

```

hircmap0 = IAPFD;
IAPAL = 0x31;
IAPAH = 0x00;
set_IAPGO;
hircmap1 = IAPFD;
clr_IAPEN;
trimvalue16bit = ((hircmap0<<1)+(hircmap1&0x01));
trimvalue16bit = trimvalue16bit - 15;
hircmap1 = trimvalue16bit&0x01;
hircmap0 = trimvalue16bit>>1;
TA=0XAA;
TA=0X55;
RCTRIM0 = hircmap0;
TA=0XAA;
TA=0X55;
RCTRIM1 = hircmap1;
}

```

13.6 Framing Error Detection

Framing error detection is provided for asynchronous modes. (Mode 1, 2, or 3.) The framing error occurs when a valid stop bit is not detected due to the bus noise or contention. The UART can detect a framing error and notify the software.

The framing error bit, FE, is located in SCON.7. This bit normally serves as SM0. While the framing error accessing enable bit SMOD0 (PCON.6) is set 1, it serves as FE flag. Actually, SM0 and FE locate in different registers.

The FE bit will be set 1 via hardware while a framing error occurs. FE can be checked in UART interrupt service routine if necessary. Note that SMOD0 should be 1 while reading or writing to FE. If FE is set, any following frames received without frame error will not clear the FE flag. The clearing has to be done via software.

13.7 Multiprocessor Communication

The N76E003 multiprocessor communication feature lets a master device send a multiple frame serial message to a slave device in a multi-slave configuration. It does this without interrupting other slave devices that may be on the same serial line. This feature can be used only in UART Mode 2 or 3. User can enable this function by setting SM2 (SCON.5) as logic 1 so that when a byte of frame is received, the serial interrupt will be generated only if the 9th bit is 1. (For Mode 2, the 9th bit is the stop bit.) When the SM2 bit is 1, serial data frames that are received with the 9th bit as 0 do not generate an interrupt. In this case, the 9th bit simply separates the slave address from the serial data.

When the master device wants to transmit a block of data to one of several slaves on a serial line, it first sends out an address byte to identify the target slave. Note that in this case, an address byte differs from a data byte. In an address byte, the 9th bit is 1 and in a data byte, it is 0. The address byte

interrupts all slaves so that each slave can examine the received byte and see if it is addressed by its own slave address. The addressed slave then clears its SM2 bit and prepares to receive incoming data bytes. The SM2 bits of slaves that were not addressed remain set, and they continue operating normally while ignoring the incoming data bytes.

Follow the steps below to configure multiprocessor communications:

1. Set all devices (masters and slaves) to UART Mode 2 or 3.
2. Write the SM2 bit of all the slave devices to 1.
3. The master device's transmission protocol is:
 - First byte: the address, identifying the target slave device, (9th bit = 1).
 - Next bytes: data, (9th bit = 0).
4. When the target slave receives the first byte, all of the slaves are interrupted because the 9th data bit is 1. The targeted slave compares the address byte to its own address and then clears its SM2 bit to receiving incoming data. The other slaves continue operating normally.
5. After all data bytes have been received, set SM2 back to 1 to wait for next address.

SM2 has no effect in Mode 0, and in Mode 1 can be used to check the validity of the stop bit. For Mode 1 reception, if SM2 is 1, the receiving interrupt will not be issue unless a valid stop bit is received.

13.8 Automatic Address Recognition

The automatic address recognition is a feature, which enhances the multiprocessor communication feature by allowing the UART to recognize certain addresses in the serial bit stream by using hardware to make the comparisons. This feature saves a great deal of software overhead by eliminating the need for the software to examine every serial address, which passes by the serial port. Only when the serial port recognizes its own address, the receiver sets RI bit to request an interrupt. The automatic address recognition feature is enabled when the multiprocessor communication feature is enabled, SM2 is set.

If desired, user may enable the automatic address recognition feature in Mode 1. In this configuration, the stop bit takes the place of the ninth data bit. RI is set only when the received command frame address matches the device's address and is terminated by a valid stop bit.

Using the automatic address recognition feature allows a master to selectively communicate with one or more slaves by invoking the “Given” slave address or addresses. All of the slaves may be contacted by using the “Broadcast” address. Two registers are used to define the slave address, SADDR, and the slave address mask, SADEN. SADEN is used to define which bits in the SADDR are to be used and which bits are “don’t care”. The SADDR mask can be logically ANDed with the SADDR to create the “Given” address, which the master will use for addressing each of the slaves. Use of the “Given” address allows multiple slaves to be recognized while excluding others.

SADDR – Slave 0 Address

7	6	5	4	3	2	1	0
SADDR[7:0]							
R/W							

Address: A9H

Reset value: 0000 0000b

Bit	Name	Description
7:0	SADDR[7:0]	Slave 0 address This byte specifies the microcontroller’s own slave address for UATRO multi-processor communication.

SADEN – Slave 0 Address Mask

7	6	5	4	3	2	1	0
SADEN[7:0]							
R/W							

Address: B9H

Reset value: 0000 0000b

Bit	Name	Description
7:0	SADEN[7:0]	Slave 0 address mask This byte is a mask byte of UATRO that contains “don’t-care” bits (defined by zeros) to form the device’s “Given” address. The don’t-care bits provide the flexibility to address one or more slaves at a time.

SADDR_1 – Slave 1 Address

7	6	5	4	3	2	1	0
SADDR_1[7:0]							
R/W							

Address: BBH

Reset value: 0000 0000b

Bit	Name	Description
7:0	SADDR_1[7:0]	Slave 1 address This byte specifies the microcontroller’s own slave address for UART1 multi-processor communication.

SADEN_1 – Slave 1 Address Mask

7	6	5	4	3	2	1	0
SADEN_1[7:0]							
R/W							

Address: BAH

Reset value: 0000 0000b

Bit	Name	Description
7:0	SADEN_1[7:0]	<p>Slave 1 address mask</p> <p>This byte is a mask byte of UART1 that contains “don’t-care” bits (defined by zeros) to form the device’s “Given” address. The don’t-care bits provide the flexibility to address one or more slaves at a time.</p>

The following examples will help to show the versatility of this scheme.

Example 1, slave 0:

SADDR = 11000000b
 SADEN = 11111101b
 Given = 110000X0b

Example 2, slave 1:

SADDR = 11000000b
 SADEN = 11111110b
 Given = 110000XB

In the above example SADDR is the same and the SADEN data is used to differentiate between the two slaves. Slave 0 requires 0 in bit 0 and it ignores bit 1. Slave 1 requires 0 in bit 1 and bit 0 is ignored. A unique address for Slave 0 would be 1100 0010 since slave 1 requires 0 in bit 1. A unique address for slave 1 would be 11000001b since 1 in bit 0 will exclude slave 0. Both slaves can be selected at the same time by an address, which has bit 0 = 0 (for slave 0) and bit 1 = 0 (for slave 1). Thus, both could be addressed with 11000000b as their “Broadcast” address.

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

Example 1, slave 0:

SADDR = 11000000b
 SADEN = 11111001b
 Given = 11000XX0b

Example 2, slave 1:

SADDR = 11100000b
 SADEN = 11111010b
 Given = 11100X0Xb

Example 3, slave 2:

```
SADDR = 11000000b
SADEN = 11111100b
Given = 110000XXb
```

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 11100110b. Slave 1 requires that bit 1 = 0 and it can be uniquely addressed by 11100101b. Slave 2 requires that bit 2 = 0 and its unique address is 11100011b. To select Slaves 0 and 1 and exclude Slave 2 use address 11100100b, since it is necessary to make bit 2 = 1 to exclude slave 2.

The “Broadcast” address for each slave is created by taking the logical OR of SADDR and SADEN. Zeros in this result are treated as “don’t-cares”, e.g.:

```
SADDR = 01010110b
SADEN = 11111100b
Broadcast = 1111111Xb
```

The use of don’t-care bits provides flexibility in defining the Broadcast address, however in most applications, interpreting the “don’t-cares” as all ones, the broadcast address will be FFH.

On reset, SADDR and SADEN are initialized to 00H. This produces a “Given” address of all “don’t cares” as well as a “Broadcast” address of all XXXXXXXXb (all “don’t care” bits). This ensures that the serial port will reply to any address, and so that it is backwards compatible with the standard 80C51 microcontrollers that do not support automatic address recognition.

14. SERIAL PERIPHERAL INTERFACE (SPI)

The N76E003 provides a Serial Peripheral Interface (SPI) block to support high-speed serial communication. SPI is a full-duplex, high-speed, synchronous communication bus between microcontrollers or other peripheral devices such as serial EEPROM, LCD driver, or D/A converter. It provides either Master or Slave mode, high-speed rate up to $F_{SYS}/2$, transfer complete and write collision flag. For a multi-master system, SPI supports Master Mode Fault to protect a multi-master conflict.

14.1 Functional Description

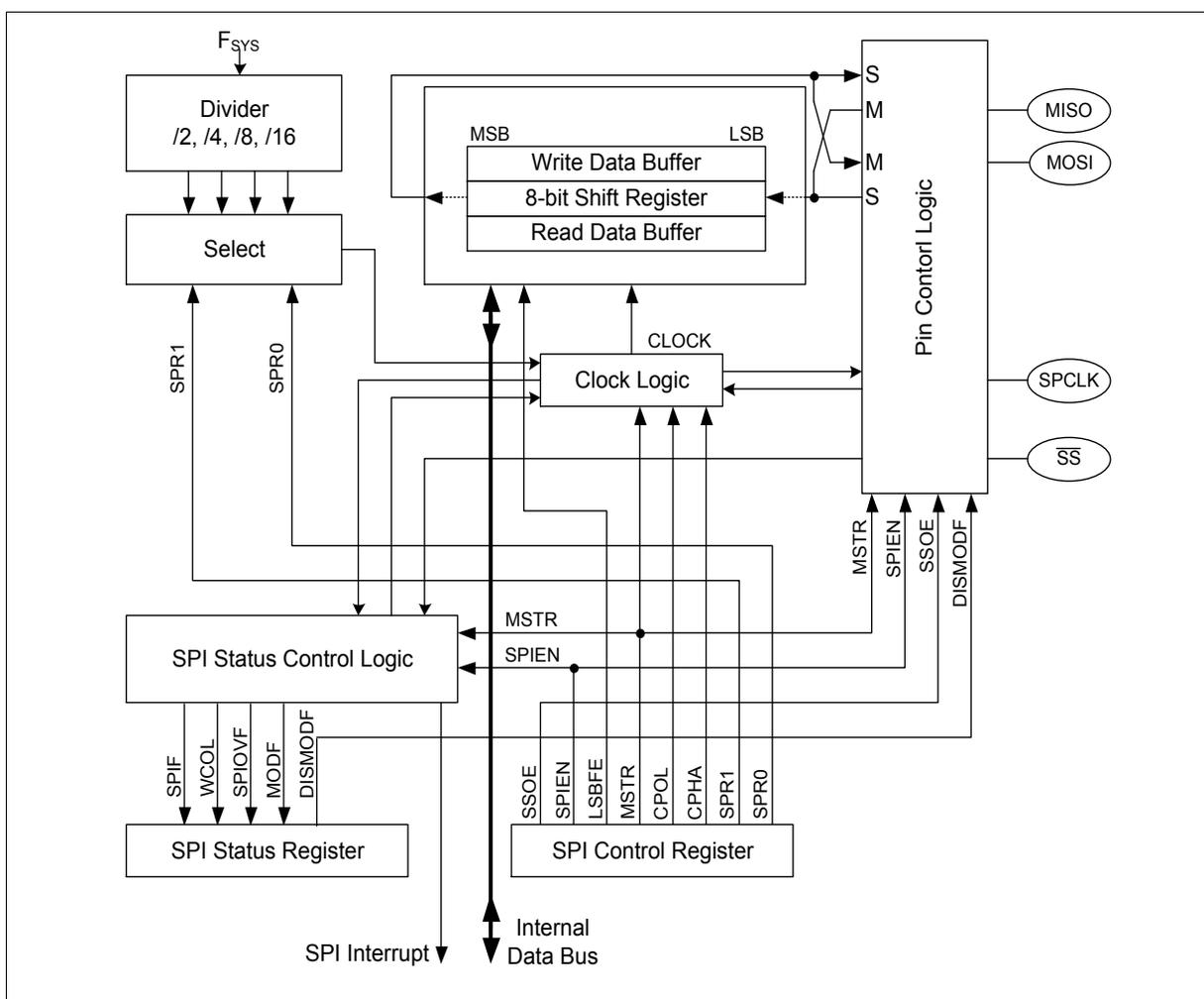


Figure 14-1. SPI Block Diagram

Figure 14-1. SPI Block Diagram shows SPI block diagram. It provides an overview of SPI architecture in this device. The main blocks of SPI are the SPI control register logic, SPI status logic, clock rate control logic, and pin control logic. For a serial data transfer or receiving, The SPI block exists a write

data buffer, a shift out register and a read data buffer. It is double buffered in the receiving and transmit directions. Transmit data can be written to the shifter until when the previous transfer is not complete. Receiving logic consists of parallel read data buffer so the shift register is free to accept a second data, as the first received data will be transferred to the read data buffer.

The four pins of SPI interface are Master-In/Slave-Out (MISO), Master-Out/Slave-In (MOSI), Shift Clock (SPCLK), and Slave Select (\overline{SS}). The MOSI pin is used to transfer a 8-bit data in series from the Master to the Slave. Therefore, MOSI is an output pin for Master device and an input for Slave. Respectively, the MISO is used to receive a serial data from the Slave to the Master.

The SPCLK pin is the clock output in Master mode, but is the clock input in Slave mode. The shift clock is used to synchronize the data movement both in and out of the devices through their MOSI and MISO pins. The shift clock is driven by the Master mode device for eight clock cycles. Eight clocks exchange one byte data on the serial lines. For the shift clock is always produced out of the Master device, the system should never exist more than one device in Master mode for avoiding device conflict.

Each Slave peripheral is selected by one Slave Select pin (\overline{SS}). The signal should stay low for any Slave access. When \overline{SS} is driven high, the Slave device will be inactivated. If the system is multi-slave, there should be only one Slave device selected at the same time. In the Master mode MCU, the \overline{SS} pin does not function and it can be configured as a general purpose I/O. However, \overline{SS} can be used as Master Mode Fault detection (see [Section 14.5 “Mode Fault Detection” on page 146](#)) via software setting if multi-master environment exists. The N76E003 also provides auto-activating function to toggle \overline{SS} between each byte-transfer.

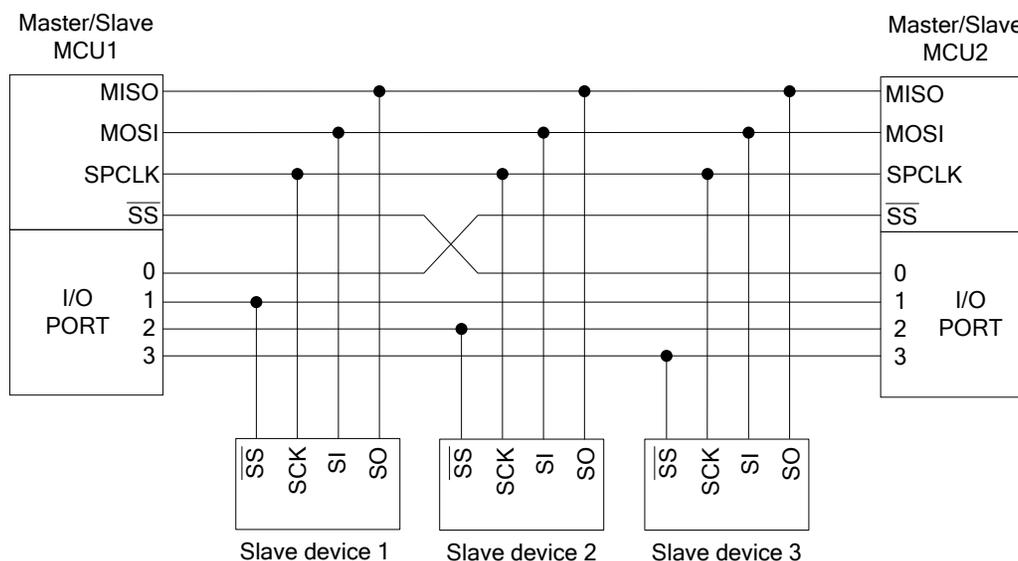


Figure 14-2. SPI Multi-Master, Multi-Slave Interconnection

Figure 14-2 shows a typical interconnection of SPI devices. The bus generally connects devices together through three signal wires, MOSI to MOSI, MISO to MISO, and SPCLK to SPCLK. The Master devices select the individual Slave devices by using four pins of a parallel port to control the four pins. MCU1 and MCU2 play either Master or Slave mode. The pins should be configured as Master Mode Fault detection to avoid multi-master conflict.

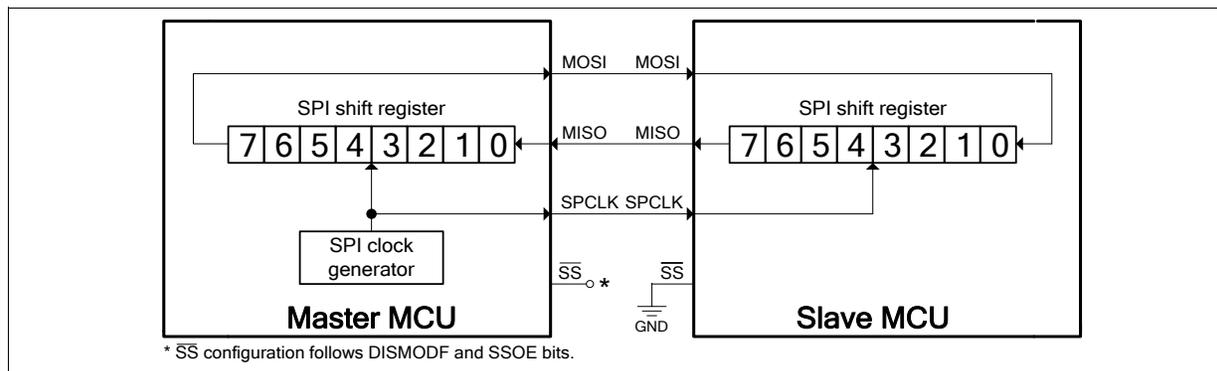


Figure 14-3. SPI Single-Master, Single-Slave Interconnection

Figure 14-3 shows the simplest SPI system interconnection, single-master and signal-slave. During a transfer, the Master shifts data out to the Slave via MOSI line. While simultaneously, the Master shifts data in from the Slave via MISO line. The two shift registers in the Master MCU and the Slave MCU can be considered as one 16-bit circular shift register. Therefore, while a transfer data pushed from Master into Slave, the data in Slave will also be pulled in Master device respectively. The transfer effectively exchanges the data, which was in the SPI shift registers of the two MCUs.

By default, SPI data is transferred MSB first. If the LSBFE (SPCR.5) is set, SPI data shifts LSB first. This bit does not affect the position of the MSB and LSB in the data register. Note that all the following description and figures are under the condition of LSBFE logic 0. MSB is transmitted and received first.

There are three SPI registers to support its operations, including SPI control register (SPCR), SPI status register (SPSR), and SPI data register (SPDR). These registers provide control, status, data storage functions, and clock rate selection. The following registers relate to SPI function.

SPCR – Serial Peripheral Control Register

7	6	5	4	3	2	1	0
SSOE	SPIEN	LSBFE	MSTR	CPOL	CPHA	SPR1	SPR0
R/W							

Address: F3H, page 0

Reset value: 0000 0000b

Bit	Name	Description
7	SSOE	<p>Slave select output enable</p> <p>This bit is used in combination with the DISMODF (SPSR.3) bit to determine the feature of \overline{SS} pin as shown in Table 14-1. Slave Select Pin Configurations. This bit takes effect only under MSTR = 1 and DISMODF = 1 condition.</p> <p>0 = \overline{SS} functions as a general purpose I/O pin.</p> <p>1 = \overline{SS} automatically goes low for each transmission when selecting external Slave device and goes high during each idle state to de-select the Slave device.</p>
6	SPIEN	<p>SPI enable</p> <p>0 = SPI function Disabled.</p> <p>1 = SPI function Enabled.</p>
5	LSBFE	<p>LSB first enable</p> <p>0 = The SPI data is transferred MSB first.</p> <p>1 = The SPI data is transferred LSB first.</p>
4	MSTR	<p>Master mode enable</p> <p>This bit switches the SPI operating between Master and Slave modes.</p> <p>0 = The SPI is configured as Slave mode.</p> <p>1 = The SPI is configured as Master mode.</p>
3	CPOL	<p>SPI clock polarity select</p> <p>CPOL bit determines the idle state level of the SPI clock. See Figure 14-4. SPI Clock Formats.</p> <p>0 = The SPI clock is low in idle state.</p> <p>1 = The SPI clock is high in idle state.</p>
2	CPHA	<p>SPI clock phase select</p> <p>CPHA bit determines the data sampling edge of the SPI clock. See Figure 14-4. SPI Clock Formats.</p> <p>0 = The data is sampled on the first edge of the SPI clock.</p> <p>1 = The data is sampled on the second edge of the SPI clock.</p>

Bit	Name	Description																				
1:0	SPR[1:0]	<p>SPI clock rate select These two bits select four grades of SPI clock divider. The clock rates below are illustrated under $F_{SYS} = 16 \text{ MHz}$ condition.</p> <table border="1"> <thead> <tr> <th>SPR1</th> <th>SPR0</th> <th>Divider</th> <th>SPI clock rate</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>2</td> <td>8M bit/s</td> </tr> <tr> <td>0</td> <td>1</td> <td>4</td> <td>4M bit/s</td> </tr> <tr> <td>1</td> <td>0</td> <td>8</td> <td>2M bit/s</td> </tr> <tr> <td>1</td> <td>1</td> <td>16</td> <td>1M bit/s</td> </tr> </tbody> </table> <p>SPR[1:0] are valid only under Master mode (MSTR = 1). If under Slave mode, the clock will automatically synchronize with the external clock on SPICLK pin from Master device up to $F_{SYS}/2$ communication speed.</p>	SPR1	SPR0	Divider	SPI clock rate	0	0	2	8M bit/s	0	1	4	4M bit/s	1	0	8	2M bit/s	1	1	16	1M bit/s
SPR1	SPR0	Divider	SPI clock rate																			
0	0	2	8M bit/s																			
0	1	4	4M bit/s																			
1	0	8	2M bit/s																			
1	1	16	1M bit/s																			

SPCR2 – Serial Peripheral Control Register 2

7	6	5	4	3	2	1	0
-	-	-	-	-	-	SPIS1	SPIS0
-	-	-	-	-	-	R/W	R/W

Address: F3H, page 1

Reset value: 0000 0000b

Bit	Name	Description																																				
7:2	-	Reserved																																				
1:0	SPIS[1:0]	<p>SPI Interval time selection between adjacent bytes SPIS[1:0] and CPHA select eight grades of SPI interval time selection between adjacent bytes. As below table:</p> <table border="1"> <thead> <tr> <th>CPHA</th> <th>SPIS1</th> <th>SPIS0</th> <th>SPI clock</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> <td>0.5</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> <td>1.0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> <td>1.5</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> <td>2.0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> <td>1.0</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>1.5</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> <td>2.0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> <td>2.5</td> </tr> </tbody> </table> <p>SPIS[1:0] are valid only under Master mode (MSTR = 1).</p>	CPHA	SPIS1	SPIS0	SPI clock	0	0	0	0.5	0	0	1	1.0	0	1	0	1.5	0	1	1	2.0	1	0	0	1.0	1	0	1	1.5	1	1	0	2.0	1	1	1	2.5
CPHA	SPIS1	SPIS0	SPI clock																																			
0	0	0	0.5																																			
0	0	1	1.0																																			
0	1	0	1.5																																			
0	1	1	2.0																																			
1	0	0	1.0																																			
1	0	1	1.5																																			
1	1	0	2.0																																			
1	1	1	2.5																																			

Table 14-1. Slave Select Pin Configurations

DISMODF	SSOE	Master Mode (MSTR = 1)	Slave Mode (MSTR = 0)
0	X	Input for Mode Fault	Input for Slave select
1	0	General purpose I/O	
1	1	Automatic output	

SPSR – Serial Peripheral Status Register

7	6	5	4	3	2	1	0
SPIF	WCOL	SPIOVF	MODF	DISMODF	-	-	-
R/W	R/W	R/W	R/W	R/W	-	-	-

Address: F4H

Reset value: 0000 0000b

Bit	Name	Description
7	SPIF	SPI complete flag This bit is set to logic 1 via hardware while an SPI data transfer is complete or an receiving data has been moved into the SPI read buffer. If ESPI (EIE .0) and EA are enabled, an SPI interrupt will be required. This bit should be cleared via software. Attempting to write to SPDR is inhibited if SPIF is set.
6	WCOL	Write collision error flag This bit indicates a write collision event. Once a write collision event occurs, this bit will be set. It should be cleared via software.
5	SPIOVF	SPI overrun error flag This bit indicates an overrun event. Once an overrun event occurs, this bit will be set. If ESPI and EA are enabled, an SPI interrupt will be required. This bit should be cleared via software.
4	MODF	Mode Fault error flag This bit indicates a Mode Fault error event. If pin is configured as Mode Fault input (MSTR = 1 and DISMODF = 0) and pin is pulled low by external devices, a Mode Fault error occurs. Instantly MODF will be set as logic 1. If ESPI and EA are enabled, an SPI interrupt will be required. This bit should be cleared via software.
3	DISMODF	Disable Mode Fault error detection This bit is used in combination with the SSOE (SPCR.7) bit to determine the feature of pin as shown in Table 14-1. Slave Select Pin Configurations . DISMODF is valid only in Master mode (MSTR = 1). 0 = Mode Fault detection Enabled. pin serves as input pin for Mode Fault detection disregard of SSOE. 1 = Mode Fault detection Disabled. The feature of pin follows SSOE bit.

SPDR – Serial Peripheral Data Register

7	6	5	4	3	2	1	0
SPDR[7:0]							
R/W							

Address: F5H

Reset value: 0000 0000b

Bit	Name	Description
7:0	SPDR[7:0]	<p>Serial peripheral data This byte is used for transmitting or receiving data on SPI bus. A write of this byte is a write to the shift register. A read of this byte is actually a read of the read data buffer. In Master mode, a write to this register initiates transmission and reception of a byte simultaneously.</p>

14.2 Operating Modes

14.2.1 Master Mode

The SPI can operate in Master mode while MSTR (SPCR.4) is set as 1. Only one Master SPI device can initiate transmissions. A transmission always begins by Master through writing to SPDR. The byte written to SPDR begins shifting out on MOSI pin under the control of SPCLK. Simultaneously, another byte shifts in from the Slave on the MISO pin. After 8-bit data transfer complete, SPIF (SPSR.7) will automatically set via hardware to indicate one byte data transfer complete. At the same time, the data received from the Slave is also transferred in SPDR. User can clear SPIF and read data out of SPDR.

14.2.2 Slave Mode

When MSTR is 0, the SPI operates in Slave mode. The SPCLK pin becomes input and it will be clocked by another Master SPI device. The \overline{CS} pin also becomes input. The Master device cannot exchange data with the Slave device until the \overline{CS} pin of the Slave device is externally pulled low. Before data transmissions occurs, the \overline{CS} of the Slave device should be pulled and remain low until the transmission is complete. If \overline{CS} goes high, the SPI is forced into idle state. If the \overline{CS} is forced to high at the middle of transmission, the transmission will be aborted and the rest bits of the receiving shifter buffer will be high and goes into idle state.

In Slave mode, data flows from the Master to the Slave on MOSI pin and flows from the Slave to the Master on MISO pin. The data enters the shift register under the control of the SPCLK from the Master device. After one byte is received in the shift register, it is immediately moved into the read data buffer and the SPIF bit is set. A read of the SPDR is actually a read of the read data buffer. To prevent an overrun and the loss of the byte that caused by the overrun, the Slave should read SPDR out and the first SPIF should be cleared before a second transfer of data from the Master device comes in the read data buffer.

14.3 Clock Formats and Data Transfer

To accommodate a wide variety of synchronous serial peripherals, the SPI has a clock polarity bit CPOL (SPCR.3) and a clock phase bit CPHA (SPCR.2). [Figure 14-4. SPI Clock Formats](#) shows that CPOL and CPHA compose four different clock formats. The CPOL bit denotes the SPCLK line level in its idle state. The CPHA bit defines the edge on which the MOSI and MISO lines are sampled. The CPOL and CPHA should be identical for the Master and Slave devices on the same system. To Communicate in different data formats with one another will result undetermined result.

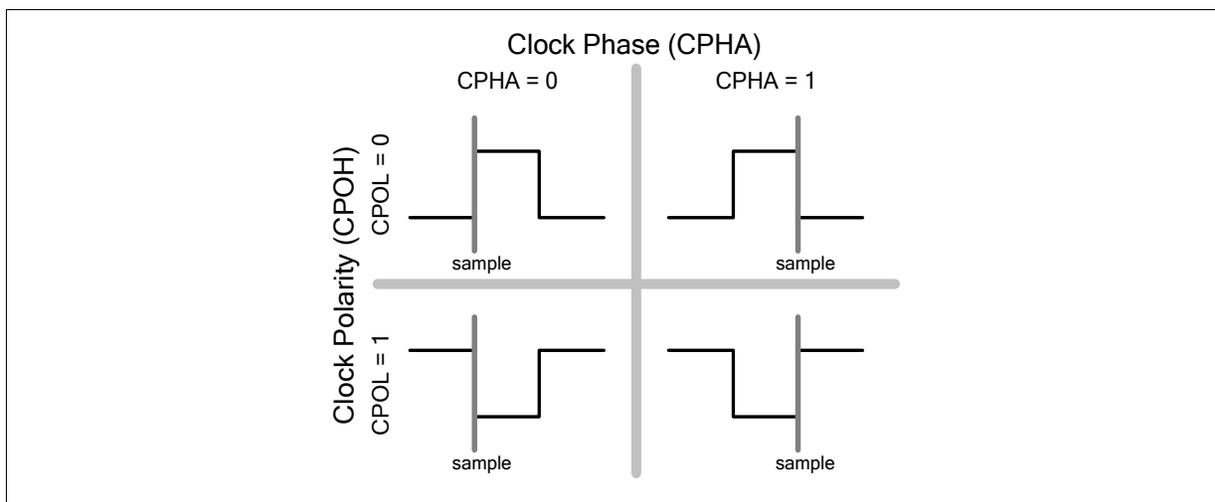


Figure 14-4. SPI Clock Formats

In SPI, a Master device always initiates the transfer. If SPI is selected as Master mode (MSTR = 1) and enabled (SPIEN = 1), writing to the SPI data register (SPDR) by the Master device starts the SPI clock and data transfer. After shifting one byte out and receiving one byte in, the SPI clock stops and SPIF (SPSR.7) is set in both Master and Slave. If SPI interrupt enable bit ESPI (EIE.0) is set 1 and global interrupt is enabled (EA = 1), the interrupt service routine (ISR) of SPI will be executed.

