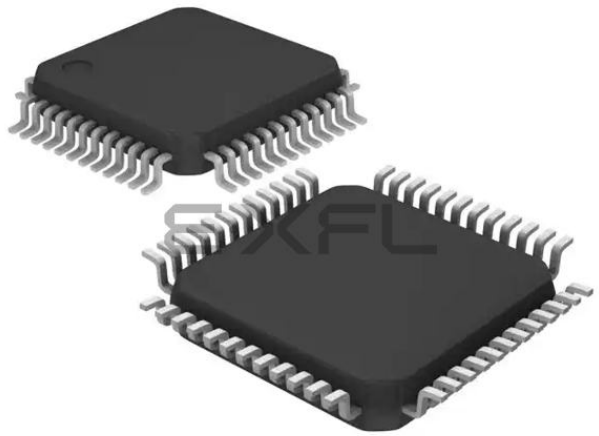


**Welcome to E-XFL.COM**

## 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 | 8051/52 |
| Core Size | 8-Bit |
| Speed | 40MHz |
| Connectivity | EBI/EMI, I²C, SPI, UART/USART |
| Peripherals | POR, PWM, WDT |
| Number of I/O | 38 |
| Program Memory Size | 16KB (16K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | 1K x 8 |
| RAM Size | 1.25K x 8 |
| Voltage - Supply (Vcc/Vdd) | 2.7V ~ 5.5V |
| Data Converters | A/D 8x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 48-LQFP |
| Supplier Device Package | - |
| Purchase URL | https://www.e-xfl.com/product-detail/nuvoton-technology-corporation-america/w79e225alg |

| INSTRUCTIONS | | ENNVM = 1<br>NVM SIZE = SRAM (1K) | |
|---|---|---|---|
| | | ADDR ≤ 1K | ADDR > 1K |
| Read access | MOVX A, @DPTR (Read) | NVM[1] | Ext memory[1] |
| | MOVX A, @R0 (Read) | NVM[2] | NOP |
| | MOVX A, @R1 (Read) | NVM[2] | NOP |
| Write access | MOVX @DPTR, A (Write) | NOP | Ext memory[1] |
| | MOVX @R0, A (Write) | NOP | NOP |
| | MOVX @R1, A (Write) | NOP | NOP |

Table 6-2: W79E225 MOVX read/write access destination

| INSTRUCTIONS | | ENNVM = 1<br>NVM SIZE = SRAM (2K) | |
|---|---|---|---|
| | | ADDR ≤ 2K | ADDR > 2K |
| Read access | MOVX A, @DPTR (Read) | NVM[1] | Ext memory[1] |
| | MOVX A, @R0 (Read) | NVM[2] | NOP |
| | MOVX A, @R1 (Read) | NVM[2] | NOP |
| Write access | MOVX @DPTR, A (Write) | NOP | Ext memory[1] |
| | MOVX @R0, A (Write) | NOP | NOP |
| | MOVX @R1, A (Write) | NOP | NOP |

Table 6-3: W79E226/227 MOVX read/write access destination

**Note:**

1. A15~A0=DPTR
2. A15~A8=XRAMAH

### 6.4.1 Operation

User is required to enable EnNVM (NVMCON.5) bit for all NVM access (read/write/erase).

Before write data to NVM Data, the page must be erased. A page is erased by setting page address which address will decode and enable page (n) on NVMADDRH and NVMADDRL, then set EER (NVMCON.7) and EnNVM (NVMCON.5). The device will then automatic execute page erase. When completed, NVMF will be set by hardware. NVMF should be cleared by software. Interrupt request will be generated if ENVM (EIE1.5) is enabled. EER bit will be cleared by hardware when erase is completed. The total erase time is about 5ms.

For write, user must set address and data to NVMADDRH/L and NVMDAT, respectively. And then set EWR (NVMCON.6) and EnNVM (NVMCON.5) to enable data write. When completed, the device will set NVMF flag. NVMF flag should be cleared by software. Similarly, interrupt request will be generated if ENVM (EIE1.5) is enabled. The program time is about 50us.

The following shows some examples of NVM operations (using W79E226/227):

**Read NVM data is by MOVX A,@DPTR/R0/R1 instruction:**

A read exceed 2k will read the external address

Example1: DPTR=0x07FF, R0/R1 = 0xFF, XRAMAH=0x07, EnNVM=1

MOVX A,@DPTR → read NVM data at address 0x07FF

MOVX A,@R0 → read NVM data at address 0x07FF

MOVX A,@R1 → read NVM data at address 0x07FF

Example2: DPTR = 0x2000, EnNVM=1, DME0=0

MOVX A,@DPTR → read external RAM data at address 0x2000,

**Erase NVM by SFR register:**

Example1: NVMADDRH = 0x07, NVMADDRL = 0xF0, page 31 will be enabled. After set EER, the page 31 will be erased.

Example2: NVMADDRH = 0x10, NVMADDRL = 0x00, invalid NVM erase instruction (address exceed NVM boundary).

**Write NVM by SFR register:**

Example1: NVMADDRH = 0x07, NVMADDRL = 0xF0

After set EWR, data will be written to the NVM address = 0x07F0 location.

Example2: NVMADDRH = 0x10, NVMADDRL = 0x00, after set EWR, invalid NVM write instruction (address exceed NVM boundary).

During erase, write is invalid. Likewise, during write, erase is invalid. An erase or write is invalid if NVMF is not clear by software. A write to NVMADDRH and NVMADDRL is invalid during Erase or Write, and a write to NVMDAT is invalid only during NVM write access.

