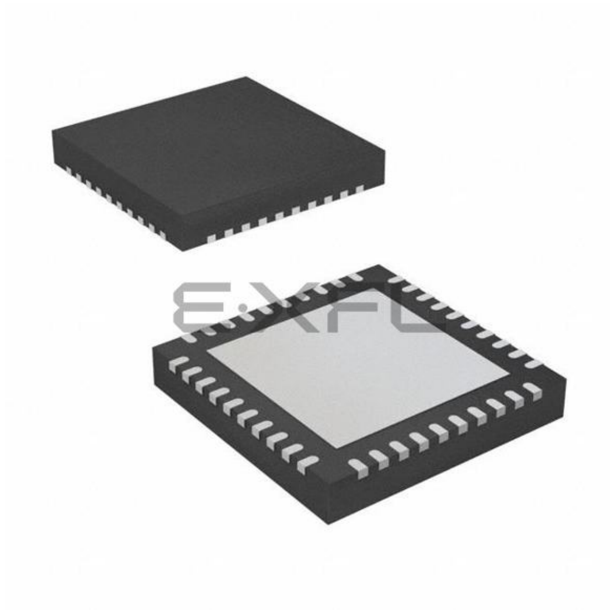

Welcome to E-XFL.COM

## 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 | 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 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/c8051f569-im |

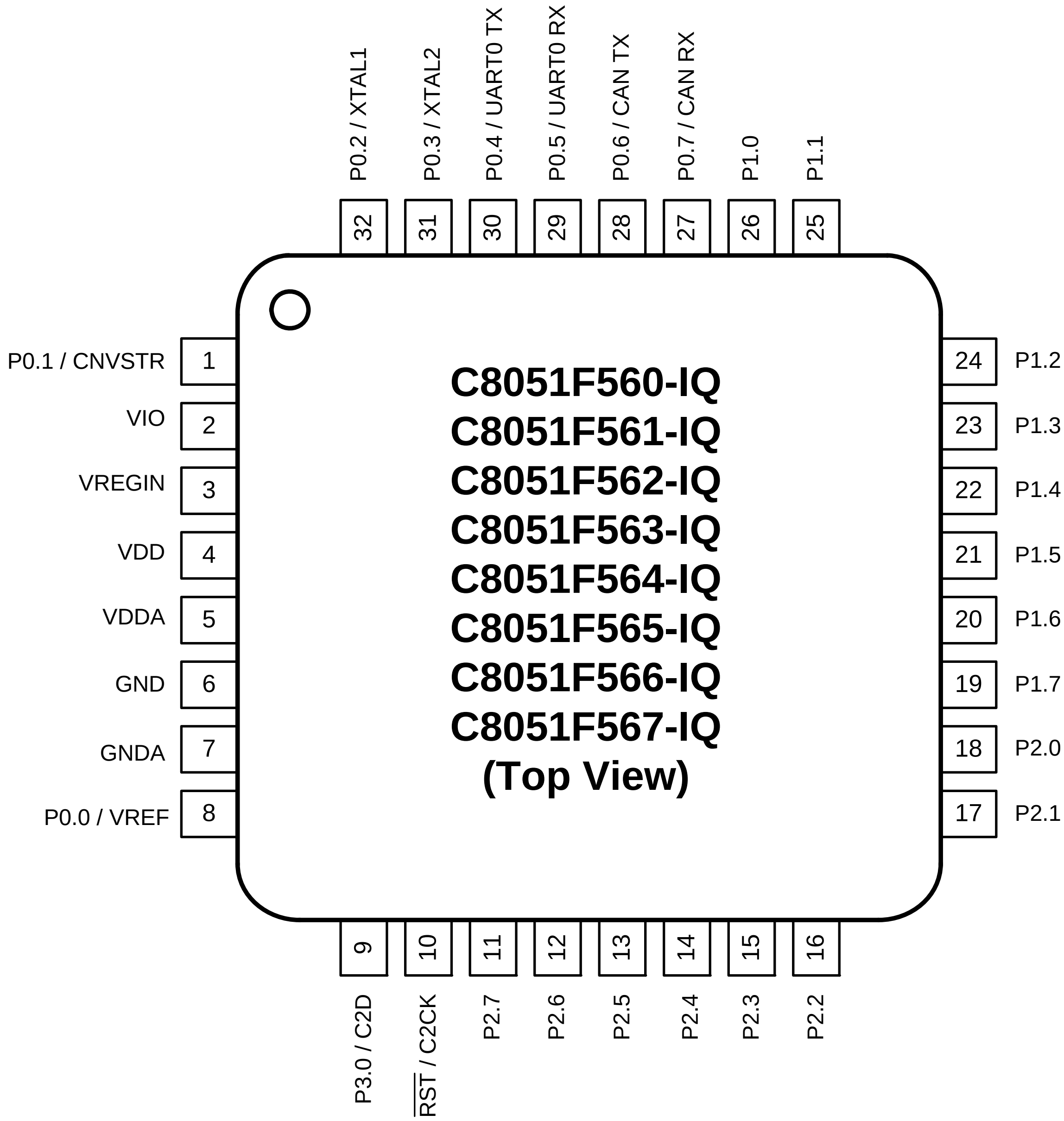


**Figure 3.2. QFP-32 Pinout Diagram (Top View)**

## Gain Register Definition 6.3. ADC0GNA: ADC0 Additional Selectable Gain

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **Name** | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | GAINADD |
| **Type** | W | W | W | W | W | W | W | W |
| **Reset** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

