Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	CIP-51 8051
Core Size	8-Bit
Speed	25MHz
Connectivity	SMBus (2-Wire/I²C), SPI, UART/USART
Peripherals	Brown-out Detect/Reset, Cap Sense, 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	512 x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 10x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	Die
Supplier Device Package	Die
Purchase URL	https://www.e-xfl.com/product-detail/silicon-labs/c8051f996-c-gdi

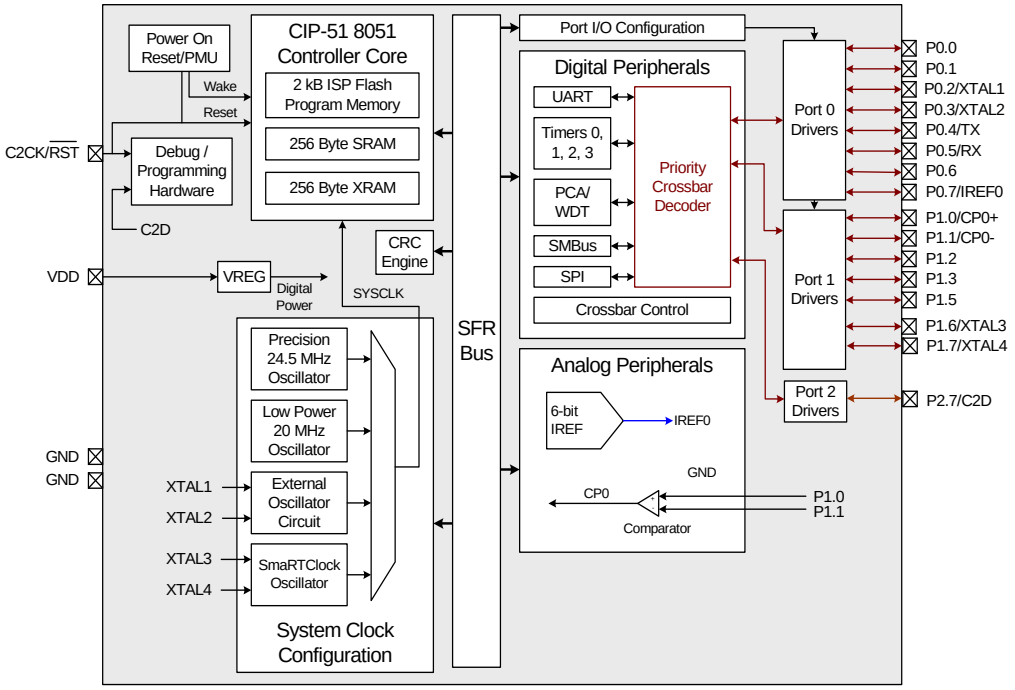

Figure 1.5. C8051F985 Block Diagram

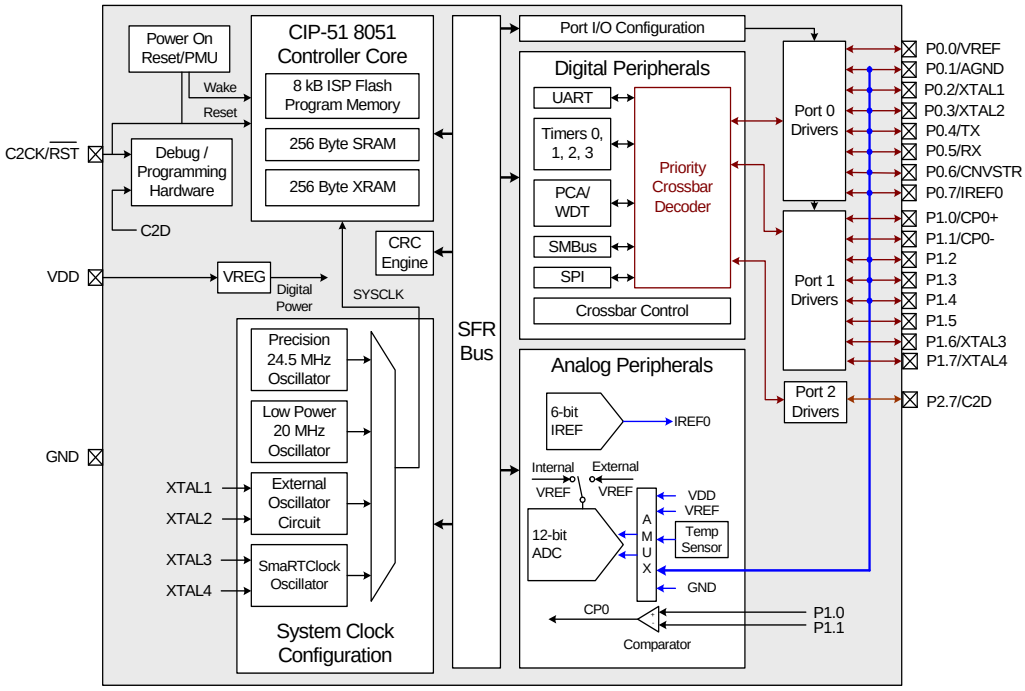

Figure 1.6. C8051F986 Block Diagram

1.5. SAR ADC with 16-bit Auto-Averaging Accumulator and Autonomous Low Power Burst Mode

C8051F99x-C8051F98x devices have a 300 ksps, 10-bit or 75 ksps 12-bit successive-approximation-register (SAR) ADC with integrated track-and-hold and programmable window detector. ADC0 also has an autonomous low power Burst Mode which can automatically enable ADC0, capture and accumulate samples, then place ADC0 in a low power shutdown mode without CPU intervention. It also has a 16-bit accumulator that can automatically average the ADC results, providing an effective 11, 12, or 13 bit ADC result without any additional CPU intervention.

