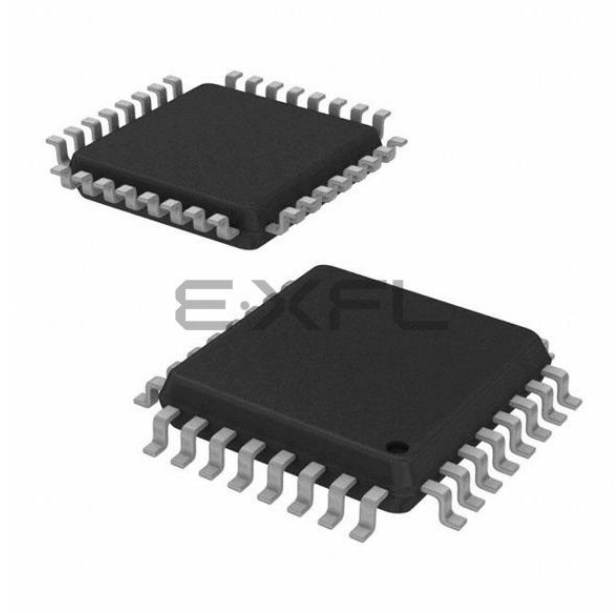


Welcome to E-XFL.COM

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), CANbus, SPI, UART/USART
Peripherals	POR, PWM, Temp Sensor, WDT
Number of I/O	25
Program Memory Size	32KB (32K 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-LQFP
Supplier Device Package	32-LQFP (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/silicon-labs/c8051f561-iq