Indirect Address = 0x08;

| Bit | Name | Function |
|---|---|---|
| 7:1 | Reserved | Must Write 0000000b. |
| 0 | GAINADD | **ADC0 Additional Gain Bit.**<br>Setting this bit add 1/64 (0.016) gain to the gain value in the ADC0GNH and ADC0GNL registers. |

**Note:** This register is accessed indirectly; See Section 6.3.2 for details for writing this register.

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

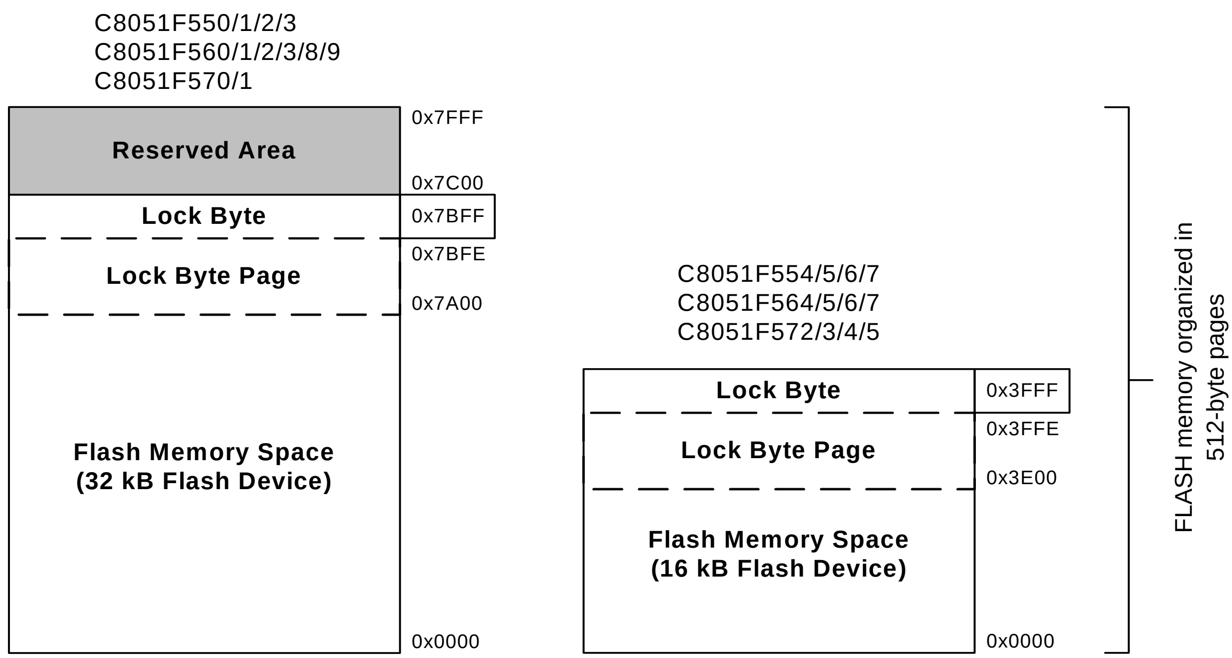


**Figure 11.2. Flash Program Memory Map**

#### 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

## SFR Definition 12.4. SFRLAST: SFR Last

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **Name** | SFRLAST[7:0] | | | | | | | |
| **Type** | R/W | | | | | | | |
| **Reset** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xA7; SFR Page = All Pages

| Bit | Name | Function |
|---|---|---|
| 7:0 | SFRLAST[7:0] | **SFR Page Stack Bits.**<br>This is the value that will go to the SFRNEXT register upon a return from interrupt.<br>Write: Sets the SFR Page in the last entry of the SFR Stack. This will cause the SFRNEXT SFR to have this SFR page value upon a return from interrupt.<br>Read: Returns the value of the SFR page contained in the last entry of the SFR stack.<br>SFR page context is retained upon interrupts/return from interrupts in a 3 byte SFR Page Stack: SFRPAGE is the first entry, SFRNEXT is the second, and SFRLAST is the third entry. The SFR stack bytes may be used alter the context in the SFR Page Stack, and will not cause the stack to “push” or “pop”. Only interrupts and return from interrupts cause pushes and pops of the SFR Page Stack. |