The ADC can sample the voltage at select GPIO pins (see Figure 1.17) and has an on-chip attenuator that allows it to measure voltages up to twice the voltage reference. Additional ADC inputs include an on-chip temperature sensor, the VDD supply voltage, and the internal digital supply voltage.

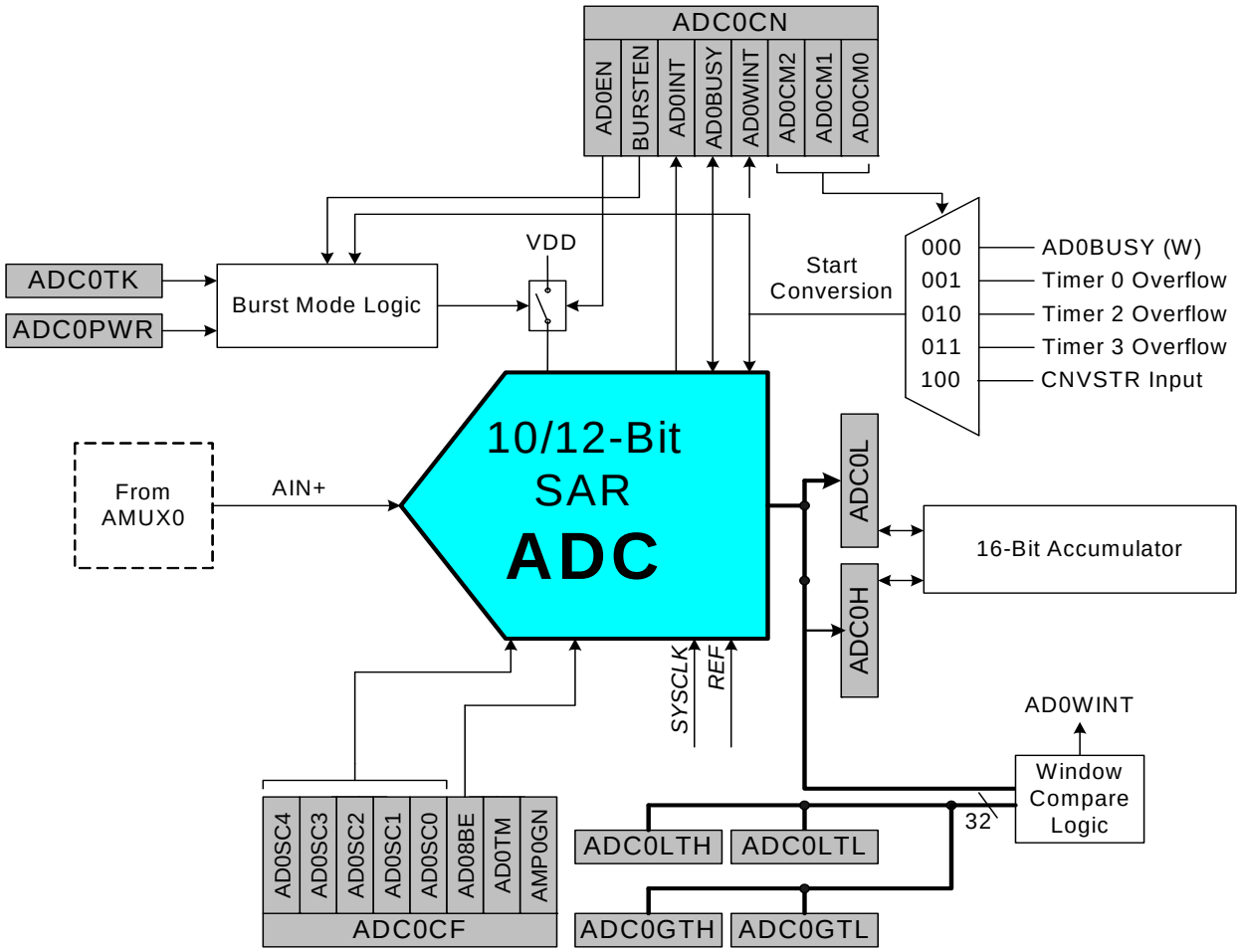

Figure 1.16. ADC0 Functional Block Diagram

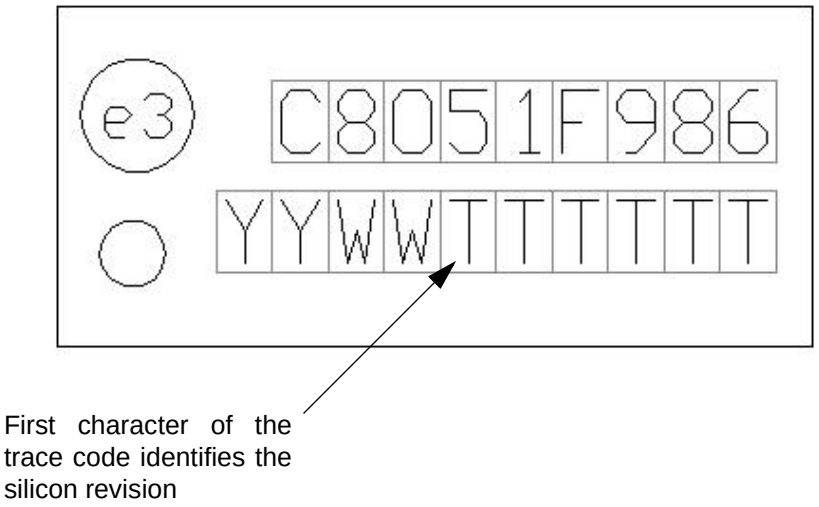

Figure 3.6. QSOP-24 Package Marking Diagram

SFR Definition 5.12. ADC0MX: ADC0 Input Channel Select

Bit	7	6	5	4	3	2	1	0
Name				AD0MX				
Type	R	R	R	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	1	1	1	1	1

SFR Page = 0x0; SFR Address = 0x96

Bit	Name	Function
7:5	Unused	Read = 000b; Write = Don't Care.
4:0	AD0MX	**AMUX0 Positive Input Selection.** Selects the positive input channel for ADC0.

00000:	Reserved.	10000:	Reserved.
00001:	P0.1	10001:	Reserved.
00010:	P0.2	10010:	Reserved.
00011:	P0.3	10011:	Reserved.
00100:	P0.4	10100:	Reserved.
00101:	P0.5	10101:	Reserved.
00110:	P0.6	10110:	Reserved.
00111:	P0.7	10111:	Reserved.
01000:	Reserved.	11000:	Reserved.
01001:	Reserved.	11001:	Reserved.
01010:	P1.2	11010:	Reserved.
01011:	P1.3	11011:	Temperature Sensor
01100:	P1.4 (only available on 24-pin devices)	11100:	V_{DD} Supply Voltage
01101:	Reserved.	11101:	Digital Supply Voltage (VREG0 Output, 1.7 V Typical)
01110:	Reserved.		
01111:	Reserved.	11110:	V_{DD} Supply Voltage
		11111:	Ground

Table 9.1. CIP-51 Instruction Set Summary

Mnemonic	Description	Bytes	Clock Cycles
Arithmetic Operations			
ADD A, Rn	Add register to A	1	1
ADD A, direct	Add direct byte to A	2	2
ADD A, @Ri	Add indirect RAM to A	1	2
ADD A, #data	Add immediate to A	2	2
ADDC A, Rn	Add register to A with carry	1	1
ADDC A, direct	Add direct byte to A with carry	2	2
ADDC A, @Ri	Add indirect RAM to A with carry	1	2
