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.

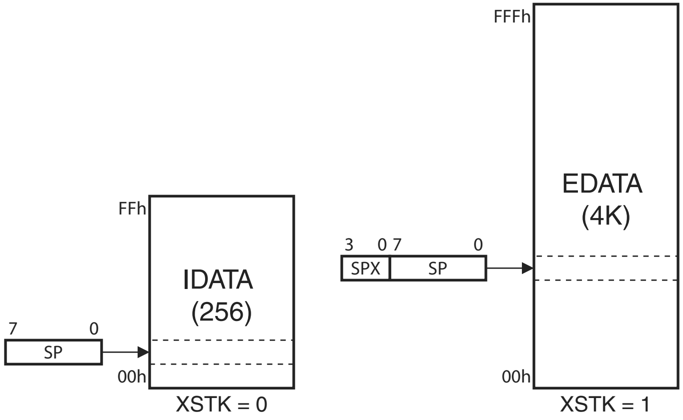
Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	8051
Core Size	8-Bit
Speed	20MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	38
Program Memory Size	32KB (32K x 8)
Program Memory Type	FLASH
EEPROM Size	8K x 8
RAM Size	4.25K x 8
Voltage - Supply (Vcc/Vdd)	2.4V ~ 3.6V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	44-TQFP
Supplier Device Package	44-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/at89lp3240-20au

tain separate copies of SP for use with each stack space. Interrupts should be disabled while swapping copies of SP in such an application to prevent illegal stack accesses.

All interrupt calls and PUSH, POP, ACALL, LCALL, RET and RETI instructions will incur a one or two-cycle penalty while the extended stack is enabled, depending on the number of stack access in each instruction. The extended stack may only exist within the internal EDATA space; it cannot be placed in XDATA. The stack will continue to use EDATA even if EDATA is disabled by setting EXRAM = 1.

Figure 3-14. Stack Configurations



3.5 In-Application Programming (IAP)

The AT89LP3240/6440 supports In-Application Programming (IAP), allowing the program memory to be modified during execution. IAP can be used to modify the user application on the fly or to use program memory for nonvolatile data storage. The same page structure write protocol for FDATA also applies to IAP (See Section 3.3.3.1 “Write Protocol” on page 14). The CPU is always placed in idle while modifying the program memory. When the write completes, the CPU will continue executing with the instruction after the MOVX @DPTR,A instruction that started the write.

To enable access to the program memory, the IAP bit (MEMCON.7) must be set to one and the IAP User Fuse must be enabled. The IAP User Fuse can disable all IAP operations. When this fuse is disabled, the IAP bit will be forced to 0. While IAP is enabled, all MOVX @DPTR instructions will access the CODE space instead of EDATA/FDATA/XDATA. IAP also allows reprogramming of the User Signature Array when SIGEN = 1. The IAP access settings are summarized in Table 3-5.

Table 3-5. IAP Access Settings

IAP	SIGEN	DMEN	MOVX @DPTR	MOVC @DPTR
0	0	0	EDATA (0000–0FFFH)	CODE (0000–FFFFH)
0	0	1	FDATA (1000–2FFFH)	CODE (0000–FFFFH)
0	1	0	EDATA (0000–0FFFH)	SIG (0000–01FFH)
0	1	1	FDATA (1000–2FFFH)	SIG (0000–01FFH)
1	0	X	CODE (0000–FFFFH)	CODE (0000–FFFFH)
1	1	X	SIG (0000–01FFH)	SIG (0000–01FFH)

5.3 Instruction Set Extensions

Table 5-8 lists the additions to the 8051 instruction set that are supported by the AT89LP3240/6440. For more information on the instruction set see Section 22. “Instruction Set Summary” on page 143. For detailed descriptions of the extended instructions see Section 22.1 “Instruction Set Extensions” on page 147.

Table 5-8. AT89LP3240/6440 Extended Instructions

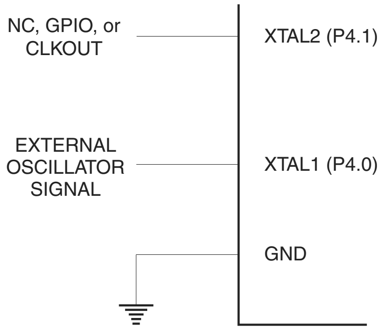
Opcode	Mnemonic	Description	Bytes	Cycles
A5 00	BREAK	Software breakpoint	2	2
A5 03	ASR M	Arithmetic shift right of M register	2	2
A5 23	LSL M	Logical shift left of M register	2	2
A5 73	JMP @A+PC	Indirect jump relative to PC	2	3
A5 90	MOV /DPTR, #data16	Move 16-bit constant to alternate data pointer	4	4
A5 93	MOVC A, @A+/DPTR	Move code location to ACC relative to alternate data pointer	2	4
A5 A3	INC /DPTR	Increment alternate data pointer	2	3
A5 A4	MAC AB	Multiply and accumulate	2	9
A5 B6	CJNE A, @R0, rel	Compare ACC to indirect RAM and jump if not equal	3	4
A5 B7	CJNE A, @R1, rel	Compare ACC to indirect RAM and jump if not equal	3	4
A5 E0	MOVX A, @/DPTR	Move external to ACC; 16-bit address in alternate data pointer	2	3/5
A5 E4	CLR M	Clear M register	2	2
A5 F0	MOVX @/DPTR, A	Move ACC to external; 16-bit address in alternate data pointer	2	3/5

- The /DPTR instructions provide support for the dual data pointer features described above (See Section 5.2).
- The ASR M, LSL M, CLR M and MAC AB instructions are part of the Multiply-Accumulate Unit (See Section 5.1).
- The JMP @A+PC instruction supports localized jump tables without using a data pointer.
- The CJNE A, @R$_i$, rel instructions allow compares of array values with non-constant values.
- The BREAK instruction is used by the On-Chip Debug system. See Section 24. on page 155.

6.2 External Clock Source

The external clock option disables the oscillator amplifier and allows XTAL1 to be driven directly by an external clock source as shown in Figure 6-2. XTAL2 may be left unconnected, used as general purpose I/O P4.1, or configured to output a divided version of the system clock.

Figure 6-2. External Clock Drive Configuration



6.3 Internal RC Oscillator

The AT89LP3240/6440 has an Internal RC oscillator (IRC) tuned to 8.0 MHz ±2.5%. When enabled as the clock source, XTAL1 and XTAL2 may be used as P4.0 and P4.1 respectively. XTAL2 may also be configured to output a divided version of the system clock. The frequency of the oscillator may be adjusted within limits by changing the RC Calibration Byte stored at byte 128 of the User Signature Array. This location may be updated using the IAP interface (location 0180H in SIG space) or by an external device programmer (UROW location 0080H). See Section 25.8 “User Signature and Analog Configuration” on page 165. A copy of the factory calibration byte is stored at byte 8 of the Atmel Signature Array (0008H in SIG space).

6.4 System Clock Out

When the AT89LP3240/6440 is configured to use either an external clock or the internal RC oscillator, the system clock divided by 2 may be output on XTAL2 (P4.1). The clock out feature is enabled by setting the COE bit in CLKREG. For example, setting COE = “1” when using the internal oscillator will result in a 4.0 MHz (±2.5%) clock output on P4.1. P4.1 must be configured as an output in order to use the clock out feature.

