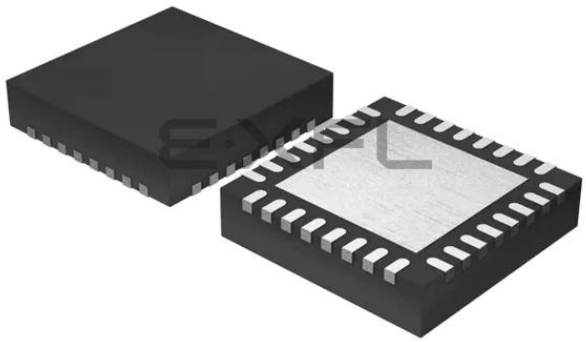

Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	8051
Core Size	8-Bit
Speed	50MHz
Connectivity	SMBus (2-Wire/I²C), SPI, UART/USART
Peripherals	POR, PWM, Temp Sensor, WDT
Number of I/O	25
Program Memory Size	16KB (16K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	2.25K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 5.25V
Data Converters	A/D 25x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	32-VFQFN Exposed Pad
Supplier Device Package	32-QFN (5x5)
Purchase URL	https://www.e-xfl.com/product-detail/silicon-labs/c8051f567-imr

8. Comparators

The C8051F55x/56x/57x devices include two on-chip programmable voltage Comparators. A block diagram of the comparators is shown in Figure 8.1, where “n” is the comparator number (0 or 1). The two Comparators operate identically except that Comparator0 can also be used a reset source. For input selection details, refer to SFR Definition 8.5 and SFR Definition 8.6.

Each Comparator offers programmable response time and hysteresis, an analog input multiplexer, and two outputs that are optionally available at the Port pins: a synchronous “latched” output (CP0, CP1), or an asynchronous “raw” output (CP0A, CP1A). The asynchronous signal is available even when the system clock is not active. This allows the Comparators to operate and generate an output with the device in STOP mode. When assigned to a Port pin, the Comparator outputs may be configured as open drain or push-pull (see Section “19.4. Port I/O Initialization” on page 174). Comparator0 may also be used as a reset source (see Section “16.5. Comparator0 Reset” on page 142).

The Comparator0 inputs are selected in the CPT0MX register (SFR Definition 8.5). The CMX0P1-CMX0P0 bits select the Comparator0 positive input; the CMX0N1-CMX0N0 bits select the Comparator0 negative input. The Comparator1 inputs are selected in the CPT1MX register (SFR Definition 8.6). The CMX1P1-CMX1P0 bits select the Comparator1 positive input; the CMX1N1-CMX1N0 bits select the Comparator1 negative input.

Important Note About Comparator Inputs: The Port pins selected as Comparator inputs should be configured as analog inputs in their associated Port configuration register, and configured to be skipped by the Crossbar (for details on Port configuration, see Section “19.1. Port I/O Modes of Operation” on page 170).

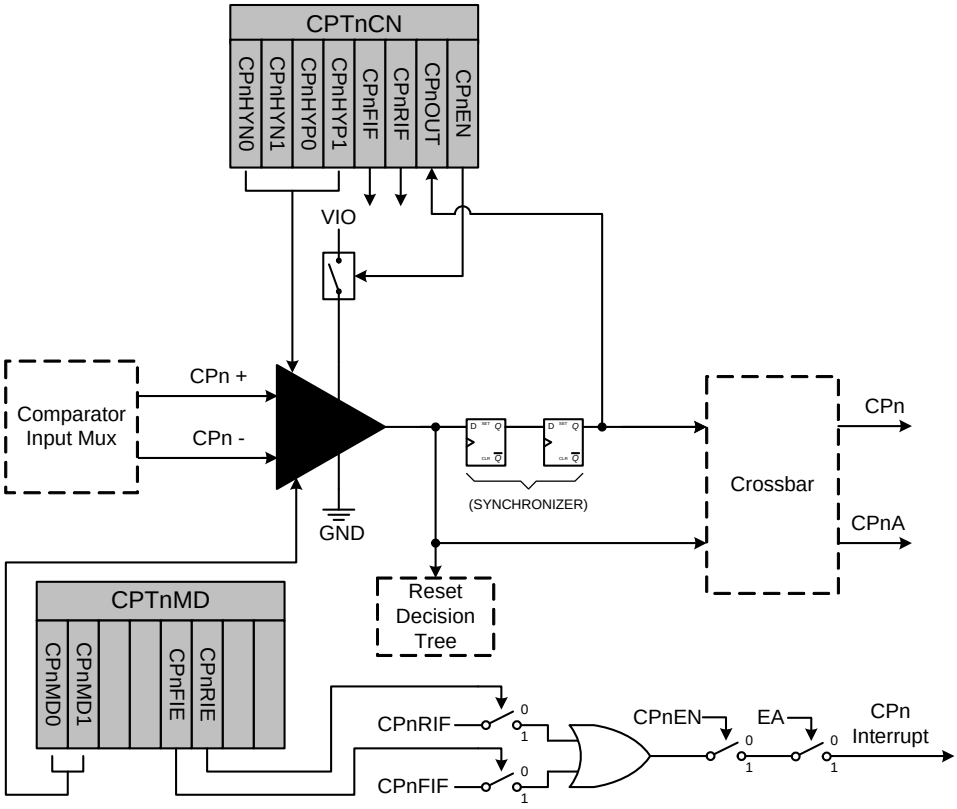

Figure 8.1. Comparator Functional Block Diagram

10.2. Instruction Set

The instruction set of the CIP-51 System Controller is fully compatible with the standard MCS-51™ instruction set. Standard 8051 development tools can be used to develop software for the CIP-51. All CIP-51 instructions are the binary and functional equivalent of their MCS-51™ counterparts, including opcodes, addressing modes and effect on PSW flags. However, instruction timing is different than that of the standard 8051.

