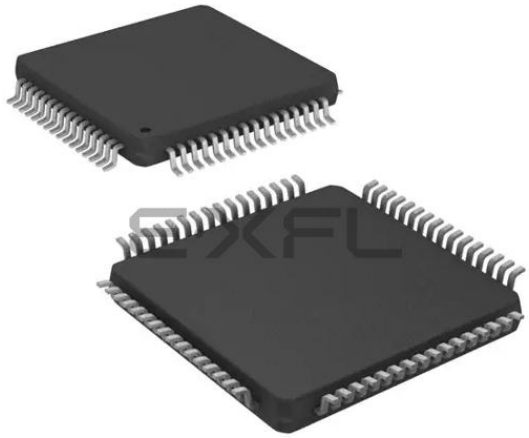


Welcome to E-XFL.COM

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	Not For New Designs
Core Processor	8051
Core Size	8-Bit
Speed	25MHz
Connectivity	SMBus (2-Wire/I²C), SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	24
Program Memory Size	64KB (64K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	4.25K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 3.6V
Data Converters	A/D 2x16b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-TQFP
Supplier Device Package	64-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/silicon-labs/c8051f065-gqr

1.1.3. Additional Features

The C8051F06x MCU family includes several key enhancements to the CIP-51 core and peripherals to improve overall performance and ease of use in end applications.

The extended interrupt handler provides 22 interrupt sources into the CIP-51, allowing the numerous analog and digital peripherals to interrupt the controller. An interrupt driven system requires less intervention by the MCU, giving it more effective throughput. The extra interrupt sources are very useful when building multi-tasking, real-time systems.

There are up to seven reset sources for the MCU: an on-board VDD monitor, a Watchdog Timer, a missing clock detector, a voltage level detection from Comparator0, a forced software reset, the CNVSTR2 input pin, and the /RST pin. The /RST pin is bi-directional, accommodating an external reset, or allowing the internally generated POR to be output on the /RST pin. Each reset source except for the VDD monitor and Reset Input pin may be disabled by the user in software; the VDD monitor is enabled/disabled via the MONEN pin. The Watchdog Timer may be permanently enabled in software after a power-on reset during MCU initialization.

The MCU has an internal, stand alone clock generator which is used by default as the system clock after any reset. If desired, the clock source may be switched on the fly to the external oscillator, which can use a crystal, ceramic resonator, capacitor, RC, or external clock source to generate the system clock. This can be extremely useful in low power applications, allowing the MCU to run from a slow (power saving) external crystal source, while periodically switching to the fast (up to 25 MHz) internal oscillator as needed.

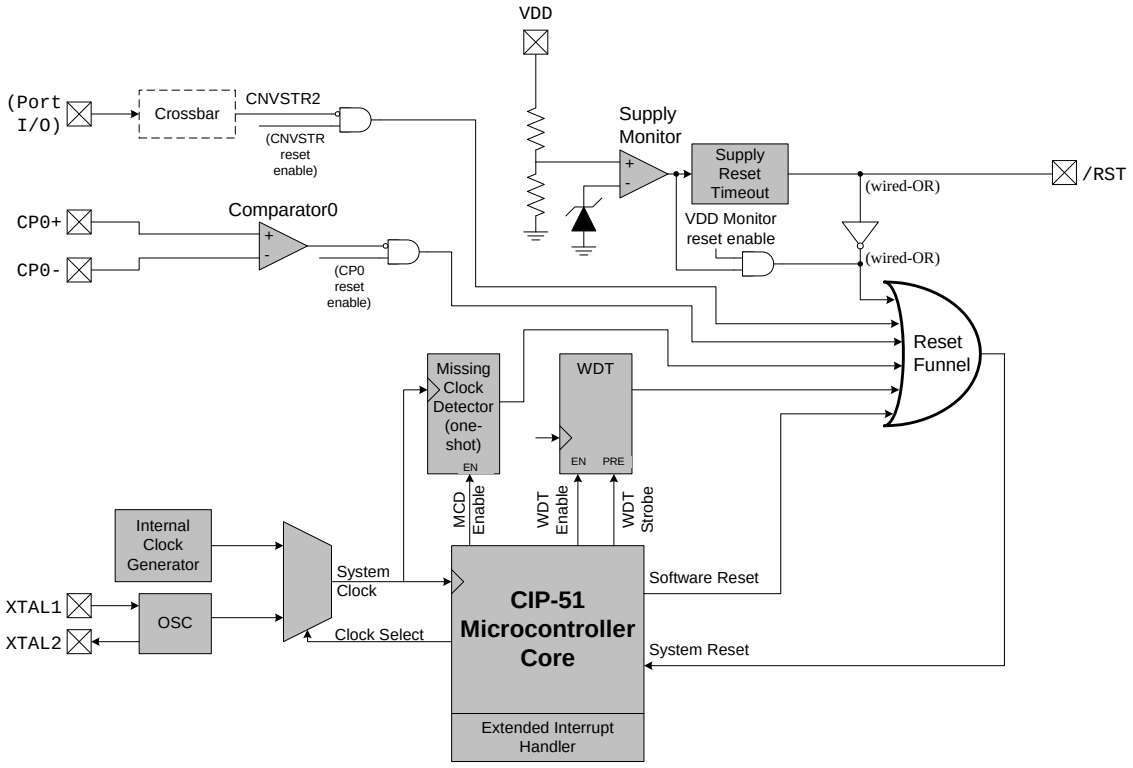


Figure 1.6. On-Board Clock and Reset

1.9. 10-Bit Analog to Digital Converter

The C8051F060/1/2/3 devices have an on-board 10-bit SAR ADC (ADC2) with a 9-channel input multiplexer and programmable gain amplifier. This ADC features a 200 ksps maximum throughput and true 10-bit performance with an INL of ±1LSB. Eight input pins are available for measurement and can be programmed as single-ended or differential inputs. Additionally, the on-chip temperature sensor can be used as an input to the ADC. The ADC is under full control of the CIP-51 microcontroller via the Special Function Registers. The ADC2 voltage reference is selected between the analog power supply (AV+) and the external VREF2 pin. User software may put ADC2 into shutdown mode to save power.