**Table 12.3. Special Function Registers (Continued)**

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

| Register | Address | Description | Page |
|---|---|---|---|
| **PCA0CPH1** | 0xEA | PCA Capture 1 High | 299 |
| **PCA0CPH2** | 0xEC | PCA Capture 2 High | 299 |
| **PCA0CPH3** | 0xEE | PCA Capture 3 High | 299 |
| **PCA0CPH4** | 0xFE | PCA Capture 4 High | 299 |
| **PCA0CPH5** | 0xCF | PCA Capture 5 High | 299 |
| **PCA0CPL0** | 0xFB | PCA Capture 0 Low | 299 |
| **PCA0CPL1** | 0xE9 | PCA Capture 1 Low | 299 |
| **PCA0CPL2** | 0xEB | PCA Capture 2 Low | 299 |
| **PCA0CPL3** | 0xED | PCA Capture 3 Low | 299 |
| **PCA0CPL4** | 0xFD | PCA Capture 4 Low | 299 |
| **PCA0CPL5** | 0xCE | PCA Capture 5 Low | 299 |
| **PCA0CPM0** | 0xDA | PCA Module 0 Mode Register | 297 |
| **PCA0CPM1** | 0xDB | PCA Module 1 Mode Register | 297 |
| **PCA0CPM2** | 0xDC | PCA Module 2 Mode Register | 297 |
| **PCA0CPM3** | 0xDD | PCA Module 3 Mode Register | 297 |
| **PCA0CPM4** | 0xDE | PCA Module 4 Mode Register | 297 |
| **PCA0CPM5** | 0xDF | PCA Module 5 Mode Register | 297 |
| **PCA0H** | 0xFA | PCA Counter High | 298 |
| **PCA0L** | 0xF9 | PCA Counter Low | 298 |
| **PCA0MD** | 0xD9 | PCA Mode | 295 |
| **PCA0PWM** | 0xD9 | PCA PWM Configuration | 296 |
| **PCON** | 0x87 | Power Control | 137 |
| **PSCTL** | 0x8F | Program Store R/W Control | 131 |
| **PSW** | 0xD0 | Program Status Word | 90 |
| **REF0CN** | 0xD1 | Voltage Reference Control | 69 |
| **REG0CN** | 0xC9 | Voltage Regulator Control | 80 |
| **RSTSRC** | 0xEF | Reset Source Configuration/Status | 143 |
| **SBCON0** | 0xAB | UART0 Baud Rate Generator Control | 244 |
| **SBRLH0** | 0xAD | UART0 Baud Rate Reload High Byte | 245 |
| **SBRLL0** | 0xAC | UART0 Baud Rate Reload Low Byte | 245 |
| **SBUF0** | 0x99 | UART0 Data Buffer | 244 |
| **SCON0** | 0x98 | UART0 Control | 241 |
| **SFR0CN** | 0x84 | SFR Page Control | 102 |
| **SFRLAST** | 0x86 | SFR Stack Last Page | 105 |
| **SFRNEXT** | 0x85 | SFR Stack Next Page | 104 |
| **SFRPAGE** | 0xA7 | SFR Page Select | 103 |

# 13. Interrupts

The C8051F55x/56x/57x devices include an extended interrupt system supporting a total of 18 interrupt sources with two priority levels. The allocation of interrupt sources between on-chip peripherals and external inputs pins varies according to the specific version of the device. Each interrupt source has one or more associated interrupt-pending flag(s) located in an SFR. When a peripheral or external source meets a valid interrupt condition, the associated interrupt-pending flag is set to logic 1.

If interrupts are enabled for the source, an interrupt request is generated when the interrupt-pending flag is set. As soon as execution of the current instruction is complete, the CPU generates an LCALL to a predetermined address to begin execution of an interrupt service routine (ISR). Each ISR must end with an RETI instruction, which returns program execution to the next instruction that would have been executed if the interrupt request had not occurred. If interrupts are not enabled, the interrupt-pending flag is ignored by the hardware and program execution continues as normal. (The interrupt-pending flag is set to logic 1 regardless of the interrupt's enable/disable state.)

Each interrupt source can be individually enabled or disabled through the use of an associated interrupt enable bit in an SFR (IE, EIE1, or EIE2). However, interrupts must first be globally enabled by setting the EA bit (IE.7) to logic 1 before the individual interrupt enables are recognized. Setting the EA bit to logic 0 disables all interrupt sources regardless of the individual interrupt-enable settings.

**Note:** Any instruction that clears a bit to disable an interrupt should be immediately followed by an instruction that has two or more opcode bytes. Using EA (global interrupt enable) as an example:

```
// in 'C':
EA = 0; // clear EA bit.
EA = 0; // this is a dummy instruction with two-byte opcode.

; in assembly:
CLR EA ; clear EA bit.
CLR EA ; this is a dummy instruction with two-byte opcode.
```

For example, if an interrupt is posted during the execution phase of a "CLR EA" opcode (or any instruction which clears a bit to disable an interrupt source), and the instruction is followed by a single-cycle instruction, the interrupt may be taken. However, a read of the enable bit will return a 0 inside the interrupt service routine. When the bit-clearing opcode is followed by a multi-cycle instruction, the interrupt will not be taken.

Some interrupt-pending flags are automatically cleared by the hardware when the CPU vectors to the ISR. However, most are not cleared by the hardware and must be cleared by software before returning from the ISR. If an interrupt-pending flag remains set after the CPU completes the return-from-interrupt (RETI) instruction, a new interrupt request will be generated immediately and the CPU will re-enter the ISR after the completion of the next instruction.

## 13.1. MCU Interrupt Sources and Vectors

The C8051F55x/56x/57x MCUs support 18 interrupt sources. Software can simulate an interrupt by setting any interrupt-pending flag to logic 1. If interrupts are enabled for the flag, an interrupt request will be generated and the CPU will vector to the ISR address associated with the interrupt-pending flag. MCU interrupt sources, associated vector addresses, priority order and control bits are summarized in Table 13.1. 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).

## 16.5. Comparator0 Reset

Comparator0 can be configured as a reset source by writing a 1 to the C0RSEF flag (RSTSRC.5). Comparator0 should be enabled and allowed to settle prior to writing to C0RSEF to prevent any turn-on chatter on the output from generating an unwanted reset. The Comparator0 reset is active-low: if the non-inverting input voltage (on CP0+) is less than the inverting input voltage (on CP0–), the device is put into the reset state. After a Comparator0 reset, the C0RSEF flag (RSTSRC.5) will read 1 signifying Comparator0 as the reset source; otherwise, this bit reads 0. The state of the RST pin is unaffected by this reset.

## 16.6. PCA Watchdog Timer Reset

The programmable Watchdog Timer (WDT) function of the Programmable Counter Array (PCA) can be used to prevent software from running out of control during a system malfunction. The PCA WDT function can be enabled or disabled by software as described in Section “26.4. Watchdog Timer Mode” on page 291; the WDT is enabled and clocked by SYSCLK/12 following any reset. If a system malfunction prevents user software from updating the WDT, a reset is generated and the WDTRSF bit (RSTSRC.5) is set to 1. The state of the RST pin is unaffected by this reset.