10.2.1. Instruction and CPU Timing

In many 8051 implementations, a distinction is made between machine cycles and clock cycles, with machine cycles varying from 2 to 12 clock cycles in length. However, the CIP-51 implementation is based solely on clock cycle timing. All instruction timings are specified in terms of clock cycles.

Due to the pipelined architecture of the CIP-51, most instructions execute in the same number of clock cycles as there are program bytes in the instruction. Conditional branch instructions take one less clock cycle to complete when the branch is not taken as opposed to when the branch is taken. Table 10.1 is the CIP-51 Instruction Set Summary, which includes the mnemonic, number of bytes, and number of clock cycles for each instruction.

SFR Definition 10.1. DPL: Data Pointer Low Byte

Bit	7	6	5	4	3	2	1	0
Name	DPL[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0x82; SFR Page = All Pages

Bit	Name	Function
7:0	DPL[7:0]	**Data Pointer Low.** The DPL register is the low byte of the 16-bit DPTR. DPTR is used to access indirectly addressed Flash memory or XRAM.

SFR Definition 10.2. DPH: Data Pointer High Byte

Bit	7	6	5	4	3	2	1	0
Name	DPH[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0x83; SFR Page = All Pages

Bit	Name	Function
7:0	DPH[7:0]	**Data Pointer High.** The DPH register is the high byte of the 16-bit DPTR. DPTR is used to access indirectly addressed Flash memory or XRAM.

12.3. SFR Page Stack Example

The following is an example that shows the operation of the SFR Page Stack during interrupts. In this example, the SFR Control register is left in the default enabled state (i.e., SFRPGEN = 1), and the CIP-51 is executing in-line code that is writing values to SPI Data Register (SFR “SPI0DAT”, located at address 0xA3 on SFR Page 0x00). The device is also using the CAN peripheral (CAN0) and the Programmable Counter Array (PCA0) peripheral to generate a PWM output. The PCA is timing a critical control function in its interrupt service and so its associated ISR that is set to high priority. At this point, the SFR page is set to access the SPI0DAT SFR (SFRPAGE = 0x00). See Figure 12.2.

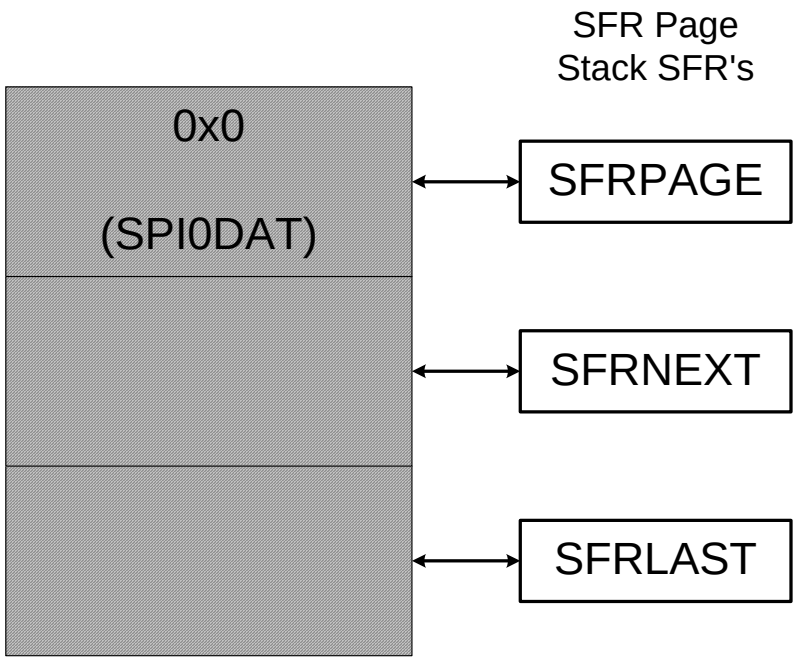

Figure 12.2. SFR Page Stack While Using SFR Page 0x0 To Access SPI0DAT

While CIP-51 executes in-line code (writing values to SPI0DAT in this example), the CAN0 Interrupt occurs. The CIP-51 vectors to the CAN0 ISR and pushes the current SFR Page value (SFR Page 0x00) into SFRNEXT in the SFR Page Stack. The SFR page needed to access CAN's SFRs is then automatically placed in the SFRPAGE register (SFR Page 0x0C). SFRPAGE is considered the "top" of the SFR Page Stack. Software can now access the CAN0 SFRs. Software may switch to any SFR Page by writing a new value to the SFRPAGE register at any time during the CAN0 ISR to access SFRs that are not on SFR Page 0x0C. See Figure 12.3.