A flexible conversion scheduling system allows ADC2 conversions to be initiated by software commands, timer overflows, or an external input signal. Conversion completions are indicated by a status bit and an interrupt (if enabled), and the resulting 10-bit data word is latched into two SFR locations upon completion.

ADC2 also contains Window Compare registers, which can be configured to interrupt the controller when ADC2 data is within or outside of a specified range. ADC2 can monitor a key voltage continuously in background mode, and not interrupt the controller unless the converted data is within the specified window.

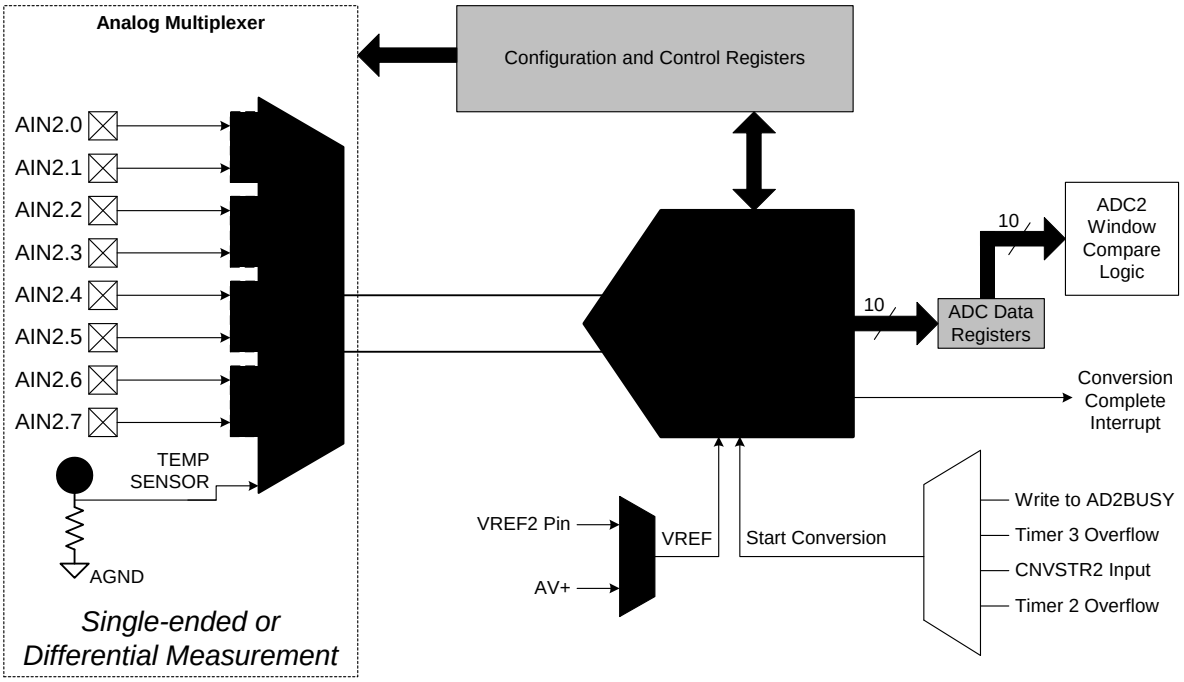
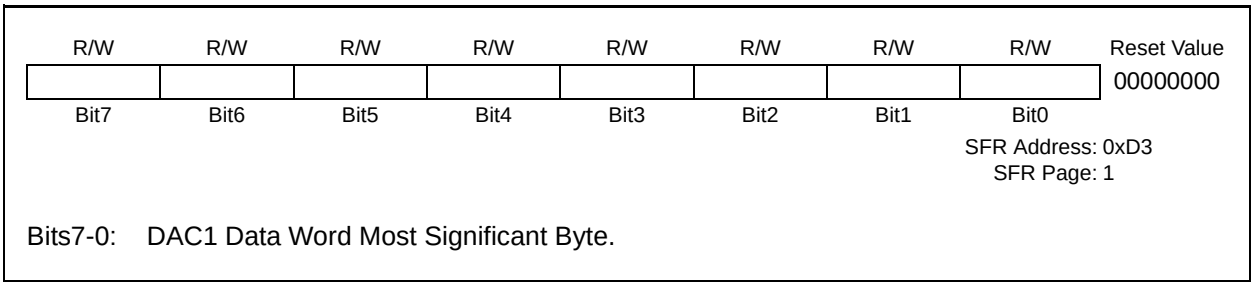


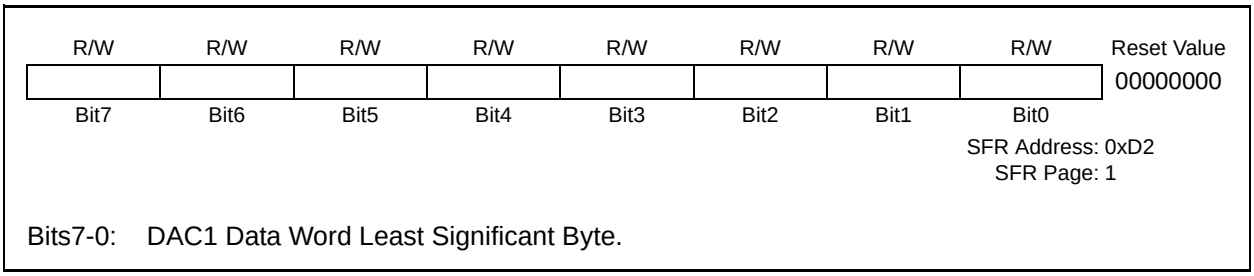
Figure 1.13. 10-Bit ADC Diagram

Figure 8.5. DAC1H: DAC1 High Byte Register



Bits7-0: DAC1 Data Word Most Significant Byte.

