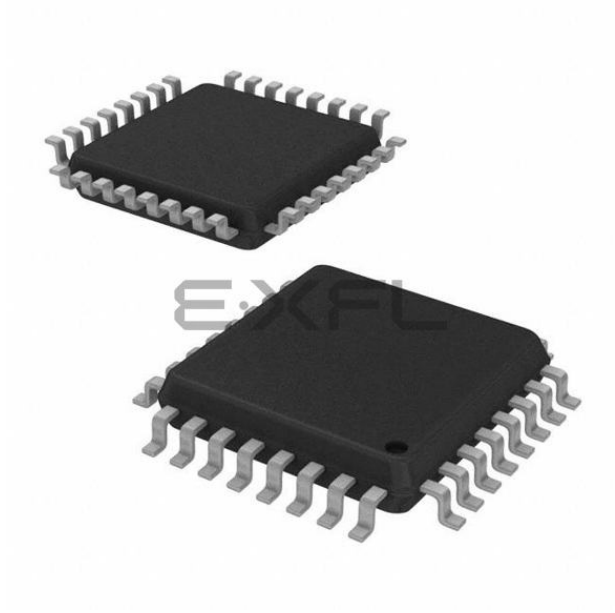

Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	8051
Core Size	8-Bit
Speed	50MHz
Connectivity	SMBus (2-Wire/I²C), LINbus, SPI, UART/USART
Peripherals	POR, PWM, Temp Sensor, WDT
Number of I/O	25
Program Memory Size	16KB (16K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	1.25K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 5.25V
Data Converters	A/D 25x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°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/c8051f540-iq

1. System Overview

C8051F54x devices are fully integrated mixed-signal System-on-a-Chip MCUs. Highlighted features are listed below. Refer to Table 2.1 for specific product feature selection and part ordering numbers.

- High-speed pipelined 8051-compatible microcontroller core (up to 50 MIPS)
- In-system, full-speed, non-intrusive debug interface (on-chip)
- LIN 2.1 peripheral (fully backwards compatible, master and slave modes) (C8051F540/2/4/6)
- True 12-bit 200 ksps 32-channel single-ended ADC with analog multiplexer
- Precision programmable 24 MHz internal oscillator that is within ±0.5% across the temperature range and for VDD voltages greater than or equal to the on-chip voltage regulator minimum output at the low setting. The oscillator is within ±1.0% for VDD voltages below this minimum output setting.
- On-chip Clock Multiplier to reach up to 50 MHz
- 16 kB (C8051F540/1/2/3) or 8 kB (C8051F544/5/6/7) of on-chip Flash memory
- 1280 bytes of on-chip RAM
- SMBus/I2C, Enhanced UART, and Enhanced SPI serial interfaces implemented in hardware
- Four general-purpose 16-bit timers
- Programmable Counter/Timer Array (PCA) with six capture/compare modules and Watchdog Timer function
- On-chip Voltage Regulator
- On-chip Power-On Reset, V_{DD} Monitor, and Temperature Sensor
- On-chip Voltage Comparator
- 25 or 18 Port I/O (5 V push-pull)

With on-chip Voltage Regulator, Power-On Reset, V_{DD} monitor, Watchdog Timer, and clock oscillator, the C8051F54x devices are truly stand-alone System-on-a-Chip solutions. The Flash memory can be reprogrammed even in-circuit, providing non-volatile data storage, and also allowing field upgrades of the 8051 firmware. User software has complete control of all peripherals, and may individually shut down any or all peripherals for power savings.

The on-chip Silicon Labs 2-Wire (C2) Development Interface allows non-intrusive (uses no on-chip resources), full speed, in-circuit debugging using the production MCU installed in the final application. This debug logic supports inspection and modification of memory and registers, setting breakpoints, single stepping, run and halt commands. All analog and digital peripherals are fully functional while debugging using C2. The two C2 interface pins can be shared with user functions, allowing in-system debugging without occupying package pins.

The devices are specified for 1.8 V to 5.25 V operation over the automotive temperature range (–40 to +125 °C). The C8051F540/1/4/5 devices are available in 32-pin QFP and QFN packages and the C8051F542/3/6/7 devices are available in 32-pin QFN packages. All package options are lead-free and RoHS compliant. See Table 2.1 for ordering information. Block diagrams are included in Figure 1.1 and Figure 1.2.