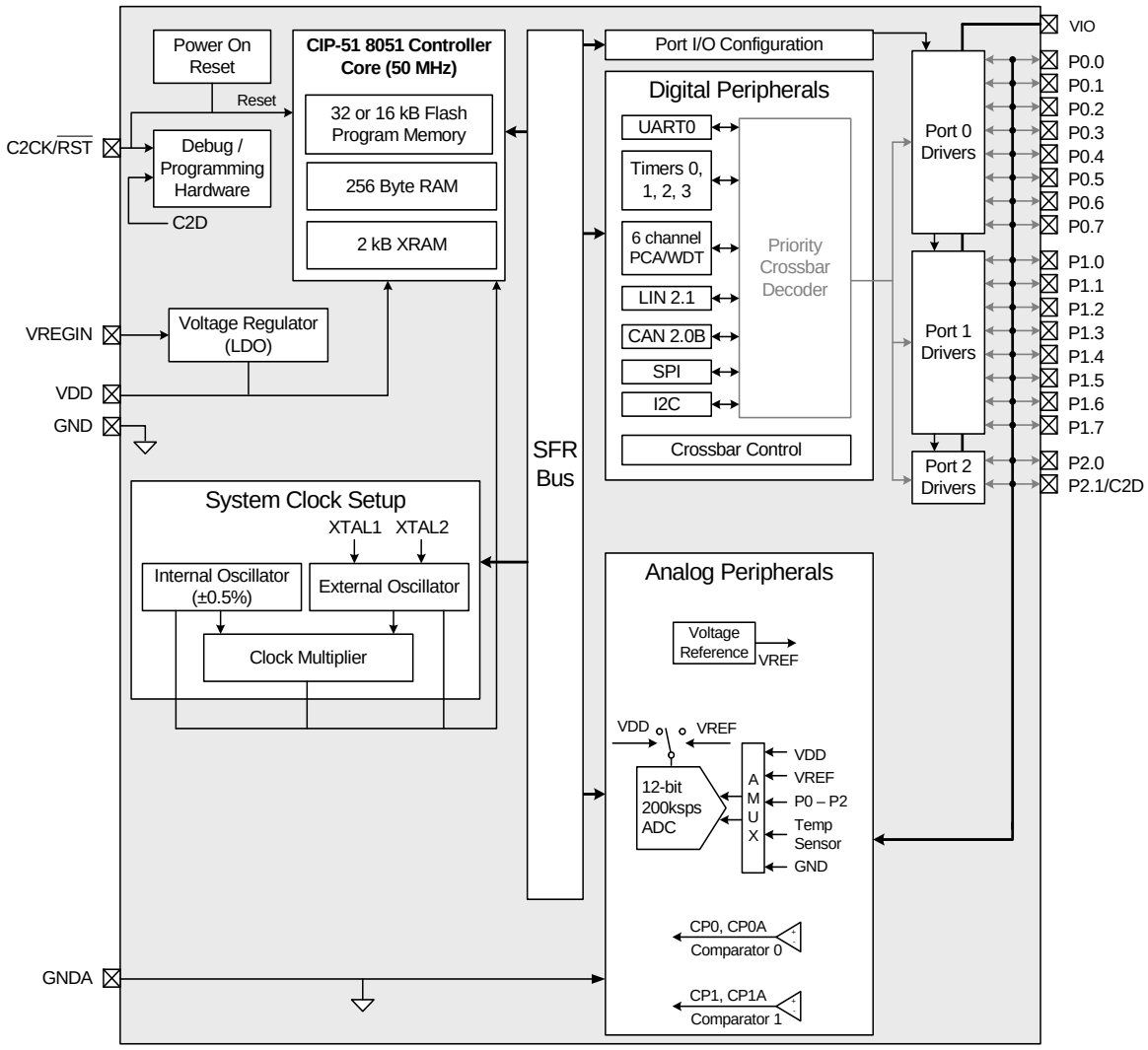


Figure 1.3. C8051F550-7 (24-pin) Block Diagram

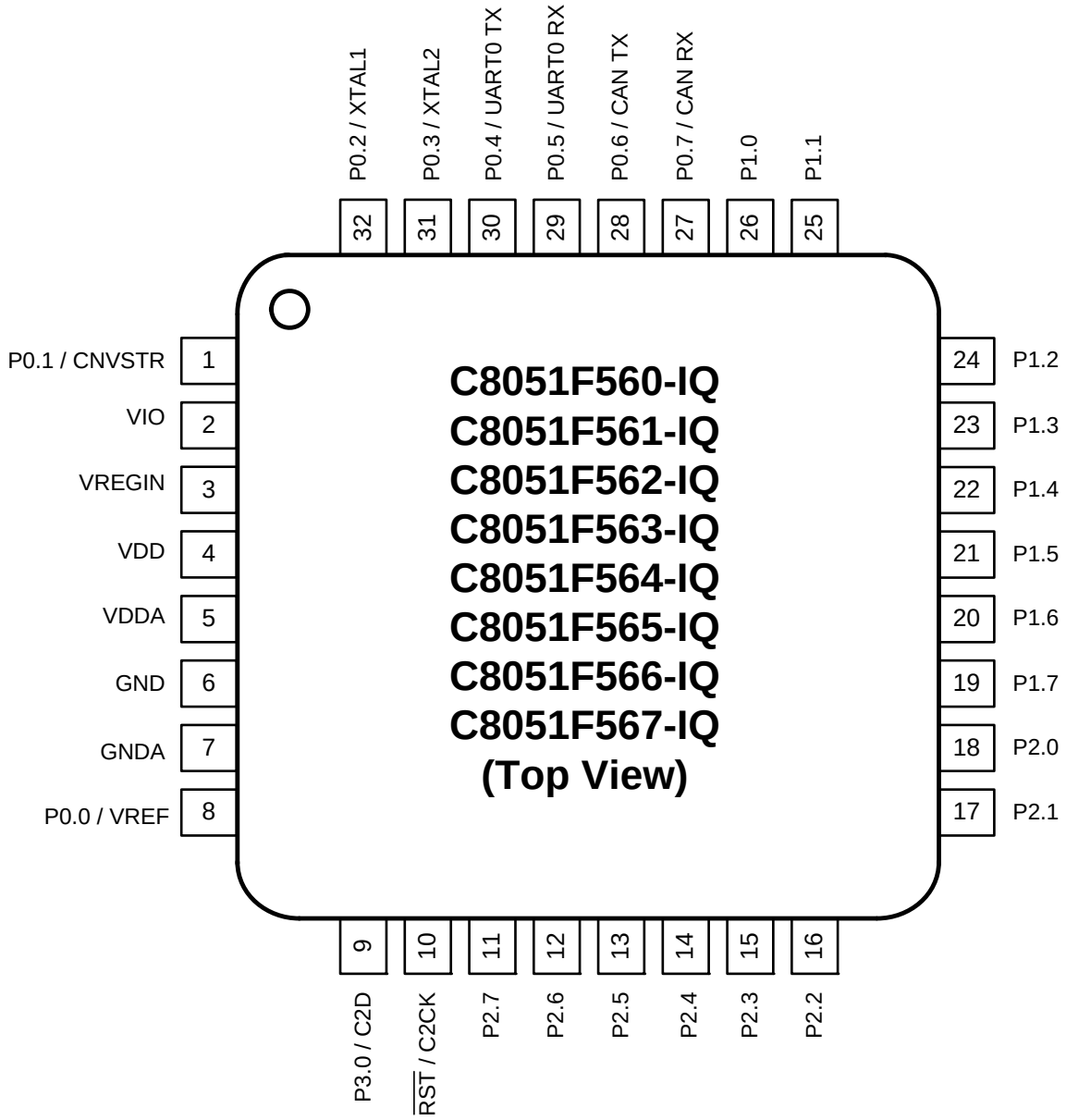


Figure 3.2. QFP-32 Pinout Diagram (Top View)

Post-Tracking Mode is selected when AD0TM is set to 01b. A programmable tracking time based on AD0TK is started immediately following the convert start signal. Conversions are started after the programmed tracking time ends. After a conversion is complete, ADC0 does not track the input. Rather, the sampling capacitor remains disconnected from the input making the input pin high-impedance until the next convert start signal.

Dual-Tracking Mode is selected when AD0TM is set to 11b. A programmable tracking time based on AD0TK is started immediately following the convert start signal. Conversions are started after the programmed tracking time ends. After a conversion is complete, ADC0 tracks continuously until the next conversion is started.

Depending on the output connected to the ADC input, additional tracking time, more than is specified in Table 5.9, may be required after changing MUX settings. See the settling time requirements described in Section “6.2.1. Settling Time Requirements” on page 52.

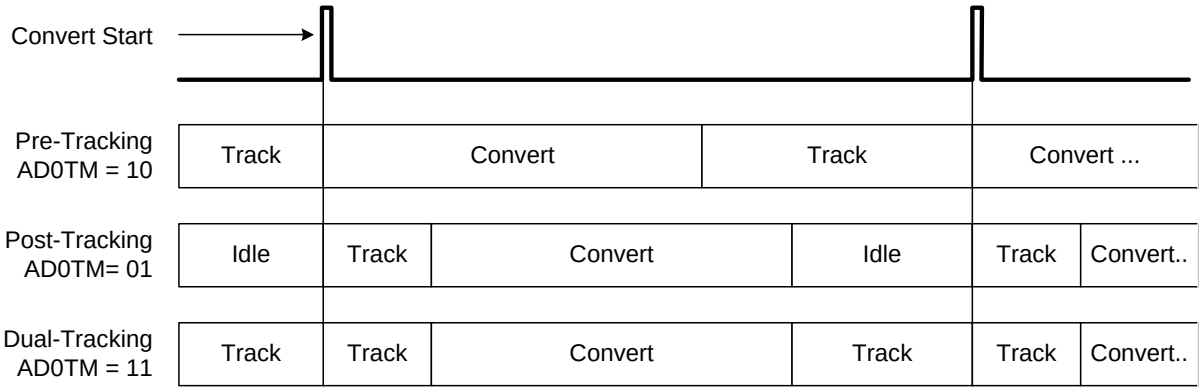


Figure 6.2. ADC0 Tracking Modes

6.1.3. Timing

ADC0 has a maximum conversion speed specified in Table 5.9. ADC0 is clocked from the ADC0 Subsystem Clock (FCLK). The source of FCLK is selected based on the BURSTEN bit. When BURSTEN is logic 0, FCLK is derived from the current system clock. When BURSTEN is logic 1, FCLK is derived from the Burst Mode Oscillator, an independent clock source with a maximum frequency of 25 MHz.

