**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 | 25MHz |
| Connectivity | I²C, SPI, UART/USART |
| Peripherals | Brown-out Detect/Reset, I²S, POR, PWM, WDT |
| Number of I/O | 18 |
| Program Memory Size | 18KB (18K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | 4K x 8 |
| RAM Size | 512 x 8 |
| Voltage - Supply (Vcc/Vdd) | 2.4V ~ 5.5V |
| Data Converters | A/D 1x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 105°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 20-TSSOP (0.173", 4.40mm Width) |
| Supplier Device Package | 20-TSSOP |
| Purchase URL | https://www.e-xfl.com/product-detail/nuvoton-technology-corporation-america/n76e885at20 |

- Part numbers and packages:

| Part Number | APROM | LDROM | Package |
|---|---|---|---|
| N76E885AT28 | 18K Bytes shared with LDROM | Up to 4K Bytes | TSSOP-28 |
| N76E885AT20 | | | TSSOP-20 |

**Table 6–2. SFR Definitions and Reset Values**

| Symbol | Definition | Address /(Page) | MSB | | | | | | | LSB[1] | Reset Value[2] |
|---|---|---|---|---|---|---|---|---|---|---|---|
| BODCON0[4] | Brown-out detection control 0 | A3H | BODEN[5] | BOV[2:0][5] | | | BOF[6] | BORST[5] | BORF | BOS[7] | POR, CCCC XC0Xb BOD, UUUU XU1Xb Others, UUUU XUUXb |
| AUXR1 | Auxiliary register 1 | A2H | SWRF | RSTPINF | T1LXTM | T0LXTM | GF2 | UART0PX | 0 | DPS | POR, 0000 0000b Software, 1U00 0000b $\overline{RST}$ pin, U100 0000b Others, UU00 0000b |
| P2 | Port 2 | A0H | (A7) 0 | (A6) P2.6 | (A5) P2.5 | (A4) P2.4 | (A3) P2.3 | (A2) P2.2 | (A1) P2.1 | (A0) P2.0 | Output latch, 0111 1111b Input, 0XXX XXXXb[3] |
| CHPCON[4] | Chip control | 9FH | SWRST | IAPFF | - | - | - | - | BS[5] | IAPEN | Software, 0000 00U0b Others, 0000 00C0b |
| EIE1 | Extensive interrupt enable 1 | 9CH | - | - | - | - | - | EWKT | ET3 | ES_1 | 0000 0000b |
| EIE | Extensive interrupt enable | 9BH | ET2 | ESPI | EFB | EWDT | EPWM | ECAP | EPI | EI2C | 0000 0000b |
| SBUF_1 | Serial port 1 data buffer | 9AH | SBUF_1[7:0] | | | | | | | | 0000 0000b |
| SBUF | Serial port 0 data buffer | 99H | SBUF[7:0] | | | | | | | | 0000 0000b |
| SCON | Serial port 0 control | 98H | (9F) SM0/FE | (9E) SM1 | (9D) SM2 | (9C) REN | (9B) TB8 | (9A) RB8 | (99) TI | (98) RI | 0000 0000b |
| CKEN[4] | Clock enable | 97H | EXTEN[1:0] | | HIRCEN | - | - | - | - | CKSWTF | 0011 0000b |
| CKSWT[4] | Clock switch | 96H | HXTST | LXTST | HIRCST | - | ECLKST | OSC[1:0] | | - | 0011 0000b |
| CKDIV | Clock divider | 95H | CKDIV[7:0] | | | | | | | | 0000 0000b |
| CAPCON2 | Input capture control 2 | 94H | - | ENF2 | ENF1 | ENF0 | - | - | - | - | 0000 0000b |
| CAPCON1 | Input capture control 1 | 93H | - | - | CAP2LS[1:0] | | CAP1LS[1:0] | | CAP0LS[1:0] | | 0000 0000b |
| CAPCON0 | Input capture control 0 | 92H | - | CAPEN2 | CAPEN1 | CAPEN0 | - | CAPF2 | CAPF1 | CAPF0 | 0000 0000b |
| SFRS[4] | SFR page selection | 91H | - | - | - | - | - | - | - | SFRPSEL | 0000 0000b |
| P1 | Port 1 | 90H | (97) 0 | (96) 0 | (95) 0 | (94) 0 | (93) 0 | (92) P1.2 | (91) P1.1 | (90) P1.0 | Output latch, 0000 0111b Input, 0000 0XXXb[3] |
| WKCON | Self Wake-up Timer control | 8FH | - | - | WKTCK | WKTF | WKTR | WKPS[2:0] | | | 0000 0000b |
| CKCON | Clock control | 8EH | - | PWMCKS | - | T1M | T0M | - | - | - | 0000 0000b |
| TH1 | Timer 1 high byte | 8DH | TH1[7:0] | | | | | | | | 0000 0000b |
| TH0 | Timer 0 high byte | 8CH | TH0[7:0] | | | | | | | | 0000 0000b |
| TL1 | Timer 1 low byte | 8BH | TL1[7:0] | | | | | | | | 0000 0000b |
| TL0 | Timer 0 low byte | 8AH | TL0[7:0] | | | | | | | | 0000 0000b |
| TMOD | Timer 0 and 1 mode | 89H | GATE | C/$\overline{T}$ | M1 | M0 | GATE | C/$\overline{T}$ | M1 | M0 | 0000 0000b |
| TCON | Timer 0 and 1control | 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 | - | POF | GF1 | GF0 | PD | IDL | POR, 0001 0000b Others, 000U 0000b |
| RWK | Self Wake-up Timer reload byte | 86H | RWK[7:0] | | | | | | | | 0000 0000b |
| DPH | Data pointer high byte | 83H | DPTR[15:8] | | | | | | | | 0000 0000b |
| DPL | Data pointer low byte | 82H | DPTR[7:0] | | | | | | | | 0000 0000b |
| SP | Stack pointer | 81H | SP[7:0] | | | | | | | | 0000 0111b |
| P0 | Port 0 | 80H | (87) P0.7 | (86) P0.6 | (85) P0.5 | (84) P0.4 | (83) P0.3 | (82) P0.2 | (81) P0.1 | (80) P0.0 | Output latch, 1111 1111b Input, XXXX XXXXb[3] |

**[1]** ( ) item means the bit address in bit-addressable SFRs.
**[2]** Reset value symbol description. 0: logic 0; 1: logic 1; U: unchanged; C: see **[5]**; X: see **[3]**, **[6]**, and **[7]**.
**[3]** All I/O pins are default input-only mode (floating) after reset. Reading back P1.2 is always 0 if RPD (CONFIG0.2) remains un-programmed 1.
**[4]** These SFRs have TA protected writing.

# 8. I/O PORT STRUCTURE AND OPERATION

The N76E885 has a maximum of 26 bit-addressable general I/O pins grouped as 4 ports, P0 to P3. Each port has its port control register (Px register). The writing and reading of a port control register have different meanings. A write to port control register sets the port output latch logic value, whereas a read gets the port pin logic state. All I/O pins except P1.2 can be configured individually as one of four I/O modes by software. These four modes are quasi-bidirectional (standard 8051 port structure), push-pull, input-only, and open-drain modes. Each port spends two special function registers PxM1 and PxM2 to select the I/O mode of port Px. The list below illustrates how to select the I/O mode of Px.n. Note that the default configuration of is input-only (high-impedance) after any reset.