ADDC A, #data	Add immediate to A with carry	2	2
SUBB A, Rn	Subtract register from A with borrow	1	1
SUBB A, direct	Subtract direct byte from A with borrow	2	2
SUBB A, @Ri	Subtract indirect RAM from A with borrow	1	2
SUBB A, #data	Subtract immediate from A with borrow	2	2
INC A	Increment A	1	1
INC Rn	Increment register	1	1
INC direct	Increment direct byte	2	2
INC @Ri	Increment indirect RAM	1	2
DEC A	Decrement A	1	1
DEC Rn	Decrement register	1	1
DEC direct	Decrement direct byte	2	2
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

12. Special Function Registers

The direct-access data memory locations from 0x80 to 0xFF constitute the special function registers (SFRs). The SFRs provide control and data exchange with the C8051F99x-C8051F98x's resources and peripherals. The CIP-51 controller core duplicates the SFRs found in a typical 8051 implementation as well as implementing additional SFRs used to configure and access the sub-systems unique to the C8051F99x-C8051F98x. This allows the addition of new functionality while retaining compatibility with the MCS-51™ instruction set. Table 12.1 and Table 12.2 list the SFRs implemented in the C8051F99x-C8051F98x device family.

The SFR registers are accessed anytime the direct addressing mode is used to access memory locations from 0x80 to 0xFF. SFRs with addresses ending in 0x0 or 0x8 (e.g., P0, TCON, SCON0, IE, etc.) are bit-addressable as well as byte-addressable. All other SFRs are byte-addressable only. Unoccupied addresses in the SFR space are reserved for future use. Accessing these areas will have an indeterminate effect and should be avoided. Refer to the corresponding pages of the data sheet, as indicated in Table 12.3, for a detailed description of each register.

Table 12.1. Special Function Register (SFR) Memory Map (Page 0x0)

