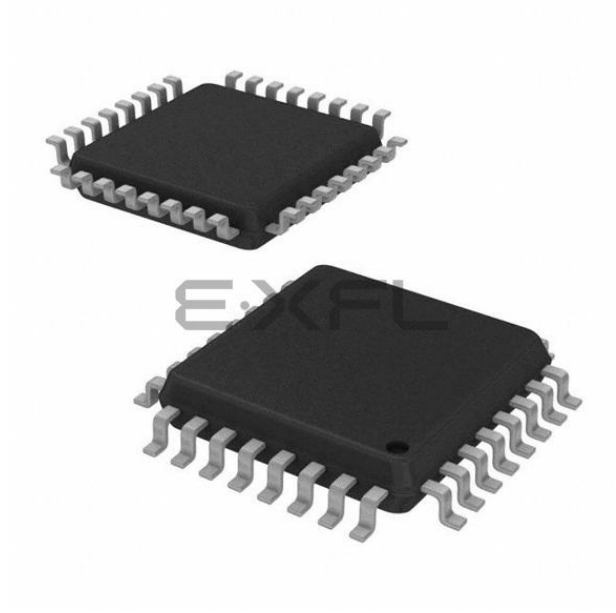

Welcome to E-XFL.COM

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	Discontinued at Digi-Key
Core Processor	8051
Core Size	8-Bit
Speed	48 MIPS
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, 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)	2.7V ~ 5.25V
Data Converters	A/D 21x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	32-LQFP
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/silicon-labs/c8051f38b-gq

Table 3.1. Pin Definitions for the C8051F388/9/A/B (Continued)

Name	Pin Numbers		Type	Description
	48-pin	32-pin		
P3.2	28	—	D I/O or A In	Port 3.2.
P3.3	27	—	D I/O or A In	Port 3.3.
P3.4	26	—	D I/O or A In	Port 3.4.
P3.5	25	—	D I/O or A In	Port 3.5.
P3.6	24	—	D I/O or A In	Port 3.6.
P3.7	23	—	D I/O or A In	Port 3.7.
P4.0	22	—	D I/O or A In	Port 4.0. See Section 20 for a complete description of Port 4.
P4.1	21	—	D I/O or A In	Port 4.1.
P4.2	20	—	D I/O or A In	Port 4.2.
P4.3	19	—	D I/O or A In	Port 4.3.
P4.4	18	—	D I/O or A In	Port 4.4.
P4.5	17	—	D I/O or A In	Port 4.5.
P4.6	16	—	D I/O or A In	Port 4.6.
P4.7	15	—	D I/O or A In	Port 4.7.

8.1. Comparator Multiplexers

C8051F388/9/A/B devices include an analog input multiplexer to connect Port I/O pins to the comparator inputs. The Comparator inputs are selected in the CPTnMX registers (SFR Definition 8.5 and SFR Definition 8.6). The CMXnP2–CMXnP0 bits select the Comparator positive input; the CMXnN2–CMXnN0 bits select the Comparator negative input.

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 “20.3. General Purpose Port I/O” on page 155).