6.5 System Clock Divider

The $CDV_{2\text{-}0}$ bits in CLKREG allow the system clock to be divided down from the selected clock source by powers of 2. The clock divider provides users with a greater frequency range when using the Internal RC Oscillator. For example, to achieve a 1 MHz system frequency when using the IRC, $CDV_{2\text{-}0}$ should be set to 011B for divide-by-8 operation. The divider can also be used to reduce power consumption by decreasing the operational frequency during non-critical periods. The resulting system frequency is given by the following equation:

$$f_{SYS} = \frac{f_{OSC}}{2^{CDV}}$$

where f_{OSC} is the frequency of the selected clock source. The clock divider will prescale the clock for the CPU and all peripherals. The value of CDV may be changed at any time without interrupting normal execution. Changes to CDV are synchronized such that the system clock will not rupt system, the response time is always more than 5 clock cycles and less than 21 clock cycles. See Figure 9-1 and Figure 9-2.

Figure 9-1. Minimum Interrupt Response Time

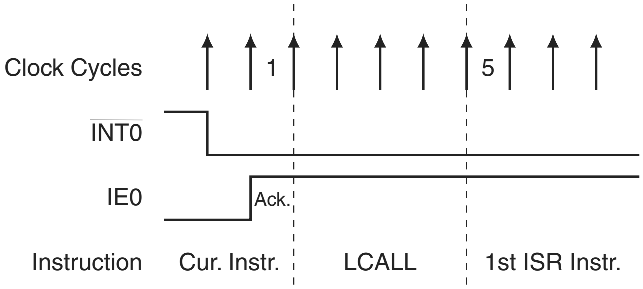


Figure 9-2. Maximum Interrupt Response Time

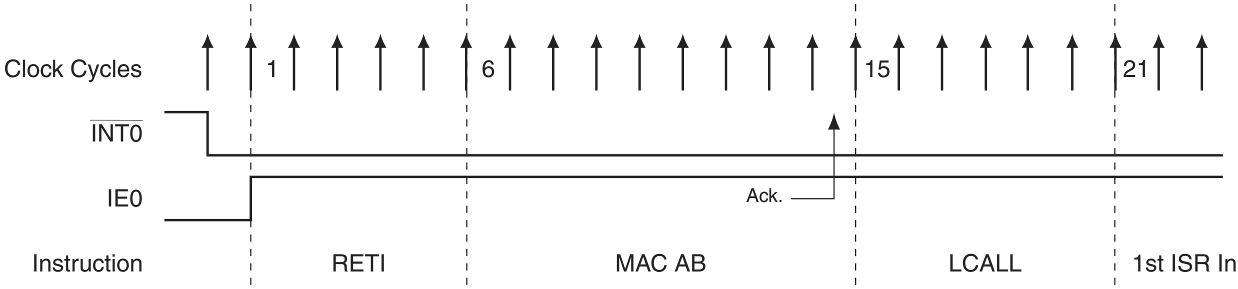


Table 9-2. IE – Interrupt Enable Register

IE = A8H								Reset Value = 0000 0000B
Bit Addressable								
	EA	EC	ET2	ES	ET1	EX1	ET0	EX0
Bit	7	6	5	4	3	2	1	0

Symbol	Function
EA	Global enable/disable. All interrupts are disabled when EA = 0. When EA = 1, each interrupt source is enabled/disabled by setting /clearing its own enable bit.
EC	Comparator Interrupt Enable
ET2	Timer 2 Interrupt Enable
ES	Serial Port Interrupt Enable
ET1	Timer 1 Interrupt Enable
EX1	External Interrupt 1 Enable
ET0	Timer 0 Interrupt Enable
EX0	External Interrupt 0 Enable

.

11.5.4 Mode 3 – Split 8-bit PWM

Timer 1 in PWM Mode 3 simply holds its count. The effect is the same as setting TR1 = 0. Timer 0 in PWM Mode 3 establishes TL0 and TH0 as two separate PWM counters in a manner similar to normal Mode 3. PWM Mode 3 on Timer 0 is shown in Figure 11-10. Only the Timer Prescaler is available to change the output frequency during PWM Mode 3. TL0 can use the Timer 0 control bits: GATE, TR0, $\overline{\text{INT0}}$, PWM0EN and TF0. TH0 is locked into a timer function and uses TR1, PWM1EN and TF1. RL0 provides the duty cycle for TL0 and RH0 provides the duty cycle for TH0.

PWM Mode 3 is for applications requiring a single PWM channel and two timers, or two PWM channels and an extra timer or counter. With Timer 0 in PWM Mode 3, the AT89LP3240/6440 can appear to have four Timer/Counters. When Timer 0 is in PWM Mode 3, Timer 1 can be turned on and off by switching it out of and into its own Mode 3. In this case, Timer 1 can still be used by the serial port as a baud rate generator or in any application not requiring an interrupt. The following formulas give the output frequency and duty cycle for Timer 0 in PWM Mode 3.

$$\text{Mode 3:} \qquad f_{out} = \frac{\text{Oscillator Frequency}}{256} \times \frac{1}{\text{TPS} + 1}$$

$$\text{Mode 3, T0:} \qquad \text{Duty Cycle \%} = 100 \times \frac{\text{RL0}}{256}$$

$$\text{Mode 3, T1:} \qquad \text{Duty Cycle \%} = 100 \times \frac{\text{RH0}}{256}$$

Figure 11-10. Timer/Counter 0 PWM Mode 3

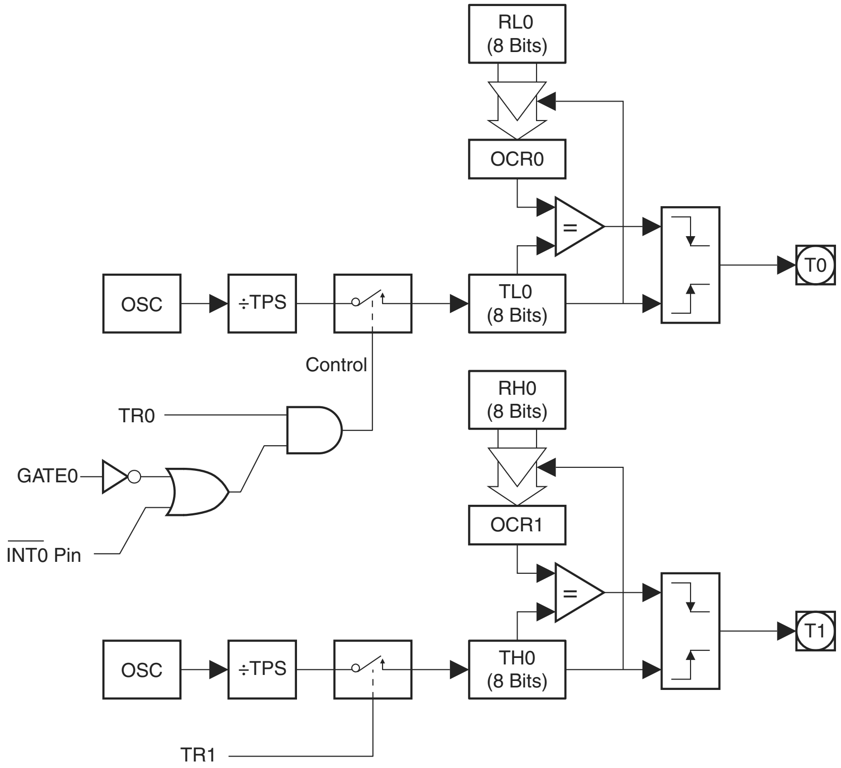
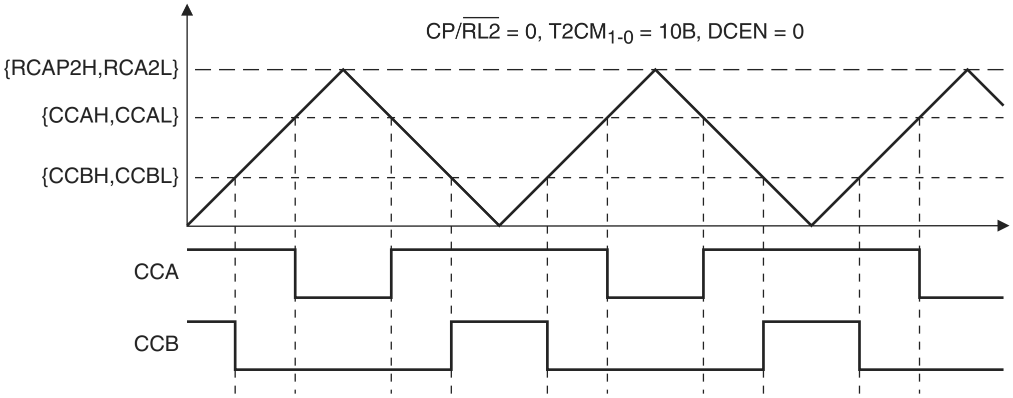


