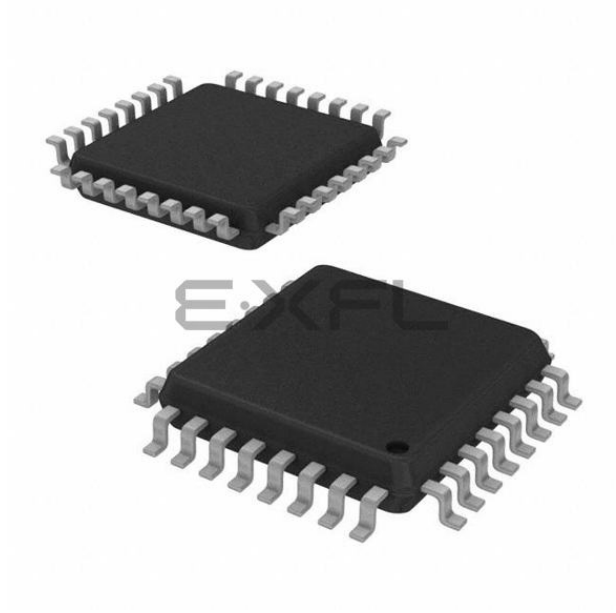


Welcome to E-XFL.COM

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	Obsolete
Core Processor	8051
Core Size	8-Bit
Speed	50MHz
Connectivity	SMBus (2-Wire/I²C), SPI, UART/USART
Peripherals	POR, PWM, Temp Sensor, WDT
Number of I/O	17
Program Memory Size	8KB (8K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	768 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 3.6V
Data Converters	A/D 8x16b; D/A 2x8b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	32-LQFP
Supplier Device Package	32-LQFP (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/silicon-labs/c8051f352r

Table of Contents

5. 24 or 16-Bit Analog to Digital Converter (ADC0)

The C8051F350/1/2/3 include a fully-differential, 24-bit (C8051F350/1) or 16-bit (C8051F352/3) Sigma-Delta Analog to Digital Converter (ADC) with on-chip calibration capabiliites. Two separate decimation filters can be programmed for throughputs of up to 1 kHz. An internal reference is available, or a differential external reference can be used for ratiometric measurements. A Programmable Gain Amplifier (PGA) is included, with eight gain settings up to 128x. The on-chip input buffers can be used to provide a high input impedance for direct connection to sensitive transducers. An 8-bit offset DAC allows for correction of large input offset voltages.