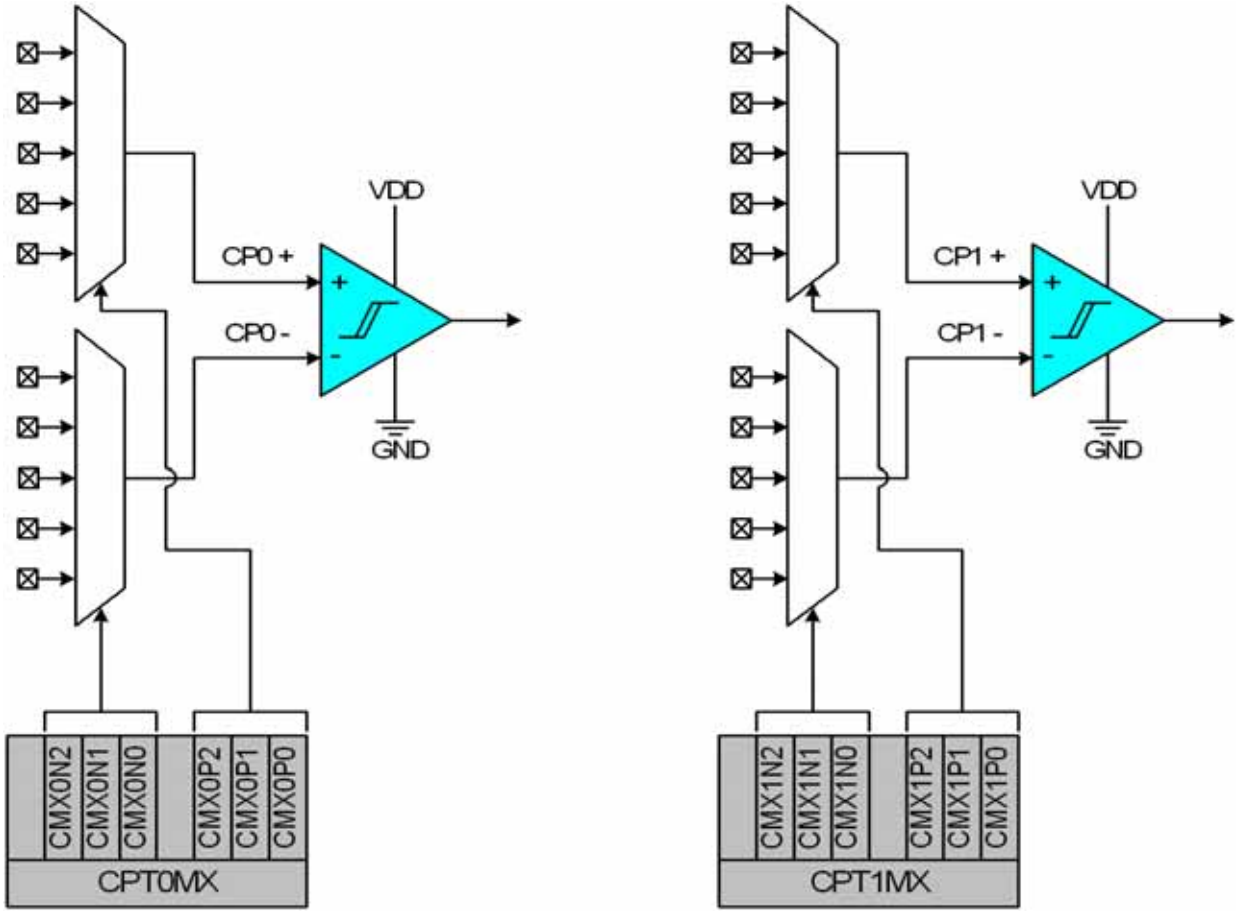

Figure 8.4. Comparator Input Multiplexer Block Diagram

SFR Definition 11.6. PSW: Program Status Word

Bit	7	6	5	4	3	2	1	0
Name	CY	AC	F0	RS[1:0]		OV	F1	PARITY
Type	R/W	R/W	R/W	R/W		R/W	R/W	R
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xD0; SFR Page = All Pages; Bit-Addressable

Bit	Name	Function
7	CY	**Carry Flag.** This bit is set when the last arithmetic operation resulted in a carry (addition) or a borrow (subtraction). It is cleared to logic 0 by all other arithmetic operations.
6	AC	**Auxiliary Carry Flag.** This bit is set when the last arithmetic operation resulted in a carry into (addition) or a borrow from (subtraction) the high order nibble. It is cleared to logic 0 by all other arithmetic operations.
5	F0	**User Flag 0.** This is a bit-addressable, general purpose flag for use under software control.
4:3	RS[1:0]	**Register Bank Select.** These bits select which register bank is used during register accesses. 00: Bank 0, Addresses 0x00-0x07 01: Bank 1, Addresses 0x08-0x0F 10: Bank 2, Addresses 0x10-0x17 11: Bank 3, Addresses 0x18-0x1F
2	OV	**Overflow Flag.** This bit is set to 1 under the following circumstances: • An ADD, ADDC, or SUBB instruction causes a sign-change overflow. • A MUL instruction results in an overflow (result is greater than 255). • A DIV instruction causes a divide-by-zero condition. The OV bit is cleared to 0 by the ADD, ADDC, SUBB, MUL, and DIV instructions in all other cases.
1	F1	**User Flag 1.** This is a bit-addressable, general purpose flag for use under software control.
0	PARITY	**Parity Flag.** This bit is set to logic 1 if the sum of the eight bits in the accumulator is odd and cleared if the sum is even.

14.2. Configuring the External Memory Interface

Configuring the External Memory Interface consists of five steps:

1. Configure the Output Modes of the associated port pins as either push-pull or open-drain (push-pull is most common), and skip the associated pins in the crossbar.
2. Configure Port latches to “park” the EMIF pins in a dormant state (usually by setting them to logic 1).
3. Select Multiplexed mode or Non-multiplexed mode.
4. Select the memory mode (on-chip only, split mode without bank select, split mode with bank select, or off-chip only).
5. Set up timing to interface with off-chip memory or peripherals.

Each of these five steps is explained in detail in the following sections. The Port selection, Multiplexed mode selection, and Mode bits are located in the EMI0CF register shown in SFR Definition 14.4.

14.3. Port Configuration

The External Memory Interface appears on Ports 4, 3, 2, and 1 when it is used for off-chip memory access. When the EMIF is used, the Crossbar should be configured to skip over the control lines P1.7 ($\overline{WR}$), P1.6 ($\overline{RD}$), and if multiplexed mode is selected P1.3 (ALE) using the P1SKIP register. For more information about configuring the Crossbar, see Section “Figure 20.1. Port I/O Functional Block Diagram (Port 0 through Port 3)” on page 147.

The External Memory Interface claims the associated Port pins for memory operations ONLY during the execution of an off-chip MOVX instruction. Once the MOVX instruction has completed, control of the Port pins reverts to the Port latches or to the Crossbar settings for those pins. See Section “20. Port Input/Output” on page 147 for more information about the Crossbar and Port operation and configuration. **The Port latches should be explicitly configured to ‘park’ the External Memory Interface pins in a dormant state, most commonly by setting them to a logic 1**.

During the execution of the MOVX instruction, the External Memory Interface will explicitly disable the drivers on all Port pins that are acting as Inputs (Data[7:0] during a READ operation, for example). The Output mode of the Port pins (whether the pin is configured as Open-Drain or Push-Pull) is unaffected by the External Memory Interface operation, and remains controlled by the PnMDOUT registers. In most cases, the output modes of all EMIF pins should be configured for push-pull mode.

14.4.2. Non-multiplexed Configuration

In Non-multiplexed mode, the Data Bus and the Address Bus pins are not shared. An example of a Non-multiplexed Configuration is shown in Figure 14.2. See Section “14.6.1. Non-multiplexed Mode” on page 98 for more information about Non-multiplexed operation.

