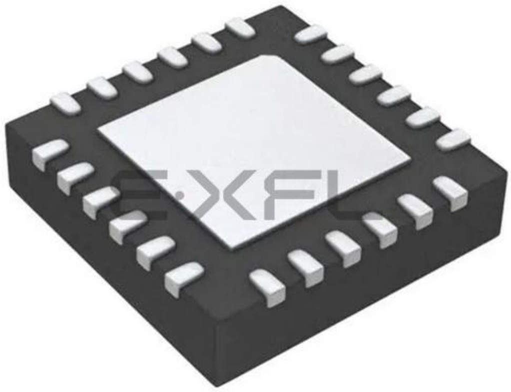

Welcome to E-XFL.COM

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), LINbus, SPI, UART/USART
Peripherals	POR, PWM, Temp Sensor, WDT
Number of I/O	18
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 18x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	24-WFQFN Exposed Pad
Supplier Device Package	24-QFN (4x4)
Purchase URL	https://www.e-xfl.com/product-detail/silicon-labs/c8051f556-im

1. System Overview

C8051F55x/56x/57x devices are fully integrated mixed-signal System-on-a-Chip MCUs. Highlighted features are listed below. Refer to Table 2.1 for specific product feature selection and part ordering numbers.

- High-speed pipelined 8051-compatible microcontroller core (up to 50 MIPS)
- In-system, full-speed, non-intrusive debug interface (on-chip)
- Controller Area Network (CAN 2.0B) Controller with 32 message objects, each with its own indentifier mask (C8051F550/1/4/5, ‘F560/1/4/5/8/9, and ‘F572/3)
- LIN 2.1 peripheral (fully backwards compatible, master and slave modes) (C8051F550/2/4/6, ‘F560/2/4/6/8, and ‘F570/2/4)
- True 12-bit 200 ksps 32-channel single-ended ADC with analog multiplexer
- Precision programmable 24 MHz internal oscillator that is within ±0.5% across the temperature range and for VDD voltages greater than or equal to the on-chip voltage regulator minimum output at the low setting. The oscillator is within ±1.0% for VDD voltages below this minimum output setting.
- On-chip Clock Multiplier to reach up to 50 MHz
- 32 kB (C8051F550-3, ‘F560-3, ‘F568-9, and ‘F570-1) or 16 kB (C8051F554-7, ‘F564-7, and ‘F572-5) of on-chip Flash memory
- 2304 bytes of on-chip RAM
- SMBus/I2C, Enhanced UART, and Enhanced SPI serial interfaces implemented in hardware
- Four general-purpose 16-bit timers
- External Data Memory Interface (C8051F568-9 and ‘F570-5) with 64 kB address space
- Programmable Counter/Timer Array (PCA) with six capture/compare modules and Watchdog Timer function
- On-chip Voltage Regulator
- On-chip Power-On Reset, V_{DD} Monitor, and Temperature Sensor
- On-chip Voltage Comparator
- 33, 25, or 18 Port I/O (5 V push-pull)

With on-chip Voltage Regulator, Power-On Reset, V_{DD} monitor, Watchdog Timer, and clock oscillator, the C8051F55x/56x/57x devices are truly stand-alone System-on-a-Chip solutions. The Flash memory can be reprogrammed even in-circuit, providing non-volatile data storage, and also allowing field upgrades of the 8051 firmware. User software has complete control of all peripherals, and may individually shut down any or all peripherals for power savings.

The on-chip Silicon Labs 2-Wire (C2) Development Interface allows non-intrusive (uses no on-chip resources), full speed, in-circuit debugging using the production MCU installed in the final application. This debug logic supports inspection and modification of memory and registers, setting breakpoints, single stepping, run and halt commands. All analog and digital peripherals are fully functional while debugging using C2. The two C2 interface pins can be shared with user functions, allowing in-system debugging without occupying package pins.

The devices are specified for 1.8 V to 5.25 V operation over the automotive temperature range (–40 to +125 °C). The C8051F568-9 and ‘F570-5 are available in 40-pin QFN packages, the C8051F560-7 devices are available in 32-pin QFP and QFN packages, and the C8051F550-7 are available in 24-pin QFN packages. All package options are lead-free and RoHS compliant. See Table 2.1 for ordering information. Block diagrams are included in Figure 1.1, Figure 1.2, and Figure 1.3.