**Table 8–1. Configuration for Different I/O Modes**

| PxM1.n | PxM2.n | I/O Type |
|---|---|---|
| 0 | 0 | Quasi-bidirectional |
| 0 | 1 | Push-pull |
| 1 | 0 | Input-only (high-impedance) |
| 1 | 1 | Open-drain |

All I/O pins can be selected as TTL level inputs or Schmitt triggered inputs by selecting corresponding bit in PxS register. Schmitt triggered input has better glitch suppression capability. All I/O pins also have bit-controllable, slew rate select ability via software. The control registers are PxSR. By default, the slew rate is slow. If user would like to increase the I/O output speed, setting the corresponding bit in PxSR, the slew rate is selected in a faster level.

There are five I/O pins those support high sink current including P0.1, P0.2, P0.3, P2.0, and P2.1. By default they have the same sink capability as other I/O pins. By setting PxnSNK, their independent bits in P1S register, they can be individually configured as high sink capability. It is suitable to drive LED or large loading without BJT devices. Note that setting PxnSNK bit only increases the sink capability but the source capability remains the same.

P1.2 is configured as an input-only pin when programming RPD (CONFIG0.2) as 0. Meanwhile, P1.2 is permanent in input-only mode and Schmitt triggered type. P1.2 also has an internal pull-up enabled by P12UP (P1M2.2). If RPD remains un-programmed, P1.2 pin functions as an external reset pin and P1.2 is not available. A read of P1.2 bit is always 0. Meanwhile, the internal pull-up is always enabled.

## 8.1 Quasi-Bidirectional Mode

The quasi-bidirectional mode, as the standard 8051 I/O structure, can rule as both input and output. When the port outputs a logic high, it is weakly driven, allowing an external device to pull the pin low.

### 8.6.1 Input and Output Data Control

These registers are I/O input and output data buffers. Reading gets the I/O input data. Writing forces the data output. All of these registers are bit-addressable.

**P0 – Port 0 (Bit-addressable)**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| P0.7 | P0.6 | P0.5 | P0.4 | P0.3 | P0.2 | P0.1 | P0.0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Address: 80H Reset value: 1111 1111b

| Bit | Name | Description |
|---|---|---|
| 7:0 | P0[7:0] | **Port 0**<br>Port 0 is an maximum 8-bit general purpose I/O port. |

**P1 – Port 1 (Bit-addressable)**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | P1.2 | P1.1 | P1.0 |
| R | R | R | R | R | R | R/W | R/W |

Address: 90H Reset value: 1111 1111b

| Bit | Name | Description |
|---|---|---|
| 7:3 | 0 | **Reserved**<br>The bits are always read as 0. |
| 2 | P1.2 | **Port 1 bit 2**<br>P1.2 is an input-only pin when RPD (CONFIG0.2) is programmed as 0. When leaving RPD un-programmed, P1.2 is always read as 0. |
| 1 | P1.1 | **Port 1 bit 1**<br>P1.1 is available when the internal oscillator or the external clock input is used as the system clock. At this moment, P1.1 functions as a general purpose I/O.<br>If the system clock is selected as the external crystal, P1.1 pin functions as XOUT.<br>A write to P1.1 is invalid and P1.1 is always read as 0. |
| 0 | P1.0 | **Port 1 bit 0**<br>P1.0 is available only when the internal oscillator is used as the system clock. At this moment, P1.0 functions as a general purpose I/O.<br>If the system clock is not selected as the internal oscillator, P1.0 pin functions as XIN. A write to P1.1 is invalid and P1.1 is always read as 0. |

**P2 – Port 2 (Bit-addressable)**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0 | P2.6 | P2.5 | P2.4 | P2.3 | P2.2 | P2.1 | P2.0 |
| R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Address: A0H Reset value: 1111 1111b

| Bit | Name | Description |
|---|---|---|
| 7 | 0 | **Reserved**<br>The bits are always read as 0. |
| 6:0 | P2[7:0] | **Port 2**<br>Port 2 is an maximum 7-bit general purpose I/O port. |

## 9.4 Mode 3 (Two Separate 8-Bit Timers)

Mode 3 has different operating methods for Timer 0 and Timer 1. For Timer/Counter 1, Mode 3 simply freezes the counter. Timer/Counter 0, however, configures TL0 and TH0 as two separate 8 bit count registers in this mode. TL0 uses the Timer/Counter 0 control bits C/$\overline{T}$, GATE, TR0, $\overline{INT0}$, and TF0. The TL0 also can be used as a 1-to-0 transition counter on pin T0 as determined by C/$\overline{T}$ (TMOD.2). TH0 is forced as a clock cycle counter and takes over the usage of TR1 and TF1 from Timer/Counter 1. Mode 3 is used in case that an extra 8 bit timer is needed. If Timer/Counter 0 is configured in Mode 3, Timer/Counter 1 can be turned on or off by switching it out of or into its own Mode 3. It can still be used in Modes 0, 1 and 2 although its flexibility is restricted. It no longer has control over its overflow flag TF1 and the enable bit TR1. However Timer 1 can still be used as a Timer/Counter and retains the use of GATE, $\overline{INT1}$ pin, T1M, and T1LXTM. It can be used as a baud rate generator for the serial port or other application not requiring an interrupt.