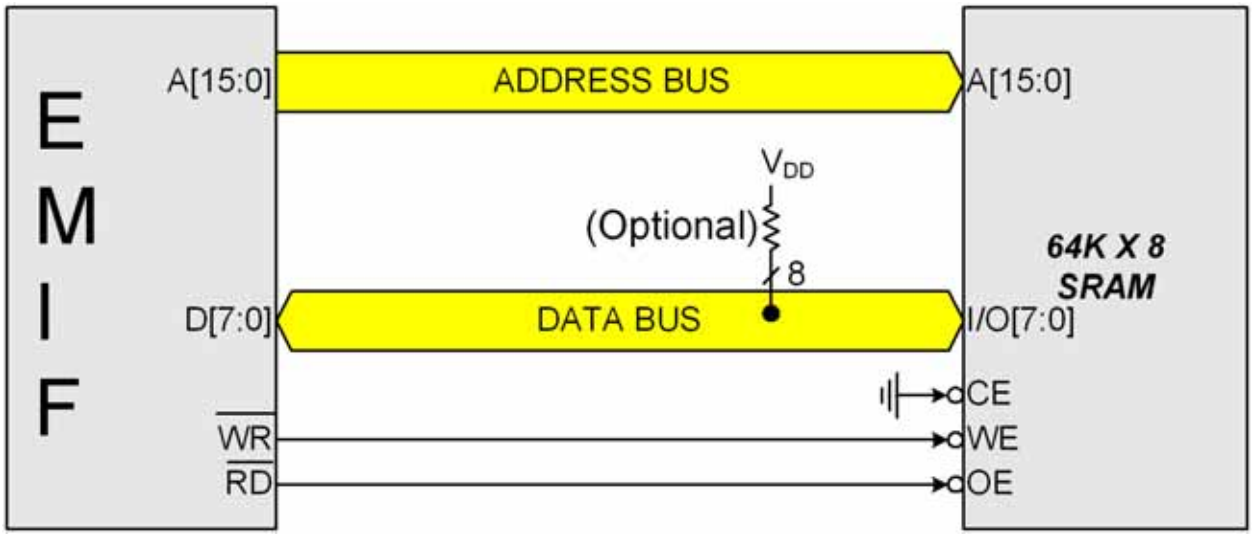

Figure 14.2. Non-multiplexed Configuration Example

14.6.1.2. 8-bit MOVX without Bank Select: EMI0CF[4:2] = 101 or 111

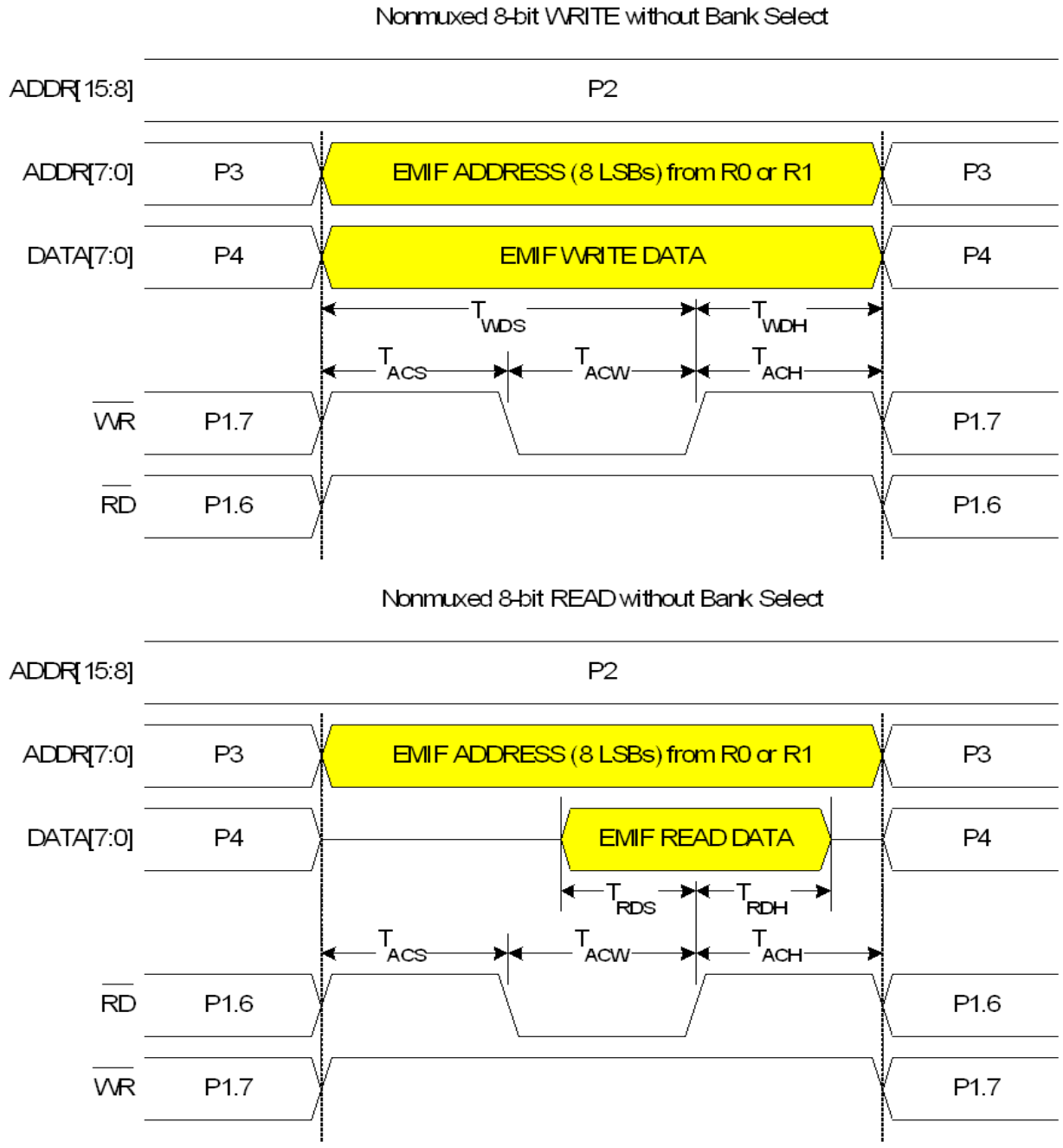

Figure 14.5. Non-Multiplexed 8-bit MOVX without Bank Select Timing

14.6.2. Multiplexed Mode

14.6.2.1. 16-bit MOVX: EMI0CF[4:2] = 001, 010, or 011

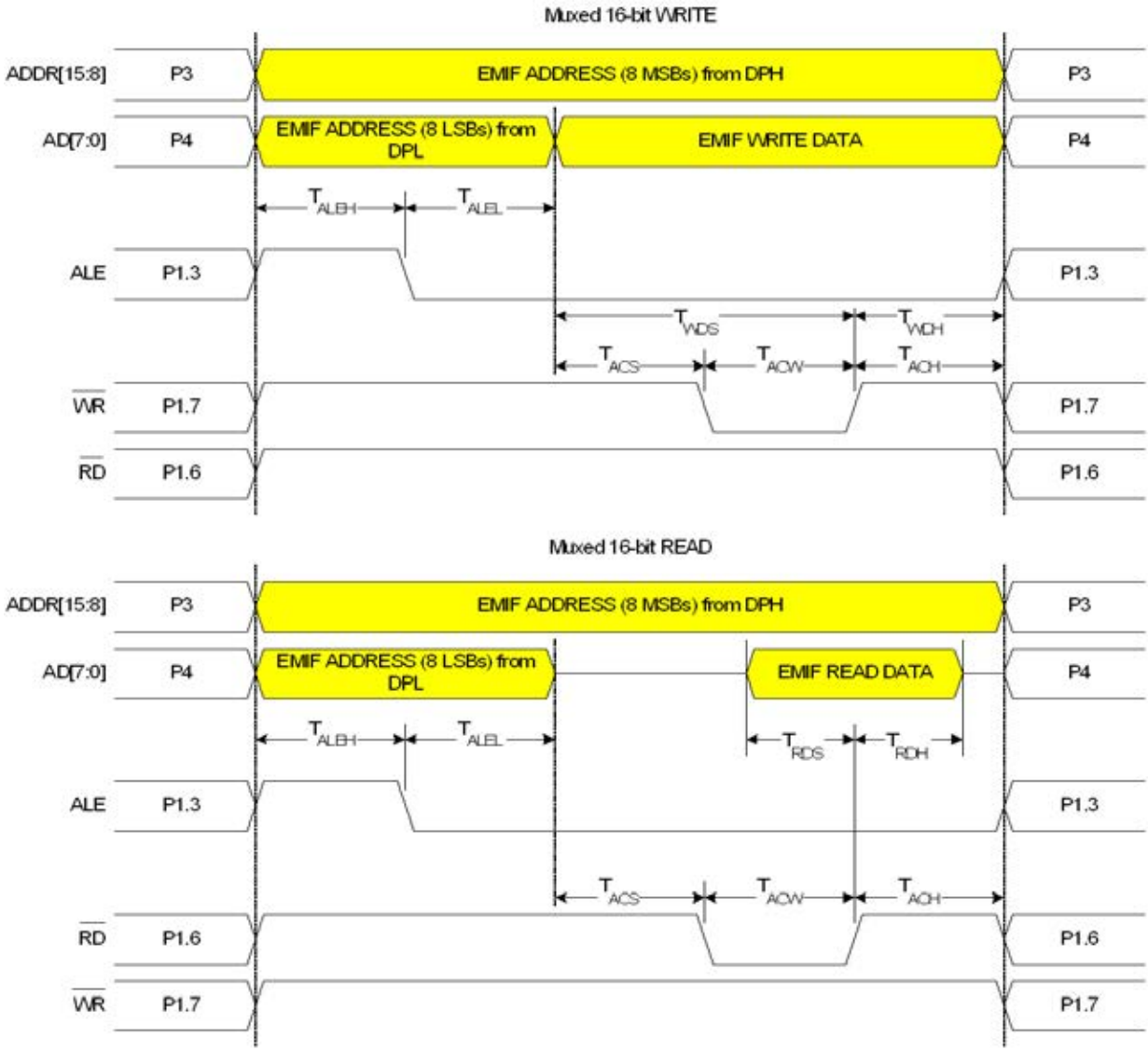

Figure 14.7. Multiplexed 16-bit MOVX Timing

Table 15.2. Special Function Registers (Continued)

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

Register	Address	Page	Description	Page
IE	0xA8	All Pages	Interrupt Enable	115
IP	0xB8	All Pages	Interrupt Priority	116
IT01CF	0xE4	0	INT0/INT1 Configuration	122
OSCICL	0xB3	All Pages	Internal Oscillator Calibration	139
OSCICN	0xB2	All Pages	Internal Oscillator Control	140
OSCLCN	0x86	All Pages	Internal Low-Frequency Oscillator Control	142
OSCXCN	0xB1	All Pages	External Oscillator Control	146
P0	0x80	All Pages	Port 0 Latch	156
P0MDIN	0xF1	All Pages	Port 0 Input Mode Configuration	156
P0MDOUT	0xA4	All Pages	Port 0 Output Mode Configuration	157
