-bit High Speed Output
1	0	0	0	0	1	0	8-bit PWM

PCA Capture Mode

To use one of the PCA modules in the capture mode, one or both of bits **CAPP*n*** and **CAPN*n*** in SFR **CCAPM*n*** should be set. The external CEX*n* input for the module is sampled for a transition. When a valid transition occurs, the PCA hardware loads the value of the PCA counter register(**CH** and **CL**) into the module's capture registers(**CCAP*n*H** and **CCAP*n*L**). If the bit **CCF*n*** for the module in the SFR **CCON** and the bit **ECCF*n*** in the SFR **CCAPM*n*** are set then an interrupt will be generated.



PCA Capture Mode

16-bit Software Timer Mode

The PCA modules can be used as software timers by setting both the **ECOM*n*** and **MAT*n*** bits in the **CCAPM*n*** register. The PCA timer will be compared to the module's capture registers and when a match occurs an interrupt will be generated if the **CCF*n*** and **ECCF*n*** bits for the module are both set.

High Speed Output Mode

In this mode the CEX*n* output (port latch) associated with the PCA module will toggle each time a match occurs between the PCA counter and the module's capture registers. To activate this mode the **TOG*n***, **MAT*n***, and **ECOM*n*** bits in the SFR **CCAPM*n*** must be set.

Pulse Width Modulator Mode

All of the PCA modules can be used as PWM outputs. The frequency of the output depends on the PCA counter. All of the modules will have the same frequency of output because they all share the PCA counter. The duty cycle of each module is independently variable using the module's capture register **CCAP*n*L** and bit **PWMMSB** in SFR **PWM*n*MSB**. When the value of the SFR **CL** is less than the value in the module's { **PWMMSB**, **CCAP*n*L** }, the output will be low. When it is equal to or greater than , the output will be high. When **CL** overflows from FF_H to 00_H, { **PWMMSB**, **CCAP*n*L** } is reloaded with the value in {**RPWMMSB**, **CCAP*n*H**}. That allows smoothly updating the PWM duty without glitches. The bits **PWM*n*** and **ECOM*n*** bits in the **CCAPM*n*** must be set to enable the PWM mode.

SFR: **PWM0MSB** *(Module-0 PWM functionality control)*

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
-	-	-	-	-	-	**RPWMMSB**	**PWMMSB**

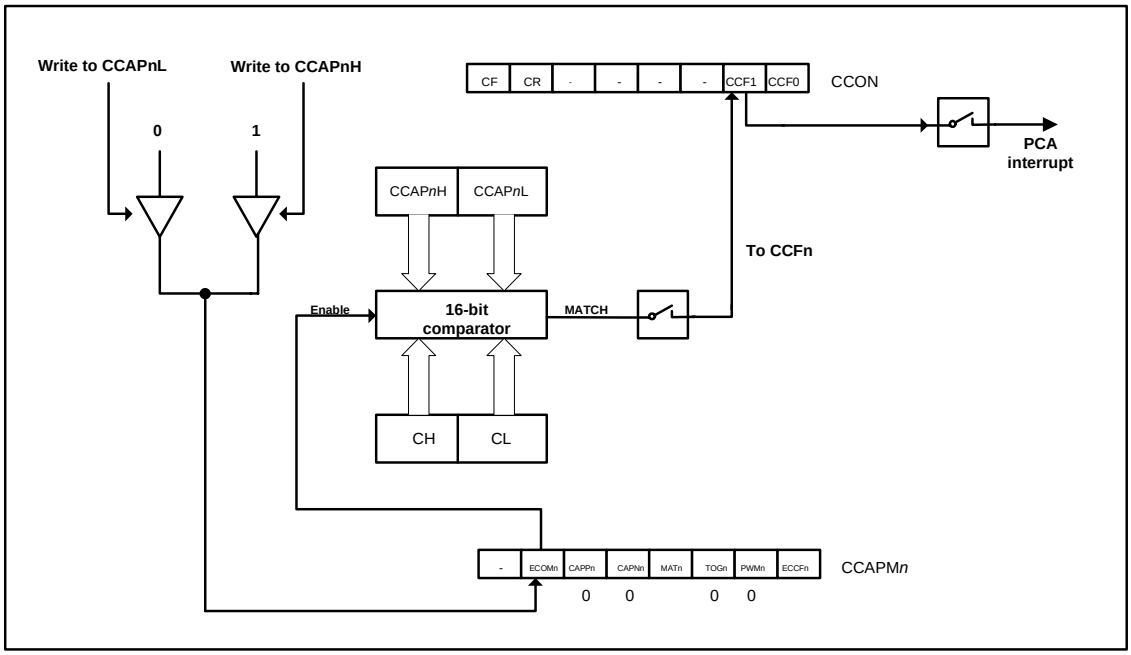
SFR**:** **PWM1MSB** *(Module-1 PWM functionality control)*

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
-	-	-	-	-	-	**RPWMMSB**	**PWMMSB**

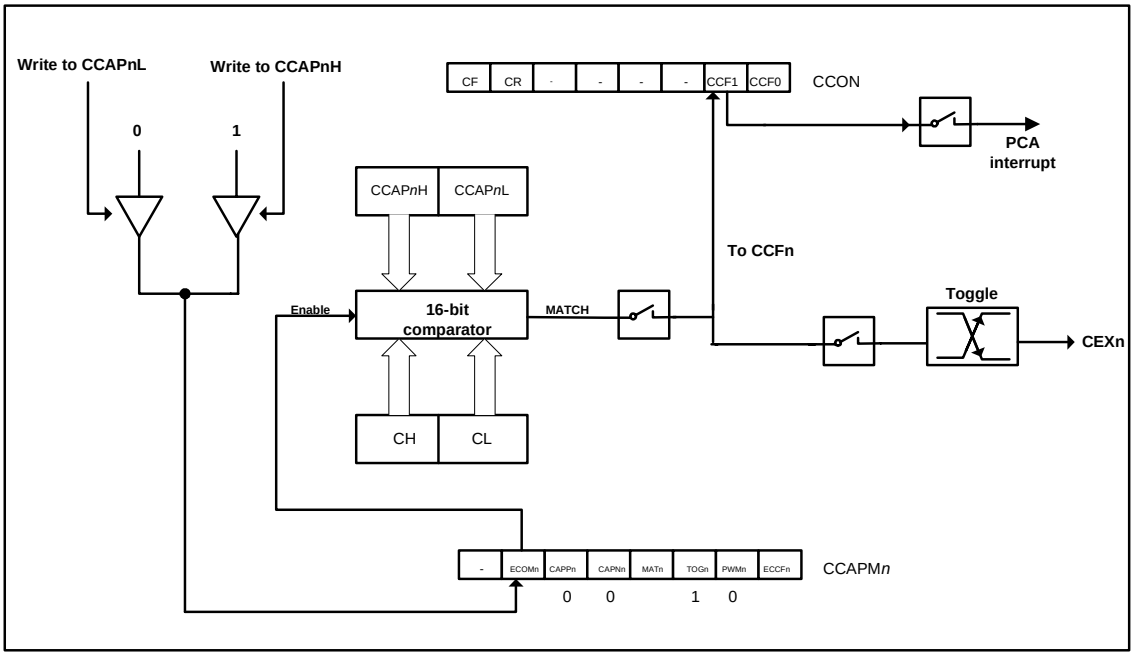
PWMMSB := concatenated with **CCAP*n*L**, used to control the duty of the PWM output
The bit **PWMMSB** is going to be combined with **CCAP*n*L** to form a 9-bit data which will be compared with PCA counter low byte **CL**, so to determine the duty of the module-*n*'s PWM output.