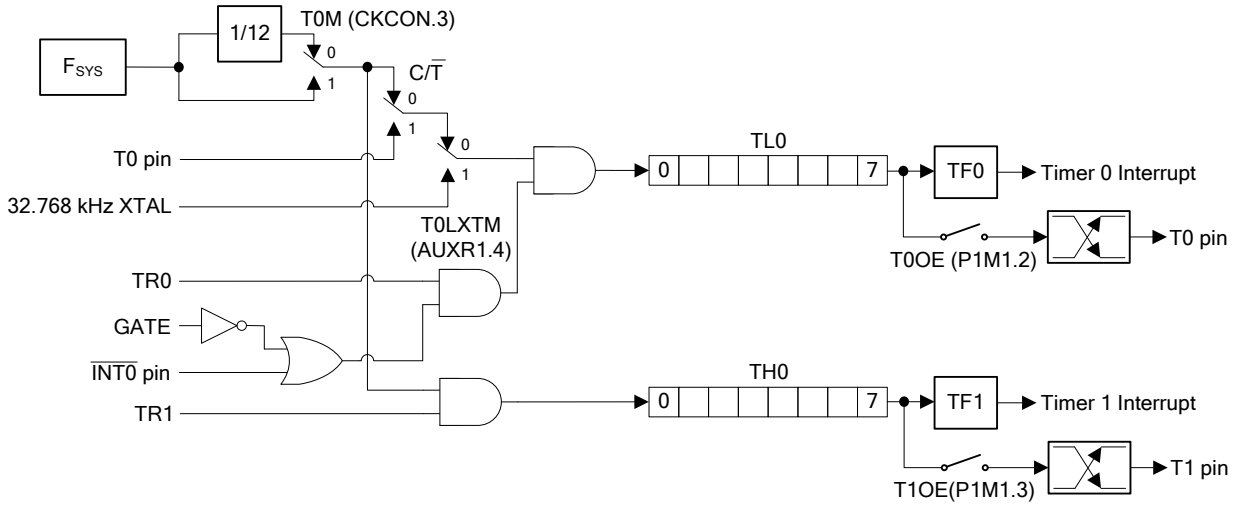


**Figure 9-4. Timer/Counter 0 in Mode 3**

## 11. TIMER 3

Timer 3 is implemented simply as a 16-bit auto-reload, up-counting timer. The user can select the pre-scale with T3PS[2:0] (T3CON[2:0]) and fill the reload value into RH3 and RL3 registers to determine its overflow rate. User then can set TR3 (T3CON.3) to start counting. When the counter rolls over FFFFH, TF3 (T3CON.4) is set as 1 and a reload is generated and causes the contents of the RH3 and RL3 registers to be reloaded into the internal 16-bit counter. If ET3 (EIE1.1) is set as 1, Timer 3 interrupt service routine will be served. TF3 is auto-cleared by hardware after entering its interrupt service routine.

Timer 3 can also be the baud rate clock source of both UARTs. For details, please see Section 14.5 "Baud Rate" on page 70.

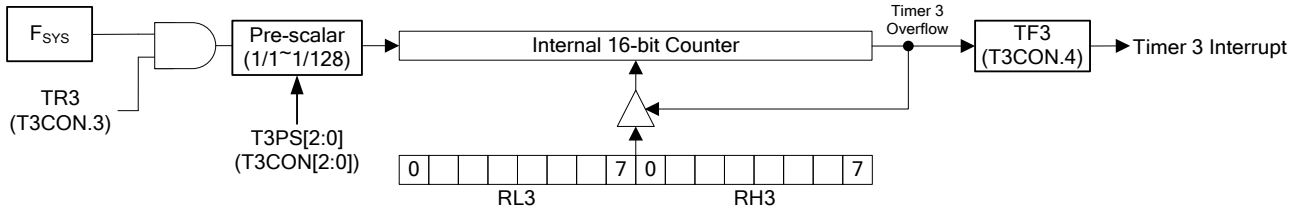


**Figure 11-1. Timer 3 Block Diagram**

**T3CON – Timer 3 Control**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SMOD_1 | SMOD0_1 | BRCK | TF3 | TR3 | T3PS[2:0] | | |
| R/W | R/W | R/W | R/W | R/W | R/W | | |

Address: C4H Reset value: 0000 0000b

| Bit | Name | Description |
|---|---|---|
| 4 | TF3 | **Timer 3 overflow flag**<br>This bit is set when Timer 3 overflows. It is automatically cleared by hardware when the program executes the Timer 3 interrupt service routine. This bit can be set or cleared by software. |
| 3 | TR3 | **Timer 3 run control**<br>0 = Timer 3 is halted.<br>1 = Timer 3 starts running.<br>Note that the reload registers RH3 and RL3 can only be written when Timer 3 is halted (TR3 bit is 0). If any of RH3 or RL3 is written if TR3 is 1, result is unpredictable. |

## 12. WATCHDOG TIMER (WDT)

The N76E885 provides one Watchdog Timer (WDT). It can be configured as a time-out reset timer to reset whole device. Once the device runs in an abnormal status or hangs up by outward interference, a WDT reset recover the system. It provides a system monitor, which improves the reliability of the system. Therefore, WDT is especially useful for system that is susceptible to noise, power glitches, or electrostatic discharge. The WDT also can be configured as a general purpose timer, of which the periodic interrupt serves as an event timer or a durational system supervisor in a monitoring system, which is able to operate during Idle or Power-down mode. WDTEN[3:0] (CONFIG4[7:4]) initialize the WDT to operate as a time-out reset timer or a general purpose timer.

**CONFIG4**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| WDTEN[3:0] | | | | - | - | - | - |
| R/W | | | | - | - | - | - |

Factory default value: 1111 1111b

| Bit | Name | Description |
|---|---|---|
| 7:4 | WDTEN[3:0] | **WDT enable**<br>This field configures the WDT behavior after MCU execution.<br>1111 = WDT is Disabled. WDT can be used as a general purpose timer via software control.<br>0101 = WDT is Enabled as a time-out reset timer and it stops running during Idle or Power-down mode.<br>Others = WDT is Enabled as a time-out reset timer and it keeps running during Idle or Power-down mode. |

The WDT is implemented with a set of divider that divides the low-speed internal oscillator clock nominal 10 kHz. The divider output is selectable and determines the time-out interval. When the time-out interval is fulfilled, it will wake the system up from Idle or Power-down mode and an interrupt event will occur if WDT interrupt is enabled. If WDT is initialized as a time-out reset timer, a system reset will occur after a period of delay if without any software action.

**WDCON – Watchdog Timer Control (TA protected)**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| WDTR | WDCLR | WDTF | WIDPD | WDTRF[1] | WDPS[2:0][2] | | |
| R/W | R/W | R/W | R/W | R/W | R/W | | |

Address: AAH Reset value: see Table 6–2. SFR Definitions and Reset Values

| Bit | Name | Description |
|---|---|---|
| 7 | WDTR | **WDT run**<br>This bit is valid only when control bits in WDTEN[3:0] (CONFIG4[7:4]) are all 1. At this time, WDT works as a general purpose timer.<br>0 = WDT Disabled.<br>1 = WDT Enabled. The WDT counter starts running. |