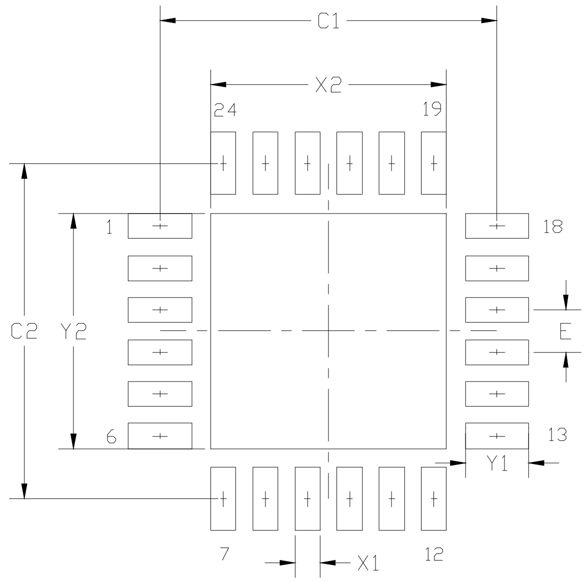

Figure 4.8. QFN-24 Landing Diagram

Table 4.8. QFN-24 Landing Diagram Dimensions

Dimension	Min	Max	Dimension	Min	Max
C1	3.90	4.00	X2	2.70	2.80
C2	3.90	4.00	Y1	0.65	0.75
E	0.50 BSC		Y2	2.70	2.80
X1	0.20	0.30			

Notes:

General

1. All dimensions shown are in millimeters (mm) unless otherwise noted.
2. This Land Pattern Design is based on the IPC-7351 guidelines.

Solder Mask Design

3. All metal pads are to be non-solder mask defined (NSMD). Clearance between the solder mask and the metal pad is to be 60 µm minimum, all the way around the pad.

Stencil Design

4. A stainless steel, laser-cut and electro-polished stencil with trapezoidal walls should be used to assure good solder paste release.
5. The stencil thickness should be 0.125 mm (5 mils).
6. The ratio of stencil aperture to land pad size should be 1:1 for all perimeter pads.
7. A 2x2 array of 1.10 mm x 1.10 mm openings on a 1.30 mm pitch should be used for the center ground pad.

Card Assembly

8. A No-Clean, Type-3 solder paste is recommended.
9. The recommended card reflow profile is per the JEDEC/IPC J-STD-020 specification for Small Body Components.

6.2. Output Code Formatting

The registers ADC0H and ADC0L contain the high and low bytes of the output conversion code. When the repeat count is set to 1, conversion codes are represented in 12-bit unsigned integer format and the output conversion code is updated after each conversion. Inputs are measured from 0 to V_{REF} x 4095/4096. Data can be right-justified or left-justified, depending on the setting of the AD0LJST bit (ADC0CN.2). Unused bits in the ADC0H and ADC0L registers are set to 0. Example codes are shown below for both right-justified and left-justified data.

Input Voltage	Right-Justified ADC0H:ADC0L (AD0LJST = 0)	Left-Justified ADC0H:ADC0L (AD0LJST = 1)
VREF x 4095/4096	0x0FFF	0xFFF0
VREF x 2048/4096	0x0800	0x8000
VREF x 2047/4096	0x07FF	0x7FF0
0	0x0000	0x0000

When the ADC0 Repeat Count is greater than 1, the output conversion code represents the accumulated result of the conversions performed and is updated after the last conversion in the series is finished. Sets of 4, 8, or 16 consecutive samples can be accumulated and represented in unsigned integer format. The repeat count can be selected using the AD0RPT bits in the ADC0CF register. The value must be right-justified (AD0LJST = 0), and unused bits in the ADC0H and ADC0L registers are set to 0. The following example shows right-justified codes for repeat counts greater than 1. Notice that accumulating 2^n samples is equivalent to left-shifting by *n* bit positions when all samples returned from the ADC have the same value.

Input Voltage	Repeat Count = 4	Repeat Count = 8	Repeat Count = 16
V_{REF} x 4095/4096	0x3FFC	0x7FF8	0xFFF0
V_{REF} x 2048/4096	0x2000	0x4000	0x8000
V_{REF} x 2047/4096	0x1FFC	0x3FF8	0x7FF0
0	0x0000	0x0000	0x0000

6.2.1. Settling Time Requirements

A minimum tracking time is required before an accurate conversion is performed. This tracking time is determined by any series impedance, including the AMUX0 resistance, the ADC0 sampling capacitance, and the accuracy required for the conversion.

Figure 6.5 shows the equivalent ADC0 input circuit. The required ADC0 settling time for a given settling accuracy (SA) may be approximated by Equation 6.1. When measuring the Temperature Sensor output, use the settling time specified in Table 5.10. When measuring V_{DD} with respect to GND, R_{TOTAL} reduces to R_{MUX}. See Table 5.9 for ADC0 minimum settling time requirements as well as the mux impedance and sampling capacitor values.

$$t = \ln\left(\frac{2^n}{SA}\right) \times R_{TOTAL} C_{SAMPLE}$$

Equation 6.1. ADC0 Settling Time Requirements

Where:
SA is the settling accuracy, given as a fraction of an LSB (for example, 0.25 to settle within 1/4 LSB).
t is the required settling time in seconds. R_{TOTAL} is the sum of the AMUX0 resistance and any external source resistance. *n* is the ADC resolution in bits (10).