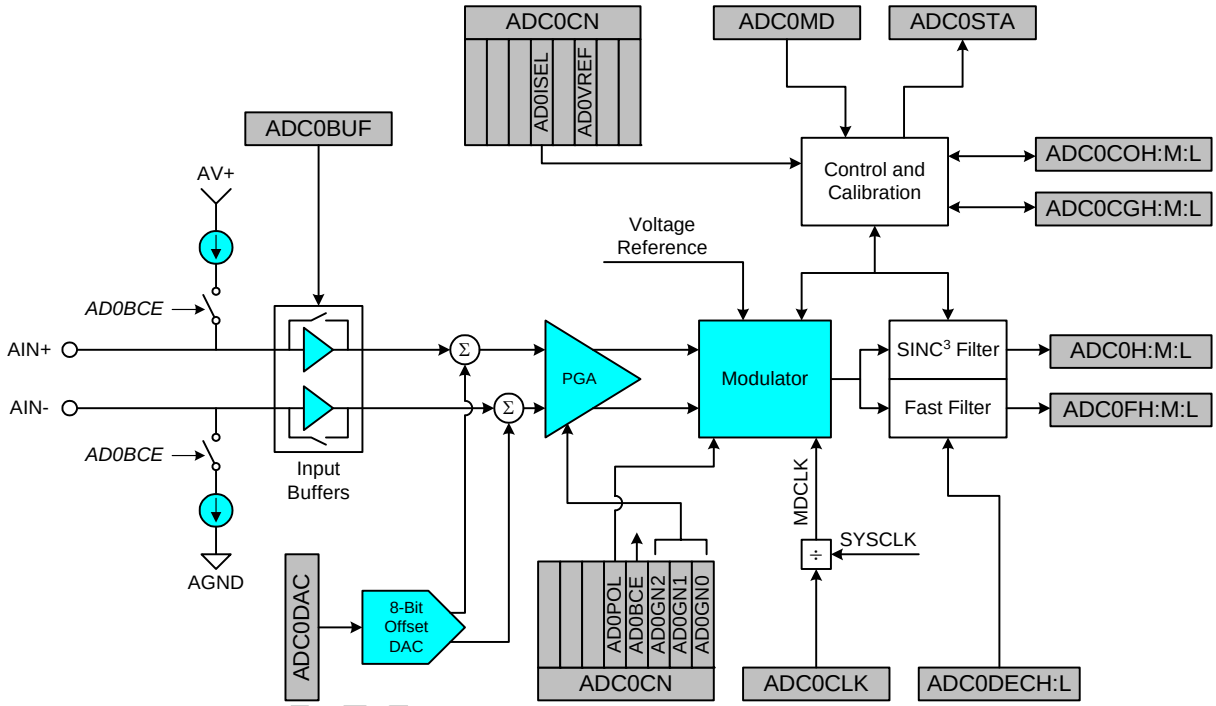


Figure 5.1. ADC0 Block Diagram

ADC0CGH, ADC0CGM, and ADC0CGL registers. The mapping of the gain register is shown in Figure 5.4.

The offset calibration value adjusts the zero point of the ADC's transfer function. It is stored as a two's complement, 24-bit number. An offset calibration which results in a full-scale positive (0x7FFFFF) or full-scale negative (0x800000) result will cause an ADC error condition.

The Offset Calibration results are stored in registers ADC0COH, ADC0COM, and ADC0COL. The weighting of the bits in the offset register (in LSBs) are shown below:

24-bit ADC (C8051F350/1)																							
ADC0COH								ADC0COM								ADC0COL							
MSB	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB
-2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}

16-bit ADC (C8051F352/3)																							
ADC0COH								ADC0COM								ADC0COL							
MSB	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB
-2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}

Figure 5.3. ADC0 Offset Calibration Register Coding

The gain calibration value adjusts the slope of the ADC's transfer function. The gain calibration regsiter can range from 0 to 2 - 2^{-23}. A gain calibration which results in either of these extremes will cause an ADC error condition.

The Gain Calibration results are stored in registers ADC0CGH, ADC0CGM, and ADC0CGL, as follows:

ADC0CGH								ADC0CGM								ADC0CGL							
MSB	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	LSB
2^{0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}	2^{-16}	2^{-17}	2^{-18}	2^{-19}	2^{-20}	2^{-21}	2^{-22}	2^{-23}

Example Decoding for Gain Register setting of 0x940000 (10010100 00000000 00000000b):

Slope Adjustment = $2^{0} + 2^{-3} + 2^{-5}$ = 1.0 + 0.125 + 0.03125 = 1.15625

Figure 5.4. ADC0 Gain Calibration Register Coding

6.1. IDAC Output Scheduling

A flexible output update mechanism allows for seamless full-scale changes and supports jitter-free updates for waveform generation. Three update modes are provided, allowing IDAC output updates on a write to the IDAC's data register, on a Timer overflow, or on an external pin edge.

6.1.1. Update Output On-Demand

In its default mode (IDAnCN.[6:4] = '111') the IDAC output is updated "on-demand" with a write to the data register (IDAn). In this mode, data is immediately latched into the IDAC after a write to its data register.

6.1.2. Update Output Based on Timer Overflow

The IDAC output update can be scheduled on a Timer overflow. This feature is useful in systems where the IDAC is used to generate a waveform of a defined sampling rate, by eliminating the effects of variable interrupt latency and instruction execution on the timing of the IDAC output. When the IDAnCM bits (IDAnCN.[6:4]) are set to '000', '001', '010' or '011', writes to the IDAC data register (IDAn) are held until an associated Timer overflow event (Timer 0, Timer 1, Timer 2 or Timer 3, respectively) occurs, at which time the data register contents are copied to the IDAC input latch, allowing the IDAC output to change to the new value.

6.1.3. Update Output Based on CNVSTR Edge

The IDAC output can also be configured to update on a rising edge, falling edge, or both edges of the external CNVSTR signal. When the IDAnCM bits (IDAnCN.[6:4]) are set to '100', '101', or '110', writes to the IDAC data register (IDAn) are held until an edge occurs on the CNVSTR input pin. The particular setting of the IDAnCM bits determines whether the IDAC output is updated on rising, falling, or both edges of CNVSTR. When a corresponding edge occurs, the data register contents are copied to the IDAC input latch, allowing the IDAC output to change to the new value.

6.2. IDAC Output Mapping

The data word mapping for the IDAC is shown in Figure 6.2. The full-scale output current of the IDAC is selected using the IDAnOMD bits (IDAnCN[1:0]). By default, the IDAC is set to a full-scale output current of 0.25 mA. The IDAnOMD bits can also be configured to provide full-scale output currents of 0.5 mA, 1 mA, or 2 mA.