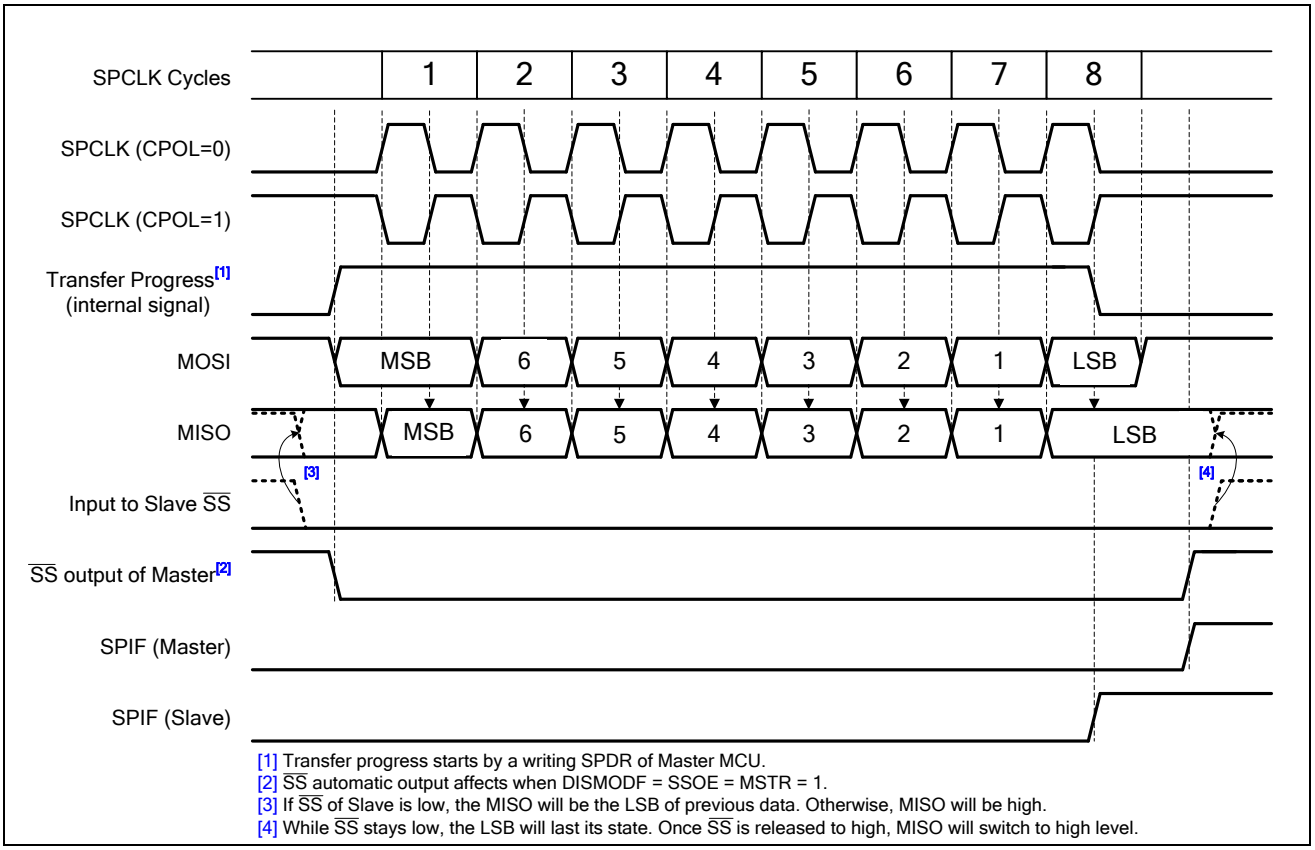


**Figure 15-6. SPI Clock and Data Format with CPHA = 1**

## 15.4 Slave Select Pin Configuration

The N76E885 SPI gives a flexible $\overline{SS}$ pin feature for different system requirements. When the SPI operates as a Slave, $\overline{SS}$ pin always rules as Slave select input. When the Master mode is enabled, $\overline{SS}$ has three different functions according to DISMODF (SPSR.3) and SSOE (SPCR.7). By default, DISMODF is 0. It means that the Mode Fault detection activates. $\overline{SS}$ is configured as a input pin to check if the Mode Fault appears. On the contrary, if DISMODF is 1, Mode Fault is inactivated and the SSOE bit takes over to control the function of the $\overline{SS}$ pin. While SSOE is 1, it means the Slave select signal will generate automatically to select a Slave device. The $\overline{SS}$ as output pin of the Master usually connects with the $\overline{SS}$ input pin of the Slave device. The $\overline{SS}$ output automatically goes low for each transmission when selecting external Slave device and goes high during each idle state to de-select the Slave device. While SSOE is 0 and DISMODF is 1, $\overline{SS}$ is no more used by the SPI and reverts to be a general purpose I/O pin.

addressed by its own address with the data direction bit “write” (SLA+W). The slave receiver mode may also be entered if arbitration is lost.

After the slave is addressed by SLA+W, it should clear its SI flag to receive the data from the master transmitter. If the AA bit is 0 during a transaction, the slave will return a non-acknowledge after the next received data byte. The slave will also become not addressed and isolate with the master. It cannot receive any byte of data with I2DAT remaining the previous byte of data, which is just received.

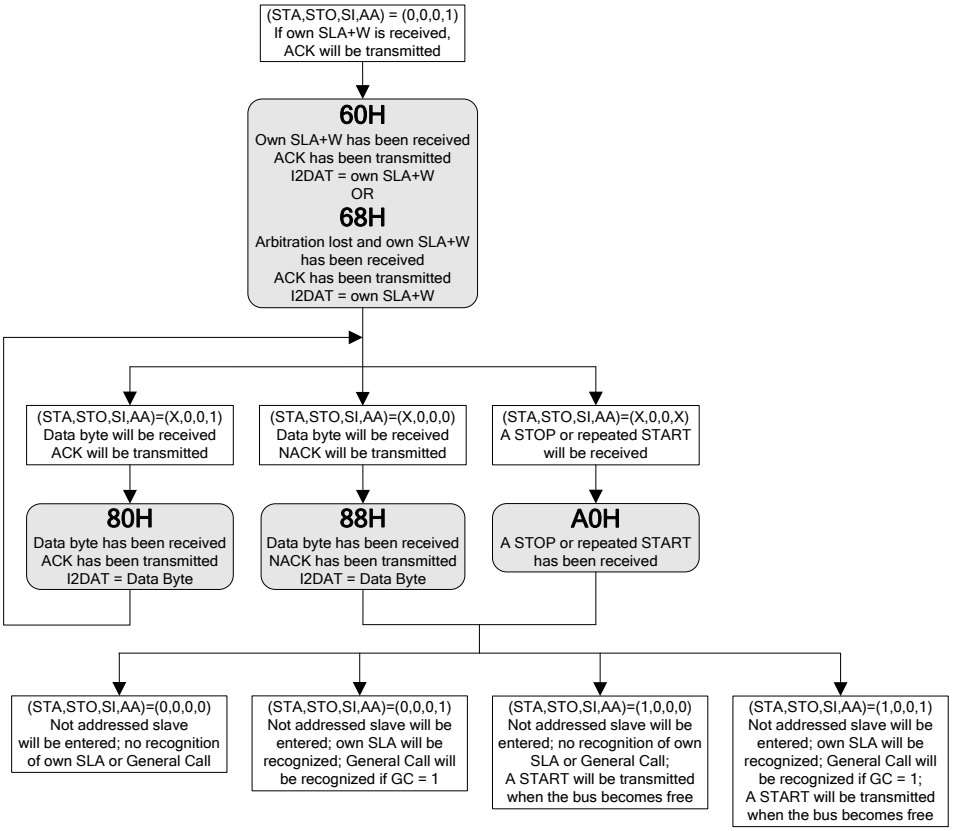


**Figure 16-9. Flow and Status of Slave Receiver Mode**

### 16.3.4 Slave Transmitter Mode

In the slave transmitter mode, several bytes of data are transmitted to a master receiver. After I2ADDR and I2CON values are given, the $I^2C$ wait until it is addressed by its own address with the data direction bit “read” (SLA+R). The slave transmitter mode may also be entered if arbitration is lost.