Figure 13-6. Dual-Slope Waveform Example



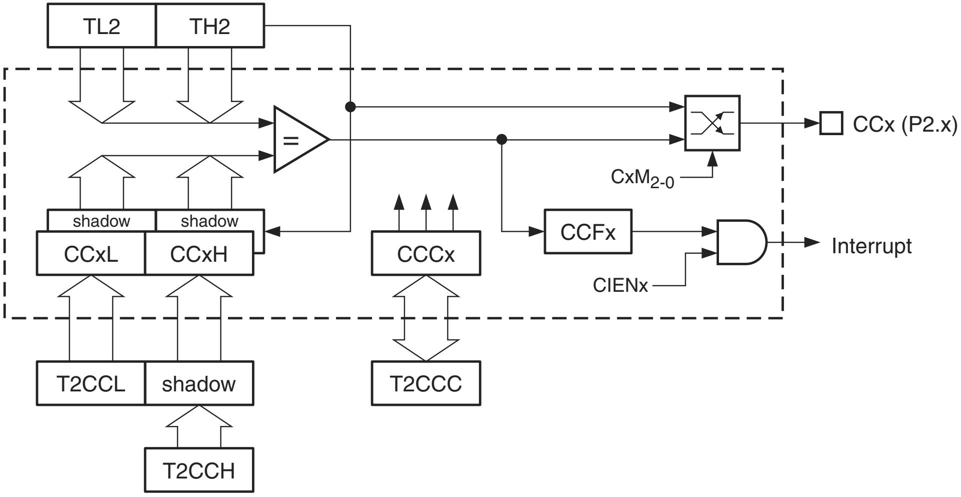
13.3.2 Timer 2 Operation for Compare Mode

Compare channels will work with any Timer 2 operating mode. The full 16-bit compare range may not be available in all modes. In order for a compare output action to take place, the compare values must be within the counting range of Timer 2. CTC*x* must be cleared to 0 for all channels if Timer 2 is operating in Baud Rate mode or errors may occur in the serial communication.

13.4 Pulse Width Modulation Mode

In Pulse Width Modulation (PWM) Mode, a compare channel can output a square wave with programmable frequency and duty cycle. Setting CCMx = 1 and CxM$_{2-0}$ = 10xB enables PWM Mode. PWM Mode is similar to Output Compare Mode except that the compare value is double-buffered. A diagram of a CCA channel in PWM Mode is shown in Figure 13-7. The PWM polarity is selectable between inverting and non-inverting modes. PWM is intended for use with Timer 2 in Auto-Reload Mode (CP/$\overline{RL2}$ = 0, DCEN = 0) using count modes 1, 2 or 3. The PWM can operate in asymmetric (edge-aligned) or symmetric (center-aligned) mode depending on the T2CM selection. The CCA PWM has variable precision from 2 to 16 bits. A trade-off between frequency and precision is made by changing the TOP value of the timer. The CCA PWM always uses the greatest precision allowable for the selected output frequency, as compared to Timer 0 and 1 whose PWMs are fixed at 8-bit precision regardless of frequency.

Figure 13-7. CCA PWM Mode Diagram



center-aligned around the timer equal to TOP point. Symmetrical PWM may be used to generate non-overlapping waveforms.

The TOP value in RCAP2L and RCAP2H is double buffered such that the output frequency is only updated at the underflow. The channel data register (CC*x*L, CC*x*H) is also double-buffered to prevent glitches. The output frequency and duty cycle for symmetrical PWM are given by the following equations:

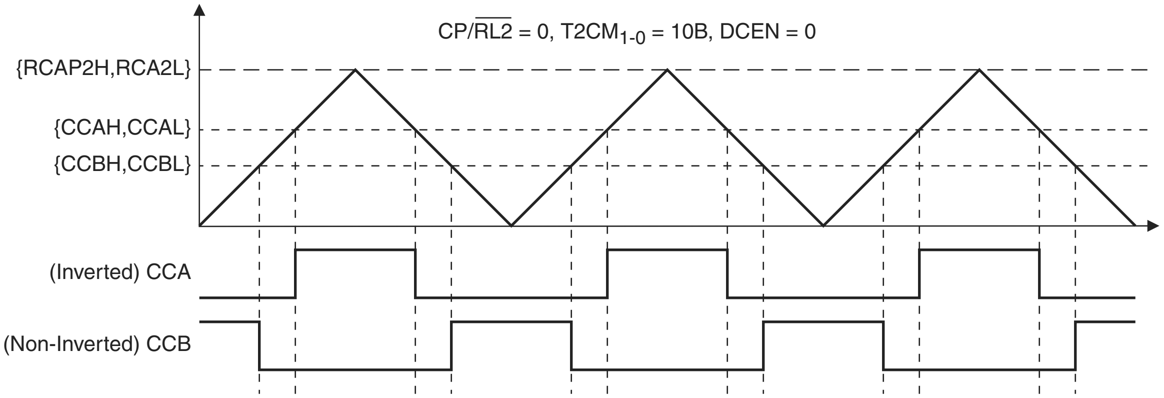
$$f_{OUT} = \frac{\text{Oscillator Frequency}}{2 \times \{\text{RCAP2H}, \text{RCAP2L}\}} \times \frac{1}{\text{TPS} + 1}$$

$$\text{Non-Inverting:} \qquad \text{Duty Cycle} = 100\% \times \frac{\{\text{CCxH}, \text{CCxL}\}}{\{\text{RCAP2H}, \text{RCAP2L}\}}$$

$$\text{Inverting:} \qquad \text{Duty Cycle} = 100\% \times \frac{\{\text{RCAP2H}, \text{RCAP2L}\} - \{\text{CCxH}, \text{CCxL}\}}{\{\text{RCAP2H}, \text{RCAP2L}\}}$$

The extreme compare values represent special cases when generating a PWM waveform. If the compare value is set equal to (or greater than) TOP, the output will remain high or low for non-inverting and inverting modes, respectively. If the compare value is set to MIN (0000H), the output will remain low or high for non-inverting and inverting modes, respectively.

Figure 13-9. Non-overlapping Waveforms Using Symmetrical PWM



13.4.2.1 *Phase and Frequency Correct PWM*

When T2CM$_{1-0}$ = 10B the Symmetrical PWM operates in phase and frequency correct mode. In this mode the compare value double buffer is only updated when the timer equals MIN (underflow). This guarantees that the resulting waveform is always symmetrical around the TOP value as shown in Figure 13-10 because the up and down count compare values are identical. The TF2 interrupt flag is only set at underflow.