Continued

| SYMBOL | DEFINITION | ADDRESS | MSB LSB | BIT_ADDRESS, SYMBOL | | | | | | | RESET |
|---|---|---|---|---|---|---|---|---|---|---|---|
| TH0 | TIMER HIGH 0 | 8CH | TH0.7 | TH0.6 | TH0.5 | TH0.4 | TH0.3 | TH0.2 | TH0.1 | TH0.0 | 0000 0000B |
| TL1 | TIMER LOW 1 | 8BH | TL1.7 | TL1.6 | TL1.5 | TL1.4 | TL1.3 | TL1.2 | TL1.1 | TL1.0 | 0000 0000B |
| TL0 | TIMER LOW 0 | 8AH | TL0.7 | TL0.6 | TL0.5 | TL0.4 | TL0.3 | TL0.2 | TL0.1 | TL0.0 | 0000 0000B |
| TMOD | TIMER MODE | 89H | GATE | $C/\overline{T}$ | M1 | M0 | GATE | $C/\overline{T}$ | M1 | M0 | 0000 0000B |
| TCON | TIMER CONTROL | 88H | (8F) TF1 | (8E) TR1 | (8D) TF0 | (8C) TR0 | (8B) IE1 | (8A) IT1 | (89) IE0 | (88) IT0 | 0000 0000B |
| PCON | POWER CONTROL | 87H | SMOD | SMOD0 | - | - | GF1 | GF0 | PD | IDL | 00xx 0000B |
| TH3 | TIMER HIGH 3 | 85H | TH3.7 | TH3.6 | TH3.5 | TH3.4 | TH3.3 | TH3.2 | TH3.1 | TH3.0 | 0000 0000B |
| TL3 | TIMER LOW 3 | 84H | TL3.7 | TL3.6 | TL3.5 | TL3.4 | TL3.3 | TL3.2 | TL3.1 | TL3.0 | 0000 0000B |
| DPH | DATA POINTER HIGH | 83H | DPH.7 | DPH.6 | DPH.5 | DPH.4 | DPH.3 | DPH.2 | DPH.1 | DPH.0 | 0000 0000B |
| DPL | DATA POINTER LOW | 82H | DPL.7 | DPL.6 | DPL.5 | DPL.4 | DPL.3 | DPL.2 | DPL.1 | DPL.0 | 0000 0000B |
| SP | STACK POINTER | 81H | SP.7 | SP.6 | SP.5 | SP.4 | SP.3 | SP.2 | SP.1 | SP.0 | 0000 0111B |
| P0 | PORT 0 | 80H | (87) INT5 | (86) INT4 | (85) INT3 | (84) INT2 | (83) /SS | (82) SPCLK | (81) MOSI | (80) MISO | 1111 1111B |

Table 7-2: Special Function Registers

### DATA POINTER HIGH

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DPH.7 | DPH.6 | DPH.5 | DPH.4 | DPH.3 | DPH.2 | DPH.1 | DPH.0 |

Mnemonic: DPH Address: 83h

This is the high byte of the standard 8032 16-bit data pointer.

### TIMER 3 LSB

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TL3.7 | TL3.6 | TL3.5 | TL3.4 | TL3.3 | TL3.2 | TL3.1 | TL3.0 |

Mnemonic: TL3 Address: 84h

| BIT | NAME | FUNCTION |
|---|---|---|
| 7-0 | Timer 3 LSB | LSB of Timer3 |

### TIMER 3 MSB

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TH3.7 | TH3.6 | TH3.5 | TH3.4 | TH3.3 | TH3.2 | TH3.1 | TH3.0 |

Mnemonic: TH3 Address: 85h

| BIT | NAME | FUNCTION |
|---|---|---|
| 7-0 | Timer 3 MSB | MSB of Timer3 |

### POWER CONTROL

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SMOD | SMOD0 | - | - | GF1 | GF0 | PD | IDL |

Mnemonic: PCON Address: 87h

| BIT | NAME | FUNCTION |
|---|---|---|
| 7 | SMOD | This bit doubles the serial port baud rate in mode 1, 2, and 3 when set to 1. |
| 6 | SMOD0 | Framing Error Detection Enable. When SMOD0 is set to 1, then SCON.7 (SCON1.7) now indicates a Frame Error and acts as the FE (FE_1) flag. When SMOD0 is 0, then SCON.7 (SCON1.7) acts as per the standard 8032 function. |
| 5-4 | - | Reserved. |
| 3-2 | GF1-0 | These two bits are general purpose user flags. |
| 1 | PD | Setting this bit causes the device to go into the POWERDOWN mode. In this mode all the clocks are stopped and program execution is frozen. |
| 0 | IDL | Setting this bit causes the device to go into the IDLE mode. In this mode the clock to the CPU is stopped, so program execution is frozen, but the clock to the serial ports, timer, PWM, ADC, SPI and interrupt blocks is not stopped, and these blocks continue operating unhindered. |

**TIMER CONTROL**

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |

Mnemonic: TCON Address: 88h

| BIT | NAME | FUNCTION |
|---|---|---|
| 7 | TF1 | Timer 1 Overflow Flag. This bit is set when Timer 1 overflows. It is cleared automatically when the program does a timer 1 interrupt service routine. Software can also set or clear this bit. |
| 6 | TR1 | Timer 1 Run Control. This bit is set or cleared by software to turn timer/counter on or off. |
| 5 | TF0 | Timer 0 Overflow Flag. This bit is set when Timer 0 overflows. It is cleared automatically when the program does a timer 0 interrupt service routine. Software can also set or clear this bit. |
| 4 | TR0 | Timer 0 Run Control. This bit is set or cleared by software to turn timer/counter on or off. |
| 3 | IE1 | Interrupt 1 Edge Detect Flag: Set by hardware when an edge/level is detected on INT1. This bit is cleared by hardware when the service routine is vectored to only if the interrupt was edge triggered. Otherwise it follows the inverse of the pin. |
| 2 | IT1 | Interrupt 1 Type Control. Set/cleared by software to specify falling edge/ low level triggered external inputs. |
| 1 | IE0 | Interrupt 0 Edge Detect Flag. Set by hardware when an edge/level is detected on INT0. This bit is cleared by hardware when the service routine is vectored to only if the interrupt was edge triggered. Otherwise it follows the inverse of the pin. |
| 0 | IT0 | Interrupt 0 Type Control: Set/cleared by software to specify falling edge/ low level triggered external inputs. |

Continued

| BIT | NAME | FUNCTION |
|---|---|---|
| 1-0 | PMOD[1:0] | PWM Mode selects bits:<br>PMOD[1:0] – Description<br>00 – Edge-aligned mode. (up counter)<br>01 – Single-shot mode. (up counter)<br>10 – Center aligned mode (up-down counter)<br>11 – Reserved |

**Brake Condition Table**

| BPEN | BKCH | BRAKE CONDITION |
|---|---|---|
| 0 | 0 | Brake On, (Software brake and keeping brake).<br>Software brake condition. When active (BPEN=BKCH=0, and BKEN=1), PWM output follows PWMnB setting. This brake has no effect on PWMRUN bit; therefore, internal PWM generator continues to run. When the brake is released, the state of PWM output depends on the current state of PWM generator output during the release. |
| 0 | 1 | Brake On, when PWM is not running (PWMRUN=0), the PWM output condition is follow PWMnB setting. When the brake is released (by disabling BKEN = 0), the PWM output resumes to the state when PWM generator stop running prior to enabling the brake.<br>Brake Off, when PWM is running (PWMRUN=1). |
| 1 | 0 | Brake On, when Brake Pin asserted, PWM output follows PWMnB setting. The PWMRUN will be clear.<br>External pin brake condition. When active (by external pin), PWM output follows PWMnB setting. PWMRUN will be cleared by hardware. BKF flag will be set. When the brake is released (by de-asserting the external pin + disabling BKEN = 0), the PWM output resumes to the state of the PWM generator output prior to the brake. |
| 1 | 1 | This brake condition (by Brake Pin) causes BKF to be set, but PWM generator continues to run. The PWM output does not follow PWMnB, instead it output continuously as per normal without affected by the brake. |

**PWM 4 LOW BITS REGISTER**

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PWM4.7 | PWM4.6 | PWM4.5 | PWM4.4 | PWM4.3 | PWM4.2 | PWM4.1 | PWM4.0 |

Mnemonic: PWM4L Address: CFh

PWM4.7-0 PWM4 Low Bits Register.