If {**CL**} < { **PWMMSB, CCAP*n*L} ,** PWM output LOW
else PWM output HIGH

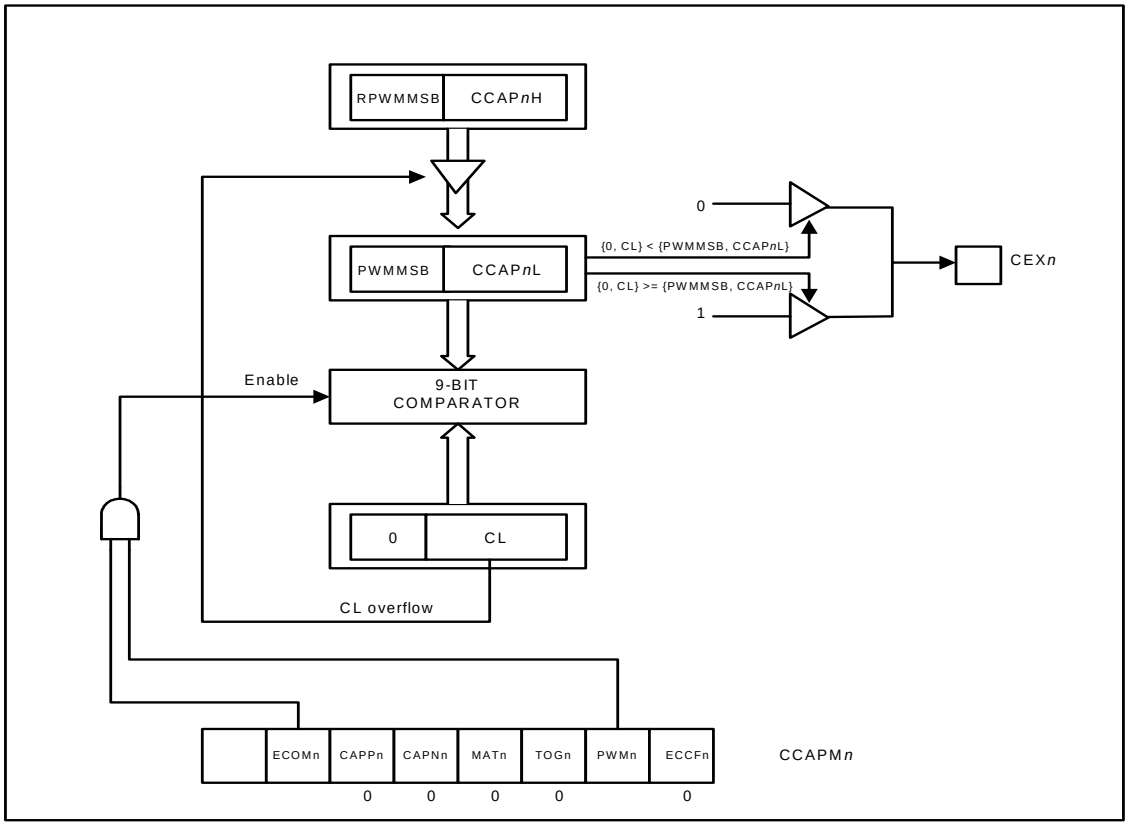
RPWMMSB := Reloaded value of **PWMMSB** while **CL** counts from FF_H to 00_H



PCA Software Timer Mode



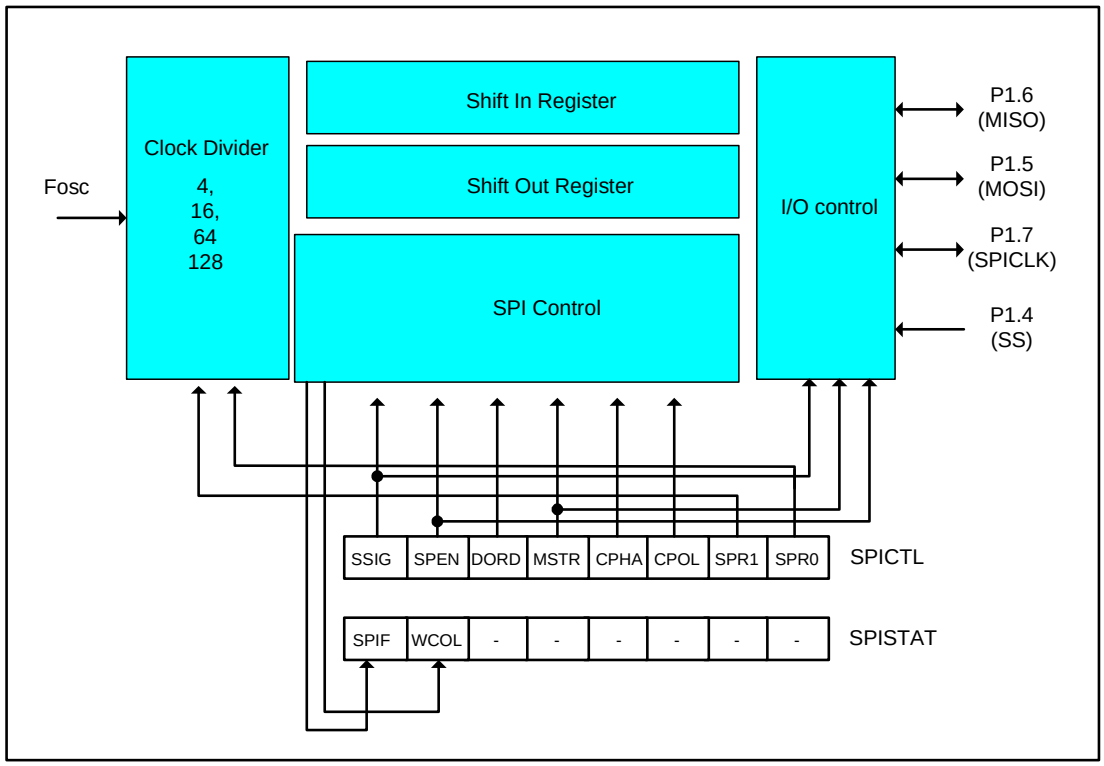
PCA High-Speed Ouput Mode


RPWMMSB
CCAPnH
PWMMSB
CCAPnL
0
1
{0, CL} < {PWMMSB, CCAPnL}
{0, CL} >= {PWMMSB, CCAPnL}
CEXn
Enable
9-BIT
COMPARATOR
0
CL
CL overflow
ECOMn
CAPPn
CAPNn
MATn
TOGn
PWMn
ECCFn
CCAPMn
0 0 0 0 0

PCA PWM mode

Serial Peripheral Interface(SPI)

The device provides another high-speed serial communication interface, the SPI interface. The SPI is a full-duplex, high-speed, synchronous communication bus with two operation modes: *Master* mode and *Slave* mode. Up to 3Mbit/s can be supported in either *Master* or *Slave* mode under the Fosc=12MHz. Two status flags are provided to signal the transfer completion and write-collision occurrence.



SPI block diagram

There are three pins implementing the SPI functionality, one of them is SPICLK(P1.7), next is MISO(P1.6), the other is MOSI(P1.5). An extra pin SS(P1.4) is designed to configure the SPI to run under *Master* or *Slave* mode. Data flows from master to slave via MOSI(Master Out Slave In) pin and flows from slave to master via MISO(Master In Slave Out) pin. The SPICLK plays as an output pin when the device works under *Master* mode, while as an input pin when the device works under *Slave* mode. If the SPI system is disabled, i.e. **SPEN**(**SPICTL**.6)=0, these pins are configured as general-purposed I/O port(P1.4 ~ P1.7).

Two devices with SPI interface communicate with each other via one synchronous clock signal , one input data signal, and one output data signal. There are two concerns the user could take care, one of them is latching data on the negative edge or positive edge of the clock signal which named *polarity*, the other is keeping the clock signal low or high while the device idle which named *phase*. Permuting those states from *polarity* and *phase*, there could be four modes formed, they are **SPI-MODE-0**, **SPI-MODE-1**, **SPI-MODE-2**, **SPI-MODE-3**.

Many device declares that they meet SPI mechanism, but few of them are adaptive to all four modes. The MPC82E52A is a device flexible to be configured to communicate to another device with **MODE-0**, **MODE-1**, **MODE-2** or **MODE-3** SPI, and play part of *Master* and *Slave*.

There is a SFR named SPICTL designed to configure the SPI behavior of the device.

SFR: **SPICTL** *(SPI ConTroLling register)*

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
SSIG	**SPEN**	**DORD**	**MSTR**	**CPOL**	**CPHA**	**SPR1**	**SPR0**

SSIG := used to determine if Ignore the pin SS
0 := (default)
Reserve the function of pin SS
1 :=
Ignore the SS pin function

SPEN := Enable the SPI
0 := (default)
Disable the SPI function. All related pins play as general-purposed I/O ports.
1 :=
Enable the SPI function.

DORD := Data ORDer
0 := (default)
Transmit/Receive the LSB of the data byte first.
1 :=
Transmit/Receive the MSB of the data byte first.

MSTR := Set to MaSTeR mode
0 := (default)
Set the SPI to play as *Slave* part.
1 :=
Set the SPI to play as *Master* part.

CPOL := Clock POLarity
0 := (default)
Set the SPICLK as LOW while the communication is kept idle. That implies the leading edge of the clock is the rising edge, and the trailing edge is the falling edge.
1 :=
Set the SPICLK as HIGH while the communication is kept idle. That implies the leading edge of the clock is the falling edge, and the trailing edge is the falling rising edge.

CPHA := Clock PHAse
0 := (default)
Data is driven when pin SS is low and changes on the trailing edge of SPICLK, and is sampled on the leading edge. (*This setting is only valid while SSIG==0*.)
1 :=
Data is driven on the leading edge of SPICLK, and is sampled on the trailing edge.

{SPR1, SPR0} := SPI clock Rate selector
{0,0} := (default)
Set the clock rate of the SPI as the frequency of the clock source over 4.
{0,1} :=
Set the clock rate of the SPI as the frequency of the clock source over 16.
{1,0} :=
Set the clock rate of the SPI as the frequency of the clock source over 64.

{1,1} :=
Set the clock rate of the SPI as the frequency of the clock source over 128.

There are two extra SFRs make relation with SPI application.

SFR: **SPIDAT** ***(SPI Data register)***

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
Data to be transmitted or Data received							

The SFR SPIDAT holds the data to be transmitted or the data received.