Concerning the Slave mode, the \overline{SPCLK} signal needs to be taken care. As shown in [Figure 14-4. SPI Clock Formats](#), when CPHA = 0, the first SPCLK edge is the sampling strobe of MSB (for an example of LSBFE = 0, MSB first). Therefore, the Slave should shift its MSB data before the first SPCLK edge. The falling edge of \overline{SPCLK} is used for preparing the MSB on MISO line. The \overline{SPCLK} pin therefore should toggle high and then low between each successive serial byte. Furthermore, if the slave writes data to the SPI data register (SPDR) while \overline{SPCLK} is low, a write collision error occurs.

When CPHA = 1, the sampling edge thus locates on the second edge of SPCLK clock. The Slave uses the first SPCLK clock to shift MSB out rather than the \overline{SPCLK} falling edge. Therefore, the \overline{SPCLK} line can remain low between successive transfers. This format may be preferred in systems having single fixed

Master and single fixed Slave. The \overline{SS} line of the unique Slave device can be tied to GND as long as only CPHA = 1 clock mode is used.

The SPI should be configured before it is enabled (SPIEN = 1), or a change of LSBFE, MSTR, CPOL, CPHA and SPR[1:0] will abort a transmission in progress and force the SPI system into idle state. Prior to any configuration bit changed, SPIEN must be disabled first.

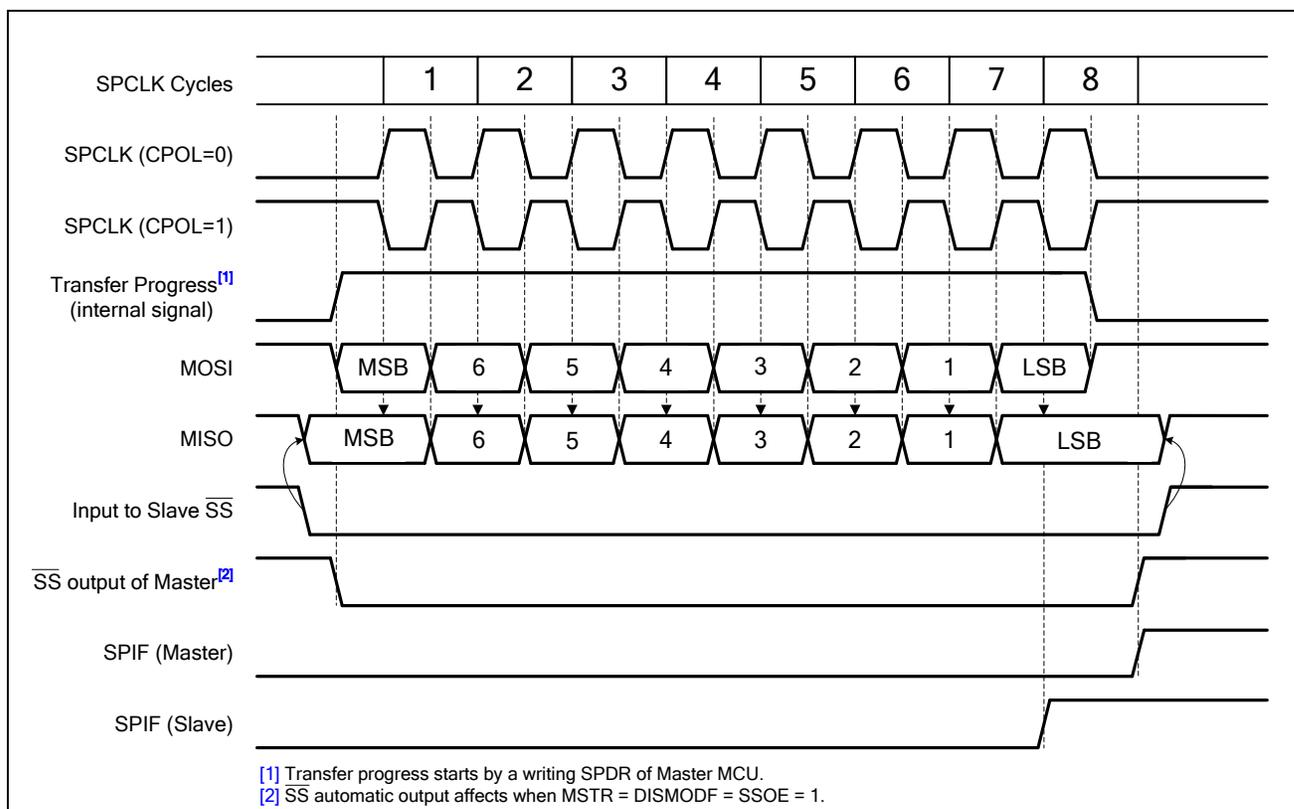


Figure 14-5. SPI Clock and Data Format with CPHA = 0

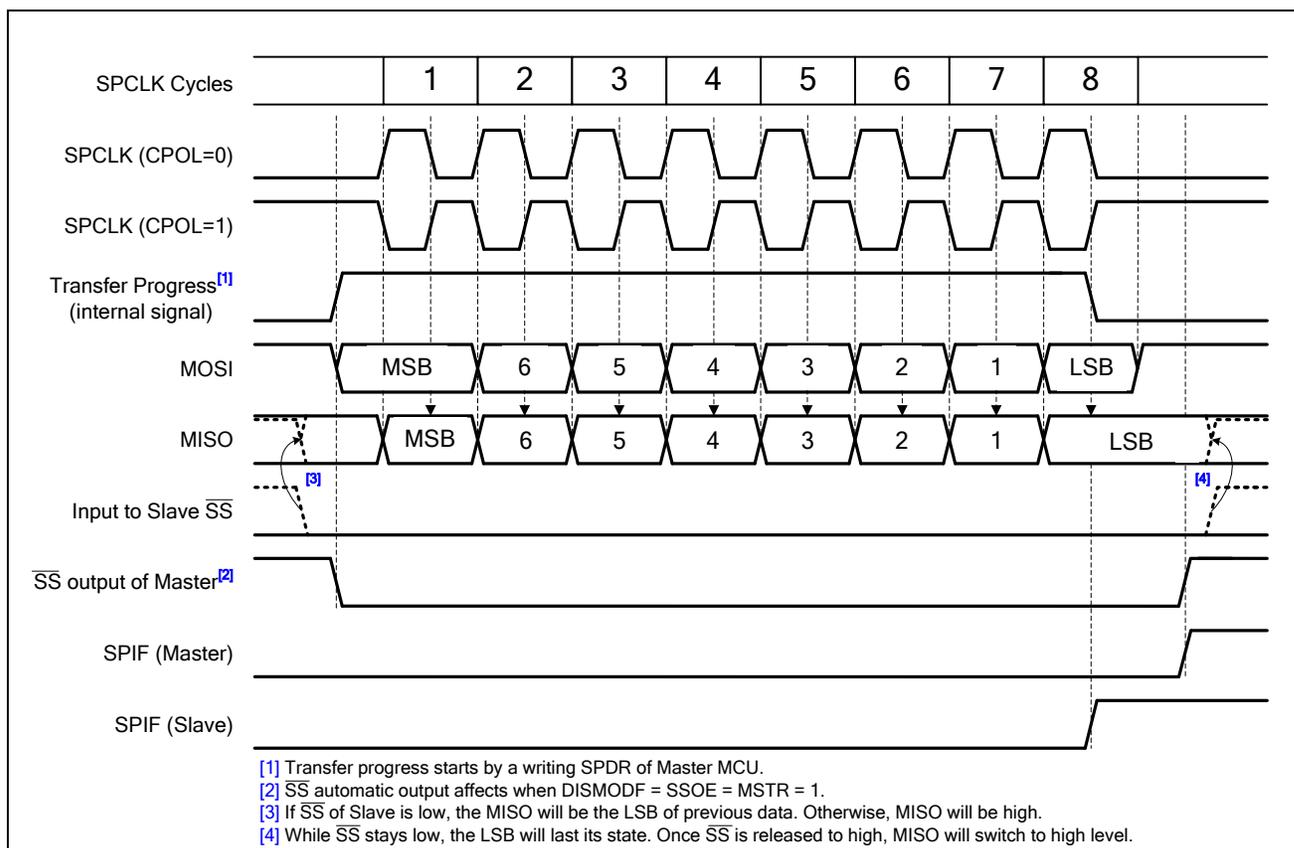


Figure 14-6. SPI Clock and Data Format with CPHA = 1

14.4 Slave Select Pin Configuration

The N76E003 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.

14.5 Mode Fault Detection

The Mode Fault detection is useful in a system where more than one SPI devices might become Masters at the same time. It may induce data contention. When the SPI device is configured as a Master and the \overline{SS} input line is configured for Mode Fault input depending on [Table 14-1. Slave Select Pin Configurations](#), a Mode Fault error occurs once the \overline{SS} is pulled low by others. It indicates that some other SPI device is trying to address this Master as if it is a Slave. Instantly the MSTR and SPIEN control bits in the SPCR are cleared via hardware to disable SPI, Mode Fault flag MODF (SPSR.4) is set and an interrupt is generated if ESPI (EIE .0) and EA are enabled.

14.6 Write Collision Error

The SPI is signal buffered in the transfer direction and double buffered in the receiving and transmit direction. New data for transmission cannot be written to the shift register until the previous transaction is complete. Write collision occurs while SPDR be written more than once while a transfer was in progress. SPDR is double buffered in the transmit direction. Any writing to SPDR cause data to be written directly into the SPI shift register. Once a write collision error is generated, WCOL (SPSR.6) will be set as 1 via hardware to indicate a write collision. In this case, the current transferring data continues its transmission. However the new data that caused the collision will be lost. Although the SPI logic can detect write collisions in both Master and Slave modes, a write collision is normally a Slave error because a Slave has no indicator when a Master initiates a transfer. During the receiving of Slave, a write to SPDR causes a write collision in Slave mode. WCOL flag needs to be cleared via software.

14.7 Overrun Error

For receiving data, the SPI is double buffered in the receiving direction. The received data is transferred into a parallel read data buffer so the shifter is free to accept a second serial byte. However, the received data should be read from SPDR before the next data has been completely shifted in. As long as the first byte is read out of the read data buffer and SPIF is cleared before the next byte is ready to be transferred, no overrun error condition occurs. Otherwise the overrun error occurs. In this condition, the second byte data will not be successfully received into the read data register and the previous data will remains. If overrun occur, SPIOVF (SPSR.5) will be set via hardware. An SPIOVF setting will also require an interrupt if enabled. [Figure 14-7. SPI Overrun Waveform](#) shows the relationship between the data receiving and the overrun error.

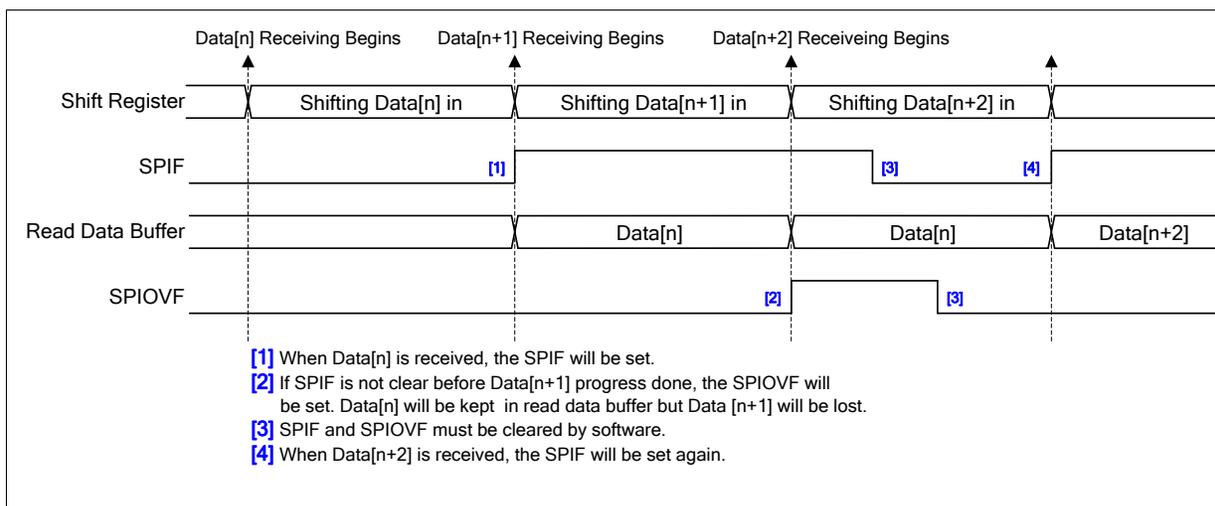


Figure 14-7. SPI Overrun Waveform

14.8 SPI Interrupt

Three SPI status flags, SPIF, MODF, and SPIOVF, can generate an SPI event interrupt requests. All of them locate in SPSR. SPIF will be set after completion of data transfer with external device or a new data have been received and copied to SPDR. MODF becomes set to indicate a low level on \overline{SS} causing the Mode Fault state. SPIOVF denotes a receiving overrun error. If SPI interrupt mask is enabled via setting ESPI (EIE.6) and EA is 1, CPU will executes the SPI interrupt service routine once any of these three flags is set. User needs to check flags to determine what event caused the interrupt. These three flags are software cleared.

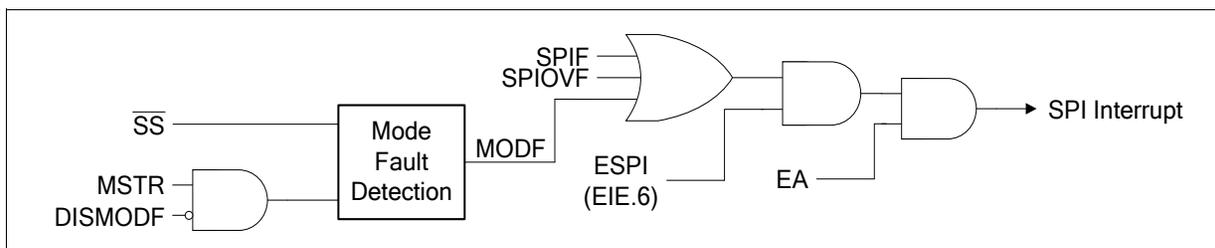


Figure 14-8. SPI Interrupt Request

15. INTER-INTEGRATED CIRCUIT (I²C)

The Inter-Integrated Circuit (I²C) bus serves as an serial interface between the microcontrollers and the I²C devices such as EEPROM, LCD module, temperature sensor, and so on. The I²C bus used two wires design (a serial data line SDA and a serial clock line SCL) to transfer information between devices.

The I²C bus uses bi-directional data transfer between masters and slaves. There is no central master and the multi-master system is allowed by arbitration between simultaneously transmitting masters. The serial clock synchronization allows devices with different bit rates to communicate via one serial bus. The I²C bus supports four transfer modes including master transmitter, master receiver, slave receiver, and slave transmitter. The I²C interface only supports 7-bit addressing mode. A special mode General Call is also available. The I²C can meet both standard (up to 100kbps) and fast (up to 400k bps) speeds.

15.1 Functional Description

For a bi-directional transfer operation, the SDA and SCL pins should be open-drain pads. This implements a wired-AND function, which is essential to the operation of the interface. A low level on a I²C bus line is generated when one or more I²C devices output a “ ”. A high level is generated when all I²C devices output “ ”, allowing the pull-up resistors to pull the line high. In N76E003, user should set output latches of SCL and SDA. As logic 1 before enabling the I²C function by setting I2CEN (I2CON.6).

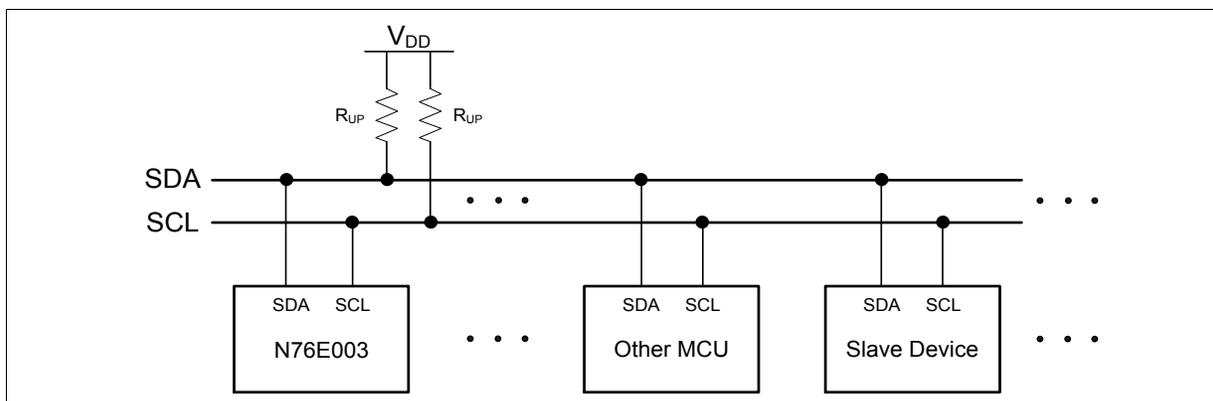


Figure 15-1. I²C Bus Interconnection

The I²C is considered free when both lines are high. Meanwhile, any device, which can operate as a master can occupy the bus and generate one transfer after generating a START condition. The bus now is considered busy before the transfer ends by sending a STOP condition. The master generates

all of the serial clock pulses and the START and STOP condition. However if there is no START condition on the bus, all devices serve as not addressed slave. The hardware looks for its own slave address or a General Call address. (The General Call address detection may be enabled or disabled by GC (I2ADDR.0).) If the matched address is received, an interrupt is requested.

Every transaction on the I²C bus is 9 bits long, consisting of 8 data bits (MSB first) and a single acknowledge bit. The number of bytes per transfer (defined as the time between a valid START and STOP condition) is unrestricted but each byte has to be followed by an acknowledge bit. The master device generates 8 clock pulse to send the 8-bit data. After the 8th falling edge of the SCL line, the device outputting data on the SDA changes that pin to an input and reads in an acknowledge value on the 9th clock pulse. After 9th clock pulse, the data receiving device can hold SCL line stretched low if next receiving is not prepared ready. It forces the next byte transaction suspended. The data transaction continues when the receiver releases the SCL line.

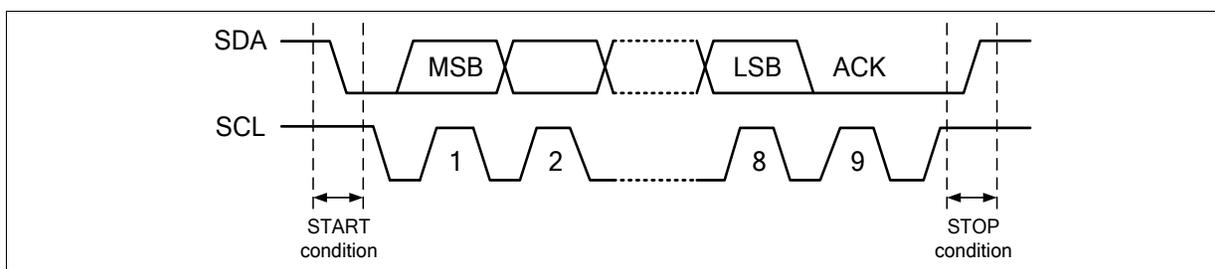


Figure 15-2. I²C Bus Protocol

15.1.1 START and STOP Condition

The protocol of the I²C bus defines two states to begin and end a transfer, START (S) and STOP (P) conditions. A START condition is defined as a high-to-low transition on the SDA line while SCL line is high. The STOP condition is defined as a low-to-high transition on the SDA line while SCL line is high. A START or a STOP condition is always generated by the master and I²C bus is considered busy after a START condition and free after a STOP condition. After issuing the STOP condition successful, the original master device will release the control authority and turn back as a not addressed slave. Consequently, the original addressed slave will become a not addressed slave. The I²C bus is free and listens to next START condition of next transfer.

A data transfer is always terminated by a STOP condition generated by the master. However, if a master still wishes to communicate on the bus, it can generate a repeated START (Sr) condition and address the pervious or another slave without first generating a STOP condition. Various combinations of read/write formats are then possible within such a transfer.

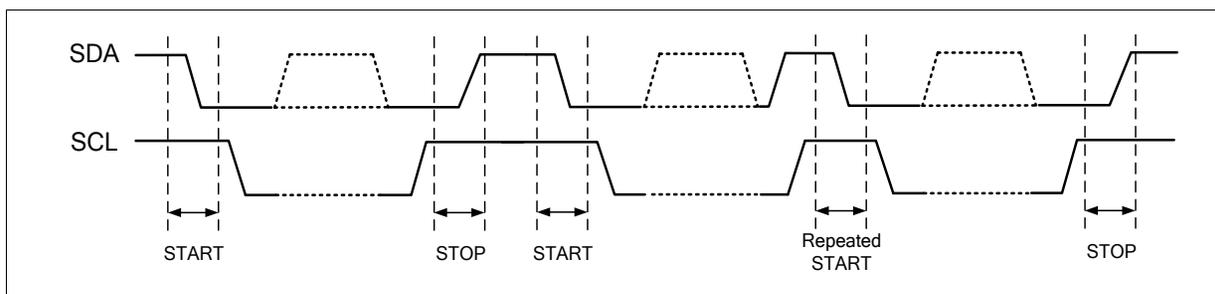


Figure 15-3. START, Repeated START, and STOP Conditions

15.1.2 7-Bit Address with Data Format

Following the START condition is generated, one byte of special data should be transmitted by the master. It includes a 7-bit long slave address (SLA) following by an 8th bit, which is a data direction bit (R/W), to address the target slave device and determine the direction of data flow. If R/W bit is 0, it indicates that the master will write information to a selected slave. Also, if R/W bit is 1, it indicates that the master will read information from the addressed slave. An address packet consisting of a slave address and a read R or a write (W) bit is called SLA+R or SLA+W, respectively. A transmission basically consists of a START condition, a SLA+W/R, one or more data packets and a STOP condition. After the specified slave is addressed by SLA+W/R, the second and following 8-bit data bytes issue by the master or the slave devices according to the R/W bit configuration.

here is an exception called “General all” address, which can address all devices by giving the first byte of data all 0. A General Call is used when a master wishes to transmit the same message to several slaves in the system. When this address is used, other devices may respond with an acknowledge or ignore it according to individual software configuration. If a device response the General Call, it operates as like in the slave-receiver mode. Note that the address 0x00 is reserved for General Call and cannot be used as a slave address, therefore, in theory, a 7-bit addressing I²C bus accepts 127 devices with their slave addresses 1 to 127.

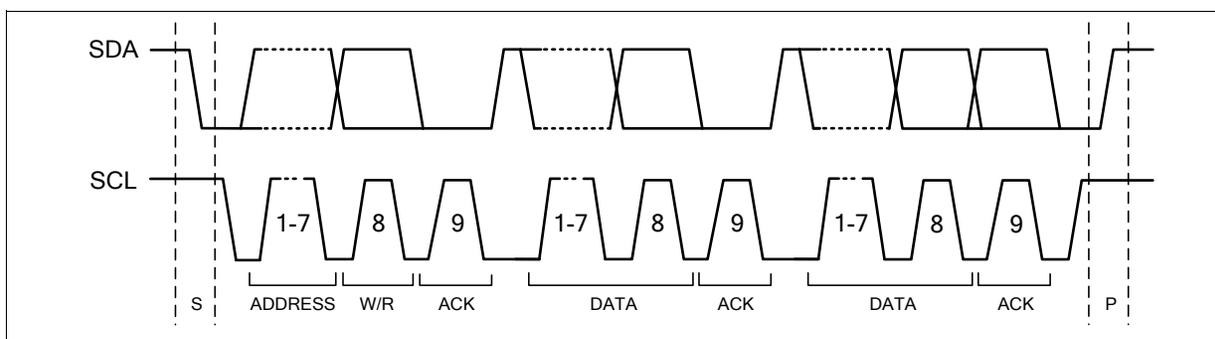


Figure 15-4. Data Format of One I²C Transfer

During the data transaction period, the data on the SDA line should be stable during the high period of the clock, and the data line can only change when SCL is low.

15.1.3 Acknowledge

The 9th SCL pulse for any transferred byte is dedicated as an Acknowledge (ACK). It allows receiving devices (which can be the master or slave) to respond back to the transmitter (which also can be the master or slave) by pulling the SDA line low. The acknowledge-related clock pulse is generated by the master. The transmitter should release control of SDA line during the acknowledge clock pulse. The ACK is an active-low signal, pulling the SDA line low during the clock pulse high duty, indicates to the transmitter that the device has received the transmitted data. Commonly, a receiver, which has been addressed is requested to generate an ACK after each byte has been received. When a slave receiver does not acknowledge (NACK) the slave address, the SDA line should be left high by the slave so that the mater can generate a STOP or a repeated START condition.

If a slave-receiver does acknowledge the slave address, it switches itself to not addressed slave mode and cannot receive any more data bytes. This slave leaves the SDA line high. The master should generate a STOP or a repeated START condition.

If a master-receiver is involved in a transfer, because the master controls the number of bytes in the transfer, it should signal the end of data to the slave-transmitter by not generating an acknowledge on the last byte. The slave-transmitter then switches to not addressed mode and releases the SDA line to allow the master to generate a STOP or a repeated START condition.

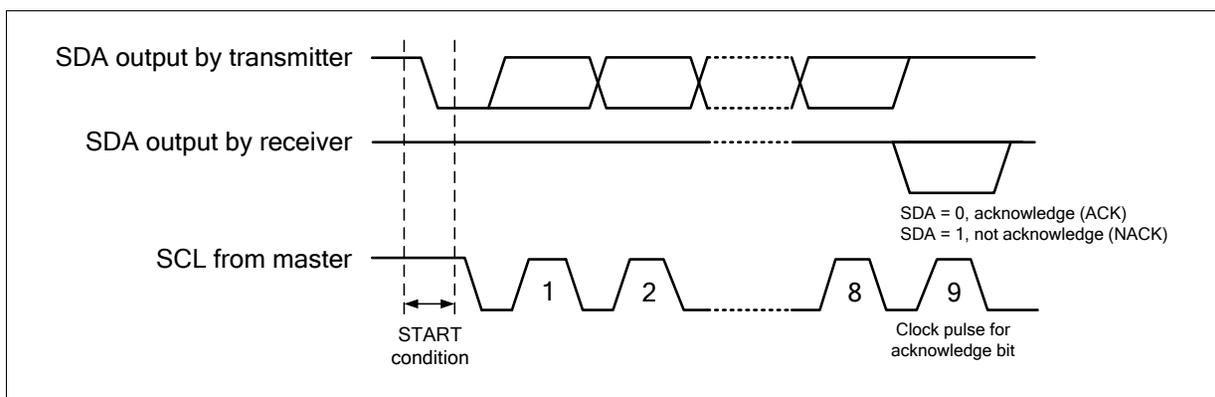


Figure 15-5. Acknowledge Bit

15.1.4 Arbitration

A master may start a transfer only if the bus is free. It is possible for two or more masters to generate a START condition. In these situations, an arbitration scheme takes place on the SDA line, while SCL is high. During arbitration, the first of the competing master devices to place 'a' '1' (high) on SDA while

another master transmits 'a'0' (low) switches off its data output stage because the level on the bus does not match its own level. The arbitration lost master switches to the not addressed slave immediately to detect its own slave address in the same serial transfer whether it is being addressed by the winning master. It also releases SDA line to high level for not affecting the data transfer continued by the winning master. However, the arbitration lost master continues generating clock pulses on SCL line until the end of the byte in which it loses the arbitration.

Arbitration is carried out by all masters continuously monitoring the SDA line after outputting data. If the value read from the SDA line does not match the value that the master has to output, it has lost the arbitration. Note that a master can only lose arbitration when it outputs a high SDA value while another master outputs a low value. Arbitration will continue until only one master remains, and this may take many bits. Its first stage is a comparison of address bits, and if both masters are trying to address the same device, arbitration continues on to the comparison of data bits or acknowledge bit.

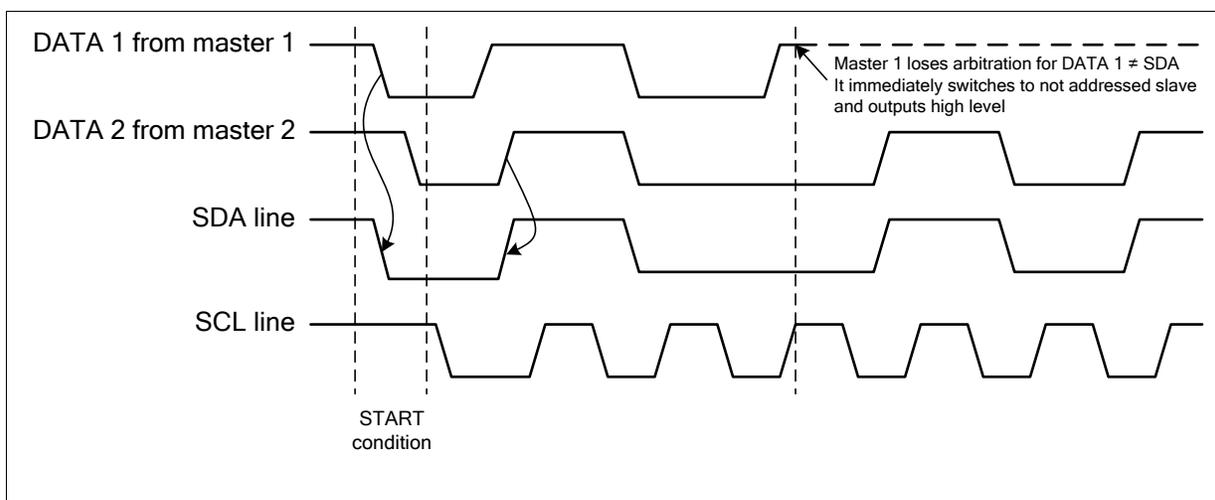


Figure 15-6. Arbitration Procedure of Two Masters

Since control of the I²C bus is decided solely on the address or master code and data sent by competing masters, there is no central master, nor any order of priority on the bus. Slaves are not involved in the arbitration procedure.