After the slave is addressed by SLA+R, it should clear its SI flag to transmit the data to the master receiver. Normally the master receiver will return an acknowledge after every byte of data is transmitted by the slave. If the acknowledge is not received, it will transmit all “1” data if it continues the transaction. It becomes a not addressed slave. If the AA flag is cleared during a transaction, the

There is a special case if a START or a repeated START condition is not successfully generated for $I^2C$ bus is obstructed by a low level on SDA line e.g. a slave device out of bit synchronization, the problem can be solved by transmitting additional clock pulses on the SCL line. The $I^2C$ hardware transmits additional clock pulses when the STA bit is set, but no START condition can be generated because the SDA line is pulled low. When the SDA line is eventually released, a normal START condition is transmitted, state 08H is entered, and the serial transaction continues. If a repeated START condition is transmitted while SDA is obstructed low, the $I^2C$ hardware also performs the same action as above. In this case, state 08H is entered instead of 10H after a successful START condition is transmitted. Note that the software is not involved in solving these bus problems.

## 16.4 Typical Structure of $I^2C$ Interrupt Service Routine

The following software example in C language for KEIL™ C51 compiler shows the typical structure of the $I^2C$ interrupt service routine including the 26 state service routines and may be used as a base for user applications. User can follow or modify it for their own application. If one or more of the five modes are not used, the associated state service routines may be removed, but care should be taken that a deleted routine can never be invoked.

```
void I2C_ISR (void) interrupt 6
{
       switch (I2STAT)
       {
              //============================================
              //Bus Error, always put in ISR for noise handling
              //============================================
              case 0x00:                          /*00H, bus error occurs*/
                     STO = 1;                     //recover from bus error
                     break;
              //===========
              //Master Mode
              //===========
              case 0x08:                          /*08H, a START transmitted*/
                     STA = 0;                     //STA  bit  should  be  cleared  by
software
                     I2DAT = SLA_ADDR1;           //load SLA+W/R
                     break;
              case 0x10:                          /*10H, a repeated START transmitted*/
                     STA = 0;
                     I2DAT = SLA_ADDR2;
                     break;
              //=======================
              //Master Transmitter Mode
              //=======================
              case 0x18:                          /*18H,    SLA+W    transmitted,    ACK
received*/
                     I2DAT = NEXT_SEND_DATA1;     //load DATA
                     break;
```

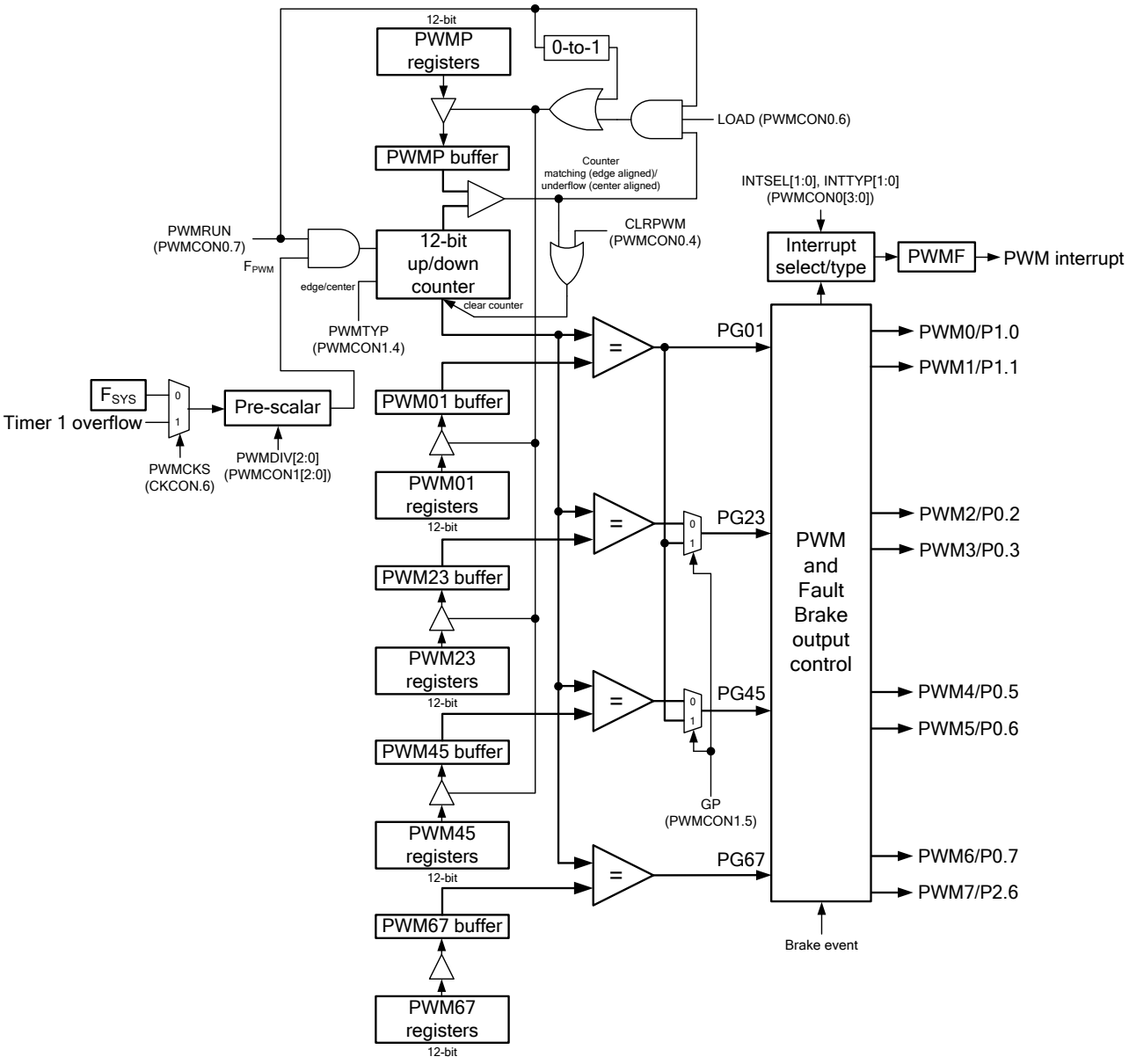


**Figure 18-1. PWM Block Diagram**

The PWM counter generates four PWM signals called PG01, PG23, PG45, and PG67. These signals will go through the PWM and Fault Brake output control circuit. It generates real PWM outputs on I/O pins. The output control circuit determines the PWM mode, dead-time insertion, mask output, Fault Brake control, and PWM polarity. The last stage is a multiplexer of PWM output or I/O function. User should set the PIOn bit to make the corresponding pin function as PWM output. Meanwhile, the general purpose I/O function can be used.

PWM channel or edges of STADC pin will automatically trigger an A/D conversion. (The hardware trigger also sets ADCS by hardware.) The effective condition is selected by ETGSEL (ADCCON0[5:4]) and ETGTYP (ADCCON1[3:2]). A trigger delay can also be inserted between external trigger point and A/D conversion. The external trigging ADC hardware with controllable trigger delay makes the N76E885 feasible for high performance motor control. Note that during ADC is busy in converting (ADCS = 1), any conversion triggered by software or hardware will be ignored and there is no warning presented.