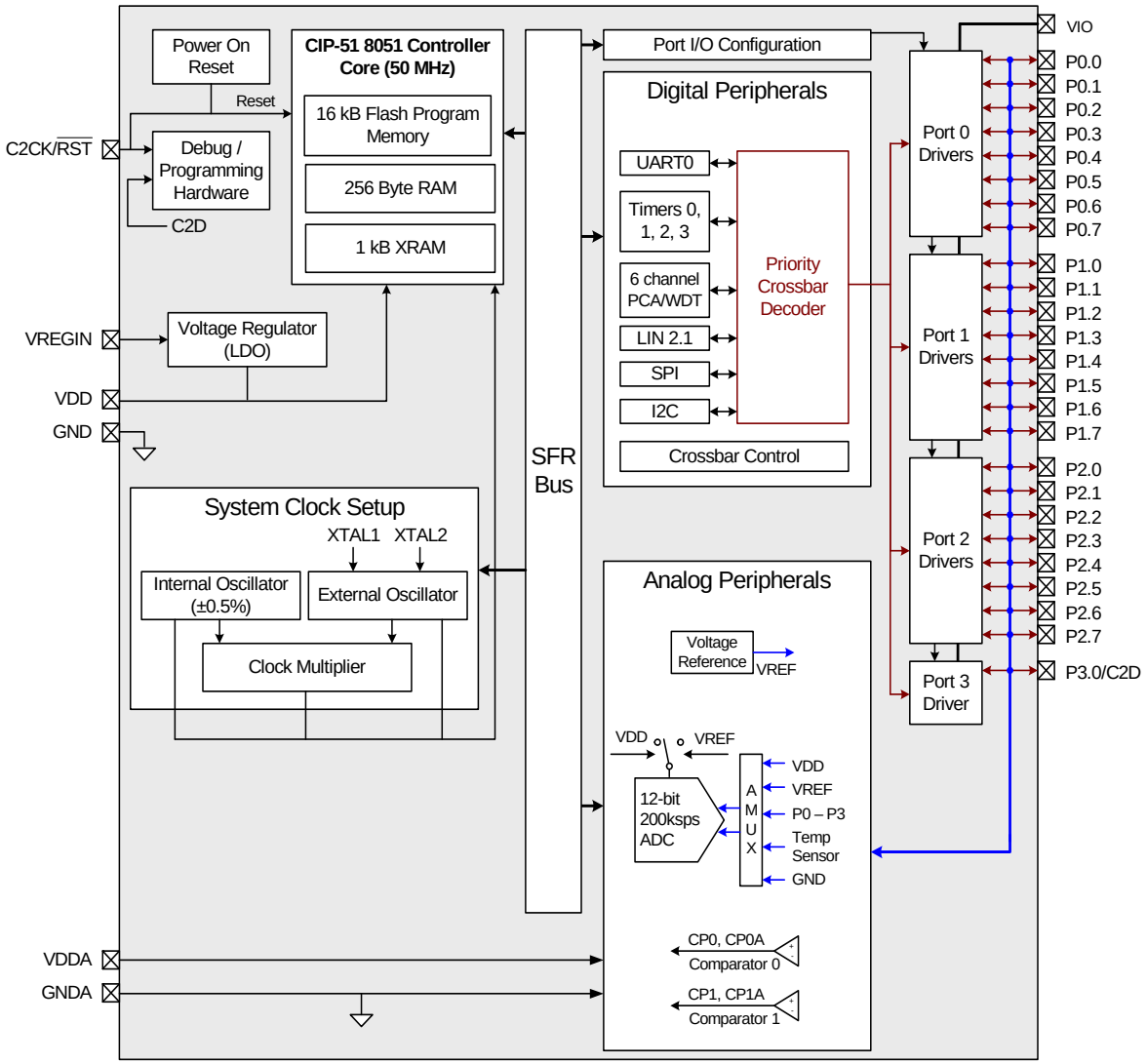

Figure 1.1. C8051F540/1/4/5 Block Diagram

Table 3.1. Pin Definitions for the C8051F54x (Continued)

Name	Pin ‘F540/1/4/5 (32-pin)	Pin ‘F542/3/6/7 (24-pin)	Type	Description
P1.3	23	14	D I/O or A In	Port 1.3.
P1.4	22	13	D I/O or A In	Port 1.4.
P1.5	21	12	D I/O or A In	Port 1.5.
P1.6	20	11	D I/O or A In	Port 1.6.
P1.7	19	10	D I/O or A In	Port 1.7.
P2.0	18	9	D I/O or A In	Port 2.0. See SFR Definition 18.20 for a description.
P2.1	17	—	D I/O or A In	Port 2.1.
P2.2	16	—	D I/O or A In	Port 2.2.
P2.3	15	—	D I/O or A In	Port 2.3.
P2.4	14	—	D I/O or A In	Port 2.4.
P2.5	13	—	D I/O or A In	Port 2.5.
P2.6	12	—	D I/O or A In	Port 2.6.
P2.7	11	—	D I/O or A In	Port 2.7.

8.1. Comparator Multiplexer

C8051F54x devices include an analog input multiplexer for each of the comparators to connect Port I/O pins to the comparator inputs. The Comparator0 inputs are selected in the CPT0MX register (SFR Definition 8.5). The CMX0P3–CMX0P0 bits select the Comparator0 positive input; the CMX0N3–CMX0N0 bits select the Comparator0 negative input. Similarly, the Comparator1 inputs are selected in the CPT1MX register using the CMX1P3-CMX1P0 bits and CMX1N3-CMX1N0 bits. The same pins are available to both multiplexers at the same time and can be used by both comparators simultaneously.

Important Note About Comparator Inputs: The Port pins selected as comparator inputs should be configured as analog inputs in their associated Port configuration register, and configured to be skipped by the Crossbar (for details on Port configuration, see Section “18.6. Special Function Registers for Accessing and Configuring Port I/O” on page 161).

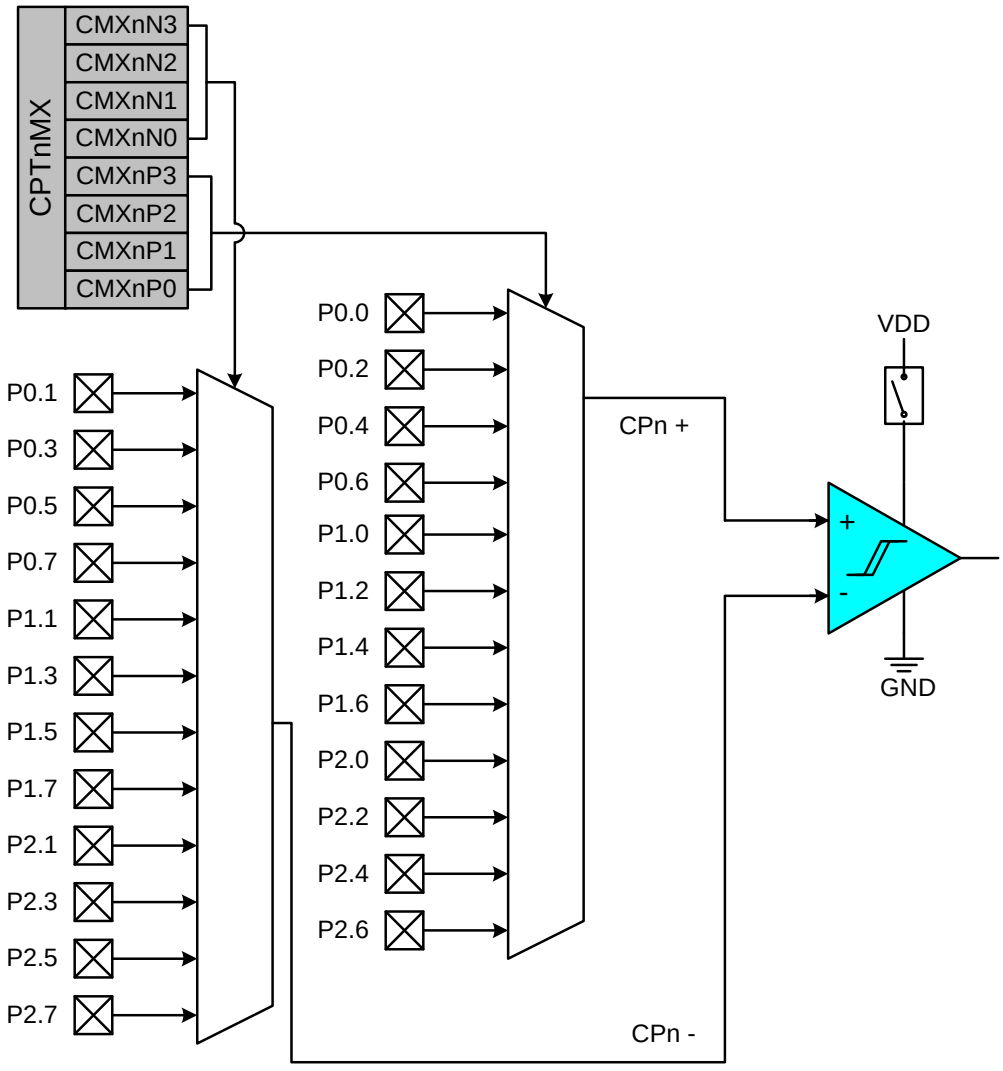

Figure 8.3. Comparator Input Multiplexer Block Diagram

10. CIP-51 Microcontroller

The MCU system controller core is the CIP-51 microcontroller. The CIP-51 is fully compatible with the MCS-51™ instruction set; standard 803x/805x assemblers and compilers can be used to develop software. The MCU family has a superset of all the peripherals included with a standard 8051. The CIP-51 also includes on-chip debug hardware (see description in Section 25), and interfaces directly with the analog and digital subsystems providing a complete data acquisition or control-system solution in a single integrated circuit.