Figure 8.6. DAC1L: DAC1 Low Byte Register



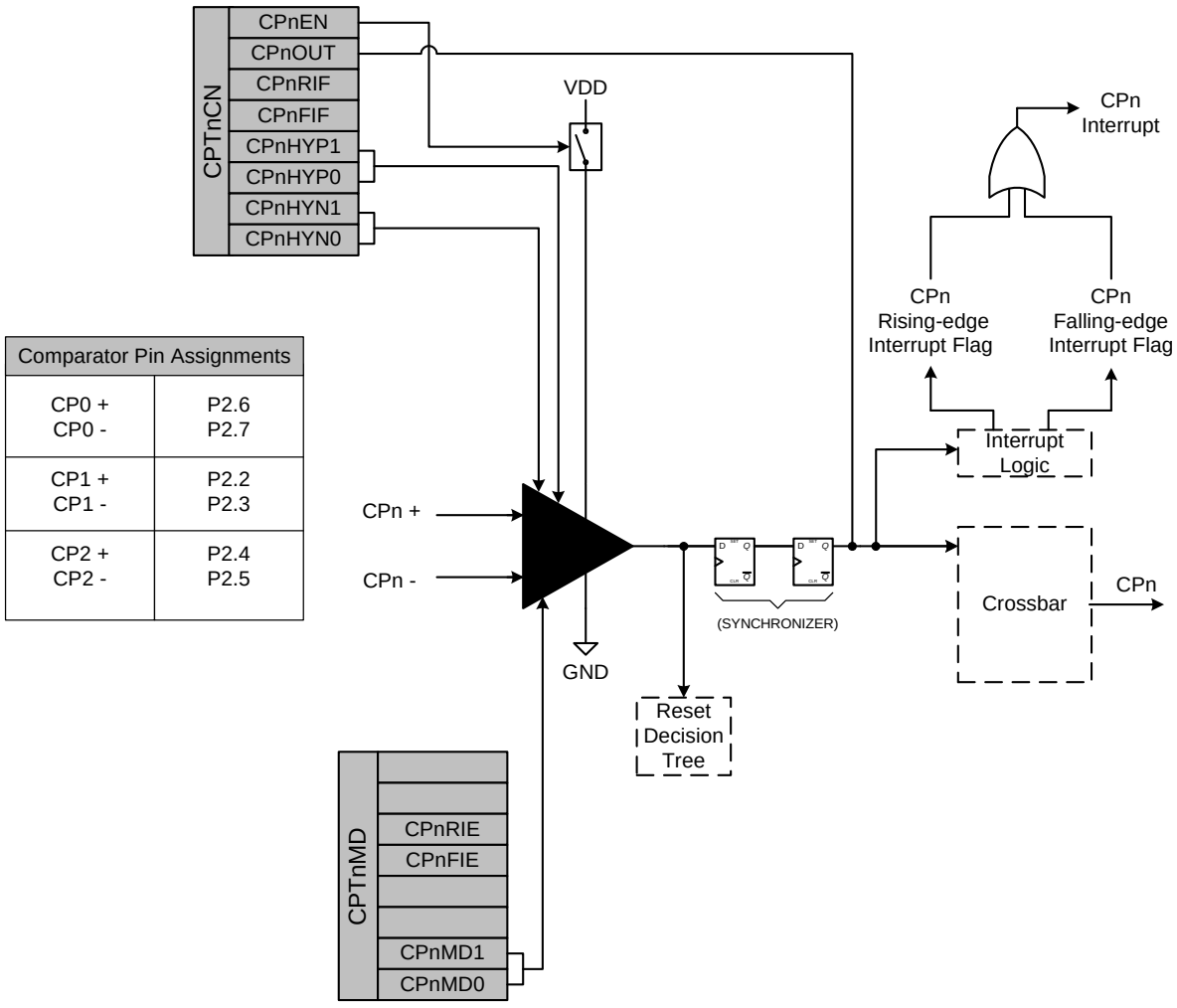
Bits7-0: DAC1 Data Word Least Significant Byte.

12. Comparators

C8051F06x family of devices include three on-chip programmable voltage comparators, shown in Figure 12.1. Each comparator offers programmable response time and hysteresis. When assigned to a Port pin, the Comparator output may be configured as open drain or push-pull, and Comparator inputs should be configured as analog inputs (see Section “18.1.5. Configuring Port 1 and 2 pins as Analog Inputs” on page 207). The Comparator may also be used as a reset source (see Section “14.5. Comparator0 Reset” on page 165).

The output of a Comparator can be polled by software, used as an interrupt source, used as a reset source, and/or routed to a Port pin. Each comparator can be individually enabled and disabled (shutdown). When disabled, the Comparator output (if assigned to a Port I/O pin via the Crossbar) defaults to the logic low state, and its supply current falls to less than 1 µA. See Section “18.1.1. Crossbar Pin Assignment and Allocation” on page 205 for details on configuring the Comparator output via the digital Crossbar. The Comparator inputs can be externally driven from -0.25 V to (VDD) + 0.25 V without damage or upset. The

Figure 12.1. Comparator Functional Block Diagram



Comparator Pin Assignments	
CP0 + CP0 -	P2.6 P2.7
CP1 + CP1 -	P2.2 P2.3
CP2 + CP2 -	P2.4 P2.5

Comparator interrupts can be generated on either rising-edge and falling-edge output transitions. (For Interrupt enable and priority control, see Section “13.3. Interrupt Handler” on page 151). The rising and/or falling -edge interrupts are enabled using the comparator’s Rising/Falling Edge Interrupt Enable Bits (CPnRIE and CPnFIE) in their respective Comparator Mode Selection Register (CPTnMD), shown in Figure 12.4. These bits allow the user to control which edge (or both) will cause a comparator interrupt. However, the comparator interrupt must also be enabled in the Extended Interrupt Enable Register (EIE1). The CPnFIF flag is set to logic 1 upon a Comparator falling-edge interrupt, and the CPnRIF flag is set to logic 1 upon the Comparator rising-edge interrupt. Once set, these bits remain set until cleared by software. The output state of a Comparator can be obtained at any time by reading the CPnOUT bit. A Comparator is enabled by setting its respective CPnEN bit to logic 1, and is disabled by clearing this bit to logic 0.Upon enabling a comparator, the output of the comparator is not immediately valid. Before using a comparator as an interrupt or reset source, software should wait for a minimum of the specified “Power-up time” as specified in Table 12.1, “Comparator Electrical Characteristics,” on page 122.