SFR Definition 6.5. ADC0H: ADC0 Data Word MSB

Bit	7	6	5	4	3	2	1	0
Name	ADC0H[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xBE; SFR Page = 0x00

Bit	Name	Function
7:0	ADC0H[7:0]	**ADC0 Data Word High-Order Bits.** For AD0LJST = 0 and AD0RPT as follows: 00: Bits 3–0 are the upper 4 bits of the 12-bit result. Bits 7–4 are 0000b. 01: Bits 4–0 are the upper 5 bits of the 14-bit result. Bits 7–5 are 000b. 10: Bits 5–0 are the upper 6 bits of the 15-bit result. Bits 7–6 are 00b. 11: Bits 7–0 are the upper 8 bits of the 16-bit result. For AD0LJST = 1 (AD0RPT must be 00): Bits 7–0 are the most-significant bits of the ADC0 12-bit result.

SFR Definition 6.6. ADC0L: ADC0 Data Word LSB

Bit	7	6	5	4	3	2	1	0
Name	ADC0L[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xBD; SFR Page = 0x00

Bit	Name	Function
7:0	ADC0L[7:0]	**ADC0 Data Word Low-Order Bits.** For AD0LJST = 0: Bits 7–0 are the lower 8 bits of the ADC0 Accumulated Result. For AD0LJST = 1 (AD0RPT must be '00'): Bits 7–4 are the lower 4 bits of the 12-bit result. Bits 3–0 are 0000b.

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.

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.

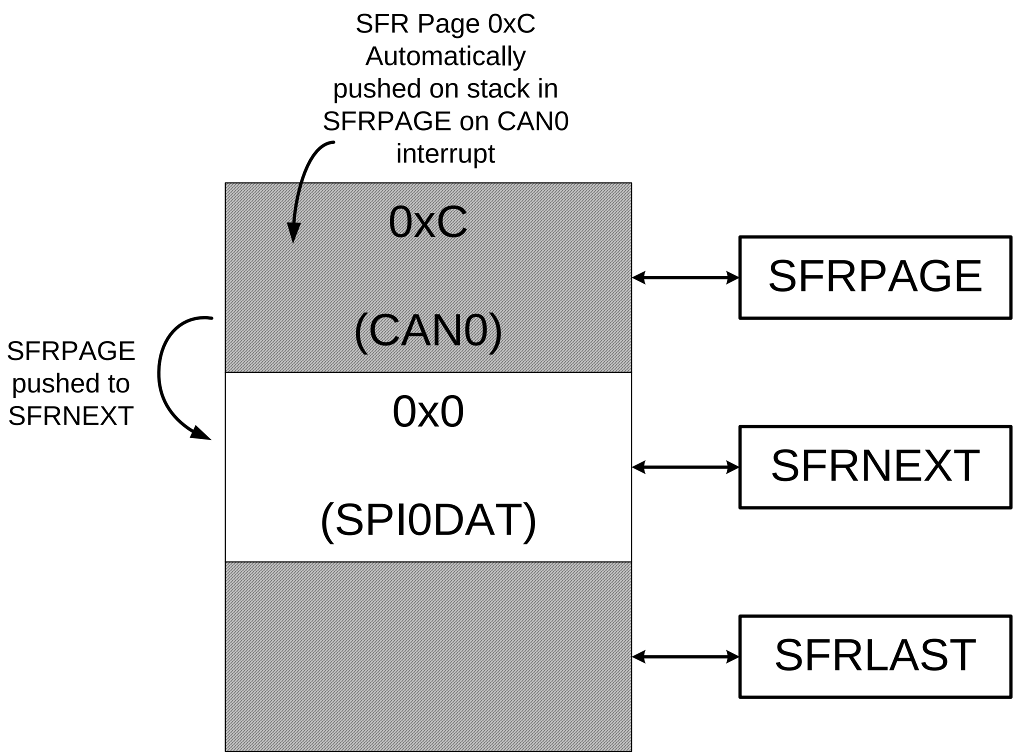

Figure 12.3. SFR Page Stack After CAN0 Interrupt Occurs

13.1.1. Interrupt Priorities

Each interrupt source can be individually programmed to one of two priority levels: low or high. A low priority interrupt service routine can be preempted by a high priority interrupt. A high priority interrupt cannot be preempted. Each interrupt has an associated interrupt priority bit in an SFR (IE, EIP1, or EIP2) used to configure its priority level. Low priority is the default. If two interrupts are recognized simultaneously, the interrupt with the higher priority is serviced first. If both interrupts have the same priority level, a fixed priority order is used to arbitrate, given in Table 13.1.

