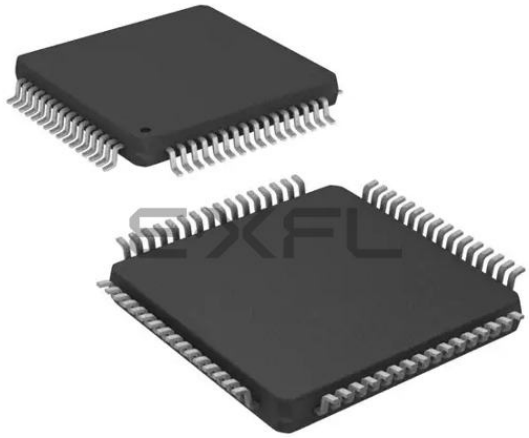


Welcome to E-XFL.COM

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	Not For New Designs
Core Processor	8051
Core Size	8-Bit
Speed	50MHz
Connectivity	EBI/EMI, SMBus (2-Wire/I²C), SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, Temp Sensor, WDT
Number of I/O	32
Program Memory Size	128KB (128K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	8.25K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 3.6V
Data Converters	A/D 8x8b, 8x12b; D/A 2x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-TQFP
Supplier Device Package	64-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/silicon-labs/c8051f125-gqr

1.3. JTAG Debug and Boundary Scan

JTAG boundary scan and debug circuitry is included which provides *non-intrusive, full speed, in-circuit debugging using the production part installed in the end application*, via the four-pin JTAG interface. The JTAG port is fully compliant to IEEE 1149.1, providing full boundary scan for test and manufacturing purposes.

Silicon Labs' debugging system supports inspection and modification of memory and registers, breakpoints, watchpoints, a stack monitor, and single stepping. No additional target RAM, program memory, timers, or communications channels are required. All the digital and analog peripherals are functional and work correctly while debugging. All the peripherals (except for the ADC and SMBus) are stalled when the MCU is halted, during single stepping, or at a breakpoint in order to keep them synchronized.

The C8051F120DK development kit provides all the hardware and software necessary to develop application code and perform in-circuit debugging with the C8051F12x or C8051F13x MCUs.

The kit includes a Windows (95 or later) development environment, a serial adapter for connecting to the JTAG port, and a target application board with a C8051F120 MCU installed. All of the necessary communication cables and a wall-mount power supply are also supplied with the development kit. Silicon Labs' debug environment is a vastly superior configuration for developing and debugging embedded applications compared to standard MCU emulators, which use on-board "ICE Chips" and target cables and require the MCU in the application board to be socketed. Silicon Labs' debug environment both increases ease of use and preserves the performance of the precision, on-chip analog peripherals.

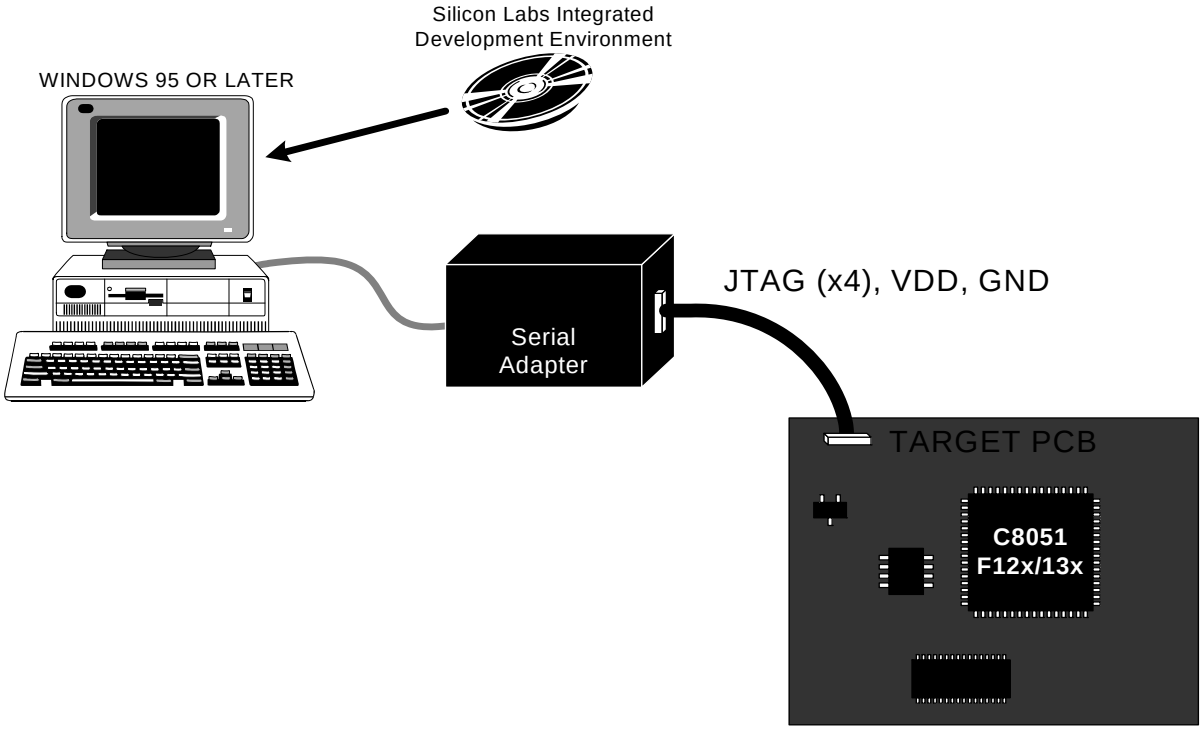


Figure 1.9. Development/In-System Debug Diagram

The Temperature Sensor transfer function is shown in Figure 5.2. The output voltage (V_{TEMP}) is the PGA input when the Temperature Sensor is selected by bits AMX0AD3-0 in register AMX0SL; this voltage will be amplified by the PGA according to the user-programmed PGA settings. Typical values for the Slope and Offset parameters can be found in Table 5.1.