When ADC0 is performing a conversion, it requires a clock source that is typically slower than FCLK. The ADC0 SAR conversion clock (SAR clock) is a divided version of FCLK. The divide ratio can be configured using the AD0SC bits in the ADC0CF register. The maximum SAR clock frequency is listed in Table 5.9.

ADC0 can be in one of three states at any given time: tracking, converting, or idle. Tracking time depends on the tracking mode selected. For Pre-Tracking Mode, tracking is managed by software and ADC0 starts conversions immediately following the convert start signal. For Post-Tracking and Dual-Tracking Modes, the tracking time after the convert start signal is equal to the value determined by the AD0TK bits plus 2 FCLK cycles. Tracking is immediately followed by a conversion. The ADC0 conversion time is always 13 SAR clock cycles plus an additional 2 FCLK cycles to start and complete a conversion. Figure 6.3 shows timing diagrams for a conversion in Pre-Tracking Mode and tracking plus conversion in Post-Tracking or Dual-Tracking Mode. In this example, repeat count is set to one.

SFR Definition 6.8. ADC0TK: ADC0 Tracking Mode Select

Bit	7	6	5	4	3	2	1	0
Name	AD0PWR[3:0]				AD0TM[1:0]		AD0TK[1:0]	
Type	R/W				R/W		R/W	
Reset	1	1	1	1	1	1	1	1

SFR Address = 0xBA; SFR Page = 0x00

Bit	Name	Function
7:4	AD0PWR[3:0]	**ADC0 Burst Power-up Time.** For BURSTEN = 0: ADC0 Power state controlled by AD0EN For BURSTEN = 1, AD0EN = 1: ADC0 remains enabled and does not enter the very low power state For BURSTEN = 1, AD0EN = 0: ADC0 enters the very low power state and is enabled after each convert start signal. The Power-up time is programmed according the following equation: $AD0PWR = \frac{Tstartup}{200ns} - 1$ or $Tstartup = (AD0PWR + 1)200ns$
3:2	AD0TM[1:0]	**ADC0 Tracking Mode Enable Select Bits.** 00: Reserved. 01: ADC0 is configured to Post-Tracking Mode. 10: ADC0 is configured to Pre-Tracking Mode. 11: ADC0 is configured to Dual Tracking Mode.
1:0	AD0TK[1:0]	**ADC0 Post-Track Time.** 00: Post-Tracking time is equal to 2 SAR clock cycles + 2 FCLK cycles. 01: Post-Tracking time is equal to 4 SAR clock cycles + 2 FCLK cycles. 10: Post-Tracking time is equal to 8 SAR clock cycles + 2 FCLK cycles. 11: Post-Tracking time is equal to 16 SAR clock cycles + 2 FCLK cycles.

6.4. Programmable Window Detector

The ADC Programmable Window Detector continuously compares the ADC0 output registers to user-programmed limits, and notifies the system when a desired condition is detected. This is especially effective in an interrupt-driven system, saving code space and CPU bandwidth while delivering faster system response times. The window detector interrupt flag (AD0WINT in register ADC0CN) can also be used in polled mode. The ADC0 Greater-Than (ADC0GTH, ADC0GTL) and Less-Than (ADC0LTH, ADC0LTL) registers hold the comparison values. The window detector flag can be programmed to indicate when measured data is inside or outside of the user-programmed limits, depending on the contents of the ADC0 Less-Than and ADC0 Greater-Than registers.

SFR Definition 8.2. CPT0MD: Comparator0 Mode Selection

Bit	7	6	5	4	3	2	1	0
Name			CP0RIE	CP0FIE			CP0MD[1:0]	
Type	R	R	R/W	R/W	R	R	R/W	
Reset	0	0	0	0	0	0	1	0

SFR Address = 0x9B; SFR Page = 0x00

Bit	Name	Function
7:6	Unused	Read = 00b, Write = Don't Care.
5	CP0RIE	**Comparator0 Rising-Edge Interrupt Enable.** 0: Comparator0 Rising-edge interrupt disabled. 1: Comparator0 Rising-edge interrupt enabled.
4	CP0FIE	**Comparator0 Falling-Edge Interrupt Enable.** 0: Comparator0 Falling-edge interrupt disabled. 1: Comparator0 Falling-edge interrupt enabled.
3:2	Unused	Read = 00b, Write = don't care.
1:0	CP0MD[1:0]	**Comparator0 Mode Select.** These bits affect the response time and power consumption for Comparator0. 00: Mode 0 (Fastest Response Time, Highest Power Consumption) 01: Mode 1 10: Mode 2 11: Mode 3 (Slowest Response Time, Lowest Power Consumption)

8.1. Comparator Multiplexer

C8051F55x/56x/57x devices include an analog input multiplexer for each of the comparators to connect Port I/O pins to the comparator inputs. The Comparator0 inputs are selected in the CPT0MX register (SFR Definition 8.5). The CMX0P3–CMX0P0 bits select the Comparator0 positive input; the CMX0N3–CMX0N0 bits select the Comparator0 negative input. Similarly, the Comparator1 inputs are selected in the CPT1MX register using the CMX1P3-CMX1P0 bits and CMX1N3–CMX1N0 bits. The same pins are available to both multiplexers at the same time and can be used by both comparators simultaneously.

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 “19.6. Special Function Registers for Accessing and Configuring Port I/O” on page 183).