## 16.7. Flash Error Reset

If a Flash read/write/erase or program read targets an illegal address, a system reset is generated. This may occur due to any of the following:

- A Flash write or erase is attempted above user code space. This occurs when PSWE is set to 1 and a MOVX write operation targets an address in or above the reserved space.
- A Flash read is attempted above user code space. This occurs when a MOVC operation targets an address in or above the reserved space.
- A Program read is attempted above user code space. This occurs when user code attempts to branch to an address in or above the reserved space.
- A Flash read, write or erase attempt is restricted due to a Flash security setting (see Section “14.3. Security Options” on page 127).
- A Flash read, write, or erase is attempted when the VDD Monitor is not enabled to the high threshold and set as a reset source.

The FERROR bit (RSTSRC.6) is set following a Flash error reset. The state of the RST pin is unaffected by this reset.

## 16.8. Software Reset

Software may force a reset by writing a 1 to the SWRSF bit (RSTSRC.4). The SWRSF bit will read 1 following a software forced reset. The state of the RST pin is unaffected by this reset.

### 17.6.1.3. 8-bit MOVX with Bank Select: EMI0CF[4:2] = 010

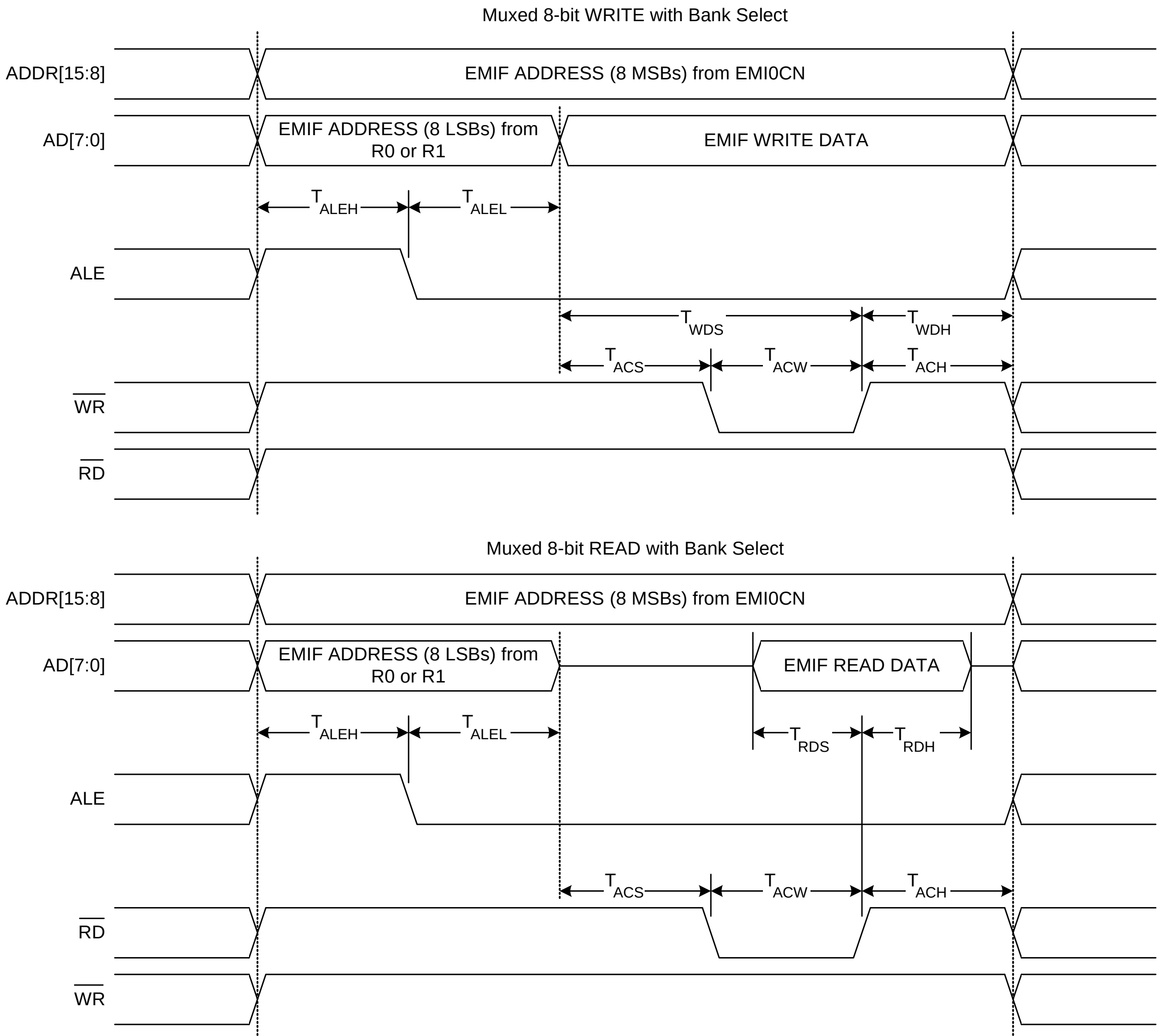


**Figure 17.5. Multiplexed 8-bit MOVX with Bank Select Timing**

## SFR Definition 19.21. P2MDIN: Port 2 Input Mode

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | P2MDIN[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