Figure 6.2. IDAC Data Word Mapping

IDAn Data Word (D7 - D0)	Output Current vs IDAnOMD bit setting			
	'11' (2 mA)	'10' (1 mA)	'01' (0.5 mA)	'00' (0.25 mA)
0x00	0 mA	0 mA	0 mA	0 mA
0x01	1/256 x 2 mA	1/256 x 1 mA	1/256 x 0.5 mA	1/256 x 0.25 mA
0x80	128/256 x 2 mA	128/256 x 1 mA	128/256 x 0.5 mA	128/256 x 0.25 mA
0xFF	255/256 x 2 mA	255/256 x 1 mA	255/256 x 0.5 mA	255/256 x 0.25 mA

Table 10.1. CIP-51 Instruction Set Summary (Continued)

Mnemonic	Description	Bytes	Clock Cycles
DEC @Ri	Decrement indirect RAM	1	2
INC DPTR	Increment Data Pointer	1	1
MUL AB	Multiply A and B	1	4
DIV AB	Divide A by B	1	8
DA A	Decimal adjust A	1	1
Logical Operations			
ANL A, Rn	AND Register to A	1	1
ANL A, direct	AND direct byte to A	2	2
ANL A, @Ri	AND indirect RAM to A	1	2
ANL A, #data	AND immediate to A	2	2
ANL direct, A	AND A to direct byte	2	2
ANL direct, #data	AND immediate to direct byte	3	3
ORL A, Rn	OR Register to A	1	1
ORL A, direct	OR direct byte to A	2	2
ORL A, @Ri	OR indirect RAM to A	1	2
ORL A, #data	OR immediate to A	2	2
ORL direct, A	OR A to direct byte	2	2
ORL direct, #data	OR immediate to direct byte	3	3
XRL A, Rn	Exclusive-OR Register to A	1	1
XRL A, direct	Exclusive-OR direct byte to A	2	2
XRL A, @Ri	Exclusive-OR indirect RAM to A	1	2
XRL A, #data	Exclusive-OR immediate to A	2	2
XRL direct, A	Exclusive-OR A to direct byte	2	2
XRL direct, #data	Exclusive-OR immediate to direct byte	3	3
CLR A	Clear A	1	1
CPL A	Complement A	1	1
RL A	Rotate A left	1	1
RLC A	Rotate A left through Carry	1	1
RR A	Rotate A right	1	1
RRC A	Rotate A right through Carry	1	1
SWAP A	Swap nibbles of A	1	1
Data Transfer			
MOV A, Rn	Move Register to A	1	1
MOV A, direct	Move direct byte to A	2	2
MOV A, @Ri	Move indirect RAM to A	1	2
MOV A, #data	Move immediate to A	2	2
MOV Rn, A	Move A to Register	1	1
MOV Rn, direct	Move direct byte to Register	2	2
MOV Rn, #data	Move immediate to Register	2	2
MOV direct, A	Move A to direct byte	2	2
MOV direct, Rn	Move Register to direct byte	2	2
MOV direct, direct	Move direct byte to direct byte	3	3
MOV direct, @Ri	Move indirect RAM to direct byte	2	2
MOV direct, #data	Move immediate to direct byte	3	3
MOV @Ri, A	Move A to indirect RAM	1	2

10.2. Register Descriptions

Following are descriptions of SFRs related to the operation of the CIP-51 System Controller. Reserved bits should not be set to logic l. Future product versions may use these bits to implement new features in which case the reset value of the bit will be logic 0, selecting the feature's default state. Detailed descriptions of the remaining SFRs are included in the sections of the datasheet associated with their corresponding system function.

Figure 10.2. SP: Stack Pointer

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000111
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

SFR Address: 0x81

Bits7-0: SP: Stack Pointer.
The Stack Pointer holds the location of the top of the stack. The stack pointer is incremented before every PUSH operation. The SP register defaults to 0x07 after reset.

Figure 10.3. DPL: Data Pointer Low 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: 0x82

Bits7-0: DPL: Data Pointer Low.
The DPL register is the low byte of the 16-bit DPTR. DPTR is used to access indirectly addressed XRAM and Flash memory.

Figure 10.4. DPH: Data Pointer 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: 0x83

Bits7-0: DPH: Data Pointer High.
The DPH register is the high byte of the 16-bit DPTR. DPTR is used to access indirectly addressed XRAM and Flash memory.