13.4.2.2 *Phase Correct PWM*

When T2CM$_{1-0}$ = 11B the Symmetrical PWM operates in phase correct mode. In this mode the compare value double buffer is updated when the timer equals MIN (underflow) and TOP (overflow). The resulting waveform may not be completely symmetrical around the TOP value as shown in Figure 13-11 because the up and down count compare values may not be identical. However, this allows the pulses to be weighted toward one edge or another. The TF2 interrupt flag is set at both underflow and overflow.

bit and prepares to receive the data bytes that follows. The slaves that are not addressed set their SM2 bits and ignore the data bytes. See "Automatic Address Recognition" on page 97.

The SM2 bit can be used to check the validity of the stop bit in Mode 1. In a Mode 1 reception, if SM2 = 1, the receive interrupt is not activated unless a valid stop bit is received.

Table 16-1. SCON – Serial Port Control Register

SCON Address = 98H Reset Value = 0000 0000B

Bit Addressable

	SM0/FE	SM1	SM2	REN	TB8	RB8	T1	RI
Bit	7	6	5	4	3	2	1	0

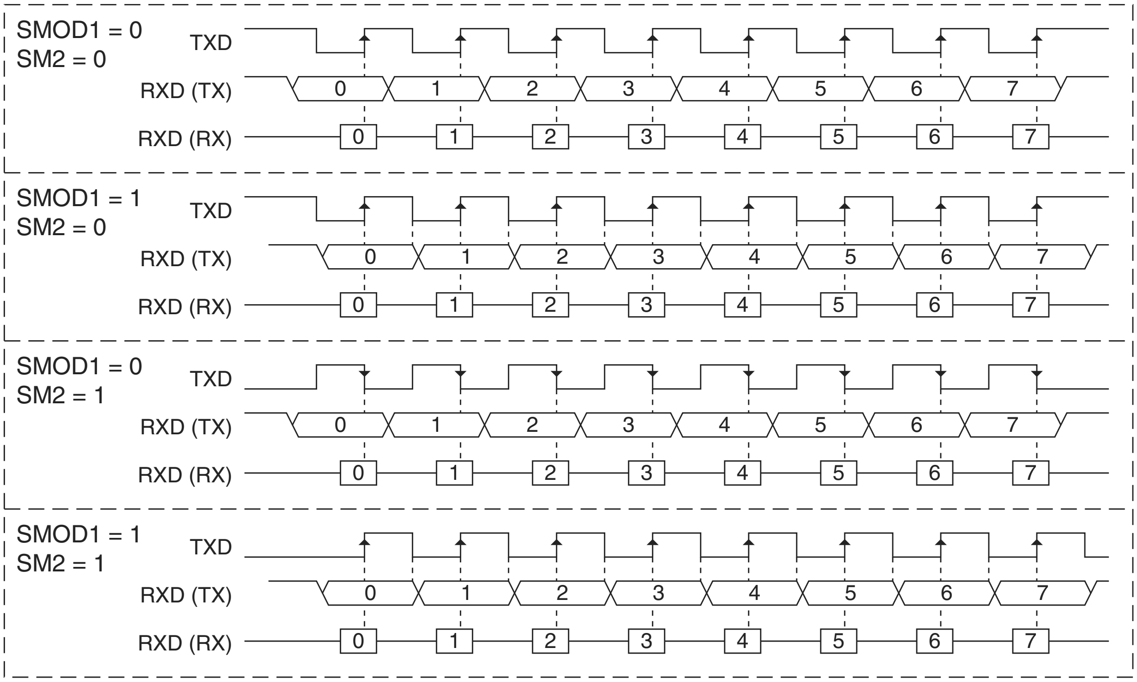
(SMOD0 = 0/1)[1]

<table>
<tr><th>Symbol</th><th>Function</th></tr>
<tr><td>FE</td><td>Framing error bit. This bit is set by the receiver when an invalid stop bit is detected. The FE bit is not cleared by valid frames and must be cleared by software. The SMOD0 bit must be set to enable access to the FE bit. FE will be set regardless of the state of SMOD0.</td></tr>
<tr><td>SM0</td><td>Serial Port Mode Bit 0, (SMOD0 must = 0 to access bit SM0)</td></tr>
<tr><td>SM1</td><td>Serial Port Mode Bit 1
<table>
<tr><th>SM0</th><th>SM1</th><th>Mode</th><th>Description</th><th>Baud Rate[2]</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>shift register</td><td>$f_{osc}/2$ or $f_{osc}/4$ or Timer 1</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>8-bit UART</td><td>variable (Timer 1 or Timer 2)</td></tr>
<tr><td>1</td><td>0</td><td>2</td><td>9-bit UART</td><td>$f_{osc}/32$ or $f_{osc}/16$</td></tr>
<tr><td>1</td><td>1</td><td>3</td><td>9-bit UART</td><td>variable (Timer 1 or Timer 2)</td></tr>
</table>
</td></tr>
<tr><td>SM2</td><td>Enables the Automatic Address Recognition feature in Modes 2 or 3. If SM2 = 1 then RI will not be set unless the received 9th data bit (RB8) is 1, indicating an address, and the received byte is a Given or Broadcast Address. In Mode 1, if SM2 = 1 then RI will not be activated unless a valid stop bit was received, and the received byte is a Given or Broadcast Address. In Mode 0, SM2 determines the idle state of the shift clock such that the clock is the inverse of SM2, i.e. when SM2 = 0 the clock idles high and when SM2 = 1 the clock idles low.</td></tr>
<tr><td>REN</td><td>Enables serial reception. Set by software to enable reception. Clear by software to disable reception.</td></tr>
<tr><td>TB8</td><td>The 9th data bit that will be transmitted in Modes 2 and 3. Set or clear by software as desired. In Mode 0, setting TB8 enables Timer 1 as the shift clock generator.</td></tr>
<tr><td>RB8</td><td>In Modes 2 and 3, the 9th data bit that was received. In Mode 1, if SM2 = 0, RB8 is the stop bit that was received. In Mode 0, RB8 is not used.</td></tr>
<tr><td>TI</td><td>Transmit interrupt flag. Set by hardware at the end of the 8th bit time in Mode 0, or at the beginning of the stop bit in the other modes, in any serial transmission. Must be cleared by software.</td></tr>
<tr><td>RI</td><td>Receive interrupt flag. Set by hardware at the end of the 8th bit time in Mode 0, or halfway through the stop bit time in the other modes, in any serial reception (except see SM2). Must be cleared by software.</td></tr>
</table>

Notes: 1. SMOD0 is located at PCON.6.

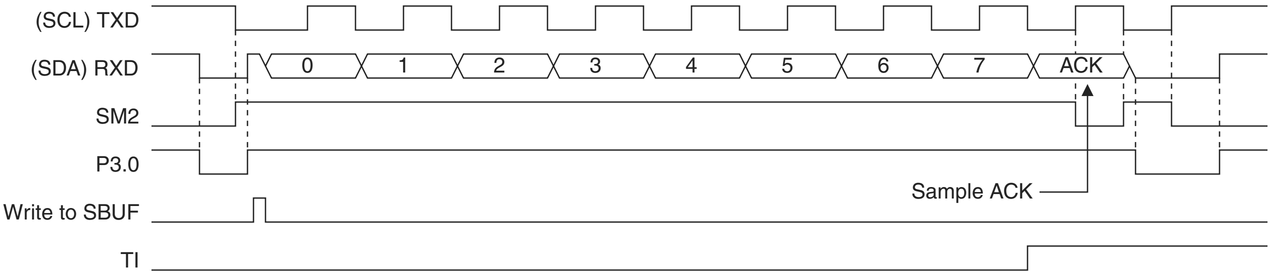
2. f_{osc} = oscillator frequency. The baud rate depends on SMOD1 (PCON.7).