F8	SPI0CN	PCA0L	PCA0H	PCA0CPL0	PCA0CPH0	CS0THL	CS0THH	VDM0CN
F0	B	P0MDIN	P1MDIN	CS0MD2	SMB0ADR	SMB0ADM	EIP1	EIP2
E8	ADC0CN	PCA0CPL1	PCA0CPH1	PCA0CPL2	PCA0CPH2	CS0DL	CS0DH	RSTSRC
E0	ACC	XBR0	XBR1	XBR2	IT01CF	FLWR	EIE1	EIE2
D8	PCA0CN	PCA0MD	PCA0CPM0	PCA0CPM1	PCA0CPM2	CS0SS	CS0SE	PCA0PWM
D0	PSW	REF0CN	CS0SCAN0	CS0SCAN1	P0SKIP	P1SKIP	IREF0CN	P0MAT
C8	TMR2CN	REG0CN	TMR2RLL	TMR2RLH	TMR2L	TMR2H	PMU0FL	P1MAT
C0	SMB0CN	SMB0CF	SMB0DAT	ADC0GTL	ADC0GTH	ADC0LTL	ADC0LTH	P0MASK
B8	IP	IREF0CN	ADC0AC	ADC0PWR	ADC0TK	ADC0L	ADC0H	P1MASK
B0	CS0CN	OSCXCN	OSCICN	OSCICL		PMU0CF	FLSCL	FLKEY
A8	IE	CLKSEL	CS0CF	CS0MX	RTC0ADR	RTC0DAT	RTC0KEY	CS0MD1
A0	P2	SPI0CFG	SPI0CKR	SPI0DAT	P0MDOUT	P1MDOUT	P2MDOUT	SFRPAGE
98	SCON0	SBUF0	CRC0CNT	CPT0CN	CRC0FLIP	CPT0MD	CRC0AUTO	CPT0MX
90	P1	TMR3CN	TMR3RLL	TMR3RLH	TMR3L	TMR3H	ADC0MX	ADC0CF
88	TCON	TMOD	TL0	TL1	TH0	TH1	CKCON	PSCTL
80	P0	SP	DPL	DPH	CRC0CN	CRC0IN	CRC0DAT	PCON
	0(8)	1(9)	2(A)	3(B)	4(C)	5(D)	6(E)	7(F)

(bit addressable)

13. Interrupt Handler

The C8051F99x-C8051F98x microcontroller family includes an extended interrupt system supporting multiple interrupt sources and 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. Refer to Table 13.1, “Interrupt Summary,” on page 140 for a detailed listing of all interrupt sources supported by the device. 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).

Each interrupt source has one or more associated interrupt-pending flag(s) located in an SFR or an indirect register. When a peripheral or external source meets a valid interrupt condition, the associated interrupt-pending flag is set to logic 1. If both global interrupts and the specific interrupt source is enabled, a CPU 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.)

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

13.1. Enabling Interrupt Sources

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.

13.2. MCU Interrupt Sources and Vectors

The CPU services interrupts by generating an LCALL to a predetermined address (the interrupt vector address) to begin execution of an interrupt service routine (ISR). The interrupt vector addresses associated with each interrupt source are listed in Table 13.1 on page 140. Software should ensure that the interrupt vector for each enabled interrupt source contains a valid interrupt service routine.

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.

14.1.2. Flash Erase Procedure

The Flash memory is organized in 512-byte pages. The erase operation applies to an entire page (setting all bytes in the page to 0xFF). To erase an entire Flash page, perform the following steps:

1. Save current interrupt state and disable interrupts.
2. Set the PSEE bit (register PSCTL).
3. Set the PSWE bit (register PSCTL).
4. Write the first key code to FLKEY: 0xA5.
5. Write the second key code to FLKEY: 0xF1.
6. Using the MOVX instruction, write a data byte to any location within the page to be erased.
7. Clear the PSWE and PSEE bits.
8. Restore previous interrupt state.

Steps 4–6 must be repeated for each 512-byte page to be erased.

Notes:

1. Flash security settings may prevent erasure of some Flash pages, such as the reserved area and the page containing the lock bytes. For a summary of Flash security settings and restrictions affecting Flash erase operations, please see Section “14.3. Security Options” on page 152.
2. 8-bit MOVX instructions cannot be used to erase or write to Flash memory at addresses higher than 0x00FF.

14.1.3. Flash Write Procedure

A write to Flash memory can clear bits to logic 0 but cannot set them; only an erase operation can set bits to logic 1 in Flash. **A byte location to be programmed should be erased before a new value is written.**

The recommended procedure for writing a single byte in Flash is as follows:

1. Save current interrupt state and disable interrupts.
2. Ensure that the Flash byte has been erased (has a value of 0xFF).
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 1024-byte sector.
8. Clear the PSWE bit.
9. Restore previous interrupt state.

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

Notes:

1. Flash security settings may prevent writes to some areas of Flash, such as the reserved area. For a summary of Flash security settings and restrictions affecting Flash write operations, please see Section “14.3. Security Options” on page 152.
2. 8-bit MOVX instructions cannot be used to erase or write to Flash memory at addresses higher than 0x00FF.

14.2. Non-volatile Data Storage

The Flash memory can be used for non-volatile data storage as well as program code. This allows data such as calibration coefficients to be calculated and stored at run time. Data is written using the MOVX write instruction and read using the MOVC instruction. MOVX read instructions always target XRAM.

14.6. Minimizing Flash Read Current

The Flash memory in the C8051F99x-C8051F98x devices is responsible for a substantial portion of the total digital supply current when the device is executing code. Below are suggestions to minimize Flash read current.

1. Use idle, suspend, or sleep modes while waiting for an interrupt, rather than polling the interrupt flag. Idle Mode is particularly well-suited for use in implementing short pauses, since the wake-up time is no more than three system clock cycles. See the Power Management chapter for details on the various low-power operating modes.
2. C8051F99x-C8051F98x devices have a one-shot timer that saves power when operating at system clock frequencies of 14 MHz or less. The one-shot timer generates a minimum-duration enable signal for the Flash sense amps on each clock cycle in which the Flash memory is accessed. This allows the Flash to remain in a low power state for the remainder of the long clock cycle.
   At clock frequencies above 14 MHz, the system clock cycle becomes short enough that the one-shot timer no longer provides a power benefit. Disabling the one-shot timer at higher frequencies reduces power consumption. The one-shot is enabled by default, and it can be disabled (bypassed) by setting the BYPASS bit (FLSCL.6) to logic 1. To re-enable the one-shot, clear the BYPASS bit to logic 0.
3. Flash read current depends on the number of address lines that toggle between sequential Flash read operations. In most cases, the difference in power is relatively small (on the order of 5%).

