July 21, 2008

C8051F50x Revision A Errata

Errata Status Summary

This document summarizes all known errata with Revision A of the C8051F50x devices.

Errata #	Title	Impact	Status	
			Revision A Device	Revision B Device
1	XTAL, RC and C modes not available	Major	Issue Exists	Fixed
2	Clock multiplier options	Minor	Issue Exists	Fixed
3	P4MDOUT not usable	Major	Issue Exists	Fixed
4	CAN Interface 2 not available	Minor	Issue Exists	Fixed
5	CAN Register Bit Alignment	Minor	Issue Exists	Fixed
6	VDD Monitor Reset and /RST pin	Minor	Issue Exists	Fixed
7	SMBus SCL Timeout not available	Minor	Issue Exists	Fixed
8	SMBus Hardware ACK behavior	Major	Issue Exists	Issue Exists

Impact Definition: Each erratum is marked with an impact, as defined below:

- Minor—Workaround exists without any impact to device capability.
- Major—Workaround does not exist, or the workaround limits the capabilities of the device.
- Information—The device behavior is not ideal but acceptable. Typically, the data sheet will be changed to match the device behavior.

Errata Details

1. **Description**: The external crystal, RC, and C modes do not work on Revision A devices.

   Impact: The only external clock option available is a CMOS clock.

   Workaround: The internal precision oscillator or an external CMOS clock can be used as the system clock for the MCU. For systems that require the LIN or CAN peripherals, the internal oscillator is accurate enough for master mode operation.

   Resolution: All of the external clock options work as documented on Revision B devices.

2. **Description**: Not all clock multiplier options in the CLKMUL SFR are available on Revision A devices. On revision A, the valid settings for the MULSEL bits are Internal Oscillator x2 (00b) and External Oscillator x2 (01b). Internal Oscillator x4 (10b) and External Oscillator x4 (11b) are not available. Also, the MULDIV bits in CLKMUL have no effect on Revision A devices.

   Impacts: Not all clock multiplier options are available.

Workaround: One of the two available options (Internal Oscillator x2 option or External Oscillator x2) will be able to generate all system clock frequencies supported by the MCU. The other clock multiplier options provide added flexibility in terms of oscillator choice, but are not required.

Resolution: All documented clock multiplier options are available on Revision B devices.

3. **Description**: On Revision A devices, the P4MDOUT register does not configure the Port 4 mode. The effective value for this register is always 0x00.

   Impacts: All Port 4 pins are always in open-drain mode and cannot be configured for push-pull mode.

   Workaround: All digital inputs can be assigned to Port 4 and the remaining ports (Ports 0-3) can be used for any necessary push-pull outputs. If Port 4 is used for the external memory interface and push-pull capability is required, external pull-ups resistors in the 1K-5K ohm range can be added to each port pin. The pull-ups should be connected to VIO.

   Resolution: The P4MDOUT register properly configures the mode of the port on Revision B devices.

4. **Description**: The CAN Interface 2 registers are not accessible on Revision A devices.

   Impacts: All CAN communication must use Interface 1 on Revision A devices.

   Workaround: The firmware can use Interface 1 to access all message objects.

   Resolution: CAN Interface 2 is fully functional on Revision B devices.

5. **Description**: The following “32-bit” CAN registers have a misaligned organization that appears as a 1-bit left shift from the expected alignment as described in the Bosch CAN User’s Guide. The affected registers are as follows:

   CAN0MV1H:CAN0MV1L (Message Valid 1 Register)
   CAN0MV2H:CAN0MV2L (Message Valid 2 Register)

   CAN0IP1H:CAN0IP1L (Interrupt Pending 1 Register)
   CAN0IP2H:CAN0IP2L (Interrupt Pending 2 Register)

   CAN0TR1H:CAN0TR1L (Transmission Request 1 Register)
   CAN0TR2H:CAN0TR2L (Transmission Request 2 Register)

   CAN0ND1H:CAN0ND1L (New Data 1 Register)
   CAN0ND2H:CAN0ND2L (New Data 2 Register)

   The Bosch CAN User’s Guide documents the bit organization for all of these registers as follows:

   Register 1 (16-9 : 8-1)
   Register 2 (32-25 : 24-17)

   On Revision A devices, the bit organization for these four sets of registers is as follows:

   Register 1 (15-8 : 7-1, 32)
   Register 2 (31-24 : 23-16)

Impacts: Firmware must be aware of the bit-shift with Revision A devices.