**PROGRAM STATUS WORD**

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CY | AC | F0 | RS1 | RS0 | OV | F1 | P |

Mnemonic: PSW Address: D0h

| BIT | NAME | FUNCTION |
|---|---|---|

| BIT | NAME | FUNCTION |
|---|---|---|
| 7 | PWMEOM | PWM Channel 0, 2 and 4 Output Mode.<br>0 = Disable PWM channels 0, 2 and 4 to pwm output pins.<br>1 = Enable PWM channels 0, 2 and 4 to pwm output pins. |
| 6 | PWMOOM | PWM Channel 1, 3 and 5 Output Mode.<br>0 = Disable PWM channels 1, 3 and 5 to pwm output pins.<br>1 = Enable PWM channels 1, 3 and 5 to pwm output pins. |
| 5 | PWM6OM | PWM Channel 6 Output Mode.<br>0 = Disable PWM channel 6 to pwm output pin.<br>1 = Enable PWM channel 6 to pwm output pin. |
| 4 | PWM7OM | PWM Channel 7 Output Mode.<br>0 = Disable PWM channel 7 to pwm output pin.<br>1 = Enable PWM channel 7 to pwm output pin. |
| 3-1 | - | Reserved. |
| 0 | BKF | The External Brake Pin Flag.<br>0 = The PWM is not brake.<br>1 = The PWM is brake by external brake pin. It will be cleared by software. |

Together with option bits (PWMEE and PWMOE), PWMEOM, PWMOOM, PWM6OM and PWM7OM control the PWM pin structure, as follow;

| PWMEE/PWMOE (OPTION BITS) | PWMEOM/PWMOOM /PWM6OM/PWM7OM | PIO.X (X = 0-7) | PIN STRUCTURES |
|---|---|---|---|
| X | 0 | X | Tri-state |
| 1 (Disable) | 1 | X | Quasi (I/O output) |
| 0 (Enable) | 1 | 0 | Push Pull (PWM output) |
| 0 (Enable) | 1 | 1 | Push Pull (I/O output) |

Table 7-2: PWM pin structures (during internal rom execution)

| PWMEE/PWMOE (OPTION BITS) | PWMEOM/PWMOOM /PWM6OM/PWM7OM | PIO.X (X = 0-7) | PIN OUTPUT | PIN STRUCTURES |
|---|---|---|---|---|
| 1 (Disable) | X | X | External access | Push Pull |
| 0 (Enable) | X | X | External access | Push Pull (strong) |

Table 7-3: PWM pin structures (during external rom execution)

**Note**: PWMEOM/PWMOOM/PWM6OM/PWM7OM are cleared to zero when CPU in reset state. Thus, the port pins that multi-function with PWM will be tristated on default. User is required to set the bits to 1 to enable GPIO/PWM outputs.

| | | |
|---|---|---|
| 7 | SPIF | SPI Interrupt Complete Flag. SPIF is set upon completion of data transfer between this device and external device or when new data has been received and copied to the SPDR. If SPIF goes high, and if ESPI is set, a serial peripheral interrupt is generated. When SPIF is set; it must be clear by software and attempts to write SPDR are inhibited if SPIF set. |
| 6 | WCOL | Write Collision Flag. If a writer collision occurs on SPI bus, WCOL is set to high by hardware. WCOL must be clear by software. |
| 5 | SPIOVF | SPI overrun flag. SPIOVF is set if a new character is received before a previously received character is read from SPDR. Once this bit is set it will prevent SPDR register form excepting new data and must be cleared first before any new data can be written. This flag is clear by software.<br>0 = No overrun.<br>1 = Overrun detected. |
| 4 | MODF | SPI Mode Error Interrupt Status Flag. MODF is set when hardware detects mode fault. This bit is cleared by software. |
| 3 | DRSS | Data Register Slave Select. Refer to above table in SPCR register. |
| 2-0 | - | Reserved. |

**Note:** Bits WCOL, MODF and SPIF are cleared by software writing “0” to them.

### SERIAL PERIPHERAL DATA I/O REGISTER

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SPD.7 | SPD.6 | SPD.5 | SPD.4 | SPD.3 | SPD.2 | SPD.1 | SPD.0 |

Mnemonic: SPDR Address: F5h

| BIT | NAME | FUNCTION |
|---|---|---|
| 7-0 | SPD | SPDR is used when transmitting or receiving data on serial bus. Only a write to this register initiates transmission or reception of a byte, and this only occurs in the master device. A read of the SPDR is actually a read of a buffer. To prevent an overrun and the loss of the byte that caused the overrun, the first SPIF must be cleared by the time a second transfer of data from the shift Register to the read buffer is initiated. |

### I2C SLAVE ADDRESS MASK ENABLE

| Bit: | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | I2CSADEN.7 | I2CSADEN.6 | I2CSADEN.5 | I2CSADEN.4 | I2CSADEN.3 | I2CSADEN.2 | I2CSADEN.1 | I2CSADEN.0 |

Mnemonic: I2CSADEN Address: F6h

| BIT | NAME | FUNCTION |
|---|---|---|
| 7-0 | I2CSADEN | This register enables the Automatic Address Recognition feature of the I2C. When a bit in the I2CSADEN is set to 1, the same bit location in I2CSADDR1 will be compared with the incoming serial port data. When I2CSADEN.n is 0, the bit becomes don't care in the comparison. This register enables the Automatic Address Recognition feature of the I2C. When all the bits of I2CSADEN are 0, interrupt will occur for any incoming address. The default value is 0xFE. |

## 8. INSTRUCTION SET

The W79E22X SERIES executes all the instructions of the standard 8051/52 family. The operations of these instructions, as well as their effects on flag and status bits, are exactly the same. However, the timing of these instructions is different in two ways. Firstly, the W79E22X SERIES machine cycle is four clock periods, while the standard-8051/52 machine cycle is twelve clock periods. Secondly, the W79E22X SERIES can fetch only once per machine cycle (i.e., four clocks per fetch), while the standard 8051/52 can fetch twice per machine cycle (i.e., six clocks per fetch).