12. Interrupt Handler

The C8051F35x family includes an extended interrupt system supporting a total of 12 interrupt sources with two priority levels. The allocation of interrupt sources between on-chip peripherals and external input 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 the Interrupt Enable and Extended Interrupt Enable SFRs. 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 that interrupts which occur when the EA bit is set to logic 0 will be held in a pending state, and will not be serviced until the EA bit is set back to logic 1.

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.

12.1. MCU Interrupt Sources and Vectors

The MCUs support 12 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 12.1 on page 100. 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).

12.2. 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 or EIP1) 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 12.1.

12.3. Interrupt Latency

Interrupt response time depends on the state of the CPU when the interrupt occurs. Pending interrupts are sampled and priority decoded each system clock cycle. Therefore, the fastest possible response time is 5 system clock cycles: 1 clock cycle to detect the interrupt and 4 clock cycles to complete the LCALL to the ISR. If an interrupt is pending when a RETI is executed, a single instruction is executed before an LCALL

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

Table 12.1. Interrupt Summary

Interrupt Source	Interrupt Vector	Priority Order	Pending Flag	Bit addressable?	Cleared by HW?	Enable Flag	Priority Control
Reset	0x0000	Top	None	N/A	N/A	Always Enabled	Always Highest
External Interrupt 0 (/INT0)	0x0003	0	IE0 (TCON.1)	Y	Y	EX0 (IE.0)	PX0 (IP.0)
Timer 0 Overflow	0x000B	1	TF0 (TCON.5)	Y	Y	ET0 (IE.1)	PT0 (IP.1)
External Interrupt 1 (/INT1)	0x0013	2	IE1 (TCON.3)	Y	Y	EX1 (IE.2)	PX1 (IP.2)
Timer 1 Overflow	0x001B	3	TF1 (TCON.7)	Y	Y	ET1 (IE.3)	PT1 (IP.3)
UART0	0x0023	4	RI0 (SCON0.0) TI0 (SCON0.1)	Y	N	ES0 (IE.4)	PS0 (IP.4)
Timer 2 Overflow	0x002B	5	TF2H (TMR2CN.7) TF2L (TMR2CN.6)	Y	N	ET2 (IE.5)	PT2 (IP.5)
SPI0	0x0033	6	SPIF (SPI0CN.7) WCOL (SPI0CN.6) MODF (SPI0CN.5) RXOVRN (SPI0CN.4)	Y	N	ESPI0 (IE.6)	PSPI0 (IP.6)
SMB0	0x003B	7	SI (SMB0CN.0)	Y	N	ESMB0 (EIE1.0)	PSMB0 (EIP1.0)
RESERVED	0x0043	8	N/A	N/A	N/A	N/A	N/A
RESERVED	0x004B	9	N/A	N/A	N/A	N/A	N/A
ADC0	0x0053	10	AD0INT (ADC0STA.5)	Y	N	EADC0 (EIE1.3)	PADC0 (EIP1.3)
Programmable Counter Array	0x005B	11	CF (PCA0CN.7) CCFn (PCA0CN.n)	Y	N	EPCA0 (EIE1.4)	PPCA0 (EIP1.4)
Comparator0	0x0063	12	CP0FIF (CPT0CN.4) CP0RIF (CPT0CN.5)	N	N	ECP0 (EIE1.5)	PCP0 (EIP1.5)
RESERVED	0x006B	13	N/A	N/A	N/A	N/A	N/A
Timer 3 Overflow	0x0073	14	TF3H (TMR3CN.7) TF3L (TMR3CN.6)	N	N	ET3 (EIE1.7)	PT3 (EIP1.7)

14.2. Power-Fail Reset / V_{DD} Monitor

When a power-down transition or power irregularity causes V_{DD} to drop below V_{RST}, the power supply monitor will drive the /RST pin low and hold the CIP-51 in a reset state (see Figure 14.2). When V_{DD} returns to a level above V_{RST}, the CIP-51 will be released from the reset state. Note that even though internal data memory contents are not altered by the power-fail reset, it is impossible to determine if V_{DD} dropped below the level required for data retention. If the PORSF flag reads '1', the data may no longer be valid. The V_{DD} monitor is enabled and selected as a reset source after power-on resets; however its defined state (enabled/disabled) is not altered by any other reset source. For example, if the V_{DD} monitor is disabled by software, and a software reset is performed, the V_{DD} monitor will still be disabled after the reset. **To protect the integrity of Flash contents, it is strongly recommended that the V_{DD} monitor remain enabled and selected as a reset source if software contains routines which erase or write Flash memory.**