12.1. Comparator Inputs

The Port pins selected as comparator inputs should be configured as analog inputs in the Port 2 Input Configuration Register (for details on Port configuration, see Section “18.1.3. Configuring Port Pins as Digital Inputs” on page 207). The inputs for Comparator are on Port 2 as follows:

Comparator Input	Port PIN
CP0 +	P2.6
CP0 -	P2.7
CP1 +	P2.2
CP1 -	P2.3
CP2 +	P2.4
CP2 -	P2.5

Programming and Debugging Support

A JTAG-based serial interface is provided for in-system programming of the Flash program memory and communication with on-chip debug support logic. The re-programmable Flash can also be read and changed a single byte at a time by the application software using the MOVC and MOVX instructions. This feature allows program memory to be used for non-volatile data storage as well as updating program code under software control.

The on-chip debug support logic facilitates full speed in-circuit debugging, allowing the setting of hardware breakpoints and watch points, starting, stopping and single stepping through program execution (including interrupt service routines), examination of the program's call stack, and reading/writing the contents of registers and memory. This method of on-chip debug is completely non-intrusive and non-invasive, requiring no RAM, Stack, timers, or other on-chip resources.

The CIP-51 is supported by development tools from Silicon Labs and third party vendors. Silicon Labs provides an integrated development environment (IDE) which interfaces to the CIP-51 via its JTAG port to provide fast and efficient in-system device programming and debugging. Third party macro assemblers and C compilers are also available.

13.1. Instruction Set

The instruction set of the CIP-51 System Controller is fully compatible with the standard MCS-51™ instruction set; standard 8051 development tools can be used to develop software for the CIP-51. All CIP-51 instructions are the binary and functional equivalent of their MCS-51™ counterparts, including opcodes, addressing modes and effect on PSW flags. However, instruction timing is different than that of the standard 8051.

13.1.1. Instruction and CPU Timing

In many 8051 implementations, a distinction is made between machine cycles and clock cycles, with machine cycles varying from 2 to 12 clock cycles in length. However, the CIP-51 implementation is based solely on clock cycle timing. All instruction timings are specified in terms of clock cycles.

Due to the pipelined architecture of the CIP-51, most instructions execute in the same number of clock cycles as there are program bytes in the instruction. Conditional branch instructions take one less clock cycle to complete when the branch is not taken as opposed to when the branch is taken. Table 13.1 is the CIP-51 Instruction Set Summary, which includes the mnemonic, number of bytes, and number of clock cycles for each instruction.

13.1.2. MOVX Instruction and Program Memory

In the CIP-51, the MOVX instruction serves three purposes: accessing on-chip XRAM, accessing off-chip XRAM, and writing to on-chip program Flash memory. The Flash access feature provides a mechanism for user software to update program code and use the program memory space for non-volatile data storage (see Section “16. Flash Memory” on page 177). The External Memory Interface provides a fast access to off-chip XRAM (or memory-mapped peripherals) via the MOVX instruction. Refer to Section “17. External Data Memory Interface and On-Chip XRAM” on page 187 for details.

Table 13.1. CIP-51 Instruction Set Summary (Continued)

Mnemonic	Description	Bytes	Clock Cycles
ANL C, /bit	AND complement of direct bit to Carry	2	2
ORL C, bit	OR direct bit to carry	2	2
ORL C, /bit	OR complement of direct bit to Carry	2	2
MOV C, bit	Move direct bit to Carry	2	2
MOV bit, C	Move Carry to direct bit	2	2
JC rel	Jump if Carry is set	2	2/3
JNC rel	Jump if Carry is not set	2	2/3
JB bit, rel	Jump if direct bit is set	3	3/4
JNB bit, rel	Jump if direct bit is not set	3	3/4
JBC bit, rel	Jump if direct bit is set and clear bit	3	3/4
Program Branching			
ACALL addr11	Absolute subroutine call	2	3
LCALL addr16	Long subroutine call	3	4
RET	Return from subroutine	1	5
RETI	Return from interrupt	1	5
AJMP addr11	Absolute jump	2	3
LJMP addr16	Long jump	3	4
SJMP rel	Short jump (relative address)	2	3
JMP @A+DPTR	Jump indirect relative to DPTR	1	3
JZ rel	Jump if A equals zero	2	2/3
JNZ rel	Jump if A does not equal zero	2	2/3
CJNE A, direct, rel	Compare direct byte to A and jump if not equal	3	3/4
CJNE A, #data, rel	Compare immediate to A and jump if not equal	3	3/4
CJNE Rn, #data, rel	Compare immediate to Register and jump if not equal	3	3/4
CJNE @Ri, #data, rel	Compare immediate to indirect and jump if not equal	3	4/5
DJNZ Rn, rel	Decrement Register and jump if not zero	2	2/3
DJNZ direct, rel	Decrement direct byte and jump if not zero	3	3/4
NOP	No operation	1	1

13.2.6.3.SFR Page Stack Example

The following is an example that shows the operation of the SFR Page Stack during interrupts.

In this example, the SFR Page Control is left in the default enabled state (i.e., SFRPGEN = 1), and the CIP-51 is executing in-line code that is writing values to Port 5 (SFR “P5”, located at address 0xD8 on SFR Page 0x0F). The device is also using the Programmable Counter Array (PCA) and the 10-bit ADC (ADC2) window comparator to monitor a voltage. The PCA is timing a critical control function in its interrupt service routine (ISR), so its interrupt is enabled and is set to *high* priority. The ADC2 is monitoring a voltage that is less important, but to minimize the software overhead its window comparator is being used with an associated ISR that is set to *low* priority. At this point, the SFR page is set to access the Port 5 SFR (SFRPAGE = 0x0F). See Figure 13.4 below.