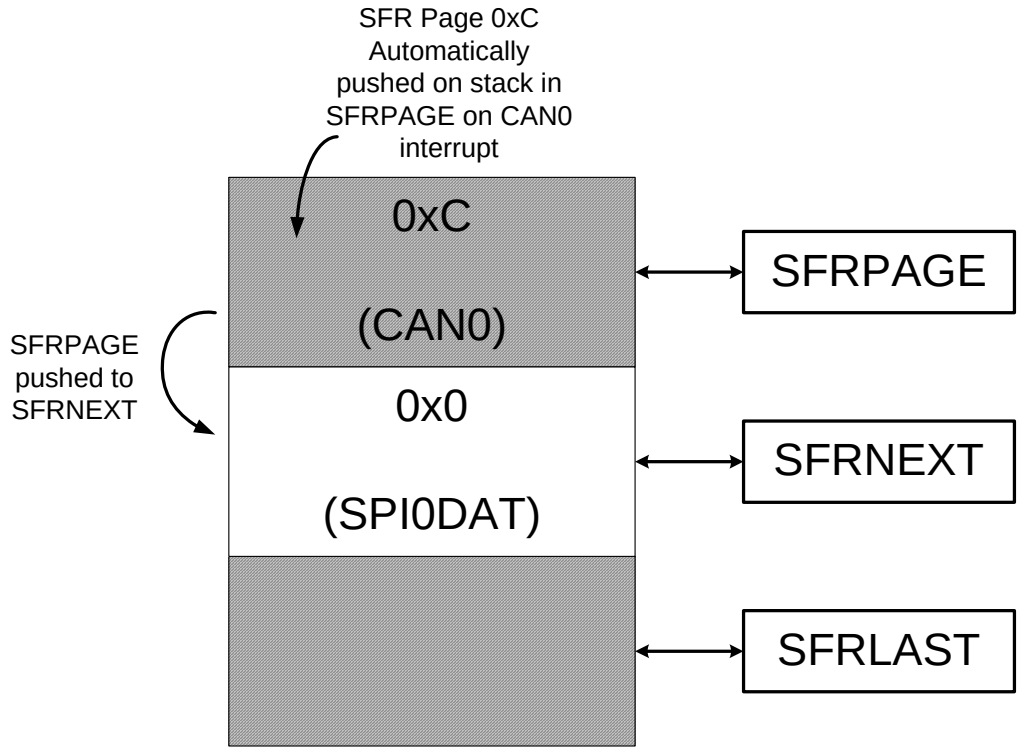

Figure 12.3. SFR Page Stack After CAN0 Interrupt Occurs

17.2. Configuring the External Memory Interface

Configuring the External Memory Interface consists of four steps:

1. Configure the Output Modes of the associated port pins as either push-pull or open-drain (push-pull is most common), and skip the associated pins in the crossbar.
2. Configure Port latches to “park” the EMIF pins in a dormant state (usually by setting them to logic 1).
3. Select the memory mode (on-chip only, split mode without bank select, split mode with bank select, or off-chip only).
4. Set up timing to interface with off-chip memory or peripherals.

Each of these four steps is explained in detail in the following sections. The Port selection and Mode bits are located in the EMI0CF register shown in SFR Definition .

17.3. Port Configuration

The External Memory Interface appears on Ports 1, 2 and 3 when it is used for off-chip memory access. These ports are multiplexed so that low-order address lines are shared with the data lines. When the EMIF is used, the Crossbar should be configured to skip over the /RD control line (P1.6) and the /WR control line (P1.7) using the P1SKIP register and also skip over the ALE control line (P1.5). For more information about configuring the Crossbar, see Section “19.6. Special Function Registers for Accessing and Configuring Port I/O” on page 183. The EMIF pinout is shown inTable 17.1 on page 146.

The External Memory Interface claims the associated Port pins for memory operations ONLY during the execution of an off-chip MOVX instruction. Once the MOVX instruction has completed, control of the Port pins reverts to the Port latches or to the Crossbar settings for those pins. See Section “19. Port Input/Output” on page 169 for more information about the Crossbar and Port operation and configuration. **The Port latches should be explicitly configured to “park” the External Memory Interface pins in a dormant state, most commonly by setting them to a logic 1**.

During the execution of the MOVX instruction, the External Memory Interface will explicitly disable the drivers on all Port pins that are acting as Inputs (Data[7:0] during a READ operation, for example). The Output mode of the Port pins (whether the pin is configured as Open-Drain or Push-Pull) is unaffected by the External Memory Interface operation, and remains controlled by the PnMDOUT registers. In most cases, the output modes of all EMIF pins should be configured for push-pull mode.