13.1.2. Interrupt Latency

Interrupt response time depends on the state of the CPU when the interrupt occurs. Pending interrupts are sampled and priority decoded each system clock cycle. Therefore, the fastest possible response time is 5 system clock cycles: 1 clock cycle to detect the interrupt and 4 clock cycles to complete the LCALL to the ISR. If an interrupt is pending when a RETI is executed, a single instruction is executed before an LCALL is made to service the pending interrupt. Therefore, the maximum response time for an interrupt (when no other interrupt is currently being serviced or the new interrupt is of greater priority) occurs when the CPU is performing an RETI instruction followed by a DIV as the next instruction. In this case, the response time is 18 system clock cycles: 1 clock cycle to detect the interrupt, 5 clock cycles to execute the RETI, 8 clock cycles to complete the DIV instruction and 4 clock cycles to execute the LCALL to the ISR. If the CPU is executing an ISR for an interrupt with equal or higher priority, the new interrupt will not be serviced until the current ISR completes, including the RETI and following instruction.

SFR Definition 13.2. IP: Interrupt Priority

Bit	7	6	5	4	3	2	1	0
Name		PSPI0	PT2	PS0	PT1	PX1	PT0	PX0
Type	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	0	0	0	0	0	0	0

SFR Address = 0xB8; Bit-Addressable; SFR Page = All Pages

Bit	Name	Function
7	Unused	Read = 1b, Write = Don't Care.
6	PSPI0	**Serial Peripheral Interface (SPI0) Interrupt Priority Control.** This bit sets the priority of the SPI0 interrupt. 0: SPI0 interrupt set to low priority level. 1: SPI0 interrupt set to high priority level.
5	PT2	**Timer 2 Interrupt Priority Control.** This bit sets the priority of the Timer 2 interrupt. 0: Timer 2 interrupt set to low priority level. 1: Timer 2 interrupt set to high priority level.
4	PS0	**UART0 Interrupt Priority Control.** This bit sets the priority of the UART0 interrupt. 0: UART0 interrupt set to low priority level. 1: UART0 interrupt set to high priority level.
3	PT1	**Timer 1 Interrupt Priority Control.** This bit sets the priority of the Timer 1 interrupt. 0: Timer 1 interrupt set to low priority level. 1: Timer 1 interrupt set to high priority level.
2	PX1	**External Interrupt 1 Priority Control.** This bit sets the priority of the External Interrupt 1 interrupt. 0: External Interrupt 1 set to low priority level. 1: External Interrupt 1 set to high priority level.
1	PT0	**Timer 0 Interrupt Priority Control.** This bit sets the priority of the Timer 0 interrupt. 0: Timer 0 interrupt set to low priority level. 1: Timer 0 interrupt set to high priority level.
0	PX0	**External Interrupt 0 Priority Control.** This bit sets the priority of the External Interrupt 0 interrupt. 0: External Interrupt 0 set to low priority level. 1: External Interrupt 0 set to high priority level.

14. Flash Memory

On-chip, re-programmable Flash memory is included for program code and non-volatile data storage. The Flash memory can be programmed in-system, a single byte at a time, through the C2 interface or by software using the MOVX instruction. Once cleared to logic 0, a Flash bit must be erased to set it back to logic 1. Flash bytes would typically be erased (set to 0xFF) before being reprogrammed. The write and erase operations are automatically timed by hardware for proper execution; data polling to determine the end of the write/erase operation is not required. Code execution is stalled during a Flash write/erase operation. Refer to Table 5.5 for complete Flash memory electrical characteristics.

14.1. Programming The Flash Memory

The simplest means of programming the Flash memory is through the C2 interface using programming tools provided by Silicon Labs or a third party vendor. This is the only means for programming a non-initialized device. For details on the C2 commands to program Flash memory, see Section “27. C2 Interface” on page 300.

To ensure the integrity of Flash contents, it is strongly recommended that the on-chip V_{DD} Monitor be enabled in any system that includes code that writes and/or erases Flash memory from software. See Section 14.4 for more details. Before performing any Flash write or erase procedure, set the FLEWT bit in Flash Scale register (FLSCL) to 1. Also, note that 8-bit MOVX instructions cannot be used to erase or write to Flash memory at addresses higher than 0x00FF.