Figure 13.4. SFR Page Stack While Using SFR Page 0x0F To Access Port 5

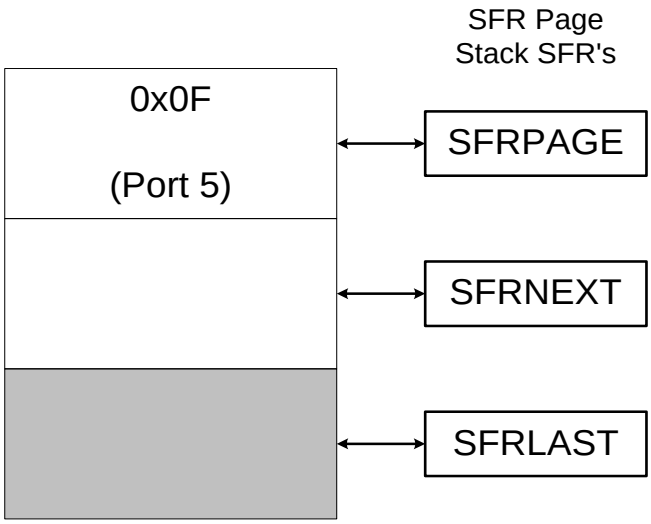


Table 13.3. Special Function Registers

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

Register	Address	SFR Page	Description	Page No.
B	0xF0	All Pages	B Register	page 150
ACC	0xE0	All Pages	Accumulator	page 150
ADC0CCF	0xBB	F	ADC0 Calibration Coefficient	page 68
ADC0CF	0xBC	0	ADC0 Configuration	page 58
ADC0CN	0xE8	0	ADC0 Control	page 60
ADC0CPT	0xBA	F	ADC0 Calibration Pointer	page 68
ADC0GTH	0xC5	0	ADC0 Greater-Than High	page 69
ADC0GTL	0xC4	0	ADC0 Greater-Than Low	page 69
ADC0H	0xBF	0	ADC0 Data Word High	page 63
ADC0L	0xBE	0	ADC0 Data Word Low	page 63
ADC0LTH	0xC7	0	ADC0 Less-Than High	page 70
ADC0LTL	0xC6	0	ADC0 Less-Than Low	page 70
ADC1CF	0xBC	1	ADC1 Configuration	page 59
ADC1CN	0xE8	1	ADC1 Control	page 61
ADC1H	0xBF	1	ADC1 Data Word High	page 65
ADC1L	0xBE	1	ADC1 Data Word Low	page 65
ADC2CF	0xBC	2	ADC2 Configuration	page 94*5
ADC2CN	0xE8	2	ADC2 Control	page 96*5
ADC2GTH	0xC5	2	ADC2 Greater-Than High	page 97*5
ADC2GTL	0xC4	2	ADC2 Greater-Than Low	page 97*5
ADC2H	0xBF	2	ADC2 Data Word High	page 95*5
ADC2L	0xBE	2	ADC2 Data Word Low	page 95*5
ADC2LTH	0xC7	2	ADC2 Less-Than High	page 98*5
ADC2LTL	0xC6	2	ADC2 Less-Than Low	page 98*5
AMX0SL	0xBB	0	ADC0 Multiplexer Channel Select	page 57
AMX2CF	0xBA	2	ADC2 Analog Multiplexer Configuration	page 94*5
AMX2SL	0xBB	2	ADC2 Analog Multiplexer Channel Select	page 93*5
CAN0ADR	0xDA	1	CAN0 Address	page 232*5
CAN0CN	0xF8	1	CAN0 Control	page 232*5
CAN0DATH	0xD9	1	CAN0 Data High	page 231*5
CAN0DATL	0xD8	1	CAN0 Data Low	page 231*5
CAN0STA	0xC0	1	CAN0 Status	page 233*5
CAN0TST	0xDB	1	CAN0 Test	page 233*5
CKCON	0x8E	0	Clock Control	page 293
CLKSEL	0x97	F	Oscillator Clock Selection Register	page 173
CPT0CN	0x88	1	Comparator 0 Control	page 120
CPT0MD	0x89	1	Comparator 0 Configuration	page 121
CPT1CN	0x88	2	Comparator 1 Control	page 120
CPT1MD	0x89	2	Comparator 1 Configuration	page 121
CPT2CN	0x88	3	Comparator 2 Control	page 120

Figure 13.24. EIP2: Extended Interrupt Priority 2

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
PDMA0	PS1	PCAN0	PADC2	PWADC2	PT4	PADC1	PT3	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

SFR Address: 0xF7
SFR Page: All Pages

Bit7: PDMA0: DMA0 Interrupt Priority Control.
This bit sets the priority of the DMA0 interrupt.
0: DMA0 interrupt set to low priority.
1: DMA0 interrupt set to high priority.

Bit6: PS1: UART1 Interrupt Priority Control.
This bit sets the priority of the UART1 interrupt.
0: UART1 interrupt set to low priority.
1: UART1 interrupt set to high priority.

Bit5: PCAN0: CAN Interrupt Priority Control.
This bit sets the priority of the CAN Interrupt.
0: CAN Interrupt set to low priority level.
1: CAN Interrupt set to high priority level.

Bit4: PADC2: ADC2 End Of Conversion Interrupt Priority Control.
This bit sets the priority of the ADC2 End of Conversion interrupt.
0: ADC2 End of Conversion interrupt set to low priority.
1: ADC2 End of Conversion interrupt set to high priority.

Bit3: PWADC2: ADC2 Window Comparator Interrupt Priority Control.
0: ADC2 Window interrupt set to low priority.
1: ADC2 Window interrupt set to high priority.

Bit2: PT4: Timer 4 Interrupt Priority Control.
This bit sets the priority of the Timer 4 interrupt.
0: Timer 4 interrupt set to low priority.
1: Timer 4 interrupt set to high priority.

Bit1: PADC1: ADC End of Conversion Interrupt Priority Control.
This bit sets the priority of the ADC1 End of Conversion Interrupt.
0: ADC1 End of Conversion interrupt set to low priority level.
1: ADC1 End of Conversion interrupt set to high priority level.

Bit0: PT3: Timer 3 Interrupt Priority Control.
This bit sets the priority of the Timer 3 interrupts.
0: Timer 3 interrupt set to low priority level.
1: Timer 3 interrupt set to high priority level.