15.2 Control Registers of I²C

There are five control registers to interface the I²C bus including I2CON, I2STAT, I2DAT, I2ADDR, and I2CLK. These registers provide protocol control, status, data transmitting and receiving functions, and clock rate configuration. For application flexibility, SDA and SCL pins can be exchanged by I2CPX (I2CON.0). The following registers relate to I²C function.

I2CON – I²C Control (Bit-addressable)

7	6	5	4	3	2	1	0
-	I2CEN	STA	STO	SI	AA	-	I2CPX
-	R/W	R/W	R/W	R/W	R/W	-	R/W

Address: C0H

Reset value: 0000 0000b

Bit	Name	Description
7	-	Reserved
6	I2CEN	I²C bus enable 0 = I ² C bus Disabled. 1 = I ² C bus Enabled. Before enabling the I ² C, SCL and SDA port latches should be set to logic 1.
5	STA	START flag When STA is set, the I ² C generates a START condition if the bus is free. If the bus is busy, the I ² C waits for a STOP condition and generates a START condition following. If STA is set while the I ² C is already in the master mode and one or more bytes have been transmitted or received, the I ² C generates a repeated START condition. Note that STA can be set anytime even in a slave mode, but STA is not hardware automatically cleared after START or repeated START condition has been detected. User should take care of it by clearing STA manually.
4	STO	STOP flag When STO is set if the I ² C is in the master mode, a STOP condition is transmitted to the bus. STO is automatically cleared by hardware once the STOP condition has been detected on the bus. The STO flag setting is also used to recover the I ² C device from the bus error state (I2STAT as 00H). In this case, no STOP condition is transmitted to the I ² C bus. If the STA and STO bits are both set and the device is original in the master mode, the I ² C bus will generate a STOP condition and immediately follow a START condition. If the device is in slave mode, STA and STO simultaneous setting should be avoid from issuing illegal I ² C frames.
3	SI	I²C interrupt flag SI flag is set by hardware when one of 26 possible I ² C status (besides F8H status) is entered. After SI is set, the software should read I2STAT register to determine which step has been passed and take actions for next step. SI is cleared by software. Before the SI is cleared, the low period of SCL line is stretched. The transaction is suspended. It is useful for the slave device to deal with previous data bytes until ready for receiving the next byte. The serial transaction is suspended until SI is cleared by software. After SI is cleared, I ² C bus will continue to generate START or repeated START condition, STOP condition, 8-bit data, or so on depending on the software configuration of controlling byte or bits. Therefore, user should take care of it by preparing suitable setting of registers before SI is software cleared.

Bit	Name	Description
2	AA	<p>Acknowledge assert flag</p> <p>If the AA flag is set, an ACK (low level on SDA) will be returned during the acknowledge clock pulse of the SCL line while the I²C device is a receiver or an own-address-matching slave.</p> <p>If the AA flag is cleared, a NACK (high level on SDA) will be returned during the acknowledge clock pulse of the SCL line while the I²C device is a receiver or an own-address-matching slave. A device with its own AA flag cleared will ignore its own salve address and the General Call. Consequently, SI will note be asserted and no interrupt is requested.</p> <p>Note that if an addressed slave does not return an ACK under slave receiver mode or not receive an ACK under slave transmitter mode, the slave device will become a not addressed slave. It cannot receive any data until its AA flag is set and a master addresses it again.</p> <p>There is a special case of I2STAT value C8H occurs under slave transmitter mode. Before the slave device transmit the last data byte to the master, AA flag can be cleared as 0. Then after the last data byte transmitted, the slave device will actively switch to not addressed slave mode of disconnecting with the master. The further reading by the master will be all FFH.</p>
1	-	Reserved
0	I2CPX	<p>I2C pins select</p> <p>0 = Assign SCL to P1.3 and SDA to P1.4. 1 = Assign SCL to P0.2 and SDA to P1.6.</p> <p>Note that I2C pins will exchange immediately once setting or clearing this bit.</p>

I2STAT – I²C Status

7	6	5	4	3	2	1	0
I2STAT[7:3]					0	0	0
R					R	R	R

Address: BDH

Reset value: 1111 1000b

Bit	Name	Description
7:3	I2STAT[7:3]	<p>I²C status code</p> <p>The MSB five bits of I2STAT contains the status code. There are 27 possible status codes. When I2STAT is F8H, no relevant state information is available and SI flag keeps 0. All other 26 status codes correspond to the I²C states. When each of these status is entered, SI will be set as logic 1 and a interrupt is requested.</p>
2:0	0	<p>Reserved</p> <p>The least significant three bits of I2STAT are always read as 0.</p>

I2DAT – I²C Data

7	6	5	4	3	2	1	0
I2DAT[7:0]							
R/W							

Address: BCH

Reset value: 0000 0000b

Bit	Name	Description
7:0	I2DAT[7:0]	<p>I²C data</p> <p>I2DAT contains a byte of the I²C data to be transmitted or a byte, which has just received. Data in I2DAT remains as long as SI is logic 1. The result of reading or writing I2DAT during I²C transceiving progress is unpredicted. While data in I2DAT is shifted out, data on the bus is simultaneously being shifted in to update I2DAT. I2DAT always shows the last byte that presented on the I²C bus. Thus the event of lost arbitration, the original value of I2DAT changes after the transaction.</p>

I2ADDR – I²C Own Slave Address

7	6	5	4	3	2	1	0
I2ADDR[7:1]							GC
R/W							R/W

Address: C1H

Reset value: 0000 0000b

Bit	Name	Description
7:1	I2ADDR[7:1]	<p>I²C device's own slave address</p> <p><u>In master mode:</u> These bits have no effect.</p> <p><u>In slave mode:</u> These 7 bits define the slave address of this I²C device by user. The master should address I²C device by sending the same address in the first byte data after a START or a repeated START condition. If the AA flag is set, this I²C device will acknowledge the master after receiving its own address and become an addressed slave. Otherwise, the addressing from the master will be ignored. Note that I2ADDR[7:1] should not remain its default value of all 0, because address 0x00 is reserved for General Call.</p>
6	GC	<p>General Call bit</p> <p><u>In master mode:</u> This bit has no effect.</p> <p><u>In slave mode:</u> 0 = The General Call is always ignored. 1 = The General Call is recognized if AA flag is 1; otherwise, it is ignored if AA is 0.</p>

I2CLK – I²C Clock

7	6	5	4	3	2	1	0
I2CLK[7:0]							
R/W							

Address: BEH

Reset value: 0000 1001b

Bit	Name	Description
7:0	I2CLK[7:0]	<p>I²C clock setting</p> <p><u>In master mode:</u> This register determines the clock rate of I²C bus when the device is in a master mode. The clock rate follows the equation, $\frac{F_{sys}}{4 \times (I2CLK + 1)}$ The default value will make the clock rate of I²C bus 400k bps if the peripheral clock is 16 MHz. Note that the I2CLK value of 00H and 01H are not valid. This is an implement limitation.</p> <p><u>In slave mode:</u> This byte has no effect. In slave mode, the I²C device will automatically synchronize with any given clock rate up to 400k bps.</p>

15.3 Operating Modes

In I²C protocol definition, there are four operating modes including master transmitter, master receiver, slave receiver, and slave transmitter. There is also a special mode called General Call. Its operating is similar to master transmitter mode.

15.3.1 Master Transmitter Mode

In the master transmitter mode, several bytes of data are transmitted to a slave receiver. The master should prepare by setting desired clock rate in I2CLK. The master transmitter mode may now be entered by setting STA (I2CON.5) bit as 1. The hardware will test the bus and generate a START condition as soon as the bus becomes free. After a START condition is successfully produced, the SI flag (I2CON.3) will be set and the status code in I2STAT show 08H. The progress is continued by loading DA with the target slave address and the data direction bit “write” (A+W). he bit should then be cleared to commence SLA+W transaction.

After the SLA+W byte has been transmitted and an acknowledge (ACK) has been returned by the addressed slave device, the SI flag is set again and I2STAT is read as 18H. The appropriate action to be taken follows user defined communication protocol by sending data continuously. After all data is transmitted, the master can send a STOP condition by setting STO (I2CON.4) and then clearing SI to terminate the transmission. A repeated START condition can also be generated without sending STOP condition to immediately initial another transmission.

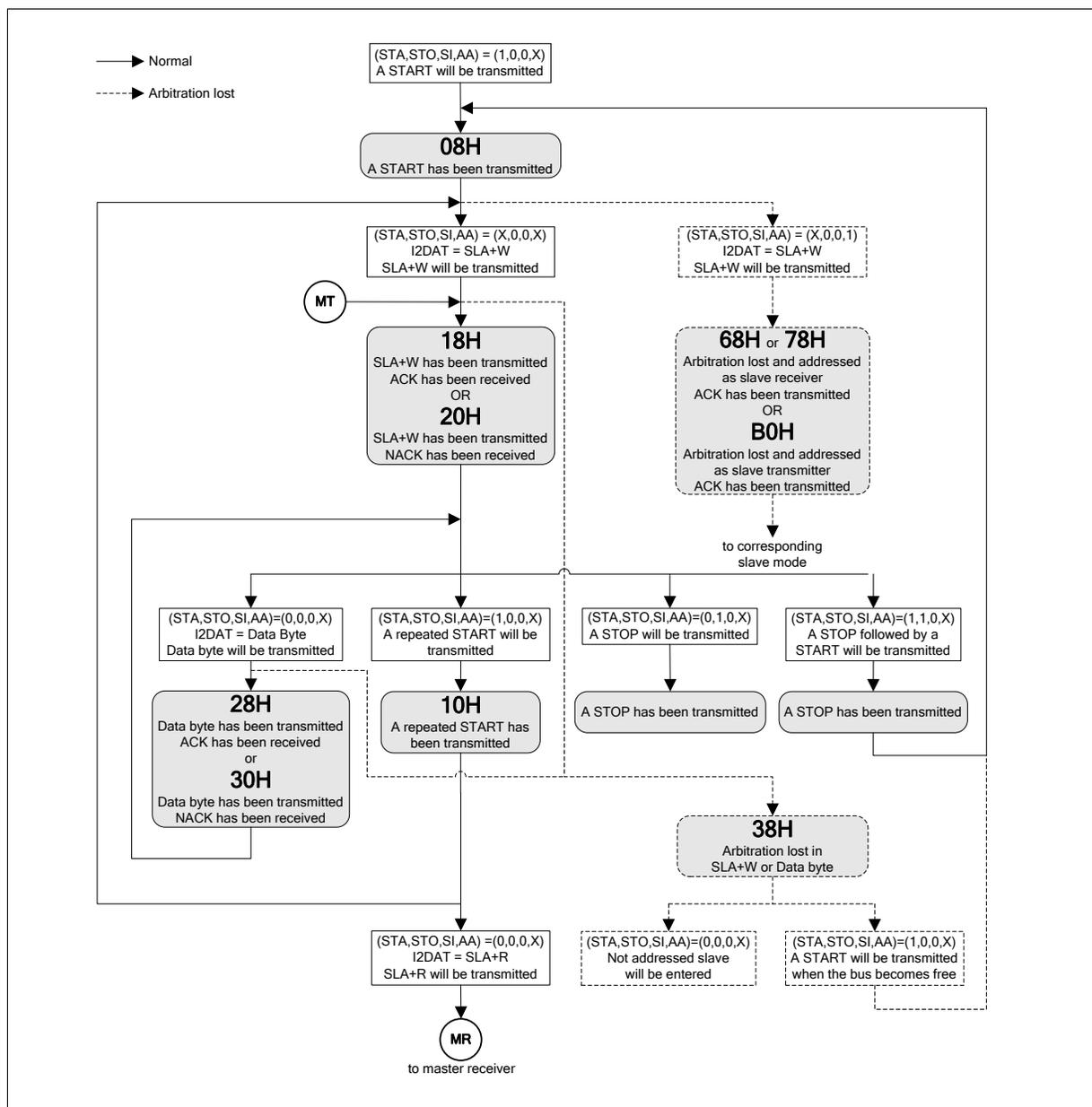


Figure 15-7. Flow and Status of Master Transmitter Mode

15.3.2 Master Receiver Mode

In the master receiver mode, several bytes of data are received from a slave transmitter. The transaction is initialized just as the master transmitter mode. Following the START condition, I2DAT should be loaded with the target slave address and the data direction bit “read” (A +). After the SLA+R byte is transmitted and an acknowledge bit has been returned, the SI flag is set again and I2STAT is read as 40H. SI flag then should be cleared to receive data from the slave transmitter. If AA flag (I2CON.2) is set, the master receiver will acknowledge the slave transmitter. If AA is cleared, the

to enable acknowledging its own slave address. After the initialization above, the I²C idles until it is addressed by its own address with the data direction bit “write” (A+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.

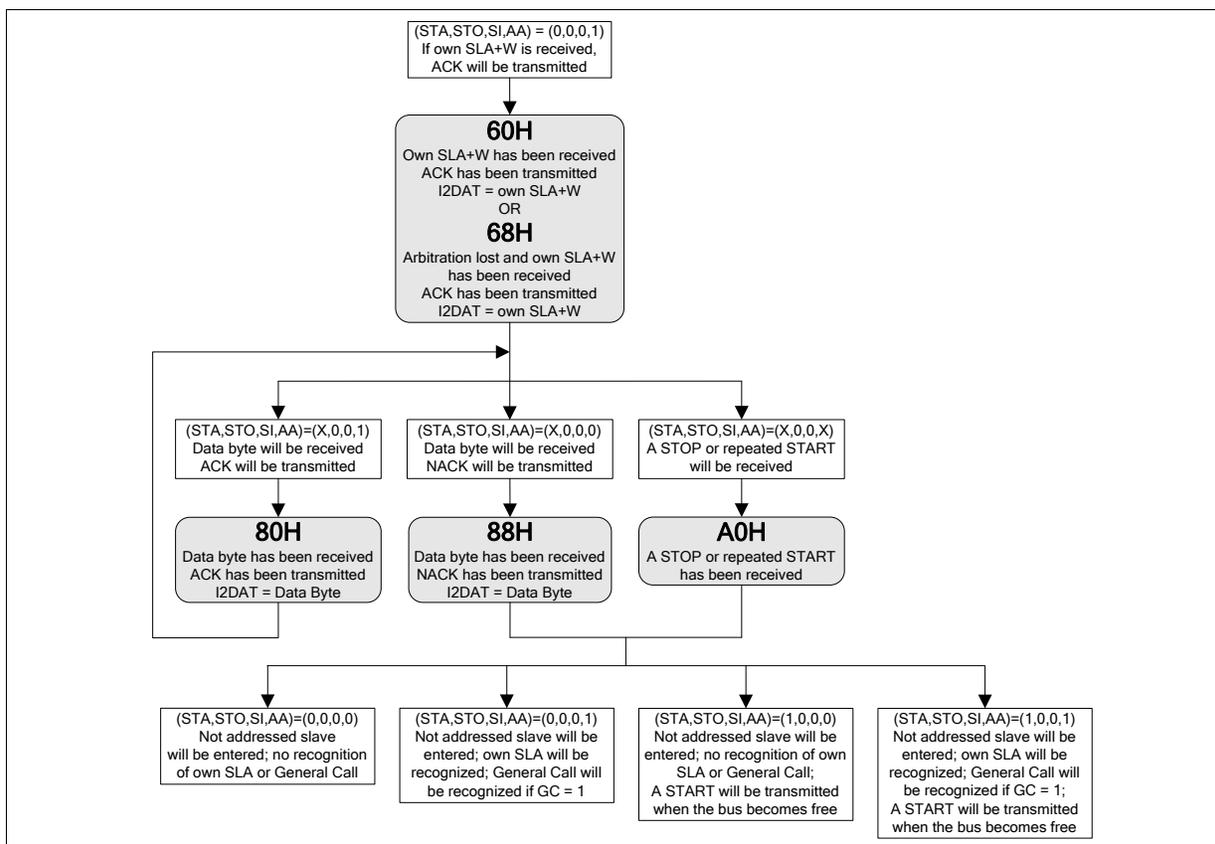


Figure 15-9. Flow and Status of Slave Receiver Mode

15.3.4 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 I²C wait until it is addressed by its own address with the data direction bit “read” (A+). 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 “ ” data if it continues

the transaction. It becomes a not addressed slave. If the AA flag is cleared during a transaction, the slave transmits the last byte of data. The next transmitting data will be all “ ” and the slave becomes not addressed.

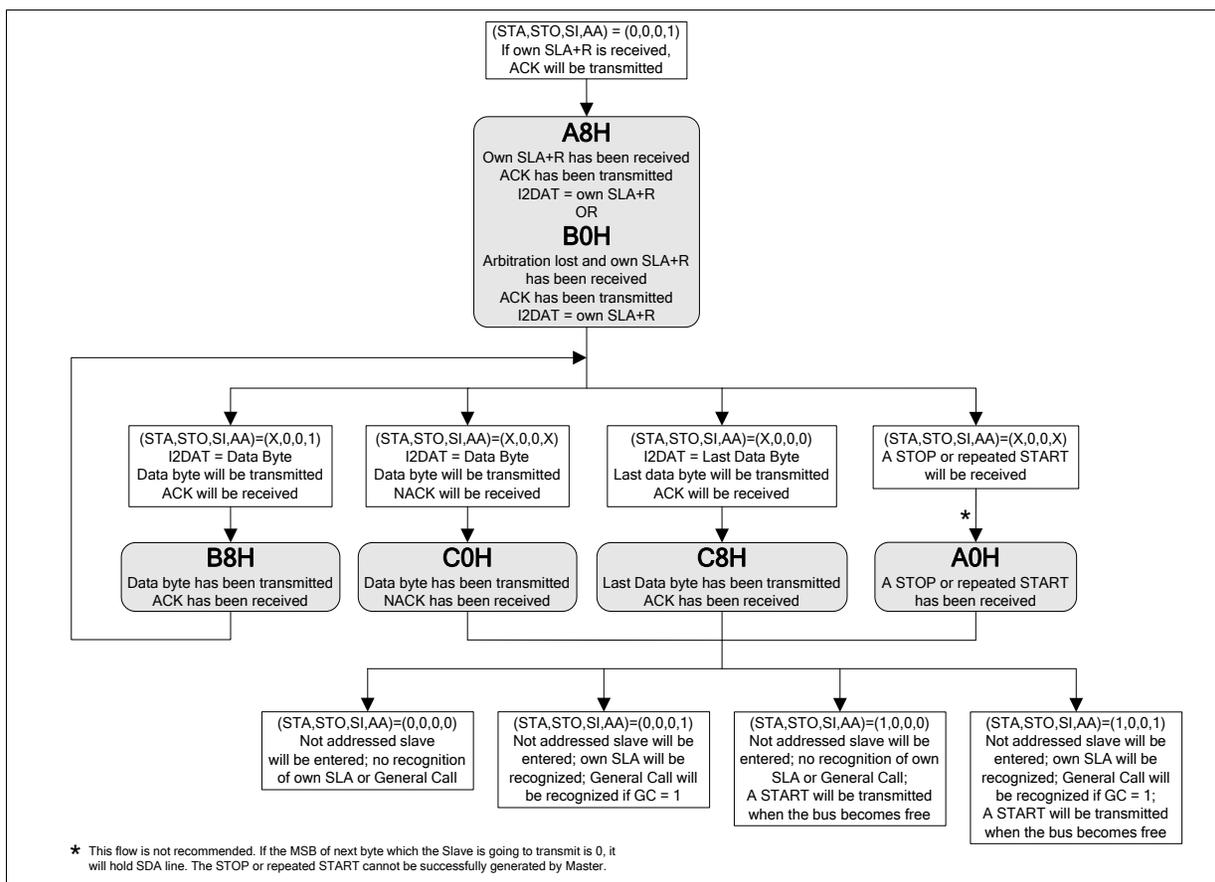


Figure 15-10. Flow and Status of Slave Transmitter Mode

15.3.5 General Call

The General Call is a special condition of slave receiver mode by being addressed with all “ ” data in slave address with data direction bit. Both GC (I2ADDR.0) bit and AA bit should be set as 1 to enable acknowledging General Calls. The slave addressed by a General Call has different status code in I2STAT with normal slave receiver mode. The General Call may also be produced if arbitration is lost.

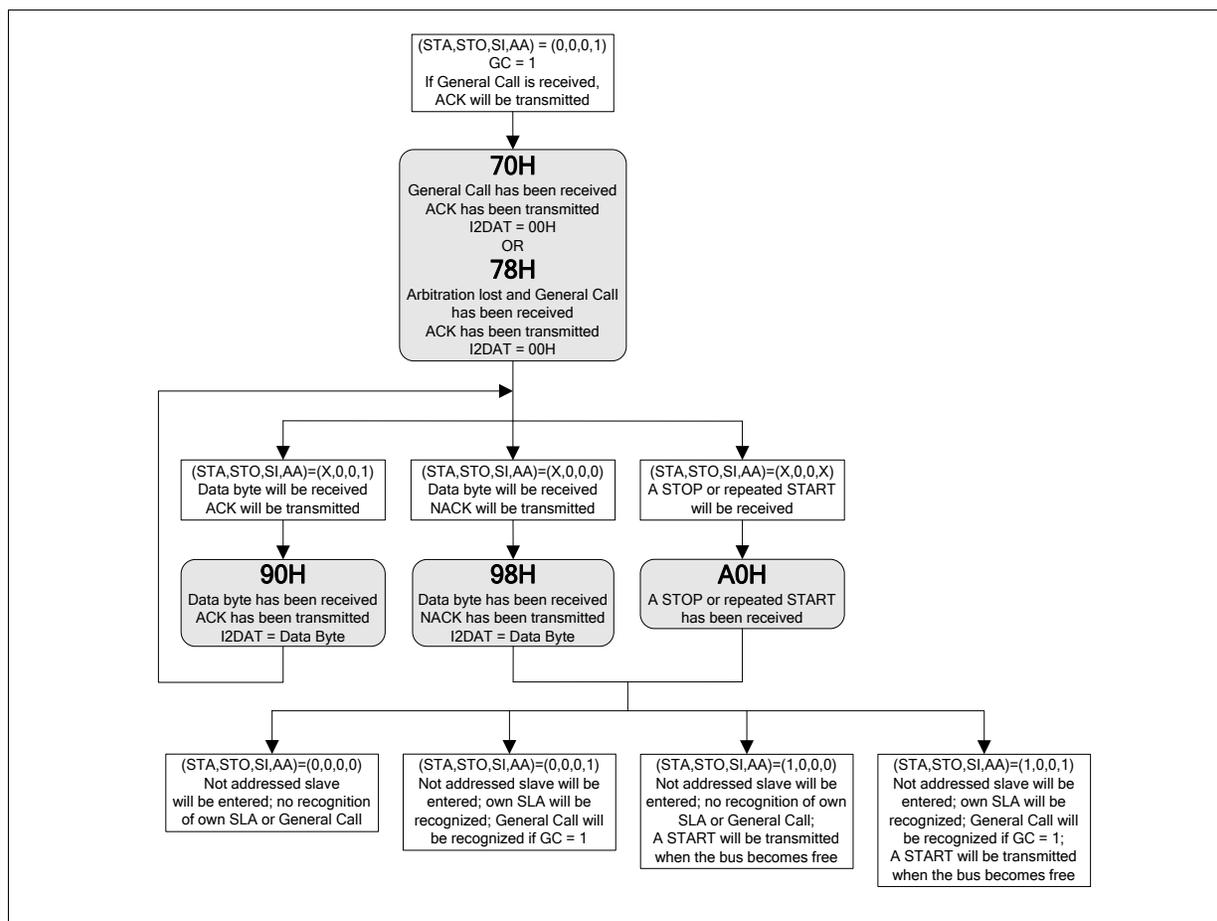


Figure 15-11. Flow and Status of General Call Mode

15.3.6 Miscellaneous States

There are two I2STAT status codes that do not correspond to the 25 defined states, which are mentioned in previous sections. These are F8H and 00H states.

The first status code F8H indicates that no relevant information is available during each transaction. Meanwhile, the SI flag is 0 and no I²C interrupt is required.

The other status code 00H means a bus error has occurred during a transaction. A bus error is caused by a START or STOP condition appearing temporarily at an illegal position such as the second through eighth bits of an address or a data byte, and the acknowledge bit. When a bus error occurs, the SI flag is set immediately. When a bus error is detected on the I²C bus, the operating device immediately switches to the not addressed slave mode, releases SDA and SCL lines, sets the SI flag, and loads I2STAT as 00H. To recover from a bus error, the STO bit should be set and then SI should be cleared. After that, STO is cleared by hardware and release the I²C bus without issuing a real STOP condition waveform on I²C bus.

There is a special case if a START or a repeated START condition is not successfully generated for I²C 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 I²C 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 I²C 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.

The following table is show the status display in I2STAT register of I²C number and description:

Master Mode		Slave Mode	
STATUS	Description	STATUS	Description
0x08	Start	0xA0	Slave Transmit Repeat Start or Stop
0x10	Master Repeat Start	0xA8	Slave Transmit Address ACK
0x18	Master Transmit Address ACK	0xB0	Slave Transmit Arbitration Lost
0x20	Master Transmit Address NACK	0xB8	Slave Transmit Data ACK
0x28	Master Transmit Data ACK	0xC0	Slave Transmit Data NACK
0x30	Master Transmit Data NACK	0xC8	Slave Transmit Last Data ACK
0x38	Master Arbitration Lost	0x60	Slave Receive Address ACK
0x40	Master Receive Address ACK	0x68	Slave Receive Arbitration Lost
0x48	Master Receive Address NACK	0x80	Slave Receive Data ACK
0x50	Master Receive Data ACK	0x88	Slave Receive Data NACK
0x58	Master Receive Data NACK	0x70	GC mode Address ACK
0x00	Bus error	0x78	GC mode Arbitration Lost
		0x90	GC mode Data ACK
		0x98	GC mode Data NACK
0xF8	Bus Released Note: tatus " xF8" exists in both master slave modes, and it won't raise interrupt.		

Note:

When I2C is enabled and I2C status is entered bus error state, SI flag is set by hardware. Until the I2C bus error is handled, SI flag will maintain its value at 1 and cannot be cleared by software. That is to

clear SI flag does not clear I2C bus error as well. When using SI flag to determine I2C status and flow, use following steps to enhance the reliability of the system.

Solution:

- Send a STOP condition to I2C bus
- If the STOP condition is invalid, disable the I2C bus and then restart the communication.