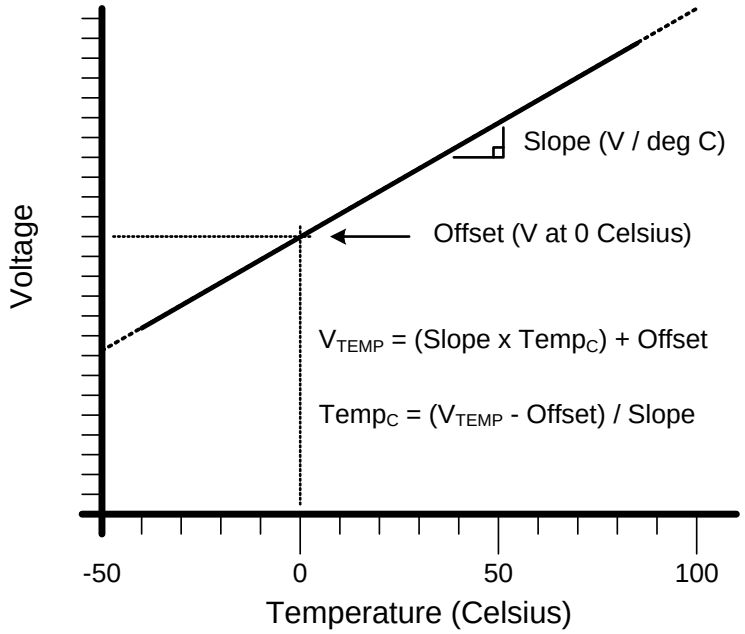
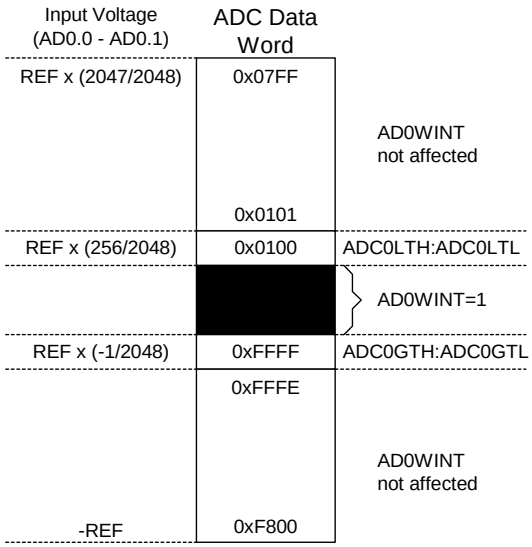
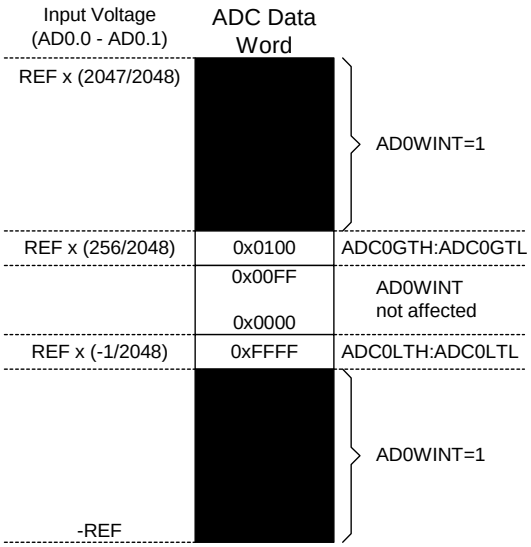


Figure 5.2. Typical Temperature Sensor Transfer Function



Given:
AMX0SL = 0x00, AMX0CF = 0x01,
AD0LJST = '0',
ADC0LTH:ADC0LTL = 0x0100,
ADC0GTH:ADC0GTL = 0xFFFF.
An ADC0 End of Conversion will cause an ADC0 Window Compare Interrupt (AD0WINT = '1') if the resulting ADC0 Data Word is < 0x0100 and > 0xFFFF. (In 2s-complement math, 0xFFFF = -1.)



Given:
AMX0SL = 0x00, AMX0CF = 0x01,
AD0LJST = '0',
ADC0LTH:ADC0LTL = 0xFFFF,
ADC0GTH:ADC0GTL = 0x0100.
An ADC0 End of Conversion will cause an ADC0 Window Compare Interrupt (AD0WINT = '1') if the resulting ADC0 Data Word is < 0xFFFF or > 0x0100. (In 2s-complement math, 0xFFFF = -1.)