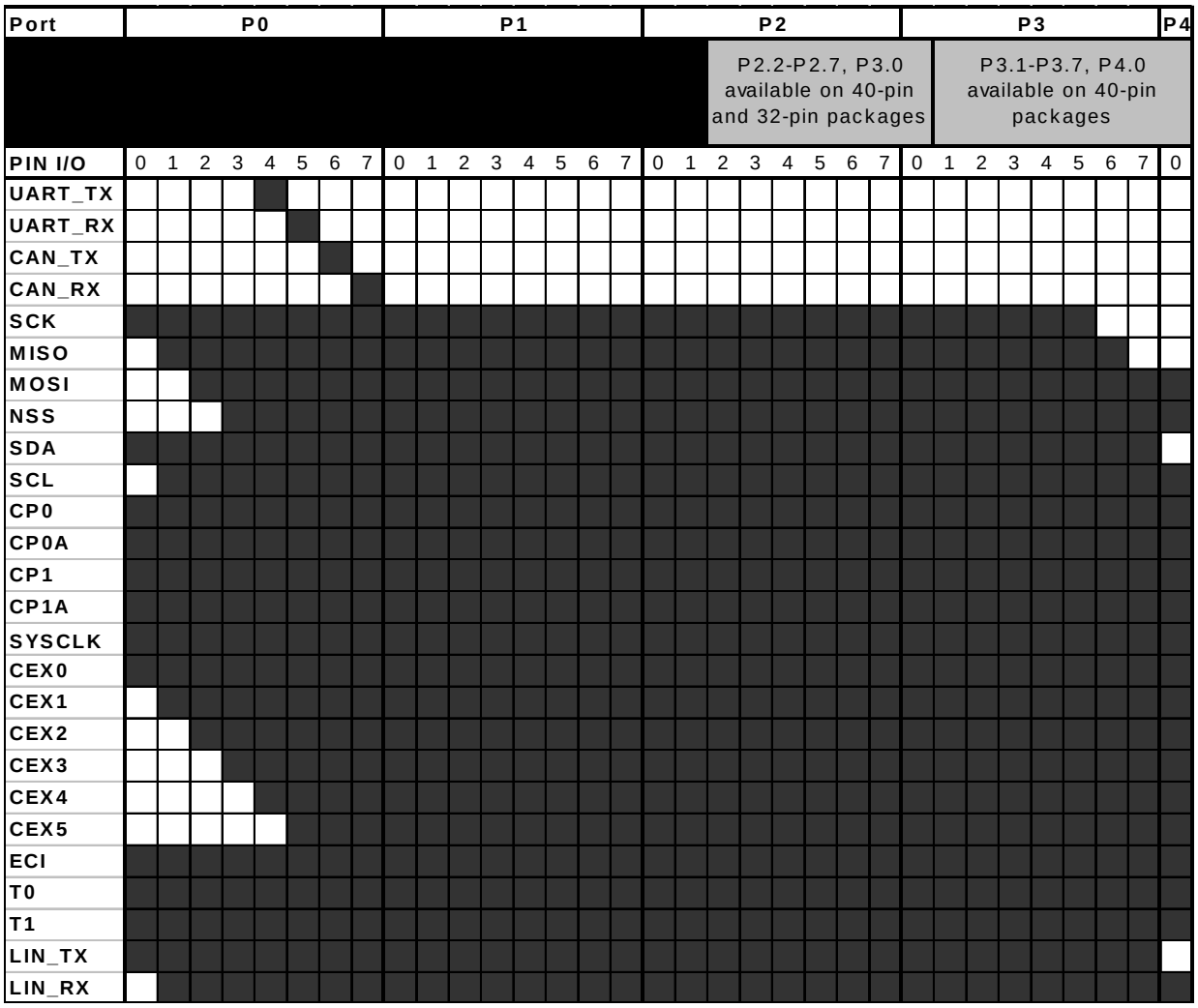

Figure 19.3. Peripheral Availability on Port I/O Pins

Registers XBR0, XBR1, and XBR2 are used to assign the digital I/O resources to the physical I/O Port pins. Note that when the SMBus is selected, the Crossbar assigns both pins associated with the SMBus (SDA and SCL); and similarly when the UART, CAN or LIN are selected, the Crossbar assigns both pins associated with the peripheral (TX and RX). UART0 pin assignments are fixed for bootloading purposes: UART TX0 is always assigned to P0.4; UART RX0 is always assigned to P0.5. CAN0 pin assignments are fixed to P0.6 for CAN_TX and P0.7 for CAN_RX. Standard Port I/Os appear contiguously after the prioritized functions have been assigned.

Important Note: The SPI can be operated in either 3-wire or 4-wire modes, pending the state of the NSSMD1–NSSMD0 bits in register SPI0CN. According to the SPI mode, the NSS signal may or may not be routed to a Port pin.

As an example configuration, if CAN0, SPI0 in 4-wire mode, and PCA0 Modules 0, 1, and 2 are enabled on the crossbar with P0.1, P0.2, and P0.5 skipped, the registers should be set as follows: XBR0 = 0x06 (CAN0 and SPI0 enabled), XBR1 = 0x0C (PCA0 modules 0, 1, and 2 enabled), XBR2 = 0x40 (Crossbar enabled), and P0SKIP = 0x26 (P0.1, P0.2, and P0.5 skipped). The resulting crossbar would look as shown in Figure 19.4.