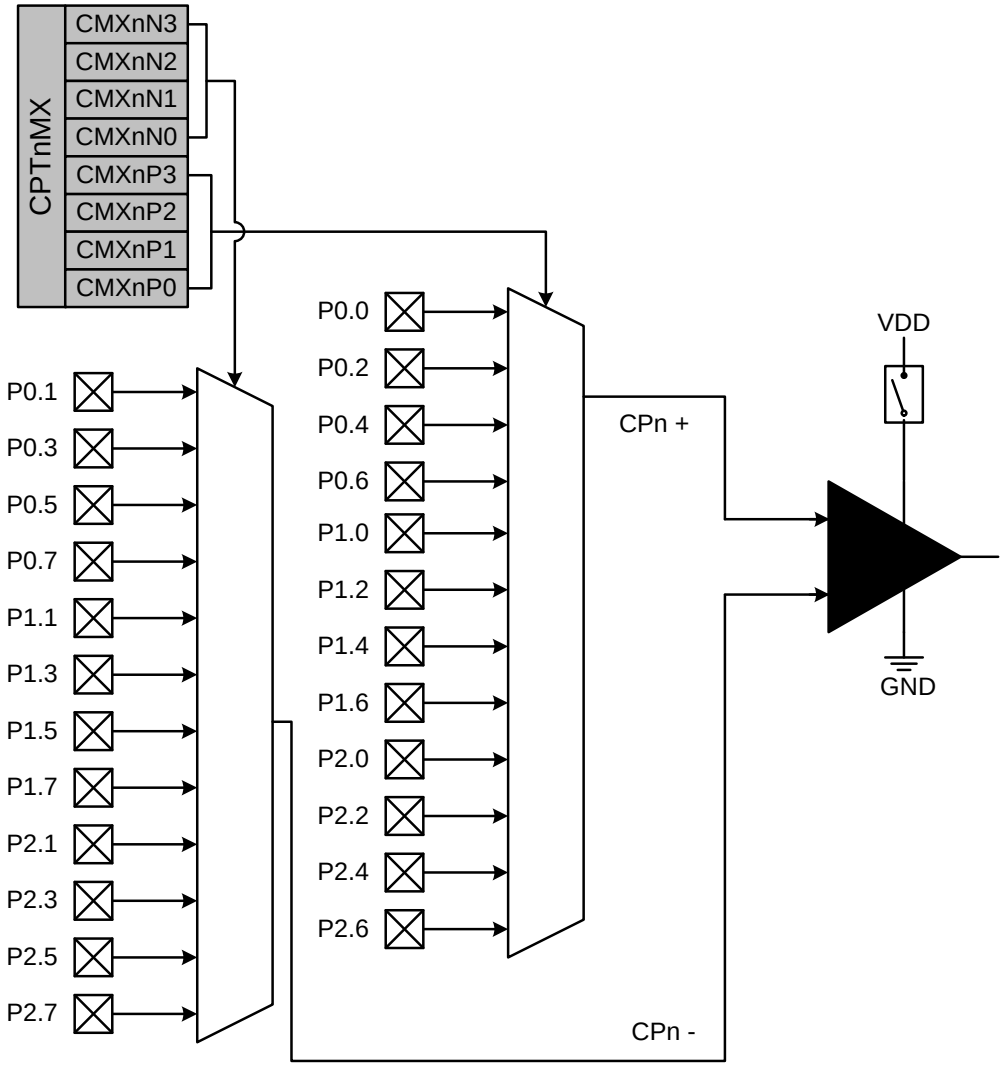


Figure 8.3. Comparator Input Multiplexer Block Diagram

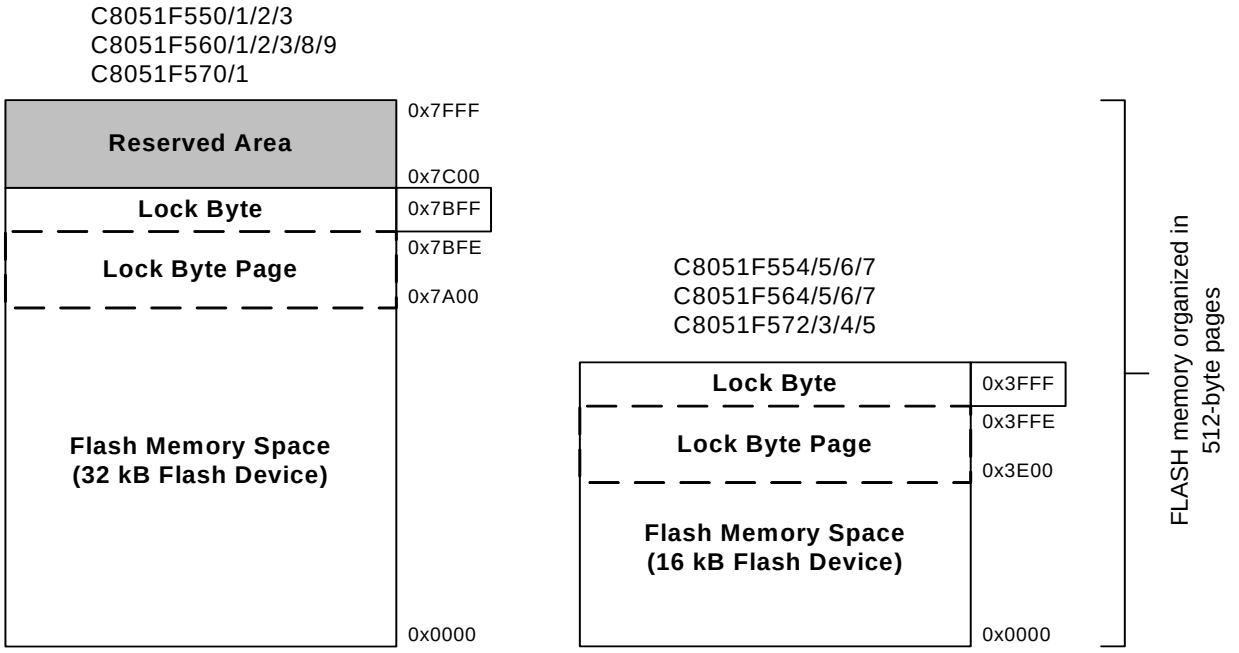


Figure 11.2. Flash Program Memory Map

11.1.1. MOVX Instruction and Program Memory

The MOVX instruction in an 8051 device is typically used to access external data memory. On the C8051F55x/56x/57x devices, the MOVX instruction is normally used to read and write on-chip XRAM, but can be re-configured to write and erase on-chip Flash memory space. MOVC instructions are always used to read Flash memory, while MOVX write instructions are used to erase and write Flash. This Flash access feature provides a mechanism for the C8051F55x/56x/57x to update program code and use the program memory space for non-volatile data storage. Refer to Section “14. Flash Memory” on page 124 for further details.