Example 1: In Revision A devices, if new data is available in message object 2, the New Data registers will read as follows:

```
CAN0ND1L = 0x04
CAN0ND1H = 0x00
CAN0ND2L = 0x00
CAN0ND2H = 0x00
```

On Revision B devices, with new data available for message object 2, the New Data registers will read as follows:

```
CAN0ND1L = 0x02
CAN0ND1H = 0x00
CAN0ND2L = 0x00
CAN0ND2H = 0x00
```

Example 2: In Revision A devices, sending data through message object 32 will set the Transmission Request registers as follows:

```
CAN0TR1L = 0x01
CAN0TR1H = 0x00
CAN0TR2L = 0x00
CAN0TR2H = 0x00
```

On Revision B devices, sending data through message object 32 will set the Transmission Request registers as follows:

```
CAN0TR1L = 0x00
CAN0TR1H = 0x00
CAN0TR2L = 0x00
CAN0TR2H = 0x80
```

Workaround: The firmware for Revision A devices can interpret these four sets of data as documented above. All of the message objects are still properly represented in these registers. The following example C code shows how to shift the registers to match the alignment presented in the Bosch Can User's Guide:

```
// Variable Definitions

typedef union {                 // Define struct to hold all 32-bits
  unsigned long l;
  unsigned char c[4];
} UU64;

UU64 NewData;                   // Stores NewData properly aligned
unsigned char carry;            // LSB that carries as new MSB

// Read data from CAN registers and perform 1-bit right shift with carry

NewData.c[3] = CAN0ND1L;        // With a big endian compiler, assign LSB
NewData.c[2] = CAN0ND1H;
NewData.c[1] = CAN0ND2L;
NewData.c[0] = CAN0ND2H;        // Assign MSB

carry = NewData.c[3] & 0x01;    // Store carry bit
NewData.l = NewData.l >> 1;     // Perform 1-bit shift to realign
if (carry) {                    // Add carry if necessary
   NewData.c[0] = NewData.c[0] | 0x80; }
```

Resolution: On Revision B devices, the bit alignment matches the Bosch CAN User's Guide.

6. **Description**: The /RST pin is not pulled low upon a VDD monitor reset.

   Impacts: External devices depending on this behavior of the /RST pin will not be informed of the VDD monitor reset. The device is still reset internally as expected.

   Workaround: If the reset of the MCU must be coordinated with the reset of another device, the MCU can check RSTSRC register at the beginning of firmware and toggle a GPIO pin if the source of the last reset was a Power-On or VDD Monitor reset.

   Resolution: The /RST pin is pulled low upon a VDD monitor reset on Revision B devices.

7. **Description**: Setting the SMBTOE bit in the SMB0CF register has no effect.

   Impacts: The SMBus SCL Timeout detection is not available. Timer 3 will not automatically reload when SCL is high to prevent a timeout interrupt, even when timeout detection is enabled by setting SMBTOE.

   Workaround: SCL can be tied to another GPIO pin that is monitored by the MCU manually through firmware to check for a timeout.

   Resolution: On Revision B devices, setting the SMBTOE bit will force Timer 3 to reload when SCL is high, which allows Timer 3 to be used as the SCL Timeout detection Timer.

8. **Description**: The Address Hardware Acknowledge mechanism of the SMBus peripheral can cause an unexpected SMBus interrupt or cause an incorrect SMBus state transition. The behavior depends on the EXTHOLD bit in the SMB0CF register.

   a) When Hardware Acknowledge is enabled (EHACK = 1b, SMB0ADM) and SDA setup and hold times are not extended (EXTHOLD = 0, SMB0CF), the SMBus hardware will generate an SMBus interrupt, whether or not the address on the bus matches the hardware address match conditions. The expected behavior is that an interrupt is only generated when the address matches. When the MCU does vector to the interrupt service routine, the SMBus peripheral will be in the appropriate state and indicate the reception of a slave address.

   b) When Hardware Acknowledge is enabled (EHACK = 1b, SMB0ADM) and SDA setup and hold times are extended (EXTHOLD = 1, SMB0CF) the SMBus hardware will incorrectly clear the Start bit (STA) on reception of a slave address, which causes the firmware to interpret the state as the "Slave Receiver -- Data Byte received" state. This will only happen when the address match conditions determined by the SMB0ADR and SMB0MASK registers are met by the address presented on the bus.

   Impacts:

   b) Once the hardware vectors to the interrupt service routine, the hardware will hold SCL low until SI is cleared. Incompliant SMBus masters that do not support SCL clock stretching will not recognize that the clock is being stretched. If the received address does not match the conditions of SMB0ADR and SMB0MASK, the slave will generate a NACK, and even if software issues a write to SMB0DAT, that write will have no effect on the bus. No data collisions will occur.

   a) Once the hardware has matched an address and vectored to the interrupt service routine, the firmware will not be able to use the Start bit to distinguish between the reception of an address byte versus the reception of a data byte. However, the hardware will still correctly acknowledge the address byte (SLA+R/W).

   Workaround:

   a) The SMBus interrupt service routine should verify an address when it is received and clear SI as soon as possible if the address does not match.

   b) It is recommended that setup and hold times should not be extended when Hardware Acknowledge is enabled. Contact mcuapps@silabs.com for alternate workarounds if these two features are required.

   Resolution: These issues are not addressed on Revision B devices.