For a 3 MHz system clock, this provides an interval range of 0.021 ms to 349.5 ms. WDTCN.7 must be logic 0 when setting this interval. Reading WDTCN returns the programmed interval. WDTCN.[2:0] reads 111b after a system reset.

Figure 14.3. WDTCN: Watchdog Timer Control Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								xxxxx111
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

SFR Address: 0xFF
SFR Page: All Pages

Bits7-0: WDT Control.
Writing 0xA5 both enables and reloads the WDT.
Writing 0xDE followed within 4 system clocks by 0xAD disables the WDT.
Writing 0xFF locks out the disable feature.

Bit4: Watchdog Status Bit (when Read).
Reading the WDTCN.[4] bit indicates the Watchdog Timer Status.
0: WDT is inactive.
1: WDT is active.

Bits2-0: Watchdog Timeout Interval Bits.
The WDTCN.[2:0] bits set the Watchdog Timeout Interval. When writing these bits, WDTCN.7 must be set to 0.

Table 17.1. AC Parameters for External Memory Interface

Parameter	Description	Min	Max	Units
T_{SYSCLK}	System Clock Period	40		ns
T_{ACS}	Address / Control Setup Time	0	$3*T_{SYSCLK}$	ns
T_{ACW}	Address / Control Pulse Width	$1*T_{SYSCLK}$	$16*T_{SYSCLK}$	ns
T_{ACH}	Address / Control Hold Time	0	$3*T_{SYSCLK}$	ns
T_{ALEH}	Address Latch Enable High Time	$1*T_{SYSCLK}$	$4*T_{SYSCLK}$	ns
T_{ALEL}	Address Latch Enable Low Time	$1*T_{SYSCLK}$	$4*T_{SYSCLK}$	ns
T_{WDS}	Write Data Setup Time	$1*T_{SYSCLK}$	$19*T_{SYSCLK}$	ns
T_{WDH}	Write Data Hold Time	0	$3*T_{SYSCLK}$	ns
T_{RDS}	Read Data Setup Time	20		ns
T_{RDH}	Read Data Hold Time	0		ns

19.1. Bosch CAN Controller Operation

The CAN Controller featured in the C8051F060/1/2/3 devices is a full implementation of Bosch's full CAN module and fully complies with CAN specification 2.0B.

The function and use of the CAN Controller is detailed in the *Bosch CAN User's Guide*. The User's Guide should be used as a reference to configure and use the CAN controller. This Silicon Labs datasheet describes how to access the CAN controller.

The CAN Control Register (CAN0CN), CAN Test Register (CAN0TST), and CAN Status Register (CAN0STA) in the CAN controller can be accessed directly or indirectly via CIP-51 SFRs. All other CAN registers must be accessed via an indirect indexing method. See "Using CAN0ADR, CAN0DATH, and CANDATL To Access CAN Registers" on page 229.

Table 19.1. CAN Register Index and Reset Values (Continued)

CAN Register Index	Register name	Reset Value	Notes
0x59	Message Valid 2	0x0000	Message valid flags for message objects (read only)

Figure 19.3. CAN0DATH: CAN Data Access Register High Byte

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

SFR Address: 0xD9
SFR Page: 1

Bit7-0: CAN0DATH: CAN Data Access Register High Byte.
The CAN0DAT Registers are used to read/write register values and data to and from the CAN Registers pointed to with the index number in the CAN0ADR Register.
The CAN0ADR Register is used to point the [CAN0DATH:CAN0DATL] to a desired CAN Register. The desired CAN Register's index number is moved into CAN0ADR. The CAN0DAT Register can then read/write to and from the CAN Register.

Figure 19.4. CAN0DATL: CAN Data Access Register Low Byte

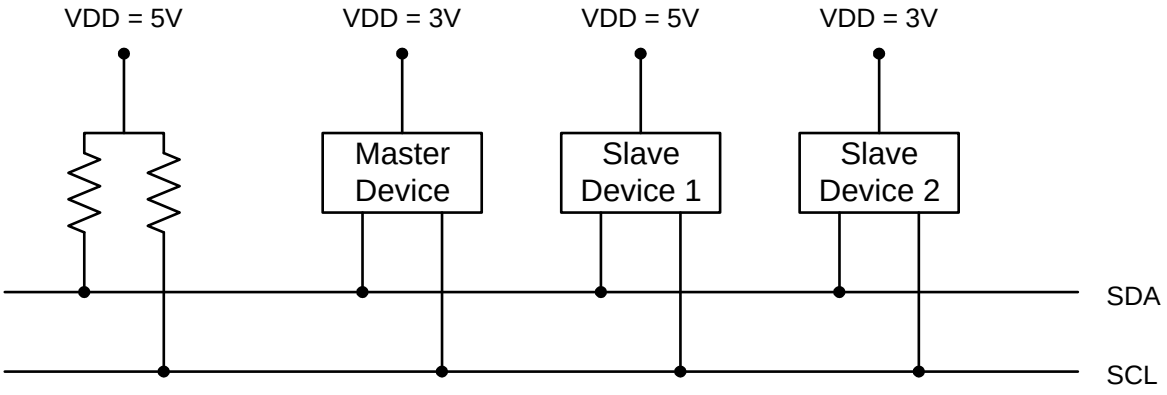
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000001
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

SFR Address: 0xD8
SFR Page: 1

Bit7-0: CAN0DATL: CAN Data Access Register Low Byte.
The CAN0DAT Registers are used to read/write register values and data to and from the CAN Registers pointed to with the index number in the CAN0ADR Register.
The CAN0ADR Register is used to point the [CAN0DATH:CAN0DATL] to a desired CAN Register. The desired CAN Register's index number is moved into CAN0ADR. The CAN0DAT Register can then read/write to and from the CAN Register.

Figure 20.2 shows a typical SMBus configuration. The SMBus0 interface will work at any voltage between 3.0 V and 5.0 V and different devices on the bus may operate at different voltage levels. The bi-directional SCL (serial clock) and SDA (serial data) lines must be connected to a positive power supply voltage through a pull-up resistor or similar circuit. Every device connected to the bus must have an open-drain or open-collector output for both the SCL and SDA lines, so that both are pulled high when the bus is free. The maximum number of devices on the bus is limited only by the requirement that the rise and fall times on the bus will not exceed 300 ns and 1000 ns, respectively.