The V_{DD} monitor must be enabled before it is selected as a reset source. Selecting the V_{DD} monitor as a reset source before it is enabled and stabilized may cause a system reset. The procedure for re-enabling the V_{DD} monitor and configuring the V_{DD} monitor as a reset source is shown below:

Step 1. Enable the V_{DD} monitor (VDMEN bit in VDM0CN = '1').
Step 2. Wait for the V_{DD} monitor to stabilize (see Table 14.1 for the V_{DD} Monitor turn-on time).
Note: This delay should be omitted if software contains routines which erase or write Flash memory.
Step 3. Select the V_{DD} monitor as a reset source (PORSF bit in RSTSRC = '1').

See Figure 14.2 for V_{DD} monitor timing; note that the reset delay is not incurred after a V_{DD} monitor reset. See Table 14.1 for complete electrical characteristics of the V_{DD} monitor.

Figure 14.3. VDM0CN: V_{DD} Monitor Control

R/W	R	R	R	R	R	R	R	Reset Value
VDMEN	V_{DD}STAT	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Variable
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

SFR Address: 0xFF

Bit7: VDMEN: V_{DD} Monitor Enable.
This bit is turns the V_{DD} monitor circuit on/off. The V_{DD} Monitor cannot generate system resets until it is also selected as a reset source in register RSTSRC (Figure 14.4). The V_{DD} Monitor must be allowed to stabilize before it is selected as a reset source. **Selecting the V_{DD} monitor as a reset source before it has stabilized may generate a system reset.** See Table 14.1 for the minimum V_{DD} Monitor turn-on time.
0: V_{DD} Monitor Disabled.
1: V_{DD} Monitor Enabled (default).

Bit6: V_{DD} STAT: V_{DD} Status.
This bit indicates the current power supply status (V_{DD} Monitor output).
0: V_{DD} is at or below the V_{DD} monitor threshold.
1: V_{DD} is above the V_{DD} monitor threshold.

Bits5-0: Reserved. Read = Variable. Write = don't care.

Figure 18.9. P0MDOUT: Port0 Output Mode Register

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: 0xA4

Bits7-0: Output Configuration Bits for P0.7-P0.0 (respectively): ignored if corresponding bit in register P0MDIN is logic 0.
0: Corresponding P0.n Output is open-drain.
1: Corresponding P0.n Output is push-pull.

(Note: When SDA and SCL appear on any of the Port I/O, each are open-drain regardless of the value of P0MDOUT).

Figure 18.10. P0SKIP: Port0 Skip Register

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: 0xD4

Bits7-0: P0SKIP[7:0]: Port0 Crossbar Skip Enable Bits.
These bits select Port pins to be skipped by the Crossbar Decoder. Port pins used as analog inputs (for ADC or Comparator) or used as special functions (VREF input, external oscillator circuit, CNVSTR input) should be skipped by the Crossbar.
0: Corresponding P0.n pin is not skipped by the Crossbar.
1: Corresponding P0.n pin is skipped by the Crossbar.

19.4.2. SMB0CN Control Register

SMB0CN is used to control the interface and to provide status information (see Figure 19.6). The higher four bits of SMB0CN (MASTER, TXMODE, STA, and STO) form a status vector that can be used to jump to service routines. MASTER and TXMODE indicate the master/slave state and transmit/receive modes, respectively.

STA and STO indicate that a START and/or STOP has been detected or generated since the last SMBus interrupt. STA and STO are also used to generate START and STOP conditions when operating as a master. Writing a ‘1’ to STA will cause the SMBus interface to enter Master Mode and generate a START when the bus becomes free (STA is not cleared by hardware after the START is generated). Writing a ‘1’ to STO while in Master Mode will cause the interface to generate a STOP and end the current transfer after the next ACK cycle. If STO and STA are both set (while in Master Mode), a STOP followed by a START will be generated.

As a receiver, writing the ACK bit defines the outgoing ACK value; as a transmitter, reading the ACK bit indicates the value received on the last ACK cycle. ACKRQ is set each time a byte is received, indicating that an outgoing ACK value is needed. When ACKRQ is set, software should write the desired outgoing value to the ACK bit before clearing SI. A NACK will be generated if software does not write the ACK bit before clearing SI. SDA will reflect the defined ACK value immediately following a write to the ACK bit; however SCL will remain low until SI is cleared. If a received slave address is not acknowledged, further slave events will be ignored until the next START is detected.