   The Flash memory is organized in rows of 64 bytes. A substantial current increase can be detected when the read address jumps from one row in the Flash memory to another. Consider a 3-cycle loop (e.g., SJMP $, or while(1);) which straddles a Flash row boundary. The Flash address jumps from one row to another on two of every three clock cycles. This can result in a current increase of up 30% when compared to the same 3-cycle loop contained entirely within a single row.

   To minimize the power consumption of small loops, it is best to locate them within a single row, if possible. To check if a loop is contained within a Flash row, divide the starting address of the first instruction in the loop by 64. If the remainder (result of modulo operation) plus the length of the loop is less than 63, then the loop fits inside a single Flash row. Otherwise, the loop will be straddling two adjacent Flash rows. If a loop executes in 20 or more clock cycles, then the transitions from one row to another will occur on relatively few clock cycles, and any resulting increase in operating current will be negligible.

SFR Definition 14.5. FLSCL: Flash Scale

Bit	7	6	5	4	3	2	1	0
Name		BYPASS						
Type	R	R/W	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

SFR Page = 0x0; SFR Address = 0xB6

Bit	Name	Function
7	Reserved	Always Write to 0.
6	BYPASS	**Flash Read Timing One-Shot Bypass.** 0: The one-shot determines the Flash read time. This setting should be used for operating frequencies less than 14 MHz. 1: The system clock determines the Flash read time. This setting should be used for frequencies greater than 14 MHz.
5:0	Reserved	Reserved. Always Write to 000000b.

Note: Operations which clear the BYPASS bit do not need to be immediately followed by a benign 3-byte instruction. For code compatibility with C8051F930/31/20/21 devices, a benign 3-byte instruction whose third byte is a don't care should follow the clear operation. See the C8051F93x-C8051F92x data sheet for more details.

SFR Definition 14.6. FLWR: Flash Write Only

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

SFR Page = All; SFR Address = 0xE5

Bit	Name	Function
7:0	FLWR[7:0]	**Flash Write Only.** All writes to this register have no effect on system operation.

15.4. Suspend Mode

Setting the Suspend Mode Select bit (PMU0CF.6) causes the system clock to be gated off and all internal oscillators disabled. The system clock source must be set to the low power internal oscillator or the precision oscillator prior to entering Suspend Mode. All digital logic (timers, communication peripherals, interrupts, CPU, etc.) stops functioning until one of the enabled wake-up sources occurs.

The following wake-up sources can be configured to wake the device from Suspend Mode:

- CS0 End of Conversion or End of Scan
- SmaRTClock Oscillator Fail
- SmaRTClock Alarm
- Port Match Event
- Comparator0 Rising Edge

Note: Upon wake-up from suspend mode, PMU0 requires two system clocks in order to update the PMU0CF wake-up flags. All flags will read back a value of '0' during the first two system clocks following a wake-up from suspend mode.

In addition, a noise glitch on $\overline{\text{RST}}$ that is not long enough to reset the device will cause the device to exit suspend. In order for the MCU to respond to the pin reset event, software must not place the device back into suspend mode for a period of 15 µs. The PMU0CF register may be checked to determine if the wake-up was due to a falling edge on the /RST pin. If the wake-up source is not due to a falling edge on $\overline{\text{RST}}$, there is no time restriction on how soon software may place the device back into suspend mode. A 4.7 kW pullup resistor to VDD is recommend for RST to prevent noise glitches from waking the device.

15.5. Sleep Mode

Setting the Sleep Mode Select bit (PMU0CF.7) turns off the internal 1.8 V regulator (VREG0) and switches the power supply of all on-chip RAM to the VDD pin (see Figure 15.1). Power to most digital logic on the chip is disconnected; only PMU0 and the SmaRTClock remain powered. Analog peripherals remain powered in two-cell mode. Only the Comparators remain functional when the device enters Sleep Mode. All other analog peripherals (CS0, ADC0, IREF0, External Oscillator, etc.) should be disabled prior to entering Sleep Mode. The system clock source must be set to the low power internal oscillator prior to entering Sleep Mode.

Important Note: The precision internal oscillator may potentially lock up after exiting Sleep mode. Systems using Sleep Mode should switch to the low power oscillator prior to entering Sleep Mode:

1. Switch the system clock to the low power oscillator (CLKSEL = 0x04).
2. Turn off the Precision Oscillator (OSCICN &= ~0x80).
3. Enter Sleep.
4. Exit Sleep.
5. Turn on the Precision Oscillator (OSCICN |= 0x80).
6. Switch the system clock to the Precision Oscillator (CLKSEL = 0x00).

GPIO pins configured as digital outputs will retain their output state during sleep mode. In two-cell mode, they will maintain the same current drive capability in sleep mode as they have in normal mode.

GPIO pins configured as digital inputs can be used during sleep mode as wakeup sources using the port match feature. In two-cell mode, they will maintain the same input level specs in sleep mode as they have in normal mode.

C8051F99x-C8051F98x devices support a wakeup request for external devices. Upon exit from sleep mode, the wake-up request signal is driven low, allowing other devices in the system to wake up from their low power modes.