LIN Register Definition 20.4. LIN0DTn: LIN0 Data Byte n

Bit	7	6	5	4	3	2	1	0
Name	DATAn[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

Indirect Address: LIN0DT1 = 0x00, LIN0DT2 = 0x01, LIN0DT3 = 0x02, LIN0DT4 = 0x03, LIN0DT5 = 0x04, LIN0DT6 = 0x05, LIN0DT7 = 0x06, LIN0DT8 = 0x07

Bit	Name	Function
7:0	DATAn[7:0]	**LIN Data Byte n.** Serial Data Byte that is received or transmitted across the LIN interface.

LIN Register Definition 20.8. LIN0SIZE: LIN0 Message Size Register

Bit	7	6	5	4	3	2	1	0
Name	ENHCHK				LINSIZE[3:0]			
Type	R/W	R	R	R	R/W			
Reset	0	0	0	0	0	0	0	0

Indirect Address = 0x0B

Bit	Name	Function
7	ENHCHK	**Checksum Selection Bit.** 0: Use the classic, specification 1.3 compliant checksum. Checksum covers the data bytes. 1: Use the enhanced, specification 2.0 compliant checksum. Checksum covers data bytes and protected identifier.
6:4	Unused	Read = 000b; Write = Don't Care
3:0	LINSIZE[3:0]	**Data Field Size.** 0000: 0 data bytes 0001: 1 data byte 0010: 2 data bytes 0011: 3 data bytes 0100: 4 data bytes 0101: 5 data bytes 0110: 6 data bytes 0111: 7 data bytes 1000: 8 data bytes 1001-1110: RESERVED 1111: Use the ID[1:0] bits (LIN0ID[5:4]) to determine the data length.

21. Controller Area Network (CAN0)

Important Documentation Note: The Bosch CAN Controller is integrated in the C8051F550/1/4/5, ‘F560/1/4/5/8/9, and ‘F572/3 devices. This section of the data sheet gives a description of the CAN controller as an overview and offers a description of how the Silicon Labs CIP-51 MCU interfaces with the on-chip Bosch CAN controller. In order to use the CAN controller, refer to Bosch’s C_CAN User’s Manual as an accompanying manual to the Silicon Labs’ data sheet.

The C8051F550/1/4/5, ‘F560/1/4/5/8/9, and ‘F572/3 devices feature a Control Area Network (CAN) controller that enables serial communication using the CAN protocol. Silicon Labs CAN facilitates communication on a CAN network in accordance with the Bosch specification 2.0A (basic CAN) and 2.0B (full CAN). The CAN controller consists of a CAN Core, Message RAM (separate from the CIP-51 RAM), a message handler state machine, and control registers. Silicon Labs CAN is a protocol controller and does not provide physical layer drivers (i.e., transceivers). Figure 21.1 shows an example typical configuration on a CAN bus.

Silicon Labs’ CAN operates at bit rates of up to 1 Mbit/second, though this can be limited by the physical layer chosen to transmit data on the CAN bus. The CAN processor has 32 Message Objects that can be configured to transmit or receive data. Incoming data, message objects and their identifier masks are stored in the CAN message RAM. All protocol functions for transmission of data and acceptance filtering is performed by the CAN controller and not by the CIP-51 MCU. In this way, minimal CPU bandwidth is needed to use CAN communication. The CIP-51 configures the CAN controller, accesses received data, and passes data for transmission via Special Function Registers (SFRs) in the CIP-51.