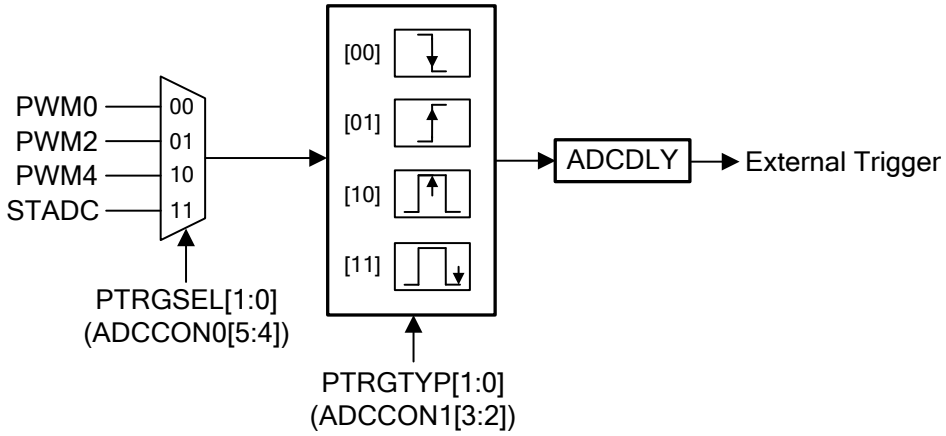


**Figure 19-2. External Triggering ADC Circuit**

### 19.1.3 ADC Conversion Result Comparator

The N76E885 ADC has a digital comparator, which compares the A/D conversion result with a 10-bit constant value given in ACMPH and ACMPL registers. The ADC comparator is enabled by setting ADCMPEN (ADCCON2.5) and each compare will be done on every A/D conversion complete moment. ADCMPO (ADCCON2.4) shows the compare result according to its output polarity setting bit ADCMPOP (ADCCON2.6). The ADC comparing result can trigger a PWM Fault Brake output directly. This function is enabled when ADFBEN (ADCCON2.7). When ADCMPO is set, it generates a ADC compare event and asserts Fault Brake. Please also see Sector 18.1.5 “Fault Brake” on page 122.

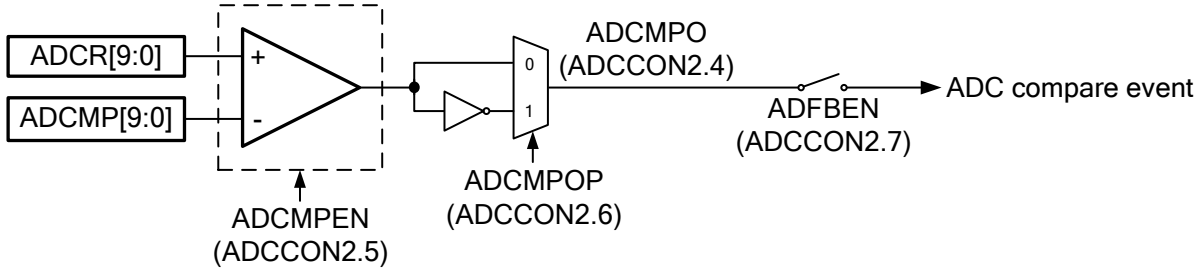


**Figure 19-3. ADC Result Comparator**

minimum reaction time of 5 clock cycles between the interrupt flag being set and the interrupt service routine being executed.

A longer response time should be anticipated if any of the three conditions are not met. If a higher or equal priority is being serviced, then the interrupt latency time obviously depends on the nature of the service routine currently being executed. If the polling cycle is not the last clock cycle of the instruction being executed, then an additional delay is introduced. The maximum response time (if no other interrupt is currently being serviced or the new interrupt is of greater priority) occurs if the device is performing a RETI, and then executes a longest 6-clock-cycle instruction as the next instruction. From the time an interrupt source is activated (not detected), the longest reaction time is 16 clock cycles. This period includes 5 clock cycles to complete RETI, 6 clock cycles to complete the longest instruction, 1 clock cycle to detect the interrupt, and 4 clock cycles to complete the hardware LCALL to the interrupt vector location.

Thus in a single-interrupt system the interrupt response time will always be more than 5 clock cycles and not more than 16 clock cycles.

## 21.6 External Interrupt Pins

The external interrupt $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$ can be used as interrupt sources. They are selectable to be either edge or level triggered depending on bits IT0 (TCON.0) and IT1 (TCON.2). The bits IE0 (TCON.1) and IE1 (TCON.3) are the flags those are checked to generate the interrupt. In the edge triggered mode, the $\overline{\text{INT0}}$ or $\overline{\text{INT1}}$ inputs are sampled every system clock cycle. If the sample is high in one cycle and low in the next, then a high to low transition is detected and the interrupts request flag IE0 or IE1 will be set. Since the external interrupts are sampled every system clock, they have to be held high or low for at least one system clock cycle. The IE0 and IE1 are automatically cleared when the interrupt service routine is called. If the level triggered mode is selected, then the requesting source has to hold the pin low till the interrupt is serviced. The IE0 and IE1 will not be cleared by the hardware on entering the service routine. In the level triggered mode, IE0 and IE1 follows the inverse value of $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$ pins. If interrupt pins continue to be held low even after the service routine is completed, the processor will acknowledge another interrupt request from the same source. Both $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$ can wake up the device from the Power-down mode.

```
        MOV     TA,#0AAh
        MOV     TA,#55h
        ORL     CHPCON,#80h                 ;software reset and reboot from APROM

        SJMP    $

;*****************************************************************
;             IAP Subroutine
;*****************************************************************
Enable_IAP:
        MOV     TA,#0AAh                    ;CHPCON is TA protected
        MOV     TA,#55h
        ORL     CHPCON,#00000001b           ;IAPEN = 1, enable IAP mode
        RET

Disable_IAP:
        MOV     TA,#0AAh
        MOV     TA,#55h
        ANL     CHPCON,#11111110b           ;IAPEN = 0, disable IAP mode
        RET

Enable_AP_Update:
        MOV     TA,#0AAh                    ;IAPUEN is TA protected
        MOV     TA,#55h
        ORL     IAPUEN,#00000001b           ;APUEN = 1, enable APROM update
        RET

Disable_AP_Update:
        MOV     TA,#0AAh
        MOV     TA,#55h
        ANL     IAPUEN,#11111110b           ;APUEN = 0, disable APROM update
        RET

Enable_CONFIG_Update:
        MOV     TA,#0AAh
        MOV     TA,#55h
        ORL     IAPUEN,#00000100b           ;CFUEN = 1, enable CONFIG update
        RET

Disable_CONFIG_Update:
        MOV     TA,#0AAh
        MOV     TA,#55h
        ANL     IAPUEN,#11111011b           ;CFUEN = 0, disable CONFIG update
        RET

Trigger_IAP:
        MOV     TA,#0AAh                    ;IAPTRG is TA protected
        MOV     TA,#55h
        ORL     IAPTRG,#00000001b           ;write '1' to IAPGO to trigger IAP process
        RET

;*****************************************************************
;             IAP APROM Function
;*****************************************************************
Erase_AP:
        MOV     IAPCN,#PAGE_ERASE_AP
        MOV     IAPFD,#0FFh
        MOV     R0,#00h
Erase_AP_Loop:
        MOV     IAPAH,R0
        MOV     IAPAL,#00h
        CALL    Trigger_IAP
        MOV     IAPAL,#80h
```

## 23.1 Idle Mode

Idle mode suspends CPU processing by holding the Program Counter. No program code are fetched and run in Idle mode. It forces the CPU state to be frozen. The Program Counter (PC), Stack Pointer (SP), Program Status Word (PSW), Accumulator (ACC), and the other registers hold their contents during Idle mode. The port pins hold the logical states they had at the time Idle was activated. Generally, it saves considerable power of typical half of the full operating power.