SFR Definition 15.1. PMU0CF: Power Management Unit Configuration[1,2,3]

Bit	7	6	5	4	3	2	1	0
Name	SLEEP	SUSPEND	CLEAR	RSTWK	RTCFWK	RTCAWK	PMATWK	CPT0WK
Type	W	W	W	R	R/W	R/W	R/W	R/W
Reset	0	0	0	Varies	Varies	Varies	Varies	Varies

SFR Page = 0x0; SFR Address = 0xB5

Bit	Name	Description	Write	Read
7	SLEEP	**Sleep Mode Select**	Writing 1 places the device in Sleep Mode.	N/A
6	SUSPEND	**Suspend Mode Select**	Writing 1 places the device in Suspend Mode.	N/A
5	CLEAR	**Wake-up Flag Clear**	Writing 1 clears all wake-up flags.	N/A
4	RSTWK	**Reset Pin Wake-up Flag**	N/A	Set to 1 if a glitch has been detected on $\overline{\text{RST}}$.
3	RTCFWK	**SmaRTClock Oscillator Fail Wake-up Source Enable and Flag**	0: Disable wake-up on SmaRTClock Osc. Fail. 1: Enable wake-up on SmaRTClock Osc. Fail.	Set to 1 if the SmaRT-Clock Oscillator has failed.
2	RTCAWK	**SmaRTClock Alarm Wake-up Source Enable and Flag**	0: Disable wake-up on SmaRTClock Alarm. 1: Enable wake-up on SmaRTClock Alarm.	Set to 1 if a SmaRTClock Alarm has occurred.
1	PMATWK	**Port Match Wake-up Source Enable and Flag**	0: Disable wake-up on Port Match Event. 1: Enable wake-up on Port Match Event.	Set to 1 if a Port Match Event has occurred.
0	CPT0WK	**Comparator0 Wake-up Source Enable and Flag**	0: Disable wake-up on Comparator0 rising edge. 1: Enable wake-up on Comparator0 rising edge.	Set to 1 if Comparator0 rising edge caused the last wake-up.

Notes:

1. Read-modify-write operations (ORL, ANL, etc.) should not be used on this register. Wake-up sources must be re-enabled each time the SLEEP or SUSPEND bits are written to 1.
2. The Low Power Internal Oscillator cannot be disabled and the MCU cannot be placed in Suspend or Sleep Mode if any wake-up flags are set to 1. Software should clear all wake-up sources after each reset and after each wake-up from Suspend or Sleep Modes.
3. PMU0 requires two system clocks to update the wake-up source flags after waking from Suspend mode. The wake-up source flags will read ‘0’ during the first two system clocks following the wake from Suspend mode.

20.1.5. RTC0ADR Autoincrement Feature

For ease of reading and writing the 32-bit CAPTURE and ALARM values, RTC0ADR automatically increments after each read or write to a CAPTUREn or ALARMn register. This speeds up the process of setting an alarm or reading the current SmaRTClock timer value. Autoincrement is always enabled.

Recommended Instruction Timing for a multi-byte register read with short strobe and auto read enabled:

```
mov RTC0ADR, #0d0h
nop
nop
nop
mov A, RTC0DAT
nop
nop
mov A, RTC0DAT
nop
nop
mov A, RTC0DAT
nop
nop
mov A, RTC0DAT
```

Recommended Instruction Timing for a multi-byte register write with short strobe enabled:

```
mov RTC0ADR, #010h
mov RTC0DAT, #05h
nop
mov RTC0DAT, #06h
nop
mov RTC0DAT, #07h
nop
mov RTC0DAT, #08h
nop
```

20.2.2. Using the SmaRTClock Oscillator in Self-Oscillate Mode

When using Self-Oscillate Mode, the XTAL3 and XTAL4 pins are internally shorted together. The following steps show how to configure SmaRTClock for use in Self-Oscillate Mode:

1. Set SmaRTClock to Self-Oscillate Mode (XMODE = 0).
2. Set the desired oscillation frequency:
   For oscillation at about 20 kHz, set BIASX2 = 0.
   For oscillation at about 40 kHz, set BIASX2 = 1.
3. The oscillator starts oscillating instantaneously.
4. Fine tune the oscillation frequency by adjusting the load capacitance (RTC0XCF).

20.2.3. Using the Low Frequency Oscillator (LFO)

The low frequency oscillator provides an ultra low power, on-chip clock source to the SmaRTClock. The typical frequency of oscillation is 16.4 kHz ±20%. No external components are required to use the LFO and the XTAL3 and XTAL4 pins do not need to be shorted together.

The following steps show how to configure SmaRTClock for use with the LFO:

1. Enable and select the Low Frequency Oscillator (LFOEN = 1).
2. The LFO starts oscillating instantaneously.

When the LFO is enabled, the SmaRTClock oscillator increments bit 1 of the 32-bit timer (instead of bit 0). This effectively multiplies the LFO frequency by 2, making the RTC timebase behave as if a 32.768 kHz crystal is connected at the output.

SFR Definition 21.6. P1MASK: Port1 Mask Register

Bit	7	6	5	4	3	2	1	0
Name	P1MASK[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Page= 0x0; SFR Address = 0xBF

Bit	Name	Function
7:0	P1MASK[7:0]	**Port 1 Mask Value.** Selects P1 pins to be compared to the corresponding bits in P1MAT. 0: P1.n pin logic value is ignored and cannot cause a Port Mismatch event. 1: P1.n pin logic value is compared to P1MAT.n.