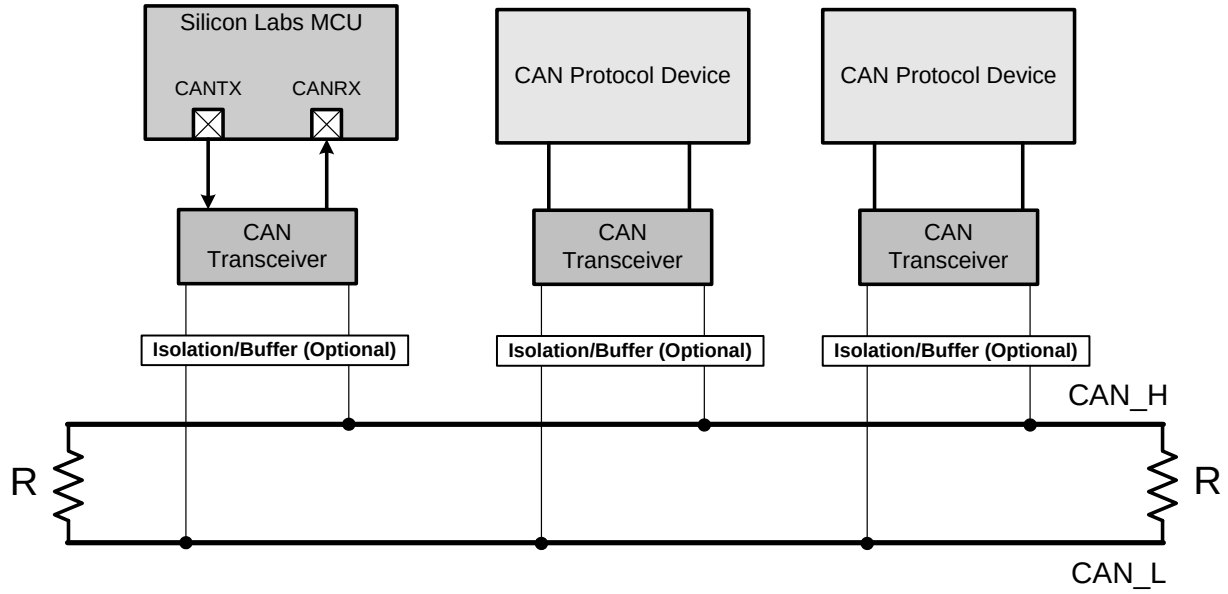

Figure 21.1. Typical CAN Bus Configuration

22.4.3. Data Register

The SMBus Data register SMB0DAT holds a byte of serial data to be transmitted or one that has just been received. Software may safely read or write to the data register when the SI flag is set. Software should not attempt to access the SMB0DAT register when the SMBus is enabled and the SI flag is cleared to logic 0, as the interface may be in the process of shifting a byte of data into or out of the register.

Data in SMB0DAT is always shifted out MSB first. After a byte has been received, the first bit of received data is located at the MSB of SMB0DAT. While data is being shifted out, data on the bus is simultaneously being shifted in. SMB0DAT always contains the last data byte present on the bus. In the event of lost arbitration, the transition from master transmitter to slave receiver is made with the correct data or address in SMB0DAT.

SFR Definition 22.3. SMB0DAT: SMBus Data

Bit	7	6	5	4	3	2	1	0
Name	SMB0DAT[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xC2; SMB0DAT = 0x00

Bit	Name	Function
7:0	SMB0DAT[7:0]	**SMBus Data.** The SMB0DAT register contains a byte of data to be transmitted on the SMBus serial interface or a byte that has just been received on the SMBus serial interface. The CPU can read from or write to this register whenever the SI serial interrupt flag (SMB0CN.0) is set to logic 1. The serial data in the register remains stable as long as the SI flag is set. When the SI flag is not set, the system may be in the process of shifting data in/out and the CPU should not attempt to access this register.

22.5. SMBus Transfer Modes

The SMBus interface may be configured to operate as master and/or slave. At any particular time, it will be operating in one of the following four modes: Master Transmitter, Master Receiver, Slave Transmitter, or Slave Receiver. The SMBus interface enters Master Mode any time a START is generated, and remains in Master Mode until it loses an arbitration or generates a STOP. An SMBus interrupt is generated at the end of all SMBus byte frames. As a receiver, the interrupt for an ACK occurs **before** the ACK. As a transmitter, interrupts occur **after** the ACK.

SFR Definition 23.5. SBRLH0: UART0 Baud Rate Generator Reload High Byte

Bit	7	6	5	4	3	2	1	0
Name	SBRLH0[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xAD; SFR Page = 0x0F

Bit	Name	Function
7:0	SBRLH0[7:0]	**High Byte of Reload Value for UART0 Baud Rate Generator.** This value is loaded into the high byte of the UART0 baud rate generator when the counter overflows from 0xFFFF to 0x0000.

SFR Definition 23.6. SBRLL0: UART0 Baud Rate Generator Reload Low Byte

Bit	7	6	5	4	3	2	1	0
Name	SBRLL0[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xAC; SFR Page = 0x0F

Bit	Name	Function
7:0	SBRLL0[7:0]	**Low Byte of Reload Value for UART0 Baud Rate Generator.** This value is loaded into the low byte of the UART0 baud rate generator when the counter overflows from 0xFFFF to 0x0000.

24.5. Serial Clock Phase and Polarity

Four combinations of serial clock phase and polarity can be selected using the clock control bits in the SPI0 Configuration Register (SPI0CFG). The CKPHA bit (SPI0CFG.5) selects one of two clock phases (edge used to latch the data). The CKPOL bit (SPI0CFG.4) selects between an active-high or active-low clock. Both master and slave devices must be configured to use the same clock phase and polarity. SPI0 should be disabled (by clearing the SPIEN bit, SPI0CN.0) when changing the clock phase or polarity. The clock and data line relationships for master mode are shown in Figure 24.5. For slave mode, the clock and data relationships are shown in Figure 24.6 and Figure 24.7. CKPHA must be set to 0 on both the master and slave SPI when communicating between two of the following devices: C8051F04x, C8051F06x, C8051F12x, C8051F31x, C8051F32x, and C8051F33x.

The SPI0 Clock Rate Register (SPI0CKR) as shown in SFR Definition 24.3 controls the master mode serial clock frequency. This register is ignored when operating in slave mode. When the SPI is configured as a master, the maximum data transfer rate (bits/sec) is one-half the system clock frequency or 12.5 MHz, whichever is slower. When the SPI is configured as a slave, the maximum data transfer rate (bits/sec) for full-duplex operation is 1/10 the system clock frequency, provided that the master issues SCK, NSS (in 4-wire slave mode), and the serial input data synchronously with the slave's system clock. If the master issues SCK, NSS, and the serial input data asynchronously, the maximum data transfer rate (bits/sec) must be less than 1/10 the system clock frequency. In the special case where the master only wants to transmit data to the slave and does not need to receive data from the slave (i.e. half-duplex operation), the SPI slave can receive data at a maximum data transfer rate (bits/sec) of 1/4 the system clock frequency. This is provided that the master issues SCK, NSS, and the serial input data synchronously with the slave's system clock.