P0SKIP	0xD4	All Pages	Port 0 Skip	157
P1	0x90	All Pages	Port 1 Latch	158
P1MDIN	0xF2	All Pages	Port 1 Input Mode Configuration	158
P1MDOUT	0xA5	All Pages	Port 1 Output Mode Configuration	159
P1SKIP	0xD5	All Pages	Port 1 Skip	159
P2	0xA0	All Pages	Port 2 Latch	160
P2MDIN	0xF3	All Pages	Port 2 Input Mode Configuration	160
P2MDOUT	0xA6	All Pages	Port 2 Output Mode Configuration	161
P2SKIP	0xD6	All Pages	Port 2 Skip	161
P3	0xB0	All Pages	Port 3 Latch	162
P3MDIN	0xF4	All Pages	Port 3 Input Mode Configuration	162
P3MDOUT	0xA7	All Pages	Port 3 Output Mode Configuration	163
P3SKIP	0xDF	All Pages	Port 3Skip	163
P4	0xC7	All Pages	Port 4 Latch	164
P4MDIN	0xF5	All Pages	Port 4 Input Mode Configuration	164
P4MDOUT	0xAE	All Pages	Port 4 Output Mode Configuration	165
PCA0CN	0xD8	All Pages	PCA Control	272
PCA0CPH0	0xFC	All Pages	PCA Capture 0 High	276
PCA0CPH1	0xEA	All Pages	PCA Capture 1 High	276
PCA0CPH2	0xEC	All Pages	PCA Capture 2 High	276
PCA0CPH3	0xEE	All Pages	PCA Capture 3High	276
PCA0CPH4	0xFE	All Pages	PCA Capture 4 High	276
PCA0CPL0	0xFB	All Pages	PCA Capture 0 Low	276
PCA0CPL1	0xE9	All Pages	PCA Capture 1 Low	276