The timing differences create an advantage for the W79E22X SERIES. There is only one fetch per machine cycle, so the number of machine cycles is usually equal to the number of operands in the instruction. (Jumps and calls do require an additional cycle to calculate the new address.) As a result, the W79E22X SERIES reduces the number of dummy fetches and wasted cycles, and therefore improves overall efficiency, compared to the standard 8051/52.

| OP-CODE | HEX CODE | BYTES | W79E22X SERIES MACHINE CYCLE | W79E22X SERIES CLOCK CYCLES | 8032 CLOCK CYCLES | W79E22X SERIES VS. 8032 SPEED RATIO |
|---|---|---|---|---|---|---|
| NOP | 00 | 1 | 1 | 4 | 12 | 3 |
| ADD A, R0 | 28 | 1 | 1 | 4 | 12 | 3 |
| ADD A, R1 | 29 | 1 | 1 | 4 | 12 | 3 |
| ADD A, R2 | 2A | 1 | 1 | 4 | 12 | 3 |
| ADD A, R3 | 2B | 1 | 1 | 4 | 12 | 3 |
| ADD A, R4 | 2C | 1 | 1 | 4 | 12 | 3 |
| ADD A, R5 | 2D | 1 | 1 | 4 | 12 | 3 |
| ADD A, R6 | 2E | 1 | 1 | 4 | 12 | 3 |
| ADD A, R7 | 2F | 1 | 1 | 4 | 12 | 3 |
| ADD A, @R0 | 26 | 1 | 1 | 4 | 12 | 3 |
| ADD A, @R1 | 27 | 1 | 1 | 4 | 12 | 3 |
| ADD A, direct | 25 | 2 | 2 | 8 | 12 | 1.5 |
| ADD A, #data | 24 | 2 | 2 | 8 | 12 | 1.5 |
| ADDC A, R0 | 38 | 1 | 1 | 4 | 12 | 3 |
| ADDC A, R1 | 39 | 1 | 1 | 4 | 12 | 3 |
| ADDC A, R2 | 3A | 1 | 1 | 4 | 12 | 3 |
| ADDC A, R3 | 3B | 1 | 1 | 4 | 12 | 3 |
| ADDC A, R4 | 3C | 1 | 1 | 4 | 12 | 3 |
| ADDC A, R5 | 3D | 1 | 1 | 4 | 12 | 3 |
| ADDC A, R6 | 3E | 1 | 1 | 4 | 12 | 3 |
| ADDC A, R7 | 3F | 1 | 1 | 4 | 12 | 3 |
| ADDC A, @R0 | 36 | 1 | 1 | 4 | 12 | 3 |
| ADDC A, @R1 | 37 | 1 | 1 | 4 | 12 | 3 |
| ADDC A, direct | 35 | 2 | 2 | 8 | 12 | 1.5 |

Stretching only affects the MOVX instruction. There is no effect on any other instruction or its timing; it is as if the state of the CPU is held for the desired period. The timing waveforms when the stretch value is zero, one, and two are shown below.

| M2 | M1 | M0 | MACHINE CYCLES | $\overline{RD}$ OR $\overline{WR}$ STROBE WIDTH IN CLOCKS | $\overline{RD}$ OR $\overline{WR}$ STROBE WIDTH @ 25 MHZ | $\overline{RD}$ OR $\overline{WR}$ STROBE WIDTH @ 40 MHZ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 2 | 2 | 80 nS | 50 nS |
| 0 | 0 | 1 | 3 (default) | 4 | 160 nS | 100 nS |
| 0 | 1 | 0 | 4 | 8 | 320 nS | 200 nS |
| 0 | 1 | 1 | 5 | 12 | 480 nS | 300 nS |
| 1 | 0 | 0 | 6 | 16 | 640 nS | 400 nS |
| 1 | 0 | 1 | 7 | 20 | 800 nS | 500 nS |
| 1 | 1 | 0 | 8 | 24 | 960 nS | 600 nS |
| 1 | 1 | 1 | 9 | 28 | 1120 nS | 700 nS |

Table 8-2: Data Memory Cycle Stretch Values

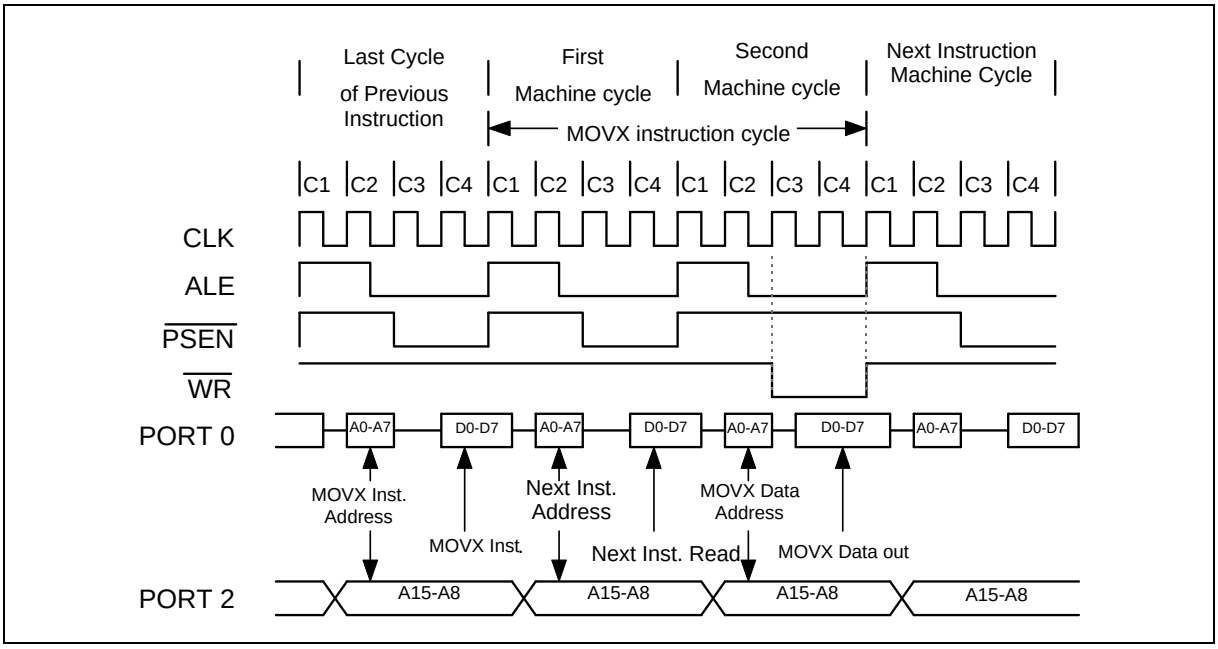



Figure 8-6: Data Memory Write with Stretch Value = 0

### 12.1.3 Mode 1

Mode 1 is the same as Mode 0, except that the timer/counter is 16 bits, instead of 13 bits.