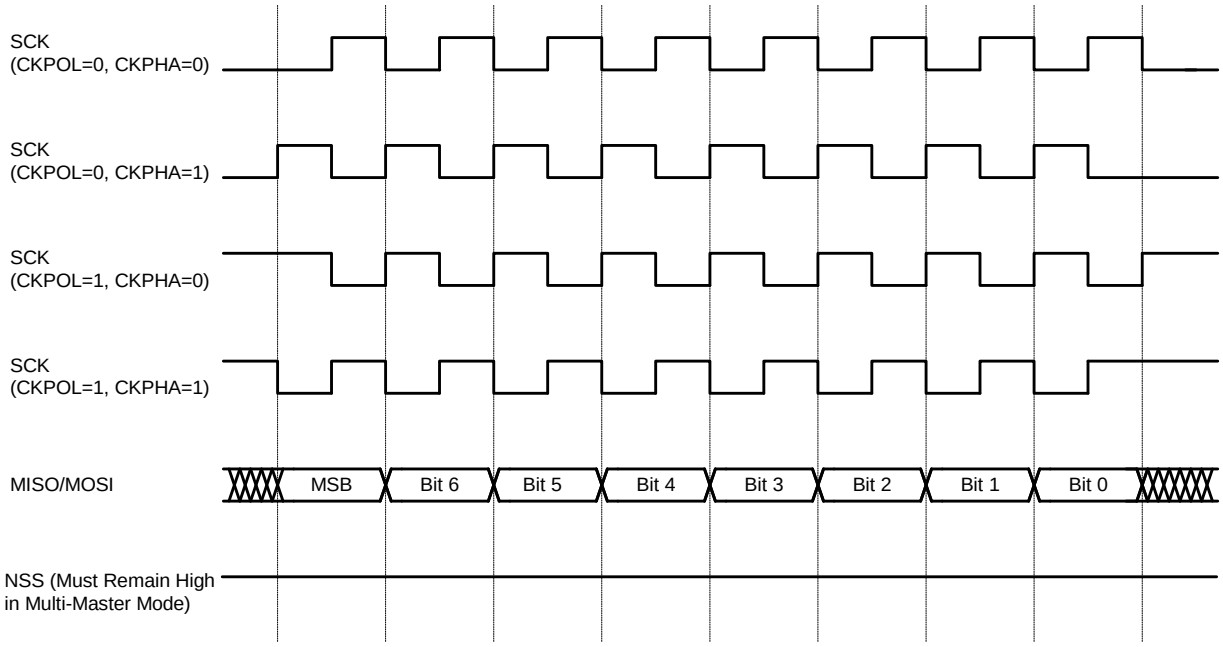

Figure 24.5. Master Mode Data/Clock Timing

Table 24.1. SPI Slave Timing Parameters

Parameter	Description	Min	Max	Units
Master Mode Timing* (See Figure 24.8 and Figure 24.9)				
T_{MCKH}	SCK High Time	1 x T_{SYSCLK}	—	ns
T_{MCKL}	SCK Low Time	1 x T_{SYSCLK}	—	ns
T_{MIS}	MISO Valid to SCK Shift Edge	1 x T_{SYSCLK} + 20	—	ns
T_{MIH}	SCK Shift Edge to MISO Change	0	—	ns
Slave Mode Timing* (See Figure 24.10 and Figure 24.11)				
T_{SE}	NSS Falling to First SCK Edge	2 x T_{SYSCLK}	—	ns
T_{SD}	Last SCK Edge to NSS Rising	2 x T_{SYSCLK}	—	ns
T_{SEZ}	NSS Falling to MISO Valid	—	4 x T_{SYSCLK}	ns
T_{SDZ}	NSS Rising to MISO High-Z	—	4 x T_{SYSCLK}	ns
T_{CKH}	SCK High Time	5 x T_{SYSCLK}	—	ns
T_{CKL}	SCK Low Time	5 x T_{SYSCLK}	—	ns
T_{SIS}	MOSI Valid to SCK Sample Edge	2 x T_{SYSCLK}	—	ns
T_{SIH}	SCK Sample Edge to MOSI Change	2 x T_{SYSCLK}	—	ns
T_{SOH}	SCK Shift Edge to MISO Change	—	4 x T_{SYSCLK}	ns
T_{SLH}	Last SCK Edge to MISO Change (CKPHA = 1 ONLY)	6 x T_{SYSCLK}	8 x T_{SYSCLK}	ns

***Note:** T_{SYSCLK} is equal to one period of the device system clock (SYSCLK).

SFR Definition 25.6. TH0: Timer 0 High Byte

Bit	7	6	5	4	3	2	1	0
Name	TH0[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0x8C; SFR Page = All Pages

Bit	Name	Function
7:0	TH0[7:0]	**Timer 0 High Byte.** The TH0 register is the high byte of the 16-bit Timer 0.