Figure 5.7. 12-Bit ADC0 Window Interrupt Example: Right Justified Differential Data

NOTES:

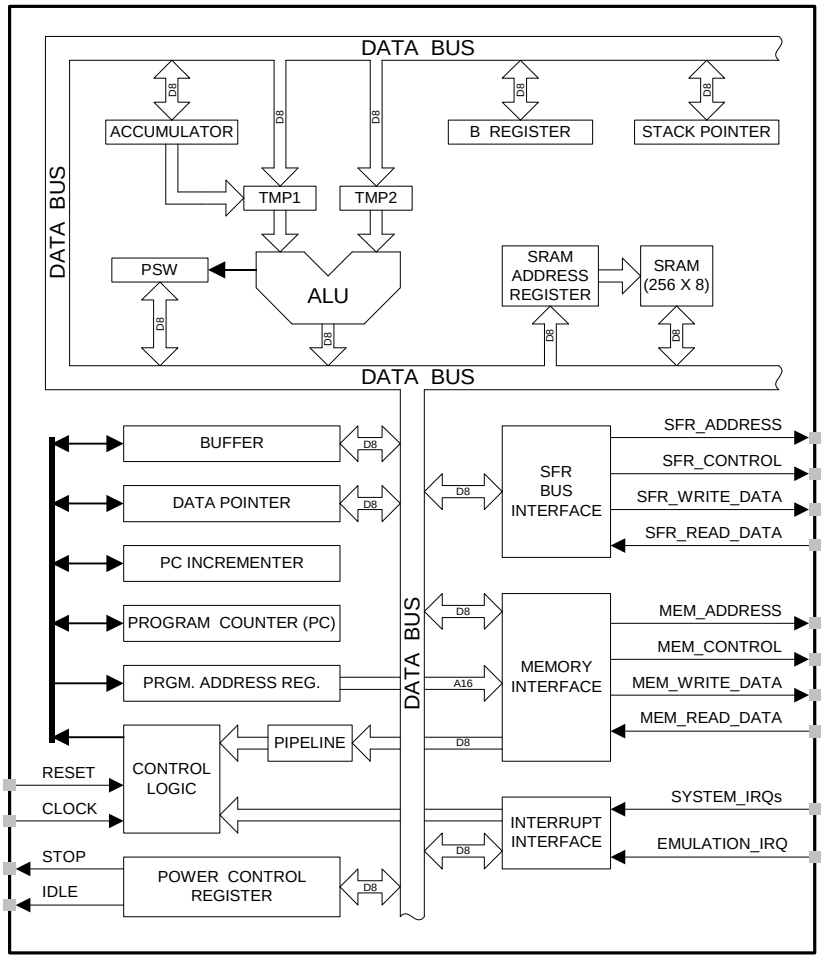


Figure 11.1. CIP-51 Block Diagram

Programming and Debugging Support

A JTAG-based serial interface is provided for in-system programming of the Flash program memory and communication with on-chip debug support logic. The re-programmable Flash can also be read and changed by the application software using the MOVC and MOVX instructions. This feature allows program memory to be used for non-volatile data storage as well as updating program code under software control.

The on-chip debug support logic facilitates full speed in-circuit debugging, allowing the setting of hardware breakpoints and watch points, starting, stopping and single stepping through program execution (including interrupt service routines), examination of the program's call stack, and reading/writing the contents of registers and memory. This method of on-chip debug is completely non-intrusive and non-invasive, requiring no RAM, Stack, timers, or other on-chip resources.

The CIP-51 is supported by development tools from Silicon Labs and third party vendors. Silicon Labs provides an integrated development environment (IDE) including editor, macro assembler, debugger and programmer. The IDE's debugger and programmer interface to the CIP-51 via its JTAG interface to provide fast and efficient in-system device programming and debugging. Third party macro assemblers and C compilers are also available.

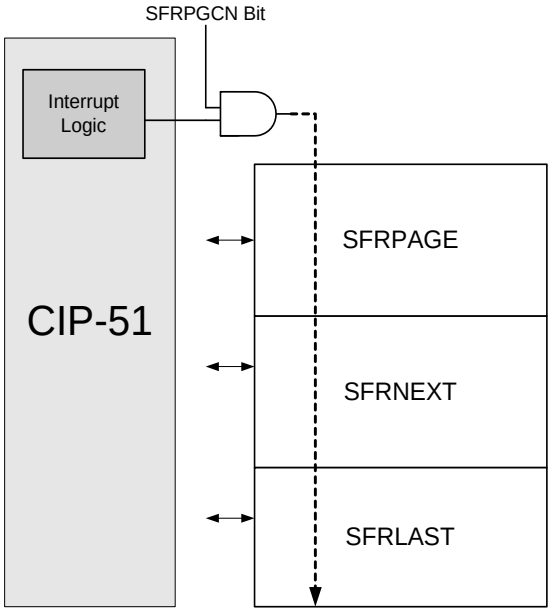


Figure 11.4. SFR Page Stack

Automatic hardware switching of the SFR Page on interrupts may be enabled or disabled as desired using the SFR Automatic Page Control Enable Bit located in the SFR Page Control Register (SFRPGCN). This function defaults to ‘enabled’ upon reset. In this way, the autoswitching function will be enabled unless disabled in software.

A summary of the SFR locations (address and SFR page) is provided in Table 11.2. in the form of an SFR memory map. Each memory location in the map has an SFR page row, denoting the page in which that SFR resides. Note that certain SFR’s are accessible from ALL SFR pages, and are denoted by the “**(ALL PAGES)**” designation. For example, the Port I/O registers P0, P1, P2, and P3 all have the “**(ALL PAGES)**” designation, indicating these SFR’s are accessible from all SFR pages regardless of the SFRPAGE register value.

Table 11.3. Special Function Registers (Continued)

SFRs are listed in alphabetical order. All undefined SFR locations are reserved.