SFR Address = 0xF3; SFR Page = 0x0F

| Bit | Name | Function |
|---|---|---|
| 7:0 | P2MDIN[7:0] | **Analog Configuration Bits for P2.7–P2.0 (respectively).**<br>Port pins configured for analog mode have their weak pull-up and digital receiver disabled. For analog mode, the pin also needs to be configured for open-drain mode in the P2MDOUT register.<br>0: Corresponding P2.n pin is configured for analog mode.<br>1: Corresponding P2.n pin is not configured for analog mode. |
| **Note:** P2.2-P2.7 are available on 40-pin and 32-pin packages. | | |

## SFR Definition 19.22. P2MDOUT: Port 2 Output Mode

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | P2MDOUT[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xA6; SFR Page = 0x0F

| Bit | Name | Function |
|---|---|---|
| 7:0 | P2MDOUT[7:0] | **Output Configuration Bits for P2.7–P2.0 (respectively).**<br>These bits are ignored if the corresponding bit in register P2MDIN is logic 0.<br>0: Corresponding P2.n Output is open-drain.<br>1: Corresponding P2.n Output is push-pull. |
| **Note:** P2.2-P2.7 are available on 40-pin and 32-pin packages. | | |

## SFR Definition 19.29. P4MDOUT: Port 4 Output Mode

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | P4MDOUT[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xAF; SFR Page = 0x0F

| Bit | Name | Function |
|---|---|---|
| 7:0 | P4MDOUT[7:0] | **Output Configuration Bits for P4.7–P4.0 (respectively).**<br>0: Corresponding P4.n Output is open-drain.<br>1: Corresponding P4.n Output is push-pull. |
| **Note:** Port 4.0 is available on 40-pin packages. | | |

## 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.**<br>0: Use the classic, specification 1.3 compliant checksum. Checksum covers the data bytes.<br>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.**<br>0000: 0 data bytes<br>0001: 1 data byte<br>0010: 2 data bytes<br>0011: 3 data bytes<br>0100: 4 data bytes<br>0101: 5 data bytes<br>0110: 6 data bytes<br>0111: 7 data bytes<br>1000: 8 data bytes<br>1001-1110: RESERVED<br>1111: Use the ID[1:0] bits (LIN0ID[5:4]) to determine the data length. |

The CAN controller clock must be less than or equal to 25 MHz. If the CIP-51 system clock is above 25 MHz, the divider in the CAN0CFG register must be set to divide the CAN controller clock down to an appropriate speed.

### 21.1.2. CAN Register Access

The CAN controller clock divider selected in the CAN0CFG SFR affects how the CAN registers can be accessed. If the divider is set to 1, then a CAN SFR can immediately be read after it is written. If the divider is set to a value other than 1, then a read of a CAN SFR that has just been written must be delayed by a certain number of cycles. This delay can be performed using a NOP or some other instruction that does not attempt to read the register. This access limitation applies to read and read-modify-write instructions that occur immediately after a write. The full list of affected instructions is ANL, ORL, MOV, XCH, and XRL.

For example, with the CAN0CFG divider set to 1, the CAN0CN SFR can be accessed as follows:

```
MOV CAN0CN, #041          ; Enable access to Bit Timing Register
MOV R7, CAN0CN            ; Copy CAN0CN to R7
```

With the CAN0CFG divider set to /2, the same example code requires an additional NOP:

```
MOV CAN0CN, #041          ; Enable access to Bit Timing Register
NOP                       ; Wait for write to complete
MOV R7, CAN0CN            ; Copy CAN0CN to R7
```

The number of delay cycles required is dependent on the divider setting. With a divider of 2, the read must wait for 1 system clock cycle. With a divider of 4, the read must wait 3 system clock cycles, and with the divider set to 8, the read must wait 7 system clock cycles. The delay only needs to be applied when reading the same register that was written. The application can write and read other CAN SFRs without any delay.

### 21.1.3. Example Timing Calculation for 1 Mbit/Sec Communication

This example shows how to configure the CAN controller timing parameters for a 1 Mbit/Sec bit rate. Table 21.1 shows timing-related system parameters needed for the calculation.

**Table 21.1. Background System Information**

| Parameter | Value | Description |
|---|---|---|
| CIP-51 system clock (SYSCLK) | 24 MHz | Internal Oscillator Max |
| CAN controller clock ($f_{sys}$) | 24 MHz | CAN0CFG divider set to 1 |
| CAN clock period ($t_{sys}$) | 41.667 ns | Derived from $1/f_{sys}$ |
| CAN time quantum ($t_q$) | 41.667 ns | Derived from $t_{sys}$ x BRP[1,2] |
| CAN bus length | 10 m | 5 ns/m signal delay between CAN nodes |
| Propogation delay time[3] | 400 ns | 2 x (transceiver loop delay + bus line delay) |

**Notes:**
1. The CAN time quantum is the smallest unit of time recognized by the CAN controller. Bit timing parameters are specified in integer multiples of the time quantum.
2. The Baud Rate Prescaler (BRP) is defined as the value of the BRP Extension Register plus 1. The BRP extension register has a reset value of 0x0000. The BRP has a reset value of 1.
3. Based on an ISO-11898 compliant transceiver. CAN does not specify a physical layer.

Each bit transmitted on a CAN network has 4 segments (Sync_Seg, Prop_Seg, Phase_Seg1, and Phase_Seg2), as shown in Figure 18.3. The sum of these segments determines the CAN bit time (1/bit rate). In this example, the desired bit rate is 1 Mbit/sec; therefore, the desired bit time is 1000 ns.