SFR: **SPISTAT** *(SPI STATe register)*

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
SPIF	**WCOL**	**-**	**-**	**-**	**-**	**-**	**-**

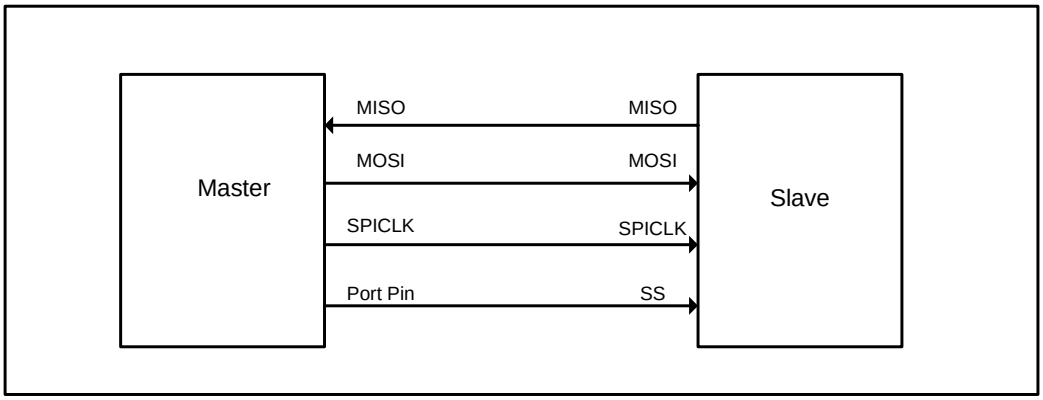
SPIF := SPI transfer completion flag.
When a serial transfer finishes, the SPIF bit is set and an interrupt is generated if both the ESPI(IE.6) bit and the EA(IE.7) bit are set. If SS is an input and is driven low when SPI is in master mode with SSIG=0, SPIF will also be set to signal the “mode change”. The SPIF is cleared in software by “writing 1 to this bit”.

WCOL := SPI Write COLlision flag
The WCOL bit is set if the SPI data register **SPIDAT** is written during a data transfer. The **WCOL** flag is cleared in software by “writing 1 to this bit”.

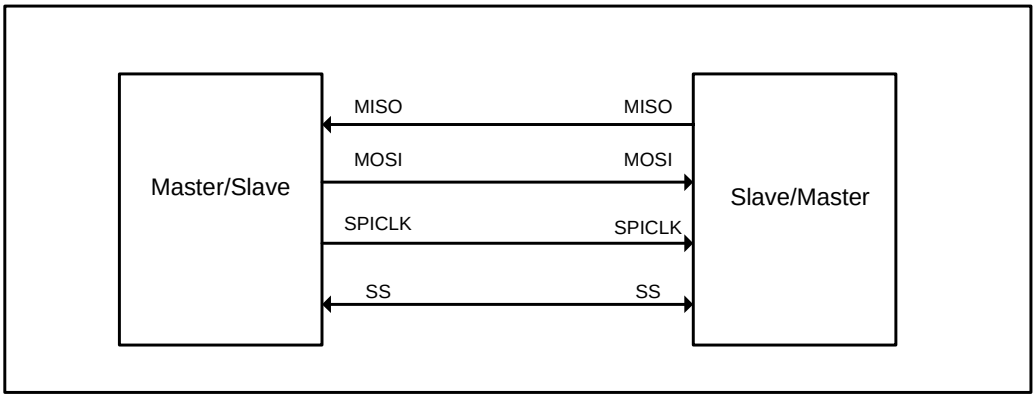
Configure the device to *Master*/*Slave* mode

SPEN	SSIG	SS	MSTR	Mode	MISO	MOSI	SPICLK	Remark
0	X	X	X	SPI disable	GPI/O	GPI/O	GPI/O	SPI is disabled.
1	0	0	0	Active Salve	output	input	input	Selected as slave
1	0	1	0	InActive Slave	Hi-Z	input	input	Not selected.
1	0	0	1→0	slave	output	input	input	Convert from Master to Slave
1	0	1	1	Master	input	output	output	SPICLK depends on CPOL
1	1	X	0	Slave	output	input	input	Slave
1	1	X	1	Master	input	output	output	Master

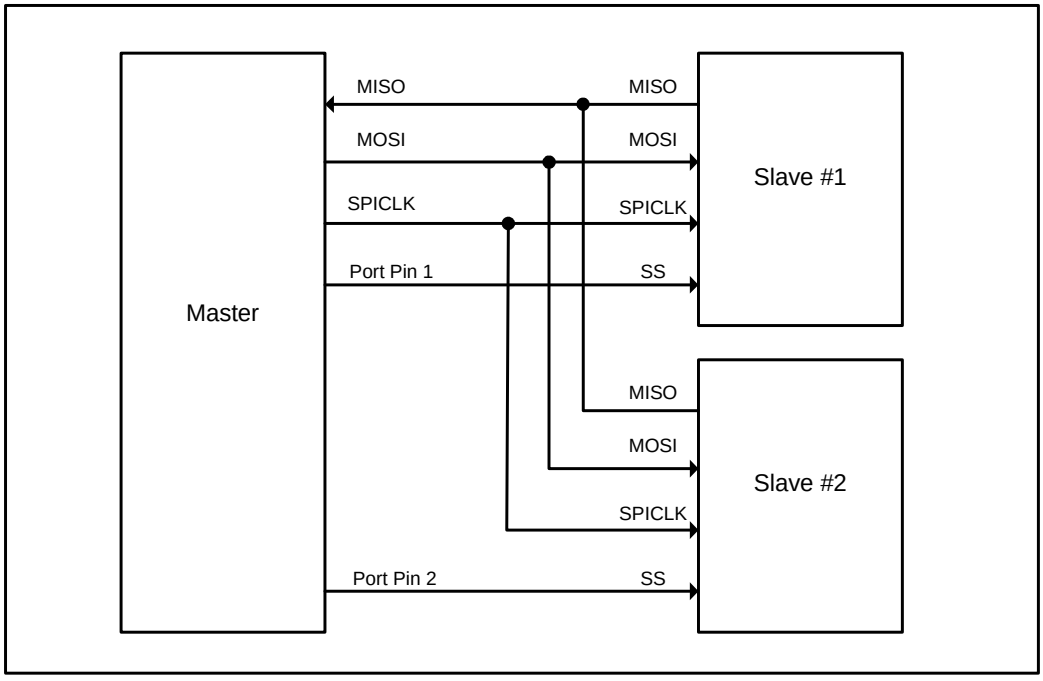
Typical Connection



SPI single master single slave configurartion



SPI dual device configuarion, where either can be a master or a slave



SPI single master multiple slaves configurartion

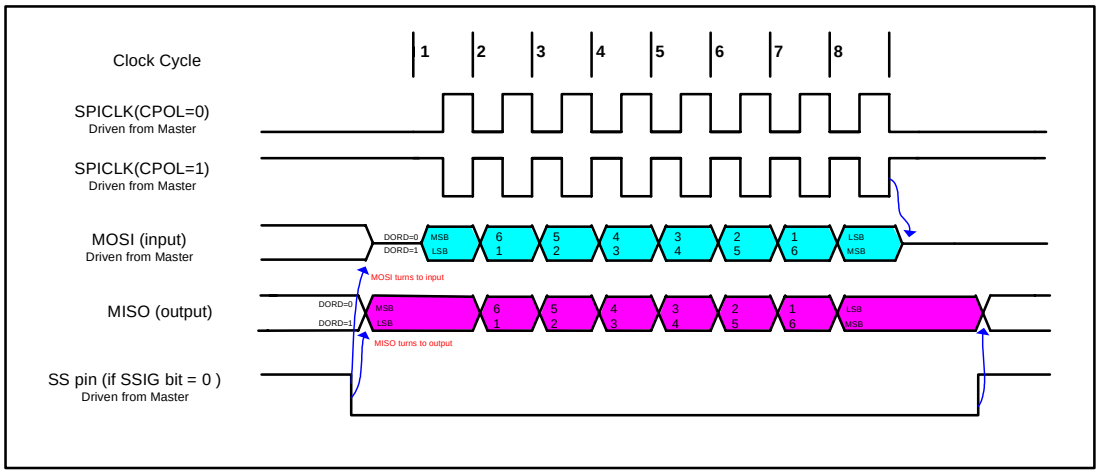
Communication

In SPI, transfers are always initiated by the master. If the SPI is enabled(**SPEN**=1) and selected as master, any instruction that use SPI data register SPIDAT as the destination will starts the SPI clock generator and a data transfer. The data will start to appear on MOSI about one half SPI bit-time to one SPI bit-time after it. Before starting the transfer, the master may select a slave by driving the SS pin of the corresponding device low. Data written to the **SPIDAT** register of the master shifted out of MOSI pin of the master to the MOSI pin of the slave. And at the same time the data in **SPIDAT** register of the selected slave is shifted out of MISO pin to the MISO pin of the master. During one byte transfer, data in the master and in the slave is interchanged. After shifting one byte, the transfer completion flag(**SPIF**) is set and an interrupt will be created if the SPI interrupt is enabled.