Register	Address	SFR Page	Description	Page No.
TMR4L	0xCC	2	Timer/Counter 4 Low Byte	page 323
WDTCN	0xFF	All Pages	Watchdog Timer Control	page 181
XBR0	0xE1	F	Port I/O Crossbar Control 0	page 245
XBR1	0xE2	F	Port I/O Crossbar Control 1	page 246
XBR2	0xE3	F	Port I/O Crossbar Control 2	page 247

Notes:

1. Refers to a register in the C8051F120/1/4/5 only.
2. Refers to a register in the C8051F122/3/6/7 and C8051F130/1/2/3 only.
3. Refers to a register in the C8051F120/1/2/3/4/5/6/7 only.
4. Refers to a register in the C8051F120/1/2/3 and C8051F130/1/2/3 only.
5. Refers to a register in the C8051F120/2/4/6 only.
6. Refers to a register in the C8051F121/3/5/7 only.
7. Refers to a register in the C8051F130/1/2/3 only.

11.3.5. Interrupt Register Descriptions

The SFRs used to enable the interrupt sources and set their priority level are described below. Refer to the datasheet section associated with a particular on-chip peripheral for information regarding valid interrupt conditions for the peripheral and the behavior of its interrupt-pending flag(s).

SFR Definition 11.12. IE: Interrupt Enable

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
EA	IEGF0	ET2	ES0	ET1	EX1	ET0	EX0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Bit Addressable

SFR Address: 0xA8
SFR Page: All Pages

Bit7: EA: Enable All Interrupts.
This bit globally enables/disables all interrupts. It overrides the individual interrupt mask settings.
0: Disable all interrupt sources.
1: Enable each interrupt according to its individual mask setting.

Bit6: IEGF0: General Purpose Flag 0.
This is a general purpose flag for use under software control.

Bit5: ET2: Enabler Timer 2 Interrupt.
This bit sets the masking of the Timer 2 interrupt.
0: Disable Timer 2 interrupt.
1: Enable Timer 2 interrupt.

Bit4: ES0: Enable UART0 Interrupt.
This bit sets the masking of the UART0 interrupt.
0: Disable UART0 interrupt.
1: Enable UART0 interrupt.

Bit3: ET1: Enable Timer 1 Interrupt.
This bit sets the masking of the Timer 1 interrupt.
0: Disable Timer 1 interrupt.
1: Enable Timer 1 interrupt.

Bit2: EX1: Enable External Interrupt 1.
This bit sets the masking of External Interrupt 1.
0: Disable External Interrupt 1.
1: Enable External Interrupt 1.

Bit1: ET0: Enable Timer 0 Interrupt.
This bit sets the masking of the Timer 0 interrupt.
0: Disable Timer 0 interrupts.
1: Enable Timer 0 interrupts.

Bit0: EX0: Enable External Interrupt 0.
This bit sets the masking of External Interrupt 0.
0: Disable External Interrupt 0.
1: Enable External Interrupt 0.

$$\text{PLL Frequency} = \text{Reference Frequency} \times \frac{\text{PLLN}}{\text{PLLM}}$$

14.7.3. Powering on and Initializing the PLL

To set up and use the PLL as the system clock after power-up of the device, the following procedure should be implemented:

Step 1. Ensure that the reference clock to be used (internal or external) is running and stable.
Step 2. Set the PLLSRC bit (PLL0CN.2) to select the desired clock source for the PLL.
Step 3. Program the Flash read timing bits, FLRT (FLSCL.5–4) to the appropriate value for the new clock rate (see **Section “15. Flash Memory” on page 199**).
Step 4. Enable power to the PLL by setting PLLPWR (PLL0CN.0) to ‘1’.
Step 5. Program the PLL0DIV register to produce the divided reference frequency to the PLL.
Step 6. Program the PLLLP3–0 bits (PLL0FLT.3–0) to the appropriate range for the divided reference frequency.
Step 7. Program the PLLICO1–0 bits (PLL0FLT.5–4) to the appropriate range for the PLL output frequency.
Step 8. Program the PLL0MUL register to the desired clock multiplication factor.
Step 9. Wait at least 5 µs, to provide a fast frequency lock.
Step 10. Enable the PLL by setting PLLEN (PLL0CN.1) to ‘1’.
Step 11. Poll PLLLCK (PLL0CN.4) until it changes from ‘0’ to ‘1’.
Step 12. Switch the System Clock source to the PLL using the CLKSEL register.

If the PLL characteristics need to be changed when the PLL is already running, the following procedure should be implemented:

Step 1. The system clock should first be switched to either the internal oscillator or an external clock source that is running and stable, using the CLKSEL register.
Step 2. Ensure that the reference clock to be used for the new PLL setting (internal or external) is running and stable.
Step 3. Set the PLLSRC bit (PLL0CN.2) to select the new clock source for the PLL.
Step 4. If moving to a faster frequency, program the Flash read timing bits, FLRT (FLSCL.5–4) to the appropriate value for the new clock rate (see **Section “15. Flash Memory” on page 199**).
Step 5. Disable the PLL by setting PLLEN (PLL0CN.1) to ‘0’.
Step 6. Program the PLL0DIV register to produce the divided reference frequency to the PLL.
Step 7. Program the PLLLP3–0 bits (PLL0FLT.3–0) to the appropriate range for the divided reference frequency.
Step 8. Program the PLLICO1-0 bits (PLL0FLT.5–4) to the appropriate range for the PLL output frequency.
Step 9. Program the PLL0MUL register to the desired clock multiplication factor.
Step 10. Enable the PLL by setting PLLEN (PLL0CN.1) to ‘1’.
Step 11. Poll PLLLCK (PLL0CN.4) until it changes from ‘0’ to ‘1’.
Step 12. Switch the System Clock source to the PLL using the CLKSEL register.
Step 13. If moving to a slower frequency, program the Flash read timing bits, FLRT (FLSCL.5–4) to the appropriate value for the new clock rate (see **Section “15. Flash Memory” on**

page 199). **Important Note: Cache reads, cache writes, and the prefetch engine should be disabled whenever the FLRT bits are changed to a lower setting.**

To shut down the PLL, the system clock should be switched to the internal oscillator or a stable external clock source, using the CLKSEL register. Next, disable the PLL by setting PLLEN (PLL0CN.1) to '0'. Finally, the PLL can be powered off, by setting PLLPWR (PLL0CN.0) to '0'. Note that the PLLEN and PLLPWR bits can be cleared at the same time.