The CIP-51 Microcontroller core implements the standard 8051 organization and peripherals as well as additional custom peripherals and functions to extend its capability (see Figure 10.1 for a block diagram). The CIP-51 includes the following features:

- Fully Compatible with MCS-51 Instruction Set
- 50 MIPS Peak Throughput with 50 MHz Clock
- 0 to 50 MHz Clock Frequency
- Extended Interrupt Handler
- Reset Input
- Power Management Modes
- On-chip Debug Logic
- Program and Data Memory Security

10.1. Performance

The CIP-51 employs a pipelined architecture that greatly increases its instruction throughput over the standard 8051 architecture. In a standard 8051, all instructions except for MUL and DIV take 12 or 24 system clock cycles to execute, and usually have a maximum system clock of 12 MHz. By contrast, the CIP-51 core executes 70% of its instructions in one or two system clock cycles, with no instructions taking more than eight system clock cycles.

11. Memory Organization

The memory organization of the CIP-51 System Controller is similar to that of a standard 8051. There are two separate memory spaces: program memory and data memory. Program and data memory share the same address space but are accessed via different instruction types. The memory organization is shown in Figure 11.1

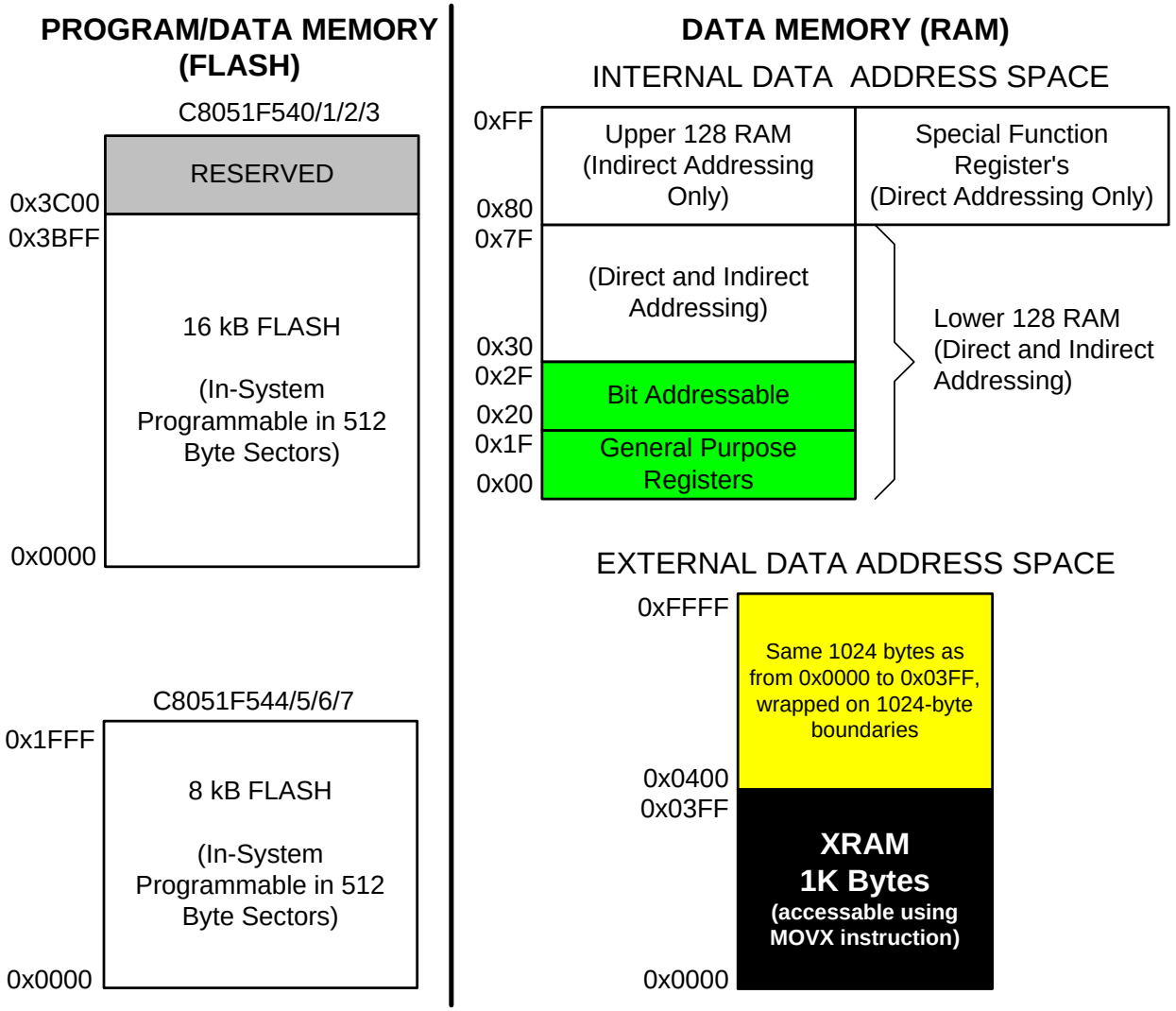

Figure 11.1. C8051F54x Memory Map

11.1. Program Memory

The CIP-51 core has a 64 kB program memory space. The C8051F54x devices implement 16 kB or 8 kB of this program memory space as in-system, re-programmable Flash memory, organized in a contiguous block from addresses 0x0000 to 0x3FFF in 16 kB devices and addresses 0x0000 to 0x1FFF in 8 kB devices. The address 0x3BFF in 16 kB devices and 0x1FFF in 8 kB devices serves as the security lock byte for the device. Addresses above 0x3BFF are reserved in the 16 kB devices.