16.1. MCU Interrupt Sources and Vectors

The C8051F388/9/A/B MCUs support several 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 16.1. 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).

16.1.1. 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 (IP, 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 16.1.

16.1.2. 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 6 system clock cycles: 1 clock cycle to detect the interrupt and 5 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 20 system clock cycles: 1 clock cycle to detect the interrupt, 6 clock cycles to execute the RETI, 8 clock cycles to complete the DIV instruction and 5 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.

Note that the CPU is stalled during Flash write operations. Interrupt service latency will be increased for interrupts occurring while the CPU is stalled. The latency for these situations will be determined by the standard interrupt service procedure (as described above) and the amount of time the CPU is stalled.

16.2. Interrupt Register Descriptions

The SFRs used to enable the interrupt sources and set their priority level are described in this section. Refer to the data sheet 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).

16.3. $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$ External Interrupt Sources

The $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$ external interrupt sources are configurable as active high or low, edge or level sensitive. The IN0PL ($\overline{\text{INT0}}$ Polarity) and IN1PL ($\overline{\text{INT1}}$ Polarity) bits in the IT01CF register select active high or active low; the IT0 and IT1 bits in TCON (Section “25.1. Timer 0 and Timer 1” on page 227) select level or edge sensitive. The table below lists the possible configurations.

IT0	IN0PL	$\overline{\text{INT0}}$ Interrupt
1	0	Active low, edge sensitive
1	1	Active high, edge sensitive
0	0	Active low, level sensitive
0	1	Active high, level sensitive

IT1	IN1PL	$\overline{\text{INT1}}$ Interrupt
1	0	Active low, edge sensitive
1	1	Active high, edge sensitive
0	0	Active low, level sensitive
0	1	Active high, level sensitive

$\overline{\text{INT0}}$ and $\overline{\text{INT1}}$ are assigned to Port pins as defined in the IT01CF register (see SFR Definition 16.7). Note that $\overline{\text{INT0}}$ and $\overline{\text{INT0}}$ Port pin assignments are independent of any Crossbar assignments. $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$ will monitor their assigned Port pins without disturbing the peripheral that was assigned the Port pin via the Crossbar. To assign a Port pin only to $\overline{\text{INT0}}$ and/or $\overline{\text{INT1}}$, configure the Crossbar to skip the selected pin(s). This is accomplished by setting the associated bit in register PnSKIP (see Section “20.1. Priority Crossbar Decoder” on page 148 for complete details on configuring the Crossbar).

IE0 (TCON.1) and IE1 (TCON.3) serve as the interrupt-pending flags for the $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$ external interrupts, respectively. If an $\overline{\text{INT0}}$ or $\overline{\text{INT1}}$ external interrupt is configured as edge-sensitive, the corresponding interrupt-pending flag is automatically cleared by the hardware when the CPU vectors to the ISR. When configured as level sensitive, the interrupt-pending flag remains logic 1 while the input is active as defined by the corresponding polarity bit (IN0PL or IN1PL); the flag remains logic 0 while the input is inactive. The external interrupt source must hold the input active until the interrupt request is recognized. It must then deactivate the interrupt request before execution of the ISR completes or another interrupt request will be generated.

18.1.3. Flash Write Procedure

Bytes in Flash memory can be written one byte at a time, or in groups of two. The FLBWE bit in register PFE0CN (SFR Definition) controls whether a single byte or a block of two bytes is written to Flash during a write operation. When FLBWE is cleared to 0, the Flash will be written one byte at a time. When FLBWE is set to 1, the Flash will be written in two-byte blocks. Block writes are performed in the same amount of time as single-byte writes, which can save time when storing large amounts of data to Flash memory.During a single-byte write to Flash, bytes are written individually, and a Flash write will be performed after each MOVX write instruction. The recommended procedure for writing Flash in single bytes is:

1. Disable interrupts.
2. Clear the FLBWE bit (register PFE0CN) to select single-byte write mode.
3. Set the PSWE bit (register PSCTL).
4. Clear the PSEE bit (register PSCTL).
5. Write the first key code to FLKEY: 0xA5.
6. Write the second key code to FLKEY: 0xF1.
7. Using the MOVX instruction, write a single data byte to the desired location within the 512-byte sector.
8. Clear the PSWE bit.
9. Re-enable interrupts.

Steps 5-7 must be repeated for each byte to be written.