Figure 16-2. Mode 0 Waveforms



Mode 0 may be used as a hardware accelerator for software emulation of serial interfaces such as a half-duplex Serial Peripheral Interface (SPI) in mode (0,0) or (1,1) or a Two-Wire Interface (TWI) in master mode. An example of Mode 0 emulating a TWI master device is shown in Figure 16-3. In this example, the start, stop, and acknowledge are handled in software while the byte transmission is done in hardware. Falling/rising edges on TXD are created by setting/clearing SM2. Rising/falling edges on RXD are forced by setting/clearing the P3.0 register bit. SM2 and P3.0 must be 1 while the byte is being transferred.

Figure 16-3. UART Mode 0 TWI Emulation (SMOD1 = 1)



Mode 0 transfers data LSB first whereas SPI or TWI are generally MSB first. Emulation of these interfaces may require bit reversal of the transferred data bytes. The following code example reverses the bits in the accumulator:

```
EX:    MOV  R7, #8
REVRS: RLC  A            ; C << msb(ACC)
       XCH  A, R6
       RRC  A            ; msb(ACC) >> B
       XCH  A, R6
       DJNZ R7, REVRS
```

In a more complex system, the following could be used to select slaves 1 and 2 while excluding slave 0:

Slave 0	SADDR = 1100 0000
	SADEN = 1111 1001
	Given = 1100 0XX0

Slave 1	SADDR = 1110 0000
	SADEN = 1111 1010
	Given = 1110 0X0X

Slave 2	SADDR = 1110 0000
	SADEN = 1111 1100
	Given = 1110 00XX

In the above example, the differentiation among the 3 slaves is in the lower 3 address bits. Slave 0 requires that bit 0 = 0 and it can be uniquely addressed by 1110 0110. Slave 1 requires that bit 1 = 0 and it can be uniquely addressed by 1110 and 0101. Slave 2 requires that bit 2 = 0 and its unique address is 1110 0011. To select Slaves 0 and 1 and exclude Slave 2, use address 1110 0100, since it is necessary to make bit 2 = 1 to exclude slave 2.

The Broadcast Address for each slave is created by taking the logic OR of SADDR and SADEN. Zeros in this result are trended as don't cares. In most cases, interpreting the don't cares as ones, the broadcast address will be FF hexadecimal.

Upon reset SADDR (SFR address 0A9H) and SADEN (SFR address 0B9H) are loaded with "0"s. This produces a given address of all "don't cares" as well as a Broadcast address of all "don't cares". This effectively disables the Automatic Addressing mode and allows the microcontroller to use standard 80C51-type UART drivers which do not make use of this feature.

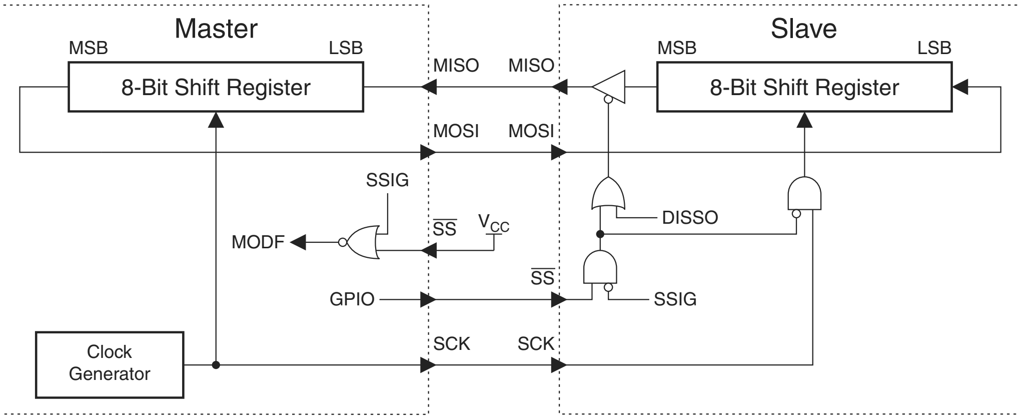
17. Enhanced Serial Peripheral Interface

The Serial Peripheral Interface (SPI) allows high-speed synchronous data transfer between the AT89LP3240/6440 and peripheral devices or between multiple AT89LP3240/6440 devices, including multiple masters and slaves on a single bus. The SPI includes the following features:

- Full-duplex, 3-wire or 4-wire Synchronous Data Transfer
- Master or Slave Operation
- Maximum Bit Frequency = $f_{OSC}/4$
- LSB First or MSB First Data Transfer
- Four Programmable Bit Rates or Timer 1-based Baud Generation (Master Mode)
- End of Transmission Interrupt Flag
- Write Collision Flag Protection
- Double-buffered Receive and Transmit
- Transmit Buffer Empty Interrupt Flag
- Mode Fault (Master Collision) Detection and Interrupt
- Wake up from Idle Mode

A block diagram of the SPI is shown below in Figure 17-1.

Figure 17-2. SPI Master-Slave Interconnection



When the SPI is configured as a Master (MSTR in SPCR is set), the operation of the $\overline{SS}$ pin depends on the setting of the Slave Select Ignore bit, SSIG. If SSIG = 1, the $\overline{SS}$ pin is a general purpose output pin which does not affect the SPI system. Typically, the pin will be driving the $\overline{SS}$ pin of an SPI Slave. If SSIG = 0, $\overline{SS}$ must be held high to ensure Master SPI operation. If the $\overline{SS}$ pin is driven low by peripheral circuitry when the SPI is configured as a Master with SSIG = 0, the SPI system interprets this as another master selecting the SPI as a slave and starting to send data to it. To avoid bus contention, the SPI system takes the following actions:

1. The MSTR bit in SPCR is cleared and the SPI system becomes a Slave. As a result of the SPI becoming a Slave, the MOSI and SCK pins become inputs.
2. The MODF Flag in SPSR is set, and if the SPI interrupt is enabled, the interrupt routine will be executed.

Thus, when interrupt-driven SPI transmission is used in Master mode, and there exists a possibility that $\overline{SS}$ may be driven low, the interrupt should always check that the MSTR bit is still set. If the MSTR bit has been cleared by a slave select, it must be set by the user to re-enable SPI Master mode.

17.1 Master Operation

An SPI master device initiates all data transfers on the SPI bus. The AT89LP3240/6440 is configured for master operation by setting MSTR = 1 in SPCR. Writing to the SPI data register (SPDR) while in master mode loads the transmit buffer. If the SPI shift register is empty, the byte in the transmit buffer is moved to the shift register; the transmit buffer empty flag, TXE, is set; and a transmission begins. The transfer may start after an initial delay, while the clock generator waits for the next full bit slot of the specified baud rate. The master shifts the data out serially on the MOSI line while providing the serial shift clock on SCK. When the transfer finishes, the SPIF flag is set to “1” and an interrupt request is generated, if enabled. The data received from the addressed SPI slave device is also transferred from the shift register to the receive buffer. Therefore, the SPIF bit flags both the transmit-complete and receive-data-ready conditions. The received data is accessed by reading SPDR.