Since the clock provided for peripheral function logic circuit like timer or serial port still remain in Idle mode, the CPU can be released from the Idle mode with any of enabled interrupt sources. User can put the device into Idle mode by writing 1 to the bit IDL (PCON.0). The instruction that sets the IDL bit is the last instruction that will be executed before the device enters Idle mode.

The Idle mode can be terminated in two ways. First, as mentioned, any enabled interrupt will cause an exit. It will automatically clear the IDL bit, terminate Idle mode, and the interrupt service routine (ISR) will be executed. After using the RETI instruction to jump out of the ISR, execution of the program will be the one following the instruction, which put the CPU into Idle mode. The second way to terminate Idle mode is with any reset other than software reset. Remember that if Watchdog reset is used to exit Idle mode, the WIDPD (WDCON.4) needs to be set 1 to let WDT keep running in Idle mode.

## 23.2 Power-Down Mode

Power-down mode is the lowest power state that the N76E885 can enter. It remain the power consumption as a ”μA” level by stopping the system clock source. Both of CPU and peripheral functions like Timers or UART are frozen. Flash memory is put into its stop mode. All activity is completely stopped and the power consumption is reduced to the lowest possible value. The device can be put into Power-down mode by writing 1 to bit PD (PCON.1). The instruction that does this action will be the last instruction to be executed before the device enters Power-down mode. In the Power-down mode, RAM maintains its content. The port pins output the values held by their own state before Power-down respectively.

There are several ways to exit the N76E885 from the Power-down mode. The first is with all resets except software reset. Brown-out reset will also wake up CPU from Power-down mode. Be sure that brown-out detection is enabled before the system enters Power-down. However, for least power consumption, it is recommended to enable low power BOD in Power-down mode. Of course the external pin reset and power-on reset will remove the Power-down status. After the external reset or power-on reset. The CPU is initialized and start executing program code from the beginning.

The second way to wake the N76E885 up from the Power-down mode is by an enabled external interrupt. The trigger on the external pin will asynchronously restart the system clock. After oscillator is

successful and then user can also disable the original clock source if power consumption is concerned. Note that if not following the steps above, the hardware will take certain actions to deal with such illegal operations as follows.

1. If user tries to disable the current clock source by changing CKEN value, the device will ignore this action. The system clock will remain the original one and CKEN will remain the original value.

2. If user tries to switch the system clock source to a disabled one by changing OSC[1:0] value, OSC[1:0] value will be updated right away. But the system clock will remain the original one and CKSWTF (CLKEN.0) flag will be set by hardware.

3. Once user switches the system clock source to an enabled but still instable one, the hardware will wait for stabilization of the target clock source and then switch to it in the background. During this waiting period, the device will continue executing the program with the original clock source and CKSWTF will be set as 1. After the stable flag of the target clock source (see CKSWT[7:3]) is set and the clock source switches successfully, CKSWTF will be cleared as 0 automatically by hardware.

Here is an illustration of switching the clock source from HIRC source to HXT.

```
       MOV    TA,#0AAh                  ;TA protection
       MOV    TA,#55h                   ;
       ORL    CKEN,#10000000b           ;Enable the HXT

;******Polling can be ignored if not disabling the original clock source******
Polling_HXT_stable:                     ;Waiting for the HXT stable
       MOV    A,CKSWT
       JNB    ACC.7, Polling_HXT_stable
;*******************************************************************************

       MOV    TA,#0AAh                  ;TA protection
       MOV    TA,#55h                   ;
       MOV    CKSWT,#02h                ;switch the clock source to the HXT

;******Disable the original HIRC clock source, for example******
       MOV    TA,#0AAh                  ;TA protection
       MOV    TA,#55h                   ;
       ANL    CKEN,#11011111b           ;Disable the IHRC
;*************************************************************
```

**CKSWT – Clock Switch (TA protected)**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| HXTST | LXTST | HIRCST | - | ECLKST | OSC[1:0] | | - |
| R | R | R | - | R | W | | - |

Address: 96H Reset value: 0011 0000b

| Bit | Name | Description |
|---|---|---|
| 7 | HXTST | **High-speed external crystal/resonator 2 MHz to 25 MHz status**<br>0 = High-speed external crystal/resonator is not stable or disabled.<br>1 = High-speed external crystal/resonator is enabled and stable. |

**Table 25–1. BOF Reset Value**

| CBODEN (CONFIG2.7) | CBORST (CONFIG2.2) | $V_{DD}$ Level | BOF |
|---|---|---|---|
| 1 | 1 | > $V_{BOD}$ always | 0 |
| 1 | 0 | < $V_{BOD}$ | 1 |
| 1 | 0 | > $V_{BOD}$ | 0 |
| 0 | X | X | 0 |

**BODCON1 – Brown-out Detection Control 1 (TA protected)**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | LPBOD[1:0] | | BODFLT |
| - | - | - | - | - | R/W | | R/W |

Address: ABH Reset value: see Table 6–2. SFR Definitions and Reset Values

| Bit | Name | Description |
|---|---|---|
| 7:3 | - | **Reserved** |
| 2:1 | LPBOD[1:0] | **Low power BOD enable**<br>00 = BOD normal mode. BOD circuit is always enabled.<br>01 = BOD low power mode 1 by turning on BOD circuit every 1.6 ms periodically.<br>10 = BOD low power mode 2 by turning on BOD circuit every 6.4 ms periodically.<br>11 = BOD low power mode 3 by turning on BOD circuit every 25.6 ms periodically. |
| 0 | BODFLT | **BOD filter control**<br>BOD has a filter which counts 32 clocks of $F_{SYS}$ to filter the power noise when MCU runs with HIRC, HXT, or ECLK as the system clock and BOD does not operates in its low power mode (LPBOD[1:0] = [0, 0]). In other conditions, the filter counts 2 clocks of LIRC.<br>Note that when CPU is halted in Power-down mode. The BOD output is permanently filtered by 2 clocks of LIRC.<br>The BOD filter avoids the power noise to trigger BOD event. This bit controls BOD filter enabled or disabled.<br>0 = BOD filter Disabled.<br>1 = BOD filter Enabled. (Power-on reset default value.) |

BORST (BODCON0.2) setting 1. After a brown-out reset, BORF (BODCON0.1) will be set as 1 via hardware. BORF will not be altered by any reset other than a power-on reset or brown-out reset itself. This bit can be set or cleared by software.