For block Flash writes, the Flash write procedure is only performed after the last byte of each block is written with the MOVX write instruction. A Flash write block is two bytes long, from even addresses to odd addresses. Writes must be performed sequentially (i.e. addresses ending in 0b and 1b must be written in order). The Flash write will be performed following the MOVX write that targets the address ending in 1b. If a byte in the block does not need to be updated in Flash, it should be written to 0xFF. The recommended procedure for writing Flash in blocks is:

1. Disable interrupts.
2. Set the FLBWE bit (register PFE0CN) to select block write mode.
3. Set the PSWE bit (register PSCTL).
4. Clear the PSEE bit (register PSCTL).
5. Write the first key code to FLKEY: 0xA5.
6. Write the second key code to FLKEY: 0xF1.
7. Using the MOVX instruction, write the first data byte to the even block location (ending in 0b).
8. Write the first key code to FLKEY: 0xA5.
9. Write the second key code to FLKEY: 0xF1.
10. Using the MOVX instruction, write the second data byte to the odd block location (ending in 1b).
11. Clear the PSWE bit.
12. Re-enable interrupts.

Steps 5–10 must be repeated for each block to be written.

SFR Definition 20.4. P0: Port 0

Bit	7	6	5	4	3	2	1	0
Name	P0[7:0]							
Type	R/W							
Reset	1	1	1	1	1	1	1	1

SFR Address = 0x80; SFR Page = All Pages; Bit Addressable

Bit	Name	Description	Write	Read
7:0	P0[7:0]	**Port 0 Data.** Sets the Port latch logic value or reads the Port pin logic state in Port cells configured for digital I/O.	0: Set output latch to logic LOW. 1: Set output latch to logic HIGH.	0: P0.n Port pin is logic LOW. 1: P0.n Port pin is logic HIGH.

SFR Definition 20.5. P0MDIN: Port 0 Input Mode

Bit	7	6	5	4	3	2	1	0
Name	P0MDIN[7:0]							
Type	R/W							
Reset	1	1	1	1	1	1	1	1

SFR Address = 0xF1; SFR Page = All Pages

Bit	Name	Function
7:0	P0MDIN[7:0]	**Analog Configuration Bits for P0.7–P0.0 (respectively).** Port pins configured for analog mode have their weak pullup, digital driver, and digital receiver disabled. 0: Corresponding P0.n pin is configured for analog mode. 1: Corresponding P0.n pin is not configured for analog mode.

the incoming slave address. Additionally, if the GCn bit in register SMBnADR is set to 1, hardware will recognize the General Call Address (0x00). Table 21.4 shows some example parameter settings and the slave addresses that will be recognized by hardware under those conditions.

Table 21.4. Hardware Address Recognition Examples (EHACK = 1)

Hardware Slave Address SLVn[6:0]	Slave Address Mask SLVMn[6:0]	GCn bit	Slave Addresses Recognized by Hardware
0x34	0x7F	0	0x34
0x34	0x7F	1	0x34, 0x00 (General Call)
0x34	0x7E	0	0x34, 0x35
0x34	0x7E	1	0x34, 0x35, 0x00 (General Call)
0x70	0x73	0	0x70, 0x74, 0x78, 0x7C

SFR Definition 21.6. SMB0ADR: SMBus0 Slave Address

Bit	7	6	5	4	3	2	1	0
Name	SLV0[6:0]							GC0
Type	R/W							R/W
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xCF; SFR Page = 0

Bit	Name	Function
7:1	SLV0[6:0]	**SMBus Hardware Slave Address.** Defines the SMBus0 Slave Address(es) for automatic hardware acknowledgment. Only address bits which have a 1 in the corresponding bit position in SLVM0[6:0] are checked against the incoming address. This allows multiple addresses to be recognized.
0	GC0	**General Call Address Enable.** When hardware address recognition is enabled (EHACK0 = 1), this bit will determine whether the General Call Address (0x00) is also recognized by hardware. 0: General Call Address is ignored. 1: General Call Address is recognized.

21.5.4. Read Sequence (Slave)

During a read sequence, an SMBus master reads data from a slave device. The slave in this transfer will be a receiver during the address byte, and a transmitter during all data bytes. When slave events are enabled (INH = 0), the interface enters Slave Receiver Mode (to receive the slave address) when a START followed by a slave address and direction bit (READ in this case) is received. If hardware ACK generation is disabled, upon entering Slave Receiver Mode, an interrupt is generated and the ACKRQ bit is set. The software must respond to the received slave address with an ACK, or ignore the received slave address with a NACK. If hardware ACK generation is enabled, the hardware will apply the ACK for a slave address which matches the criteria set up by SMB0ADR and SMB0ADM. The interrupt will occur after the ACK cycle.

If the received slave address is ignored (by software or hardware), slave interrupts will be inhibited until the next START is detected. If the received slave address is acknowledged, zero or more data bytes are transmitted. If the received slave address is acknowledged, data should be written to SMB0DAT to be transmitted. The interface enters slave transmitter mode, and transmits one or more bytes of data. After each byte is transmitted, the master sends an acknowledge bit; if the acknowledge bit is an ACK, SMB0DAT should be written with the next data byte. If the acknowledge bit is a NACK, SMB0DAT should not be written to before SI is cleared (an error condition may be generated if SMB0DAT is written following a received NACK while in slave transmitter mode). The interface exits slave transmitter mode after receiving a STOP. The interface will switch to slave receiver mode if SMB0DAT is not written following a Slave Transmitter interrupt. Figure 21.8 shows a typical slave read sequence. Two transmitted data bytes are shown, though any number of bytes may be transmitted. Notice that all of the “data byte transferred” interrupts occur **after** the ACK cycle in this mode, regardless of whether hardware ACK generation is enabled.