For –I (Industrial Grade) parts, parts programmed at a cold temperature below 0 °C may exhibit weakly programmed flash memory bits. If programmed at 0 °C or higher, there is no problem reading Flash across the entire temperature range of -40 °C to 125 °C. This temperature restriction does not apply to –A (Automotive Grade) devices.

14.1.1. Flash Lock and Key Functions

Flash writes and erases by user software are protected with a lock and key function. The Flash Lock and Key Register (FLKEY) must be written with the correct key codes, in sequence, before Flash operations may be performed. The key codes are: 0xA5, 0xF1. The timing does not matter, but the codes must be written in order. If the key codes are written out of order, or the wrong codes are written, Flash writes and erases will be disabled until the next system reset. Flash writes and erases will also be disabled if a Flash write or erase is attempted before the key codes have been written properly. The Flash lock resets after each write or erase; the key codes must be written again before a following Flash operation can be performed. The FLKEY register is detailed in SFR Definition 14.2.

15.2. Stop Mode

Setting the Stop Mode Select bit (PCON.1) causes the controller core to enter Stop mode as soon as the instruction that sets the bit completes execution. In Stop mode the internal oscillator, CPU, and all digital peripherals are stopped; the state of the external oscillator circuit is not affected. Each analog peripheral (including the external oscillator circuit) may be shut down individually prior to entering Stop Mode. Stop mode can only be terminated by an internal or external reset. On reset, the device performs the normal reset sequence and begins program execution at address 0x0000.

If enabled, the Missing Clock Detector will cause an internal reset and thereby terminate the Stop mode. The Missing Clock Detector should be disabled if the CPU is to be put to in STOP mode for longer than the MCD timeout of 100 µs.

15.3. Suspend Mode

Setting the SUSPEND bit (OSCICN.5) causes the hardware to halt the CPU and the high-frequency internal oscillator, and go into Suspend mode as soon as the instruction that sets the bit completes execution. All internal registers and memory maintain their original data. Most digital peripherals are not active in Suspend mode. The exception to this is the Port Match feature.

Suspend mode can be terminated by three types of events, a port match (described in Section “19.5. Port Match” on page 179), a Comparator low output (if enabled), or a device reset event. When Suspend mode is terminated, the device will continue execution on the instruction following the one that set the SUSPEND bit. If the wake event was configured to generate an interrupt, the interrupt will be serviced upon waking the device. If Suspend mode is terminated by an internal or external reset, the CIP-51 performs a normal reset sequence and begins program execution at address 0x0000.

Note: Before entering suspend mode, firmware must set the ZTCEN bit in REF0CN (SFR Definition 7.1).

SFR Definition 17.1. EMI0CN: External Memory Interface Control

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

SFR Address = 0xAA; SFR Page = 0x00

Bit	Name	Function
7:0	PGSEL[7:0]	**XRAM Page Select Bits.** The XRAM Page Select Bits provide the high byte of the 16-bit external data memory address when using an 8-bit MOVX command, effectively selecting a 256-byte page of RAM. 0x00: 0x0000 to 0x00FF 0x01: 0x0100 to 0x01FF ... 0xFE: 0xFE00 to 0xFEFF 0xFF: 0xFF00 to 0xFFFF