### 21.2.4. CAN Register Assignment

The standard Bosch CAN registers are mapped to SFR space as shown below and their full definitions are available in the CAN User's Guide. The name shown in the Name column matches what is provided in the CAN User's Guide. One additional SFR which is not a standard Bosch CAN register, CAN0CFG, is provided to configure the CAN clock. All CAN registers are located on SFR Page 0x0C.

**Table 21.2. Standard CAN Registers and Reset Values**

| CAN Addr. | Name | SFR Name (High) | SFR Addr. | SFR Name (Low) | SFR Addr. | 16-bit SFR | Reset Value |
|---|---|---|---|---|---|---|---|
| 0x00 | CAN Control Register | — | — | CAN0CN | 0xC0 | — | 0x01 |
| 0x02 | Status Register | — | — | CAN0STAT | 0x94 | — | 0x00 |
| 0x04 | Error Counter[1] | CAN0ERRH | 0x97 | CAN0ERRL | 0x96 | CAN0ERR | 0x0000 |
| 0x06 | Bit Timing Register[2] | CAN0BTH | 0x9B | CAN0BTL | 0x9A | CAN0BT | 0x2301 |
| 0x08 | Interrupt Register[1] | CAN0IIDH | 0x9D | CAN0IIDL | 0x9C | CAN0IID | 0x0000 |
| 0x0A | Test Register | — | — | CAN0TST | 0x9E | — | 0x00[3,4] |
| 0x0C | BRP Extension Register[2] | — | — | CAN0BRPE | 0xA1 | — | 0x00 |
| 0x10 | IF1 Command Request | CAN0IF1CRH | 0xBF | CAN0IF1CRL | 0xBE | CAN0IF1CR | 0x0001 |
| 0x12 | IF1 Command Mask | CAN0IF1CMH | 0xC3 | CAN0IF1CML | 0xC2 | CAN0IF1CM | 0x0000 |
| 0x14 | IF1 Mask 1 | CAN0IF1M1H | 0xC5 | CAN0IF1M1L | 0xC4 | CAN0IF1M1 | 0xFFFF |
| 0x16 | IF1 Mask 2 | CAN0IF1M2H | 0xC7 | CAN0IF1M2L | 0xC6 | CAN0IF1M2 | 0xFFFF |
| 0x18 | IF1 Arbitration 1 | CAN0IF1A1H | 0xCB | CAN0IF1A1L | 0xCA | CAN0IF1A1 | 0x0000 |
| 0x1A | IF1 Arbitration 2 | CAN0IF1A2H | 0xCD | CAN0IF1A2L | 0xCC | CAN0IF1A2 | 0x0000 |
| 0x1C | IF1 Message Control | CAN0IF1MCH | 0xD3 | CAN0IF1MCL | 0xD2 | CAN0IF1MC | 0x0000 |
| 0x1E | IF1 Data A 1 | CAN0IF1DA1H | 0xD5 | CAN0IF1DA1L | 0xD4 | CAN0IF1DA1 | 0x0000 |
| 0x20 | IF1 Data A 2 | CAN0IF1DA2H | 0xD7 | CAN0IF1DA2L | 0xD6 | CAN0IF1DA2 | 0x0000 |
| 0x22 | IF1 Data B 1 | CAN0IF1DB1H | 0xDB | CAN0IF1DB1L | 0xDA | CAN0IF1DB1 | 0x0000 |
| 0x24 | IF1 Data B 2 | CAN0IF1DB2H | 0xDD | CAN0IF1DB2L | 0xDC | CAN0IF1DB2 | 0x0000 |
| 0x40 | IF2 Command Request | CAN0IF2CRH | 0xDF | CAN0IF2CRL | 0xDE | CAN0IF2CR | 0x0001 |
| 0x42 | IF2 Command Mask | CAN0IF2CMH | 0xE3 | CAN0IF2CML | 0xE2 | CAN0IF2CM | 0x0000 |
| 0x44 | IF2 Mask 1 | CAN0IF2M1H | 0xEB | CAN0IF2M1L | 0xEA | CAN0IF2M1 | 0xFFFF |
| 0x46 | IF2 Mask 2 | CAN0IF2M2H | 0xED | CAN0IF2M2L | 0xEC | CAN0IF2M2 | 0xFFFF |
| 0x48 | IF2 Arbitration 1 | CAN0IF2A1H | 0xEF | CAN0IF2A1L | 0xEE | CAN0IF2A1 | 0x0000 |
| 0x4A | IF2 Arbitration 2 | CAN0IF2A2H | 0xF3 | CAN0IF2A2L | 0xF2 | CAN0IF2A2 | 0x0000 |
| 0x4C | IF2 Message Control | CAN0IF2MCH | 0xCF | CAN0IF2MCL | 0xCE | CAN0IF2MC | 0x0000 |
| 0x4E | IF2 Data A 1 | CAN0IF2DA1H | 0xF7 | CAN0IF2DA1L | 0xF6 | CAN0IF2DA1 | 0x0000 |

**Notes:**

1. Read-only register.
2. Write-enabled by CCE.
3. The reset value of CAN0TST could also be r0000000b, where r signifies the value of the CAN RX pin.
4. Write-enabled by Test.