SFR Definition 21.7. P1MAT: Port1 Match Register

Bit	7	6	5	4	3	2	1	0
Name	P1MAT[7:0]							
Type	R/W							
Reset	1	1	1	1	1	1	1	1

SFR Page = 0x0; SFR Address = 0xCF

Bit	Name	Function
7:0	P1MAT[7:0]	**Port 1 Match Value.** Match comparison value used on Port 1 for bits in P1MASK which are set to 1. 0: P1.n pin logic value is compared with logic LOW. 1: P1.n pin logic value is compared with logic HIGH.

SFR Definition 21.15. P1MDIN: Port1 Input Mode

Bit	7	6	5	4	3	2	1	0
Name	P1MDIN[7:0]							
Type	R/W							
Reset	1	1	1	1	1	1	1	1

SFR Page = 0x0; SFR Address = 0xF2

Bit	Name	Function
7:0	P1MDIN[7:0]	**Analog Configuration Bits for P1.7–P1.0 (respectively).** Port pins configured for analog mode have their weak pullup and digital receiver disabled. The digital driver is not explicitly disabled. 0: Corresponding P1.n pin is configured for analog mode. 1: Corresponding P1.n pin is not configured for analog mode.

SFR Definition 21.16. P1MDOUT: Port1 Output Mode

Bit	7	6	5	4	3	2	1	0
Name	P1MDOUT[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Page = 0x0; SFR Address = 0xA5

Bit	Name	Function
7:0	P1MDOUT[7:0]	**Output Configuration Bits for P1.7–P1.0 (respectively).** These bits control the digital driver even when the corresponding bit in register P1MDIN is logic 0. 0: Corresponding P1.n Output is open-drain. 1: Corresponding P1.n Output is push-pull.

SFR Definition 21.19. P2MDOUT: Port2 Output Mode

Bit	7	6	5	4	3	2	1	0
Name	P2MDOUT							
Type	R/W	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

SFR Page = 0x0; SFR Address = 0xA6

Bit	Name	Function
7	P2MDOUT	**Output Configuration Bits for P2.7.** These bits control the digital driver. 0: P2.7 Output is open-drain. 1: P2.7 Output is push-pull.
6:0	Unused	Read = 0000000b; Write = Don't Care.

SFR Definition 21.20. P2DRV: Port2 Drive Strength

Bit	7	6	5	4	3	2	1	0
Name	P2DRV							
Type	R/W	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

SFR Page = 0xF; SFR Address = 0x9D

Bit	Name	Function
7	P2DRV	**Drive Strength Configuration Bits for P2.7.** Configures digital I/O Port cells to high or low output drive strength. 0: P2.7 Output has low output drive strength. 1: P2.7 Output has high output drive strength.
6:0	Unused	Read = 0000000b; Write = Don't Care.

24. Enhanced Serial Peripheral Interface (SPI0)

The Enhanced Serial Peripheral Interface (SPI0) provides access to a flexible, full-duplex synchronous serial bus. SPI0 can operate as a master or slave device in both 3-wire or 4-wire modes, and supports multiple masters and slaves on a single SPI bus. The slave-select (NSS) signal can be configured as an input to select SPI0 in slave mode, or to disable Master Mode operation in a multi-master environment, avoiding contention on the SPI bus when more than one master attempts simultaneous data transfers. NSS can also be configured as a chip-select output in master mode, or disabled for 3-wire operation. Additional general purpose port I/O pins can be used to select multiple slave devices in master mode.