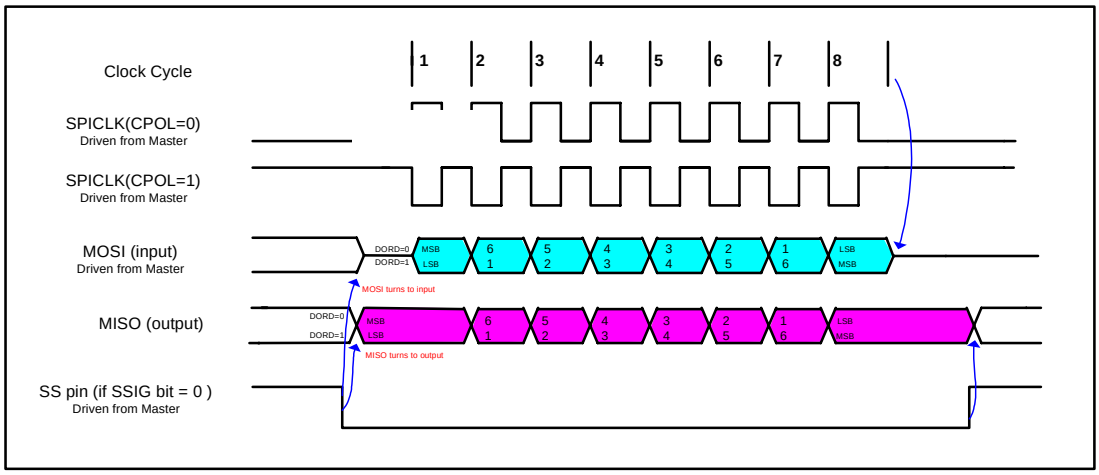
If **SPEN**=1, **SSIG**=0, SS pin=1 and **MSTR**=1, the SPI is enabled in master mode. Before the instruction that use **SPIDAT** as the destination register, the master is in idle state and can be selected as slave device by any other master drives the idle master SS pin low. Once this happened, **MSTR** bit of the idle master is cleared by hardware and changes its state a selected slave. User software should always check the **MSTR** bit. If this bit is cleared by the mode change of SS pin and the user wants to continue to use the SPI as a master later, the user must set the **MSTR** bit again, otherwise it will always stay in slave mode.

The SPI is single buffered in transmit direction and double buffered in receive direction. New data for transmission can not be written to the shift register until the previous transaction is complete. The **WCOL** bit is set to signal data collision when the data register is written during transaction. In this case, the data currently being transmitted will continue to be transmitted, but the new data which causing the collision will be lost. For receiving data, received data is transferred into a internal parallel read data buffer so that the shift register is free to accept a second byte. However, the received byte must be read from the data register(**SPIDAT**) before the next byte has been completely transferred. Otherwise the previous byte is lost. **WCOL** can be cleared in software by “writing 1 to the bit”.

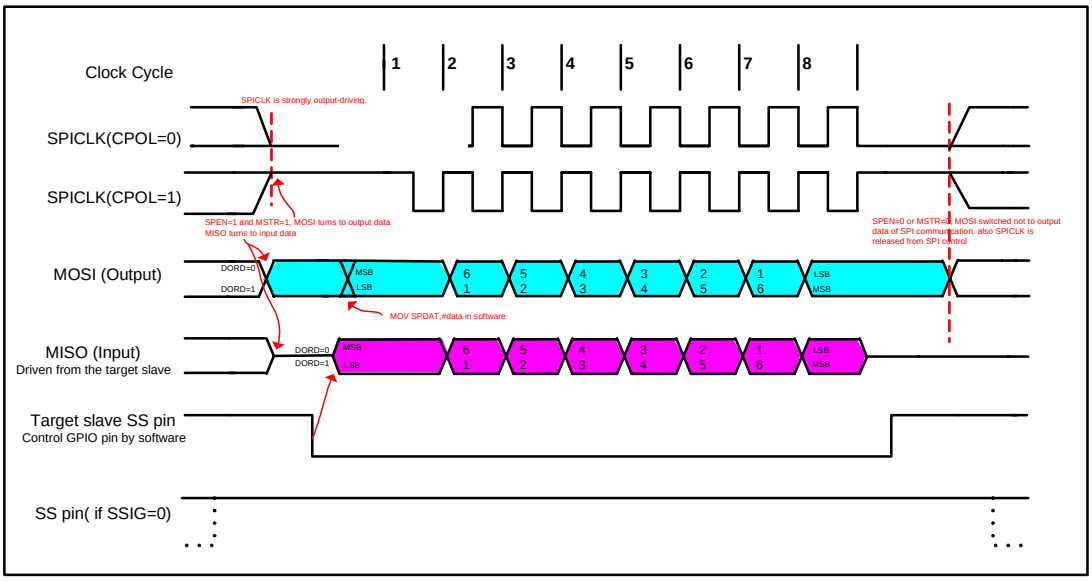
Typical Timing Diagram



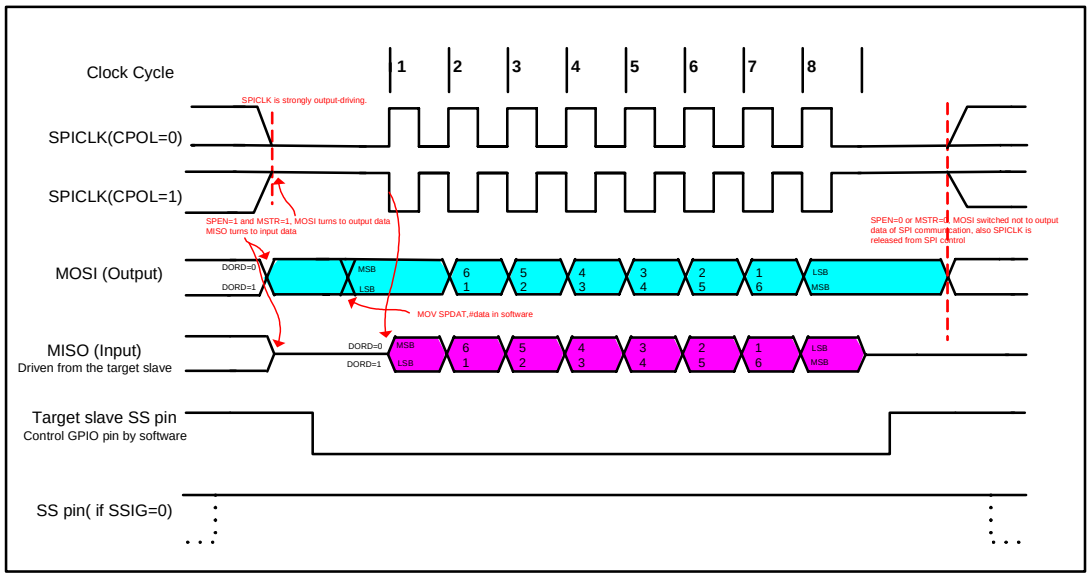
SPI slave transfer format with CPHA=0



SPI slave transfer format with CPHA=1

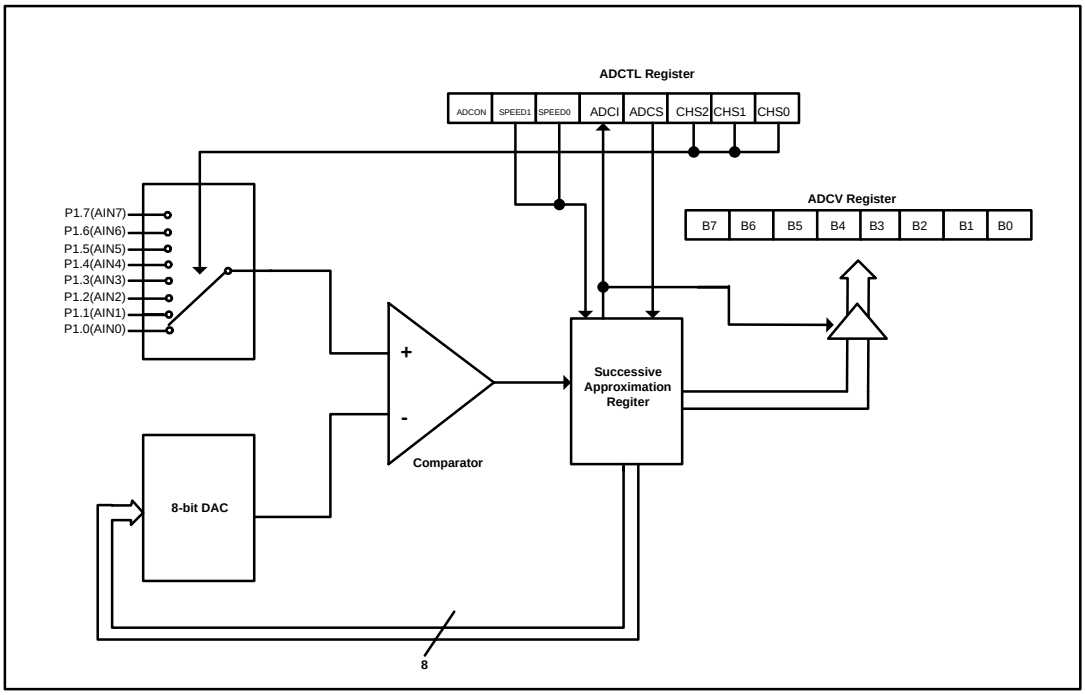


SPI master transfer format with CPHA=0



SPI master transfer format with CPHA=1

Analog to Digital Converter



The ADC on MPC82E52A is an 8-bit resolution, successive-approximation approach, medium-speed A/D converter. V_{REFP} / V_{REFM} is the positive/negative reference voltage input for internal voltage-scaling DAC use, the typical sink current on it is 600uA ~ 1mA. For MPC82E52A, these two references are internally tied to VDD and GND, separately.