### 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.**<br>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.

**Table 22.4. SMBus Status Decoding (Continued)**

| Mode | Values Read: Status Vector | ACKRQ | ARBLOST | ACK | Current SMbus State | Typical Response Options | Values to Write: STA | STO | ACK | Next Status Vector Expected |
|---|---|---|---|---|---|---|---|---|---|---|
| Slave Transmitter | 0100 | 0 | 0 | 0 | A slave byte was transmitted; NACK received. | No action required (expecting STOP condition). | 0 | 0 | X | 0001 |
| | | 0 | 0 | 1 | A slave byte was transmitted; ACK received. | Load SMB0DAT with next data byte to transmit. | 0 | 0 | X | 0100 |
| | | 0 | 1 | X | A Slave byte was transmitted; error detected. | No action required (expecting Master to end transfer). | 0 | 0 | X | 0001 |
| | 0101 | 0 | X | X | An illegal STOP or bus error was detected while a Slave Transmission was in progress. | Clear STO. | 0 | 0 | X | — |
| Slave Receiver | 0010 | 1 | 0 | X | A slave address + R/W was received; ACK requested. | If Write, Acknowledge received address | 0 | 0 | 1 | 0000 |
| | | | | | | If Read, Load SMB0DAT with data byte; ACK received address | 0 | 0 | 1 | 0100 |
| | | | | | | NACK received address. | 0 | 0 | 0 | — |
| | | 1 | 1 | X | Lost arbitration as master; slave address + R/W received; ACK requested. | If Write, Acknowledge received address | 0 | 0 | 1 | 0000 |
| | | | | | | If Read, Load SMB0DAT with data byte; ACK received address | 0 | 0 | 1 | 0100 |
| | | | | | | NACK received address. | 0 | 0 | 0 | — |
| | | | | | | Reschedule failed transfer; NACK received address. | 1 | 0 | 0 | 1110 |
| | 0001 | 0 | 0 | X | A STOP was detected while addressed as a Slave Transmitter or Slave Receiver. | Clear STO. | 0 | 0 | X | — |
| | | 1 | 1 | X | Lost arbitration while attempting a STOP. | No action required (transfer complete/aborted). | 0 | 0 | 0 | — |
| | 0000 | 1 | 0 | X | A slave byte was received; ACK requested. | Acknowledge received byte; Read SMB0DAT. | 0 | 0 | 1 | 0000 |
| | | | | | | NACK received byte. | 0 | 0 | 0 | — |
| Bus Error Condition | 0010 | 0 | 1 | X | Lost arbitration while attempting a repeated START. | Abort failed transfer. | 0 | 0 | X | — |
| | | | | | | Reschedule failed transfer. | 1 | 0 | X | 1110 |
| | 0001 | 0 | 1 | X | Lost arbitration due to a detected STOP. | Abort failed transfer. | 0 | 0 | X | — |
| | | | | | | Reschedule failed transfer. | 1 | 0 | X | 1110 |
| | 0000 | 1 | 1 | X | Lost arbitration while transmitting a data byte as master. | Abort failed transfer. | 0 | 0 | 0 | — |
| | | | | | | Reschedule failed transfer. | 1 | 0 | 0 | 1110 |

## 24.3. SPI0 Slave Mode Operation

When SPI0 is enabled and not configured as a master, it will operate as a SPI slave. As a slave, bytes are shifted in through the MOSI pin and out through the MISO pin by a master device controlling the SCK signal. A bit counter in the SPI0 logic counts SCK edges. When 8 bits have been shifted through the shift register, the SPIF flag is set to logic 1, and the byte is copied into the receive buffer. Data is read from the receive buffer by reading SPI0DAT. A slave device cannot initiate transfers. Data to be transferred to the master device is pre-loaded into the shift register by writing to SPI0DAT. Writes to SPI0DAT are double-buffered, and are placed in the transmit buffer first. If the shift register is empty, the contents of the transmit buffer will immediately be transferred into the shift register. When the shift register already contains data, the SPI will load the shift register with the transmit buffer’s contents after the last SCK edge of the next (or current) SPI transfer.

When configured as a slave, SPI0 can be configured for 4-wire or 3-wire operation. The default, 4-wire slave mode, is active when NSSMD1 (SPI0CN.3) = 0 and NSSMD0 (SPI0CN.2) = 1. In 4-wire mode, the NSS signal is routed to a port pin and configured as a digital input. SPI0 is enabled when NSS is logic 0, and disabled when NSS is logic 1. The bit counter is reset on a falling edge of NSS. Note that the NSS signal must be driven low at least 2 system clocks before the first active edge of SCK for each byte transfer. Figure 24.4 shows a connection diagram between two slave devices in 4-wire slave mode and a master device.

3-wire slave mode is active when NSSMD1 (SPI0CN.3) = 0 and NSSMD0 (SPI0CN.2) = 0. NSS is not used in this mode, and is not mapped to an external port pin through the crossbar. Since there is no way of uniquely addressing the device in 3-wire slave mode, SPI0 must be the only slave device present on the bus. It is important to note that in 3-wire slave mode there is no external means of resetting the bit counter that determines when a full byte has been received. The bit counter can only be reset by disabling and re-enabling SPI0 with the SPIEN bit. Figure 24.3 shows a connection diagram between a slave device in 3-wire slave mode and a master device.

## 24.4. SPI0 Interrupt Sources

When SPI0 interrupts are enabled, the following four flags will generate an interrupt when they are set to logic 1:

All of the following bits must be cleared by software.