11.2. Data Memory

The C8051F55x/56x/57x devices include 2304 bytes of RAM data memory. 256 bytes of this memory is mapped into the internal RAM space of the 8051. The other 2048 bytes of this memory is on-chip “external” memory. The data memory map is shown in Figure 11.1 for reference.

11.2.1. Internal RAM

There are 256 bytes of internal RAM mapped into the data memory space from 0x00 through 0xFF. The lower 128 bytes of data memory are used for general purpose registers and scratch pad memory. Either direct or indirect addressing may be used to access the lower 128 bytes of data memory. Locations 0x00 through 0x1F are addressable as four banks of general purpose registers, each bank consisting of eight byte-wide registers. The next 16 bytes, locations 0x20 through 0x2F, may either be addressed as bytes or as 128 bit locations accessible with the direct addressing mode.

The upper 128 bytes of data memory are accessible only by indirect addressing. This region occupies the same address space as the Special Function Registers (SFR) but is physically separate from the SFR space. The addressing mode used by an instruction when accessing locations above 0x7F determines whether the CPU accesses the upper 128 bytes of data memory space or the SFRs. Instructions that use direct addressing will access the SFR space. Instructions using indirect addressing above 0x7F access the upper 128 bytes of data memory. Figure 11.1 illustrates the data memory organization of the

Table 12.1. Special Function Register (SFR) Memory Map for Pages 0x00 and 0x0F

Address	Page	0(8)	1(9)	2(A)	3(B)	4(C)	5(D)	6(E)	7(F)
F8	0F	SPI0CN	PCA0L SN0	PCA0H SN1	PCA0CPL0 SN2	PCA0CPH0 SN3	PCACPL4	PCACPH4	VDM0CN
F0	0F	B (All Pages)	P0MAT P0MDIN	P0MASK P1MDIN	P1MAT P2MDIN	P1MASK P3MDIN		EIP1 EIP1	EIP2 EIP2
E8	0F	ADC0CN	PCA0CPL1	PCA0CPH1	PCA0CPL2	PCA0CPH2	PCA0CPL3	PCA0CPL3	RSTSRC
E0	0F	ACC (All Pages)	XBR0	XBR1	CCH0CN	IT01CF		EIE1 (All Pages)	EIE2 (All Pages)
D8	0F	PCA0CN	PCA0MD PCA0PWM	PCA0CPM0	PCA0CPM1	PCA0CPM2	PCA0CPM3	PCA0CPM4	PCA0CPM5
D0	0F	PSW (All Pages)	REF0CN	LIN0DATA	LIN0ADDR	P0SKIP	P1SKIP	P2SKIP	P3SKIP
C8	0F	TMR2CN	REG0CN LIN0CF	TMR2RLL	TMR2RLH	TMR2L	TMR2H	PCA0CPL5	PCA0CPH5
C0	0F	SMB0CN	SMB0CF	SMB0DAT	ADC0GTL	ADC0GTH	ADC0LTL	ADC0LTH	XBR2
B8	0F	IP (All Pages)		ADC0TK	ADC0MX	ADC0CF	ADC0L	ADC0H	
B0	0F	P3 (All Pages)	P2MAT	P2MASK EMI0CF			P4 (All Pages)	FLSCL (All Pages)	FLKEY (All Pages)
A8	0F	IE (All Pages)	SMOD0	EMI0CN EMI0TC	SBCON0	SBRLL0	SBRLH0	P3MAT P3MDOUT	P3MASK P4MDOUT
A0	0F	P2 (All Pages)	SPI0CFG OSCICN	SPI0CKR OSCICRS	SPI0DAT	P0MDOUT	P1MDOUT	P2MDOUT	SFRPAGE (All Pages)
98	0F	SCON0	SBUF0	CPT0CN	CPT0MD	CPT0MX	CPT1CN	CPT1MD OSCIFIN	CPT1MX OSCXCN
90	0F	P1 (All Pages)	TMR3CN	TMR3RLL	TMR3RLH	TMR3L	TMR3H		CLKMUL
88	0F	TCON (All Pages)	TMOD (All Pages)	TL0 (All Pages)	TL1 (All Pages)	TH0 (All Pages)	TH1 (All Pages)	CKCON (All Pages)	PSCTL CLKSEL
80	0F	P0 (All Pages)	SP (All Pages)	DPL (All Pages)	DPH (All Pages)	SFR0CN	SFRNEXT (All Pages)	SFRLAST (All Pages)	PCON (All Pages)
		0(8)	1(9)	2(A)	3(B)	4(C)	5(D)	6(E)	7(F)

(bit addressable)

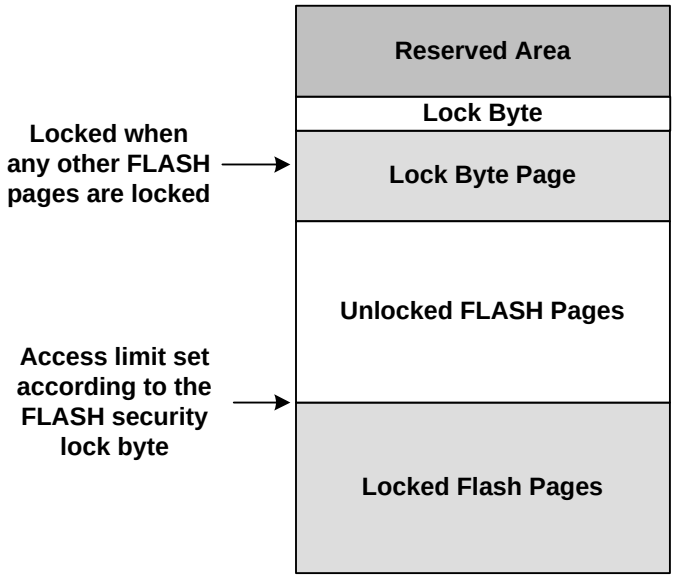
14.2. Non-volatile Data Storage

The Flash memory can be used for non-volatile data storage as well as program code. This allows data such as calibration coefficients to be calculated and stored at run time. Data is written using the MOVX write instruction and read using the MOVC instruction. Note: MOVX read instructions always target XRAM.