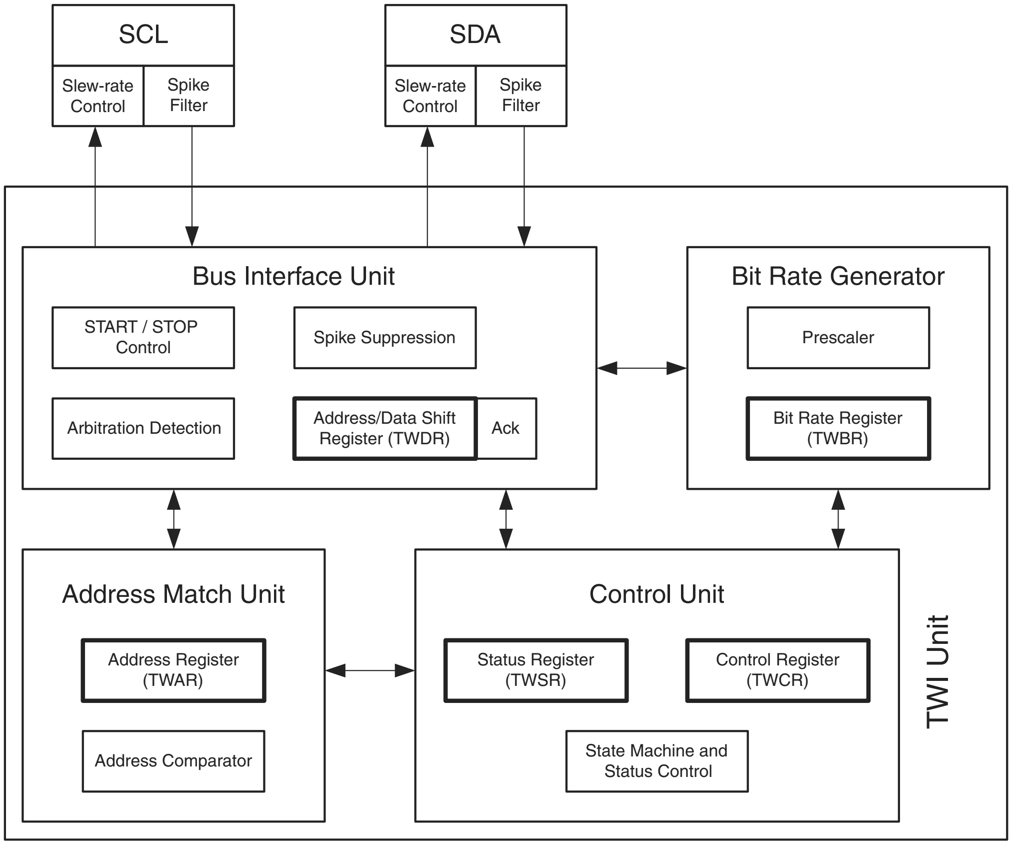
While the TXE flag is set, the transmit buffer is empty. TXE can be cleared by software or by writing to SPDR. Writing to SPDR will clear TXE and load the transmit buffer. The user may load the buffer while the shift register is busy, i.e. before the current transfer completes. When the current transfer completes, the queued byte in the transmit buffer is moved to the shift register and the next transfer commences. TXE will generate an interrupt if the SPI interrupt is enabled

It is the user software's responsibility to ensure that these illegal arbitration conditions never occur. This implies that in multi-master systems, all data transfers must use the same composition of SLA+R/W and data packets. In other words: All transmissions must contain the same number of data packets, otherwise the result of the arbitration is undefined.

18.3 Overview of the TWI Module

The TWI module is comprised of several submodules, as shown in Figure 18-9. All registers drawn in a thick line are accessible through the AT89LP data bus.

Figure 18-9. Overview of the TWI Module



18.3.1 SCL and SDA Pins

These pins interface the AT89LP TWI with the rest of the MCU system. The output drivers contain a slew-rate limiter in order to conform to the TWI specification. The input stages contain a spike suppression unit removing spikes shorter than 50 ns.

18.3.2 Bit Rate Generator Unit

This unit controls the period of SCL when operating in a Master mode. The SCL period is controlled by settings in the TWI Bit Rate Register (TWBR). Slave operation does not depend on the Bit Rate setting, but the CPU clock frequency in the slave must be at least 16 times higher than the SCL frequency. Note that slaves may prolong the SCL low period, thereby reducing the

Table 18-9. Status Codes for Slave Transmitter Mode

Status Code (TWSR)	Status of the Two-wire Serial Bus and Two-wire Serial Interface Hardware	Application Software Response					Next Action Taken by TWI Hardware
			To TWCR				
		To/from TWDR	STA	STO	TWIF	AA	
A8h	Own SLA+R has been received; ACK has been returned	Load data byte	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
		Load data byte	X	0	1	1	Data byte will be transmitted and ACK should be received
B0h	Arbitration lost in SLA+R/W as master; own SLA+R has been received; ACK has been returned	Load data byte	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
		Load data byte	X	0	1	1	Data byte will be transmitted and ACK should be received
B8h	Data byte in TWDR has been transmitted; ACK has been received	Load data byte	X	0	1	0	Last data byte will be transmitted and NOT ACK should be received
		Load data byte	X	0	1	1	Data byte will be transmitted and ACK should be received
C0h	Data byte in TWDR has been transmitted; NOT ACK has been received	No action	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		No action	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if GC = “1”
		No action	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		No action	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if GC = “1”; a START condition will be transmitted when the bus becomes free
C8h	Last data byte in TWDR has been transmitted (AA = “0”); ACK has been received	No action	0	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA
		No action	0	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if GC = “1”
		No action	1	0	1	0	Switched to the not addressed Slave mode; no recognition of own SLA or GCA; a START condition will be transmitted when the bus becomes free
		No action	1	0	1	1	Switched to the not addressed Slave mode; own SLA will be recognized; GCA will be recognized if GC = “1”; a START condition will be transmitted when the bus becomes free

- When using the Internal RC Oscillator during debug, DDA is located on the XTAL1/P4.0 pin. The P4.0 I/O function cannot be emulated in this mode.
- When using the External Clock during debug, DDA is located on the XTAL2/P4.1 pin and the system clock drives XTAL1/P4.0. The P4.1 I/O and CLKOUT functions cannot be emulated in this mode.
- When using the Crystal Oscillator during debug, DDA is located on the P4.3 pin and the crystal connects to XTAL1/P4.0 and XTAL2/P4.1. The P4.3 I/O function cannot be emulated in this mode.

25. Programming the Flash Memory

The Atmel AT89LP3240/6440 microcontroller features 64K bytes of on-chip In-System Programmable Flash program memory and 8K bytes of nonvolatile Flash data memory. In-System Programming allows programming and reprogramming of the microcontroller positioned inside the end system. Using a simple 4-wire SPI interface, the programmer communicates serially with the AT89LP3240/6440 microcontroller, reprogramming all nonvolatile memories on the chip. In-System Programming eliminates the need for physical removal of the chips from the system. This will save time and money, both during development in the lab, and when updating the software or parameters in the field. The programming interface of the AT89LP3240/6440 includes the following features:

- Four-wire serial SPI Programming Interface or 11-pin Parallel Interface
- Active-low Reset Entry into Programming
- Slave Select allows multiple devices on same interface
- User Signature Array
- Flexible Page Programming
- Row Erase Capability
- Page Write with Auto-Erase Commands
- Programming Status Register

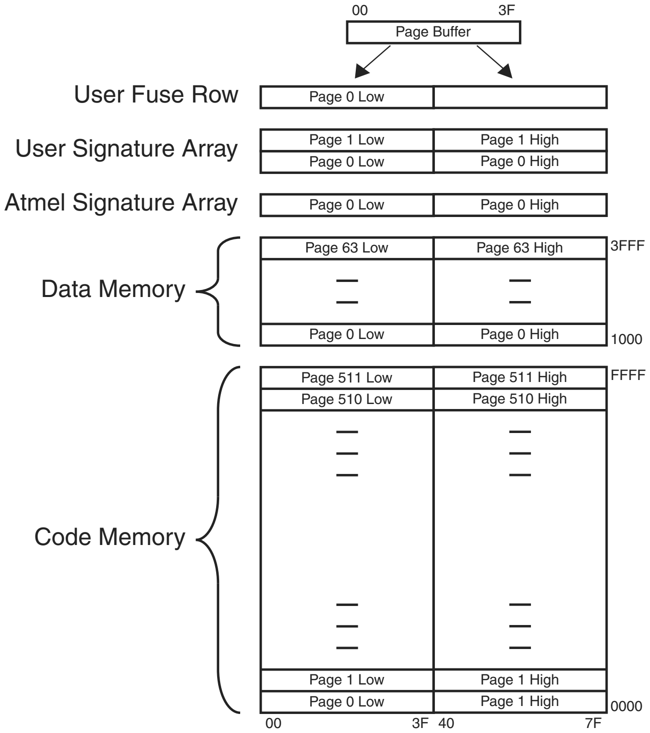
For more detailed information on In-System Programming, refer to the Application Note entitled “AT89LP In-System Programming Specification”.