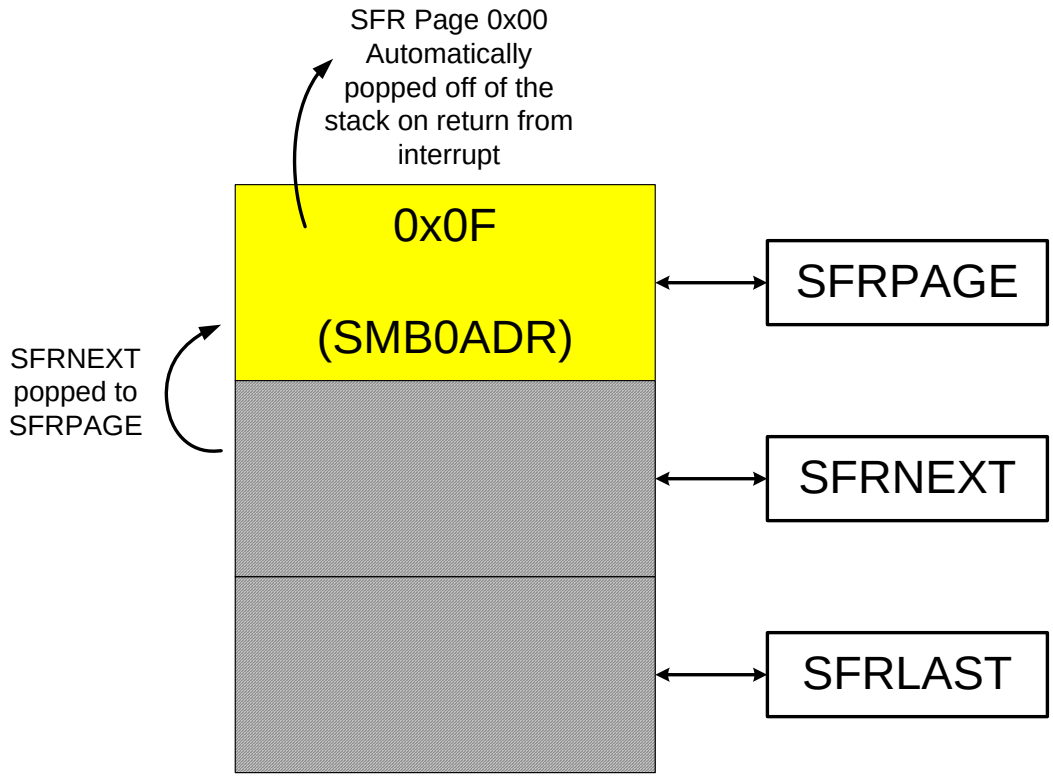

Figure 12.6. SFR Page Stack Upon Return From SPI0 Interrupt

In the example above, all three bytes in the SFR Page Stack are accessible via the SFRPAGE, SFRNEXT, and SFRLAST special function registers. If the stack is altered while servicing an interrupt, it is possible to return to a different SFR Page upon interrupt exit than selected prior to the interrupt call. Direct access to the SFR Page stack can be useful to enable real-time operating systems to control and manage context switching between multiple tasks.

Push operations on the SFR Page Stack only occur on interrupt service, and pop operations only occur on interrupt exit (execution on the RETI instruction). The automatic switching of the SFRPAGE and operation of the SFR Page Stack as described above can be disabled in software by clearing the SFR Automatic Page Enable Bit (SFRPGEN) in the SFR Page Control Register (SFR0CN). See SFR Definition 12.1.

SFR Definition 12.2. SFRPAGE: SFR Page

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

SFR Address = 0xA7; SFR Page = All Pages

Bit	Name	Function
7:0	SFRPAGE[7:0]	**SFR Page Bits.** Represents the SFR Page the C8051 core uses when reading or modifying SFRs. Write: Sets the SFR Page. Read: Byte is the SFR page the C8051 core is using. When enabled in the SFR Page Control Register (SFR0CN), the C8051 core will automatically switch to the SFR Page that contains the SFRs of the corresponding peripheral/function that caused the interrupt, and return to the previous SFR page upon return from interrupt (unless SFR Stack was altered before a returning from the interrupt). SFRPAGE is the top byte of the SFR Page Stack, and push/pop events of this stack are caused by interrupts (and not by reading/writing to the SFRPAGE register)

Table 12.1. Special Function Register (SFR) Memory Map for Pages 0x0 and 0xF

Address	Page	0(8)	1(9)	2(A)	3(B)	4(C)	5(D)	6(E)	7(F)
F8	0F	SPI0CN	PCA0L SN0	PCA0H SN1	PCA0CPL0 SN2	PCA0CPH0 SN3	PCACPL4	PCACPH4	VDM0CN
F0	0F	B (All Pages)	P0MAT P0MDIN	P0MASK P1MDIN	P1MAT P2MDIN	P1MASK P3MDIN		EIP1 EIP1	EIP2 EIP2
E8	0F	ADC0CN	PCA0CPL1	PCA0CPH1	PCA0CPL2	PCA0CPH2	PCA0CPL3	PCA0CPL3	RSTSRC
E0	0F	ACC (All Pages)	XBR0	XBR1	CCH0CN	IT01CF		EIE1 (All Pages)	EIE2 (All Pages)
D8	0F	PCA0CN	PCA0MD PCA0PWM	PCA0CPM0	PCA0CPM1	PCA0CPM2	PCA0CPM3	PCA0CPM4	PCA0CPM5
D0	0F	PSW (All Pages)	REF0CN	LIN0DATA	LIN0ADDR	P0SKIP	P1SKIP	P2SKIP	P3SKIP
C8	0F	TMR2CN	REG0CN LIN0CF	TMR2RLL	TMR2RLH	TMR2L	TMR2H	PCA0CPL5	PCA0CPH5
C0	0F	SMB0CN	SMB0CF	SMB0DAT	ADC0GTL	ADC0GTH	ADC0LTL	ADC0LTH	XBR2
B8	0F	IP (All Pages)		ADC0TK	ADC0MX	ADC0CF	ADC0L	ADC0H	
B0	0F	P3 (All Pages)	P2MAT	P2MASK				FLSCL (All Pages)	FLKEY (All Pages)
A8	0F	IE (All Pages)	SMOD0	EMI0CN	SBCON0	SBRLL0	SBRLH0	P3MAT P3MDOUT	P3MASK
A0	0F	P2 (All Pages)	SPI0CFG OSCICN	SPI0CKR OSCICRS	SPI0DAT	P0MDOUT	P1MDOUT	P2MDOUT	SFRPAGE (All Pages)
98	0F	SCON0	SBUF0	CPT0CN	CPT0MD	CPT0MX	CPT1CN	CPT1MD OSCIFIN	CPT1MX OSCXCN
90	0F	P1 (All Pages)	TMR3CN	TMR3RLL	TMR3RLH	TMR3L	TMR3H		CLKMUL
88	0F	TCON (All Pages)	TMOD (All Pages)	TL0 (All Pages)	TL1 (All Pages)	TH0 (All Pages)	TH1 (All Pages)	CKCON (All Pages)	PSCTL CLKSEL
80	0F	P0 (All Pages)	SP (All Pages)	DPL (All Pages)	DPH (All Pages)	SFR0CN	SFRNEXT (All Pages)	SFRLAST (All Pages)	PCON (All Pages)
		0(8)	1(9)	2(A)	3(B)	4(C)	5(D)	6(E)	7(F)

(bit addressable)