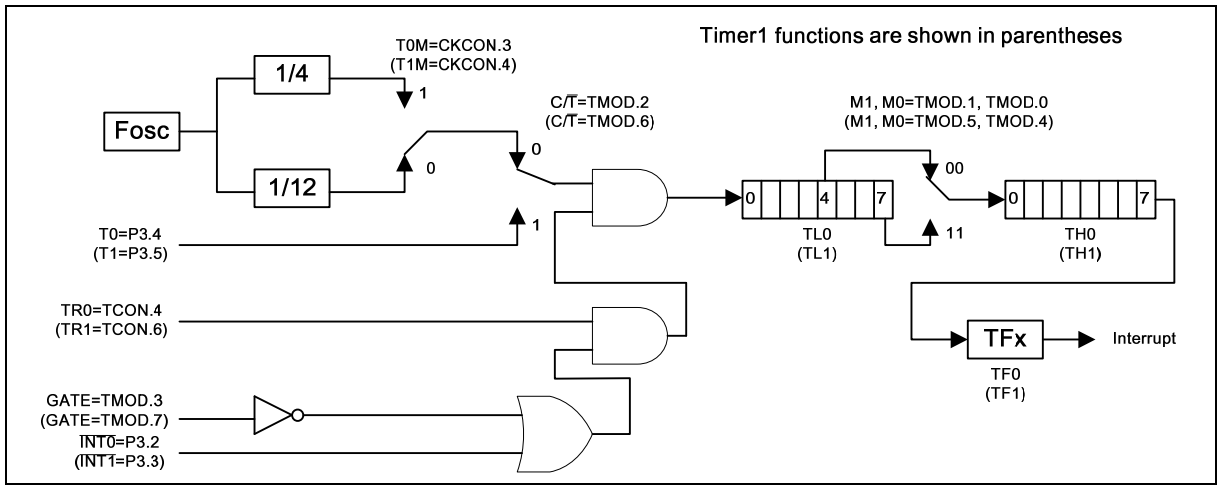



Figure 12-1: Timer/Counters 0 & 1 in Mode 0 and Mode 1

### 12.1.4 Mode 2

In Mode 2, the timer/counter is in the Auto Reload Mode. In this mode, TLx acts as an 8 bits count register, while THx holds the reload value. When the TLx register overflows from FFh to 00h, the TFx bit in TCON is set and TLx is reloaded with the contents of THx, and the counting process continues. The reload operation leaves the contents of the THx register unchanged. Counting is enabled by the TRx bit and proper setting of GATE and $\overline{INTx}$ pins. As in the other two modes 0 and 1, mode 2 allows counting of either clock cycles (clock/12 or clock/4) or pulses on pin Tn.

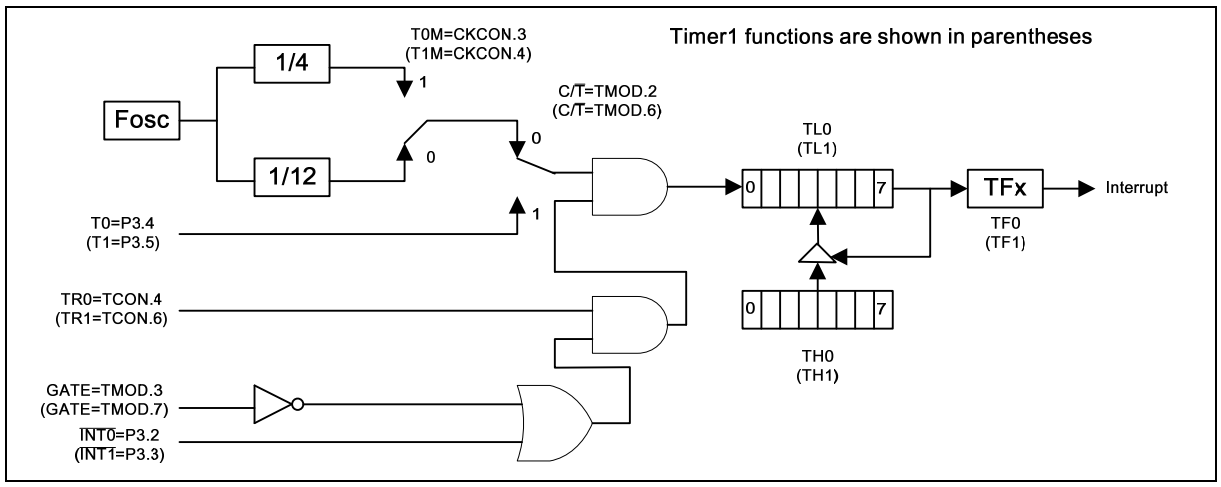



Figure 12-2: Timer/Counter 0 & 1 in Mode 2

## 14.2 PWM Control Registers

The overall functioning of the PWM module is controlled by the contents of the PWMCON1 register. The operation of most of the control bits is straightforward. For example PWM0I is an invert bit for each output which causes results in the output to have the opposite value compared to its non-inverted output. The transfer of the data from the Counter and Compare registers to the control registers is controlled by the PWMCON1.6 (load) while PWMCON1.7 (PWMRUN) allows the PWM to be either in the run or idle state.

If the Brake pin is not used to control the brake function, the “Brake when PWM is not running” function can be used to cause the outputs to have a given state when the PWM is halted. This approach should be used only in time critical situations when there is not sufficient time to use the approach outlined above, since going from the Brake state to run without causing an undefined state on the outputs is not straightforward. A discussion on this topic is included in the section on PWMCON2.

The Brake function, which is controlled by the contents of the PWMCON2 register, is somewhat unique. In general, when Brake is asserted, the eight PWM outputs are forced to a user selected state, namely the state selected by PWMCON3. As shown in the description of the operation of the PWMCON2 register, if PWMCON2.4, BKEN, is a “1” brake is asserted under the control PWMCON2.7, BKCH, and PWMCON2.5, BPEN. As shown, if both are a “0”, brake is asserted. If PWMCON2.7 is a “1”, brake is asserted when the PWMRUN bit, PWMCON1.7, is a “0”. If PWMCON2.6, BKPS, is a “1”, brake is asserted when the Brake Pin, P1.1, has the same polarity as PWMCON2.6. When brake is asserted in response to this pin, the PWMRUN bit in PWMCON1.7 is automatically cleared, and BKF (PWMCON4.0) flag will be set. When both BKCH and BPEN are “1”, BKF will be set when Brake pin is asserted, but PWM generator continues to run. With this special condition, the PWM output does not follow PWMnB, instead it output continuously as per normal without affected by the brake.

Since the Brake Pin being asserted will automatically clear the PWMRUN (PWMCON1.7) and BKF (PWMCON4.0) flag will be set, the user program can poll this bit or enable PWM’s brake interrupt to determine when the Brake Pin causes a brake to occur. The other method for detecting a brake caused by the Brake Pin would be to tie the Brake Pin to one of the external interrupt pins. This latter approach is needed if the Brake signal is of insufficient length to ensure that it can be captured by a polling routine. When, after being asserted, the condition causing the brake is removed, the PWM outputs go to whatever state that had immediately prior to the brake. This means that in order to go from brake being asserted to having the PWM run without going through an indeterminate state, care must be taken. If the Brake Pin causes brake to be asserted, the following prototype code will allow the PWM to go from brake to run smoothly by software polling BKF flag or enable PWM’s interrupt.