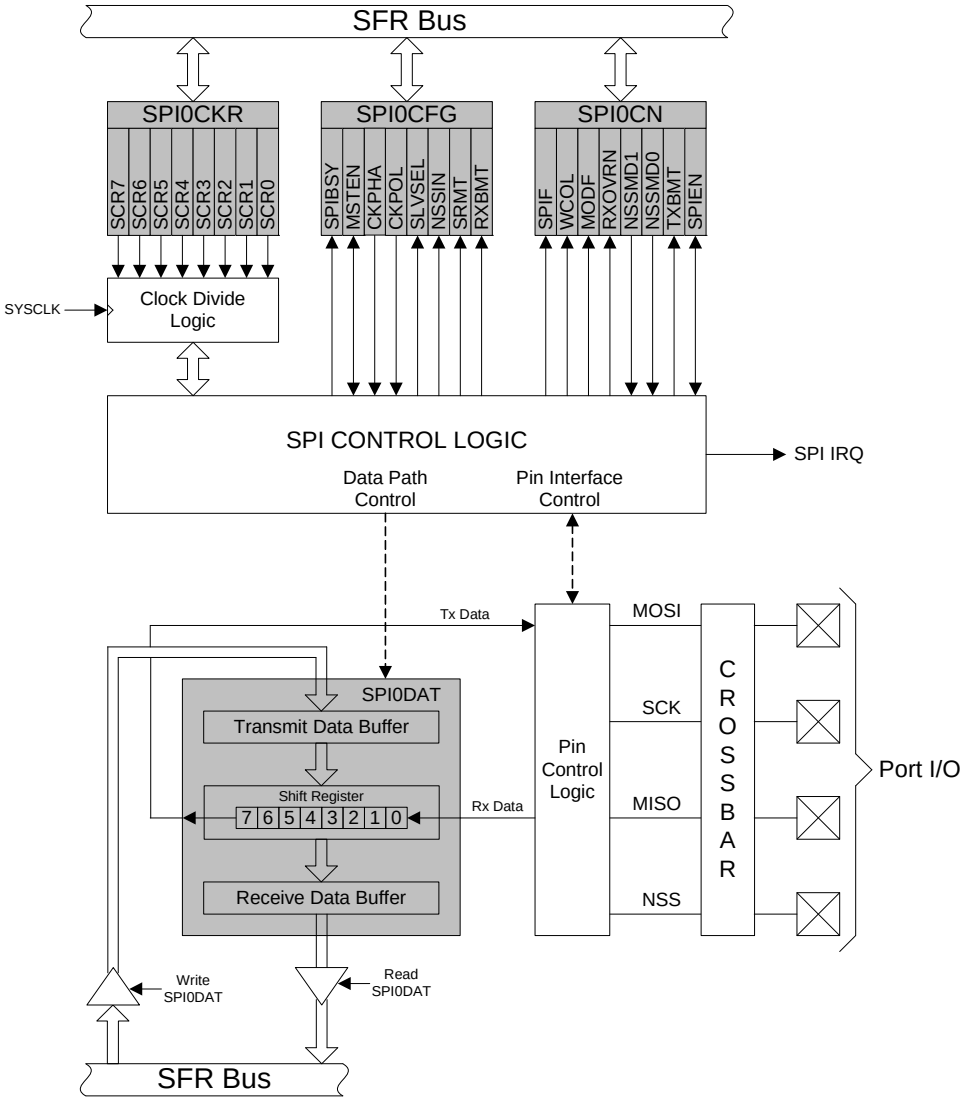

Figure 24.1. SPI Block Diagram

24.1. Signal Descriptions

The four signals used by SPI0 (MOSI, MISO, SCK, NSS) are described below.

24.1.1. Master Out, Slave In (MOSI)

The master-out, slave-in (MOSI) signal is an output from a master device and an input to slave devices. It is used to serially transfer data from the master to the slave. This signal is an output when SPI0 is operating as a master and an input when SPI0 is operating as a slave. Data is transferred most-significant bit first. When configured as a master, MOSI is driven by the MSB of the shift register in both 3- and 4-wire mode.

24.1.2. Master In, Slave Out (MISO)

The master-in, slave-out (MISO) signal is an output from a slave device and an input to the master device. It is used to serially transfer data from the slave to the master. This signal is an input when SPI0 is operating as a master and an output when SPI0 is operating as a slave. Data is transferred most-significant bit first. The MISO pin is placed in a high-impedance state when the SPI module is disabled and when the SPI operates in 4-wire mode as a slave that is not selected. When acting as a slave in 3-wire mode, MISO is always driven by the MSB of the shift register.

24.1.3. Serial Clock (SCK)

The serial clock (SCK) signal is an output from the master device and an input to slave devices. It is used to synchronize the transfer of data between the master and slave on the MOSI and MISO lines. SPI0 generates this signal when operating as a master. The SCK signal is ignored by a SPI slave when the slave is not selected (NSS = 1) in 4-wire slave mode.

24.1.4. Slave Select (NSS)

The function of the slave-select (NSS) signal is dependent on the setting of the NSSMD1 and NSSMD0 bits in the SPI0CN register. There are three possible modes that can be selected with these bits:

1. NSSMD[1:0] = 00: 3-Wire Master or 3-Wire Slave Mode: SPI0 operates in 3-wire mode, and NSS is disabled. When operating as a slave device, SPI0 is always selected in 3-wire mode. Since no select signal is present, SPI0 must be the only slave on the bus in 3-wire mode. This is intended for point-to-point communication between a master and one slave.
2. NSSMD[1:0] = 01: 4-Wire Slave or Multi-Master Mode: SPI0 operates in 4-wire mode, and NSS is enabled as an input. When operating as a slave, NSS selects the SPI0 device. When operating as a master, a 1-to-0 transition of the NSS signal disables the master function of SPI0 so that multiple master devices can be used on the same SPI bus.
3. NSSMD[1:0] = 1x: 4-Wire Master Mode: SPI0 operates in 4-wire mode, and NSS is enabled as an output. The setting of NSSMD0 determines what logic level the NSS pin will output. This configuration should only be used when operating SPI0 as a master device.

See Figure 24.2, Figure 24.3, and Figure 24.4 for typical connection diagrams of the various operational modes. **Note that the setting of NSSMD bits affects the pinout of the device.** When in 3-wire master or 3-wire slave mode, the NSS pin will not be mapped by the crossbar. In all other modes, the NSS signal will be mapped to a pin on the device. See Section “21. Port Input/Output” on page 215 for general purpose port I/O and crossbar information.

24.2. SPI0 Master Mode Operation

A SPI master device initiates all data transfers on a SPI bus. SPI0 is placed in master mode by setting the Master Enable flag (MSTEN, SPI0CN.6). Writing a byte of data to the SPI0 data register (SPI0DAT) when in master mode writes to the transmit buffer. If the SPI shift register is empty, the byte in the transmit buffer is moved to the shift register, and a data transfer begins. The SPI0 master immediately shifts out the data serially on the MOSI line while providing the serial clock on SCK. The SPIF (SPI0CN.7) flag is set to logic