The ARBLOST bit indicates that the interface has lost an arbitration. This may occur anytime the interface is transmitting (master or slave). A lost arbitration while operating as a slave indicates a bus error condition. ARBLOST is cleared by hardware each time SI is cleared.

The SI bit (SMBus Interrupt Flag) is set at the beginning and end of each transfer, after each byte frame, or when an arbitration is lost; see Table 19.3 for more details.

Important note about the SI bit: The SMBus interface is stalled while SI is set; thus SCL is held low, and the bus is stalled until software clears SI.

Table 19.3 lists all sources for hardware changes to the SMB0CN bits. Refer to Table 19.4 for SMBus status decoding using the SMB0CN register.

19.4.3. 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.

Figure 19.7. SMB0DAT: SMBus Data Register

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: 0xC2

Bits7-0: SMB0DAT: SMBus Data.
The SMB0DAT register contains a byte of data to be transmitted on the SMBus serial interface or a byte that has just been received on the SMBus serial interface. The CPU can read from or write to this register whenever the SI serial interrupt flag (SMB0CN.0) is set to logic 1. The serial data in the register remains stable as long as the SI flag is set. When the SI flag is not set, the system may be in the process of shifting data in/out and the CPU should not attempt to access this register.

22.2. Timer 2

Timer 2 is a 16-bit timer formed by two 8-bit SFRs: TMR2L (low byte) and TMR2H (high byte). Timer 2 may operate in 16-bit auto-reload mode or (split) 8-bit auto-reload mode. The T2SPLIT bit (TMR2CN.3) defines the Timer 2 operation mode.

Timer 2 may be clocked by the system clock, the system clock divided by 12, or the external oscillator source divided by 8. The external clock mode is ideal for real-time clock (RTC) functionality, where the internal oscillator drives the system clock while Timer 2 (and/or the PCA) is clocked by an external precision oscillator. Note that the external oscillator source divided by 8 is synchronized with the system clock.

22.2.1. 16-bit Timer with Auto-Reload

When T2SPLIT (TMR2CN.3) is zero, Timer 2 operates as a 16-bit timer with auto-reload. Timer 2 can be clocked by SYSCLK, SYSCLK divided by 12, or the external oscillator clock source divided by 8. As the 16-bit timer register increments and overflows from 0xFFFF to 0x0000, the 16-bit value in the Timer 2 reload registers (TMR2RLH and TMR2RLL) is loaded into the Timer 2 register as shown in Figure 22.11, and the Timer 2 High Byte Overflow Flag (TMR2CN.7) is set. If Timer 2 interrupts are enabled (if IE.5 is set), an interrupt will be generated on each Timer 2 overflow. Additionally, if Timer 2 interrupts are enabled and the TF2LEN bit is set (TMR2CN.5), an interrupt will be generated each time the lower 8 bits (TMR2L) overflow from 0xFF to 0x00.