25.1 Physical Interface

The AT89LP3240/6440 provides a standard programming command set with two physical interfaces: a bit-serial and a byte-parallel interface. Normal Flash programming utilizes the Serial Peripheral Interface (SPI) pins of an AT89LP3240/6440 microcontroller. The SPI is a full-duplex synchronous serial interface consisting of four wires: Serial Clock (SCK), Master-In/Slave-out (MISO), Master-out/Slave-in (MOSI), and an active-low chip select and frame signal ($\overline{\text{SS}}$). When programming an AT89LP3240/6440 device, the programmer always operates as the SPI master, and the target system always operates as the SPI slave. To enter or remain in Programming mode the device’s reset line ($\overline{\text{RST}}$) must be held active (low). With the addition of VDD and GND, an AT89LP3240/6440 microcontroller can be programmed with a minimum of seven connections as shown in Figure 25-1.

In addition to being a chip select, the $\overline{\text{SS}}$ pin also is used to frame a command packet. $\overline{\text{SS}}$ must go low before the start of a command and must return high to complete the command. **$\overline{\text{SS}}$ must NOT be tied to ground as this will prevent the interface from recognizing mutiple commands**. $\overline{\text{SS}}$ should be connected to the programming master for correct operation.

Figure 25-3. AT89LP6440 Memory Organization



25.3 Command Format

Programming commands consist of an opcode byte, two address bytes, and zero or more data bytes. In addition, all command packets must start with a two-byte preamble of AAH and 55H. The preamble increases the noise immunity of the programming interface by making it more difficult to issue unintentional commands. Figure 25-4 on page 161 shows a simplified flow chart of a command sequence.

A sample command packet is shown in Figure 25-5 on page 161. The $\overline{\text{SS}}$ pin defines the packet frame. $\overline{\text{SS}}$ must be brought low before the first byte in a command is sent and brought back high after the final byte in the command has been sent. The command is not complete until $\overline{\text{SS}}$ returns high. Command bytes are issued serially on MOSI. Data output bytes are received serially on MISO. Packets of variable length are supported by returning $\overline{\text{SS}}$ high when the final required byte has been transmitted. In some cases command bytes have a don't care value. Don't care bytes in the middle of a packet must be transmitted. Don't care bytes at the end of a packet may be ignored.

Page oriented instructions always include a full 16-bit address. The higher order bits select the page and the lower order bits select the byte within that page. The AT89LP3240/6440 allocates 6 bits for byte address, 1 bit for low/high half page selection and 9 bits for page address. The half page to be accessed is always fixed by the page address and half select as transmitted. The byte address specifies the starting address for the first data byte. After each data byte has been transmitted, the byte address is incremented to point to the next data byte. This allows a page command to linearly sweep the bytes within a page. If the byte address is incremented past the last byte in the half page, the byte address will roll over to the first byte in the same half page. While loading bytes into the page buffer, overwriting previously loaded bytes will result in data corruption.

For a summary of available commands, see Table 25-2 on page 162.

Figure 25-4. Command Sequence Flow Chart

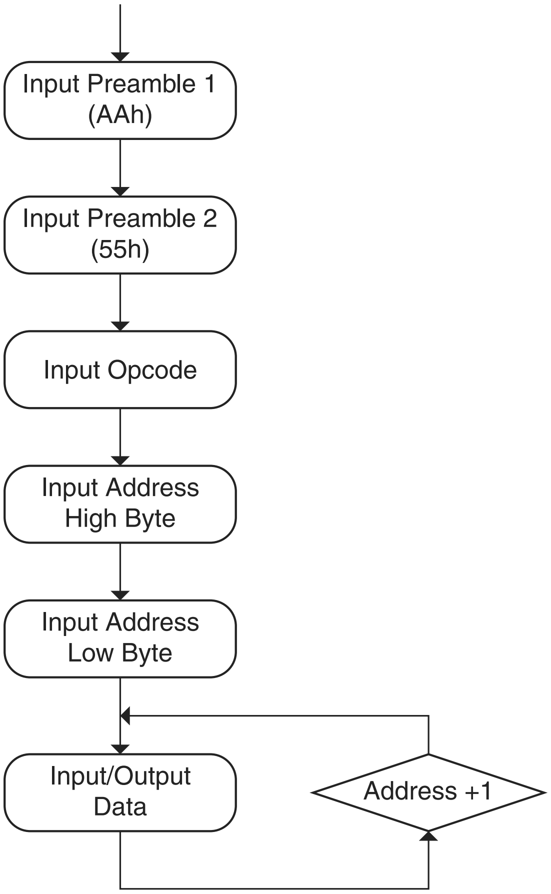


Figure 25-5. ISP Command Packet (Serial)

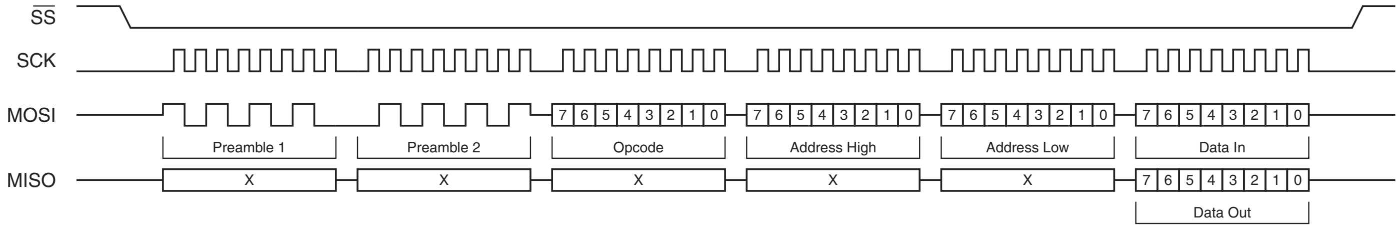
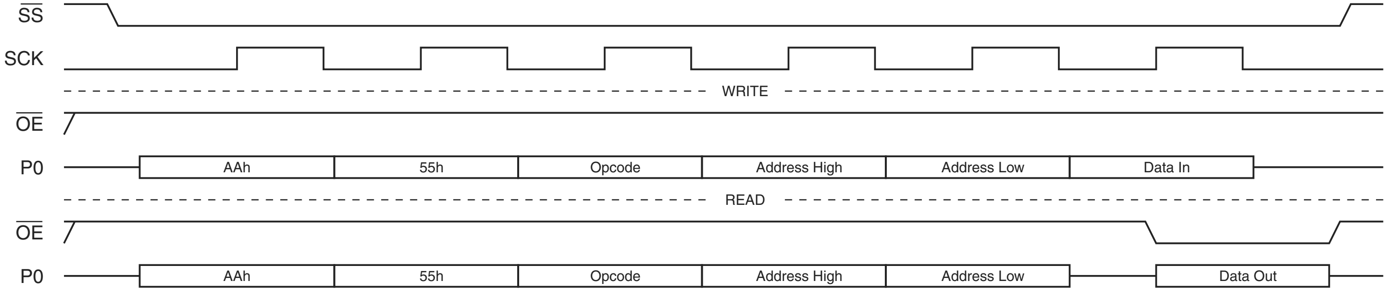


Figure 25-6. ISP Command Packet (Parallel)



25.4 Status Register

The current state of the memory may be accessed by reading the status register. The status register is shown in Table 25-3.