For example:

```
while(SI != 0)
{
    if (I2STAT == 0x00)
    {
        STO = 1;           // Check bus status if bus error, first send stop
    }
    SI = 0;
    if(SI!=0)             // If SI still keep 1
    {
        I2CEN = 0;       // please first disable I2C.
        I2CEN = 1 ;     // Then enable I2C for clear SI.
        SI = 0;
        I2CEN = 0;      // At last disable I2C for next a new transfer
    }
}
```

15.4 Typical Structure of I²C Interrupt Service Routine

The following software example in C language for KEIL™ C51 compiler shows the typical structure of the I²C 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
        //=====
    }
```

```

software
    case 0x08: /*08H, a START transmitted*/
        STA = 0; //STA bit should be cleared by

        I2DAT = SLA_ADDR1; //load SLA+W/R
        break;
    case 0x10: /*10H, a repeated START transmitted*/
        STA = 0;
        I2DAT = SLA_ADDR2;
        break;

    //=====
    //Master Transmitter Mode
    //=====
received*/
    case 0x18: /*18H, SLA+W transmitted, ACK

        I2DAT = NEXT_SEND_DATA1; //load DATA
        break;
    case 0x20: /*20H, SLA+W transmitted, NACK

        STO = 1; //transmit STOP
        AA = 1; //ready for ACK own SLA+W/R or

General Call
        break;
    case 0x28: /*28H, DATA transmitted, ACK

        if (Conti_TX_Data) //if continuing to send DATA
            I2DAT = NEXT_SEND_DATA2;
        else //if no DATA to be sent
        {
            STO = 1;
            AA = 1;
        }
        break;
received*/
    case 0x30: /*30H, DATA transmitted, NACK

        STO = 1;
        AA = 1;
        break;

    //=====
    //Master Mode
    //=====
received*/
    case 0x38: /*38H, arbitration lost*/
        STA = 1; //retry to transmit START if bus free
        break;

    //=====
    //Master Receiver Mode
    //=====
received*/
    case 0x40: /*40H, SLA+R transmitted, ACK

        AA = 1; //ACK next received DATA
        break;
received*/
    case 0x48: /*48H, SLA+R transmitted, NACK

```

```

        STO = 1;
        AA = 1;
        break;
    case 0x50:                                /*50H, DATA received, ACK
transmitted*/
        DATA_RECEIVED1 = I2DAT;            //store received DATA
        if (To_RX_Last_Data1)                //if last DATA will be received
            AA = 0;                          //not ACK next received DATA
        else                                  //if continuing receiving DATA
            AA = 1;
        break;
    case 0x58:                                /*58H, DATA received, NACK
transmitted*/
        DATA_RECEIVED_LAST1 = I2DAT;
        STO = 1;
        AA = 1;
        break;
        //=====
        //Slave Receiver and General Call Mode
        //=====
    case 0x60:                                /*60H, own SLA+W received, ACK
returned*/
        AA = 1;
        break;
    case 0x68:                                /*68H, arbitration lost in SLA+W/R
        own SLA+W received, ACK returned */
        AA = 0;                              //not ACK next received DATA after
        //arbitration lost
        STA = 1;                              //retry to transmit START if bus free
        break;
    case 0x70:                                /*70H, General Call received, ACK
returned
        AA = 1;
        break;
    case 0x78:                                /*78H, arbitration lost in SLA+W/R
        General Call received, ACK
returned*/
        AA = 0;
        STA = 1;
        break;
    case 0x80:                                /*80H, previous own SLA+W, DATA
received,
        ACK returned*/
        DATA_RECEIVED2 = I2DAT;
        if (To_RX_Last_Data2)
            AA = 0;
        else
            AA = 1;
        break;
    case 0x88:                                /*88H, previous own SLA+W, DATA
received,

```

```

mode
DATA_RECEIVED_LAST2 = I2DAT;
AA = 1; //wait for ACK next Master addressing
break;
case 0x90: /*90H, previous General Call, DATA
received, ACK returned*/
DATA_RECEIVED3 = I2DAT;
if (To_RX_Last_Data3)
AA = 0;
else
AA = 1;
break;
case 0x98: /*98H, previous General Call, DATA
received, NACK returned, not addressed SLAVE
mode entered*/
DATA_RECEIVED_LAST3 = I2DAT;
AA = 1;
break;
//=====
//Slave Mode
//=====
case 0xA0: /*A0H, STOP or repeated START
received while still addressed SLAVE mode*/
AA = 1;
break;
//=====
//Slave Transmitter Mode
//=====
case 0xA8: /*A8H, own SLA+R received, ACK
returned*/
I2DAT = NEXT_SEND_DATA3;
AA = 1; //when AA is "1", not last data to be
//transmitted
break;
case 0xB0: /*B0H, arbitration lost in SLA+W/R
own SLA+R received, ACK returned */
I2DAT = DUMMY_DATA;
AA = 0; //when AA is "0", last data to be
//transmitted
STA = 1; //retry to transmit START if bus free
break;
case 0xB8: /*B8H, previous own SLA+R, DATA
transmitted, ACK received*/
I2DAT = NEXT_SEND_DATA4;
if (To_TX_Last_Data) //if last DATA will be transmitted

```

```

        AA = 0;
    else
        AA = 1;
    break;
case 0xC0:
    /*C0H, previous own SLA+R, DATA
    NACK received, not addressed SLAVE
    entered*/
    AA = 1;
    break;
case 0xC8:
    /*C8H, previous own SLA+R, last DATA
    mitted, ACK received, not addressed
    mode entered*/
    AA = 1;
    break;
} //end of switch (I2STAT)