• Rewrite PWMCON2 to change from Brake Pin enabled to S/W Brake.

• Write PWM (0, 2, 4, 6) Compare register to always “1”, FFFh, or always “0”, 000h, to initialize PWM output to a High or Low, respectively.

• Clear BKF flag.

• Set PWMCON1 to enable PWMRUN and Load.

• Poll Brake Pin until it is no longer active.

• Poll PWMCON1 to find that Load Bit PWMCON1.6 is “0”. When “0”:

• Write PWMP (0, 2, 4, and 6) Counter register for desired pulse widths and counter reload values.

• Set PWMCON1 to Run and Transfer.

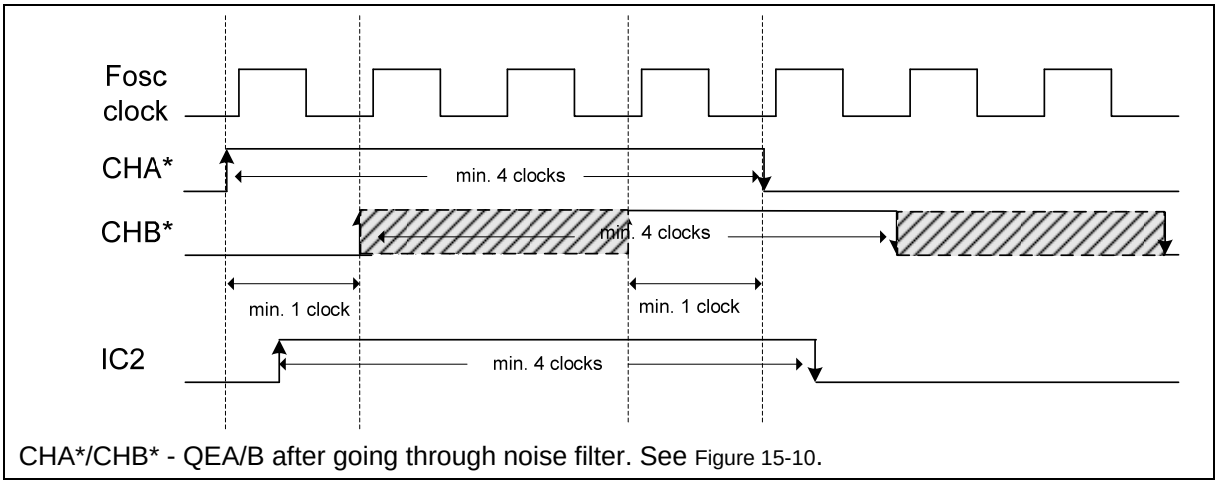



Figure 15-11: QEA/QEB/IC2 timing requirement.

### 15.2.1 Free-counting mode

Pulse counter up or down counts according to direction index (DIR). When overflow or underflow occurs, it sets flag QEIF.

### 15.2.2 Compare-counting mode

Pulse counter up or down counts according to direction index (DIR). On up counting, QEIF will be asserted when PLSCNT overflows from MAXCNT to zero on the next QEA edge for x2 counting mode, and on QEA/QEB edge for x4 counting mode. On down counting, QEIF will be asserted when PLSCNT underflows from zero to MAXCNT on the next QEA edge for x2 counting mode, and on QEA/QEB edge for x4 counting mode. This mode provides the position of a rotor to user. If a quadrature encoder output 1024 pulses to QEA per round, user can write MAXCNT with 4095 in x4 mode or 2047 in x2 mode and reset PLSCNT at initial before rotor runs. When the PLSCNT reaches MAXCNT, it means rotor runs one round on next QEA edge.

### 15.2.3 X2/X4 Counting modes

In X2 counting mode, the pulse counter increases or decreases one on every QEA edge based on the phase relationship of QEA and QEB signals, however:-

In X4 counting mode, the pulse counter increases or decreases one on every QEA and QEB edge based on the phase relationship of QEA and QEB signals.

### 15.2.4 Direction of Count

If QEA lead QEB, the pulse counter is increased by 1. If QEA lags QEB, the pulse counter is decreased by 1. The QEI control logic generates a signal that sets the DIR bit (QEICON.3); this in turn determines the direction of the count. When QEA leads QEB, DIR is set (= 1), and the position counter increments on every active edge. When QEA lags QEB, DIR is cleared, and the position counter decrements on every active edge. Refer to below table.

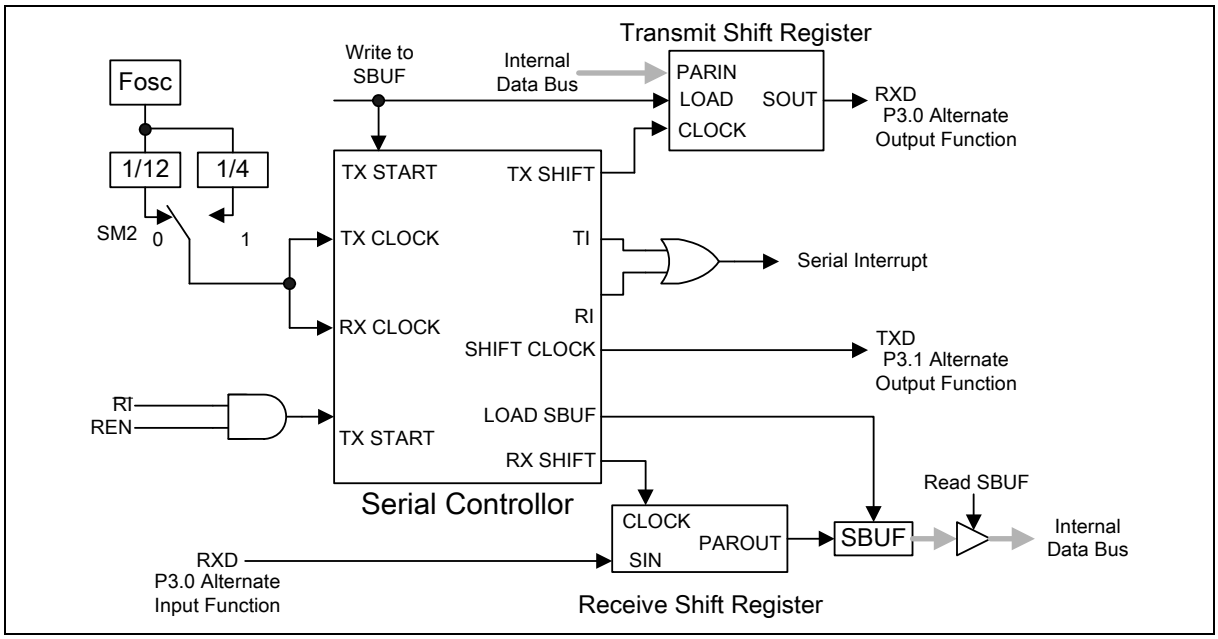
## 16. SERIAL PORT

The W79E22X SERIES has two enhanced serial ports that are functionally similar to the serial port of the original 8052 family. Both the serial ports are full-duplex ports, and the W79E22X SERIES provides additional features, such as Frame Error Detection and Automatic Address Recognition. The serial port 0 can use Timer 1 or 2 as baud rate generator, but the serial port 1 only uses Timer 1 as baud rate generator. However, note that if both serial ports are enabled the baud rate setting control of UART1 is also from the setting of UART0. The serial ports are capable of synchronous and asynchronous communication. In synchronous mode, the W79E22X SERIES generates the clock and operates in half-duplex mode. In asynchronous mode, the serial ports can simultaneously transmit and receive data. The transmit registers and the receive buffers are both addressed as SBUF (SBUF1 for the second serial port), but any write to SBUF/SBUF1 writes to the transmit register while any read from SBUF/SBUF1 reads from the receive buffer. Both serial ports can operate in four modes, as described below. The descriptions are for serial port 0, however, it also apply to the second serial port.