SFR Definition 14.5. PLL0CN: PLL Control

R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	Reset Value
-	-	-	PLLLCK	0	PLLSRC	PLLEN	PLLPWR	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

SFR Address: 0x89
SFR Page: F

Bits 7–5: UNUSED: Read = 000b; Write = don't care.
Bit 4: PLLCK: PLL Lock Flag.
0: PLL Frequency is not locked.
1: PLL Frequency is locked.
Bit 3: RESERVED. Must write to '0'.
Bit 2: PLLSRC: PLL Reference Clock Source Select Bit.
0: PLL Reference Clock Source is Internal Oscillator.
1: PLL Reference Clock Source is External Oscillator.
Bit 1: PLLEN: PLL Enable Bit.
0: PLL is held in reset.
1: PLL is enabled. PLLPWR must be '1'.
Bit 0: PLLPWR: PLL Power Enable.
0: PLL bias generator is de-activated. No static power is consumed.
1: PLL bias generator is active. Must be set for PLL to operate.

15.2. Security Options

The CIP-51 provides security options to protect the Flash memory from inadvertent modification by software as well as prevent the viewing of proprietary program code and constants. The Program Store Write Enable (PSCTL.0), Program Store Erase Enable (PSCTL.1), and Flash Write/Erase Enable (FLACL.0) bits protect the Flash memory from accidental modification by software. These bits must be explicitly set to logic 1 before software can write or erase the Flash memory. Additional security features prevent proprietary program code and data constants from being read or altered across the JTAG interface or by software running on the system controller.

A set of security lock bytes protect the Flash program memory from being read or altered across the JTAG interface. Each bit in a security lock-byte protects one 16k-byte block of memory. Clearing a bit to logic 0 in the Read Lock Byte prevents the corresponding block of Flash memory from being read across the JTAG interface. Clearing a bit in the Write/Erase Lock Byte protects the block from JTAG erasures and/or writes. The Scratchpad area is read or write/erase locked when all bits in the corresponding security byte are cleared to logic 0.

On the C8051F12x and C8051F130/1, the security lock bytes are located at 0x1FBFE (Write/Erase Lock) and 0x1FBFF (Read Lock), as shown in Figure 15.2. On the C8051F132/3, the security lock bytes are located at 0x0FFFE (Write/Erase Lock) and 0x0FFFF (Read Lock), as shown in Figure 15.3. The 1024-byte sector containing the lock bytes can be written to, but not erased, by software. An attempted read of a read-locked byte returns undefined data. Debugging code in a read-locked sector is not possible through the JTAG interface. The lock bits can always be read from and written to logic 0 regardless of the security setting applied to the block containing the security bytes. This allows additional blocks to be protected after the block containing the security bytes has been locked.

Important Note: To ensure protection from external access, the block containing the lock bytes must be Write/Erase locked. On the 128 kB devices (C8051F12x and C8051F130/1), the block containing the security bytes is 0x18000-0x1BFFF, and is locked by clearing bit 7 of the Write/Erase Lock Byte. On the 64 kB devices (C8051F132/3), the block containing the security bytes is 0x0C000-0x0FFFF, and is locked by clearing bit 3 of the Write/Erase Lock Byte. If the page containing the security bytes is not Write/Erase locked, it is still possible to erase this page of Flash memory through the JTAG port and reset the security bytes.

When the page containing the security bytes has been Write/Erase locked, a JTAG full device erase must be performed to unlock any areas of Flash protected by the security bytes. A JTAG full device erase is initiated by performing a normal JTAG erase operation on either of the security byte locations. This operation must be initiated through the JTAG port, and cannot be performed from firmware running on the device.

17.6.2.2.8-bit MOVX without Bank Select: EMI0CF[4:2] = ‘001’ or ‘011’.