SFR Definition 25.7. TH1: Timer 1 High Byte

Bit	7	6	5	4	3	2	1	0
Name	TH1[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0x8D; SFR Page = All Pages

Bit	Name	Function
7:0	TH1[7:0]	**Timer 1 High Byte.** The TH1 register is the high byte of the 16-bit Timer 1.

SFR Definition 25.9. TMR2RLL: Timer 2 Reload Register Low Byte

Bit	7	6	5	4	3	2	1	0
Name	TMR2RLL[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xCA; SFR Page = 0x00

Bit	Name	Function
7:0	TMR2RLL[7:0]	**Timer 2 Reload Register Low Byte.** TMR2RLL holds the low byte of the reload value for Timer 2.

SFR Definition 25.10. TMR2RLH: Timer 2 Reload Register High Byte

Bit	7	6	5	4	3	2	1	0
Name	TMR2RLH[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xCB; SFR Page = 0x00

Bit	Name	Function
7:0	TMR2RLH[7:0]	**Timer 2 Reload Register High Byte.** TMR2RLH holds the high byte of the reload value for Timer 2.

SFR Definition 25.16. TMR3L: Timer 3 Low Byte

Bit	7	6	5	4	3	2	1	0
Name	TMR3L[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0x94; SFR Page = 0x00

Bit	Name	Function
7:0	TMR3L[7:0]	**Timer 3 Low Byte.** In 16-bit mode, the TMR3L register contains the low byte of the 16-bit Timer 3. In 8-bit mode, TMR3L contains the 8-bit low byte timer value.

SFR Definition 25.17. TMR3H Timer 3 High Byte

Bit	7	6	5	4	3	2	1	0
Name	TMR3H[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0x95; SFR Page = 0x00

Bit	Name	Function
7:0	TMR3H[7:0]	**Timer 3 High Byte.** In 16-bit mode, the TMR3H register contains the high byte of the 16-bit Timer 3. In 8-bit mode, TMR3H contains the 8-bit high byte timer value.

DOCUMENT CHANGE LIST

Revision 0.5 to Revision 1.0

- Updated “2. Ordering Information” to include -A (Automotive) devices and automotive qualification information.
- Updated Figure 4.8 on page 35.
- Updated supply current related specifications throughout “5. Electrical Characteristics” .
- Updated SFR Definition 7.1 to change VREF high setting to 2.20 V from 2.25 V.
- Updated Figure 8.1 to indicate that Comparators are powered from V_{IO} and not V_{DDA}.
- Updated the Gain Table in “6.3.1. Calculating the Gain Value” to fix the ADC0GNH Value in the last row.
- Updated Table 10.1 with correct timing for all branch instructions, MOVC, and CPL A.
- Updated “14.2. Non-volatile Data Storage” to clarify behavior of 8-bit MOVX instructions and when writing/erasing Flash.
- Updated SFR Definition 14.3 (FLSCL) to include FLEWT bit definition. This bit must be set before writing or erasing Flash. Also updated Table 5.5 to reflect new Flash Write and Erase timing.
- Updated “16.7. Flash Error Reset” with an additional cause of a Flash Error reset.
- Updated “19.1.3. Interfacing Port I/O in a Multi-Voltage System” to remove note regarding interfacing to voltages above VIO.
- Updated “22. SMBus” to remove all hardware ACK features, including SMB0ADM and SMB0ADR SFRs.
- Updated SFR Definition 23.1 (SCON0) to correct SFR Page to 0x00 from All Pages.
- All items from the C8051F55x-F56x-57x Errata dated November 5th, 2009 are incorporated into this data sheet.

Revision 1.0 to Revision 1.1

- Updated “1. System Overview” with a voltage range specification for the internal oscillator.
- Updated Table 5.6, “Internal High-Frequency Oscillator Electrical Characteristics,” on page 42 with new conditions for the internal oscillator accuracy. The internal oscillator accuracy is dependent on the operating voltage range.
- Updated “5. Electrical Characteristics” to remove the internal oscillator curve across temperature diagram.
- Updated Figure 6.4 on Page 51 with new timing diagram when using CNVSTR pin.
- Updated SFR Definition 7.1 (REF0CN) with oscillator suspend requirement for ZTCEN.
- Fixed incorrect cross references in “8. Comparators” .
- Updated SFR Definition 9.1 (REG0CN) with a new definition for Bit 6. The bit 6 reset value is 1b and must be written to 1b.
- Update “15.3. Suspend Mode” with note regarding ZTCEN.
- Added Port 2 Event and Port 3 Events to wake-up sources in “18.2.1. Internal Oscillator Suspend Mode”
- Updated “20. Local Interconnect Network (LIN0)” with a voltage range specification for the internal oscillator.
- Updated LIN Register Definitions 20.9 and 20.10 with correct reset values.
- Updated “21. Controller Area Network (CAN0)” with a voltage range specification for the internal oscillator.
- Updated C2 Register Definitions 27.2 and 27.3 with correct C2 and SFR Addresses.

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS®

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com