### 16.1 Mode 0

This mode provides half-duplex, synchronous communication with external devices. In this mode, serial data is transmitted and received on the RXD line, and the W79E22X SERIES provides the shift clock on TxD, whether the device is transmitting or receiving. Eight bits are transmitted or received per frame, LSB first. The baud rate is 1/12 or 1/4 of the oscillator frequency, as determined by the SM2 bit (SCON.5; 0 = 1/12; 1 = 1/4). This programmable baud rate is the only difference between the standard 8051/52 and the W79E22X SERIES in mode 0.

Any write to SBUF starts transmission. The shift clock is activated, and data is shifted out on RxD until all eight bits are transmitted. If SM2 is 1, the data appears on RxD one clock period before the falling edge of the shift clock on TxD. Then, the clock remains low for two clock periods before going high again. If SM2 is 0, the data appears on RxD three clock periods before the falling edge of the shift clock on TxD, and the clock on TxD remains low for six clock periods before going high again. This ensures that, at the receiving end, the data on the RxD line can be clocked on the rising edge of the shift clock or latched when the clock is low. The TI flag is set high in C1 following the end of transmission. The functional block diagram is shown below.



| | |
|---|---|
| STA | I2C START Flag. Setting STA to logic 1 to enter master mode, the I2C hardware sends a START or repeat START condition to bus when the bus is free. |
| STO | I2C STOP Flag. In master mode, setting STO to transmit a STOP condition to bus then I2C hardware will check the bus condition if a STOP condition is detected this flag will be cleared by hardware automatically. In a slave mode, setting STO resets I2C hardware to the "not addressed slave mode". |
| SI | I2C Port 1 Interrupt Flag. When a new SIO state is present in the S1STA register, the SI flag is set by hardware, and if the EA and EI2C1 bits are both set, the I2C1 interrupt is requested. SI must be cleared by software. |
| AA | Assert Acknowledge control bit. When AA=1 prior to address or data received, an acknowledged (low level to SDA) will be returned during the acknowledge clock pulse on the SCL line when; 1.) A slave is acknowledging the address sent from master, 2.) The receiver devices are acknowledging the data sent by transmitter. When AA=0 prior to address or data received, a not acknowledged (high level to SDA) will be returned during the acknowledge clock pulse on the SCL line. |
| I2CIN | By default it is zero and input are allows to come in through SDA pin. As when it is 1 input is disallow and to prevent leakage current. During Power-Down mode input is disallow. |

### 17.2.4 Status Register, I2STATUS

I2STATUS is an 8-bit read-only register. The three least significant bits are always 0. The five most significant bits contain the status code. There are 23 possible status codes. When I2STATUS contains F8H, no serial interrupt is requested. All other I2STATUS values correspond to defined I2C ports states. When each of these states is entered, a status interrupt is requested (SI = 1). A valid status code is present in I2STATUS one machine cycle after SI is set by hardware and is still present one machine cycle after SI has been reset by software.

In addition, state 00H stands for a Bus Error. A Bus Error occurs when a START or STOP condition is present at an illegal position in the format frame. Examples of illegal positions are during the serial transfer of an address byte, a data byte or an acknowledge bit.

### 17.2.5 I2C Clock Baud Rate Control, I2CLK

The data baud rate of I2C is determines by I2CLK register when I2C port is in a master mode. It is not important when I2C port is in a slave mode. In the slave modes, SIO will automatically synchronize with any clock frequency up to 400 KHz from master I2C device.

The data baud rate of I2C setting conforms to the following equation.

Data Baud Rate of I2C = $F_{CPU}$ / (I2CLK + 1), where $F_{CPU}$ = $F_{OSC}$/4.

For example, if $F_{OSC}$=16MHz, the I2CLK=40(28H), the data baud rate of I2C = (16MHz/4)/(40+1) = 97.56K bits/sec.

### 17.2.6 I2C Time-out Counter, I2Timer

In W79E22X SERIES, the I2C logic block provides a 14-bit timer-out counter that helps user to deal with bus pending problem. When SI is cleared user can set ENTI=1 to start the time-out counter. If I2C bus is pended too long to get any valid signal from devices on bus, the time-out counter overflows cause TIF=1 to request an I2C interrupt. The I2C interrupt is requested in the condition of either SI=1 or TIF=1. Flags SI and TIF must be cleared by software.

### 17.2.7 I2C Maskable Slave Address

This register enables the Automatic Address Recognition feature of the I2C. When a bit in the I2CSADEN is set to 1, the same bit location in I2CSADDR1 will be compared with the incoming serial