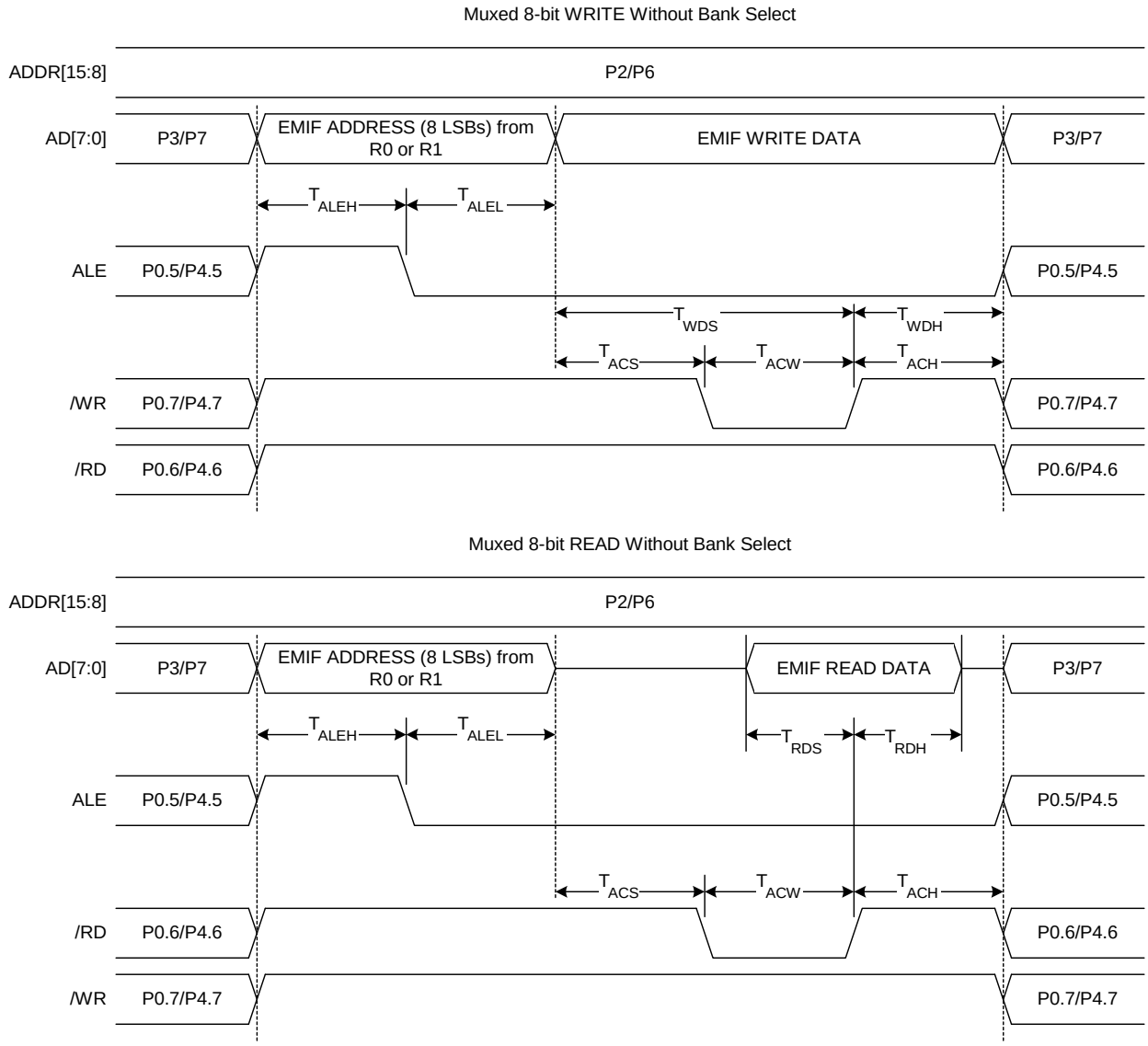
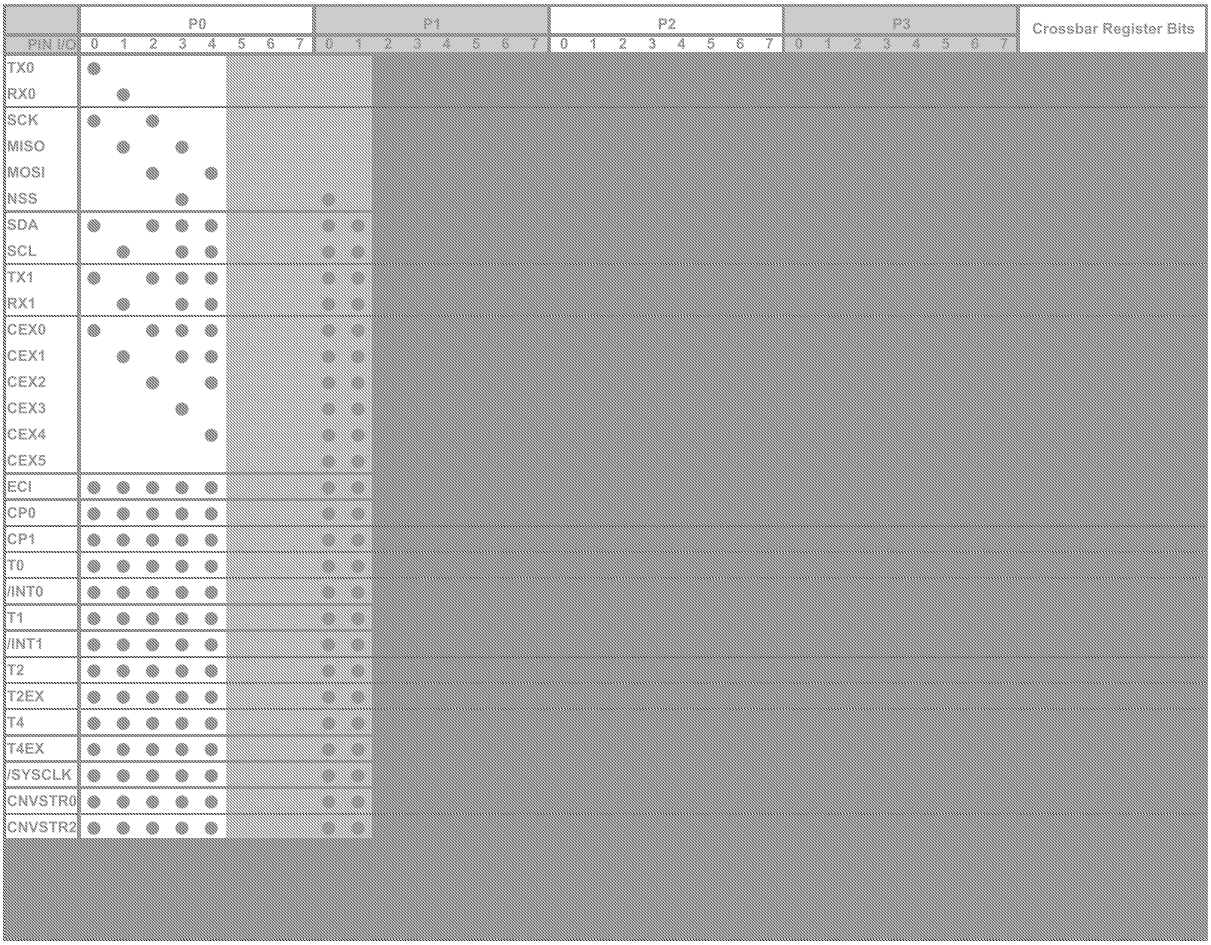