SI = 0;
I2C_ISR
while(STO);
error
} //end of I2C_ISR
    
```

15.5 I²C Time-Out

There is a 14-bit time-out counter, which can be used to deal with the I²C bus hang-up. If the time-out counter is enabled, the counter starts up counting until it overflows. Meanwhile I2TOF will be set by hardware and requests I²C interrupt. When time-out counter is enabled, setting flag SI to high will reset counter and restart counting up after SI is cleared. If the I²C bus hangs up, it causes the SI flag not set for a period. The 14-bit time-out counter will overflow and require the interrupt service.

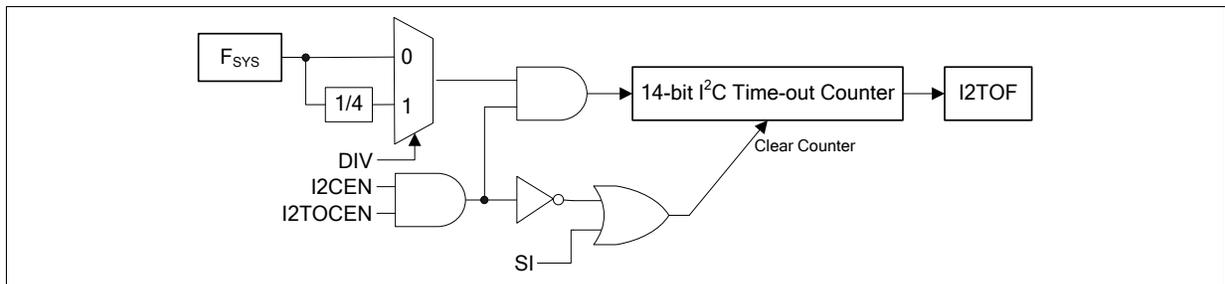


Figure 15-12. I²C Time-Out Counter

I2TOC – I²C Time-out Counter

7	6	5	4	3	2	1	0
-	-	-	-	-	I2TOCEN	DIV	I2TOF
-	-	-	-	-	R/W	R/W	R/W

Address: BFH

Reset value: 0000 0000b

Bit	Name	Description
2	I2TOCEN	I²C time-out counter enable 0 = I ² C time-out counter Disabled. 1 = I ² C time-out counter Enabled. Note: please always enable I ² C interrupt when enable I ² C time-out counter function
1	DIV	I²C time-out counter clock divider 0 = The clock of I ² C time-out counter is F _{SYS} /1. 1 = The clock of I ² C time-out counter is F _{SYS} /4.
0	I2TOF	I²C time-out flag This flag is set by hardware if 14-bit I ² C time-out counter overflows. It is cleared by software.

15.6 I²C Interrupt

There are two I²C flags, SI and I2TOF. Both of them can generate an I²C event interrupt requests. If I²C interrupt mask is enabled via setting EI2C (EIE.0) and EA as 1, CPU will execute the I²C interrupt service routine once any of these two flags is set. User needs to check flags to determine what event caused the interrupt. Both of I²C flags are cleared by software.

16. PIN INTERRUPT

The N76E003 provides pin interrupt input for each I/O pin to detect pin state if button or keypad set is used. A maximum 8-channel pin interrupt detection can be assigned by I/O port sharing. The pin interrupt is generated when any key is pressed on a keyboard or keypad, which produces an edge or level triggering event. Pin interrupt may be used to wake the CPU up from Idle or Power-down mode.

Each channel of pin interrupt can be enabled and polarity controlled independently by PIPEN and PINEN register. PICON selects which port that the pin interrupt is active. It also defines which type of pin interrupt is used – level detect or edge detect. Each channel also has its own interrupt flag. There are total eight pin interrupt flags located in PIF register. The respective flags for each pin interrupt channel allow the interrupt service routine to poll on which channel on which the interrupt event occurs. All flags in PIF register are set by hardware and should be cleared by software.

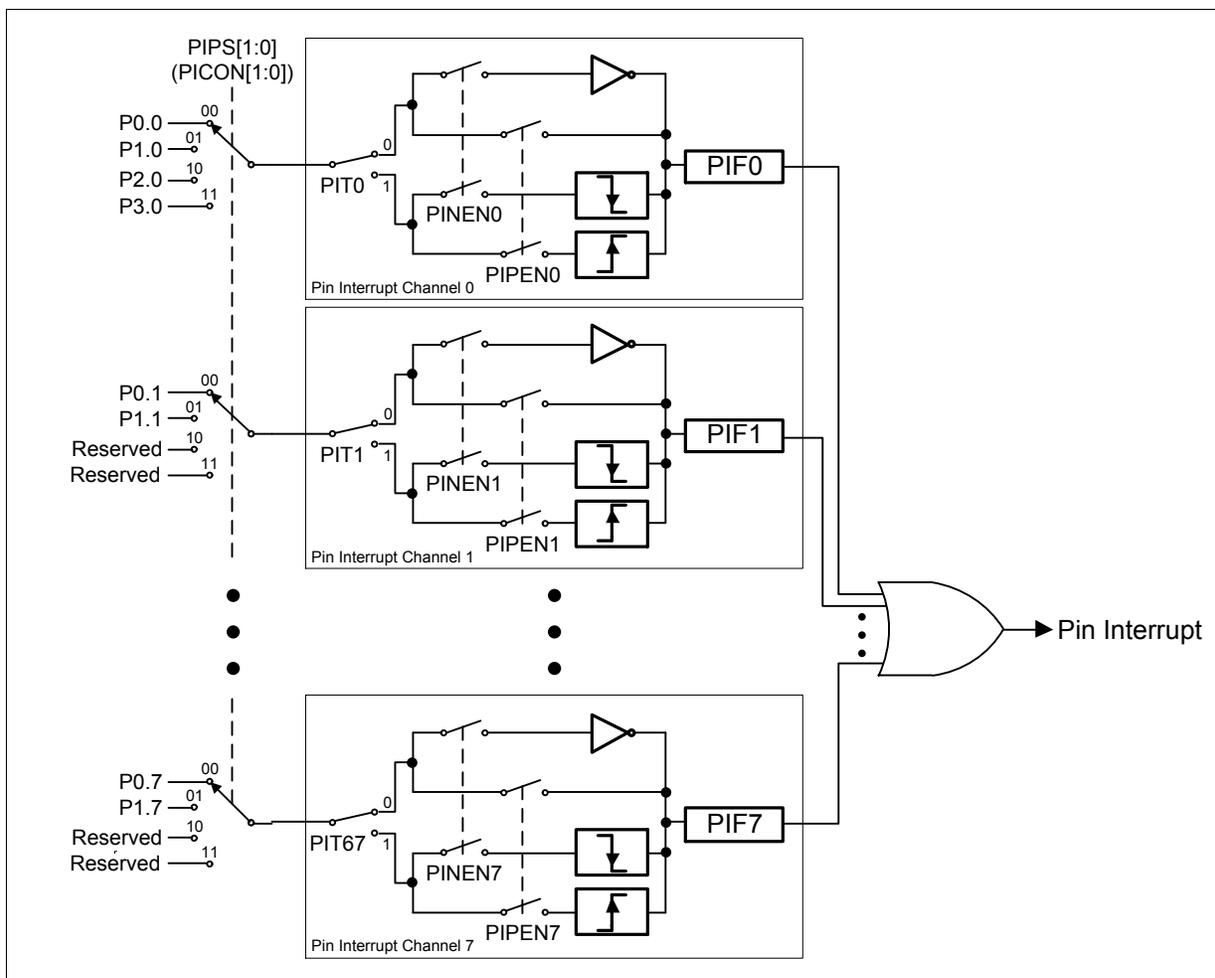


Figure 16-1. Pin Interface Block Diagram

Pin interrupt is generally used to detect an edge transient from peripheral devices like keyboard or keypad. During idle state, the system prefers to enter Power-down mode to minimize power consumption and waits for event trigger. Pin interrupt can wake up the device from Power-down mode.

PICON – Pin Interrupt Control

7	6	5	4	3	2	1	0
PIT67	PIT45	PIT3	PIT2	PIT1	PIT0	PIPS[1:0]	
R/W							

Address: E9H

Reset value: 0000 0000b

Bit	Name	Description
7	PIT67	Pin interrupt channel 6 and 7 type select This bit selects which type that pin interrupt channel 6 and 7 is triggered. 0 = Level triggered. 1 = Edge triggered.
6	PIT45	Pin interrupt channel 4 and 5 type select This bit selects which type that pin interrupt channel 4 and 5 is triggered. 0 = Level triggered. 1 = Edge triggered.
5	PIT3	Pin interrupt channel 3 type select This bit selects which type that pin interrupt channel 3 is triggered. 0 = Level triggered. 1 = Edge triggered.
4	PIT2	Pin interrupt channel 2 type select This bit selects which type that pin interrupt channel 2 is triggered. 0 = Level triggered. 1 = Edge triggered.
3	PIT1	Pin interrupt channel 1 type select This bit selects which type that pin interrupt channel 1 is triggered. 0 = Level triggered. 1 = Edge triggered.
2	PIT0	Pin interrupt channel 0 type select This bit selects which type that pin interrupt channel 0 is triggered. 0 = Level triggered. 1 = Edge triggered.
1:0	PIPS[:0]	Pin interrupt port select This field selects which port is active as the 8-channel of pin interrupt. 00 = Port 0. 01 = Port 1. 10 = Port 2. 11 = Port 3.

PIEN – Pin Interrupt Negative Polarity Enable.

7	6	5	4	3	2	1	0
PINEN7	PINEN6	PINEN5	PINEN4	PINEN3	PINEN2	PINEN1	PINEN0
R/W							

Address: EAH

Reset value: 0000 0000b

Bit	Name	Description
n	PIENn	<p>Pin interrupt channel n negative polarity enable</p> <p>This bit enables low-level/falling edge triggering pin interrupt channel n. The level or edge triggered selection depends on each control bit PITn in PICON.</p> <p>0 = Low-level/falling edge detect Disabled.</p> <p>1 = Low-level/falling edge detect Enabled.</p>

PIPEN – Pin Interrupt Positive Polarity Enable.

7	6	5	4	3	2	1	0
PIPEN7	PIPEN6	PIPEN5	PIPEN4	PIPEN3	PIPEN2	PIPEN1	PIPEN0
R/W							

Address: EBH

Reset value: 0000 0000b

Bit	Name	Description
n	PIPENn	<p>Pin interrupt channel n positive polarity enable</p> <p>This bit enables high-level/rising edge triggering pin interrupt channel n. The level or edge triggered selection depends on each control bit PITn in PICON.</p> <p>0 = High-level/rising edge detect Disabled.</p> <p>1 = High-level/rising edge detect Enabled.</p>

PIF – Pin Interrupt Flags

7	6	5	4	3	2	1	0
PIF7	PIF6	PIF5	PIF4	PIF3	PIF2	PIF1	PIF0
R (level) R/W (edge)							

Address: ECH

Reset value: 0000 0000b

Bit	Name	Description
n	PIFn	<p>Pin interrupt channel n flag</p> <p>If the edge trigger is selected, this flag will be set by hardware if the channel n of pin interrupt detects an enabled edge trigger. This flag should be cleared by software.</p> <p>If the level trigger is selected, this flag follows the inverse of the input signal's logic level on the channel n of pin interrupt. Software cannot control it.</p>

17. PULSE WIDTH MODULATED (PWM)

The PWM (Pulse Width Modulation) signal is a useful control solution in wide application field. It can be used on motor driving, fan control, backlight brightness tuning, LED light dimming, or simulating as a simple digital to analog converter output through a low pass filter circuit.

The N76E003 PWM is especially designed for motor control by providing three pairs, maximum 16-bit resolution of PWM output with programmable period and duty. The architecture makes user easy to drive the one-phase or three-phase brushless DC motor (BLDC), or three-phase AC induction motor. Each of six PWM can be configured as one of independent mode, complementary mode, or synchronous mode. If the complementary mode is used, a programmable dead-time insertion is available to protect MOS turn-on simultaneously. The PWM waveform can be edge-aligned or center-aligned with variable interrupt points.

17.1 Functional Description

17.1.1 PWM Generator

The PWM generator is clocked by the system clock or Timer 1 overflow divided by a PWM clock prescaler selectable from 1/1~1/128. The PWM period is defined by effective 16-bit period registers, {PWMPH, PWMPH}. The period is the same for all PWM channels for they share the same 16-bit period counter. The duty of each PWM is determined independently by the value of duty registers {PWM0H, PWM0L}, {PWM1H, PWM1L}, {PWM2H, PWM2L}, {PWM3H, PWM3L}, {PWM4H, PWM4L}, and {PWM5H, PWM5L}. With six duty registers, six PWM output can be generated independently with different duty cycles. The interval and duty of PWM signal is generated by a 16-bit counter comparing with the period and duty registers.

To facilitate the three-phase motor control, a group mode can be used by setting GP (PWMCON1.5), which makes {PWM0H, PWM0L} and {PWM1H, PWM1L} duty register decide duties of the PWM outputs. In a three-phase motor control application, two-group PWM outputs generally are given the same duty cycle. When the group mode is enabled, {PWM2H, PWM2L}, {PWM3H, PWM3L}, {PWM4H, PWM4L} and {PWM5H, PWM5L} registers have no effect. This means {PWM2H, PWM2L} and {PWM4H, PWM4L} both as same as {PWM0H, PWM0L}. Also {PWM3H, PWM3L} and {PWM5H, PWM5L} are same as {PWM1H, PWM1L}.

Note that enabling PWM does not configure the I/O pins into their output mode automatically. User should configure I/O output mode via software manually.

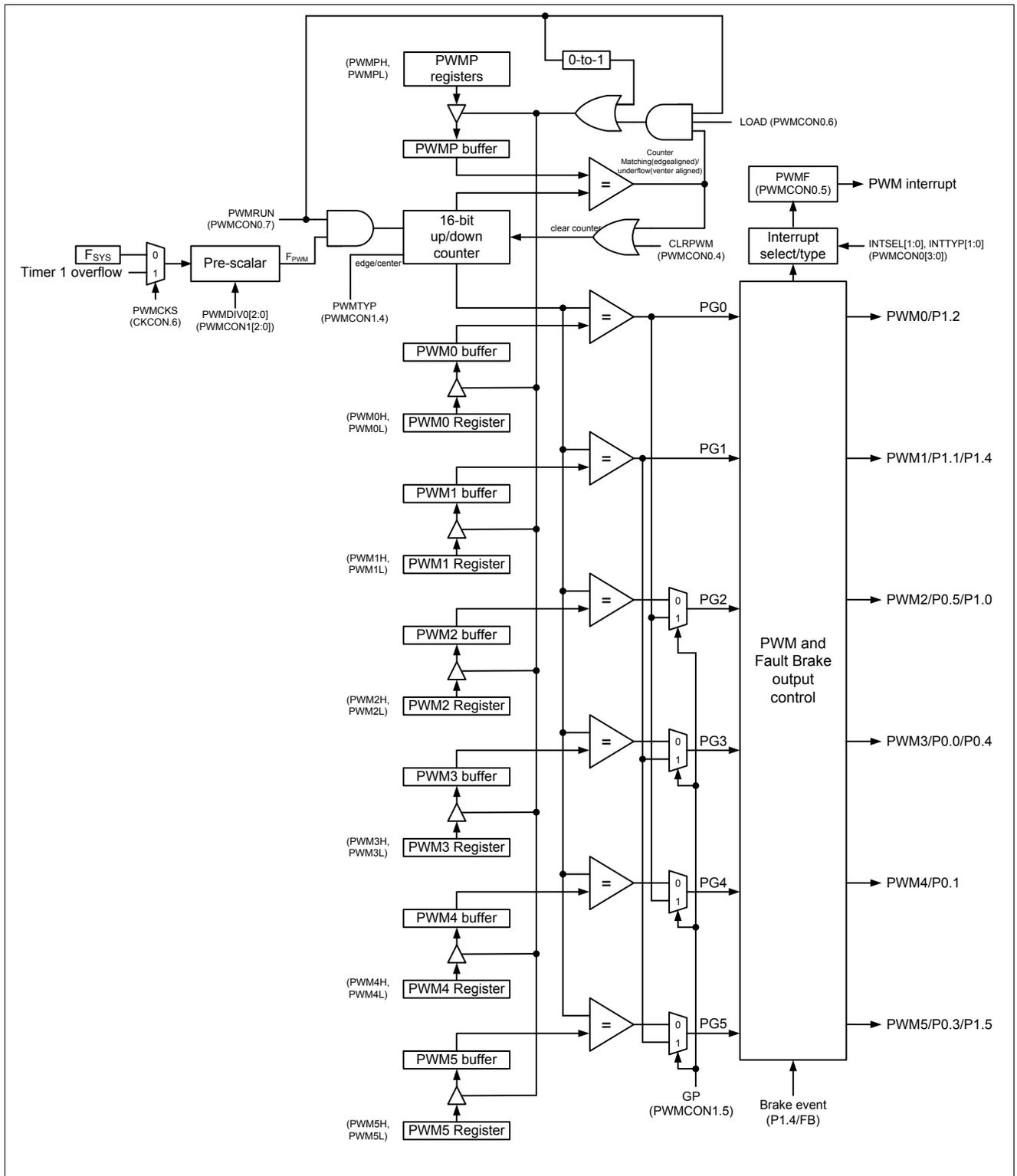


Figure 17-1. PWM Block Diagram

The PWM counter generates six PWM signals called PG0, PG1, PG2, PG3, PG4, and PG5. 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 PIO_n bit to make the corresponding pin function as PWM output. Meanwhile, the general purpose I/O function can be used.

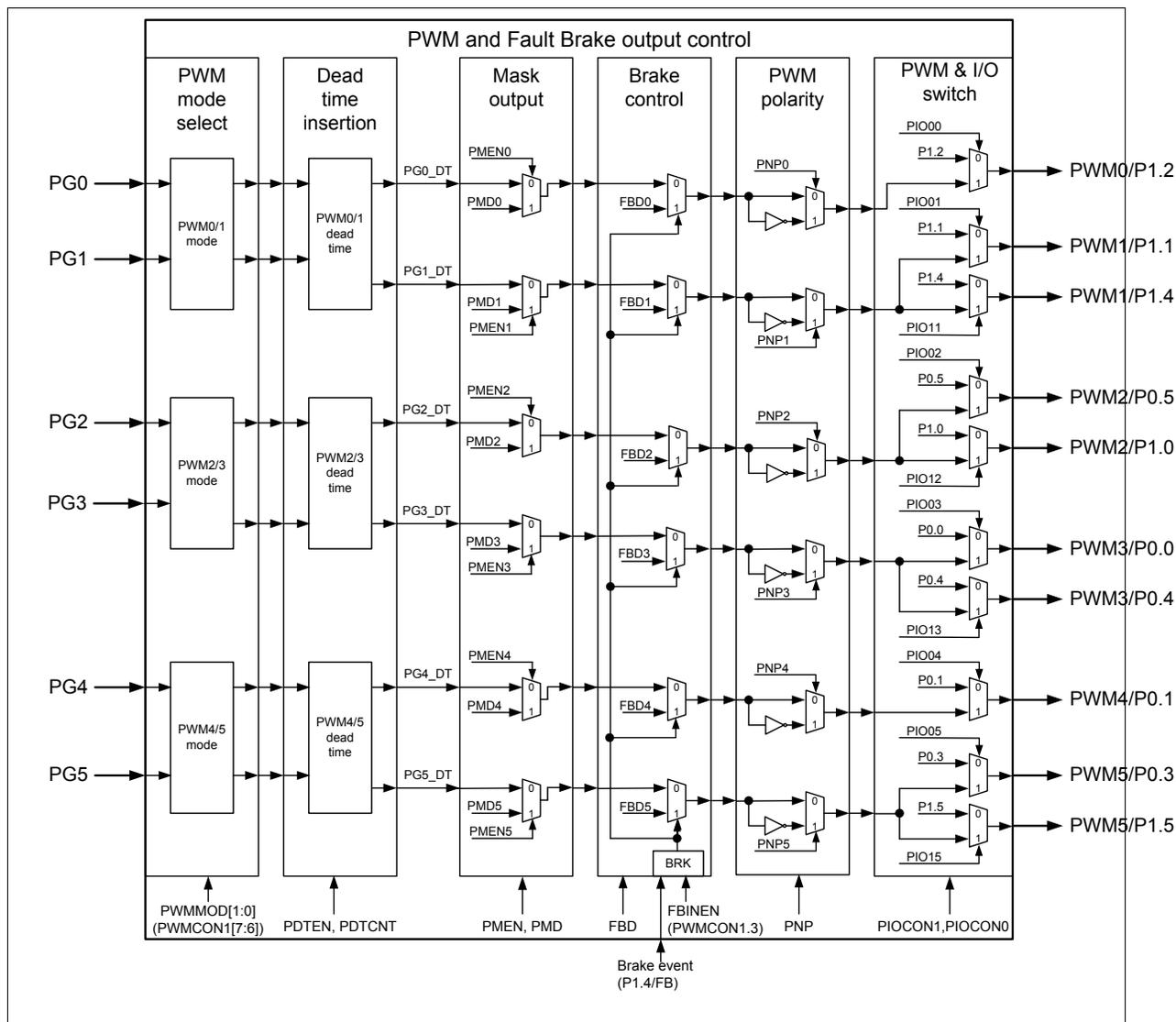


Figure 17-2. PWM and Fault Brake Output Control Block Diagram

User should follow the initialization steps below to start generating the PWM signal output. In the first step by setting CLR PWM (PWMCON0.4), it ensures the 16-bit up counter reset for the accuracy of the first duration. After initialization and setting {PWMPH, PWML} and all {PWMnH, PWMnL} registers, PWMRUN (PWMCON0.7) can be set as logic 1 to trigger the 16-bit counter running. PWM starts to generate waveform on its output pins. The hardware for all period and duty control registers are double buffered designed. Therefore, {PWMPH, PWML} and all {PWMnH, PWMnL} registers can be

written to at any time, but the period and duty cycle of PWM will not be updated immediately until the LOAD (PWMC0.6) is set and previous period is complete. This prevents glitches when updating the PWM period or duty.

A loading of new period and duty by setting LOAD should be ensured complete by monitoring it and waiting for a hardware automatic clearing LOAD bit. Any updating of PWM control registers during LOAD bit as logic 1 will cause unpredictable output.

PWMC0 – PWM Control 0 (Bit-addressable)

7	6	5	4	3	2	1	0
PWMRUN	LOAD	PWMF	CLRPWM	-	-	-	-
R/W	R/W	R/W	R/W	-	-	-	-

Address: D8H

Reset value: 0000 0000b

Bit	Name	Description
7	PWMRUN	PWM run enable 0 = PWM stays in idle. 1 = PWM starts running.
6	LOAD	PWM new period and duty load This bit is used to load period and duty control registers in their buffer if new period or duty value needs to be updated. The loading will act while a PWM period is completed. The new period and duty affected on the next PWM cycle. After the loading is complete, LOAD will be automatically cleared via hardware. The meaning of writing and reading LOAD bit is different. <u>Writing:</u> 0 = No effect. 1 = Load new period and duty in their buffers while a PWM period is completed. <u>Reading:</u> 0 = A loading of new period and duty is finished. 1 = A loading of new period and duty is not yet finished.
5	PWMF	PWM flag This flag is set according to definitions of INTSEL[2:0] and INTTYP[1:0] in PWMINTC. This bit is cleared by software.
4	CLRPWM	Clear PWM counter Setting this bit clears the value of PWM 16-bit counter for resetting to 0000H. After the counter value is cleared, CLRPWM will be automatically cleared via hardware. The meaning of writing and reading CLRPWM bit is different. <u>Writing:</u> 0 = No effect. 1 = Clearing PWM 16-bit counter. <u>Reading:</u> 0 = PWM 16-bit counter is completely cleared. 1 = PWM 16-bit counter is not yet cleared.

PWMCON1 – PWM Control 1

7	6	5	4	3	2	1	0
PWMMOD[1:0]		GP	PWMTYP	FBINEN	PWMDIV[2:0]		
R/W		R/W	R/W	R/W	R/W		

Address: DFH

Reset value: 0000 0000b

Bit	Name	Description
5	GP	Group mode enable This bit enables the group mode. If enabled, the duty of first three pairs of PWM are decided by PWM01H and PWM01L rather than their original duty control registers. 0 = Group mode Disabled. 1 = Group mode Enabled.
2:0	PWMDIV[2:0]	PWM clock divider This field decides the pre-scale of PWM clock source. 000 = 1/1. 001 = 1/2 010 = 1/4. 011 = 1/8. 100 = 1/16. 101 = 1/32. 110 = 1/64. 111 = 1/128.

CKCON – Clock Control

7	6	5	4	3	2	1	0
-	PWMCKS	-	T1M	T0M	-	CLOEN	-
-	R/W	-	R/W	R/W	-	R/W	-

Address: 8EH

Reset value: 0000 0000b

Bit	Name	Description
6	PWMCKS	PWM clock source select 0 = The clock source of PWM is the system clock F_{SYS} . 1 = The clock source of PWM is the overflow of Timer 1.

PWMPL – PWM Period Low Byte

7	6	5	4	3	2	1	0
PWMP[7:0]							
R/W							

Address: D9H

reset value: 0000 0000b

Bit	Name	Description
7:0	PWMP[7:0]	PWM period low byte This byte with PWMPH controls the period of the PWM generator signal.

PWMPH – PWM Period High Byte

7	6	5	4	3	2	1	0
PWMP[15:8]							
R/W							

Address: D1H

reset value: 0000 0000b

Bit	Name	Description
7:0	PWMP[15:8]	PWM period high byte This byte with PWMPH controls the period of the PWM generator signal.

PWM0L – PWM0 Duty Low Byte

7	6	5	4	3	2	1	0
PWM0[7:0]							
R/W							

Address: DAH

reset value: 0000 0000b

Bit	Name	Description
7:0	PWM0[7:0]	PWM0 duty low byte This byte with PWM0H controls the duty of the output signal PG0 from PWM generator.

PWM0H – PWM0 Duty High Byte

7	6	5	4	3	2	1	0
PWM0[15:8]							
R/W							

Address: D2H

reset value: 0000 0000b

Bit	Name	Description
7:0	PWM0[15:8]	PWM0 duty high byte This byte with PWM0L controls the duty of the output signal PG0 from PWM generator.

PWM1L – PWM/1 Duty Low Byte

7	6	5	4	3	2	1	0
PWM1[7:0]							
R/W							

Address: DBH

reset value: 0000 0000b

Bit	Name	Description
7:0	PWM1[7:0]	PWM1 duty low byte This byte with PWM1H controls the duty of the output signal PG1 from PWM generator.

PWM1H – PWM1 Duty High Byte

7	6	5	4	3	2	1	0
PWM1[15:8]							
R/W							

Address: D3H

reset value: 0000 0000b

Bit	Name	Description
7:0	PWM1[15:8]	PWM1 duty high byte This byte with PWM1L controls the duty of the output signal PG1 from PWM generator.

PWM2L – PWM2 Duty Low Byte

7	6	5	4	3	2	1	0
PWM2[7:0]							
R/W							

Address: DCH

reset value: 0000 0000b

Bit	Name	Description
7:0	PWM2[7:0]	PWM2 duty low byte This byte with PWM2H controls the duty of the output signal PG2 from PWM generator.

PWM2H – PWM2 Duty High Byte

7	6	5	4	3	2	1	0
PWM2[15:8]							
R/W							

Address: D4H

reset value: 0000 0000b

Bit	Name	Description
7:0	PWM2[15:8]	PWM2 duty high byte This byte with PWM2L controls the duty of the output signal PG2 from PWM generator.

PWM3L – PWM3 Duty Low Byte

7	6	5	4	3	2	1	0
PWM3[7:0]							
R/W							

Address: DDH

reset value: 0000 0000b

Bit	Name	Description
7:0	PWM3[7:0]	PWM3 duty low byte This byte with PWM3H controls the duty of the output signal PG3 from PWM generator.

PWM3H – PWM3 Duty High Byte

7	6	5	4	3	2	1	0
PWM3[15:8]							
R/W							

Address: D5H

reset value: 0000 0000b

Bit	Name	Description
7:0	PWM3[15:8]	PWM3 duty high byte This byte with PWM3L controls the duty of the output signal PG3 from PWM generator.

PWM4L – PWM4 Duty Low Byte

7	6	5	4	3	2	1	0
PWM4[7:0]							
R/W							

Address: CCH, Page:1

reset value: 0000 0000b

Bit	Name	Description
7:0	PWM4[7:0]	PWM4 duty low byte This byte with PWM4H controls the duty of the output signal PG4 from PWM generator.

PWM4H – PWM4 Duty High Byte

7	6	5	4	3	2	1	0
PWM4[15:8]							
R/W							

Address: C4H, Page:1

reset value: 0000 0000b

Bit	Name	Description
7:0	PWM4[15:8]	PWM4 duty high byte This byte with PWM4L controls the duty of the output signal PG4 from PWM generator.

PWM5L – PWM5 Duty Low Byte

7	6	5	4	3	2	1	0
PWM5[7:0]							
R/W							

Address: CDH, Page:1

reset value: 0000 0000b

Bit	Name	Description
7:0	PWM5[7:0]	PWM5 duty low byte This byte with PWM5H controls the duty of the output signal PG5 from PWM generator.

PWM5H – PWM5 Duty High Byte

7	6	5	4	3	2	1	0
PWM5[15:8]							
R/W							

Address: C5H, Page:1

reset value: 0000 0000b

Bit	Name	Description
7:0	PWM5[15:8]	PWM5 duty high byte This byte with PWM5L controls the duty of the output signal PG5 from PWM generator.

PIOCON0 – PWM or I/O Select

7	6	5	4	3	2	1	0
-	-	PIO05	PIO04	PIO03	PIO02	PIO01	PIO00
-	-	R/W	R/W	R/W	R/W	R/W	R/W

Address: DEH

Reset value: 0000 0000b

Bit	Name	Description
5	PIO05	P0.3/PWM5 pin function select 0 = P0.3/PWM5 pin functions as P0.3. 1 = P0.3/PWM5 pin functions as PWM5 output.
4	PIO04	P0.1/PWM4 pin function select 0 = P0.1/PWM4 pin functions as P0.1. 1 = P0.1/PWM4 pin functions as PWM4 output.
3	PIO03	P0.0/PWM3 pin function select 0 = P0.0/PWM3 pin functions as P0.0. 1 = P0.0/PWM3 pin functions as PWM3 output.
2	PIO02	P1.0/PWM2 pin function select 0 = P1.0/PWM2 pin functions as P1.0. 1 = P1.0/PWM2 pin functions as PWM2 output.
1	PIO01	P1.1/PWM1 pin function select 0 = P1.1/PWM1 pin functions as P1.1. 1 = P1.1/PWM1 pin functions as PWM1 output.
0	PIO00	P1.2/PWM0 pin function select 0 = P1.2/PWM0 pin functions as P1.2. 1 = P1.2/PWM0 pin functions as PWM0 output.

PIOCON1 – PWM or I/O Select

7	6	5	4	3	2	1	0
-	-	PIO15	-	PIO13	PIO12	PIO11	-
-	-	R/W	-	R/W	R/W	R/W	-

Address: C6H, Page:1

Reset value: 0000 0000b

Bit	Name	Description
5	PIO15	P1.5/PWM5 pin function select 0 = P1.5/PWM5 pin functions as P1.5. 1 = P1.5/PWM5 pin functions as PWM5 output.

Bit	Name	Description
3	PIO13	P0.4/PWM3 pin function select 0 = P0.4/PWM3 pin functions as P0.4. 1 = P0.4/PWM3 pin functions as PWM3 output.
2	PIO12	P0.5/PWM2 pin function select 0 = P0.5/PWM2 pin functions as P0.5. 1 = P0.5/PWM2 pin functions as PWM2 output.
1	PIO11	P1.4/PWM1 pin function select 0 = P1.4/PWM1 pin functions as P1.4. 1 = P1.4/PWM1 pin functions as PWM1 output.

17.1.2 PWM Types

The PWM generator provides two PWM types: edge-aligned or center-aligned. PWM type is selected by PWMTYP (PWMCON1.4).

PWMCON1 – PWM Control 1

7	6	5	4	3	2	1	0
PWMMOD[1:0]		GP	PWMTYP	FBINEN	PWMDIV[2:0]		
R/W		R/W	R/W	R/W	R/W		

Address: DFH

Reset value: 0000 0000b

Bit	Name	Description
4	PWMTYP	PWM type select 0 = Edge-aligned PWM. 1 = Center-aligned PWM.

17.1.2.1 Edge-Aligned Type

In edge-aligned mode, the 16-bit counter uses single slop operation by counting up from 0000H to {PWMPH, PWMPH} and then starting from 0000H. The PWM generator signal (PGn before PWM and Fault Brake output control) is cleared on the compare match of 16-bit counter and the duty register {PWMnH, PWMnL} and set at the 16-bit counter is 0000H. The result PWM output waveform is left-edge aligned.

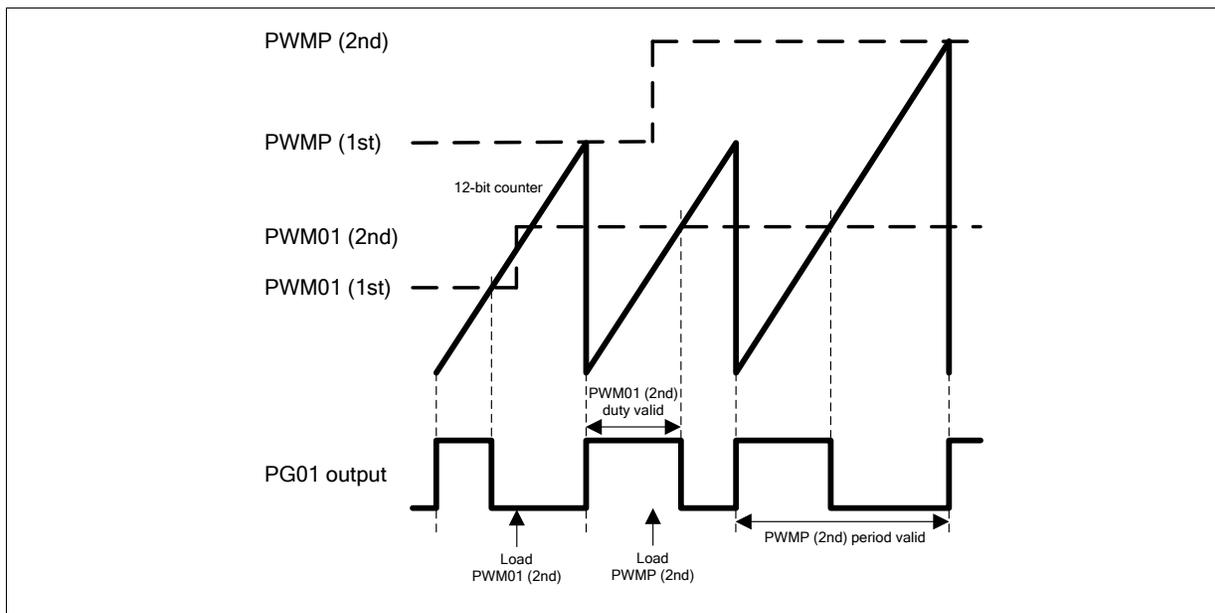


Figure 17-3. PWM Edge-aligned Type Waveform

The output frequency and duty cycle for edge-aligned PWM are given by following equations:

$$\text{PWM frequency} = \frac{F_{\text{PWM}}}{\{\text{PWMPH}, \text{PWML}\} + 1}$$

(F_{PWM} is the PWM clock source frequency divided by PWMDIV).

$$\text{PWM high level duty} = \frac{\{\text{PWMnH}, \text{PWMnL}\}}{\{\text{PWMPH}, \text{PWML}\} + 1}$$

17.1.2.2 Center-Aligned Type

In center-aligned mode, the 16-bit counter use dual slop operation by counting up from 0000H to {PWMPH, PWML} and then counting down from {PWMPH, PWML} to 0000H. The PGN signal is cleared on the up-count compare match of 16-bit counter and the duty register {PWMnH, PWMnL} and set on the down-count compare match. Center-aligned PWM may be used to generate non-overlapping waveforms.

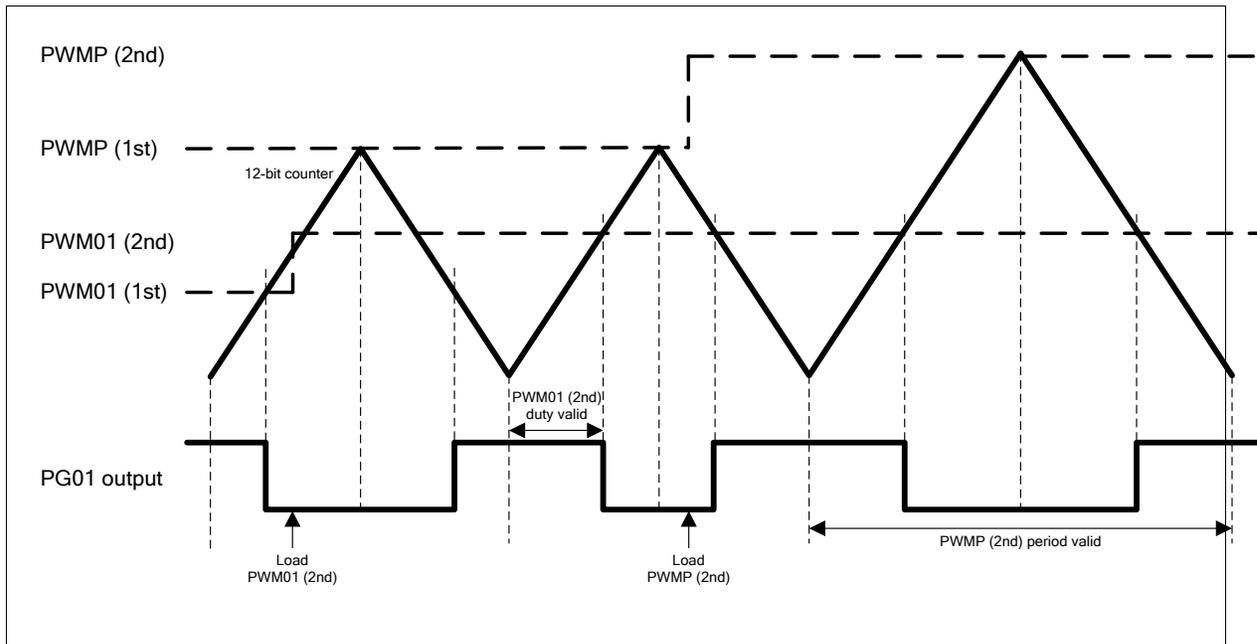


Figure 17-4. PWM Center-aligned Type Waveform

The output frequency and duty cycle for center-aligned PWM are given by following equations:

PWM frequency = $\frac{F_{PWM}}{2 \times \{PWMPH, PWMP L\}}$ (F_{PWM} is the PWM clock source frequency divided by PWMDIV).

PWM high level duty = $\frac{\{PWMnH, PWMnL\}}{\{PWMPH, PWMP L\}}$.

17.1.3 Operation Modes

After PGN signals pass through the first stage of the PWM and Fault Brake output control circuit. The PWM mode selection circuit generates different kind of PWM output modes with six-channel, three-pair signal PG0~PG5 . It supports independent mode, complementary mode, and synchronous mode.

PWMCON1 – PWM Control 1

7	6	5	4	3	2	1	0
PWMMOD[1:0]		GP	PWMTYP	FBINEN	PWMDIV[2:0]		
R/W		R/W	R/W	R/W	R/W		

Address: DFH

Reset value: 0000 0000b

Bit	Name	Description
7:6	PWMMOD[1:0]	PWM mode select 00 = Independent mode. 01 = Complementary mode. 10 = Synchronized mode. 11 = Reserved.

17.1.3.1 Independent Mode

Independent mode is enabled when PWMMOD[1:0] (PWMCON1[7:6]) is [0:0]. It is the default mode of PWM. PG0, PG1, PG2, PG3, PG4 and PG5 output PWM signals independently.

17.1.3.2 Complementary Mode with Dead-Time Insertion

Complementary mode is enabled when PWMMOD[1:0] = [0:1]. In this mode, PG0/2/4 output PWM signals the same as the independent mode. However, PG1/3/5 output the out-phase PWM signals of PG0/2/4 correspondingly, and ignore PG1/3/5 Duty register {PWMnH, PWMnL} (n:1/3/5). This mode makes PG0/PG1 a PWM complementary pair and so on PG2/PG3 and PG4/PG5.

In a real motor application, a complementary PWM output always has a need of “dead-time” insertion to prevent damage of the power switching device like GPIBs due to being active on simultaneously of the upper and lower switches of the half bridge, even in a “μs” duration. For a power switch device physically cannot switch on/off instantly. For the N76E003 PWM, each PWM pair share a 9-bit dead-time down-counter PDMCNT used to produce the off time between two PWM signals in the same pair. On implementation, a 0-to-1 signal edge delays after PDMCNT timer underflows. The timing diagram illustrates the complementary mode with dead-time insertion of PG0/PG1 pair. Pairs of PG2/PG3 and PG4/PG5 have the same dead-time circuit. Each pair has its own dead-time enabling bit in the field of PD TEN[3:0].

Note that the PDMCNT and PD TEN registers are all TA write protection. The dead-time control are also valid only when the PWM is configured in its complementary mode.

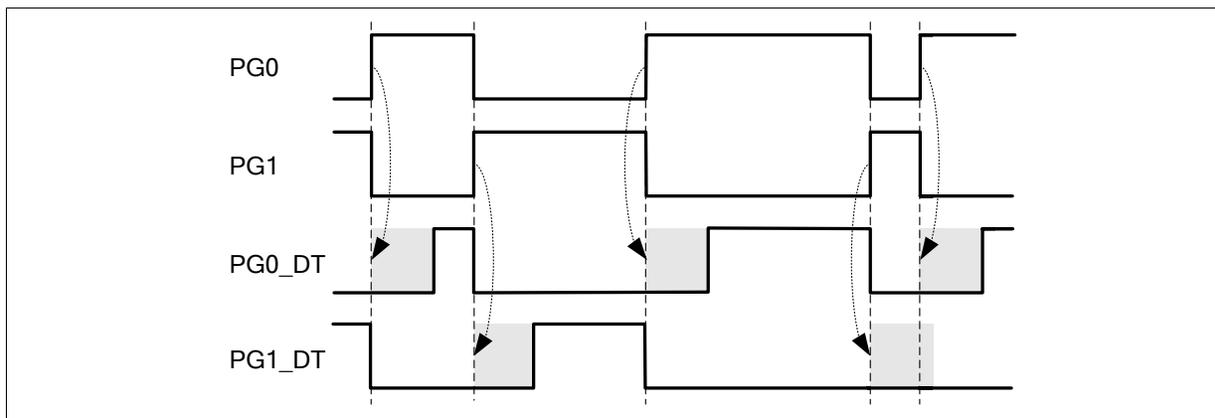


Figure 17-5. PWM Complementary Mode with Dead-time Insertion

PDTEN – PWM Dead-time Enable (TA protected)

7	6	5	4	3	2	1	0
-	-	-	PDTCNT.8	-	PDT45EN	PDT23EN	PDT01EN
-	-	-	R/W	-	R/W	R/W	R/W

Address: F9H

Reset value: 0000 0000b

Bit	Name	Description
4	PDTCNT.8	PWM dead-time counter bit 8 See PDTCNT register.
2	PDT45EN	PWM4/5 pair dead-time insertion enable This bit is valid only when PWM4/5 is under complementary mode. 0 = No delay on GP4/GP5 pair signals. 1 = Insert dead-time delay on the rising edge of GP4/GP5 pair signals.
1	PDT23EN	PWM2/3 pair dead-time insertion enable This bit is valid only when PWM2/3 is under complementary mode. 0 = No delay on GP2/GP3 pair signals. 1 = Insert dead-time delay on the rising edge of GP2/GP3 pair signals.
0	PDT01EN	PWM0/1 pair dead-time insertion enable This bit is valid only when PWM0/1 is under complementary mode. 0 = No delay on GP0/GP1 pair signals. 1 = Insert dead-time delay on the rising edge of GP0/GP1 pair signals.

PDTCNT – PWM Dead-time Counter (TA protected)

7	6	5	4	3	2	1	0
PDTCNT[7:0]							
R/W							

Address: FAH

Reset value: 0000 0000b

Bit	Name	Description
7:0	PDTCNT[7:0]	<p>PWM dead-time counter low byte</p> <p>This 8-bit field combined with PDTEN.4 forms a 9-bit PWM dead-time counter PDTCNT. This counter is valid only when PWM is under complementary mode and the correspond PDTEN bit for PWM pair is set.</p> $\text{PWM dead-time} = \frac{\text{PDTCNT} + 1}{F_{\text{SYS}}}$ <p>Note that user should not modify PDTCNT during PWM run time.</p>

17.1.3.3 Synchronous Mode

Synchronous mode is enabled when PWMMOD[1:0] = [1:0]. In this mode, PG0/2/4 output PWM signals the same as the independent mode. PG1/3/5 output just the same in-phase PWM signals of PG0/2/4 correspondingly.

17.1.4 Mask Output Control

Each PWM signal can be software masked by driving a specified level of PWM signal. The PWM mask output function is quite useful when controlling Electrical Commutation Motor like a BLDC. PMEN contains six bits, those determine which channel of PWM signal will be masked. PMD set the individual mask level of each PWM channel. The default value of PMEN is 00H, which makes all outputs of PWM channels follow signals from PWM generator. Note that the masked level is reversed or not by PNP setting on PWM output pins.

PMEN – PWM Mask Enable

7	6	5	4	3	2	1	0
-	-	PMEN5	PMEN4	PMEN3	PMEN2	PMEN1	PMEN0
-	-	R/W	R/W	R/W	R/W	R/W	R/W

Address: FBH

Reset value: 0000 0000b

Bit	Name	Description
n	PMENn	<p>PWMn mask enable</p> <p>0 = PWMn signal outputs from its PWM generator. 1 = PWMn signal is masked by PMDn.</p>

PMD – PWM Mask Data

7	6	5	4	3	2	1	0
-	-	PMD5	PMD4	PMD3	PMD2	PMD1	PMD0
-	-	R/W	R/W	R/W	R/W	R/W	R/W

Address: FCH

Reset value: 0000 0000b

Bit	Name	Description
n	PMDn	PWMn mask data The PWMn signal outputs mask data once its corresponding PMENn is set. 0 = PWMn signal is masked by 0. 1 = PWMn signal is masked by 1.

17.1.5 Fault Brake

The Fault Brake function is usually implemented in conjunction with an enhanced PWM circuit. It rules as a fault detection input to protect the motor system from damage. Fault Brake pin input (FB) is valid when FBINEN (PWMCON1.3) is set. When Fault Brake is asserted PWM signals will be individually overwritten by FBD corresponding bits. PWMRUN (PWMCON0.7) will also be automatically cleared by hardware to stop PWM generating. The PWM 16-bit counter will also be reset as 0000H. A indicating flag FBF will be set by hardware to assert a Fault Brake interrupt if enabled. FBD data output remains even after the FBF is cleared by software. User should resume the PWM output only by setting PWMRUN again. Meanwhile the Fault Brake state will be released and PWM waveform outputs on pins as usual. Fault Brake input has a polarity selection by FBINLS (FBD.6) bit. Note that the Fault Brake signal feed in FB pin should be longer than eight-system-clock time for FB pin input has a permanent $8/F_{SYS}$ de-bouncing, which avoids fake Fault Brake event by input noise. The other path to trigger a Fault Brake event is the ADC compare event. It asserts the Fault Brake behavior just the same as FB pin input. See [Sector 18.1.3 “ADC Conversion Result Comparator” on page 193](#).

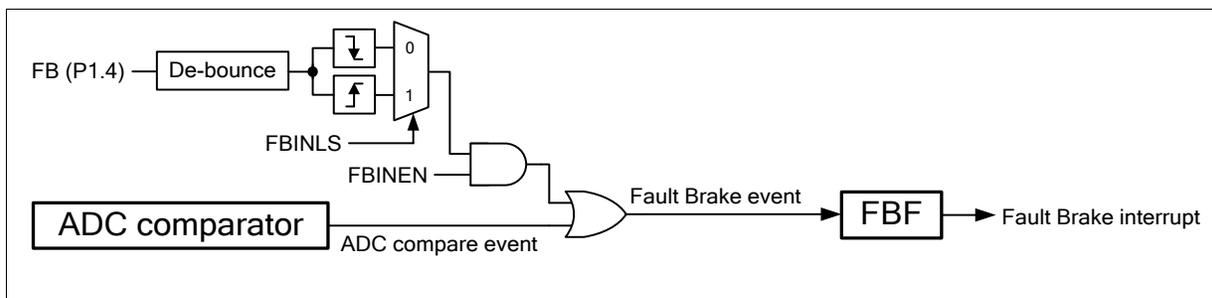


Figure 17-6. Fault Brake Function Block Diagram

PWMCON1 – PWM Control 1

7	6	5	4	3	2	1	0
PWMMOD[1:0]		GP	PWMTYP	FBINEN	PWMDIV[2:0]		
R/W		R/W	R/W	R/W	R/W		

Address: DFH

Reset value: 0000 0000b

Bit	Name	Description
3	FBINEN	FB pin input enable 0 = PWM output Fault Braked by FB pin input Disabled. 1 = PWM output Fault Braked by FB pin input Enabled. Once an edge, which matches FBINLS (FBD.6) selection, occurs on FB pin, PWM0~5 output Fault Brake data in FBD register and PWM6/7 remains their states. PWMRUN (PWMCON0.7) will also be automatically cleared by hardware. The PWM output resumes when PWMRUN is set again.

FBD – PWM Fault Brake Data

7	6	5	4	3	2	1	0
FBF	FBINLS	FBD5	FBD4	FBD3	FBD2	FBD1	FBD0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Address: D7H

Reset value: 0000 0000b

Bit	Name	Description
7	FBF	Fault Brake flag This flag is set when FBINEN is set as 1 and FB pin detects an edge, which matches FBINLS (FBD.6) selection. This bit is cleared by software. After FBF is cleared, Fault Brake data output will not be released until PWMRUN (PWMCON0.7) is set.
6	FBINLS	FB pin input level selection 0 = Falling edge. 1 = Rising edge.
N	FBDn	PWMn Fault Brake data 0 = PWMn signal is overwritten by 0 once Fault Brake asserted. 1 = PWMn signal is overwritten by 1 once Fault Brake asserted.

17.1.6 Polarity Control

Each PWM output channel has its independent polarity control bit, PNP0~PNP5. The default is high active level on all control fields implemented with positive logic. It means the power switch is ON when PWM outputs high level and OFF when low level. User can easily configure all setting with positive logic and then set PNP bit to make PWM actually outputs according to the negative logic.

PNP – PWM Negative Polarity

7	6	5	4	3	2	1	0
-	-	PNP5	PNP4	PNP3	PNP2	PNP1	PNP0
-	-	R/W	R/W	R/W	R/W	R/W	R/W

Address: D6H

Reset value: 0000 0000b

Bit	Name	Description
n	PNPn	PWMn negative polarity output enable 0 = PWMn signal outputs directly on PWMn pin. 1 = PWMn signal outputs inversely on PWMn pin.

17.2 PWM Interrupt

The PWM module has a flag PWMF (PWMCON0.5) to indicate certain point of each complete PWM period. The indicating PWM channel and point can be selected by INTSEL[2:0] and INTTYP[1:0] (PWMINTC[2:0] and [5:4]). Note that the center point and the end point interrupts are only available when PWM operates in its center-aligned type. PWMF is cleared by software.

PWMINTC – PWM Interrupt Control

7	6	5	4	3	2	1	0
-	-	INTTYP1	INTTYP0	-	INTSEL2	INTSEL1	INTSEL0
-	-	R/W	R/W	-	R/W	R/W	R/W

Address: B7H, Page:1

Reset value: 0000 0000b

Bit	Name	Description
5:4	INTTYP[1:0]	PWM interrupt type select These bit select PWM interrupt type. 00 = Falling edge on PWM0/1/2/3/4/5 pin. 01 = Rising edge on PWM0/1/2/3/4/5 pin. 10 = Central point of a PWM period. 11 = End point of a PWM period. Note that the central point interrupt or the end point interrupt is only available while PWM operates in center-aligned type.
2:0	INTSEL[2:0]	PWM interrupt pair select These bits select which PWM channel asserts PWM interrupt when PWM interrupt type is selected as falling or rising edge on PWM0/1/2/3/4/5 pin.. 000 = PWM0. 001 = PWM1. 010 = PWM2. 011 = PWM3. 100 = PWM4. 101 = PWM5. Others = PWM0.

The PWM interrupt related with PWM waveform is shown as figure below.

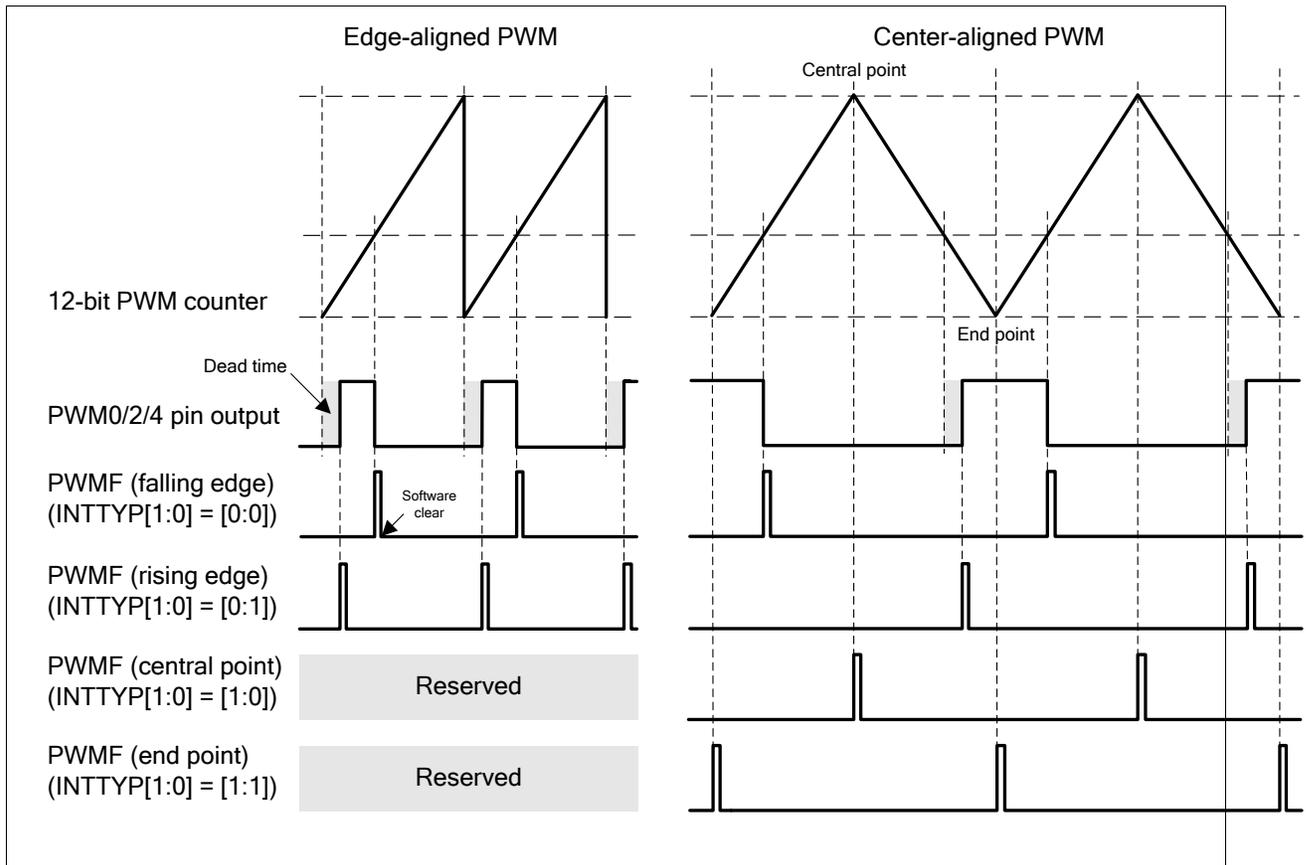


Figure 17-7. PWM Interrupt Type

Fault Brake event requests another interrupt, Fault Brake interrupt. It has different interrupt vector from PWM interrupt. When either Fault Brake pin input event or ADC compare event occurs, FBF (FBD.7) will be set by hardware. It generates Fault Brake interrupt if enabled. The Fault Brake interrupt enable bit is EFB (EIE.5). FBF is cleared via software.

18. 12-BIT ANALOG-TO-DIGITAL CONVERTER (ADC)

The N76E003 is embedded with a 12-bit SAR ADC. The ADC (analog-to-digital converter) allows conversion of an analog input signal to a 12-bit binary representation of that signal. The N76E003 is selected as 8-channel inputs in single end mode. The internal band-gap voltage also can be the internal ADC input. The analog input, multiplexed into one sample and hold circuit, charges a sample and hold capacitor. The output of the sample and hold capacitor is the input into the converter. The converter then generates a digital result of this analog level via successive approximation and stores the result in the result registers.

18.1 Functional Description

18.1.1 ADC Operation

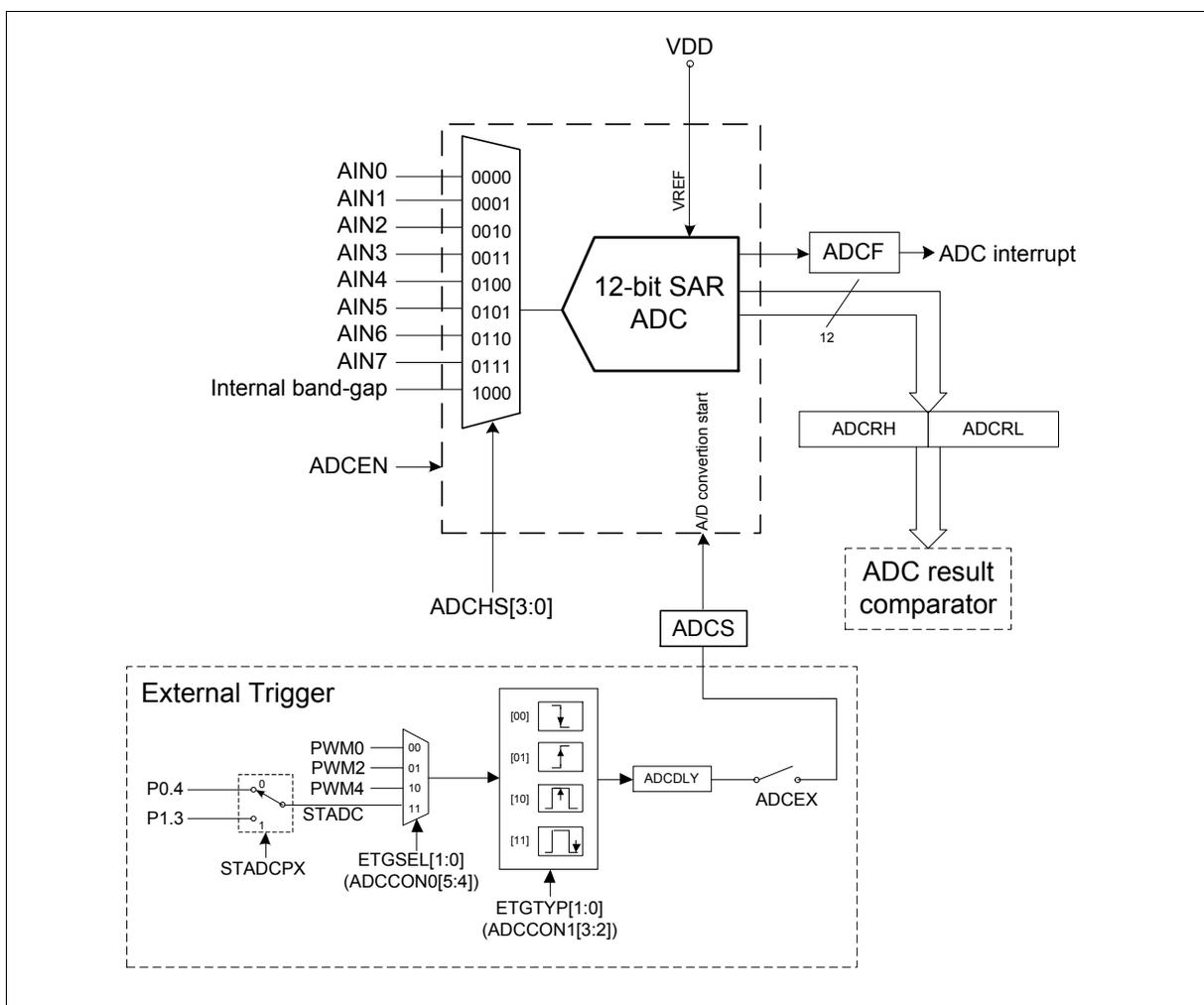


Figure 18-1. 12-bit ADC Block Diagram

Before ADC operation, the ADC circuit should be enabled by setting ADCEN (ADCCON1.0). This makes ADC circuit active. It consume extra power. Once ADC is not used, clearing ADCEN to turn off ADC circuit saves power.

The ADC analog input pin should be specially considered. ADCHS[2:0] are channel selection bits that control which channel is connected to the sample and hold circuit. User needs to configure selected ADC input pins as input-only (high impedance) mode via respective bits in PxMn registers. This configuration disconnects the digital output circuit of each selected ADC input pin. But the digital input circuit still works. Digital input may cause the input buffer to induce leakage current. To disable the digital input buffer, the respective bits in AINDIDS should be set. Configuration above makes selected ADC analog input pins pure analog inputs to allow external feeding of the analog voltage signals. Also, the ADC clock rate needs to be considered carefully. The ADC maximum clock frequency is listed in Table 31-9. ADC Electrical Characteristics Clock above the maximum clock frequency degrades ADC performance unpredictably.

An A/D conversion is initiated by setting the ADCS bit (ADCCON0.6). When the conversion is complete, the hardware will clear ADCS automatically, set ADCF (ADCCON0.7) and generate an interrupt if enabled. The new conversion result will also be stored in ADCRH (most significant 8 bits)

and ADCRL (least significant 4 bits). The 12-bit ADC result value is $4095 \times \frac{V_{AIN}}{V_{REF}}$.

By the way, digital circuitry inside and outside the device generates noise which might affect the accuracy of ADC measurements. If conversion accuracy is critical, the noise level can be reduced by applying the following techniques:

1. Keep analog signal paths as short as possible. Make sure to run analog signals tracks well away from high-speed digital tracks.
2. Place the device in Idle mode during a conversion.
3. If any AIN pins are used as digital outputs, it is essential that these do not switch while a conversion is in progress.

18.1.2 ADC Conversion Triggered by External Source

Besides setting ADCS via software, the N76E003 is enhanced by supporting hardware triggering method to start an A/D conversion. If ADCEX (ADCCON1.1) is set, edges or period points on selected PWM channel or edges of STADC pin will automatically trigger an A/D conversion. (The hardware trigger also sets ADCS by hardware.) For application flexibility, STADC pin can be exchanged by STADCPX (ADCCON1.6).

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 triggering ADC hardware with controllable trigger delay makes the N76E003 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.

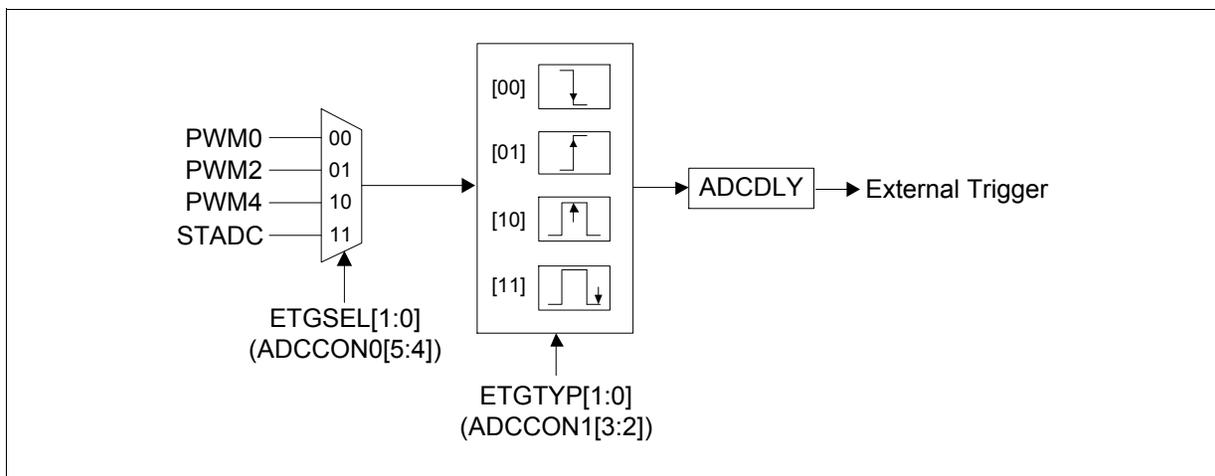


Figure 18-2. External Triggering ADC Circuit

18.1.3 ADC Conversion Result Comparator

The N76E003 ADC has a digital comparator, which compares the A/D conversion result with a 12-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 129.

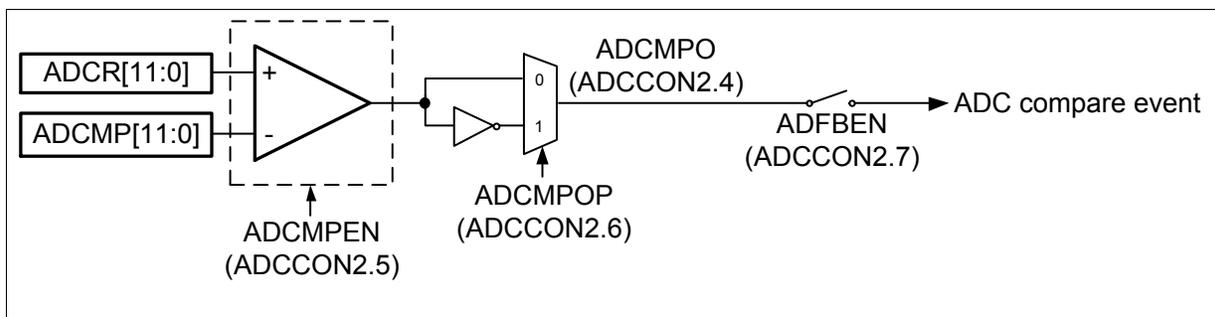


Figure 18-3. ADC Result Comparator

18.1.4 Internal Band-gap

At room temperature, all N76E003 band-gap voltage values will be calibrated within the range of 1.17V to 1.30V. If you want to get the actual band-gap value for N76E003, read the 2 bytes value after the UID address and the actually valid bit is 12. The first byte is the upper 8 bits, and the lower 4 bits of the second byte are the lower 4 bits of the 12 bit.

Reading and calculation steps:

1. Read a band-gap value with IAP by reading UID;
2. Merge the upper 8 bits and the lower 4 bits;
3. Use the following formula to convert to an actual voltage value.

Formula as following

$$Bandgap_Voltage = \frac{Bandgap_Value \times 3072}{4096} (mV)$$

For example:

Read the 2 bytes value after the UID address, wherein the first byte value is 0x64, and the second byte value is 0x0E, merged as 0x64E = 1614. The conversion result is as follows:

$$Bandgap_Voltage = \frac{1614 \times 3072}{4096} = 1210.5 (mV)$$