Conversion is invoked since ADCS bit is set. The converter takes around a fourth cycles to sample analog input data and other three fourths cycles in successive-approximation steps. Total conversion time is controlled by two register bits – **SPEED1** and **SPEED0**. Analog input source comes from P1, one of the eight-channels is multiplexed by analog multiplexer into the comparator. When conversion is completed, the result will be saved onto **ADCV** register. After the result has been loaded onto **ADCV** register, **ADCI** will be set. **ADCI** associated with its enable register **AUXR**.4(**EADCI**), shares **ESPI** bit with **SPI** block to control the interrupt. **ADCI** should be cleared in software. The ADC interrupt service routine vectors to $2B_H$. When the chip enters idle mode or powerdown mode, the power of ADC is turned off by hardware.

$$\mathbf{ADCV} = 256 \times \frac{\mathbf{Vin} - \mathbf{V_{REFM}}}{\mathbf{V_{REFP}} - \mathbf{V_{REFM}}}$$

SFR: **ADCTL** *(ADC ConTroL register)*

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
ADCON	**SPEED1**	**SPEED0**	**ADCI**	**ADCS**	**CHS2**	**CHS1**	**CHS0**

ADCON :=

When clear shut down the power of ADC block. When set turn on the power of ADC block.

{SPEED1, SPEED0} := Conversion speed selector

{0,0} := (default)

A conversion takes 840 clock cycles

{0,1} :=

A conversion takes 630 clock cycles

{1,0} :=

A conversion takes 420 clock cycles

{1,1} :=

A conversion takes 210 clock cycles

ADCS := ADC Start control

Set to start an A/D conversion. It will be automatically cleared by the device after the device has finished the conversion.

ADCI := ADC Interrupt flag

It will be set by the device after the device has finished a conversion, and should be cleared by the user's software.

{CHS2, CHS1, CHS0} := Input CHannel Selector

{0,0,0} := (default)

Set P1.0 as the A/D channel input

{0,0,1} :=

Set P1.1 as the A/D channel input

{0,1,0} :=

Set P1.2 as the A/D channel input

{0,1,1} :=

Set P1.3 as the A/D channel input

{1,0,0} :=

Set P1.4 as the A/D channel input

{1,0,1} :=

Set P1.5 as the A/D channel input

{1,1,0} :=

Set P1.6 as the A/D channel input

{1,1,1} :=

Set P1.7 as the A/D channel input

SFR: **ADCV** ***(***ADC Value register***):***

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
Achieved Conversion Value							

The ADCV is the final result from the A/D conversion.

Built-In Oscillator

There is an oscillator built in the MPC82E52A which can be used as the oscillating source replacing the external crystal oscillator in some specific applications.

To enable the built-in oscillator, an user must configure the device by clearing(enable) the bit **ENROSC** in NVM register **OR2** via a general writer.

Making use of the built-in oscillator saves the cost of a crystal oscillator.

Typically, the frequency of the built-in oscillator is designed as 6MHz at 25℃. Dealing with temperature variation, the frequency could vary from 4.2MHz to 7.8MHz (~30%). It is designed for applications which don't ask very precise oscillating frequency, but not those applications asking high precision of oscillator frequency.

Power-Up and Low Voltage Detector and Reset

This device will never start to work until the power supply reachs about 3.3V, however, that is not stable voltage supply to provide well writing for the embedded flash.

There is a Low-Voltage detector(LVD) built in this device. While the voltage supply spans cross about 3.7V, the LVD will set bit **LVF**(**PCON**.5). Initially the bit **LVF** will be set after power-up, and the user should clear it for further detecting.

There is another bit **ENLVFI**(**AUXR**.2) designed to decide if to enable an interrupt to the MCU while the bit **LVF**(**PCON**.5) is set. This mechanism is convenient to inform the MCU take some emergent action.

The user can configure the device to inhibit the flash read and write actions from ISP and IAP statements by clearing bit **LVFWP**(**OR0**.7) while the voltage level falls under 3.7V.

If the user never likes to let the device work under power supply less than 3.7V, he can configure the device to automatically go to reset state by clearing bit **ENLVR**(**OR0**.6). It is named Low-Voltage-Reset.

Power Management

IDLE Mode

An instruction setting **PCON**.0 causes the device go into the idle mode, the internal clock is gated off to the CPU but not to the interrupt, timer, PCA, SPI, ADC, WDT and serial port functions.

There are two ways to terminate the idle. Activation of any enabled interrupt will cause **PCON**.0 to be cleared by hardware, terminating the idle mode. The interrupt will be serviced, and following RETI instruction, the next instruction to be executed will be the one following the instruction that puts the device into idle. Another way to wake-up from idle is to pull pin RST high to generate internal hardware reset.

Save Power Consumption under IDLE Mode

A clock divider(CLKDIV) associated with idle mode is used to slow down the system clock source in order to save power in advance. Everybody knows that slower the clock oscillates, less power consumed. Software could program this clock divider register prior to enter the idle mode. When the chip enters the idle mode, the clock is switched to the divider. When the chip exits the idle mode, clocking will return to the original clock behavior.

SFR: **PCON2**

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
-	**-**	**-**	**-**	**-**	**CKS2**	**CKS1**	**CKS0**

{CKS2, CKS1, CKS0}: Clock selector under idle mode

{0, 0,0} : = (default)
In idle mode, clock is not divided (default state)
{0, 0, 1} : =
In idle mode, clock is divided by 2
{0, 1, 0} : =
In idle mode, clock is divided by 4
{0, 1, 1} : =
In idle mode, clock is divided by 8
{1, 0,0} : =
In idle mode, clock is divided by 16
{1, 0, 1} : =
In idle mode, clock is divided by 32
{1, 1, 0} : =
In idle mode, clock is divided by 64
{1, 1, 1} : =
In idle mode, clock is divided by 128

POWER-DOWN Mode

An instruction setting **PCON**.1 causes the device go into the *POWER-DOWN* mode. In the *POWER-DOWN* mode, the on-chip oscillator is stopped. The contents of on-chip RAM and SFRs are maintained.

The power-down mode can be woken-up by either pin RST event or interrupt from INT0. When it is woken-up by RST, the program will execute from the address 0x0000. Be carefully to keep RST pin active for at least 10ms in order for a stable clock. If it is woken-up from pin INT0, the CPU will rework through jumping to related interrupt service routine. Before the CPU rework, the clock is blocked and counted until 32768 in order for debouncing the unstable clock. To use INT0 wake-up, interrupt-related registers have to be enabled and programmed accurately before entering power-down. ***Pay attention to add at least one "NOP" instruction subsequent to the power-down instruction if I/O wake-up is used.***

SFR: **PCON**

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
SMOD	*reserved0*	**LVF**	**POF**	**GF1**	**GF0**	**PD**	**IDL**

SMOD:= Double baud rate of UART interface

0: =

Keep normal baud rate when the UART is used in mode 1,2 or 3.

1 : =

Double baud rate when the UART is used in mode 1,2 or 3.

***reserved0*:=** no use. Default set as 0. The user can not set it.

LVF:= Low-Voltage Flag

After power-up, this bit will be initially set. The user should clear it by his program.
Continuously on operating, it will be set if the power supply drops under 3.7V.

POF:= Power-On flag

This bit will be set after the device was powered on.
It must be cleared by the user's software.

PD:= Power-Down switch

Set this bit to drive the device enter *POWER-DOWN* mode.

IDL:= Idle flag

Set this bit to drive the device enter *IDLE* mode.

Reset and Boot Entrance

There could be five conditions will cause the device to be reset.

- Power-Up
- RST pin press
- Watch Dog Timer overflows
- Power supply drops (option reset)
- Software invokes

On the power-up moment, the device will load the NVM bit **HWBS**(**OR0**.3) into a bit named **SWBS** in SFR **ISPCR**(explained later), and then determine the boot entrance according to **SWBS**.

If a reset is caused exclusive Power-Up, the boot entrance is determined only by SWBS, while no preloading happens from **HWBS** to **SWBS**.

The RST pin is used to reset this device. It is connected into the device to a Schmitt Trigger buffer, so to get excellent noise immunity.

Any positive pulse from RST pin must be kept at least 10us plus 36 oscillation cycles, or the device cannot be reset.

In System Programming and In Application Programming

In System Programming(ISP)

To develop a good program for ISP function, the user has to understand the architecture of the embedded flash.

The embedded flash consists of 16 pages. Each page contains 512 bytes.