14.3. Security Options

The CIP-51 provides security options to protect the Flash memory from inadvertent modification by software as well as to prevent the viewing of proprietary program code and constants. The Program Store Write Enable (bit PSWE in register PSCTL) and the Program Store Erase Enable (bit PSEE in register PSCTL) bits protect the Flash memory from accidental modification by software. PSWE must be explicitly set to 1 before software can modify the Flash memory; both PSWE and PSEE must be set to 1 before software can erase Flash memory. Additional security features prevent proprietary program code and data constants from being read or altered across the C2 interface.

A Security Lock Byte located at the last byte of Flash user space offers protection of the Flash program memory from access (reads, writes, or erases) by unprotected code or the C2 interface. The Flash security mechanism allows the user to lock *n* 512-byte Flash pages, starting at page 0 (addresses 0x0000 to 0x01FF), where *n* is the ones complement number represented by the Security Lock Byte. **Note that the page containing the Flash Security Lock Byte is unlocked when no other Flash pages are locked (all bits of the Lock Byte are 1) and locked when any other Flash pages are locked (any bit of the Lock Byte is 0).** See example in Figure 14.1.



Security Lock Byte:	11111101b
1s Complement:	00000010b
Flash pages locked:	3 (First two Flash pages + Lock Byte Page)

Figure 14.1. Flash Program Memory Map

SFR Definition 17.3. EMI0TC: External Memory Timing Control

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

SFR Address = 0xAA; SFR Page = 0x0F

Bit	Name	Function
7:6	EAS[1:0]	**EMIF Address Setup Time Bits.** 00: Address setup time = 0 SYSCLK cycles. 01: Address setup time = 1 SYSCLK cycle. 10: Address setup time = 2 SYSCLK cycles. 11: Address setup time = 3 SYSCLK cycles.
5:2	EWR[3:0]	**EMIF $\overline{WR}$ and $\overline{RD}$ Pulse-Width Control Bits.** 0000: $\overline{WR}$ and $\overline{RD}$ pulse width = 1 SYSCLK cycle. 0001: $\overline{WR}$ and $\overline{RD}$ pulse width = 2 SYSCLK cycles. 0010: $\overline{WR}$ and $\overline{RD}$ pulse width = 3 SYSCLK cycles. 0011: $\overline{WR}$ and $\overline{RD}$ pulse width = 4 SYSCLK cycles. 0100: $\overline{WR}$ and $\overline{RD}$ pulse width = 5 SYSCLK cycles. 0101: $\overline{WR}$ and $\overline{RD}$ pulse width = 6 SYSCLK cycles. 0110: $\overline{WR}$ and $\overline{RD}$ pulse width = 7 SYSCLK cycles. 0111: $\overline{WR}$ and $\overline{RD}$ pulse width = 8 SYSCLK cycles. 1000: $\overline{WR}$ and $\overline{RD}$ pulse width = 9 SYSCLK cycles. 1001: $\overline{WR}$ and $\overline{RD}$ pulse width = 10 SYSCLK cycles. 1010: $\overline{WR}$ and $\overline{RD}$ pulse width = 11 SYSCLK cycles. 1011: $\overline{WR}$ and $\overline{RD}$ pulse width = 12 SYSCLK cycles. 1100: $\overline{WR}$ and $\overline{RD}$ pulse width = 13 SYSCLK cycles. 1101: $\overline{WR}$ and $\overline{RD}$ pulse width = 14 SYSCLK cycles. 1110: $\overline{WR}$ and $\overline{RD}$ pulse width = 15 SYSCLK cycles. 1111: $\overline{WR}$ and $\overline{RD}$ pulse width = 16 SYSCLK cycles.
1:0	EAH[1:0]	**EMIF Address Hold Time Bits.** 00: Address hold time = 0 SYSCLK cycles. 01: Address hold time = 1 SYSCLK cycle. 10: Address hold time = 2 SYSCLK cycles. 11: Address hold time = 3 SYSCLK cycles.

18.3. Clock Multiplier

The Clock Multiplier generates an output clock which is 4 times the input clock frequency scaled by a programmable factor of 1, 2/3, 2/4 (or 1/2), 2/5, 2/6 (or 1/3), or 2/7. The Clock Multiplier's input can be selected from the external oscillator, or the internal or external oscillators divided by 2. This produces three possible base outputs which can be scaled by a programmable factor: Internal Oscillator x 2, External Oscillator x 2, or External Oscillator x 4. See Section 18.1 on page 157 for details on system clock selection.

The Clock Multiplier is configured via the CLKMUL register (SFR Definition 18.5). The procedure for configuring and enabling the Clock Multiplier is as follows:

1. Reset the Multiplier by writing 0x00 to register CLKMUL.
2. Select the Multiplier input source via the MULSEL bits.
3. Select the Multiplier output scaling factor via the MULDIV bits
4. Enable the Multiplier with the MULEN bit (CLKMUL | = 0x80).
5. Delay for >5 µs.
6. Initialize the Multiplier with the MULINIT bit (CLKMUL | = 0xC0).
7. Poll for MULRDY $\geq$ 1.

Important Note: When using an external oscillator as the input to the Clock Multiplier, the external source must be enabled and stable before the Multiplier is initialized. See "18.4. External Oscillator Drive Circuit" on page 164 for details on selecting an external oscillator source.

The Clock Multiplier allows faster operation of the CIP-51 core and is intended to generate an output frequency between 25 and 50 MHz. The clock multiplier can also be used with slow input clocks. However, if the clock is below the minimum Clock Multiplier input frequency (FCM_{min}), the generated clock will consist of four fast pulses followed by a long delay until the next input clock rising edge. The average frequency of the output is equal to 4x the input, but the instantaneous frequency may be faster. See Figure 18.2 below for more information.