Figure 17.8. Multiplexed 8-bit MOVX without Bank Select Timing



**(EMIFLE = 1; EMIF in Multiplexed Mode; P1MDIN = 0xE3;
XBR0 = 0x05; XBR1 = 0x14; XBR2 = 0x42)**

Figure 18.6. Crossbar Example

SFR Definition 20.3. SPI0CKR: SPI0 Clock Rate

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
SCR7	SCR6	SCR5	SCR4	SCR3	SCR2	SCR1	SCR0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

SFR Address: 0x9D
SFR Page: 0

Bits 7–0: SCR7–SCR0: SPI0 Clock Rate.
These bits determine the frequency of the SCK output when the SPI0 module is configured for master mode operation. The SCK clock frequency is a divided version of the system clock, and is given in the following equation, where *SYSCLK* is the system clock frequency and *SPI0CKR* is the 8-bit value held in the SPI0CKR register.

$$f_{SCK} = \frac{\text{SYSCLK}}{2 \times (\text{SPI0CKR} + 1)}$$

for 0 <= SPI0CKR <= 255

Example: If SYSCLK = 2 MHz and SPI0CKR = 0x04,

$$f_{SCK} = \frac{2000000}{2 \times (4 + 1)}$$

$$f_{SCK} = 200kHz$$

SFR Definition 20.4. SPI0DAT: SPI0 Data

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

SFR Address: 0x9B
SFR Page: 0

Bits 7–0: SPI0DAT: SPI0 Transmit and Receive Data.
The SPI0DAT register is used to transmit and receive SPI0 data. Writing data to SPI0DAT places the data into the transmit buffer and initiates a transfer when in Master Mode. A read of SPI0DAT returns the contents of the receive buffer.

SFR Definition 21.1. SCON0: UART0 Control

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
SM00	SM10	SM20	REN0	TB80	RB80	TI0	RI0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Bit Addressable

SFR Address: 0x98
SFR Page: 0

Bits7–6: SM00–SM10: Serial Port Operation Mode:
Write:
When written, these bits select the Serial Port Operation Mode as follows:

SM00	SM10	Mode
0	0	Mode 0: Synchronous Mode
0	1	Mode 1: 8-Bit UART, Variable Baud Rate
1	0	Mode 2: 9-Bit UART, Fixed Baud Rate
1	1	Mode 3: 9-Bit UART, Variable Baud Rate

Reading these bits returns the current UART0 mode as defined above.

Bit5: SM20: Multiprocessor Communication Enable.
The function of this bit is dependent on the Serial Port Operation Mode.
Mode 0: No effect
Mode 1: Checks for valid stop bit.
0: Logic level of stop bit is ignored.
1: RI0 will only be activated if stop bit is logic level 1.
Mode 2 and 3: Multiprocessor Communications Enable.
0: Logic level of ninth bit is ignored.
1: RI0 is set and an interrupt is generated only when the ninth bit is logic 1 and the received address matches the UART0 address or the broadcast address.

Bit4: REN0: Receive Enable.
This bit enables/disables the UART0 receiver.
0: UART0 reception disabled.
1: UART0 reception enabled.

Bit3: TB80: Ninth Transmission Bit.
The logic level of this bit will be assigned to the ninth transmission bit in Modes 2 and 3. It is not used in Modes 0 and 1. Set or cleared by software as required.

Bit2: RB80: Ninth Receive Bit.
The bit is assigned the logic level of the ninth bit received in Modes 2 and 3. In Mode 1, if SM20 is logic 0, RB80 is assigned the logic level of the received stop bit. RB8 is not used in Mode 0.

Bit1: TI0: Transmit Interrupt Flag.
Set by hardware when a byte of data has been transmitted by UART0 (after the 8th bit in Mode 0, or at the beginning of the stop bit in other modes). When the UART0 interrupt is enabled, setting this bit causes the CPU to vector to the UART0 interrupt service routine. This bit must be cleared manually by software

Bit0: RI0: Receive Interrupt Flag.
Set by hardware when a byte of data has been received by UART0 (as selected by the SM20 bit). When the UART0 interrupt is enabled, setting this bit causes the CPU to vector to the UART0 interrupt service routine. This bit must be cleared manually by software.