Table 12.2. Special Function Registers

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

Register	Address	Description	Page
ACC	0xE0	Accumulator	82
ADC0CF	0xBC	ADC0 Configuration	40
ADC0CN	0xE8	ADC0 Control	42
ADC0GTH	0xC4	ADC0 Greater-Than Compare High	44
ADC0GTL	0xC3	ADC0 Greater-Than Compare Low	44
ADC0H	0xBE	ADC0 High	41
ADC0L	0xBD	ADC0 Low	41
ADC0LTH	0xC6	ADC0 Less-Than Compare Word High	45
ADC0LTL	0xC5	ADC0 Less-Than Compare Word Low	45
ADC0MX	0xBB	ADC0 Mux Configuration	59
ADC0TK	0xBA	ADC0 Tracking Mode Select	43
B	0xF0	B Register	82
CCH0CN	0xE3	Cache Control	125
CKCON	0x8E	Clock Control	228
CLKMUL	0x97	Clock Multiplier	141
CLKSEL	0x8F	Clock Select	136
CPT0CN	0x9A	Comparator0 Control	65
CPT0MD	0x9B	Comparator0 Mode Selection	66
CPT0MX	0x9C	Comparator0 MUX Selection	70
CPT1CN	0x9D	Comparator1 Control	65
CPT1MD	0x9E	Comparator1 Mode Selection	66
CPT1MX	0x9F	Comparator1 MUX Selection	70
DPH	0x83	Data Pointer High	81
DPL	0x82	Data Pointer Low	81
EIE1	0xE6	Extended Interrupt Enable 1	111
EIE2	0xE7	Extended Interrupt Enable 2	111
EIP1	0xF6	Extended Interrupt Priority 1	112
EIP2	0xF7	Extended Interrupt Priority 2	113
EMI0CN	0xAA	External Memory Interface Control	88
FLKEY	0xB7	Flash Lock and Key	123
FLSCL	0xB6	Flash Scale	124
IE	0xA8	Interrupt Enable	109
IP	0xB8	Interrupt Priority	110
IT01CF	0xE4	INT0/INT1 Configuration	116
LIN0ADR	0xD3	LIN0 Address	177

SFR Definition 17.1. CLKSEL: Clock Select

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

SFR Address = 0x8F; SFR Page = 0x0F;

Bit	Name	Function
7:2	Unused	Read = 000000b; Write = Don't Care
1:0	CLKSL[1:0]	**System Clock Source Select Bits.** 00: SYSCLK derived from the Internal Oscillator and scaled per the IFCN bits in register OSCICN. 01: SYSCLK derived from the External Oscillator circuit. 10: SYSCLK derived from the Clock Multiplier. 11: reserved.

SFR Definition 18.2. XBR1: Port I/O Crossbar Register 1

Bit	7	6	5	4	3	2	1	0
Name	T1E	T0E	ECIE	PCA0ME[2:0]			SYSCKE	Reserved
Type	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xE2; SFR Page = 0x0F

Bit	Name	Function
7	T1E	**T1 Enable.** 0: T1 unavailable at Port pin. 1: T1 routed to Port pin.
6	T0E	**T0 Enable.** 0: T0 unavailable at Port pin. 1: T0 routed to Port pin.
5	ECIE	**PCA0 External Counter Input Enable.** 0: ECI unavailable at Port pin. 1: ECI routed to Port pin.
4:2	PCA0ME[2:0]	**PCA Module I/O Enable Bits.** 000: All PCA I/O unavailable at Port pins. 001: CEX0 routed to Port pin. 010: CEX0, CEX1 routed to Port pins. 011: CEX0, CEX1, CEX2 routed to Port pins. 100: CEX0, CEX1, CEX2, CEX3 routed to Port pins. 101: CEX0, CEX1, CEX2, CEX3, CEX4 routed to Port pins. 110: CEX0, CEX1, CEX2, CEX3, CEX4, CEX5 routed to Port pins. 111: RESERVED
1	SYSCKE	**$\overline{\text{SYSCLK}}$ Output Enable.** 0: $\overline{\text{SYSCLK}}$ unavailable at Port pin. 1: $\overline{\text{SYSCLK}}$ output routed to Port pin.
0	Reserved	Always Write to 0.

Table 19.3. Autobaud Parameters Examples

System Clock (MHz)	Prescaler	Divider
25	1	312
24.5	1	306
24	1	300
22.1184	1	276
16	1	200
12.25	0	306
12	0	300
11.0592	0	276
8	0	200

19.3. LIN Master Mode Operation

The master node is responsible for the scheduling of messages and sends the header of each frame containing the SYNCH BREAK FIELD, SYNCH FIELD, and IDENTIFIER FIELD. The steps to schedule a message transmission or reception are listed below.

1. Load the 6-bit Identifier into the LIN0ID register.
2. Load the data length into the LIN0SIZE register. Set the value to the number of data bytes or "1111b" if the data length should be decoded from the identifier. Also, set the checksum type, classic or enhanced, in the same LIN0SIZE register.
3. Set the data direction by setting the TXRX bit (LIN0CTRL.5). Set the bit to 1 to perform a master transmit operation, or set the bit to 0 to perform a master receive operation.
4. If performing a master transmit operation, load the data bytes to transmit into the data buffer (LIN0DT1 to LIN0DT8).
5. Set the STREQ bit (LIN0CTRL.0) to start the message transfer. The LIN controller will schedule the message frame and request an interrupt if the message transfer is successfully completed or if an error has occurred.

This code segment shows the procedure to schedule a message in a transmission operation:

```
LIN0ADR   = 0x08;               // Point to LIN0CTRL
LIN0DAT  |= 0x20;               // Select to transmit data
LIN0ADR   = 0x0E;               // Point to LIN0ID
LIN0DAT   = 0x11;               // Load the ID, in this example 0x11
LIN0ADR   = 0x0B;               // Point to LIN0SIZE
LIN0DAT  = ( LIN0DAT & 0xF0 ) | 0x08;      // Load the size with 8

LIN0ADR  = 0x00;               // Point to Data buffer first byte
for (i=0; i<8; i++)
{
   LIN0DAT = i + 0x41;         // Load the buffer with ‘A’, ‘B’, ...
   LIN0ADR++;                  // Increment the address to the next buffer
}
LIN0ADR  = 0x08;               // Point to LIN0CTRL
LIN0DAT  = 0x01;               // Start Request
```

The application should perform the following steps when an interrupt is requested.

3. The LIN controller does not directly support LIN Version 1.3 Extended Frames. If the application detects an unknown identifier (e.g. extended identifier), it has to write a 1 to the STOP bit (LIN0CTRL.7) instead of setting the DTACK (LIN0CTRL.4) bit. At that time, steps 2 through 5 can then be skipped. In this situation, the LIN controller stops the processing of LIN communication until the next SYNC BREAK is received.
4. Changing the configuration of the checksum during a transaction will cause the interface to reset and the transaction to be lost. To prevent this, the checksum should not be configured while a transaction is in progress. The same applies to changes in the LIN interface mode from slave mode to master mode and from master mode to slave mode.

19.5. Sleep Mode and Wake-Up

To reduce the system's power consumption, the LIN Protocol Specification defines a Sleep Mode. The message used to broadcast a Sleep Mode request must be transmitted by the LIN master application in the same way as a normal transmit message. The LIN slave application must decode the Sleep Mode Frame from the Identifier and data bytes. After that, it has to put the LIN slave node into the Sleep Mode by setting the SLEEP bit (LIN0CTRL.6).