Figure 20.2. Typical SMBus Configuration



20.1. Supporting Documents

It is assumed the reader is familiar with or has access to the following supporting documents:

1. The I2C-bus and how to use it (including specifications), Philips Semiconductor.
2. The I2C-Bus Specification -- Version 2.0, Philips Semiconductor.
3. System Management Bus Specification -- Version 1.1, SBS Implementers Forum.

20.2. SMBus Protocol

Two types of data transfers are possible: data transfers from a master transmitter to an addressed slave receiver (WRITE), and data transfers from an addressed slave transmitter to a master receiver (READ). The master device initiates both types of data transfers and provides the serial clock pulses on SCL. Note: multiple master devices on the same bus are supported. If two or more masters attempt to initiate a data transfer simultaneously, an arbitration scheme is employed with a single master always winning the arbitration. Note that it is not necessary to specify one device as the master in a system; any device who transmits a START and a slave address becomes the master for that transfer.

A typical SMBus transaction consists of a START condition followed by an address byte (Bits7-1: 7-bit slave address; Bit0: R/W direction bit), one or more bytes of data, and a STOP condition. Each byte that is received (by a master or slave) must be acknowledged (ACK) with a low SDA during a high SCL (see Figure 20.3). If the receiving device does not ACK, the transmitting device will read a “not acknowledge” (NACK), which is a high SDA during a high SCL.

Figure 22.9. SSTA0: UART0 Status and Clock Selection Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
FE0	RXOV0	TXCOL0	SMOD0	S0TCLK1	S0TCLK0	S0RCLK1	S0RCLK0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

SFR Address: 0x91
SFR Page: 0

Bit7: FE0: Frame Error Flag.†
This flag indicates if an invalid (low) STOP bit is detected.
0: Frame Error has not been detected.
1: Frame Error has been detected.

Bit6: RXOV0: Receive Overrun Flag.†
This flag indicates new data has been latched into the receive buffer before software has read the previous byte.
0: Receive overrun has not been detected.
1: Receive Overrun has been detected.

Bit5: TXCOL0: Transmit Collision Flag.†
This flag indicates user software has written to the SBUF0 register while a transmission is in progress.
0: Transmission Collision has not been detected.
1: Transmission Collision has been detected.

Bit4: SMOD0: UART0 Baud Rate Doubler Enable.
This bit enables/disables the divide-by-two function of the UART0 baud rate logic for configurations described in the UART0 section.
0: UART0 baud rate divide-by-two enabled.
1: UART0 baud rate divide-by-two disabled.

Bits3-2: UART0 Transmit Baud Rate Clock Selection Bits.

S0TCLK1	S0TCLK0	**Serial Transmit Baud Rate Clock Source**
0	0	Timer 1 generates UART0 TX Baud Rate
0	1	Timer 2 Overflow generates UART0 TX baud rate
1	0	Timer 3 Overflow generates UART0 TX baud rate
1	1	Timer 4 Overflow generates UART0 TX baud rate

Bits1-0: UART0 Receive Baud Rate Clock Selection Bits.

S0RCLK1	S0RCLK0	**Serial Receive Baud Rate Clock Source**
0	0	Timer 1 generates UART0 RX Baud Rate
0	1	Timer 2 Overflow generates UART0 RX baud rate
1	0	Timer 3 Overflow generates UART0 RX baud rate
1	1	Timer 4 Overflow generates UART0 RX baud rate

† Note: FE0, RXOV0, and TXCOL0 are flags only, and no interrupt is generated by these conditions.

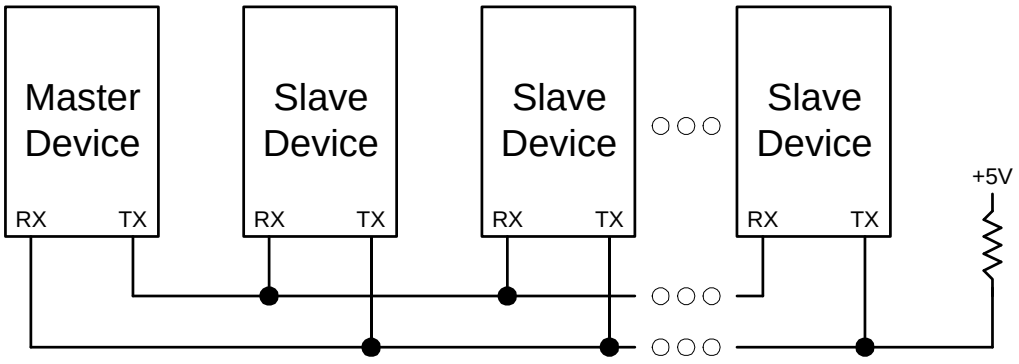
23.3. Multiprocessor Communications

9-Bit UART mode supports multiprocessor communication between a master processor and one or more slave processors by special use of the ninth 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 ninth bit is logic 1; in a data byte, the ninth bit is always set to logic 0.

Setting the MCE1 bit (SCON.5) of a slave processor configures its UART such that when a stop bit is received, the UART will generate an interrupt only if the ninth bit is logic one (RB81 = 1) signifying an address byte has been received. In the UART interrupt handler, software should compare the received address with the slave's own assigned 8-bit address. If the addresses match, the slave should 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 should reset 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



25.2.2. Software Timer (Compare) Mode

In Software Timer mode, the PCA0 counter/timer is compared to the module's 16-bit capture/compare register (PCA0CPHn and PCA0CPLn). When a match occurs, the Capture/Compare Flag (CCFn) in PCA0CN is set to logic 1 and an interrupt request is generated if CCF interrupts are enabled. The CCFn bit is not automatically cleared by hardware when the CPU vectors to the interrupt service routine, and must be cleared by software. Setting the ECOMn and MATn bits in the PCA0CPMn register enables Software Timer mode.

Important Note About Capture/Compare Registers: When writing a 16-bit value to the PCA0 Capture/Compare registers, the low byte should always be written first. Writing to PCA0CPLn clears the ECOMn bit to '0'; writing to PCA0CPHn sets ECOMn to '1'.

Figure 25.5. PCA Software Timer Mode Diagram

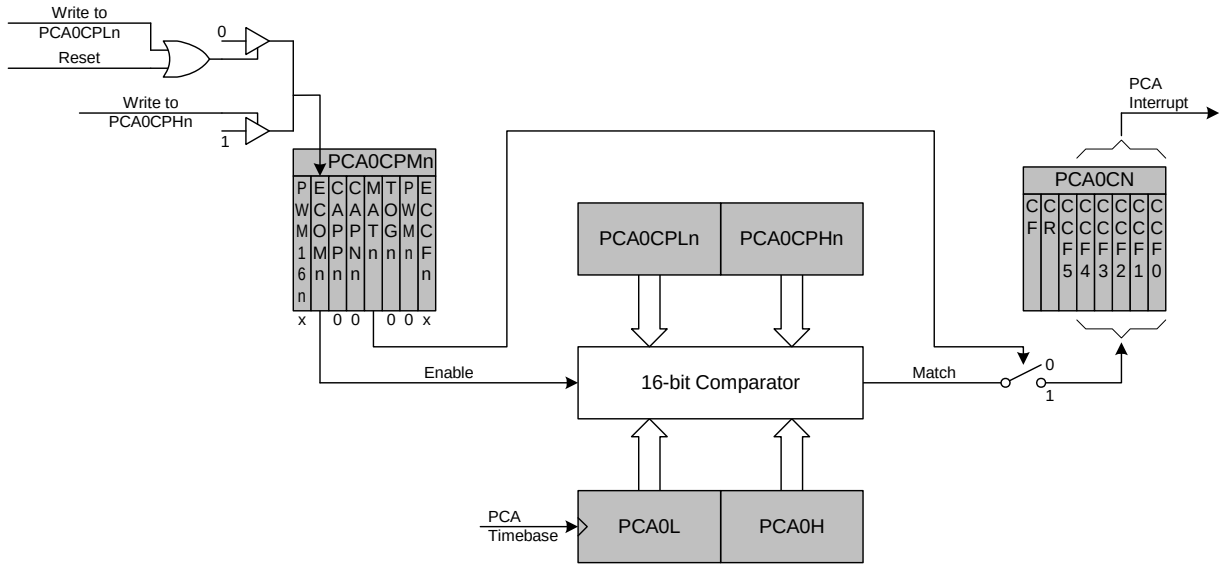


Table 26.2. Boundary Data Register Bit Definitions (C8051F061/3/5/7) (Continued)

EXTEST provides access to both capture and update actions, while Sample only performs a capture.

Bit	Action	Target
87, 89, 91, 93, 95, 97, 99, 101	Capture	P6.n input from pin (follows P0.n numbering scheme)†
	Update	P6.n output to pin (follows P0.n numbering scheme)†
102, 104, 106, 108, 110, 112, 114, 116	Capture	P7.n output enable from MCU (follows P0.n numbering scheme)†
	Update	P7.n output enable to pin (follows P0.n numbering scheme)†
103, 105, 107, 109, 111, 113, 115, 117	Capture	P7.n input from pin (follows P0.n numbering scheme)†
	Update	P7.n output to pin (follows P0.n numbering scheme)†
† Not connected to pins in this device package.		

26.1.1. EXTEST Instruction

The EXTEST instruction is accessed via the IR. The Boundary DR provides control and observability of all the device pins as well as the Weak Pullup feature. All inputs to on-chip logic are set to logic 1.

26.1.2. SAMPLE Instruction

The SAMPLE instruction is accessed via the IR. The Boundary DR provides observability and presetting of the scan-path latches.

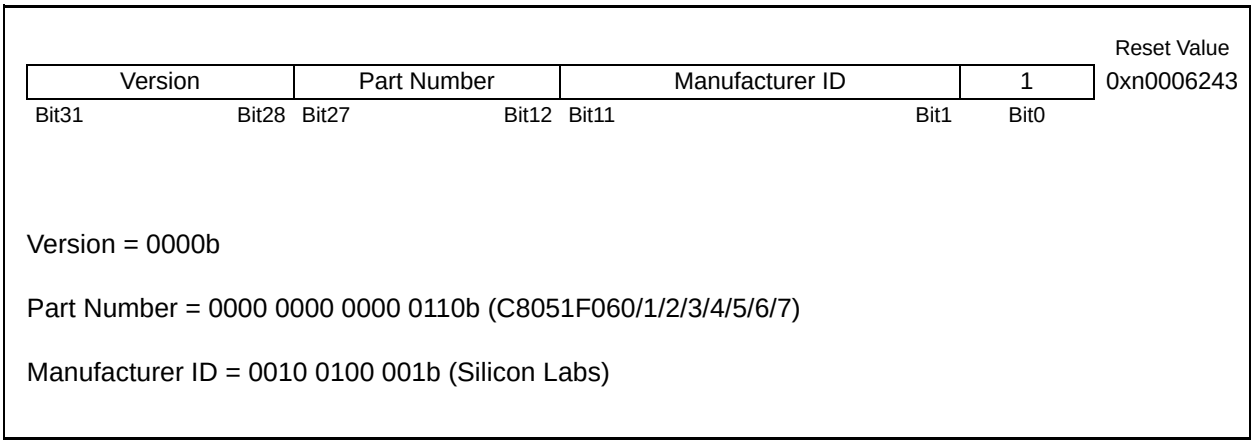
26.1.3. BYPASS Instruction

The BYPASS instruction is accessed via the IR. It provides access to the standard JTAG Bypass data register.

26.1.4. IDCODE Instruction

The IDCODE instruction is accessed via the IR. It provides access to the 32-bit Device ID register.

Figure 26.2. DEVICEID: JTAG Device ID Register



Version = 0000b

Part Number = 0000 0000 0000 0110b (C8051F060/1/2/3/4/5/6/7)

Manufacturer ID = 0010 0100 001b (Silicon Labs)