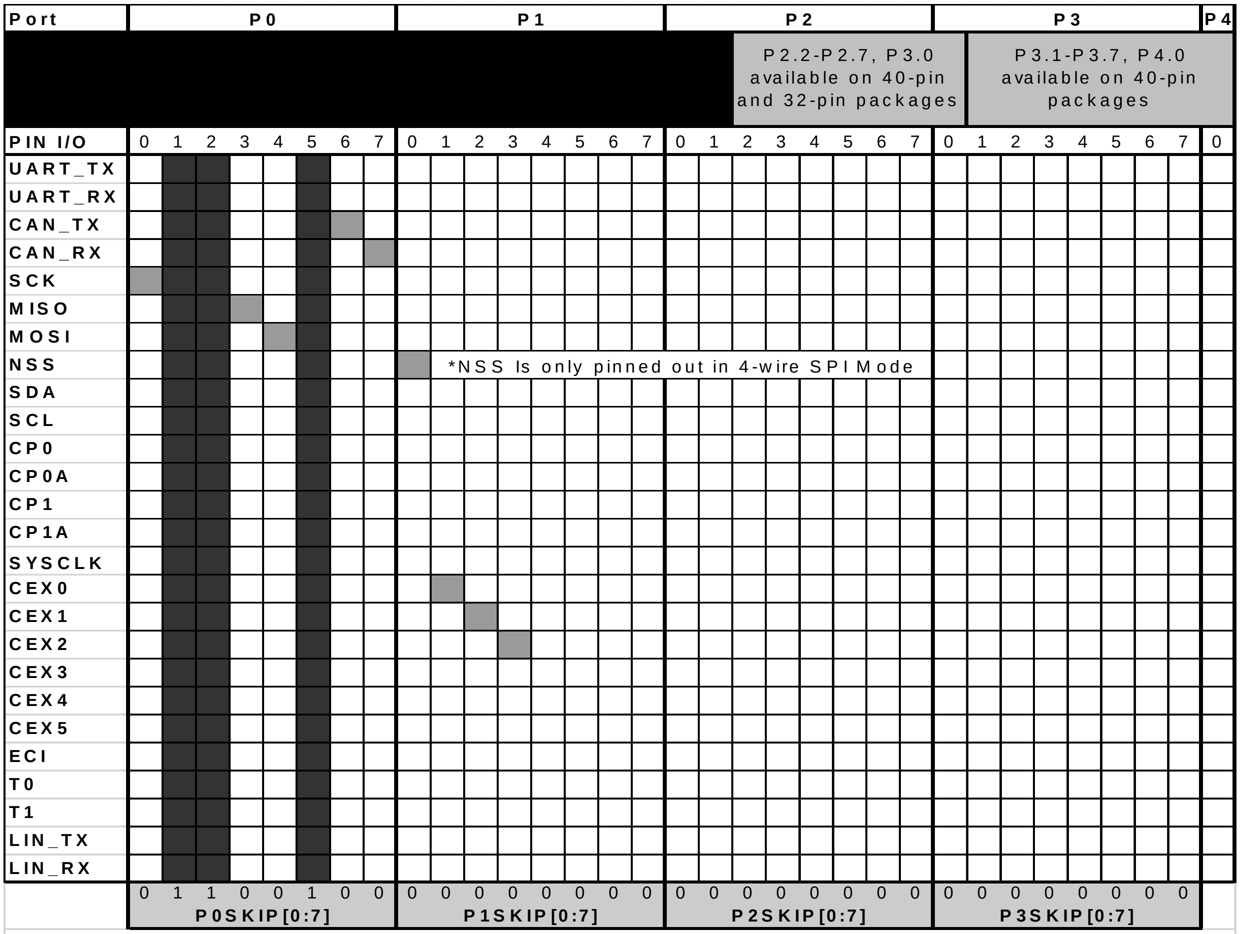

Figure 19.4. Crossbar Priority Decoder in Example Configuration

19.4. Port I/O Initialization

Port I/O initialization consists of the following steps:

1. Select the input mode (analog or digital) for all Port pins, using the Port Input Mode register (PnMDIN).
2. Select the output mode (open-drain or push-pull) for all Port pins, using the Port Output Mode register (PnMDOUT).
3. Select any pins to be skipped by the I/O Crossbar using the Port Skip registers (PnSKIP).
4. Assign Port pins to desired peripherals.
5. Enable the Crossbar (XBARE = 1).

All Port pins must be configured as either analog or digital inputs. Port 4 C8051F568-9 and ‘F570-5 is a digital-only Port. Any pins to be used as Comparator or ADC inputs should be configured as an analog inputs. When a pin is configured as an analog input, its weak pullup, digital driver, and digital receiver are disabled. This process saves power and reduces noise on the analog input. Pins configured as digital inputs may still be used by analog peripherals; however this practice is not recommended.

Additionally, all analog input pins should be configured to be skipped by the Crossbar (accomplished by setting the associated bits in PnSKIP). Port input mode is set in the PnMDIN register, where a 1 indicates a digital input, and a 0 indicates an analog input. All pins default to digital inputs on reset. See SFR Definition 19.13 for the PnMDIN register details.

The output driver characteristics of the I/O pins are defined using the Port Output Mode registers (PnMDOUT). Each Port Output driver can be configured as either open drain or push-pull. This selection is required even for the digital resources selected in the XBRn registers, and is not automatic. The only exception to this is the SMBus (SDA, SCL) pins, which are configured as open-drain regardless of the PnMDOUT settings. When the WEAKPUD bit in XBR2 is 0, a weak pullup is enabled for all Port I/O configured as open-drain. WEAKPUD does not affect the push-pull Port I/O. Furthermore, the weak pullup is turned off on an output that is driving a 0 to avoid unnecessary power dissipation.

Registers XBR0, XBR1, and XBR2 must be loaded with the appropriate values to select the digital I/O functions required by the design. Setting the XBARE bit in XBR2 to 1 enables the Crossbar. Until the Crossbar is enabled, the external pins remain as standard Port I/O (in input mode), regardless of the XBRn Register settings. For given XBRn Register settings, one can determine the I/O pin-out using the Priority Decode Table; as an alternative, the Configuration Wizard utility of the Silicon Labs IDE software will determine the Port I/O pin-assignments based on the XBRn Register settings.