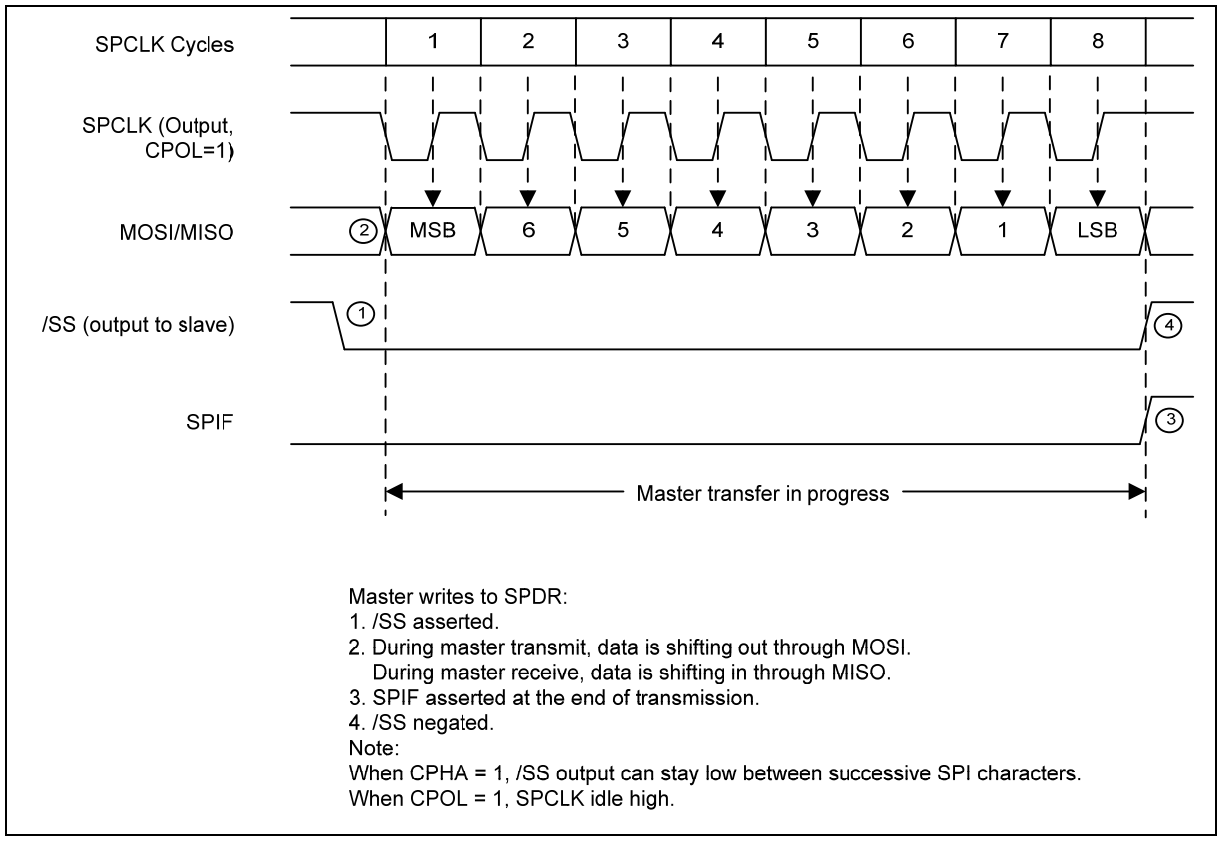



Figure 18-5: Master Mode Transmission (CPOL = 1, CPHA = 1)

### 18.3.2 Slave Mode

When in slave mode, the SPCLK pin becomes input and it will be clock by another master SPI device. The $\overline{SS}$ pin also becomes input. Similarly, before data transmissions occurs, and remain low until the transmission completed. If $\overline{SS}$ goes high, the SPI is forced into idle state. If the $\overline{SS}$ is forced to high at the middle of transmission, the transmission will be aborted and the receiving shifter buffer will be high and goes into idle states.

Data flows from master to slave on MOSI pin and flows from slave to master on MISO pin. The SPDR is used when transmitting or receiving data on the serial bus. Only a write to this register initiates transmission or reception of a byte, and this only occurs in the master device. At the completion of transferring a byte of data, the SPIF status bit is set in both the master and slave devices.

A read of the SPDR is actually a read of a buffer. To prevent an overrun and the loss of the byte that caused the overrun, the first SPIF must be cleared by the time a second transfer of data from the shift register to the read buffer is initiated.

## 20. TIMED ACCESS PROTECTION

The W79E22X SERIES has features like the Watchdog Timer, wait-state control signal and power-on/fail reset flag that are crucial to the proper operation of the system. If these features are unprotected, errant code may write critical control bits, resulting in incorrect operation and loss of control. To prevent this, the W79E22X SERIES provides has a timed-access protection scheme that controls write access to critical bits.

In this scheme, protected bits have a timed write-enable window. A write is successful only if this window is active; otherwise, the write is discarded. The write-enable window is opened in two steps. First, the software writes AAh to the Timed Access (TA) register. This starts a counter, which expires in three machine cycles. Then, if the software writes 55h to the TA register before the counter expires, the write-enable window is opened for three machine cycles. After three machine cycles, the window automatically closes, and the procedure must be repeated again to access protected bits.

The suggested code for opening the write-enable window is;

```
TA      REG     0C7h                ; Define new register TA, located at 0C7h
        MOV     TA, #0AAh
        MOV     TA, #055h
```

Five examples, some correct and some incorrect, of using timed-access protection are shown below.

Example 1: Valid access

```
        MOV     TA, #0AAh           ; 3 M/C ; Note: M/C = Machine Cycles
        MOV     TA, #055h           ; 3 M/C
        MOV     WDCON, #00h         ; 3 M/C
```

Example 2: Valid access

```
        MOV     TA, #0AAh           ; 3 M/C
        MOV     TA, #055h           ; 3 M/C
        NOP                         ; 1 M/C
        SETB    EWT                 ; 2 M/C
```

Example 3: Valid access

```
        MOV     TA, #0Aah           ; 3 M/C
        MOV     TA, #055h           ; 3 M/C
        ORL     WDCON, #00000010B   ; 3M/C
```

Example 4: Invalid access

```
        MOV     TA, #0AAh           ; 3 M/C
        MOV     TA, #055h           ; 3 M/C
        NOP                         ; 1 M/C
        NOP                         ; 1 M/C
        CLR     POR                 ; 2 M/C
```

```
TA              EQU             C7H
SFRAL           EQU             ACH
SFRAH           EQU             ADH
SFRFD           EQU             AEH
SFRCN           EQU             AFH

ORG 000H
LJMP 100H                               ; JUMP TO MAIN PROGRAM
;*****************************************************************************
;* 1. TIMER0 SERVICE VECTOR ORG = 0BH
;*****************************************************************************
ORG 000B

CLR   TR0                               ; TR0 = 0, STOP TIMER0
MOV TL0, R6
MOV TH0, R7
RETI
;*****************************************************************************
;*  4KB LDFlash MAIN PROGRAM
;*****************************************************************************
ORG 100H

MAIN_4K:
MOV TA, #AAH
MOV TA, #55H
MOV CHPCON, #03H                        ; CHPCON = 03H, ENABLE IN-SYSTEM PROGRAMMING.

MOV SFRCN, #0H
MOV TCON, #00H                          ; TCON = 00H, TR = 0 TIMER0 STOP
MOV TMOD, #01H                          ; TMOD = 01H, SET TIMER0 A 16BIT TIMER
MOV IP, #00H                            ; IP = 00H
MOV IE, #82H                            ; IE = 82H, TIMER0 INTERRUPT ENABLED
MOV R6, #F0H
MOV R7, #FFH
MOV TL0, R6
MOV TH0, R7
MOV TCON, #10H                          ; TCON = 10H, TR0 = 1, GO
MOV PCON, #01H                          ; ENTER IDLE MODE

UPDATE_APFlash:
MOV TCON, #00H                          ; TCON = 00H, TR = 0 TIM0 STOP
MOV IP, #00H                            ; IP = 00H
MOV IE, #82H                            ; IE = 82H, TIMER0 INTERRUPT ENABLED
```