Dealing with flash, the user must erase it in page unit before writing (programming) data into it.

Erasing flash means setting the content of that flash as *FF*h. Two erase modes are available in this chip. One is *mass mode* and the other is *page mode*. The *mass mode* gets more performance, but it erases the entire flash. The page mode is something performance less, but it is flexible since it erases flash in page unit.

Unlike RAM's real-time operation, to erase flash or to write (program) flash often takes long time so to wait finish.

Furthermore, it is a quite complex timing procedure to erase/program flash. Fortunately, the MPC82E52A carried with convenient mechanism to help the user read/change the flash content. Just filling the target address and data into several SFR, and triggering the built-in ISP automation, the user can easily erase, read, and program the embedded flash.

There are several SFR designed to help the user implement the ISP functionality.

SFR: **IFD** *(ISP Flash Data register)*

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
Data to be written into flash, or data got from flash							

IFD is the data port register for ISP/IAP operation. The data in IFD will be written into the desired address in operating ISP write and it is the data window of readout in operating ISP read.

SFR: **IFADRH** *(ISP Flash ADdRess High byte)*

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
ISP/IAP address High byte							

IFADRH is the high byte address for all ISP/IAP operation.

SFR: **IFADRL** *(ISP Flash ADdRess Low byte)*

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
ISP/IAP address Low byte							

IFADRL is the low byte address for all ISP/IAP operation.

SFR: **IFMT** *(ISP Flash-operating Mode Table)*

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
-	**-**	**-**	**-**	**-**	***reserve0***	**Mode Selection**	

Mode Selection		To Operate
0	*0*	Standby
0	*1*	AP-memory read
1	*0*	AP-memory/Data-flash program
1	*1*	AP-memory/Data-flash page erase

SFR: **SCMD** *(ISP Sequential Command register to trigger ISP/IAP operation)*

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
ISP-Command / Device ID							

SCMD is the command port for triggering ISP activity. If **SCMD** is filled with sequential 46_H, $B9_H$ and if **ISPCR**.7 = 1, ISP activity will be triggered.
When this register is read, the device ID of MPC82E52A will be returned (2 bytes). The MSB byte of this device ID is $F2_H$ and LSB byte 02_H. **IFADRL**.0 is used to select HIGH/LOW byte of the device ID.

SFR: **ISPCR** *(ISP ContRol register)*

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
ISPEN	**SWBS**	**SWRST**	**CFAIL**	**-**	**WAIT**		

ISPEN:= Determine if to Enable ISP/IAP functionality

0: =

Disable ISP program to change flash.

1 : =

Enable ISP program to change flash.

SWBS:= SoftWare Boot entrance Selector

0: =

Boot from main-memory.

1 : =

Boot from ISP memory.

Note: This bit will be loaded with **HWBS(OR0.3)** *after power-up moment.*

SWRST:= SoftWare ReSet trigger

Setting this bit will cause the device reset.

CFAIL:= ISP/IAP Command FAIL flag

0: =

The last ISP/IAP command has finished successfully.

1 : =

The last ISP/IAP command fails. It could be caused since the access of flash memory was inhibited.

WAIT:= Waiting time selection while the flash is busy.

	CPU Wait time (Oscillator cycle)			
ISPCR[2:0]	Page Erase	Program	Read	Recommended System clock
0 0 0	672384	1760	2	30M
0 0 1	504288	1320	2	24M
0 1 0	420240	1100	2	20M
0 1 1	252144	660	2	12M
1 0 0	126072	330	2	6M
1 0 1	63036	165	2	3M
1 1 0	42024	110	2	2M
1 1 1	21012	55	2	1M

***Notice*: Software reset actions could reset other SFR, but it never influences bits ISPEN and SWBS. The ISPEN and SWBS only will be reset by power-up action, while not software reset.**

Procedures demonstrating ISP function

```
IFMT ← xxxxx011B                    /* choice page-erasing command */
ISPCR ← 100xx010B                   /* set ISPEN=1 to enable flash change.
                                       set WAIT=010, 10942 MC; assumed 10M X's*/
IFADRH ← (page address high byte)   /* specify the address of the page to be erased */
IFADRL ← (page address low byte)
SCMD ← 46h                          /* trig ISP activity */
SCMD ← B9h
(CPU progressing will be hold here )
(CPU continues)
```

Erase a specific flash page

```
IFMT ← xxxxx010B                    /* choice byte-programming command */
ISPCR ← 100xx010B                   /* set ISPEN=1 to enable flash change.
                                       set WAIT=010, 60 MC; assumed 10M X's*/
IFADRH ← (Address high byte)        /* specify the address to be programmed */
IFADRL ← (Address low byte)
IFD ← (byte date to be written into flash)   /* prepare data source */
SCMD ← 46h                          /* trig ISP activity */
SCMD ← B9h
(CPU progressing will be hold here)
(CPU continues)
```

Program a byte into flash

```
IFMT ← xxxxx001B                    /* choice byte-read command */
ISPCR ← 100xx010B                   /* set ISPEN=1 to enable flash change.
                                       set WAIT=010, 11 MC; assumed 10M X's*/
IFADRH ← (Address high byte)        /* specify the address to be read */
IFADRL ← (Address low byte)
SCMD ← 46h                          /* trig ISP activity */
SCMD ← B9h
(CPU progressing will be hold here)
(CPU continues and currently IFD contain the desired data byte )
```

Read a byte from flash

Switching from ISP program to AP program

The device permits the user normally start running his AP program as soon as the ISP program has finished updating the flash content. Just program an instruction at the tail of ISP program as

```
ISPCR ← 001xxxxxB
```

which disables flash-writing authority, set **SWBS** *0*, and trigger a software reset. After that, the system will be reset (not powered-up), and the system will refer to **SWBS** so to startup from AP program entrance. For power-up procedure, the **HWBS** will be referred to decide the program entrance, but for software reset, **SWBS** will be referred to.

Switch to the ISP program from AP program

The device also permits the user program switches directly to the ISP program. Just program an instruction in the AP program as

```
ISPCR ← x11xxxxx_B
```

which sets **SWBS** *1* to direct the device boot from AP program, and trigger a software reset. After that, the system will be reset (not powered-up), and the system will refer to **SWBS** so to startup from ISP program entrance.

In-Application Program (IAP)

The In-Application Program feature is designed for user to Read/Write nonvolatile *data flash*. It may bring great help to store parameters those should be independent of power-up and power-done action. In other words, the user can store data in *data flash* memory, and after he shutting down the MCU and rebooting the MCU, he can get the original value, which he had stored in.

The user can program the *data flash* according to the same way as ISP program, so he should get deeper understanding related to SFR **IFD, IFADRL, IFADRH, IFMT, SCMD, and ISPCR**.

The *data flash* can be programmed by the AP program as well as the ISP program.

The ISP program may program the AP memory and *data flash*, while the AP program may program the *data flash* but not the ISP memory. If the AP program desires to change the ISP memory associated with specific address space, the hardware will ignore it.

Avoid Inadvertent Data Lost from IAP/ISP

If the user invoke ISP/IAP function in his application, it is possible the MCU inadvertently jumps to those ISP/IAP statements while the power supply drops under specific level. The ISP/IAP statements could destroy the flash content.

To avoid the MCU inadvertently jumps to IAP/ISP instructions while the voltage falls under a specific level on power-up and power-off moment, it is strongly suggested that the user enables the Low-Voltage-Reset function by clearing bit **ENLVR**(**OR0**.6), so to makes this device never work under Low-Voltage.

Also the user should make great use of **LVF**(**PCON**.5) and **ENLVFI**(**AUXR**.2) to detect the voltage dropping.

Instructions Set

DATA TRASFER			
MNEMONIC	**DESCRIPTION**	**BYT**	**CYC**
MOV A, Rn	Move register to Acc	1	1
MOV A, direct	Move direct byte o Acc	2	2
MOV A, @Ri	Move indirect RAM to Acc	1	2
MOV A, #data	Move immediate data to Acc	2	2
MOV Rn, A	Move Acc to register	1	2
MOV Rn, direct	Move direct byte to register	2	4
MOV Rn, #data	Move immediate data to register	2	2
MOV direct, A	Move Acc to direct byte	2	3
MOV direct, Rn	Move register to direct byte	2	3
MOV direct, direct	Move direct byte to direct byte	3	4
MOV direct, @Ri	Move indirect RAM to direct byte	2	4
MOV direct, #data	Move immediate data to direct byte	3	3
MOV @Ri, A	Move Acc to indirect RAM	1	3
MOV @Ri, direct	Move direct byte to indirect RAM	2	3
MOV @Ri, #data	Move immediate data to indirect RAM	2	3
MOV DPTR,#data16	Load DPTR with a 16-bit constant	3	3
MOVC A,@A+DPTR	Move code byte relative to DPTR to Acc	1	4
MOVC A,@A+PC	Move code byte relative to PC to Acc	1	4
PUSH direct	PUSH DIRECT BYTE ONTO STACK	2	4
POP direct	POP DIRECT BYTE FROM STACK	2	3
XCH A, Rn	EXCHANGE REGISTER WITH ACC	1	3
XCH A, direct	EXCHANGE DIRECT BYTE WITH ACC	2	4
XCH A, @Ri	EXCHANGE INDIRECT RAM WITH ACC	1	4
XCHD A, @Ri	EXCHANGE LOW-ORDER DIGIT INDIRECT RAM WITH ACC	1	4