The Crossbar must be enabled to use Port pins as standard Port I/O in output mode. Port output drivers are disabled while the Crossbar is disabled.

SFR Definition 19.3. XBR2: Port I/O Crossbar Register 1

Bit	7	6	5	4	3	2	1	0
Name	WEAKPUD	XBARE	Reserved					LIN0E
Type	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xC7; SFR Page = 0x0F

Bit	Name	Function
7	WEAKPUD	**Port I/O Weak Pullup Disable.** 0: Weak Pullups enabled (except for Ports whose I/O are configured for analog mode). 1: Weak Pullups disabled.
6	XBARE	**Crossbar Enable.** 0: Crossbar disabled. 1: Crossbar enabled.
5:1	Reserved	Always Write to 00000b.
0	LIN0E	**LIN I/O Output Enable.** 0: LIN I/O unavailable at Port pin. 1: LIN_TX, LIN_RX routed to Port pins.

Table 20.3. Autobaud Parameters Examples

System Clock (MHz)	Prescaler	Divider
25	1	312
24.5	1	306
24	1	300
22.1184	1	276
16	1	200
12.25	0	306
12	0	300
11.0592	0	276
8	0	200

20.3. LIN Master Mode Operation

The master node is responsible for the scheduling of messages and sends the header of each frame containing the SYNCH BREAK FIELD, SYNCH FIELD, and IDENTIFIER FIELD. The steps to schedule a message transmission or reception are listed below.

1. Load the 6-bit Identifier into the LIN0ID register.
2. Load the data length into the LIN0SIZE register. Set the value to the number of data bytes or "1111b" if the data length should be decoded from the identifier. Also, set the checksum type, classic or enhanced, in the same LIN0SIZE register.
3. Set the data direction by setting the TXRX bit (LIN0CTRL.5). Set the bit to 1 to perform a master transmit operation, or set the bit to 0 to perform a master receive operation.
4. If performing a master transmit operation, load the data bytes to transmit into the data buffer (LIN0DT1 to LIN0DT8).
5. Set the STREQ bit (LIN0CTRL.0) to start the message transfer. The LIN controller will schedule the message frame and request an interrupt if the message transfer is successfully completed or if an error has occurred.

This code segment shows the procedure to schedule a message in a transmission operation:

```
LIN0ADR   = 0x08;               // Point to LIN0CTRL
LIN0DAT  |= 0x20;               // Select to transmit data
LIN0ADR   = 0x0E;               // Point to LIN0ID
LIN0DAT   = 0x11;               // Load the ID, in this example 0x11
LIN0ADR   = 0x0B;               // Point to LIN0SIZE
LIN0DAT  = ( LIN0DAT & 0xF0 ) | 0x08;      // Load the size with 8

LIN0ADR  = 0x00;               // Point to Data buffer first byte
for (i=0; i<8; i++)
{
   LIN0DAT = i + 0x41;         // Load the buffer with ‘A’, ‘B’, ...
   LIN0ADR++;                  // Increment the address to the next buffer
}
LIN0ADR  = 0x08;               // Point to LIN0CTRL
LIN0DAT  = 0x01;               // Start Request
```

LIN Register Definition 20.9. LIN0DIV: LIN0 Divider Register

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

Indirect Address = 0x0C

Bit	Name	Function
7:0	DIVLSB	**LIN Baud Rate Divider Least Significant Bits.** The 8 least significant bits for the baud rate divider. The 9th and most significant bit is the DIV9 bit (LIN0MUL.0). The valid range for the divider is 200 to 511.

LIN Register Definition 20.10. LIN0MUL: LIN0 Multiplier Register

Bit	7	6	5	4	3	2	1	0
Name	PRESCL[1:0]		LINMUL[4:0]					DIV9
Type	R/W		R/W					R/W
Reset	1	1	1	1	1	1	1	1

Indirect Address = 0x0D

Bit	Name	Function
7:6	PRESCL[1:0]	**LIN Baud Rate Prescaler Bits.** These bits are the baud rate prescaler bits.
5:1	LINMUL[4:0]	**LIN Baud Rate Multiplier Bits.** These bits are the baud rate multiplier bits. These bits are not used in slave mode.
0	DIV9	**LIN Baud Rate Divider Most Significant Bit.** The most significant bit of the baud rate divider. The 8 least significant bits are in LIN0DIV. The valid range for the divider is 200 to 511.