```
#define set_IAPEN
BIT_TMP=EA;EA=0;TA=0xAA;TA=0x55;CHPCON|=SET_BIT0 ;EA=BIT_TMP
#define set_IAPGO
BIT_TMP=EA;EA=0;TA=0xAA;TA=0x55;IAPTRG|=SET_BIT0 ;EA=BIT_TMP
#define clr_IAPEN
BIT_TMP=EA;EA=0;TA=0xAA;TA=0x55;CHPCON&=~SET_BIT0;EA=BIT_TMP

void READ_BANDGAP()
{
    UINT8 BandgapHigh,BandgapLow;
    Set_IAPEN; // Enable IAPEN
    IAPAL = 0x0C;
    IAPAH = 0x00;
    IAPCN = 0x04;
    set_IAPGO; // Trig set IAPGO
    BandgapHigh = IAPFD;
    IAPAL = 0x0d;
```

```

IAPAH = 0x00;
IAPCN = 0x04;
set_IAPGO; // Trig set IAPGO
BandgapLow = IAPFD;
BandgapLow = BandgapLow&0x0F;
Clr_IAPEN; // Disable IAPEN
Bandgap_Value = (BandgapHigh<<4)+BandgapLow;
Bandgap_voltage = 3072/(0x1000/Bandgap_Value);
}

```

Band-gap as ADC input to calculate the VDD value:

N76E003 internal embedded band-gap voltage also can be the internal ADC input. This input is useful to measure Vref value then means can know the VDD value from ADC convert result. For a more accuracy result when band-gap as ADC input, always give up the first three times convert data in register after ADC enable.

```

double Bandgap_voltage,VDD_voltage;
void ADC_Bypass (void) // The first three times convert should be
bypass
{
    unsigned char ozc;
    for (ozc=0;ozc<0x03;ozc++)
    {
        clr_ADCF;
        set_ADCS;
        while(ADCF == 0);
    }
}
void main (void)
{
    double bgvalue;
    READ_BANDGAP();
    Enable_ADC_BandGap;
    ADC_Bypass();
    clr_ADCF;
    set_ADCS;
    while(ADCF == 0);
    bgvalue = (ADCRH<<4) + ADCRL;
    VDD_voltage = (0xFFF/bgvalue)*Bandgap_voltage;
}

```

```
        printf ("\n Bandgap voltage = %e", Bandgap_Voltage);  
        printf ("\n VDD voltage = %e", VDD_Voltage);  
    while(1);  
}
```

18.2 Control Registers of ADC

ADCCON0 – ADC Control 0 (Bit-addressable)

7	6	5	4	3	2	1	0
ADCF	ADCS	ETGSEL1	ETGSEL0	ADCHS3	ADCHS2	ADCHS1	ADCHS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Address: E8H

Reset value: 0000 0000b

Bit	Name	Description
7	ADCF	<p>ADC flag This flag is set when an A/D conversion is completed. The ADC result can be read. While this flag is 1, ADC cannot start a new converting. This bit is cleared by software.</p>
6	ADCS	<p>A/D converting software start trigger Setting this bit 1 triggers an A/D conversion. This bit remains logic 1 during A/D converting time and is automatically cleared via hardware right after conversion complete. The meaning of writing and reading ADCS bit is different.</p> <p><u>Writing:</u> 0 = No effect. 1 = Start an A/D converting.</p> <p><u>Reading:</u> 0 = ADC is in idle state. 1 = ADC is busy in converting.</p>
5:4	ETGSEL[1:0]	<p>External trigger source select When ADCEX (ADCCON1.1) is set, these bits select which pin output triggers ADC conversion. 00 = PWM0. 01 = PWM2. 10 = PWM4. 11 = STADC pin.</p>
3:0	ADCHS[3:0]	<p>A/D converting channel select This field selects the activating analog input source of ADC. If ADCEN is 0, all inputs are disconnected. 0000 = AIN0. 0001 = AIN1. 0010 = AIN2. 0011 = AIN3. 0100 = AIN4. 0101 = AIN5. 0110 = AIN6. 0111 = AIN7. 1000 = Internal band-gap voltag. Others = Reserved.</p>

ADCCON1 – ADC Control 1

7	6	5	4	3	2	1	0
-	STADCPX	-	-	ETGTYP[1:0]		ADCEX	ADCEN
-	R/W	-	-	R/W		R/W	R/W

Address: E1H

Reset value: 0000 0000b

Bit	Name	Description
7	-	Reserved
6	STADCPX	External start ADC trigger pin select 0 = Assign STADC to P0.4. 1 = Assign STADC to P1.3. Note that STADC will exchange immediately once setting or clearing this bit.
5:4	-	Reserved
3:2	ETGTYP[1:0]	External trigger type select When ADCEX (ADCCON1.1) is set, these bits select which condition triggers ADC conversion. 00 = Falling edge on PWM0/2/4 or STADC pin. 01 = Rising edge on PWM0/2/4 or STADC pin. 10 = Central point of a PWM period. 11 = End point of a PWM period. Note that the central point interrupt or the period point interrupt is only available for PWM center-aligned type.
1	ADCEX	ADC external conversion trigger select This bit select the methods of triggering an A/D conversion. 0 = A/D conversion is started only via setting ADCS bit. 1 = A/D conversion is started via setting ADCS bit or by external trigger source depending on ETGSEL[1:0] and ETGTYP[1:0]. Note that while ADCS is 1 (busy in converting), the ADC will ignore the following external trigger until ADCS is hardware cleared.
0	ADCEN	ADC enable 0 = ADC circuit off. 1 = ADC circuit on.

ADCCON2 – ADC Control 2

7	6	5	4	3	2	1	0
ADFBEN	ADCMPOP	ADCMPEN	ADCMPO	-	-	-	ADCDLY.8
R/W	R/W	R/W	R	-	-	-	R/W

Address: E2H

Reset value: 0000 0000b

Bit	Name	Description
7	ADFBEN	ADC compare result asserting Fault Brake enable 0 = ADC asserting Fault Brake Disabled. 1 = ADC asserting Fault Brake Enabled. Fault Brake is asserted once its compare result ADCMPO is 1. Meanwhile, PWM channels output Fault Brake data. PWMRUN (PWMCON0.7) will also be automatically cleared by hardware. The PWM output resumes when PWMRUN is set again.
6	ADCMPOP	ADC comparator output polarity 0 = ADCMPO is 1 if ADCR[11:0] is greater than or equal to ADCMP[11:0]. 1 = ADCMPO is 1 if ADCR[11:0] is less than ADCMP[11:0].

Bit	Name	Description
5	ADCMPEN	ADC result comparator enable 0 = ADC result comparator Disabled. 1 = ADC result comparator Enabled.
4	ADCMPO	ADC comparator output value This bit is the output value of ADC result comparator based on the setting of ACMPOP. This bit updates after every A/D conversion complete.
3:1	-	Reserved
0	ADCDLY.8	ADC external trigger delay counter bit 8 See ADCDLY register.

AINDIDS – ADC Channel Digital Input Disconnect

7	6	5	4	3	2	1	0
P11DIDS	P03DIDS	P04DIDS	P05DIDS	P06DIDS	P07DIDS	P30DIDS	P17DIDS
R/W							

Address: F6H

Reset value: 0000 0000b

Bit	Name	Description
n	AINnDIDS	ADC Channel digital input disable 0 = ADC channel n digital input Enabled. 1 = ADC channel n digital input Disabled. ADC channel n is read always 0.

ADCDLY – ADC Trigger Delay Counter

7	6	5	4	3	2	1	0
ADCDLY[7:0]							
R/W							

Address: E3H

Reset value: 0000 0000b

Bit	Name	Description
7:0	ADCDLY[7:0]	ADC external trigger delay counter low byte This 8-bit field combined with ADCCON2.0 forms a 9-bit counter. This counter inserts a delay after detecting the external trigger. An A/D converting starts after this period of delay. External trigger delay time = $\frac{ADCDLY}{F_{ADC}}$ Note that this field is valid only when ADCEX (ADCCON1.1) is set. User should not modify ADCDLY during PWM run time if selecting PWM output as the external ADC trigger source.

ADCRH – ADC Result High Byte

7	6	5	4	3	2	1	0
ADCR[11:4]							
R							

Address: C3H

Reset value: 0000 0000b

Bit	Name	Description
7:0	ADCR[11:4]	ADC result high byte The most significant 8 bits of the ADC result stored in this register.

ADCRL – ADC Result Low Byte

7	6	5	4	3	2	1	0
-	-	-	-	ADCR[3:0]			
-	-	-	-	R			

Address: C2H

Reset value: 0000 0000b

Bit	Name	Description
3:0	ADCR[3:0]	ADC result low byte The least significant 4 bits of the ADC result stored in this register.

ADCMPL – ADC Compare High Byte

7	6	5	4	3	2	1	0
ADCMPL[11:4]							
W/R							

Address: CFH

Reset value: 0000 0000b

Bit	Name	Description
7:0	ADCMPL[11:4]	ADC compare high byte The most significant 8 bits of the ADC compare value stores in this register.

ADCMPL – ADC Compare Low Byte

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

Address: CEH

Reset value: 0000 0000b

Bit	Name	Description
3:0	ADCMPL[3:0]	ADC compare low byte The least significant 4 bits of the ADC compare value stores in this register.

19. TIMED ACCESS PROTECTION (TA)

The N76E003 has several features such as WDT and Brown-out detection that are crucial to proper operation of the system. If leaving these control registers unprotected, errant code may write undetermined value into them and results in incorrect operation and loss of control. To prevent this risk, the N76E003 has a protection scheme, which limits the write access to critical SFRs. This protection scheme is implemented using a timed access (TA). The following registers are related to the TA process.

TA – Timed Access

7	6	5	4	3	2	1	0
TA[7:0]							
W							

Address: C7H

Reset value: 0000 0000b

Bit	Name	Description
7:0	TA[7:0]	<p>Timed access The timed access register controls the access to protected SFRs. To access protected bits, user should first write AAH to the TA and immediately followed by a write of 55H to TA. After these two steps, a writing permission window is opened for 4 clock cycles during this period that user may write to protected SFRs.</p>

In timed access method, the bits, which are protected, have a timed write enable window. A write is successful only if this window is active, otherwise the write will be discarded. When the software writes AAH to TA, a counter is started. This counter waits for 3 clock cycles looking for a write of 55H to TA. If the second write of 55H occurs within 3 clock cycles of the first write of AAH, then the timed access window is opened. It remains open for 4 clock cycles during which user may write to the protected bits. After 4 clock cycles, this window automatically closes. Once the window closes, the procedure should be repeated to write another protected bits. Not that the TA protected SFRs are required timed access for writing but reading is not protected. User may read TA protected SFR without giving AAH and 55H to TA register. The suggestion code for opening the timed access window is shown below.

```

(CLR  EA)                ;if any interrupt is enabled, disable temporarily
(MOV  TA, #0AAH
(MOV  TA, #55H
(Instruction that writes a TA protected register)
(SETB EA)                ;resume interrupts enabled
    
```

Any enabled interrupt should be disabled during this procedure to avoid delay between these three writings. If there is no interrupt enabled, the CLR EA and SETB EA instructions can be left out.

Examples of timed assess are shown to illustrate correct or incorrect writing process.

Example 1,

```
MOV    TA, #0AAH           ;3 clock cycles
MOV    TA, #55H           ;3 clock cycles
ORL    WDCON, #data       ;4 clock cycles
```

Example 2,

```
MOV    TA, #0AAH           ;3 clock cycles
MOV    TA, #55H           ;3 clock cycles
NOP                               ;1 clock cycle
ANL    BODCON0, #data     ;4 clock cycles
```

Example 3,

```
MOV    TA, #0AAH           ;3 clock cycles
MOV    TA, #55H           ;3 clock cycles
MOV    WDCON, #data1      ;3 clock cycles
ORL    BODCON0, #data2   ;4 clock cycles
```

Example 4,