If the SLEEP bit (LIN0CTRL.6) of the LIN slave application is not set and there is no bus activity for four seconds (specified bus idle timeout), the IDLTOUT bit (LIN0ST.6) is set and an interrupt request is generated. After that the application may assume that the LIN bus is in Sleep Mode and set the SLEEP bit (LIN0CTRL.6).

Sending a wake-up signal from the master or any slave node terminates the Sleep Mode of the LIN bus. To send a wake-up signal, the application has to set the WUPREQ bit (LIN0CTRL.1). After successful transmission of the wake-up signal, the DONE bit (LIN0ST.0) of the master node is set and an interrupt request is generated. The LIN slave does not generate an interrupt request after successful transmission of the wake-up signal but it generates an interrupt request if the master does not respond to the wake-up signal within 150 milliseconds. In that case, the ERROR bit (LIN0ST.2) and TOUT bit (LIN0ERR.2) are set. The application then has to decide whether or not to transmit another wake-up signal.

All LIN nodes that detect a wake-up signal will set the WAKEUP (LIN0ST.1) and DONE bits (LIN0ST.0) and generate an interrupt request. After that, the application has to clear the SLEEP bit (LIN0CTRL.6) in the LIN slave.

19.6. Error Detection and Handling

The LIN controller generates an interrupt request and stops the processing of the current frame if it detects an error. The application has to check the type of error by processing LIN0ERR. After that, it has to reset the error register and the ERROR bit (LIN0ST.2) by writing a 1 to the RSTERR bit (LIN0CTRL.2). Starting a new message with the LIN controller selected as master or sending a Wakeup signal with the LIN controller selected as a master or slave is possible only if the ERROR bit (LIN0ST.2) is set to 0.

All transactions are initiated by a master, with one or more addressed slave devices as the target. The master generates the START condition and then transmits the slave address and direction bit. If the transaction is a WRITE operation from the master to the slave, the master transmits the data a byte at a time waiting for an ACK from the slave at the end of each byte. For READ operations, the slave transmits the data waiting for an ACK from the master at the end of each byte. At the end of the data transfer, the master generates a STOP condition to terminate the transaction and free the bus. Figure 20.3 illustrates a typical SMBus transaction.

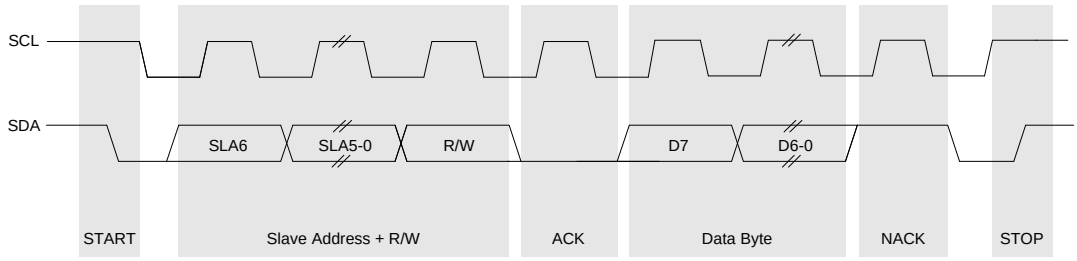

Figure 20.3. SMBus Transaction

20.3.1. Transmitter Vs. Receiver

On the SMBus communications interface, a device is the “transmitter” when it is sending an address or data byte to another device on the bus. A device is a “receiver” when an address or data byte is being sent to it from another device on the bus. The transmitter controls the SDA line during the address or data byte. After each byte of address or data information is sent by the transmitter, the receiver sends an ACK or NACK bit during the ACK phase of the transfer, during which time the receiver controls the SDA line.

20.3.2. Arbitration

A master may start a transfer only if the bus is free. The bus is free after a STOP condition or after the SCL and SDA lines remain high for a specified time (see Section “20.3.5. SCL High (SMBus Free) Timeout” on page 190). In the event that two or more devices attempt to begin a transfer at the same time, an arbitration scheme is employed to force one master to give up the bus. The master devices continue transmitting until one attempts a HIGH while the other transmits a LOW. Since the bus is open-drain, the bus will be pulled LOW. The master attempting the HIGH will detect a LOW SDA and lose the arbitration. The winning master continues its transmission without interruption; the losing master becomes a slave and receives the rest of the transfer if addressed. This arbitration scheme is non-destructive: one device always wins, and no data is lost.

20.3.3. Clock Low Extension

SMBus provides a clock synchronization mechanism, similar to I^2C, which allows devices with different speed capabilities to coexist on the bus. A clock-low extension is used during a transfer in order to allow slower slave devices to communicate with faster masters. The slave may temporarily hold the SCL line LOW to extend the clock low period, effectively decreasing the serial clock frequency.

20.3.4. SCL Low Timeout

If the SCL line is held low by a slave device on the bus, no further communication is possible. Furthermore, the master cannot force the SCL line high to correct the error condition. To solve this problem, the SMBus protocol specifies that devices participating in a transfer must detect any clock cycle held low longer than 25 ms as a “timeout” condition. Devices that have detected the timeout condition must reset the communication no later than 10 ms after detecting the timeout condition.

When the SMBTOE bit in SMB0CF is set, Timer 3 is used to detect SCL low timeouts. Timer 3 is forced to reload when SCL is high, and allowed to count when SCL is low. With Timer 3 enabled and configured to

20.5.4. Read Sequence (Slave)

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

If the received slave address is ignored, slave interrupts will be inhibited until the next START is detected. If the received slave address is acknowledged, zero or more data bytes are transmitted. If the received slave address is acknowledged, data should be written to SMB0DAT to be transmitted. The interface enters Slave Transmitter Mode, and transmits one or more bytes of data. After each byte is transmitted, the master sends an acknowledge bit; if the acknowledge bit is an ACK, SMB0DAT should be written with the next data byte. If the acknowledge bit is a NACK, SMB0DAT should not be written to before SI is cleared (Note: an error condition may be generated if SMB0DAT is written following a received NACK while in Slave Transmitter Mode). The interface exits Slave Transmitter Mode after receiving a STOP. Note that the interface will switch to Slave Receiver Mode if SMB0DAT is not written following a Slave Transmitter interrupt. Figure 20.8 shows a typical slave read sequence. Two transmitted data bytes are shown, though any number of bytes may be transmitted. Notice that all of the ‘data byte transferred’ interrupts occur **after** the ACK cycle in this mode.