1. The SPI Interrupt Flag, SPIF (SPI0CN.7) is set to logic 1 at the end of each byte transfer. This flag can occur in all SPI0 modes.
2. The Write Collision Flag, WCOL (SPI0CN.6) is set to logic 1 if a write to SPI0DAT is attempted when the transmit buffer has not been emptied to the SPI shift register. When this occurs, the write to SPI0DAT will be ignored, and the transmit buffer will not be written.This flag can occur in all SPI0 modes.
3. The Mode Fault Flag MODF (SPI0CN.5) is set to logic 1 when SPI0 is configured as a master, and for multi-master mode and the NSS pin is pulled low. When a Mode Fault occurs, the MSTEN and SPIEN bits in SPI0CN are set to logic 0 to disable SPI0 and allow another master device to access the bus.
4. The Receive Overrun Flag RXOVRN (SPI0CN.4) is set to logic 1 when configured as a slave, and a transfer is completed and the receive buffer still holds an unread byte from a previous transfer. The new byte is not transferred to the receive buffer, allowing the previously received data byte to be read. The data byte which caused the overrun is lost.

## 24.5. Serial Clock Phase and Polarity

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

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

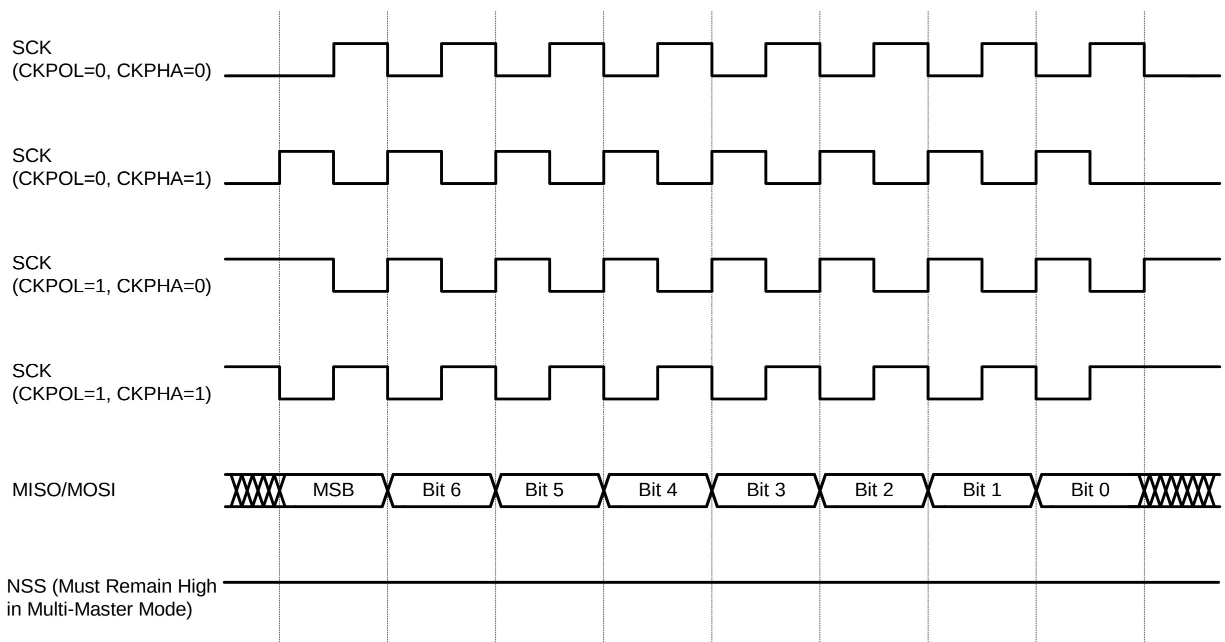


**Figure 24.5. Master Mode Data/Clock Timing**

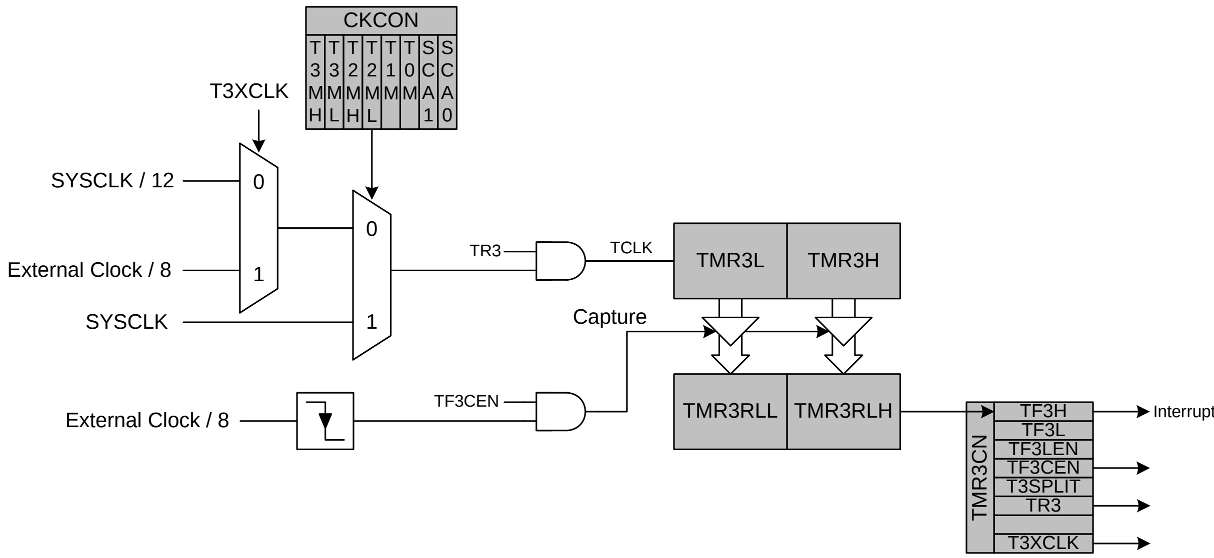


**Figure 25.9. Timer 3 External Oscillator Capture Mode Block Diagram**