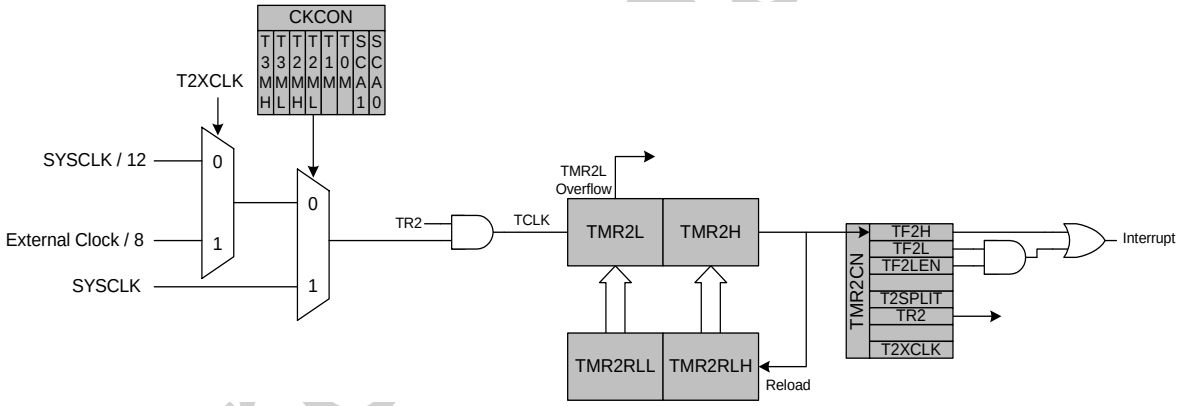


Figure 22.11. Timer 2 16-Bit Mode Block Diagram

23.2.1. Edge-triggered Capture Mode

In this mode, a valid transition on the CEXn pin causes the PCA to capture the value of the PCA counter/timer and load it into the corresponding module's 16-bit capture/compare register (PCA0CPLn and PCA0CPHn). The CAPPn and CAPNn bits in the PCA0CPMn register are used to select the type of transition that triggers the capture: low-to-high transition (positive edge), high-to-low transition (negative edge), or either transition (positive or negative edge). When a capture 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. If both CAPPn and CAPNn bits are set to logic 1, then the state of the Port pin associated with CEXn can be read directly to determine whether a rising-edge or falling-edge caused the capture.

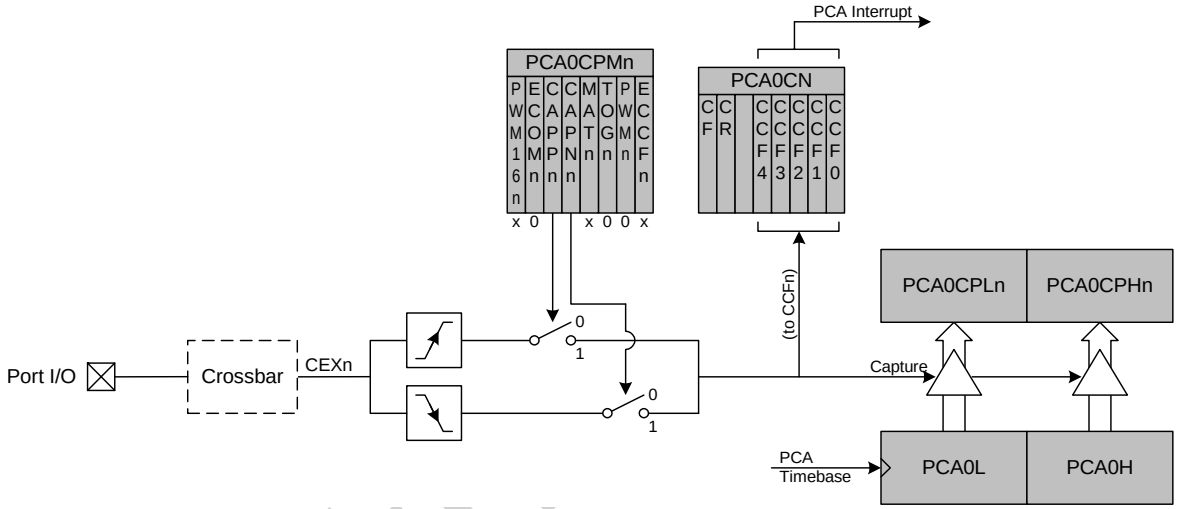


Figure 23.4. PCA Capture Mode Diagram

Note: The CEXn input signal must remain high or low for at least 2 system clock cycles to be recognized by the hardware.

Equation 23.4. Watchdog Timer Offset in PCA Clocks

$$Offset = (256 \times PCA0CPL4) + (256 - PCA0L)$$

The WDT reset is generated when PCA0L overflows while there is a match between PCA0CPH2 and PCA0H. Software may force a WDT reset by writing a '1' to the CCF2 flag (PCA0CN.2) while the WDT is enabled.

23.3.2. Watchdog Timer Usage

To configure the WDT, perform the following tasks:

- Disable the WDT by writing a '0' to the WDTE bit.
- Select the desired PCA clock source (with the CPS2-CPS0 bits).
- Load PCA0CPL2 with the desired WDT update offset value.
- Configure the PCA Idle mode (set CIDL if the WDT should be suspended while the CPU is in Idle mode).
- Enable the WDT by setting the WDTE bit to '1'.

The PCA clock source and Idle mode select cannot be changed while the WDT is enabled. The watchdog timer is enabled by setting the WDTE or WDLCK bits in the PCA0MD register. When WDLCK is set, the WDT cannot be disabled until the next system reset. If WDLCK is not set, the WDT is disabled by clearing the WDTE bit.

The WDT is enabled following any reset. The PCA0 counter clock defaults to the system clock divided by 12, PCA0L defaults to 0x00, and PCA0CPL2 defaults to 0x00. Using Equation 23.4, this results in a WDT timeout interval of 256 system clock cycles. Table 23.3 lists some example timeout intervals for typical system clocks.