LIN Register Definition 20.11. LIN0ID: LIN0 Identifier Register

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

Indirect Address = 0x0E

Bit	Name	Function
7:6	Unused	Read = 00b; Write = Don't Care.
5:0	ID[5:0]	**LIN Identifier Bits.** These bits form the data identifier. If the LINSIZE bits (LIN0SIZE[3:0]) are 1111b, bits ID[5:4] are used to determine the data size and are interpreted as follows: 00: 2 bytes 01: 2 bytes 10: 4 bytes 11: 8 bytes

T2MH	T2XCLK	TMR2H Clock Source
0	0	SYSCLK/12
0	1	External Clock/8
1	X	SYSCLK

T2ML	T2XCLK	TMR2L Clock Source
0	0	SYSCLK/12
0	1	External Clock/8
1	X	SYSCLK

The TF2H bit is set when TMR2H overflows from 0xFF to 0x00; the TF2L bit is set when TMR2L overflows from 0xFF to 0x00. When Timer 2 interrupts are enabled (IE.5), an interrupt is generated each time TMR2H overflows. If Timer 2 interrupts are enabled and TF2LEN (TMR2CN.5) is set, an interrupt is generated each time either TMR2L or TMR2H overflows. When TF2LEN is enabled, software must check the TF2H and TF2L flags to determine the source of the Timer 2 interrupt. The TF2H and TF2L interrupt flags are not cleared by hardware and must be manually cleared by software.

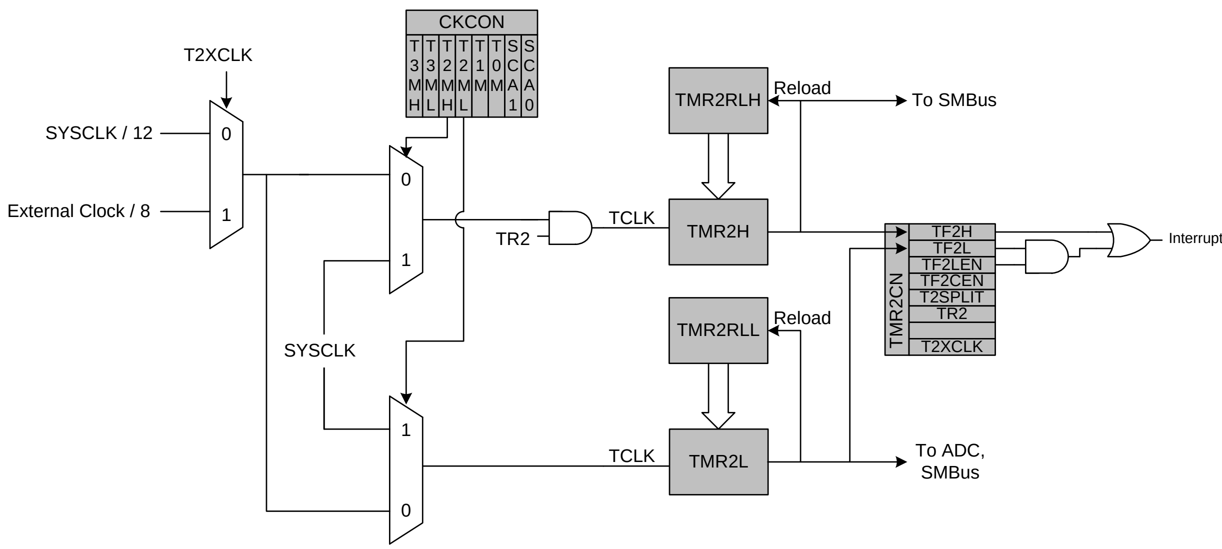

Figure 25.5. Timer 2 8-Bit Mode Block Diagram

25.2.3. External Oscillator Capture Mode

Capture Mode allows the external oscillator to be measured against the system clock. Timer 2 can be clocked from the system clock, or the system clock divided by 12, depending on the T2ML (CKCON.4), and T2XCLK bits. When a capture event is generated, the contents of Timer 2 (TMR2H:TMR2L) are loaded into the Timer 2 reload registers (TMR2RLH:TMR2RLL) and the TF2H flag is set. A capture event is generated by the falling edge of the clock source being measured, which is the external oscillator / 8. By recording the difference between two successive timer capture values, the external oscillator frequency can be determined with respect to the Timer 2 clock. The Timer 2 clock should be much faster than the capture clock to achieve an accurate reading. Timer 2 should be in 16-bit auto-reload mode when using Capture Mode.

For example, if T2ML = 1b and TF2CEN = 1b, Timer 2 will clock every SYSCLK and capture every external clock divided by 8. If the SYSCLK is 24 MHz and the difference between two successive captures is 5984, then the external clock frequency is as follows:

24 MHz/(5984/8) = 0.032086 MHz or 32.086 kHz