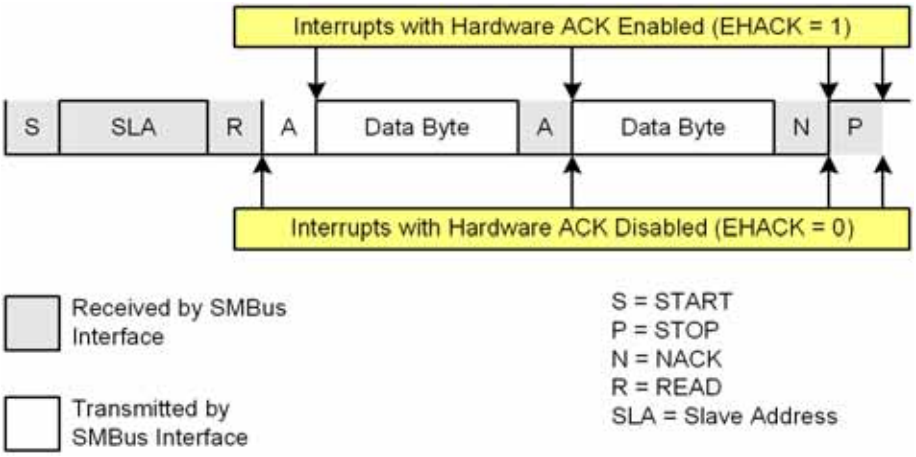

Figure 21.8. Typical Slave Read Sequence

21.6. SMBus Status Decoding

The current SMBus status can be easily decoded using the SMB0CN register. The appropriate actions to take in response to an SMBus event depend on whether hardware slave address recognition and ACK generation is enabled or disabled. Table 21.5 describes the typical actions when hardware slave address recognition and ACK generation is disabled. Table 21.6 describes the typical actions when hardware slave address recognition and ACK generation is enabled. In the tables, STATUS VECTOR refers to the four upper bits of SMB0CN: MASTER, TXMODE, STA, and STO. The shown response options are only the typical responses; application-specific procedures are allowed as long as they conform to the SMBus specification. Highlighted responses are allowed by hardware but do not conform to the SMBus specification.

1. Clear RI1 to 0
2. Read SBUF1
3. Check RI1, and repeat at Step 1 if RI1 is set to 1.

If the extra bit function is enabled (XBE1 = 1) and the parity function is disabled (PE1 = 0), the extra bit for the oldest byte in the FIFO can be read from the RBX1 bit (SCON1.2). If the extra bit function is not enabled, the value of the stop bit for the oldest FIFO byte will be presented in RBX1. When the parity function is enabled (PE1 = 1), hardware will check the received parity bit against the selected parity type (selected with S1PT[1:0]) when receiving data. If a byte with parity error is received, the PERR1 flag will be set to 1. This flag must be cleared by software. Note: when parity is enabled, the extra bit function is not available.

23.3.3. Multiprocessor Communications

UART1 supports multiprocessor communication between a master processor and one or more slave processors by special use of the extra data bit. When a master processor wants to transmit to one or more slaves, it first sends an address byte to select the target(s). An address byte differs from a data byte in that its extra bit is logic 1; in a data byte, the extra bit is always set to logic 0.

Setting the MCE1 bit (SMOD1.7) of a slave processor configures its UART such that when a stop bit is received, the UART will generate an interrupt only if the extra bit is logic 1 (RBX1 = 1) signifying an address byte has been received. In the UART interrupt handler, software will compare the received address with the slave's own assigned address. If the addresses match, the slave will clear its MCE1 bit to enable interrupts on the reception of the following data byte(s). Slaves that weren't addressed leave their MCE1 bits set and do not generate interrupts on the reception of the following data bytes, thereby ignoring the data. Once the entire message is received, the addressed slave resets its MCE1 bit to ignore all transmissions until it receives the next address byte.

Multiple addresses can be assigned to a single slave and/or a single address can be assigned to multiple slaves, thereby enabling "broadcast" transmissions to more than one slave simultaneously. The master processor can be configured to receive all transmissions or a protocol can be implemented such that the master/slave role is temporarily reversed to enable half-duplex transmission between the original master and slave(s).