Table 23.3. Watchdog Timer Timeout Intervals†

System Clock (Hz)	PCA0CPL2	Timeout Interval (ms)
24,500,000	255	32.1
24,500,000	128	16.2
24,500,000	32	4.1
18,432,000	255	42.7
18,432,000	128	21.5
18,432,000	32	5.5
11,059,200	255	71.1
11,059,200	128	35.8
11,059,200	32	9.2
3,060,000††	255	257
3,060,000††	128	129.5
3,060,000††	32	33.1
32,000	255	24576
32,000	128	12384
32,000	32	3168

†Assumes SYSCLK / 12 as the PCA clock source, and a PCA0L value of 0x00 at the update time.

††Internal oscillator reset frequency.

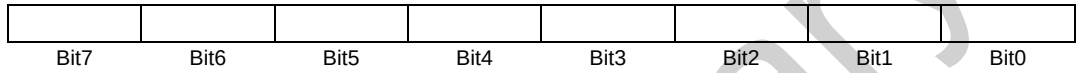
24. C2 Interface

C8051F350/1/2/3 devices include an on-chip Silicon Labs 2-Wire (C2) debug interface to allow Flash programming, boundary scan functions, and in-system debugging with the production part installed in the end application. The C2 interface uses a clock signal (C2CK) and a bi-directional C2 data signal (C2D) to transfer information between the device and a host system. See the C2 Interface Specification for details on the C2 protocol.

24.1. C2 Interface Registers

The following describes the C2 registers necessary to perform Flash programming and boundary scan functions through the C2 interface. All C2 registers are accessed through the C2 interface as described in the C2 Interface Specification.

Figure 24.1. C2ADD: C2 Address Register

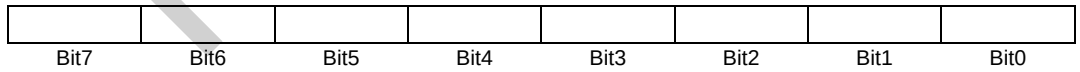


Reset Value
00000000

Bits7-0: The C2ADD register is accessed via the C2 interface to select the target Data register for C2 Data Read and Data Write commands.

Address	Description
0x00	Selects the Device ID register for Data Read instructions
0x01	Selects the Revision ID register for Data Read instructions
0x02	Selects the C2 Flash Programming Control register for Data Read/Write instructions
0xB4	Selects the C2 Flash Programming Data register for Data Read/Write instructions

Figure 24.2. DEVICEID: C2 Device ID Register



Reset Value
00001011

This read-only register returns the 8-bit device ID: 0x0B (C8051F350/1/2/3).

24.2. C2 Pin Sharing

The C2 protocol allows the C2 pins to be shared with user functions so that in-system debugging, Flash programming, and boundary scan functions may be performed. This is possible because C2 communication is typically performed when the device is in the halt state, where all on-chip peripherals and user software are stalled. In this halted state, the C2 interface can safely ‘borrow’ the C2CK (/RST) and C2D (P2.0) pins. In most applications, external resistors are required to isolate C2 interface traffic from the user application. A typical isolation configuration is shown in Figure 24.6.

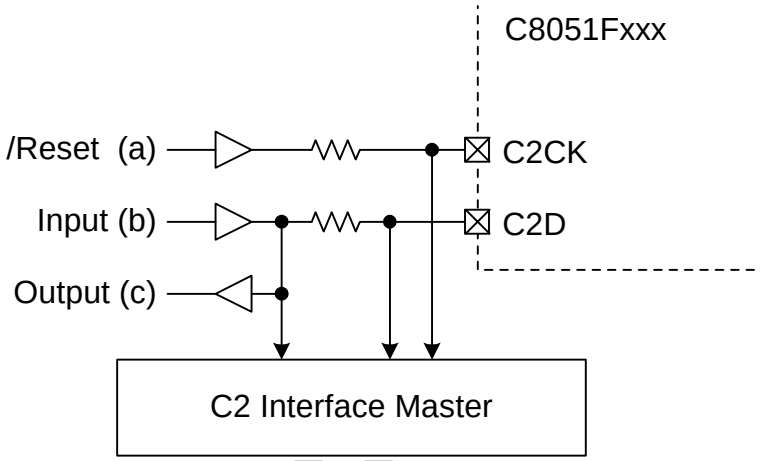


Figure 24.6. Typical C2 Pin Sharing

The configuration in Figure 24.6 assumes the following:

1. The user input (b) cannot change state while the target device is halted.
2. The /RST pin on the target device is used as an input only.

Additional resistors may be necessary depending on the specific application.