**BODCON0 – Brown-out Detection Control 0 (TA protected)**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| BODEN | BOV[2:0] | | | BOF | BORST | BORF | BOS |
| R/W | R/W | | | R/W | R/W | R/W | R |

Address: A3H Reset value: see Table 6–2. SFR Definitions and Reset Values

| Bit | Name | Description |
|---|---|---|
| 1 | BORF | **Brown-out reset flag**<br>When the MCU is reset by brown-out event, this bit will be set via hardware. This flag is recommended to be cleared via software. |

## 26.3 External Reset

The external reset pin $\overline{RST}$ is an input with a Schmitt trigger. An external reset is accomplished by holding the $\overline{RST}$ pin low for at least 24 system clock cycles to ensure detection of a valid hardware reset signal. The reset circuitry then synchronously applies the internal reset signal. Thus, the reset is a synchronous operation and requires the clock to be running to cause an external reset.

Once the device is in reset condition, it will remain as long as $\overline{RST}$ pin is low. After the $\overline{RST}$ high is removed, the MCU will exit the reset state and begin code executing from address 0000H. If an external reset applies while CPU is in Power-down mode, the way to trigger a hardware reset is slightly different. Since the Power-down mode stops system clock, the reset signal will asynchronously cause the system clock resuming. After the system clock is stable, MCU will enter the reset state.

There is a RSTPINF (AUXR1.6) flag, which indicates an external reset took place. After the external reset, this bit will be set as 1 via hardware. RSTPINF will not change after any reset other than a power-on reset or the external reset itself. This bit can be cleared via software.

**AUXR1 – Auxiliary Register 1**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SWRF | RSTPINF | T1LXTM | T0LXTM | GF2 | UART0PX | 0 | DPS |
| R/W | R/W | R/W | R/W | R/W | R/W | R | R/W |

Address: A2H Reset value: see Table 6–2. SFR Definitions and Reset Values

| Bit | Name | Description |
|---|---|---|
| 6 | RSTPINF | **External reset flag**<br>When the MCU is reset by the external reset, this bit will be set via hardware. It is recommended that the flag be cleared via software. |

**CHPCON – Chip Control (TA protected)**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SWRST | IAPFF | - | - | - | - | BS | IAPEN |
| W | R/W | - | - | - | - | R/W | R/W |

Address: 9FH Reset value: see Table 6–2. SFR Definitions and Reset Values

| Bit | Name | Description |
|---|---|---|
| 7 | SWRST | **Software reset**<br>To set this bit as logic 1 will cause a software reset. It will automatically be cleared via hardware after reset is finished. |

**AUXR1 – Auxiliary Register 1**

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SWRF | RSTPINF | T1LXTM | T0LXTM | GF2 | UART0PX | 0 | DPS |
| R/W | R/W | R/W | R/W | R/W | R/W | R | R/W |

Address: A2H Reset value: see Table 6–2. SFR Definitions and Reset Values

| Bit | Name | Description |
|---|---|---|
| 7 | SWRF | **Software reset flag**<br>When the MCU is reset via software reset, this bit will be set via hardware. It is recommended that the flag be cleared via software. |

The software demo code is listed below.

```
ANL    AUXR1,#01111111b    ;software reset flag clear
...
...
CLR    EA
MOV    TA,#0AAh
MOV    TA,#55h
ORL    CHPCON,#10000000b   ;software reset
```

## 26.6 Boot Select

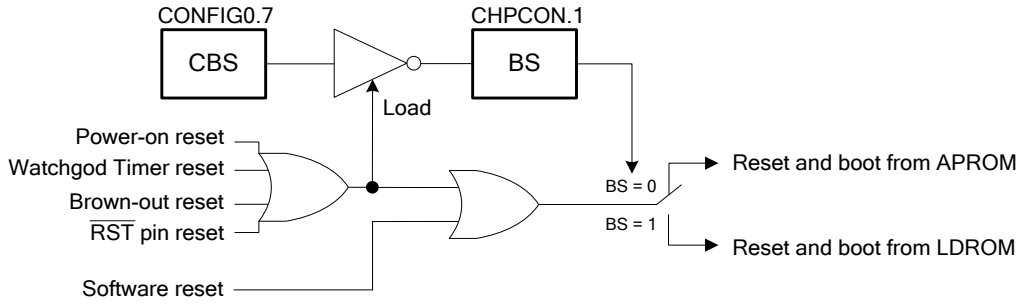


**Figure 26-1. Boot Selecting Diagram**

The N76E885 provides user a flexible boot selection for variant application. The SFR bit BS in CHPCON.1 determines MCU booting from APROM or LDROM after any source of reset. If reset occurs and BS is 0, MCU will reboot from address 0000H of APROM. Else, the CPU will reboot from

# 28. ON-CHIP-DEBUGGER (OCD)

## 28.1 Functional Description

The N76E885 is embedded in an on-chip-debugger (OCD) providing developers with a low cost method for debugging user code, which is available on each package. The OCD gives debug capability of complete program flow control with eight hardware address breakpoints, single step, free running, and non-intrusive commands for memory access. The OCD system does not occupy any locations in the memory map and does not share any on-chip peripherals.

When the OCDEN (CONFIG0.4) is programmed as 0 and LOCK (CONFIG0.1) remains un-programmed as 1, the OCD is activated. The OCD cannot operate if chip is locked. The OCD system uses a two-wire serial interface, OCDDA and OCDCK, to establish communication between the target device and the controlling debugger host. OCDDA is an input/output pin for debug data transfer and OCDCK is an input pin for synchronization with OCDDA data. The P1.2/$\overline{\text{RST}}$ pin is also necessary for OCD mode entry and exit. The N76E885 supports OCD with Flash Memory control path by ICP writer mode, which shares the same three pins of OCD interface.

The N76E885 uses OCDDA, OCDCK, and P1.2/$\overline{\text{RST}}$ pins to interface with the OCD system. When designing a system where OCD will be used, the following restrictions must be considered for correct operation:

1. If P1.2/$\overline{\text{RST}}$ is configured as external reset pin, it cannot be connected directly to $V_{DD}$ and any external capacitors connected must be removed.

2. If P1.2/$\overline{\text{RST}}$ is configured as input pin P1.2, any external input source must be isolated.

3. All external reset sources must be disconnected.

4. Any external component connected on OCDDA and OCDCK must be isolated.

## 28.2 Limitation of OCD

The N76E885 is a fully-featured microcontroller that multiplexes several functions on its limited I/O pins. Some device functionality must be sacrificed to provide resources for OCD system. The OCD has the following limitations:

1. The P1.2/$\overline{\text{RST}}$ pin needs to be used for OCD mode selection. Therefore, neither P1.2 input nor an external reset source can be emulated.

2. The OCDDA pin is physically located on the same pin as P0.0. Therefore, neither its I/O function nor shared multi-functions can be emulated.