Table 25-3. Status Register

	–	–	–	–	$\overline{\text{LOAD}}$	SUCCESS	$\overline{\text{WRTINH}}$	$\overline{\text{BUSY}}$
Bit	7	6	5	4	3	2	1	0

Symbol	Function
$\overline{\text{LOAD}}$	Load flag. Cleared low by the load page buffer command and set high by the next memory write. This flag signals that the page buffer was previously loaded with data by the load page buffer command.
SUCCESS	Success flag. Cleared low at the start of a programming cycle and will only be set high if the programming cycle completes without interruption from the brownout detector.
$\overline{\text{WRTINH}}$	Write Inhibit flag. Cleared low by the brownout detector (BOD) whenever programming is inhibited due to V_{DD} falling below the minimum required programming voltage. If a BOD episode occurs during programming, the SUCCESS flag will remain low after the cycle is complete.
$\overline{\text{BUSY}}$	Busy flag. Cleared low whenever the memory is busy programming or if write is currently inhibited.

25.5 $\overline{\text{DATA}}$ Polling

The AT89LP3240/6440 implements $\overline{\text{DATA}}$ polling to indicate the end of a programming cycle. While the device is busy, any attempted read of the last byte written will return the data byte with the MSB complemented. Once the programming cycle has completed, the true value will be accessible. During Erase the data is assumed to be FFH and $\overline{\text{DATA}}$ polling will return 7FH. When writing multiple bytes in a page, the $\overline{\text{DATA}}$ value will be the last data byte loaded before programming begins, not the written byte with the highest physical address within the page.

25.6 Flash Security

The AT89LP3240/6440 provides two Lock Bits for Flash Code and Data Memory security. Lock bits can be left unprogrammed (FFh) or programmed (00h) to obtain the protection levels listed in Table 25-4. Lock bits can only be erased (set to FFh) by Chip Erase. Lock bit mode 2 disables programming of all memory spaces, including the User Signature Array and User Configuration Fuses. User fuses must be programmed before enabling Lock bit mode 2 or 3. Lock bit mode 3 implements mode 2 and also blocks reads from the code and data memories; however, reads of the User Signature Array, Atmel Signature Array, and User Configuration Fuses are still allowed.

The Lock Bits will not disable FDATA or IAP programming initiated by the application software.

Table 25-4. Lock Bit Protection Modes

Program Lock Bits (by address)			
Mode	**00h**	**01h**	**Protection Mode**
1	FFh	FFh	No program lock features
2	00h	FFh	Further programming of the Flash is disabled
3	00h	00h	Further programming of the Flash is disabled and verify (read) is also disabled; OCD is disabled

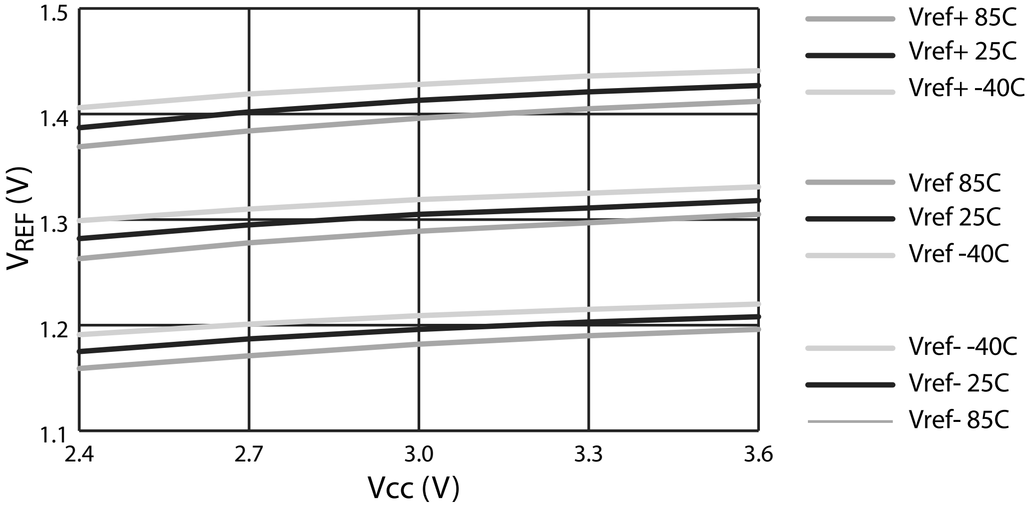
26.11 Dual Analog Comparator Characteristics

The values shown in this table are valid for T_A = -40°C to 85°C and V_{DD} = 2.4 to 3.6V, unless otherwise noted.

Table 26-8. Dual Analog Comparator Characteristics

Symbol	Parameter	Condition	Min	Max	Units
V_{CM}	Common Mode Input Voltage		GND	V_{DD}	V
V_{OS}	Input Offset Voltage	V_{DD} = 3.6V		20	mV
V_{AREF}	Analog Reference Voltage		1.23	1.36	V
$V_{\Delta REF}$	Reference Delta Voltage		90	120	mV
t_{CMP}	Comparator Propagation Delay	$V_{IN+} – V_{IN-}$ = 20mV; V_{DD} = 2.4V		200	ns
t_{AREF}	Reference Settling Time		3		µs

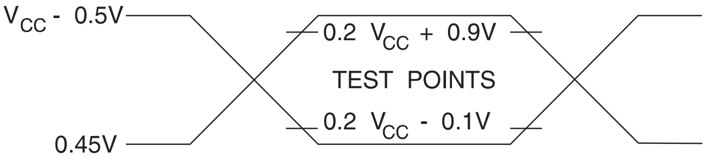
Figure 26-23. Analog Reference Voltage Typical Characteristics



26.13 Test Conditions

26.13.1 AC Testing Input/Output Waveform

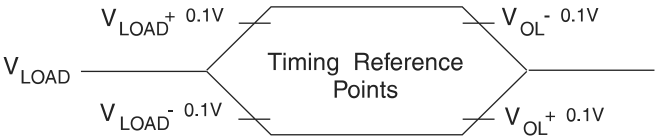
Figure 26-24. AC Testing Input/Output Waveform[1]



Note: 1. AC Inputs during testing are driven at V_{DD} - 0.5V for a logic “1” and 0.45V for a logic “0”. Timing measurements are made at V_{IH} min. for a logic “1” and V_{IL} max. for a logic “0”.

26.13.2 Float Waveform

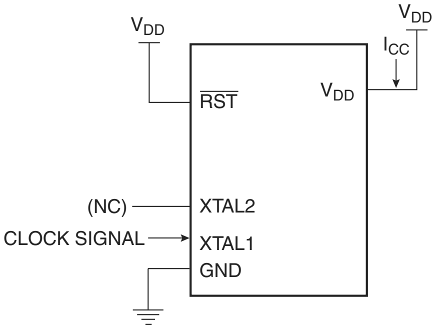
Figure 26-25. Float Waveform[1]



Note: 1. For timing purposes, a port pin is no longer floating when a 100 mV change from load voltage occurs. A port pin begins to float when 100 mV change from the loaded V_{OH}/V_{OL} level occurs.

26.13.3 I_{CC} Test Condition: Active Mode

Figure 26-26. Connection Diagram for I_{CC} Active Measurement. All Other Pins are Disconnected



For active supply current measurements all ports are configured in quasi-bidirectional mode. Timers 0, 1 and 2 are configured to be free running in their default timer modes. The CPU executes a simple random number generator that accesses RAM, the SFR bus and exercises the ALU and hardware multiplier.