ARITHEMATIC OPERATIONS			
MNEMONIC	**DESCRIPTION**	**BYT**	**CYC**
ADD A, Rn	ADD REGISTER TO ACC	1	2
ADD A, direct	ADD DIRECT BYTE TO ACC	2	3
ADD A, @Ri	ADD INDIRECT RAM TO ACC	1	3
ADD A, #data	ADD IMMEDIATE DATA TO ACC	2	2
ADDC A, Rn	ADD REGISTER TO ACC WITH CARRY	1	2
ADDC A, direct	ADD DIRECT BYTE TO ACC WITH CARRY	2	3
ADDC A, @Ri	ADD INDIRECT RAM TO ACC WITH CARRY	1	3
ADDC A, #data	ADD IMMEDIATE DATA TO ACC WITH CARRY	2	2
SUBB A, Rn	SUBTRACT REGISTER FROM ACC WITH BORROW	1	2
SUBB A, direct	SUBTRACT DIRECT BYTE FROM ACC WITH BORROW	2	3
SUBB A, @Ri	SUBTRACT INDIRECT RAM FROM ACC WITH BORROW	1	3
SUBB A, #data	SUBTRACT IMMEDIATE DATA FROM ACC WITH BORROW	2	2
INC A	INCREMENT ACC	1	2
INC Rn	INCREMENT REGISTER	1	3
INC direct	INCREMENT DIRECT BYTE	2	4
INC @Ri	INCREMENT INDIRECT RAM	1	4
DEC A	DECREMENT ACC	1	2
DEC Rn	DECREMENT REGISTER	1	3
DEC direct	DECREMENT DIRECT BYTE	2	4
DEC @Ri	DECREMENT INDIRECT RAM	1	4
INC DPTR	INCREMENT DPTR	1	1
MUL AB	MULTIPLY A AND B	1	4
DIV AB	DIVIDE A BY B	1	5
DA A	DECIMAL ADJUST ACC	1	4

LOGIC OPERATION			
MNEMONIC	**DESCRIPTION**	**BYT**	**CYC**
ANL A, Rn	AND REGISTER TO ACC	1	2
ANL A, direct	AND DIRECT BYTE TO ACC	2	3
ANL A, @Ri	AND INDIRECT RAM TO ACC	1	3
ANL A, #data	AND IMMEDIATE DATA TO ACC	2	2
ANL direct, A	AND ACC TO DIRECT BYTE	2	4
ANL direct, #data	AND IMMEDIATE DATA TO DIRECT BYTE	3	4
ORL A, Rn	OR REGISTER TO ACC	1	2
ORL A, direct	OR DIRECT BYTE TO ACC	2	3
ORL A, @Ri	OR INDIRECT RAM TO ACC	1	3
ORL A, #data	OR IMMEDIATE DATA TO ACC	2	2
ORL direct, A	OR ACC TO DIRECT BYTE	2	4
ORL direct, #data	OR IMMEDIATE DATA TO DIRECT BYTE	3	4
XRL A, Rn	EXCLUSIVE-OR REGISTER TO ACC	1	2
XRL A, direct	EXCLUSIVE-OR DIRECT BYTE TO ACC	2	3
XRL A, @Ri	EXCLUSIVE-OR INDIRECT RAM TO ACC	1	3
XRL A, #data	EXCLUSIVE-OR IMMEDIATE DATA TO ACC	2	2
XRL direct, A	EXCLUSIVE-OR ACC TO DIRECT BYTE	2	4
XRL direct, #data	EXCLUSIVE-OR IMMEDIATE DATA TO DIRECT BYTE	3	4
CLR A	CLEAR ACC	1	1
CPL A	COMPLEMENT ACC	1	2
RL A	ROTATE ACC LEFT	1	1
RLC A	ROTATE ACC LEFT THROUGH THE CARRY	1	1
RR A	ROTATE ACC RIGHT	1	1
RRC A	ROTATE ACC RIGHT THROUGH THE CARRY	1	1
SWAP A	SWAP NIBBLES WITHIN THE ACC	1	1

BOOLEAN VARIABLE MANIPULATION			
MNEMONIC	**DESCRIPTION**	**BYT**	**CYC**
CLR C	CLEAR CARRY	1	1
CLR bit	CLEAR DIRECT BIT	2	4
SETB C	SET CARRY	1	1
SETB bit	SET DIRECT BIT	2	4
CPL C	COMPLEMENT CARRY	1	1
CPL bit	COMPLEMENT DIRECT BIT	2	4
ANL C, bit	AND DIRECT BIT TO CARRY	2	3
ANL C, /bit	AND COMPLEMENT OF DIRECT BIT TO CARRY	2	3
ORL C, bit	OR DIRECT BIT TO CARRY	2	3
ORL C, /bit	OR COMPLEMENT OF DIRECT BIT TO CARRY	2	3
MOV C, bit	MOVE DIRECT BIT TO CARRY	2	3
MOV bit, C	MOVE CARRY TO DIRECT BIT	2	4

BOOLEAN VARIABLE BRANCH			
MNEMONIC	**DESCRIPTION**	**BYT**	**CYC**
JC rel	JUMP IF CARRY IS SET	2	3
JNC rel	JUMP IF CARRY NOT SET	2	3
JB bit, rel	JUMP IF DIRECT BIT IS SET	3	4
JNB bit, rel	JUMP IF DIRECT BIT NOT SET	3	4
JBC bit, rel	JUMP IF DIRECT BIT IS SET AND THEN CLEAR BIT	3	5

PROAGRAM BRACHING			
MNEMONIC	**DESCRIPTION**	**BYT**	**CYC**
ACALL addr11	ABSOLUTE SUBROUTINE CALL	2	6
LCALL addr16	LONG SUBROUTINE CALL	3	6
RET	RETURN FROM SUBROUTINE	1	4
RETI	RETURN FROM INTERRUPT SUBROUTINE	1	4
AJMP addr11	ABSOLUTE JUMP	2	3
LJMP addr16	LONG JUMP	3	4
SJMP rel	SHORT JUMP	2	3
JMP @A+DPTR	JUMP INDIRECT RELATIVE TO DPTR	1	3
JZ rel	JUMP IF ACC IS ZERO	2	3
JNZ rel	JUMP IF ACC NOT ZERO	2	3
CJNE A, direct, rel	COMPARE DIRECT BYTE TO ACC AND JUMP IF NOT EQUAL	3	5
CJNE A, #data, rel	COMPARE IMMEDIATE DATA TO ACC AND JUMP IF NOT EQUAL	3	4
CJNE Rn, #data, rel	COMPARE IMMEDIATE DATA TO REGISTER AND JUMP IF NOT EQUAL	3	4
CJNE @Ri, #data, rel	COMPARE IMMEDIATE DATA TO INDIRECT RAM AND JUMP IF NOT	3	5
	EQUAL		
DJNZ Rn, rel	DECREMENT REGISTER AND JUMP IF NOT EQUAL	2	4
DJNZ direct, rel	DECREMENT DIRECT BYTE AND JUMP IF NOT EQUAL	3	5
NOP	NO OPERATION	1	1

****** INHIBITED INSTRUCTION ******			
MNEMONIC	**DESCRIPTION**	**BYT**	**CYC**
MOVX A, @Ri	Move external RAM(8-bit address) to Acc	1	3
MOVX A, @DPTR	MOVE EXTERNAL RAM(16-BIT ADDRESS) TO ACC	1	2
MOVX @Ri, A	MOVE ACC TO EXTERNAL RAM(8-BIT ADDRESS)	1	3
MOVX @DPTR, A	MOVE ACC TO EXTERNAL RAM(16-BIT ADDRESS)	1	2

Absolute Maximum Rating

Parameter	Rating
Operating Voltage	4.5V ~ 5.5V
Operating temperature under bias	0 ~ 70^{o}C
Storage temperature	0 ~ 125^{o}C
Voltage on any pin	-0.5 ~ 5.5V
Operating Frequency	DC ~ 25MHz