SFR Definition 21.2. SSTA0: UART0 Status and Clock Selection

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
FE0	RXOV0	TXCOL0	SMOD0	S0TCLK1	S0TCLK0	S0RCLK1	S0RCLK0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

SFR Address: 0x91
SFR Page: 0

Bit7: FE0: Frame Error Flag.*
This flag indicates if an invalid (low) STOP bit is detected.
0: Frame Error has not been detected
1: Frame Error has been detected.

Bit6: RXOV0: Receive Overrun Flag.*
This flag indicates new data has been latched into the receive buffer before software has read the previous byte.
0: Receive overrun has not been detected.
1: Receive Overrun has been detected.

Bit5: TXCOL0: Transmit Collision Flag.*
This flag indicates user software has written to the SBUF0 register while a transmission is in progress.
0: Transmission Collision has not been detected.
1: Transmission Collision has been detected.

Bit4: SMOD0: UART0 Baud Rate Doubler Enable.
This bit enables/disables the divide-by-two function of the UART0 baud rate logic for configurations described in the UART0 section.
0: UART0 baud rate divide-by-two enabled.
1: UART0 baud rate divide-by-two disabled.

Bits3–2: UART0 Transmit Baud Rate Clock Selection Bits

S0TCLK1	S0TCLK0	**Serial Transmit Baud Rate Clock Source**
0	0	Timer 1 generates UART0 TX Baud Rate
0	1	Timer 2 Overflow generates UART0 TX baud rate
1	0	Timer 3 Overflow generates UART0 TX baud rate
1	1	Timer 4 Overflow generates UART0 TX baud rate

Bits1–0: UART0 Receive Baud Rate Clock Selection Bits

S0RCLK1	S0RCLK0	**Serial Receive Baud Rate Clock Source**
0	0	Timer 1 generates UART0 RX Baud Rate
0	1	Timer 2 Overflow generates UART0 RX baud rate
1	0	Timer 3 Overflow generates UART0 RX baud rate
1	1	Timer 4 Overflow generates UART0 RX baud rate

***Note:** FE0, RXOV0, and TXCOL0 are flags only, and no interrupt is generated by these conditions.

SFR Definition 23.5. TL1: Timer 1 Low Byte

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

SFR Address: 0x8B
SFR Page: 0

Bits 7–0: TL1: Timer 1 Low Byte.
The TL1 register is the low byte of the 16-bit Timer 1.

SFR Definition 23.6. TH0: Timer 0 High Byte

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

SFR Address: 0x8C
SFR Page: 0

Bits 7–0: TH0: Timer 0 High Byte.
The TH0 register is the high byte of the 16-bit Timer 0.

SFR Definition 23.7. TH1: Timer 1 High Byte

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

SFR Address: 0x8D
SFR Page: 0

Bits 7–0: TH1: Timer 1 High Byte.
The TH1 register is the high byte of the 16-bit Timer 1.

24.2.6. 16-Bit Pulse Width Modulator Mode

Each PCA0 module may also be operated in 16-Bit PWM mode. In this mode, the 16-bit capture/compare module defines the number of PCA0 clocks for the low time of the PWM signal. When the PCA0 counter matches the module contents, the output on CEXn is asserted high; when the counter overflows, CEXn is asserted low. To output a varying duty cycle, new value writes should be synchronized with PCA0 CCFn match interrupts. 16-Bit PWM Mode is enabled by setting the ECOMn, PWMn, and PWM16n bits in the PCA0CPMn register. For a varying duty cycle, CCFn should also be set to logic 1 to enable match interrupts. The duty cycle for 16-Bit PWM Mode is given by Equation 24.3.

Important Note About Capture/Compare Registers: When writing a 16-bit value to the PCA0 Capture/Compare registers, the low byte should always be written first. Writing to PCA0CPLn clears the ECOMn bit to ‘0’; writing to PCA0CPHn sets ECOMn to ‘1’.

Equation 24.3. 16-Bit PWM Duty Cycle

$$DutyCycle = \frac{(65536 - PCA0CPn)}{65536}$$

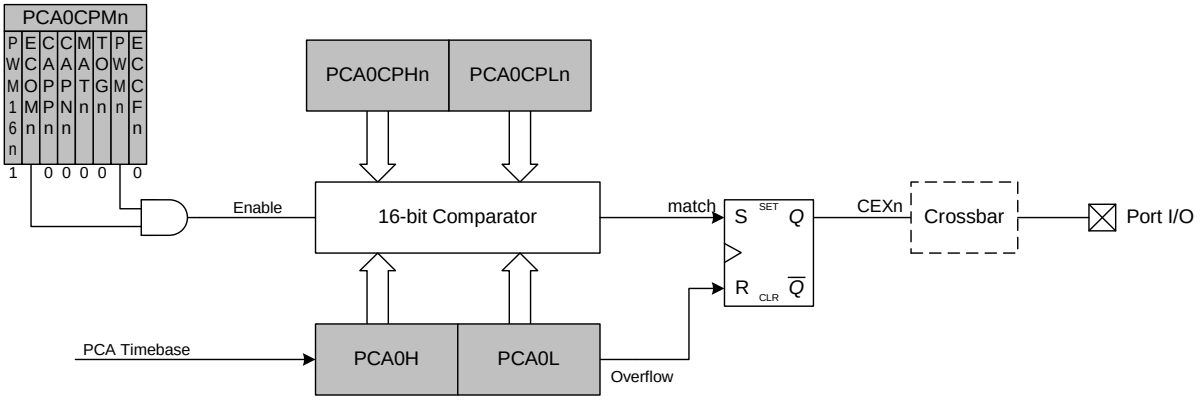


Figure 24.9. PCA 16-Bit PWM Mode

NOTES: