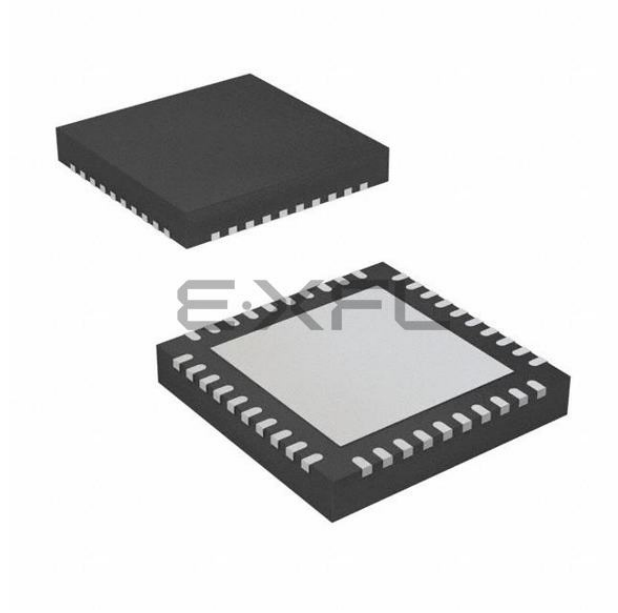

Welcome to E-XFL.COM

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	CANbus, EBI/EMI, SMBus (2-Wire/I²C), SPI, UART/USART
Peripherals	POR, PWM, Temp Sensor, WDT
Number of I/O	33
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 32x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	40-VFQFN Exposed Pad
Supplier Device Package	40-QFN (6x6)
Purchase URL	https://www.e-xfl.com/product-detail/silicon-labs/c8051f573-im

List of Registers

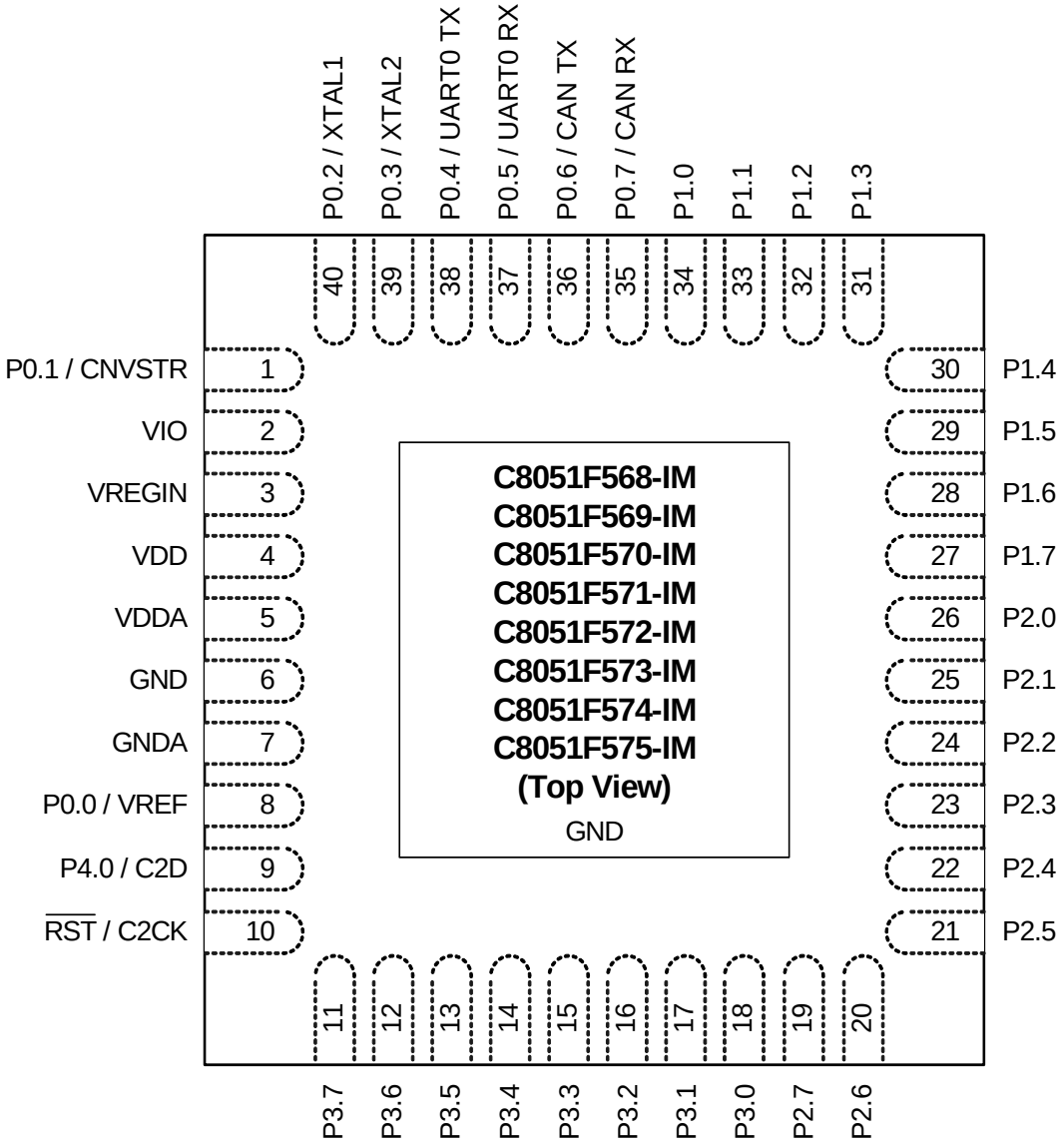

Figure 3.1. QFN-40 Pinout Diagram (Top View)

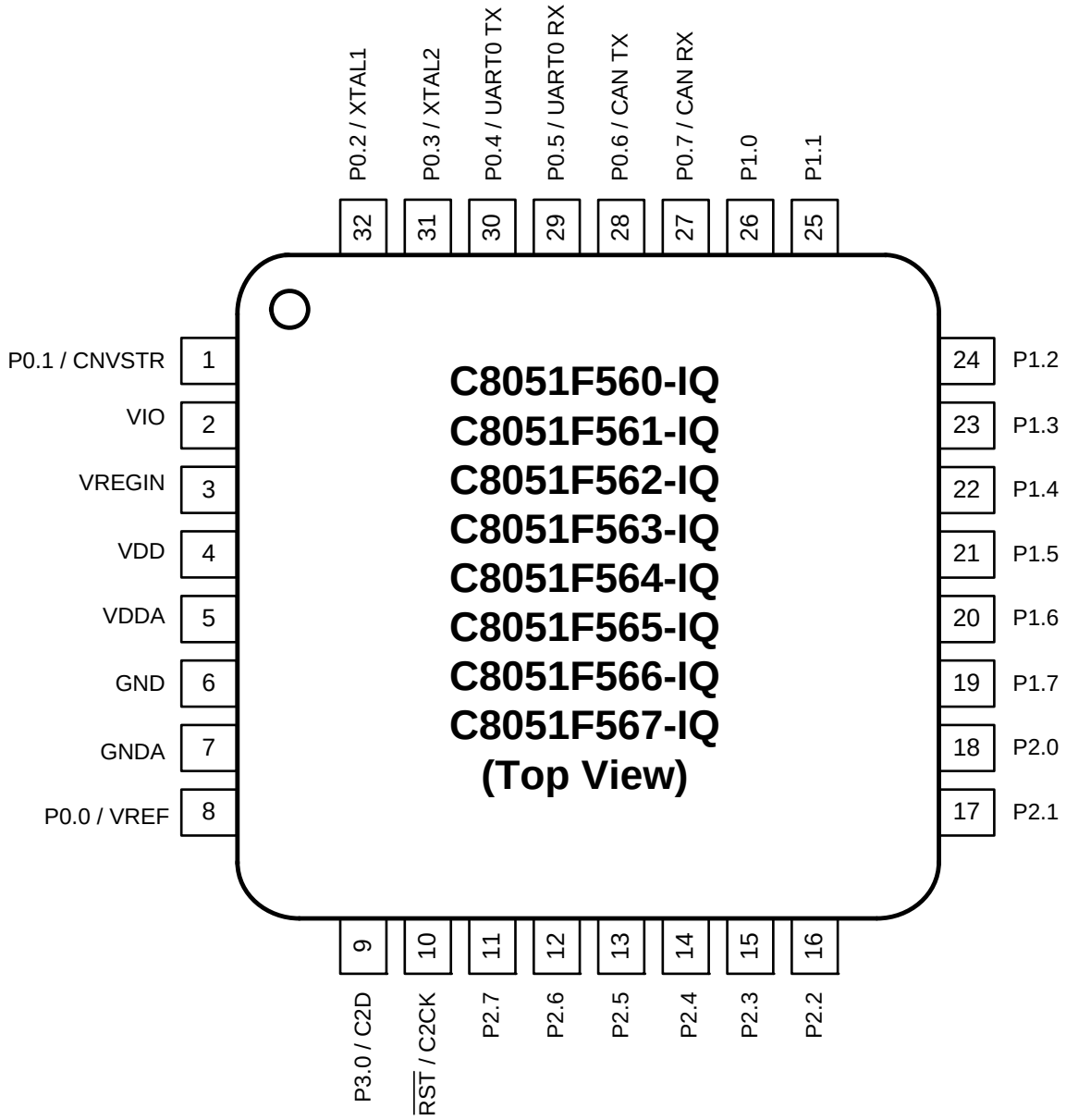

Figure 3.2. QFP-32 Pinout Diagram (Top View)

Table 5.2. Global Electrical Characteristics (Continued)
–40 to +125 °C, 24 MHz system clock unless otherwise specified.

Parameter	Conditions	Min	Typ	Max	Units
I_{DD} Frequency Sensitivity [4,5]	V_{DD} = 2.1 V, F ≤ 12.5 MHz, T = 25 °C	—	0.43	—	mA/MHz
	V_{DD} = 2.1 V, F > 12.5 MHz, T = 25 °C	—	0.33	—	mA/MHz
	V_{DD} = 2.6 V, F ≤ 12.5 MHz, T = 25 °C	—	0.60	—	mA/MHz
	V_{DD} = 2.6 V, F > 12.5 MHz, T = 25 °C	—	0.42	—	mA/MHz
Digital Supply Current—CPU Inactive (Idle Mode, not fetching instructions from Flash)					
I_{DD} [4]	V_{DD} = 2.1 V, F = 200 kHz	—	50	—	µA
	V_{DD} = 2.1 V, F = 1.5 MHz	—	410	—	µA
	V_{DD} = 2.1 V, F = 25 MHz	—	6.5	8.0	mA
	V_{DD} = 2.1 V, F = 50 MHz	—	13	16	mA
I_{DD} [4]	V_{DD} = 2.6 V, F = 200 kHz	—	67	—	µA
	V_{DD} = 2.6 V, F = 1.5 MHz	—	530	—	µA
	V_{DD} = 2.6 V, F = 25 MHz	—	8.0	15	mA
	V_{DD} = 2.6 V, F = 50 MHz	—	16	25	mA
I_{DD} Supply Sensitivity [4]	F = 25 MHz	—	55	—	%/V
	F = 1 MHz	—	58	—	
I_{DD} Frequency Sensitivity [4,6]	V_{DD} = 2.1V, F ≤ 12.5 MHz, T = 25 °C	—	0.26	—	mA/MHz
	V_{DD} = 2.1V, F > 12.5 MHz, T = 25 °C	—	0.26	—	
	V_{DD} = 2.6V, F ≤ 12.5 MHz, T = 25 °C	—	0.34	—	
	V_{DD} = 2.6V, F > 12.5 MHz, T = 25 °C	—	0.34	—	
Digital Supply Current [4] (Stop or Suspend Mode)	Oscillator not running, V_{DD} Monitor Disabled				
	Temp = 25 °C	—	1	—	µA
	Temp = 60 °C	—	6	—	
	Temp= 125 °C	—	70	—	

Notes:
1. Given in Table 5.4 on page 41.
2. V_{IO} should not be lower than the V_{DD} voltage.
3. SYSCLK must be at least 32 kHz to enable debugging.
4. Guaranteed by characterization. Does not include oscillator supply current.
5. IDD estimation for different frequencies.
6. Idle IDD estimation for different frequencies.

6.5. ADC0 Analog Multiplexer

ADC0 includes an analog multiplexer to enable multiple analog input sources. Any of the following may be selected as an input: P0.0–P3.7, the on-chip temperature sensor, the core power supply (V_{DD}), or ground (GND). **ADC0 is single-ended and all signals measured are with respect to GND.** The ADC0 input channels are selected using the ADC0MX register as described in SFR Definition 6.13.

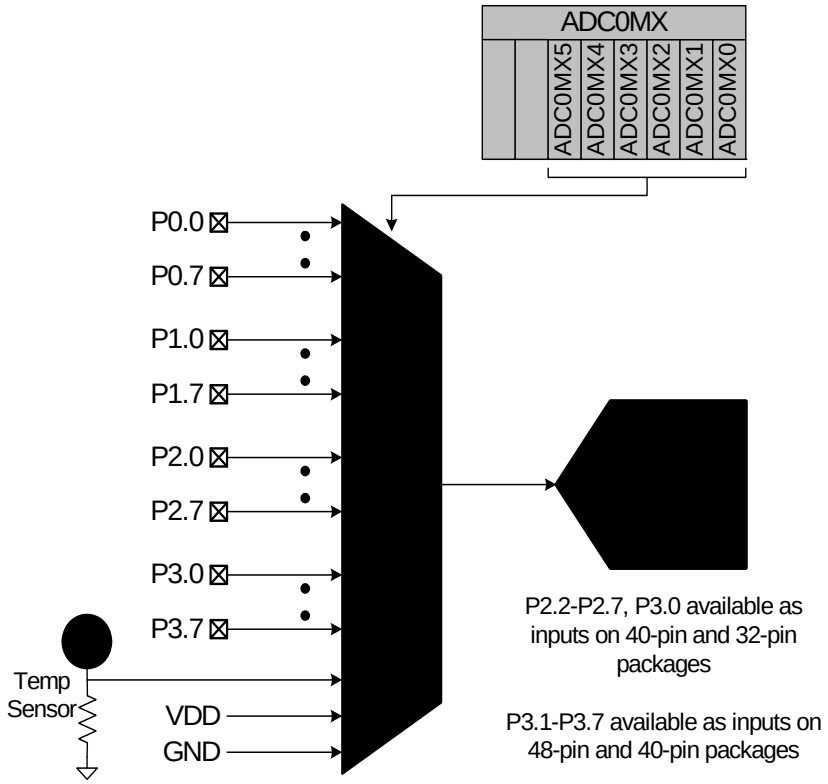

Figure 6.8. ADC0 Multiplexer Block Diagram

Important Note About ADC0 Input Configuration: Port pins selected as ADC0 inputs should be configured as analog inputs, and should be skipped by the Digital Crossbar. To configure a Port pin for analog input, set to 0 the corresponding bit in register PnMDIN. To force the Crossbar to skip a Port pin, set to 1 the corresponding bit in register PnSKIP. See Section “19. Port Input/Output” on page 169 for more Port I/O configuration details.

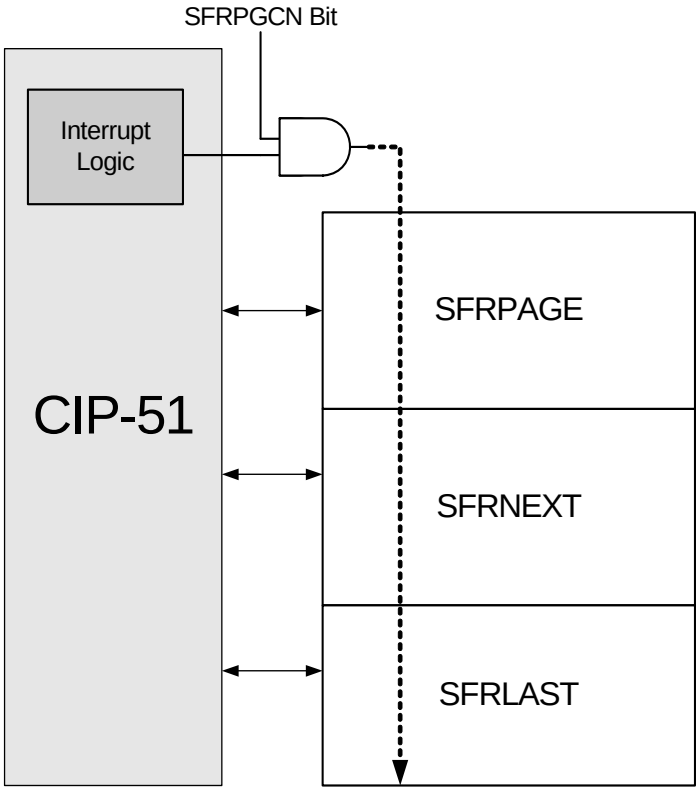

Figure 12.1. SFR Page Stack

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 (SFR0CN). 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) are provided in Table 12.3 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. Certain SFRs 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 SFRs are accessible from all SFR pages regardless of the SFRPAGE register value.

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.

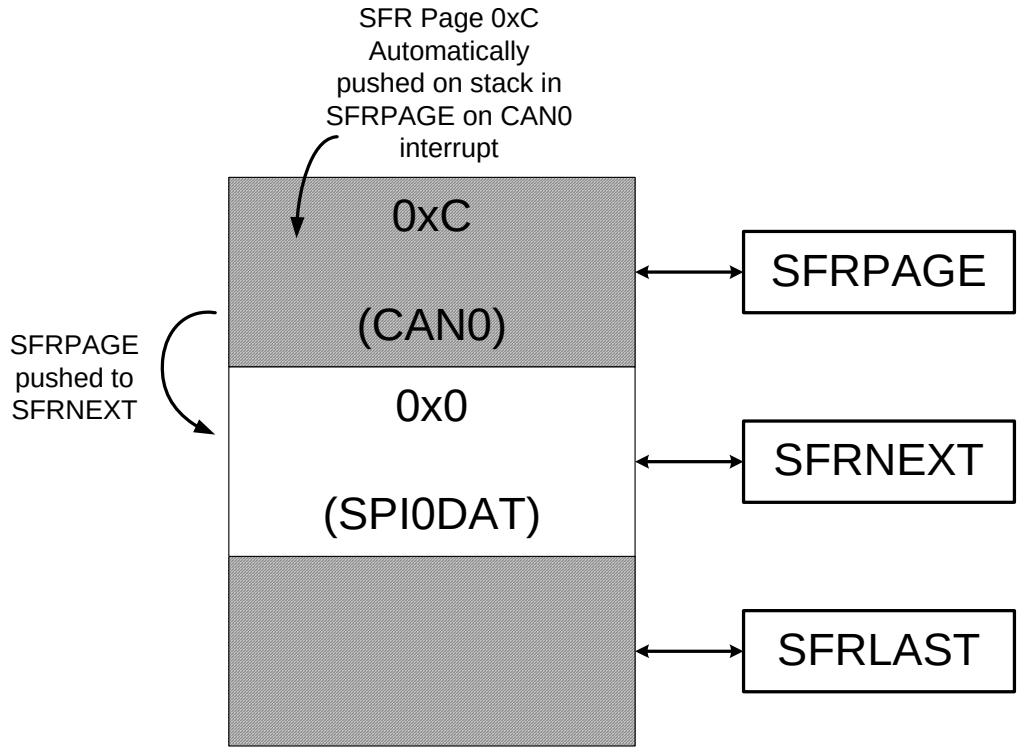

Figure 12.3. SFR Page Stack After CAN0 Interrupt Occurs

14.4.2. PSWE Maintenance

1. Reduce the number of places in code where the PSWE bit (b0 in PSCTL) is set to a 1. There should be exactly one routine in code that sets PSWE to a 1 to write Flash bytes and one routine in code that sets PSWE and PSEE both to a 1 to erase Flash pages.
2. Minimize the number of variable accesses while PSWE is set to a 1. Handle pointer address updates and loop variable maintenance outside the "PSWE = 1;... PSWE = 0;" area. Code examples showing this can be found in ”AN201: Writing to Flash from Firmware" available from the Silicon Laboratories web site.
3. Disable interrupts prior to setting PSWE to a 1 and leave them disabled until after PSWE has been reset to '0'. Any interrupts posted during the Flash write or erase operation will be serviced in priority order after the Flash operation has been completed and interrupts have been re-enabled by software.
4. Make certain that the Flash write and erase pointer variables are not located in XRAM. See your compiler documentation for instructions regarding how to explicitly locate variables in different memory areas.
5. Add address bounds checking to the routines that write or erase Flash memory to ensure that a routine called with an illegal address does not result in modification of the Flash.

14.4.3. System Clock

1. If operating from an external crystal, be advised that crystal performance is susceptible to electrical interference and is sensitive to layout and to changes in temperature. If the system is operating in an electrically noisy environment, use the internal oscillator or use an external CMOS clock.
2. If operating from the external oscillator, switch to the internal oscillator during Flash write or erase operations. The external oscillator can continue to run, and the CPU can switch back to the external oscillator after the Flash operation has completed.

Additional Flash recommendations and example code can be found in “AN201: Writing to Flash from Firmware" available from the Silicon Laboratories web site.

SFR Definition 16.2. RSTSRC: Reset Source

Bit	7	6	5	4	3	2	1	0
Name		FERROR	C0RSEF	SWRSF	WDTRSF	MCDRSF	PORSF	PINRSF
Type	R	R	R/W	R/W	R	R/W	R/W	R
Reset	0	Varies	Varies	Varies	Varies	Varies	Varies	Varies

SFR Address = 0xEF; SFR Page = 0x00

Bit	Name	Description	Write	Read
7	Unused	**Unused.**	Don't care.	0
6	FERROR	**Flash Error Reset Flag.**	N/A	Set to 1 if Flash read/write/erase error caused the last reset.
5	C0RSEF	**Comparator0 Reset Enable and Flag.**	Writing a 1 enables Comparator0 as a reset source (active-low).	Set to 1 if Comparator0 caused the last reset.
4	SWRSF	**Software Reset Force and Flag.**	Writing a 1 forces a system reset.	Set to 1 if last reset was caused by a write to SWRSF.
3	WDTRSF	**Watchdog Timer Reset Flag.**	N/A	Set to 1 if Watchdog Timer overflow caused the last reset.
2	MCDRSF	**Missing Clock Detector Enable and Flag.**	Writing a 1 enables the Missing Clock Detector. The MCD triggers a reset if a missing clock condition is detected.	Set to 1 if Missing Clock Detector timeout caused the last reset.
1	PORSF	**Power-On/V_{DD} Monitor Reset Flag, and V_{DD} monitor Reset Enable.**	Writing a 1 enables the V_{DD} monitor as a reset source. **Writing 1 to this bit before the V_{DD} monitor is enabled and stabilized may cause a system reset.**	Set to 1 anytime a power-on or V_{DD} monitor reset occurs. **When set to 1 all other RSTSRC flags are indeterminate.**
0	PINRSF	**HW Pin Reset Flag.**	N/A	Set to 1 if $\overline{\text{RST}}$ pin caused the last reset.

Note: Do not use read-modify-write operations on this register

19. Port Input/Output

Digital and analog resources are available through 33 (C8051F568-9 and ‘F570-5), 25 (C8051F550-7) or 18 (C8051F550-7) I/O pins. Port pins P0.0-P4.0 on the C8051F568-9 and ‘F570-5, port pins P0.0-P3.0 on theC8051F560-7, and port pins P0.0-P2.1 on the C8051F550-7 can be defined as general-purpose I/O (GPIO), assigned to one of the internal digital resources, or assigned to an analog function as shown in Figure 19.3. Port pin P4.0 on the C8051F568-9 and ‘F570-5 can be used as GPIO and is shared with the C2 Interface Data signal (C2D). Similarly, port pin P3.0 is shared with C2D on the C8051F560-7 and port pin P2.1 on the C8051F550-7. The designer has complete control over which functions are assigned, limited only by the number of physical I/O pins. This resource assignment flexibility is achieved through the use of a Priority Crossbar Decoder. Note that the state of a Port I/O pin can always be read in the corresponding Port latch, regardless of the Crossbar settings.

The Crossbar assigns the selected internal digital resources to the I/O pins based on the Priority Decoder (Figure 19.3 and Figure 19.4). The registers XBR0, XBR1, XBR2 are defined in SFR Definition 19.1 and SFR Definition 19.2 and are used to select internal digital functions.

The Port I/O cells are configured as either push-pull or open-drain in the Port Output Mode registers (PnMDOUT, where n = 0,1). Complete Electrical Specifications for Port I/O are given in Table 5.3 on page 40.

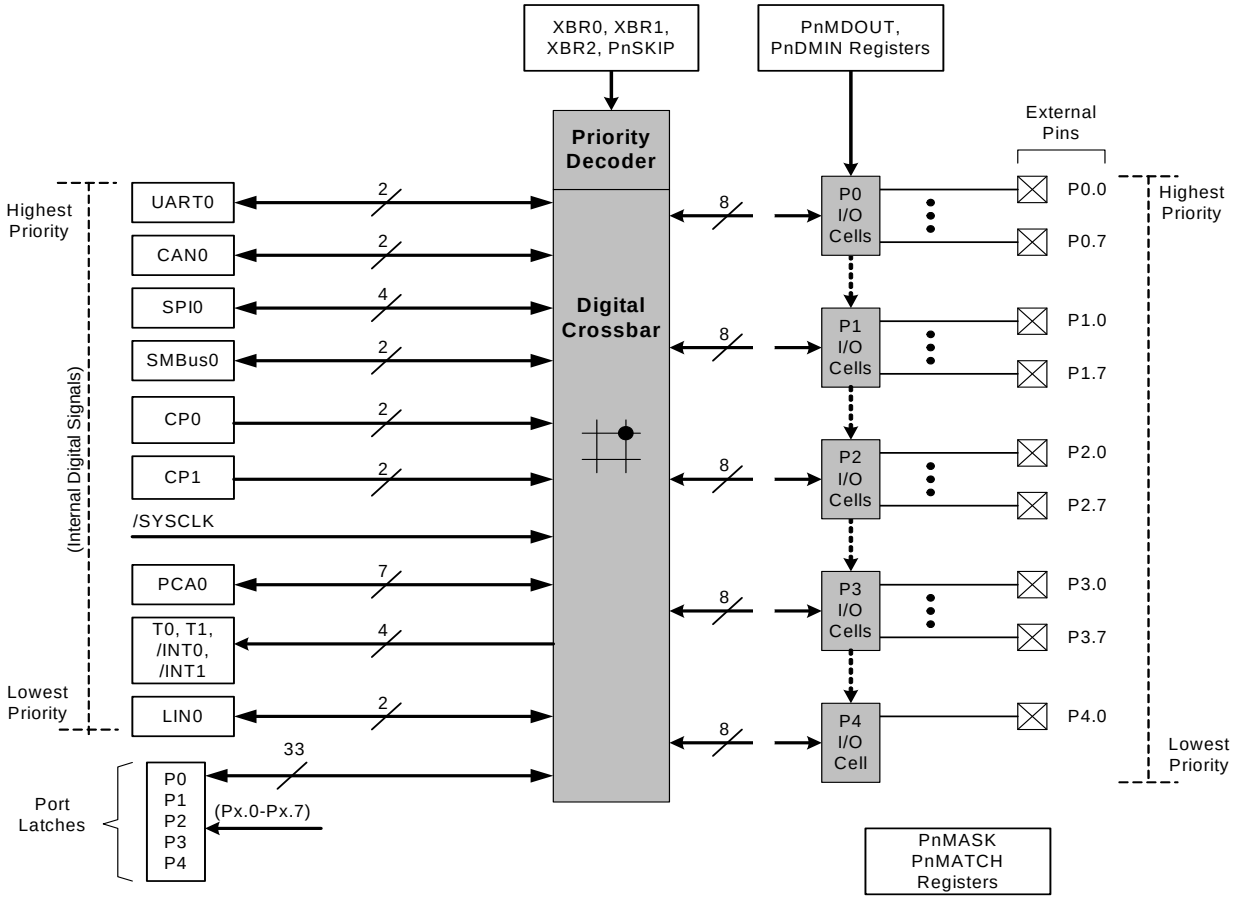

Figure 19.1. Port I/O Functional Block Diagram

SFR Definition 19.10. P3MASK: Port 3 Mask Register

Bit	7	6	5	4	3	2	1	0
Name	P3MASK[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xAF; SFR Page = 0x00

Bit	Name	Function
7:0	P3MASK[7:0]	**Port 1 Mask Value.** Selects P3 pins to be compared to the corresponding bits in P3MAT. 0: P3.n pin logic value is ignored and cannot cause a Port Mismatch event. 1: P3.n pin logic value is compared to P3MAT.n.
Note: P3.0 is available on 40-pin and 32-pin packages. P3.1-P3.7 are available on 40-pin packages		

SFR Definition 19.11. P3MAT: Port 3 Match Register

Bit	7	6	5	4	3	2	1	0
Name	P3MAT[7:0]							
Type	R/W							
Reset	1	1	1	1	1	1	1	1

SFR Address = 0xAE; SFR Page = 0x00

Bit	Name	Function
7:0	P3MAT[7:0]	**Port 3 Match Value.** Match comparison value used on Port 3 for bits in P3MAT which are set to 1. 0: P3.n pin logic value is compared with logic LOW. 1: P3.n pin logic value is compared with logic HIGH.
Note: P3.0 is available on 40-pin and 32-pin packages. P3.1-P3.7 are available on 40-pin packages		

3. The LIN controller does not directly support LIN Version 1.3 Extended Frames. If the application detects an unknown identifier (e.g. extended identifier), it has to write a 1 to the STOP bit (LIN0CTRL.7) instead of setting the DTACK (LIN0CTRL.4) bit. At that time, steps 2 through 5 can then be skipped. In this situation, the LIN controller stops the processing of LIN communication until the next SYNC BREAK is received.
4. Changing the configuration of the checksum during a transaction will cause the interface to reset and the transaction to be lost. To prevent this, the checksum should not be configured while a transaction is in progress. The same applies to changes in the LIN interface mode from slave mode to master mode and from master mode to slave mode.

20.5. Sleep Mode and Wake-Up

To reduce the system's power consumption, the LIN Protocol Specification defines a Sleep Mode. The message used to broadcast a Sleep Mode request must be transmitted by the LIN master application in the same way as a normal transmit message. The LIN slave application must decode the Sleep Mode Frame from the Identifier and data bytes. After that, it has to put the LIN slave node into the Sleep Mode by setting the SLEEP bit (LIN0CTRL.6).

If the SLEEP bit (LIN0CTRL.6) of the LIN slave application is not set and there is no bus activity for four seconds (specified bus idle timeout), the IDLTOUT bit (LIN0ST.6) is set and an interrupt request is generated. After that the application may assume that the LIN bus is in Sleep Mode and set the SLEEP bit (LIN0CTRL.6).

Sending a wake-up signal from the master or any slave node terminates the Sleep Mode of the LIN bus. To send a wake-up signal, the application has to set the WUPREQ bit (LIN0CTRL.1). After successful transmission of the wake-up signal, the DONE bit (LIN0ST.0) of the master node is set and an interrupt request is generated. The LIN slave does not generate an interrupt request after successful transmission of the wake-up signal but it generates an interrupt request if the master does not respond to the wake-up signal within 150 milliseconds. In that case, the ERROR bit (LIN0ST.2) and TOUT bit (LIN0ERR.2) are set. The application then has to decide whether or not to transmit another wake-up signal.

All LIN nodes that detect a wake-up signal will set the WAKEUP (LIN0ST.1) and DONE bits (LIN0ST.0) and generate an interrupt request. After that, the application has to clear the SLEEP bit (LIN0CTRL.6) in the LIN slave.

20.6. Error Detection and Handling

The LIN controller generates an interrupt request and stops the processing of the current frame if it detects an error. The application has to check the type of error by processing LIN0ERR. After that, it has to reset the error register and the ERROR bit (LIN0ST.2) by writing a 1 to the RSTERR bit (LIN0CTRL.2). Starting a new message with the LIN controller selected as master or sending a Wakeup signal with the LIN controller selected as a master or slave is possible only if the ERROR bit (LIN0ST.2) is set to 0.

SFR Definition 20.3. LIN0CF: LIN0 Control Mode Register

Bit	7	6	5	4	3	2	1	0
Name	LINEN	MODE	ABAUD					
Type	R/W	R/W	R/W	R	R	R	R	R
Reset	0	1	1	0	0	0	0	0

SFR Address = 0xC9; SFR Page = 0x0F

Bit	Name	Function
7	LINEN	**LIN Interface Enable Bit.** 0: LIN0 is disabled. 1: LIN0 is enabled.
6	MODE	**LIN Mode Selection Bit.** 0: LIN0 operates in slave mode. 1: LIN0 operates in master mode.
5	ABAUD	**LIN Mode Automatic Baud Rate Selection.** This bit only has an effect when the MODE bit is configured for slave mode. 0: Manual baud rate selection is enabled. 1: Automatic baud rate selection is enabled.
4:0	Unused	Read = 00000b; Write = Don’t Care

SFR Definition 21.1. CAN0CFG: CAN Clock Configuration

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

SFR Address = 0x92; SFR Page = 0x0C

Bit	Name	Function
7:2	Unused	Read = 000000b; Write = Don't Care.
1:0	SYSDIV[1:0]	**CAN System Clock Divider Bits.** The CAN controller clock is derived from the CIP-51 system clock. The CAN controller clock must be less than or equal to 25 MHz. 00: CAN controller clock = System Clock/1. 01: CAN controller clock = System Clock/2. 10: CAN controller clock = System Clock/4. 11: CAN controller clock = System Clock/8.

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.

23.3. Configuration and Operation

UART0 provides standard asynchronous, full duplex communication. It can operate in a point-to-point serial communications application, or as a node on a multi-processor serial interface. To operate in a point-to-point application, where there are only two devices on the serial bus, the MCE0 bit in SMOD0 should be cleared to 0. For operation as part of a multi-processor communications bus, the MCE0 and XBE0 bits should both be set to 1. In both types of applications, data is transmitted from the microcontroller on the TX0 pin, and received on the RX0 pin. The TX0 and RX0 pins are configured using the crossbar and the Port I/O registers, as detailed in Section “19. Port Input/Output” on page 169.

In typical UART communications, The transmit (TX) output of one device is connected to the receive (RX) input of the other device, either directly or through a bus transceiver, as shown in Figure 23.5.

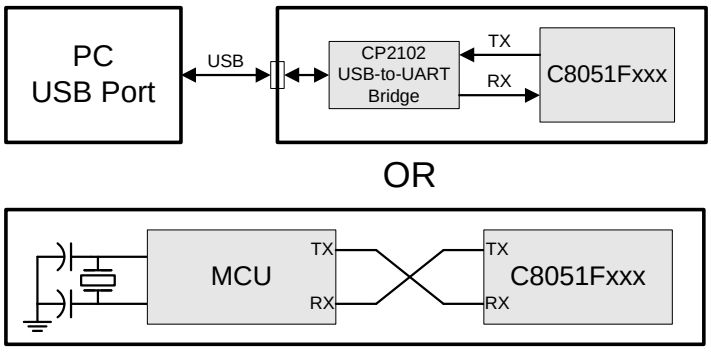

Figure 23.5. Typical UART Interconnect Diagram

23.3.1. Data Transmission

Data transmission is double-buffered and begins when software writes a data byte to the SBUF0 register. Writing to SBUF0 places data in the Transmit Holding Register, and the Transmit Holding Register Empty flag (THRE0) will be cleared to 0. If the UART’s shift register is empty (i.e., no transmission in progress), the data will be placed in the Transmit Holding Register until the current transmission is complete. The TI0 Transmit Interrupt Flag (SCON0.1) will be set at the end of any transmission (the beginning of the stop-bit time). If enabled, an interrupt will occur when TI0 is set.

Note: 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.

If the extra bit function is enabled (XBE0 = 1) and the parity function is disabled (PE0 = ‘0’), the value of the TBX0 (SCON0.3) bit will be sent in the extra bit position. When the parity function is enabled (PE0 = 1), hardware will generate the parity bit according to the selected parity type (selected with S0PT[1:0]), and append it to the data field. Note: when parity is enabled, the extra bit function is not available.

23.3.2. Data Reception

Data reception can begin any time after the REN0 Receive Enable bit (SCON0.4) is set to logic 1. After the stop bit is received, the data byte will be stored in the receive FIFO if the following conditions are met: the receive FIFO (3 bytes deep) must not be full, and the stop bit(s) must be logic 1. In the event that the receive FIFO is full, the incoming byte will be lost, and a Receive FIFO Overrun Error will be generated (OVR0 in register SCON0 will be set to logic 1). If the stop bit(s) were logic 0, the incoming data will not be stored in the receive FIFO. If the reception conditions are met, the data is stored in the receive FIFO, and

SFR Definition 23.1. SCON0: Serial Port 0 Control

Bit	7	6	5	4	3	2	1	0
Name	OVR0	PERR0	THRE0	REN0	TBX0	RBX0	TI0	RI0
Type	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
Reset	0	0	1	0	0	0	0	0

SFR Definition 25.8. TMR2CN: Timer 2 Control

Bit	7	6	5	4	3	2	1	0
Name	TF2H	TF2L	TF2LEN	TF2CEN	T2SPLIT	TR2		T2XCLK
Type	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xC8; Bit-Addressable; SFR Page = 0x00

Bit	Name	Function
7	TF2H	**Timer 2 High Byte Overflow Flag.** Set by hardware when the Timer 2 high byte overflows from 0xFF to 0x00. In 16 bit mode, this will occur when Timer 2 overflows from 0xFFFF to 0x0000. When the Timer 2 interrupt is enabled, setting this bit causes the CPU to vector to the Timer 2 interrupt service routine. This bit is not automatically cleared by hardware.
6	TF2L	**Timer 2 Low Byte Overflow Flag.** Set by hardware when the Timer 2 low byte overflows from 0xFF to 0x00. TF2L will be set when the low byte overflows regardless of the Timer 2 mode. This bit is not automatically cleared by hardware.
5	TF2LEN	**Timer 2 Low Byte Interrupt Enable.** When set to 1, this bit enables Timer 2 Low Byte interrupts. If Timer 2 interrupts are also enabled, an interrupt will be generated when the low byte of Timer 2 overflows.
4	TF2CEN	**Timer 2 Capture Mode Enable.** 0: Timer 2 Capture Mode is disabled. 1: Timer 2 Capture Mode is enabled.
3	T2SPLIT	**Timer 2 Split Mode Enable.** When this bit is set, Timer 2 operates as two 8-bit timers with auto-reload. 0: Timer 2 operates in 16-bit auto-reload mode. 1: Timer 2 operates as two 8-bit auto-reload timers.
2	TR2	**Timer 2 Run Control.** Timer 2 is enabled by setting this bit to 1. In 8-bit mode, this bit enables/disables TMR2H only; TMR2L is always enabled in split mode.
1	Unused	Read = 0b; Write = Don’t Care
0	T2XCLK	**Timer 2 External Clock Select.** This bit selects the external clock source for Timer 2. If Timer 2 is in 8-bit mode, this bit selects the external oscillator clock source for both timer bytes. However, the Timer 2 Clock Select bits (T2MH and T2ML in register CKCON) may still be used to select between the external clock and the system clock for either timer. 0: Timer 2 clock is the system clock divided by 12. 1: Timer 2 clock is the external clock divided by 8 (synchronized with SYSCLK).

Table 26.3. Watchdog Timer Timeout Intervals[1]

System Clock (Hz)	PCA0CPL5	Timeout Interval (ms)
24,000,000	255	32.8
24,000,000	128	16.5
24,000,000	32	4.2
3,000,000	255	262.1
3,000,000	128	132.1
3,000,000	32	33.8
187,500[2]	255	4194
187,500[2]	128	2114
187,500[2]	32	541

Notes:

1. Assumes SYSCLK/12 as the PCA clock source, and a PCA0L value of 0x00 at the update time.
2. Internal SYSCLK reset frequency = Internal Oscillator divided by 128.