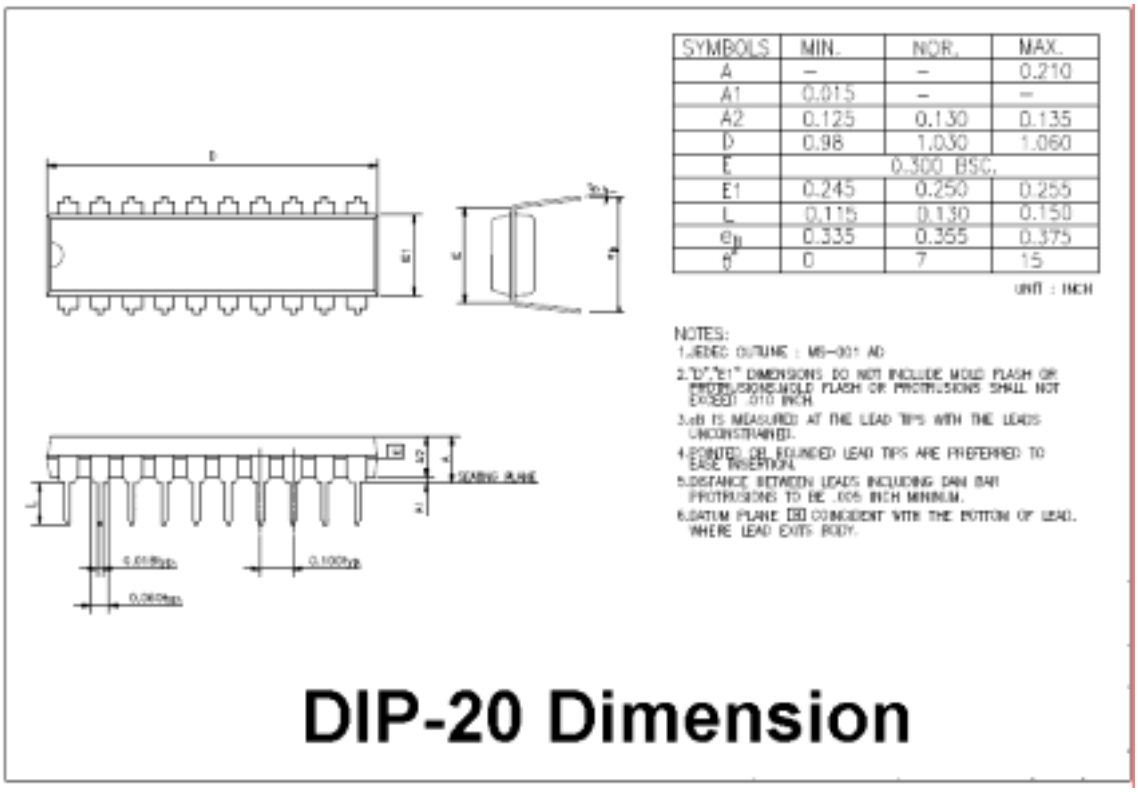
DC Characteristics

VSS = 0V, TA = 25 ℃, V_{CC} = 5.0V unless otherwise specified

Symbol	Parameter	Test Condition	Limits			Unit
			min	typ	max	
V_{IH}	Input High voltage	4.5V < Vcc <5.5V	2.2			V
V_{IL}	Input Low voltage	4.5V < Vcc <5.5V			0.8	V
I_{OL}	Output Low current	V_{PIN} =0.45V	16	20		mA
I_{OH1}	Output High current(push-pull)	V_{PIN} =2.4V	4	5		mA
I_{OH2}	Output High current(Quasi-bidirection)	V_{PIN} =2.4V		220		uA
I_{IL1}	Logic 0 input current(Quasi-bidirection)	V_{PIN} =0.45V		17	30	uA
I_{IL2}	Logic 0 input current(Input-Only)	V_{PIN} =0.45V		0	10	uA
I_{LK}	Input Leakage current(Open-Drain output)	V_{PIN} = V_{CC}		0	10	uA
I_{H2L}	Logic 1 to 0 transition current(P1,3)	V_{PIN} =1.8V		225	600	uA
I_{OP}	Operating current	F_{OSC} = 12MHz		12	20	mA
I_{RESET}	Reset Current	V_{CC} =5.0V		11	20	mA
I_{IDLE}	Idle mode current @ 20MHz	V_{CC} =5.0V		6	12	mA
I_{PD}	Power down current	V_{CC} =5.0V		0.5	10	uA
R_{RST}	Internal reset pull-down resistance	V_{CC} =5.0V		100		Kohm

Package Dimension

20-pin PDIP (MPC82E52AE)

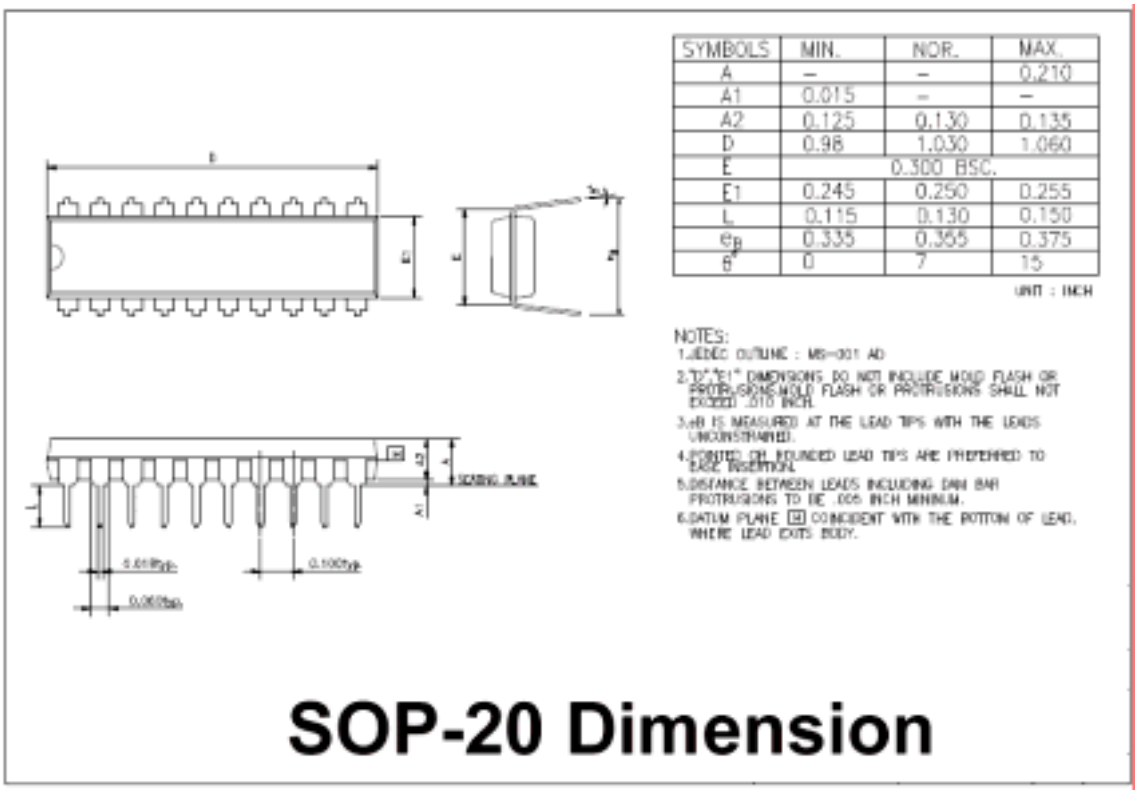


SYMBOLS	MIN.	NOR.	MAX.
A	–	–	0.210
A1	0.015	–	–
A2	0.125	0.130	0.135
D	0.98	1.030	1.060
E		0.300 BSC.	
E1	0.245	0.250	0.255
L	0.115	0.130	0.150
e_B	0.335	0.355	0.375
θ°	0	7	15

UNIT : INCH

NOTES:
1.JEDEC OUTLINE : MS-001 AD
2."D","E1" DIMENSIONS DO NOT INCLUDE MOLD FLASH OR PROTRUSIONS.MOLD FLASH OR PROTRUSIONS SHALL NOT EXCEED .010 INCH.
3.eB IS MEASURED AT THE LEAD TIPS WITH THE LEADS UNCONSTRAINED.
4.POINTED OR ROUNDED LEAD TIPS ARE PREFERRED TO EASE INSERTION.
5.DISTANCE BETWEEN LEADS INCLUDING DAM BAR PROTRUSIONS TO BE .005 INCH MINIMUM.
6.DATUM PLANE [H] COINCIDENT WITH THE BOTTOM OF LEAD, WHERE LEAD EXITS BODY.

20-pin SOP (MPC82E52AS)



SYMBOLS	MIN.	NOR.	MAX.
A	–	–	0.210
A1	0.015	–	–
A2	0.125	0.130	0.135
D	0.98	1.030	1.060
E		0.300 BSC.	
E1	0.245	0.250	0.255
L	0.115	0.130	0.150
e_B	0.335	0.355	0.375
θ°	0	7	15

UNIT : INCH

NOTES:
1.JEDEC OUTLINE : MS-001 AD
2."D","E1" DIMENSIONS DO NOT INCLUDE MOLD FLASH OR PROTRUSIONS.MOLD FLASH OR PROTRUSIONS SHALL NOT EXCEED .010 INCH.
3.eB IS MEASURED AT THE LEAD TIPS WITH THE LEADS UNCONSTRAINED.
4.POINTED OR ROUNDED LEAD TIPS ARE PREFERRED TO EASE INSERTION.
5.DISTANCE BETWEEN LEADS INCLUDING DAM BAR PROTRUSIONS TO BE .005 INCH MINIMUM.
6.DATUM PLANE [H] COINCIDENT WITH THE BOTTOM OF LEAD, WHERE LEAD EXITS BODY.

Version History

Version	Date	Page	Description
A1	2005/09		Initial issue