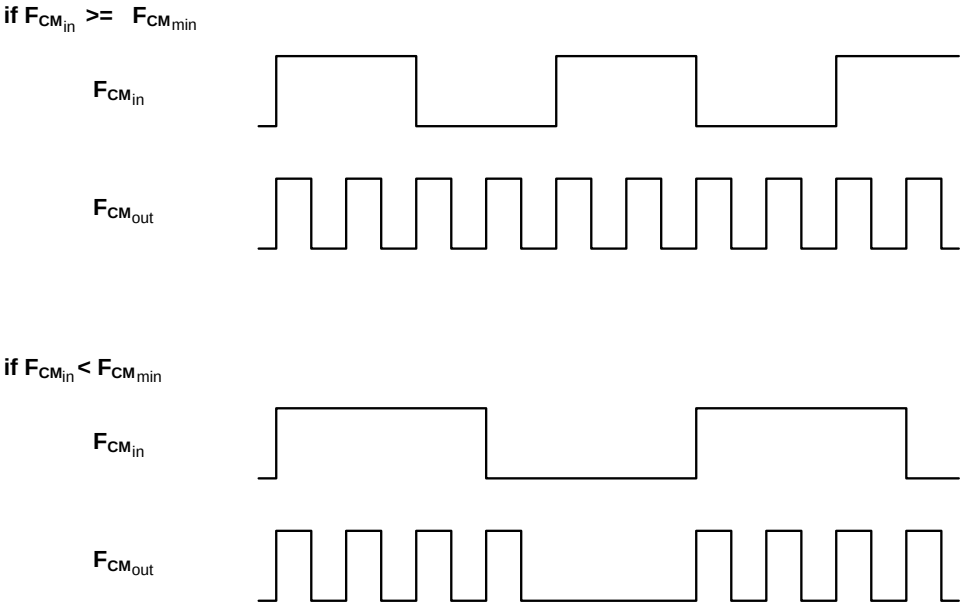


Figure 18.2. Example Clock Multiplier Output

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.

21.2. CAN Registers

CAN registers are classified as follows:

1. **CAN Controller Protocol Registers**: CAN control, interrupt, error control, bus status, test modes.
2. **Message Object Interface Registers**: Used to configure 32 Message Objects, send and receive data to and from Message Objects. The CIP-51 MCU accesses the CAN message RAM via the Message Object Interface Registers. Upon writing a message object number to an IF1 or IF2 Command Request Register, the contents of the associated Interface Registers (IF1 or IF2) will be transferred to or from the message object in CAN RAM.
3. **Message Handler Registers**: These read only registers are used to provide information to the CIP-51 MCU about the message objects (MSGVLD flags, Transmission Request Pending, New Data Flags) and Interrupts Pending (which Message Objects have caused an interrupt or status interrupt condition).

For the registers other than CAN0CFG, refer to the Bosch CAN User's Guide for information on the function and use of the CAN Control Protocol Registers.

21.2.1. CAN Controller Protocol Registers

The CAN Control Protocol Registers are used to configure the CAN controller, process interrupts, monitor bus status, and place the controller in test modes.

The registers are: CAN Control Register (CAN0CN), CAN Clock Configuration (CAN0CFG), CAN Status Register (CAN0STA), CAN Test Register (CAN0TST), Error Counter Register, Bit Timing Register, and the Baud Rate Prescaler (BRP) Extension Register.

21.2.2. Message Object Interface Registers

There are two sets of Message Object Interface Registers used to configure the 32 Message Objects that transmit and receive data to and from the CAN bus. Message objects can be configured for transmit or receive, and are assigned arbitration message identifiers for acceptance filtering by all CAN nodes.

Message Objects are stored in Message RAM, and are accessed and configured using the Message Object Interface Registers.

21.2.3. Message Handler Registers

The Message Handler Registers are read only registers. The message handler registers provide interrupt, error, transmit/receive requests, and new data information.

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 Address = 0x98; Bit-Addressable; SFR Page = 0x00

Bit	Name	Function
7	OVR0	**Receive FIFO Overrun Flag.** 0: Receive FIFO Overrun has not occurred 1: Receive FIFO Overrun has occurred; A received character has been discarded due to a full FIFO.
6	PERR0	**Parity Error Flag.** When parity is enabled, this bit indicates that a parity error has occurred. It is set to 1 when the parity of the oldest byte in the FIFO does not match the selected Parity Type. 0: Parity error has not occurred 1: Parity error has occurred. This bit must be cleared by software.
5	THRE0	**Transmit Holding Register Empty Flag.** THRE0 can have a momentary glitch high when the UART Transmit Holding Register is not empty. The glitch will occur some time after SBUF0 was written with the previous byte and does not occur if THRE0 is checked in the instruction(s) immediately following the write to SBUF0. When firmware writes SBUF0 and SBUF0 is not empty, TX0 will be stuck low until the next device reset. Firmware should use or poll on TI0 rather than THRE0 for asynchronous UART writes that may have a random delay in between transactions. 0: Transmit Holding Register not Empty—do not write to SBUF0. 1: Transmit Holding Register Empty—it is safe to write to SBUF0.
4	REN0	**Receive Enable.** This bit enables/disables the UART receiver. When disabled, bytes can still be read from the receive FIFO. 0: UART1 reception disabled. 1: UART1 reception enabled.
3	TBX0	**Extra Transmission Bit.** The logic level of this bit will be assigned to the extra transmission bit when XBE0 is set to 1. This bit is not used when Parity is enabled.
2	RBX0	**Extra Receive Bit.** RBX0 is assigned the value of the extra bit when XBE1 is set to 1. If XBE1 is cleared to 0, RBX1 will be assigned the logic level of the first stop bit. This bit is not valid when Parity is enabled.
1	TI0	**Transmit Interrupt Flag.** Set to a 1 by hardware after data has been transmitted, at the beginning of the STOP 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.
0	RI0	**Receive Interrupt Flag.** Set to 1 by hardware when a byte of data has been received by UART0 (set at the STOP bit sampling time). When the UART0 interrupt is enabled, setting this bit to 1 causes the CPU to vector to the UART0 interrupt service routine. This bit must be cleared manually by software. Note that RI0 will remain set to ‘1’ as long as there is data still in the UART FIFO. After the last byte has been shifted from the FIFO to SBUF0, RI0 can be cleared.

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. Enhanced Serial Peripheral Interface (SPI0)

The Enhanced Serial Peripheral Interface (SPI0) provides access to a flexible, full-duplex synchronous serial bus. SPI0 can operate as a master or slave device in both 3-wire or 4-wire modes, and supports multiple masters and slaves on a single SPI bus. The slave-select (NSS) signal can be configured as an input to select SPI0 in slave mode, or to disable Master Mode operation in a multi-master environment, avoiding contention on the SPI bus when more than one master attempts simultaneous data transfers. NSS can also be configured as a chip-select output in master mode, or disabled for 3-wire operation. Additional general purpose port I/O pins can be used to select multiple slave devices in master mode.

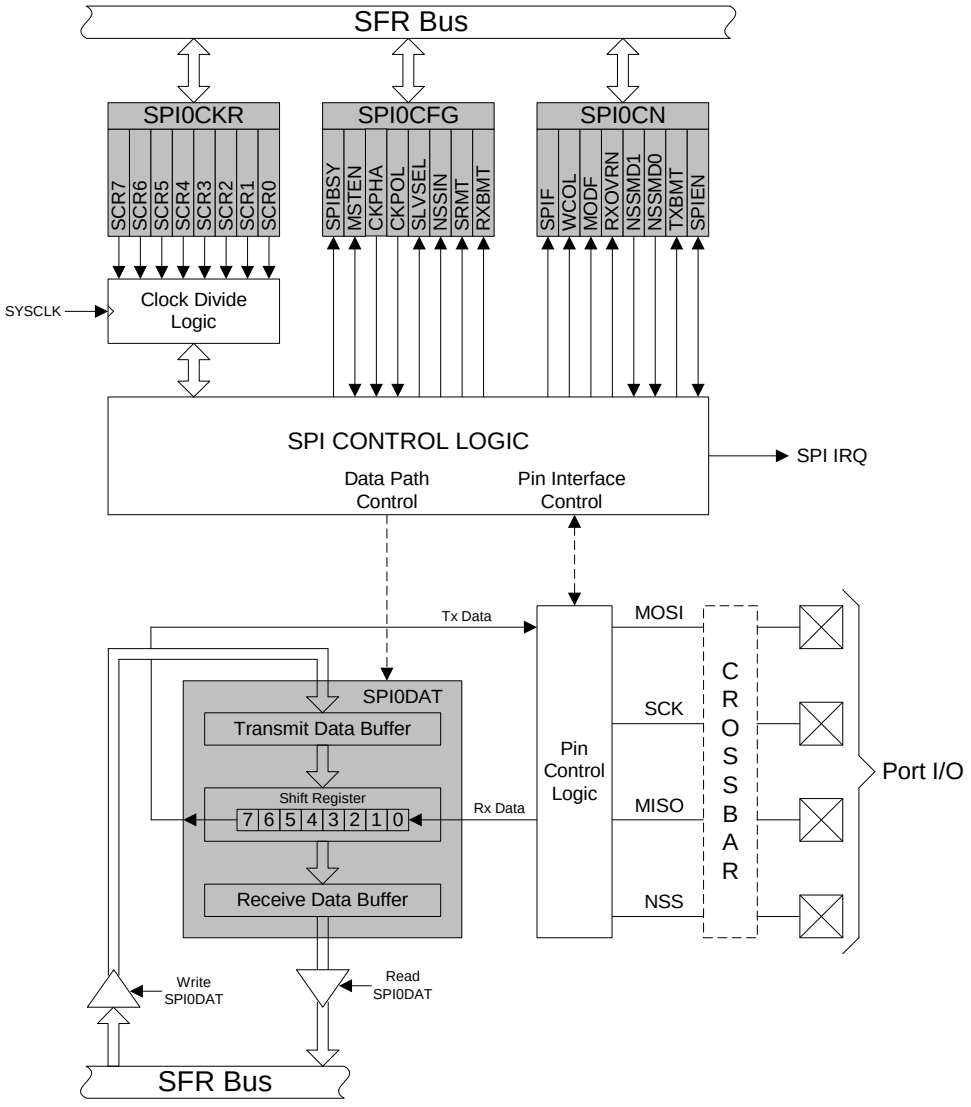


Figure 24.1. SPI Block Diagram

SFR Definition 24.3. SPI0CKR: SPI0 Clock Rate

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

SFR Address = 0xA2; SFR Page = 0x00

Bit	Name	Function
7:0	SCR[7:0]	**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. $f_{SCK} = \frac{SYSCLK}{2 \times (SPI0CKR[7:0] + 1)}$ for 0 <= SPI0CKR <= 255 Example: If SYSCLK = 2 MHz and SPI0CKR = 0x04, $f_{SCK} = \frac{2000000}{2 \times (4 + 1)}$ $f_{SCK} = 200\ kHz$

SFR Definition 24.4. SPI0DAT: SPI0 Data

Bit	7	6	5	4	3	2	1	0
Name	SPI0DAT[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xA3; SFR Page = 0x00

Bit	Name	Function
7:0	SPI0DAT[7:0]	**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.

C2 Register Definition 27.2. DEVICEID: C2 Device ID

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

C2 Address = 0xFD; SFR Address = 0xFD; SFR Page = 0xF

Bit	Name	Function
7:0	DEVICEID[7:0]	**Device ID.** This read-only register returns the 8-bit device ID: 0x22 (C8051F55x/56x/57x).

C2 Register Definition 27.3. REVID: C2 Revision ID

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

C2 Address = 0xFE; SFR Address = 0xFE; SFR Page = 0xF

Bit	Name	Function
7:0	REVID[7:0]	**Revision ID.** This read-only register returns the 8-bit revision ID. For example: 0x00 = Revision A.