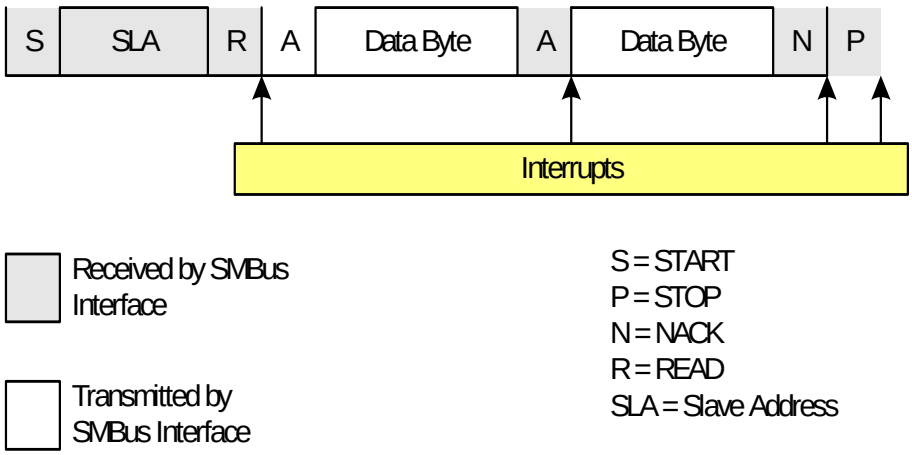

Figure 20.8. Typical Slave Read Sequence

20.6. SMBus Status Decoding

The current SMBus status can be easily decoded using the SMB0CN register. In the tables, STATUS VECTOR refers to the four upper bits of SMB0CN: MASTER, TXMODE, STA, and STO. The shown response options are only the typical responses; application-specific procedures are allowed as long as they conform to the SMBus specification. Highlighted responses are allowed by hardware but do not conform to the SMBus specification.

21.2. Data Format

UART0 has a number of available options for data formatting. Data transfers begin with a start bit (logic low), followed by the data bits (sent LSB-first), a parity or extra bit (if selected), and end with one or two stop bits (logic high). The data length is variable between 5 and 8 bits. A parity bit can be appended to the data, and automatically generated and detected by hardware for even, odd, mark, or space parity. The stop bit length is selectable between 1 and 2 bit times, and a multi-processor communication mode is available for implementing networked UART buses. All of the data formatting options can be configured using the SMOD0 register, shown in SFR Definition 21.2. Figure 21.2 shows the timing for a UART0 transaction without parity or an extra bit enabled. Figure 21.3 shows the timing for a UART0 transaction with parity enabled (PE0 = 1). Figure 21.4 is an example of a UART0 transaction when the extra bit is enabled (XBE0 = 1). Note that the extra bit feature is not available when parity is enabled, and the second stop bit is only an option for data lengths of 6, 7, or 8 bits.

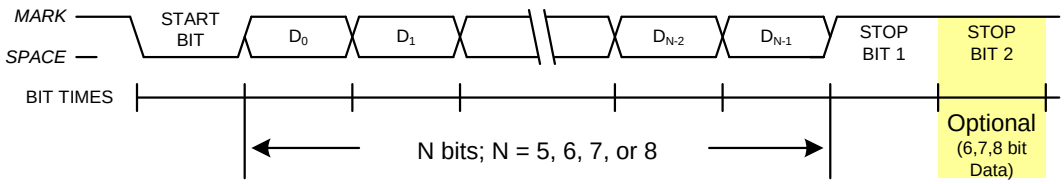

Figure 21.2. UART0 Timing Without Parity or Extra Bit

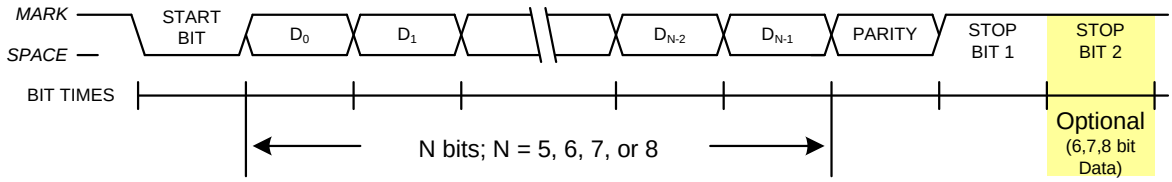

Figure 21.3. UART0 Timing With Parity

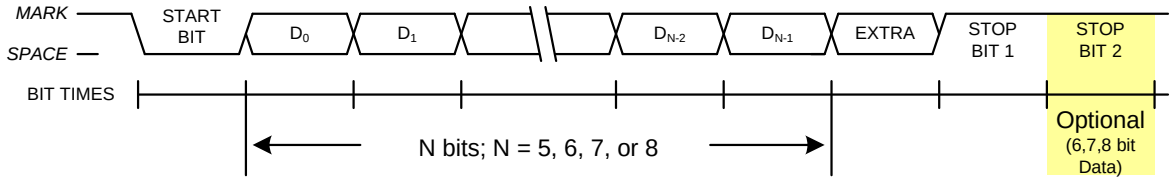

Figure 21.4. UART0 Timing With Extra Bit

This mode allows software to determine the external oscillator frequency when an RC network or capacitor is used to generate the clock source.