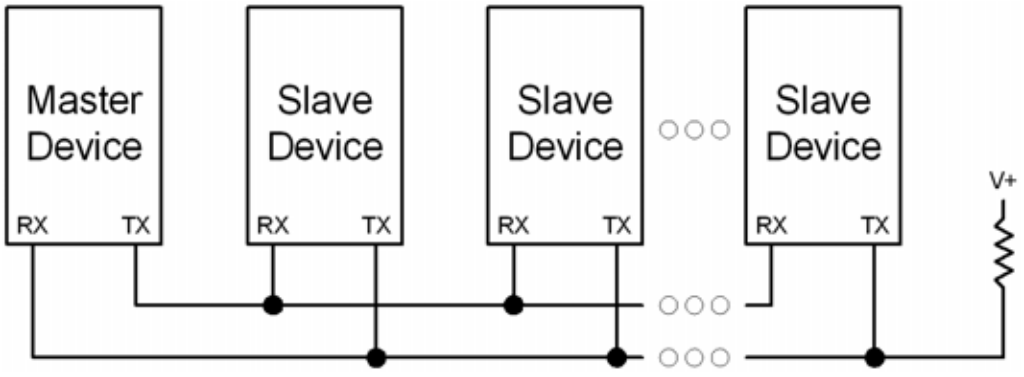

Figure 23.6. UART Multi-Processor Mode Interconnect Diagram

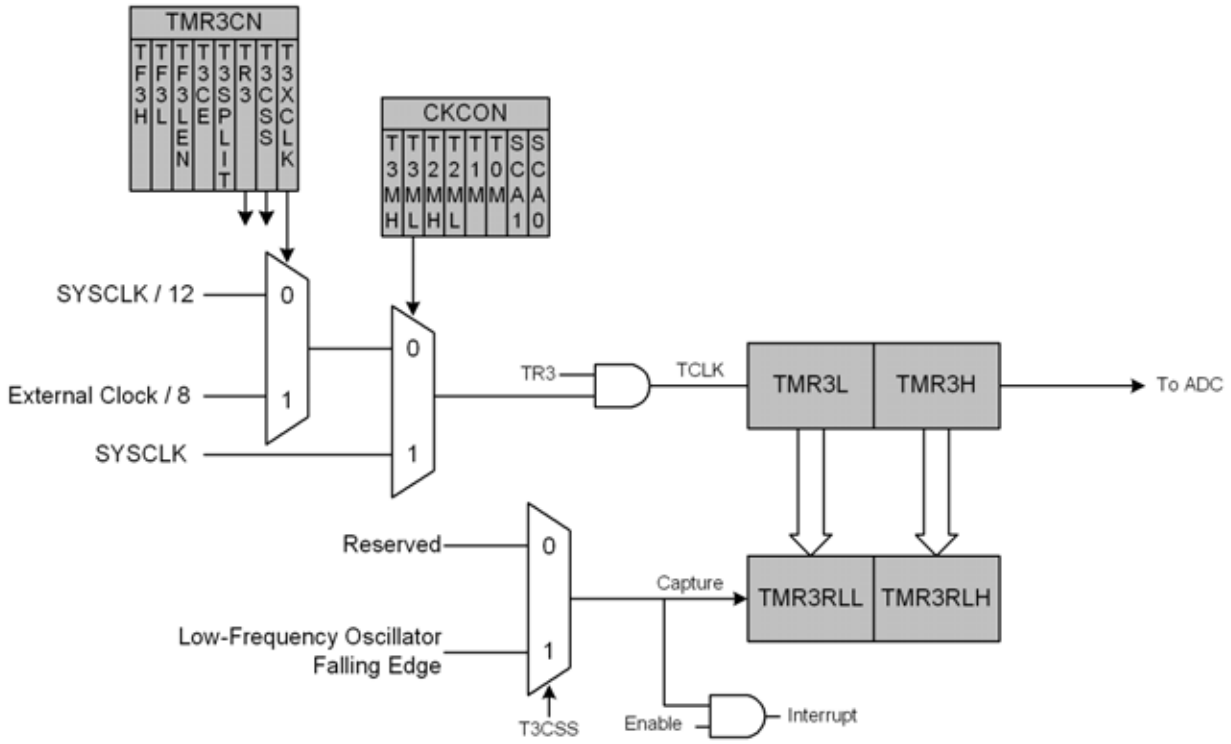

Figure 25.10. Timer 3 Capture Mode (T3SPLIT = 0)

When T3SPLIT = 1, the Timer 3 registers (TMR3H and TMR3L) act as two 8-bit counters. Each counter counts up independently and overflows from 0xFF to 0x00. Each time a capture event is received, the contents of the Timer 3 registers are latched into the Timer 3 Reload registers (TMR3RLH and TMR3RLL). A Timer 3 interrupt is generated if enabled.

SFR Definition 25.19. TMR4CN: Timer 4 Control

Bit	7	6	5	4	3	2	1	0
Name	TF4H	TF4L	TF4LEN		T4SPLIT	TR4		T4XCLK
Type	R/W	R/W	R/W	R	R/W	R/W	R	R/W
Reset	0	0	0	0	0	0	0	0

SFR Address = 0x91; SFR Page = F

Bit	Name	Function
7	TF4H	**Timer 4 High Byte Overflow Flag.** Set by hardware when the Timer 4 high byte overflows from 0xFF to 0x00. In 16 bit mode, this will occur when Timer 4 overflows from 0xFFFF to 0x0000. When the Timer 4 interrupt is enabled, setting this bit causes the CPU to vector to the Timer 4 interrupt service routine. This bit is not automatically cleared by hardware.
6	TF4L	**Timer 4 Low Byte Overflow Flag.** Set by hardware when the Timer 4 low byte overflows from 0xFF to 0x00. TF4L will be set when the low byte overflows regardless of the Timer 4 mode. This bit is not automatically cleared by hardware.
5	TF4LEN	**Timer 4 Low Byte Interrupt Enable.** When set to 1, this bit enables Timer 4 Low Byte interrupts. If Timer 4 interrupts are also enabled, an interrupt will be generated when the low byte of Timer 4 overflows.
4	Unused	Read = 0b; Write = don't care.
3	T4SPLIT	**Timer 4 Split Mode Enable.** When this bit is set, Timer 4 operates as two 8-bit timers with auto-reload. 0: Timer 4 operates in 16-bit auto-reload mode. 1: Timer 4 operates as two 8-bit auto-reload timers.
2	TR4	**Timer 4 Run Control.** Timer 4 is enabled by setting this bit to 1. In 8-bit mode, this bit enables/disables TMR4H only; TMR4L is always enabled in split mode.
1	Unused	Read = 0b; Write = don't care.
0	T4XCLK	**Timer 4 External Clock Select.** This bit selects the external clock source for Timer 4. However, the Timer 4 Clock Select bits (T4MH and T4ML in register CKCON1) may still be used to select between the external clock and the system clock for either timer. 0: Timer 4 clock is the system clock divided by 12. 1: Timer 4 clock is the external clock divided by 8 (synchronized with SYSCLK).

SFR Definition 25.28. TMR5H Timer 5 High Byte

Bit	7	6	5	4	3	2	1	0
Name	TMR5H[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xCD; SFR Page = F

Bit	Name	Function
7:0	TMR5H[7:0]	**Timer 5 High Byte.** In 16-bit mode, the TMR5H register contains the high byte of the 16-bit Timer 5. In 8-bit mode, TMR5H contains the 8-bit high byte timer value.