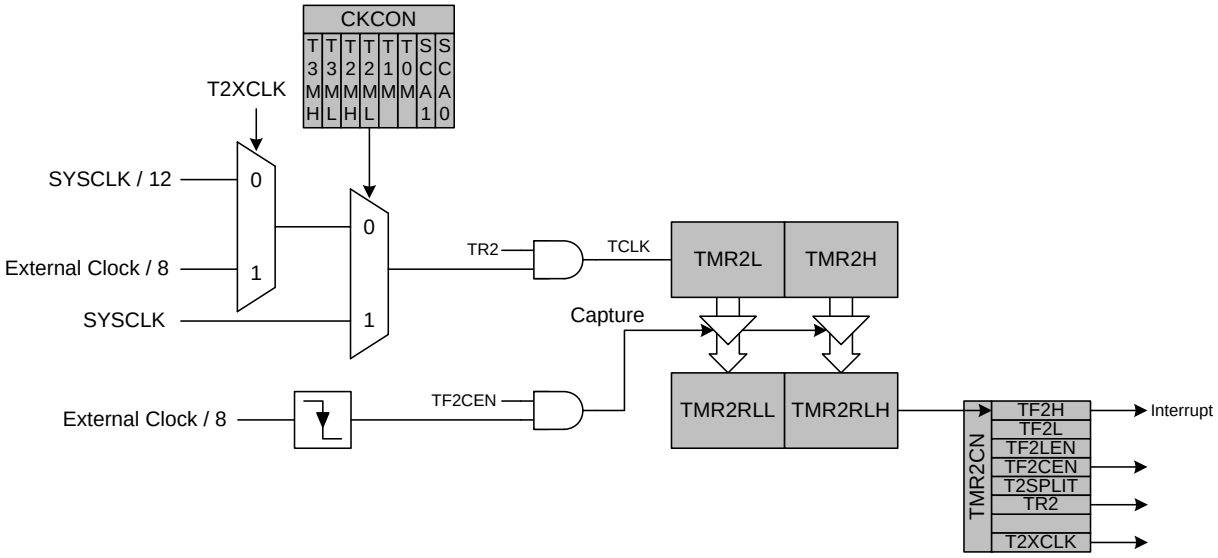

Figure 23.6. Timer 2 External Oscillator Capture Mode Block Diagram

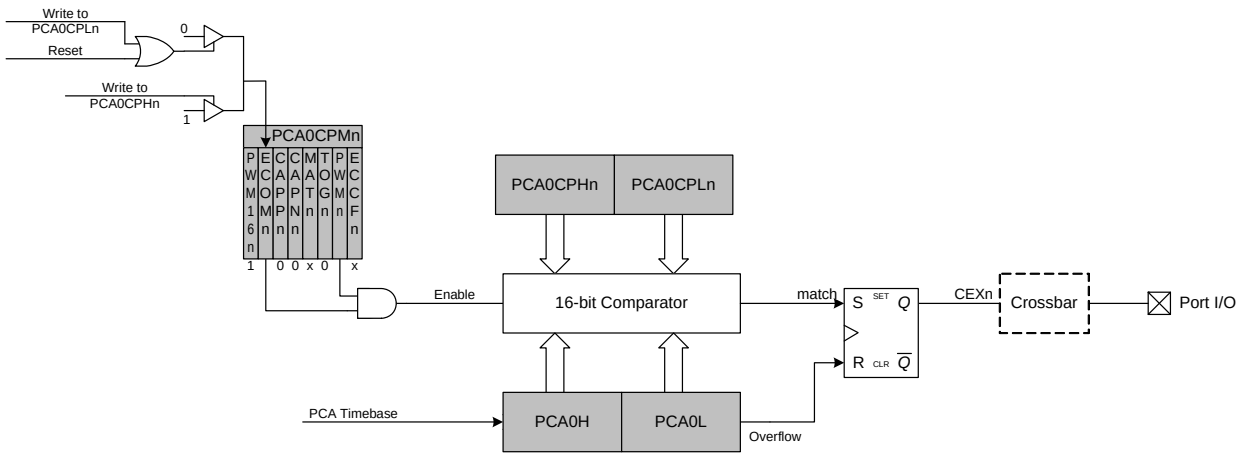

Figure 24.10. PCA 16-Bit PWM Mode

24.4. Watchdog Timer Mode

A programmable watchdog timer (WDT) function is available through the PCA Module 5. The WDT is used to generate a reset if the time between writes to the WDT update register (PCA0CPH5) exceed a specified limit. The WDT can be configured and enabled/disabled as needed by software.

With the WDTE bit set in the PCA0MD register, Module 5 operates as a watchdog timer (WDT). The Module 5 high byte is compared to the PCA counter high byte; the Module 5 low byte holds the offset to be used when WDT updates are performed. **The Watchdog Timer is enabled on reset. Writes to some PCA registers are restricted while the Watchdog Timer is enabled.** The WDT will generate a reset shortly after code begins execution. To avoid this reset, the WDT should be explicitly disabled (and optionally re-configured and re-enabled if it is used in the system).

24.4.1. Watchdog Timer Operation

While the WDT is enabled:

- PCA counter is forced on.
- Writes to PCA0L and PCA0H are not allowed.
- PCA clock source bits (CPS[2:0]) are frozen.
- PCA Idle control bit (CIDL) is frozen.
- Module 5 is forced into software timer mode.
- Writes to the Module 5 mode register (PCA0CPM5) are disabled.

While the WDT is enabled, writes to the CR bit will not change the PCA counter state; the counter will run until the WDT is disabled. The PCA counter run control bit (CR) will read zero if the WDT is enabled but user software has not enabled the PCA counter. If a match occurs between PCA0CPH5 and PCA0H while the WDT is enabled, a reset will be generated. To prevent a WDT reset, the WDT may be updated with a write of any value to PCA0CPH5. Upon a PCA0CPH5 write, PCA0H plus the offset held in PCA0CPL5 is loaded into PCA0CPH5 (See Figure 24.11).

DOCUMENT CHANGE LIST

Revision 0.1 to Revision 1.0

- Updated “2. Ordering Information” to include -A (Automotive) devices and automotive qualification information.
- Updated Figure 4.6.
- Updated supply current related specifications throughout “6. Electrical Characteristics” .
- Updated SFR Definition 7.1 (REF0CN) to change VREF high setting to 2.20 V from 2.25 V.
- Updated Figure 8.1 to indicate that Comparators are powered from V_{IO} and not V_{DDA}.
- Updated the Gain Table in “5.3.1. Calculating the Gain Value” to fix the ADC0GNH Value in the last row.
- Updated Table 10.1 with correct timing for all branch instructions, MOVC, and CPL A.
- Updated Table 14.1 to indicate behavior when performing a Flash operation in reserved space.
- Updated “14.1. Programming the Flash Memory” to clarify behavior of 8-bit MOVX instructions and when writing/erasing Flash.
- Updated SFR Definition 14.3 (FLSCL) to include FLEWT bit definition. This bit must be set before writing or erasing Flash. Also updated Table 6.5 to reflect new Flash Write and Erase timing.
- Updated “16.7. Flash Error Reset” with an additional cause of a Flash Error reset.
- Updated “18.1.3. Interfacing Port I/O in a Multi-Voltage System” to remove note regarding interfacing to voltages above VIO.
- Updated “20. SMBus” to remove all hardware ACK features, including SMB0ADM and SMB0ADR SFRs.
- Updated SFR Definition 21.1(SCON0) to correct SFR Page to 0x00 from All Pages.
- Updated CP Register Definition 24.2 with proper Device ID.

Note: All items from the C8051F54x Errata dated November 5th, 2009 are incorporated into this data sheet.

Revision 1.0 to Revision 1.1

- Updated “1. System Overview” with a voltage range specification for the internal oscillator.
- Updated Figure 5.4, “12-Bit ADC Burst Mode Example With Repeat Count Set to 4,” on page 33 with new timing diagram when using CNVSTR pin.
- Updated Table 6.6, “Internal High-Frequency Oscillator Electrical Characteristics,” on page 53 with new conditions for the internal oscillator accuracy. The internal oscillator accuracy is dependent on the operating voltage range.
- Updated “6. Electrical Characteristics” to remove the internal oscillator curve across temperature diagram.
- Updated SFR Definition 7.1 (REF0CN) with oscillator suspend requirement for ZTCEN.
- Fixed incorrect cross references in “8. Comparators” .
- Updated SFR Definition 9.1 (REG0CN) with a new definition for Bit 6. The bit 6 reset value is 1b and must be written to 1b.
- Updated Figure 11.2, “Flash Program Memory Map,” on page 86 with correct address for start of lock byte page from 0x3900 to 0x3A00.
- Updated “15.3. Suspend Mode” with note regarding ZTCEN.
- Added Port 2 Event and Port 3 Event to wake-up sources in “17.2.1. Internal Oscillator Suspend Mode”
- Updated “19. Local Interconnect Network (LIN)” with a voltage range specification for the internal oscillator.
- Updated LIN Register Definitions for LIN0MUL and LIN0DIV to correct the reset value.
- Updated C2 Register Definitions 25.2 and 25.3 with correct C2 and SFR addresses.