```
MOV    TA, #0AAH           ;3 clock cycles
NOP                               ;1 clock cycle
MOV    TA, #55H           ;3 clock cycles
ANL    BODCON0, #data     ;4 clock cycles
```

In the first example, the writing to the protected bits is done before the 3-clock-cycle window closes. In example 2, however, the writing to BODCON0 does not complete during the window opening, there will be no change of the value of BODCON0. In example 3, the WDCON is successful written but the BODCON0 write is out of the 3-clock-cycle window. Therefore, the BODCON0 value will not change either. In Example 4, the second write 55H to TA completes after 3 clock cycles of the first write TA of AAH, and thus the timed access window is not opened at all, and the write to the protected byte affects nothing.

20. INTERRUPT SYSTEM

20.1 Interrupt Overview

The purpose of the interrupt is to make the software deal with unscheduled or asynchronous events. The N76E003 has a four-priority-level interrupt structure with 18 interrupt sources. Each of the interrupt sources has an individual priority setting bits, interrupt vector and enable bit. In addition, the interrupts can be globally enabled or disabled. When an interrupt occurs, the CPU is expected to service the interrupt. This service is specified as an Interrupt Service Routine (ISR). The ISR resides at a predetermined address as shown in [Table 20-1. Interrupt Vectors](#). When the interrupt occurs if enabled, the CPU will vector to the respective location depending on interrupt source, execute the code at this location, stay in an interrupt service state until the ISR is done. Once an ISR has begun, it can be interrupted only by a higher priority interrupt. The ISR should be terminated by a return from interrupt instruction RETI. This instruction will force the CPU return to the instruction that would have been next when the interrupt occurred.

Table 20-1. Interrupt Vectors

Source	Vector Address	Vector Number	Source	Vector Address	Vector Number
Reset	0000H	-	SPI interrupt	004BH	9
External interrupt 0	0003H	0	WDT interrupt	0053H	10
Timer 0 overflow	000BH	1	ADC interrupt	005BH	11
External interrupt 1	0013H	2	Input capture interrupt	0063H	12
Timer 1 overflow	001BH	3	PWM interrupt	006BH	13
Serial port 0 interrupt	0023H	4	Fault Brake interrupt	0073H	14
Timer 2 event	002BH	5	Serial port 1 interrupt	007BH	15
I ² C status/timer-out interrupt	0033H	6	Timer 3 overflow	0083H	16
Pin interrupt	003BH	7	Self Wake-up Timer interrupt	008BH	17
Brown-out detection interrupt	0043H	8			

20.2 Enabling Interrupts

Each of individual interrupt sources can be enabled or disabled through the use of an associated interrupt enable bit in the IE and EIE SFRs. There is also a global enable bit EA bit (IE.7), which can be cleared to disable all the interrupts at once. It is set to enable all individually enabled interrupts. Setting the EA bit to logic 0 disables all interrupt sources regardless of the individual interrupt-enable settings. Note that interrupts which occur when the EA bit is set to logic 0 will be held in a pending state, and will not be serviced until the EA bit is set back to logic 1. All interrupt flags that generate interrupts can also be set via software. Thereby software initiated interrupts can be generated.

Note that every interrupts, if enabled, is generated by a setting as logic 1 of its interrupt flag no matter by hardware or software. User should take care of each interrupt flag in its own interrupt service routine (ISR). Most of interrupt flags should be cleared by writing it as logic 0 via software to avoid recursive interrupt requests.

IE – Interrupt Enable (Bit-addressable)

7	6	5	4	3	2	1	0
EA	EADC	EBOD	ES	ET1	EX1	ET0	EX0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Address: A8H

Reset value: 0000 0000b

Bit	Name	Description
7	EA	Enable all interrupt This bit globally enables/disables all interrupts that are individually enabled. 0 = All interrupt sources Disabled. 1 = Each interrupt Enabled depending on its individual mask setting. Individual interrupts will occur if enabled.
6	EADC	Enable ADC interrupt 0 = ADC interrupt Disabled. 1 = Interrupt generated by ADCF (ADCCON0.7) Enabled.
5	EBOD	Enable brown-out interrupt 0 = Brown-out detection interrupt Disabled. 1 = Interrupt generated by BOF (BODCON0.3) Enabled.
4	ES	Enable serial port 0 interrupt 0 = Serial port 0 interrupt Disabled. 1 = Interrupt generated by TI (SCON.1) or RI (SCON.0) Enabled.
3	ET1	Enable Timer 1 interrupt 0 = Timer 1 interrupt Disabled. 1 = Interrupt generated by TF1 (TCON.7) Enabled.
2	EX1	Enable external interrupt 1 0 = External interrupt 1 Disabled. 1 = Interrupt generated by <u> </u> pin (P1.7) Enabled.
1	ET0	Enable Timer 0 interrupt 0 = Timer 0 interrupt Disabled. 1 = Interrupt generated by TF0 (TCON.5) Enabled.

Bit	Name	Description
0	EX0	Enable external interrupt 0 0 = External interrupt 0 Disabled. 1 = Interrupt generated by <u> </u> pin (P3.0) Enabled.

EIE – Extensive Interrupt Enable

7	6	5	4	3	2	1	0
ET2	ESPI	EFB	EWDT	EPWM	ECAP	EPI	EI2C
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Address: 9BH

Reset value: 0000 0000b

Bit	Name	Description
7	ET2	Enable Timer 2 interrupt 0 = Timer 2 interrupt Disabled. 1 = Interrupt generated by TF2 (T2CON.7) Enabled.
6	ESPI	Enable SPI interrupt 0 = SPI interrupt Disabled. 1 = Interrupt generated by SPIF (SPSR.7), SPIOVF (SPSR.5), or MODF (SPSR.4) Enable.
5	EFB	Enable Fault Brake interrupt 0 = Fault Brake interrupt Disabled. 1 = Interrupt generated by FBF (FBD.7) Enabled.
4	EWDT	Enable WDT interrupt 0 = WDT interrupt Disabled. 1 = Interrupt generated by WDTF (WDCON.5) Enabled.
3	EPWM	Enable PWM interrupt 0 = PWM interrupt Disabled. 1 = Interrupt generated by PWMF (PWMCON0.5) Enabled.
2	ECAP	Enable input capture interrupt 0 = Input capture interrupt Disabled. 1 = Interrupt generated by any flags of CAPF[2:0] (CAPCON0[2:0]) Enabled.
1	EPI	Enable pin interrupt 0 = Pin interrupt Disabled. 1 = Interrupt generated by any flags in PIF register Enabled.
0	EI2C	Enable I²C interrupt 0 = I ² C interrupt Disabled. 1 = Interrupt generated by SI (I2CON.3) or I2TOF (I2TOC.0) Enabled.

EIE1 – Extensive Interrupt Enable 1

7	6	5	4	3	2	1	0
-	-	-	-	-	EWKT	ET3	ES_1
-	-	-	-	-	R/W	R/W	R/W

Address: 9CH

Reset value: 0000 0000b

Bit	Name	Description
2	EWKT	Enable WKT interrupt 0 = WKT interrupt Disabled. 1 = Interrupt generated by WKTF (WKCON.4) Enabled.

Bit	Name	Description
1	ET3	Enable Timer 3 interrupt 0 = Timer 3 interrupt Disabled. 1 = Interrupt generated by TF3 (T3CON.4) Enabled.
0	ES_1	Enable serial port 1 interrupt 0 = Serial port 1 interrupt Disabled. 1 = Interrupt generated by TI_1 (SCON_1.1) or RI_1 (SCON_1.0) Enabled.

20.3 Interrupt Priorities

There are four priority levels for all interrupts. They are level highest, high, low, and lowest; and they are represented by level 3, level 2, level 1, and level 0. The interrupt sources can be individually set to one of four priority levels by setting their own priority bits. Table 20-2. Interrupt Priority Level Setting lists four priority setting. Naturally, a low level priority interrupt can itself be interrupted by a high level priority interrupt, but not by any same level interrupt or lower level. In addition, there exists a pre-defined natural priority among the interrupts themselves. The natural priority comes into play when the interrupt controller has to resolve simultaneous requests having the same priority level.

In case of multiple interrupts, the following rules apply:

1. While a low priority interrupt handler is running, if a high priority interrupt arrives, the handler will be interrupted and the high priority handler will run. When the high priority handler does “ E ”, the low priority handler will resume. When this handler does “ E ”, control is passed back to the main program.
2. If a high priority interrupt is running, it cannot be interrupted by any other source – even if it is a high priority interrupt which is higher in natural priority.
3. A low-priority interrupt handler will be invoked only if no other interrupt is already executing. Again, the low priority interrupt cannot preempt another low priority interrupt, even if the later one is higher in natural priority.
4. If two interrupts occur at the same time, the interrupt with higher priority will execute first. If both interrupts are of the same priority, the interrupt which is higher in natural priority will be executed first. This is the only context in which the natural priority matters.

This natural priority is defined as shown on Table 20-3. Characteristics of Each Interrupt Source. It also summarizes the interrupt sources, flag bits, vector addresses, enable bits, priority bits, natural priority and the permission to wake up the CPU from Power-down mode. For details of waking CPU up from Power-down mode, please see [Section 22.1 “Power-Down Mode” on page 226](#).

Table 20-2. Interrupt Priority Level Setting

Interrupt Priority Control Bits		Interrupt Priority Level
IPH / EIPH / EIPH1	IP / EIP / EIP2	
0	0	Level 0 (lowest)
0	1	Level 1
1	0	Level 2
1	1	Level 3 (highest)

Table 20-3. Characteristics of Each Interrupt Source

Interrupt Source	Vector Address	Interrupt Flag(s)	Enable Bit	Natural Priority	Priority Control Bits	Power-down Wake-up
Reset	0000H	-	Always Enabled	Highest	-	Yes
External interrupt 0	0003H	IE0 ^[1]	EX0	1	PX0, PX0H	Yes
Brown-out	0043H	BOF (BODCON0.3)	EBOD	2	PBOD, PBODH	Yes
Watchdog Timer	0053H	WDTF (WDCON.5)	EWDT	3	PWDT, PWDTH	Yes
Timer 0	000BH	TF0 ^[2]	ET0	4	PT0, PT0H	No
I ² C status/time-out	0033h	SI + I2TOF (I2TOC.0)	EI2C	5	PI2C, PI2CH	No
ADC	005Bh	ADCF	EADC	6	PADC, PADCH	No
External interrupt 1	0013H	IE1 ^[1]	EX1	7	PX1, PX1H	Yes
Pin interrupt	003BH	PIF0 to PIF7 (PIF) ^[3]	EPI	8	PPI, PPIH	Yes
Timer 1	001BH	TF1 ^[2]	ET1	9	PT1, PT1H	No
Serial port 0	0023H	RI + TI	ES	10	PS, PSH	No
Fault Brake event	0073h	FBF (FBD.7)	EFB	11	PFB, PFBH	No
SPI	004Bh	SPIF (SPSR.7) + MODF (SPSR.4) + SPIOVF (SPSR.5)	ESPI	12	PSPI, PSPIH	No
Timer 2	002BH	TF2 ^[2]	ET2	13	PT2, PT2H	No
Input capture	0063H	CAPF[2:0] (CAPCON0[2:0])	ECAP	14	PCAP, PCAPH	No
PWM interrupt	006BH	PWMF	EPWM	15	PPWM, PPWMH	No
Serial port 1	007BH	RI_1 + TI_1	ES_1	16	PS_1, PSH_1	No
Timer 3	0083H	TF3 ^[2] (T3CON.4)	ET3	17	PT3, PT3H	No
Self Wake-up Timer	008BH	WKTF (WKCON.4)	EWKT	18	PWKT, PWKTH	Yes

[1] While the external interrupt pin is set as edge triggered (I_{tx} = 1), its own flag I_{ex} will be automatically cleared if the interrupt service routine (ISR) is executed. While as level triggered (I_{tx} = 0), I_{ex} follows the inverse of respective pin state. It is not controlled via software.

[2] TF0, TF1, or TF3 is automatically cleared if the interrupt service routine (ISR) is executed. On the contrary, be aware that TF2 is not.

[3] If level triggered is selected for pin interrupt channel n, PIF_n flag reflects the respective channel state. It is not controlled via software.

IP – Interrupt Priority (Bit-addressable)^[1]

7	6	5	4	3	2	1	0
-	PADC	PBOD	PS	PT1	PX1	PT0	PX0
-	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Address: B8H

Reset value: 0000 0000b

Bit	Name	Description
6	PADC	ADC interrupt priority low bit
5	PBOD	Brown-out detection interrupt priority low bit
4	PS	Serial port 0 interrupt priority low bit
3	PT1	Timer 1 interrupt priority low bit
2	PX1	External interrupt 1 priority low bit
1	PT0	Timer 0 interrupt priority low bit
0	PX0	External interrupt 0 priority low bit

[1] IP is used in combination with the IPH to determine the priority of each interrupt source. See [Table 20-2. Interrupt Priority Level Setting](#) for correct interrupt priority configuration.

IPH – Interrupt Priority High^[2]

7	6	5	4	3	2	1	0
-	PADCH	PBODH	PSH	PT1H	PX1H	PT0H	PX0H
-	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Address: B7H, Page0

Reset value: 0000 0000b

Bit	Name	Description
6	PADC	ADC interrupt priority high bit
5	PBOD	Brown-out detection interrupt priority high bit
4	PSH	Serial port 0 interrupt priority high bit
3	PT1H	Timer 1 interrupt priority high bit
2	PX1H	External interrupt 1 priority high bit
1	PT0H	Timer 0 interrupt priority high bit
0	PX0H	External interrupt 0 priority high bit

[2] IPH is used in combination with the IP respectively to determine the priority of each interrupt source. See [Table 20-2. Interrupt Priority Level Setting](#) for correct interrupt priority configuration.

EIP – Extensive Interrupt Priority^[3]

7	6	5	4	3	2	1	0
PT2	PSPI	PFB	PWDT	PPWM	PCAP	PPI	PI2C
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Address: EFH

Reset value: 0000 0000b

Bit	Name	Description
7	PT2	Timer 2 interrupt priority low bit
6	PSPI	SPI interrupt priority low bit
5	PFB	Fault Brake interrupt priority low bit

Bit	Name	Description
4	PWDT	WDT interrupt priority low bit
3	PPWM	PWM interrupt priority low bit
2	PCAP	Input capture interrupt priority low bit
1	PPI	Pin interrupt priority low bit
0	PI2C	I ² C interrupt priority low bit

[3] EIP is used in combination with the EIPH to determine the priority of each interrupt source. See [Table 20-2. Interrupt Priority Level Setting](#) for correct interrupt priority configuration.

EIPH – Extensive Interrupt Priority High^[4]

7	6	5	4	3	2	1	0
PT2H	PSPIH	PFBH	PWDTH	PPWMH	PCAPH	PPIH	PI2CH
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Address: F7H

Reset value: 0000 0000b

Bit	Name	Description
7	PT2H	Timer 2 interrupt priority high bit
6	PSPIH	SPI interrupt priority high bit
5	PFBH	Fault Brake interrupt priority high bit
4	PWDTH	WDT interrupt priority high bit
3	PPWMH	PWM interrupt priority high bit
2	PCAPH	Input capture interrupt priority high bit
1	PPIH	Pin interrupt priority high bit
0	PI2CH	I ² C interrupt priority high bit

[4] EIPH is used in combination with the EIP to determine the priority of each interrupt source. See [Table 20-2. Interrupt Priority Level Setting](#) for correct interrupt priority configuration.

EIP1 – Extensive Interrupt Priority 1^[5]

7	6	5	4	3	2	1	0
-	-	-	-	-	PWKT	PT3	PS_1
-	-	-	-	-	R/W	R/W	R/W

Address: FEH, Page: 0

Reset value: 0000 0000b

Bit	Name	Description
2	PWKT	WKT interrupt priority low bit
1	PT3	Timer 3 interrupt priority low bit
0	PS_1	Serial port 1 interrupt priority low bit

[5] EIP1 is used in combination with the EIPH1 to determine the priority of each interrupt source. See [Table 20-2. Interrupt Priority Level Setting](#) for correct interrupt priority configuration.

EIPH1 – Extensive Interrupt Priority High 1^[6]

7	6	5	4	3	2	1	0
-	-	-	-	-	PWKTH	PT3H	PSH_1
-	-	-	-	-	R/W	R/W	R/W

Address: FFH, Page: 0

Reset value: 0000 0000b

Bit	Name	Description
2	PWKTH	WKT interrupt priority high bit
1	PT3H	Timer 3 interrupt priority high bit
0	PSH_1	Serial port 1 interrupt priority high bit

[6] EIPH1 is used in combination with the EIP1 to determine the priority of each interrupt source. See [Table 20-2. Interrupt Priority Level Setting](#) for correct interrupt priority configuration.

20.4 Interrupt Service

The interrupt flags are sampled every system clock cycle. In the same cycle, the sampled interrupts are polled and their priority is resolved. If certain conditions are met then the hardware will execute an internally generated LCALL instruction, which will vector the process to the appropriate interrupt vector address. The conditions for generating the LCALL are,

1. An interrupt of equal or higher priority is not currently being serviced.
2. The current polling cycle is the last cycle of the instruction currently being executed.
3. The current instruction does not involve a write to any enabling or priority setting bits and is not a RETI.

If any of these conditions are not met, then the LCALL will not be generated. The polling cycle is repeated every system clock cycle. If an interrupt flag is active in one cycle but not responded to for the above conditions are not met, if the flag is not still active when the blocking condition is removed, the denied interrupt will not be serviced. This means that the interrupt flag, which was once active but not serviced is not remembered. Every polling cycle is new.

The processor responds to a valid interrupt by executing an LCALL instruction to the appropriate service routine. This action may or may not clear the flag, which caused the interrupt according to different interrupt source. The hardware LCALL behaves exactly like the software LCALL instruction. This instruction saves the Program Counter contents onto the Stack RAM but does not save the Program Status Word (PSW). The PC is reloaded with the vector address of that interrupt, which caused the LCALL. Execution continues from the vectored address until an RETI instruction is executed. On execution of the RETI instruction, the processor pops the Stack and loads the PC with the contents at the top of the stack. User should take care that the status of the stack. The processor does not notice anything if the stack contents are modified and will proceed with execution from the

address put back into PC. Note that a simple RET instruction would perform exactly the same process as a RETI instruction, but it would not inform the Interrupt controller that the interrupt service routine is completed. RET would leave the controller still thinking that the service routine is underway, making future interrupts impossible.

20.5 Interrupt Latency

The response time for each interrupt source depends on several factors, such as the nature of the interrupt and the instruction underway. Each interrupt flags are polled and priority decoded each system clock cycle. If a request is active and all three previous conditions are met, then the hardware generated LCALL is executed. This LCALL itself takes 4 clock cycles to be completed. Thus, there is a 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.

20.6 External Interrupt Pins

The external interrupt $\overline{INT0}$ and $\overline{INT1}$ 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 $\overline{INT0}$ or $\overline{INT1}$ 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 $\overline{\text{IE0}}$ and $\overline{\text{IE1}}$ 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 $\overline{\text{IE0}}$ and $\overline{\text{IE1}}$ can wake up the device from the Power-down mode.

TCON – Timer 0 and 1 Control (Bit-addressable)

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
R/W	R/W	R/W	R/W	R (level) R/W (edge)	R/W	R (level) R/W (edge)	R/W

Address: 88H

Reset value: 0000 0000b

Bit	Name	Description
3	IE1	External interrupt 1 edge flag If IT1 = 1 (falling edge trigger), this flag will be set by hardware when a falling edge is detected. It remain set until cleared via software or cleared by hardware in the beginning of its interrupt service routine. If IT1 = 0 (low level trigger), this flag follows the inverse of the $\overline{\text{IE1}}$ input signal's logic level. Software cannot control it.
2	IT1	External interrupt 1 type select This bit selects by which type that $\overline{\text{IE1}}$ is triggered. 0 = $\overline{\text{IE1}}$ is low level triggered. 1 = $\overline{\text{IE1}}$ is falling edge triggered.
1	IE0	External interrupt 0 edge flag If IT0 = 1 (falling edge trigger), this flag will be set by hardware when a falling edge is detected. It remain set until cleared via software or cleared by hardware in the beginning of its interrupt service routine. If IT0 = 0 (low level trigger), this flag follows the inverse of the $\overline{\text{IE0}}$ input signal's logic level. Software cannot control it.
0	IT0	External interrupt 0 type select This bit selects by which type that $\overline{\text{IE0}}$ is triggered. 0 = $\overline{\text{IE0}}$ is low level triggered. 1 = $\overline{\text{IE0}}$ is falling edge triggered.

21. IN-APPLICATION-PROGRAMMING (IAP)

Unlike A's real-time operation, to update flash data often takes long time. Furthermore, it is a quite complex timing procedure to erase, program, or read flash data. The N76E003 carried out the flash operation with convenient mechanism to help user re-programming the flash content by In-Application-Programming (IAP). IAP is an in-circuit electrical erasure and programming method through software.

After IAP enabling by setting IAPEN (CHPCON.0 with TA protected) and setting the enable bit in IAPUEN that allows the target block to be updated, user can easily fill the 16-bit target address in IAPAH and IAPAL, data in IAPFD, and command in IAPCN. Then the IAP is ready to begin by setting a triggering bit IAPGO (IAPTRG.0). Note that IAPTRG is also TA protected. At this moment, the CPU holds the Program Counter and the built-in IAP automation takes over to control the internal charge-pump for high voltage and the detail signal timing. The erase and program time is internally controlled disregard of the operating voltage and frequency. Nominally, a page-erase time is 5 ms and a byte-program time is 23.5 μs. After IAP action completed, the Program Counter continues to run the following instructions. The IAPGO bit will be automatically cleared. An IAP failure flag, IAPFF (CHPCON.6), can be check whether the previous IAP operation was successful or not. Through this progress, user can easily erase, program, and verify the Flash Memory by just taking care of pure software.

The following registers are related to IAP processing.

CONFIG2

7	6	5	4	3	2	1	0
CBODEN	-	CBOV[1:0]		BOIAP	CBORST	-	-
R/W	-	R/W		R/W	R/W	-	-

Factory default value: 1111 1111b

Bit	Name	Description
3	BOIAP	Brown-out inhibiting IAP This bit decide whether IAP erasing or programming is inhibited by brown-out status. This bit is valid only when brown-out detection is enabled. 1 = IAP erasing or programming is inhibited if V _{DD} is lower than V _{BOD} . 0 = IAP erasing or programming is allowed under any workable V _{DD} .

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
6	IAPFF	IAP fault flag

Bit	Name	Description
		The hardware will set this bit after IAPGO (ISPTRG.0) is set if any of the following condition is met: (1) The accessing address is oversize. (2) IAPCN command is invalid. (3) IAP erases or programs updating un-enabled block. (4) IAP erasing or programming operates under V _{BOD} while BOIAP (CONFIG2.5) remains un-programmed 1 with BODEN (BODCON0.7) as 1 and BORST (BODCON0.2) as 0. This bit should be cleared via software.
0	IAPEN	IAP enable 0 = IAP function Disabled. 1 = IAP function Enabled. Once enabling IAP function, the HIRC will be turned on for timing control. To clear IAPEN should always be the last instruction after IAP operation to stop internal oscillator if reducing power consumption is concerned.

IAPUEN – IAP Updating Enable (TA protected)

7	6	5	4	3	2	1	0
-	-	-	-	-	CFUEN	LDUEN	APUEN
-	-	-	-	-	R/W	R/W	R/W

Address: A5H

Reset value: 0000 0000b

Bit	Name	Description
2	CFUEN	CONFIG bytes updated enable 0 = Inhibit erasing or programming CONFIG bytes by IAP. 1 = Allow erasing or programming CONFIG bytes by IAP.
1	LDUEN	LDROM updated enable 0 = Inhibit erasing or programming LDROM by IAP. 1 = Allow erasing or programming LDROM by IAP.
0	APUEN	APROM updated enable 0 = Inhibit erasing or programming APROM by IAP. 1 = Allow erasing or programming APROM by IAP.

IAPCN – IAP Control

7	6	5	4	3	2	1	0
IAPB[1:0]		FOEN	FCEN	FCTRL[3:0]			
R/W		R/W	R/W	R/W			

Address: AFH

Reset value: 0011 0000b

Bit	Name	Description
7:6	IAPB[1:0]	IAP control This byte is used for IAP command. For details, see Table 21-1. IAP Modes and Command Codes .
5	FOEN	
4	FCEN	
3:0	FCTRL[3:0]	

IAPAH – IAP Address High Byte

7	6	5	4	3	2	1	0
IAPA[15:8]							
R/W							

Address: A7H

Reset value: 0000 0000b

Bit	Name	Description
7:0	IAPA[15:8]	IAP address high byte IAPAH contains address IAPA[15:8] for IAP operations.

IAPAL – IAP Address Low Byte

7	6	5	4	3	2	1	0
IAPA[7:0]							
R/W							

Address: A6H

Reset value: 0000 0000b

Bit	Name	Description
7:0	IAPA[7:0]	IAP address low byte IAPAL contains address IAPA[7:0] for IAP operations.

IAPFD – IAP Flash Data

7	6	5	4	3	2	1	0
IAPFD[7:0]							
R/W							

Address: AEH

Reset value: 0000 0000b

Bit	Name	Description
7:0	IAPFD[7:0]	IAP flash data This byte contains flash data, which is read from or is going to be written to the Flash Memory. User should write data into IAPFD for program mode before triggering IAP processing and read data from IAPFD for read/verify mode after IAP processing is finished.

IAPTRG – IAP Trigger (TA protected)

7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	IAPGO
-	-	-	-	-	-	-	W

Address: A4H

Reset value: 0000 0000b

Bit	Name	Description
0	IAPGO	<p>IAP go IAP begins by setting this bit as logic 1. After this instruction, the CPU holds the Program Counter (PC) and the IAP hardware automation takes over to control the progress. After IAP action completed, the Program Counter continues to run the following instruction. The IAPGO bit will be automatically cleared and always read as logic 0.</p> <p>Before triggering an IAP action, interrupts (if enabled) should be temporary disabled for hardware limitation. The program process should follows below.</p> <pre> CLR EA MOV TA, #0AAH MOV TA, #55H ORL IAPTRG, #01H (SETB EA) </pre>

21.1 IAP Commands

The N76E003 provides a wide range of applications to perform IAP to APROM, LDROM, or CONFIG bytes. The IAP action mode and the destination of the flash block are defined by IAP control register IAPCN.

Table 21-1. IAP Modes and Command Codes

IAP Mode	IAPCN				IAPA[15:0] {IAPAH, IAPAL}	IAPFD[7:0]
	IAPB[1:0]	FOEN	FCEN	FCTRL[3:0]		
Company ID read	XX ^[1]	0	0	1011	X	DAH
Device ID read	XX	0	0	1100	Low-byte DID: 0000H High-byte DID: 0001H	Low-byte DID: 50H High-byte DID: 36H
96-bit Unique Code read	XX	0	0	0100	0000H to 000BH	Data out
APROM page-erase	00	1	0	0010	Address in ^[2]	FFH
LDROM page-erase	01	1	0	0010	Address in ^[2]	FFH
APROM byte-program	00	1	0	0001	Address in	Data in
LDROM byte-program	01	1	0	0001	Address in	Data in
APROM byte-read	00	0	0	0000	Address in	Data out
LDROM byte-read	01	0	0	0000	Address in	Data out
All CONFIG bytes erase	11	1	0	0010	0000H	FFH

IAP Mode	IAPCN				IAPA[15:0] {IAPAH, IAPAL}	IAPFD[7:0]
	IAPB[1:0]	FOEN	FCEN	FCTRL[3:0]		
CONFIG byte-program	11	1	0	0001	CONFIG0: 0000H CONFIG1: 0001H CONFIG2: 0002H CONFIG4: 0004H	Data in
CONFIG byte-read	11	0	0	0000	CONFIG0: 0000H CONFIG1: 0001H CONFIG2: 0002H CONFIG4: 0004H	Data out

[1] "X" means "don't care".

[2] Each page is 128 Bytes size. Therefore, the address should be the address pointed to the target page.

21.2 IAP User Guide

IAP facilitates the updating flash contents in a convenient way; however, user should follow some restricted laws in order that the IAP operates correctly. Without noticing warnings will possible cause undetermined results even serious damages of devices. Furthermore, this paragraph will also support useful suggestions during IAP procedures.

(1) If no more IAP operation is needed, user should clear IAPEN (CHPCON.0). It will make the system void to trigger IAP unaware. Furthermore, IAP requires the HIRC running. If the external clock source is selected, disabling IAP will stop the HIRC for saving power consumption. Note that a write to IAPEN is TA protected.

(2) When the LOCK bit (CONFIG0.1) is activated, IAP reading, writing, or erasing can still be valid.

During IAP progress, interrupts (if enabled) should be disabled temporarily by clearing EA bit for implement limitation.

Do not attempt to erase or program to a page that the code is currently executing. This will cause unpredictable program behavior and may corrupt program data.

21.3 Using Flash Memory as Data Storage

In general application, there is a need of data storage, which is non-volatile so that it remains its content even after the power is off. Therefore, in general application user can read back or update the data, which rules as parameters or constants for system control. The Flash Memory array of the N76E003 supports IAP function and any byte in the Flash Memory array may be read using the MOVC instruction and thus is suitable for use as non-volatile data storage. IAP provides erase and program function that makes it easy for one or more bytes within a page to be erased and programmed in a routine. IAP performs in the application under the control of the microcontroller's firmware. Be aware of Flash Memory writing endurance of 100,000 cycles. A demo is illustrated as follows.

Assembly demo code:

```

;*****
;   This code illustrates how to use IAP to make APROM 201h as a byte of
;   Data Flash when user code is executed in APROM.
;*****
PAGE_ERASE_AP          EQU          00100010b
BYTE_PROGRAM_AP       EQU          00100001b

    ORG    0000h

    MOV    TA,#0Aah          ;CHPCON is TA protected
    MOV    TA,#55h
    ORL    CHPCON,#00000001b ;IAPEN = 1, enable IAP mode

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

    MOV    IAPCN,#PAGE_ERASE_AP ;Erase page 200h~27Fh
    MOV    IAPAH,#02h
    MOV    IAPAL,#00h
    MOV    IAPFD,#0FFh
    MOV    TA,#0Aah          ;IAPTRG is TA protected
    MOV    TA,#55h
    ORL    IAPTRG,#00000001b ;write '1' to IAPGO to trigger IAP process
    MOV    IAPCN,#BYTE_PROGRAM_AP ;Program 201h with 55h
    MOV    IAPAH,#02h
    MOV    IAPAL,#01h
    MOV    IAPFD,#55h
    MOV    TA,#0Aah
    MOV    TA,#55h
    ORL    IAPTRG,#00000001b

    MOV    TA,#0Aah
    MOV    TA,#55h
    ANL    IAPUEN,#11111110b ;APUEN = 0, disable APROM update

    MOV    TA,#0Aah
    MOV    TA,#55h
    ANL    CHPCON,#11111110b ;IAPEN = 0, disable IAP mode

    MOV    DPTR,#201h
    CLR    A
    MOVC  A,@A+DPTR          ;Read content of address 201h
    MOV    P0,A

    SJMP  $

```

C language demo code:

```

//*****
//      This code illustrates how to use IAP to make APROM 201h as a byte of
//      Data Flash when user code is executed in APROM.
//*****
#define      PAGE_ERASE_AP      0x22
#define      BYTE_PROGRAM_AP    0x21

/*Data Flash, as part of APROM, is read by MOVX. Data Flash can be defined as
  128-element array in "code" area from absolute address 0x0200      */

volatile unsigned char code Data_Flash[128] _at_ 0x0200;

Main (void)
{
    TA = 0xAA;                //CHPCON is TA protected
    TA = 0x55;
    CHPCON |= 0x01;          //IAPEN = 1, enable IAP mode

    TA = 0xAA;                //IAPUEN is TA protected
    TA = 0x55;
    IAPUEN |= 0x01;         //APUEN = 1, enable APROM update

    IAPCN = PAGE_ERASE_AP;    //Erase page 200h~27Fh
    IAPAH = 0x02;
    IAPAL = 0x00;
    IAPFD = 0xFF;
    TA = 0xAA;                //IAPTRG is TA protected
    TA = 0x55;
    IAPTRG |= 0x01;         //write '1' to IAPGO to trigger IAP process

    IAPCN = BYTE_PROGRAM_AP; // Program 201h with 55h
    IAPAH = 0x02;
    IAPAL = 0x01;
    IAPFD = 0x55;
    TA = 0xAA;
    TA = 0x55;
    IAPTRG |= 0x01;         //write '1' to IAPGO to trigger IAP process

    TA = 0xAA;                //IAPUEN is TA protected
    TA = 0x55;
    IAPUEN &= ~0x01;        //APUEN = 0, disable APROM update

    TA = 0xAA;                //CHPCON is TA protected
    TA = 0x55;
    CHPCON &= ~0x01;        //IAPEN = 0, disable IAP mode

    P0 = Data_Flash[1];      //Read content of address 200h+1

    while(1);
}

```

21.4 In-System-Programming (ISP)

The Flash Memory supports both hardware programming and In-Application-Programming (IAP). If the product is just under development or the end product needs firmware updating in the hand of an end user, the hardware programming mode will make repeated programming difficult and inconvenient. In-

System-Programming (ISP) makes it easy and possible. ISP performs Flash Memory updating without removing the microcontroller from the system. It allows a device to be re-programmed under software control. Furthermore, the capability to update the application firmware makes wide range of applications possible.

User can develop a custom Boot Code that resides in LDROM. The maximum size of LDROM is 4K Bytes. User developed Boot Code can be re-programmed by parallel writer or In-Circuit-Programming (ICP) tool.

General speaking, an ISP is carried out by a communication between PC and MCU. PC transfers the new User Code to MCU through serial port. Then Boot Code receives it and re-programs into User Code through IAP commands. Nuvoton provides ISP firmware and PC application for N76E003. It makes user quite easy perform ISP through UART port. Please visit Nuvoton 8-bit Microcontroller website: [Nuvoton 80C51 Microcontroller Technical Support](http://www.nuvoton.com.tw/Support/80C51_Microcontroller_Technical_Support). A simple ISP demo code is given below.

Assembly demo code:

```

;*****
; This code illustrates how to do APROM and CONFIG IAP from LDROM.
; APROM are re-programmed by the code to output P1 as 55h and P0 as aah.
; The CONFIG2 is also updated to disable BOD reset.
; User needs to configure CONFIG0 = 0x7F, CONFIG1 = 0xFE, CONFIG2 = 0xFF.
;*****
PAGE_ERASE_AP EQU 00100010b
BYTE_PROGRAM_AP EQU 00100001b
BYTE_READ_AP EQU 00000000b
ALL_ERASE_CONFIG EQU 11100010b
BYTE_PROGRAM_CONFIG EQU 11100001b
BYTE_READ_CONFIG EQU 11000000b

ORG 0000h

CLR EA ;disable all interrupts
CALL Enable_IAP

CALL Enable_AP_Update
CALL Erase_AP ;erase AP data
CALL Program_AP ;programming AP data
CALL Disable_AP_Update
CALL Program_AP_Verify ;verify Programmed AP data

CALL Read_CONFIG ;read back CONFIG2
CALL Enable_CONFIG_Update
CALL Erase_CONFIG ;erase CONFIG bytes
CALL Program_CONFIG ;programming CONFIG2 with new data
CALL Disable_CONFIG_Update
CALL Program_CONFIG_Verify ;verify Programmed CONFIG2

CALL Disable_IAP
MOV TA,#0Aah ;TA protection
MOV TA,#55h ;
ANL CHPCON,#11111101b ;BS = 0, reset to APROM
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

```

```

CALL    Trigger_IAP
INC     R0
CJNE   R0,#44h,Erase_AP_Loop
RET

Program_AP:
MOV     IAPCN,#BYTE_PROGRAM_AP
MOV     IAPAH,#00h
MOV     IAPAL,#00h
MOV     DPTR,#AP_code
Program_AP_Loop:
CLR     A
MOVC   A,@A+DPTR
MOV     IAPFD,A
CALL    Trigger_IAP
INC     DPTR
INC     IAPAL
MOV     A,IAPAL
CJNE   A,#14,Program_AP_Loop
RET

Program_AP_Verify:
MOV     IAPCN,#BYTE_READ_AP
MOV     IAPAH,#00h
MOV     IAPAL,#00h
MOV     DPTR,#AP_code
Program_AP_Verify_Loop:
CALL    Trigger_IAP
CLR     A
MOVC   A,@A+DPTR
MOV     B,A
MOV     A,IAPFD
CJNE   A,B,Program_AP_Verify_Error
INC     DPTR
INC     IAPAL
MOV     A,IAPAL
CJNE   A,#14,Program_AP_Verify_Loop
RET

Program_AP_Verify_Error:
CALL    Disable_IAP
MOV     P0,#00h
SJMP   $

;*****
;
;       IAP CONFIG Function
;*****
Erase_CONFIG:
MOV     IAPCN,#ALL_ERASE_CONFIG
MOV     IAPAH,#00h
MOV     IAPAL,#00h
MOV     IAPFD,#0FFh
CALL    Trigger_IAP
RET

Read_CONFIG:
MOV     IAPCN,#BYTE_READ_CONFIG
MOV     IAPAH,#00h
MOV     IAPAL,#02h
CALL    Trigger_IAP
MOV     R7,IAPFD
RET

```

Program_CONFIG:

```

MOV    IAPCN,#BYTE_PROGRAM_CONFIG
MOV    IAPAH,#00h
MOV    IAPAL,#02h
MOV    A,R7
ANL    A,#11111011b
MOV    IAPFD,A           ;disable BOD reset
MOV    R6,A             ;temp data
CALL   Trigger_IAP
RET

```

Program_CONFIG_Verify:

```

MOV    IAPCN,#BYTE_READ_CONFIG
MOV    IAPAH,#00h
MOV    IAPAL,#02h
CALL   Trigger_IAP
MOV    B,R6
MOV    A,IAPFD
CJNE  A,B,Program_CONFIG_Verify_Error
RET

```

Program_CONFIG_Verify_Error:

```

CALL   Disable_IAP
MOV    P0,#00h
SJMP  $

```

```

;*****
;          APROM code
;*****
AP_code:
DB     75h,0B1h, 00h           ;OPCODEs of "MOV    P0M1,#0"
DB     75h,0B3h, 00h           ;OPCODEs of "MOV    P1M1,#0"
DB     75h, 90h, 55h           ;OPCODEs of "MOV    P1,#55h"
DB     75h,080h,0Aah           ;OPCODEs of "MOV    P0,#0Aah"
DB     80h,0Feh                ;OPCODEs of "SJMP  $"

END

```

22. POWER MANAGEMENT

The N76E003 has several features that help user to control the power consumption of the device. The power reduced feature has two option modes: Idle mode and Power-down mode, to save the power consumption. For a stable current consumption, the state and mode of each pin should be taken care of. The minimum power consumption can be attained by giving the pin state just the same as the external pulls for example output 1 if pull-high is used or output 0 if pull-low. If the I/O pin is floating, user is recommended to leave it as quasi-bidirectional mode. If P2.0 is configured as a input-only pin, it should have an external pull-up or pull-low, or enable its internal pull-up by setting P20UP (P2S.7).

PCON – Power Control

7	6	5	4	3	2	1	0
SMOD	SMOD0	-	POF	GF1	GF0	PD	IDL
R/W	R/W	-	R/W	R/W	R/W	R/W	R/W

Address: 87H

Reset value: see [Table 6-2. SFR Definitions and Reset Values](#)

Bit	Name	Description
1	PD	<p>Power-down mode</p> <p>Setting this bit puts CPU into Power-down mode. Under this mode, both CPU and peripheral clocks stop and Program Counter (PC) suspends. It provides the lowest power consumption. After CPU is woken up from Power-down, this bit will be automatically cleared via hardware and the program continue executing the interrupt service routine (ISR) of the very interrupt source that woke the system up before. After return from the ISR, the device continues execution at the instruction, which follows the instruction that put the system into Power-down mode. Note that If IDL bit and PD bit are set simultaneously, CPU will enter Power-down mode. Then it does not go to Idle mode after exiting Power-down.</p>
0	IDL	<p>Idle mode</p> <p>Setting this bit puts CPU into Idle mode. Under this mode, the CPU clock stops and Program Counter (PC) suspends but all peripherals keep activated. After CPU is woken up from Idle, this bit will be automatically cleared via hardware and the program continue executing the ISR of the very interrupt source that woke the system up before. After return from the ISR, the device continues execution at the instruction which follows the instruction that put the system into Idle mode.</p>

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.

22.1 Power-Down Mode

Power-down mode is the lowest power state that the N76E003 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 N76E003 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 N76E003 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 stable, the device executes the interrupt service routine (ISR) for the corresponding external interrupt. After the ISR is completed, the program execution returns to the instruction after the one, which puts the device into Power-down mode and continues. Interrupts that allows to wake up CPU from Power-down mode includes external interrupt $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$, pin interrupt, WDT interrupt, WKT interrupt, and brown-out interrupt.

23. CLOCK SYSTEM

The N76E003 has a wide variety of clock sources and selection features that allow it to be used in a wide range of applications while maximizing performance and minimizing power consumption. The N76E003 provides three options of the system clock sources including internal oscillator, or external clock from X_{IN} pin via software. The N76E003 is embedded with two internal oscillators: one 10 kHz low-speed and one 16 MHz high-speed, which is factory trimmed to $\pm 2\%$ under all conditions. A clock divider CKDIV is also available on N76E003 for adjustment of the flexibility between power consumption and operating performance.

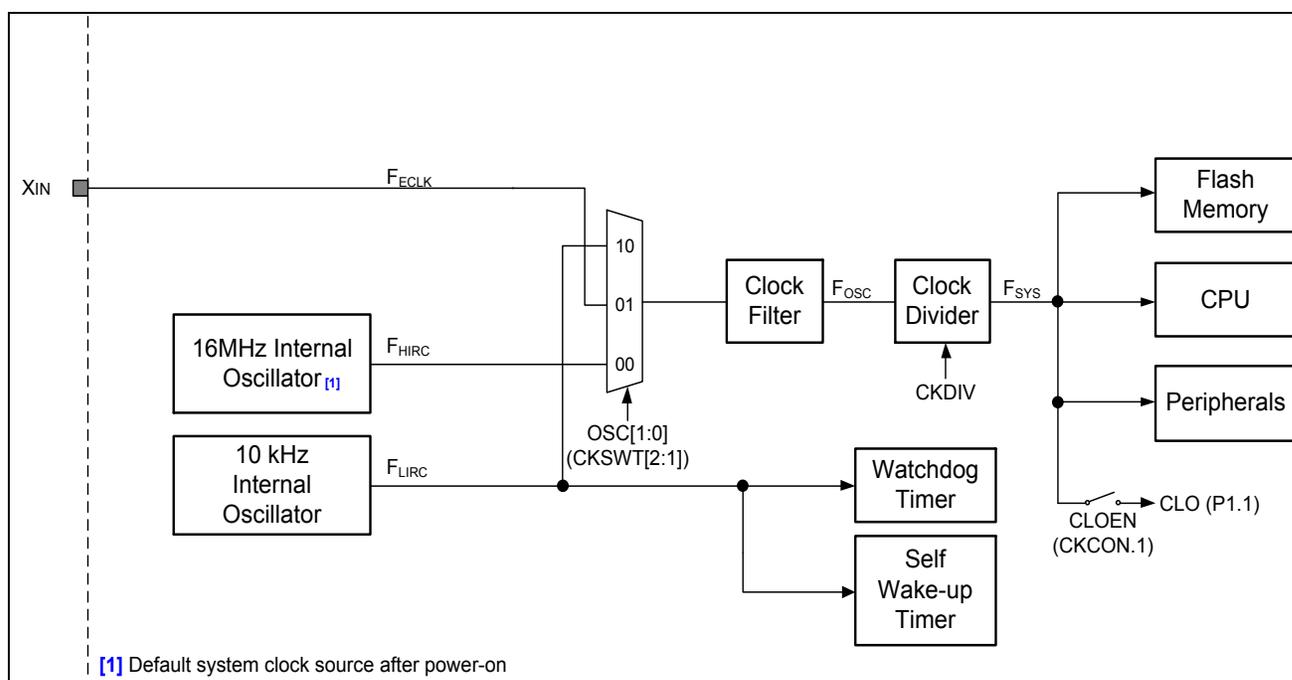


Figure 23-1 Clock System Block Diagram

23.1 System Clock Sources

There are a total of three system clock sources selectable in the N76E003 including high-speed internal oscillator, low-speed internal oscillator and external clock input. Each of them can be the system clock source in the N76E003. Different active system clock sources also affect multi-function of P3.0/XIN.

23.1.1 Internal Oscillators

There are two internal oscillators in the N76E003 – one 16 MHz high-speed internal oscillator (HIRC) and one 10 kHz low-speed (LIRC). Both of them can be selected as the system clock. HIRC can be

enabled by setting HIRCEN (CKEN.5). LIRC is enabled after device is powered up. User can set OSC[1:0] (CKSWT [2:1]) as [1,1] to select the HIRC as the system clock. By setting OSC[1:0] as [1,0], LIRC will be selected as the system clock. Note that after the N76E003 is powered, HIRC and LIRC will be both enabled and HIRC is default selected as the system clock source. While using internal oscillators, X_{IN} automatically switch as one general purpose I/O P3.0 to expand the number of general purpose I/O. The I/O output mode of P3.0 can be selected by configuring P3M1 and P3M2 registers.

23.2 System Clock Switching

The N76E003 supports clock source switching on-the-fly by controlling CKSWT and CKEN registers via software. It provides a wide flexibility in application. Note that these SFRs are writing TA protected for precaution. With this clock source control, the clock source can be switched between the external clock source and the internal oscillator, even between the high and low-speed internal oscillator. However, during clock source switching, the device requires some amount of warm-up period for an original disabled clock source. Therefore, use should follow steps below to ensure a complete clock source switching. User can enable the target clock source by writing proper value into CKEN register, wait for the clock source stable by polling its status bit in CKSWT register, and switch to the target clock source by changing OSC[1:0] (CKSWT[2:1]). After these step, the clock source switching 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 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.

CKSWT – Clock Switch (TA protected)

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

Address: 96H

Reset value: 0011 0000b

Bit	Name	Description
7	-	Reserved
6	-	Reserved
5	HIRCST	High-speed internal oscillator 16 MHz status 0 = High-speed internal oscillator is not stable or disabled. 1 = High-speed internal oscillator is enabled and stable.
-	-	Reserved
3	ECLKST	External clock input status 0 = External clock input is not stable or disabled. 1 = External clock input is enabled and stable.
2:1	OSC[1:0]	Oscillator selection bits This field selects the system clock source. 00 = Internal 16 MHz oscillator. 01 = External clock source according to EXTEN[1:0] (CKEN[7:6]) setting. 10 = Internal 10 kHz oscillator. 11 = Reserved. Note that this field is write only. The read back value of this field may not correspond to the present system clock source.

CKEN – Clock Enable (TA protected)

7	6	5	4	3	2	1	0
EXTEN[1:0]		HIRCEN	-	-	-	-	CKSWTF
R/W		R/W	-	-	-	-	R

Address: 97H

Reset value: 0011 0000b

Bit	Name	Description
7:6	EXTEN[1:0]	External clock source enable 11 = External clock input via XIN Enabled. Others = external clock input is disable. P30 work as general purpose I/O.
5	HIRCEN	High-speed internal oscillator 16 MHz enable 0 = The high-speed internal oscillator Disabled. 1 = The high-speed internal oscillator Enabled. Note that once IAP is enabled by setting IAPEN (CHPCON.0), the high-speed internal 16 MHz oscillator will be enabled automatically. The hardware will also set HIRCEN and HIRCST bits. After IAPEN is cleared, HIRCEN and EHCST resume the original values.
4:1	-	Reserved
0	CKSWTF	Clock switch fault flag 0 = The previous system clock source switch was successful. 1 = User tried to switch to an instable or disabled clock source at the previous system clock source switch. If switching to an instable clock source, this bit remains 1 until the clock source is stable and switching is successful.

23.3 System Clock Divider

The oscillator frequency (F_{OSC}) can be divided down, by an integer, up to 1/510 by configuring a dividing register, CKDIV, to provide the system clock (F_{SYS}). This feature makes it possible to temporarily run the MCU at a lower rate, reducing power consumption. By dividing the clock, the MCU can retain the ability to respond to events other than those that can cause interrupts (i.e. events that allow exiting the Idle mode) by executing its normal program at a lower rate. This can often result in lower power consumption than in Idle mode. This can allow bypassing the oscillator start-up time in cases where Power-down mode would otherwise be used. The value of CKDIV may be changed by the program at any time without interrupting code execution.

CKDIV – Clock Divider

7	6	5	4	3	2	1	0
CKDIV[7:0]							
R/W							

Address: 95H

Reset value: 0000 0000b

Bit	Name	Description
7:0	CKDIV[7:0]	<p>Clock divider The system clock frequency F_{SYS} follows the equation below according to CKDIV value.</p> <p>$F_{SYS} = F_{OSC}$, while CKDIV = 00H, and</p> <p>$F_{SYS} = \frac{F_{OSC}}{2 \times CKDIV}$, while CKDIV = 01H to FFH.</p>

23.4 System Clock Output

The N76E003 provides a CLO pin (P1.1) that outputs the system clock. Its frequency is the same as F_{SYS} . The output enable bit is CLOEN (CKCON.1). CLO output stops when device is put in its Power-down mode because the system clock is turned off. Note that when noise problem or power consumption is important issue, user had better not enable CLO output.

Reset value: 0000 0000b

CKCON – Clock Control

7	6	5	4	3	2	1	0
-	PWMCKS	-	T1M	T0M	-	CLOEN	-
-	R/W	-	R/W	R/W	-	R/W	-

Address: 8EH, Page: 0

Bit	Name	Description
1	CLOEN	System clock output enable 0 = System clock output Disabled. 1 = System clock output Enabled from CLO pin (P1.1).

24. POWER MONITORING

To prevent incorrect execution during power up and power drop, The N76E003 provide two power monitor functions, power-on detection and brown-out detection.

24.1 Power-On Reset (POR)

The power-on detection function is designed for detecting power up after power voltage reaches to a level where system can work. After power-on detected, the POF (PCON.4) will be set 1 to indicate a cold reset, a power-on reset complete. The POF flag can be cleared via software.

PCON – Power Control

7	6	5	4	3	2	1	0
SMOD	SMOD0	-	POF	GF1	GF0	PD	IDL
R/W	R/W	-	R/W	R/W	R/W	R/W	R/W

Address: 87H

Reset value: see [Table 6-2. SFR Definitions and Reset Values](#)

Bit	Name	Description
4	POF	Power-on reset flag This bit will be set as 1 after a power-on reset. It indicates a cold reset, a power-on reset complete. This bit remains its value after any other resets. This flag is recommended to be cleared via software.

Notice:

N76E003 provides power-on detection to prevent incorrect execution during power up and power drop. The power-on detection function is designed for detecting power up after power voltage reaches to a level where system can work. The N76E003 POR-detect-voltage stables at one value which falls between 1.3 V to 1.5 V. When N76E003 runs in power down mode, the core runs under a low power consumption condition. Every time N76E003 wakes up from power-down mode, power consumption condition changes to normal power consumption. It can cause the core voltage glitch to less than 1.5 V. The POR will be trigger, and MCU will reset.

Workaround:

POR is for use by VDD power-on. After power-on, the system uses LVR for power detection. Strongly suggests that disable POR function every time after power-on reset at the initial part of Customer code.

To disable POR:

At SFR address, FDH is the PORDIS register to control disable POR function through software.

PORDIS – POR disable (TA protected)

7	6	5	4	3	2	1	0
PORDIS[7:0]							
W							

Address: FDH, Page: 0

Reset value: 0000 0000b

Bit	Name	Description
7:0	PORDIS[7:0]	POR disable Writing 5AH to the PORDIS and immediately followed by a writing of A5H will disable POR.

Since PORDIS register is TA protected, please always follow list code step to disable POR.

```
sfr PORDIS = 0XFD;
  TA =0XAA;
  TA= 0X55;
  PORDIS = 0X5A;
  TA=0XAA;
  TA=0X55;
  PORDIS = 0XA5;
```

24.2 Brown-Out Detection (BOD)

The other power monitoring function brown-out detection (BOD) circuit is used for monitoring the V_{DD} level during execution. There are eight CONFIG selectable brown-out trigger levels available for wide voltage applications. These eight nominal levels are 2.2V, 2.7V, 3.7V and 4.4V selected via setting CBOV[1:0] (CONFIG2[5:4]). BOD level can also be changed by setting BOV[1:0] (BODCON0[6:4]) after power-on. When V_{DD} drops to the selected brown-out trigger level (V_{BOD}), the BOD logic will either reset the MCU or request a brown-out interrupt. User may decide to being reset or generating a brown-out interrupt according to different applications. V_{BOD} also can be set by software after power-on. Note that BOD output is not available until 2~3 LIRC clocks after software enabling.

The BOD will request the interrupt while V_{DD} drops below V_{BOD} while BORST (BODCON0.2) is 0. In this case, BOF (BODCON0.3) will be set as 1. After user cleared this flag whereas V_{DD} remains below V_{BOD} , BOF will not set again. BOF just acknowledge user a power drop occurs. The BOF will also be set as 1 after V_{DD} goes higher than V_{BOD} to indicate a power resuming. The BOD circuit provides an useful status indicator BOS (BODCON0.0), which is helpful to tell a brown-out event or power resuming event occurrence. If the BORST bit is set as 1, this will enable brown-out reset function. After a brown-out reset, BORF (BODCON0.1) will be set as 1 via hardware. It will not be altered by reset other than power-on. This bit can be cleared by software. Note that all bits in BODCON0 is writing protected by timed access (TA).

The N76E003 provides low power BOD mode for saving current consumption and remaining BOD functionality with limited detection response. By setting LPBOD[1:0] (BODCON1[2:1]), the BOD circuit can be periodically enabled to sense the power voltage nominally every 1.6 ms, 6.4 ms, or 25.6 ms. It saves much power but also provides low-speed power voltage sensing. Note that the hysteresis feature will disappear in low power BOD mode.

For a noise sensitive system, the N76E003 has a BOD filter which filters the power noise to avoid BOD event triggering unconsciously. The BOD filter is enabled by default and can be disabled by setting BODFLT (BODCON1.0) as 0 if user requires a rapid BOD response. The minimum brown-out detect pulse width is listed in [Table 24-2](#).

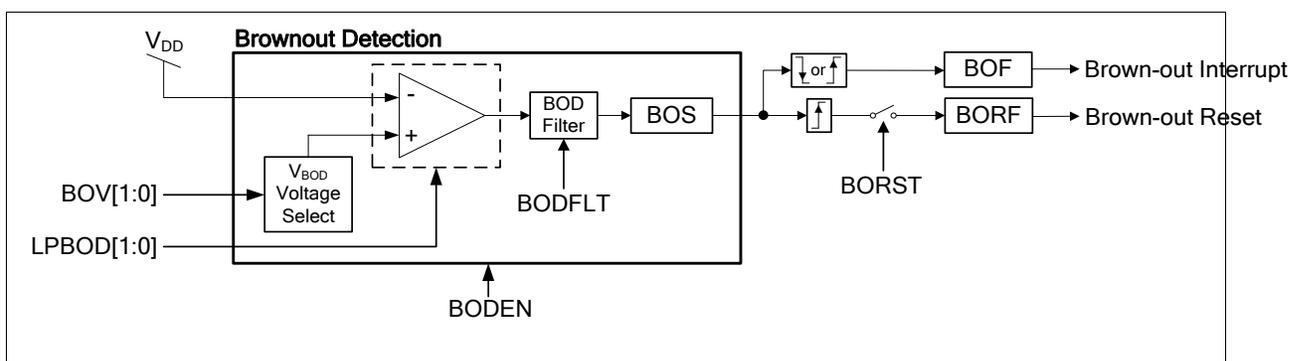


Figure 24-1. Brown-out Detection Block Diagram

CONFIG2

7	6	5	4	3	2	1	0
CBODEN	-	CBOV[1:0]		BOIAP	CBORST	-	-
R/W	-	R/W		R/W	R/W	-	-

Factory default value: 1111 1111b

Bit	Name	Description
7	CBODEN	CONFIG brown-out detect enable 1 = Brown-out detection circuit on. 0 = Brown-out detection circuit off.
5:4	CBOV[1:0]	CONFIG brown-out voltage select 11 = V _{BOD} is 2.2V. 10 = V _{BOD} is 2.7V. 01 = V _{BOD} is 3.7V. 00 = V _{BOD} is 4.4V.
2	CBORST	CONFIG brown-out reset enable This bit decides whether a brown-out reset is caused by a power drop below V _{BOD} . 1 = Brown-out reset Enabled. 0 = Brown-out reset Disabled.

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

7	6	5	4	3	2	1	0
BODEN ^[1]		BOV[1:0] ^[1]		BOF ^[2]	BORST ^[1]	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
7	BODEN	Brown-out detection enable 0 = Brown-out detection circuit off. 1 = Brown-out detection circuit on. Note that BOD output is not available until 2~3 LIRC clocks after enabling.
6:4	BOV[1:0]	Brown-out voltage select 11 = V _{BOD} is 2.2V. 10 = V _{BOD} is 2.7V. 01 = V _{BOD} is 3.7V. 00 = V _{BOD} is 4.4V.
3	BOF	Brown-out interrupt flag This flag will be set as logic 1 via hardware after a V _{DD} dropping below or rising above V _{BOD} event occurs. If both EBOD (EIE.2) and EA (IE.7) are set, a brown-out interrupt requirement will be generated. This bit should be cleared via software.
2	BORST	Brown-out reset enable This bit decides whether a brown-out reset is caused by a power drop below V _{BOD} . 0 = Brown-out reset when V _{DD} drops below V _{BOD} Disabled. 1 = Brown-out reset when V _{DD} drops below V _{BOD} Enabled.
1	BORF	Brown-out reset flag 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.
0	BOS	Brown-out status This bit indicates the V _{DD} voltage level comparing with V _{BOD} while BOD circuit is enabled. It keeps 0 if BOD is not enabled. 0 = V _{DD} voltage level is higher than V _{BOD} or BOD is disabled. 1 = V _{DD} voltage level is lower than V _{BOD} . Note that this bit is read-only.

[1] BODEN, BOV[1:0], and BORST are initialized by being directly loaded from CONFIG2 bit 7, [6:4], and 2 after all resets.

[2] BOF reset value depends on different setting of CONFIG2 and V_{DD} voltage level. Please check [Table 24-1](#).

Table 24-1. 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]	<p>Low power BOD enable 00 = BOD normal mode. BOD circuit is always enabled. 01 = BOD low power mode 1 by turning on BOD circuit every 1.6 ms periodically. 10 = BOD low power mode 2 by turning on BOD circuit every 6.4 ms periodically. 11 = BOD low power mode 3 by turning on BOD circuit every 25.6 ms periodically.</p>
0	BODFLT	<p>BOD filter control BOD has a filter which counts 32 clocks of F_{sys} to filter the power noise when MCU runs with HIRC, or ECLK as the system clock and BOD does not operate in its low power mode (LPBOD[1:0] = [0, 0]). In other conditions, the filter counts 2 clocks of LIRC. Note that when CPU is halted in Power-down mode. The BOD output is permanently filtered by 2 clocks of LIRC. The BOD filter avoids the power noise to trigger BOD event. This bit controls BOD filter enabled or disabled. 0 = BOD filter Disabled. 1 = BOD filter Enabled. (Power-on reset default value.)</p>

Table 24-2. Minimum Brown-out Detect Pulse Width

BODFLT (BODCON1.1)	BOD Operation Mode	System Clock Source	Minimum Brown-out Detect Pulse Width
0	Normal mode (LPBOD[1:0] = [0,0])	Any clock source	Typ. 1 μ s
	Low power mode 1 (LPBOD[1:0] = [0,1])	Any clock source	16 (1/F _{LIRC})
	Low power mode 2 (LPBOD[1:0] = [1,0])	Any clock source	64 (1/F _{LIRC})
	Low power mode 3 (LPBOD[1:0] = [1,1])	Any clock source	256 (1/ F _{LIRC})
1	Normal mode (LPBOD[1:0] = [0,0])	HIRC/ECLK	Normal operation: 32 (1/F _{SYS}) Idle mode: 32 (1/F _{SYS}) Power-down mode: 2 (1/F _{LIRC})
		LIRC	2 (1/F _{LIRC})
	Low power mode 1 (LPBOD[1:0] = [0,1])	Any clock source	18 (1/F _{LIRC})
	Low power mode 2 (LPBOD[1:0] = [1,0])	Any clock source	66 (1/F _{LIRC})
	Low power mode 3 (LPBOD[1:0] = [1,1])	Any clock source	258 (1/ F _{LIRC})

25. RESET

The N76E003 has several options to place device in reset condition. It also offers the software flags to indicate the source, which causes a reset. In general, most SFRs go to their Reset value irrespective of the reset condition, but there are several reset source indicating flags whose state depends on the source of reset. User can read back these flags to determine the cause of reset using software. There are six ways of putting the device into reset state. They are power-on reset, brown-out reset, external reset, hard fault reset, WDT reset, and software reset.

25.1 Power-On Reset

The N76E003 incorporates an internal power-on reset. During a power-on process of rising power supply voltage V_{DD} , the power-on reset will hold the MCU in reset mode when V_{DD} is lower than the voltage reference threshold. This design makes CPU not access program flash while the V_{DD} is not adequate performing the flash reading. If an undetermined operating code is read from the program flash and executed, this will put CPU and even the whole system in to an erroneous state. After a while, V_{DD} rises above the threshold where the system can work, the selected oscillator will start and then program code will execute from 0000H. At the same time, a power-on flag POF (PCON.4) will be set 1 to indicate a cold reset, a power-on reset complete. Note that the contents of internal RAM will be undetermined after a power-on. It is recommended that user gives initial values for the RAM block.

The POF is recommended to be cleared to 0 via software to check if a cold reset or warm reset performed after the next reset occurs. If a cold reset caused by power off and on, POF will be set 1 again. If the reset is a warm reset caused by other reset sources, POF will remain 0. User may take a different course to check other reset flags and deal with the warm reset event.

PCON – Power Control

7	6	5	4	3	2	1	0
SMOD	SMOD0	-	POF	GF1	GF0	PD	IDL
R/W	R/W	-	R/W	R/W	R/W	R/W	R/W

Address: 87H

Reset value: see [Table 6-2. SFR Definitions and Reset Values](#)

Bit	Name	Description
4	POF	Power-on reset flag This bit will be set as 1 after a power-on reset. It indicates a cold reset, a power-on reset complete. This bit remains its value after any other resets. It is recommended that the flag be cleared via software.

25.2 Brown-Out Reset

The brown-out detection circuit is used for monitoring the V_{DD} level during execution. When V_{DD} drops to the selected brown-out trigger level (V_{BOD}), the brown-out detection logic will reset the MCU if

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[1:0]		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 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.

25.3 External Reset and Hard Fault 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	HardF	-	GF2	UART0PX	0	DPS
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 When the MCU is reset by the external reset pin, this bit will be set via hardware. It is recommended that the flag be cleared via software.

Bit	Name	Description
5	HardF	<p>Hard Fault reset flag Once Program Counter (PC) is over flash size, MCU will be reset and this bit will be set via hardware. It is recommended that the flag be cleared via software.</p> <p>Note: If MCU run in OCD debug mode and OCDEN = 0, hard fault reset will be disabled and only HardF flag be asserted.</p>

25.4 Hard Fault Reset

If Program Counter (PC) is over flash size, Hard Fault reset will occur. After Hard Fault reset, HardF (AUXR1.5) flag will be set via hardware. HardF will not change after any reset other than a power-on reset or the Hard Fault reset itself. This bit can be cleared via software. If MCU run in OCD debug mode and OCDEN = 0, hard fault reset will be disabled and only HardF flag be asserted.

25.5 Watchdog Timer Reset

The WDT is a free running timer with programmable time-out intervals and a dedicated internal clock source. User can clear the WDT at any time, causing it to restart the counter. When the selected time-out occurs but no software response taking place for a while, the WDT will reset the system directly and CPU will begin execution from 0000H.

Once a reset due to WDT occurs, the WDT reset flag WDTRF (WDCON.3) will be set. This bit keeps unchanged after any reset other than a power-on reset or WDT reset itself. User can clear WDTRF via software.

WDCON – Watchdog Timer Control (TA protected)

7	6	5	4	3	2	1	0
WDTR	WDCLR	WDTF	WIDPD	WDTRF	WDPS[2:0]		
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
3	WDTRF	<p>WDT reset flag When the MCU is reset by WDT time-out event, this bit will be set via hardware. It is recommended that the flag be cleared via software.</p>

25.6 Software Reset

The N76E003 provides a software reset, which allows the software to reset the whole system just similar to an external reset, initializing the MCU as it reset state. The software reset is quite useful in

the end of an ISP progress. For example, if an ISP of Boot Code updating User Code finishes, a software reset can be asserted to re-boot CPU to execute new User Code immediately. Writing 1 to SWRST (CHPCON.7) will trigger a software reset. Note that this bit is writing TA protection. The instruction that sets the SWRST bit is the last instruction that will be executed before the device reset. See demo code below.

If a software reset occurs, SWRF (AUXR.7) will be automatically set by hardware. User can check it as the reset source indicator. SWRF keeps unchanged after any reset other than a power-on reset or software reset itself. SWRF can 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 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	HardF	-	GF2	UART0PX	0	DPS
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 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
```

25.7 Boot Select

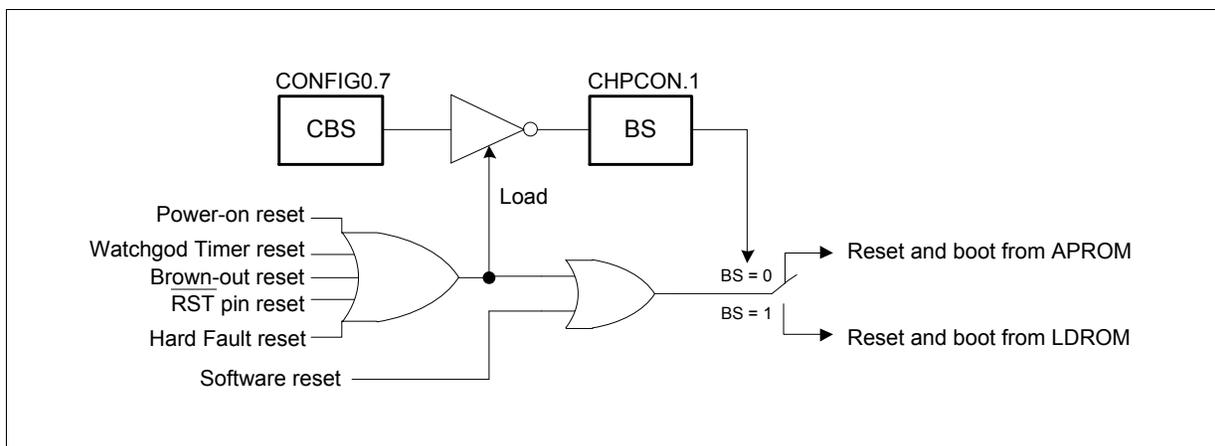


Figure 25-1. Boot Selecting Diagram

The N76E003 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 address 0000H of LDROM. Note that BS is loaded from the inverted value of CBS bit in CONFIG0.7 after all resets except software reset.

CONFIG0

7	6	5	4	3	2	1	0
CBS	-	OCDPWM	OCDEN	-	RPD	LOCK	-
R/W	-	R/W	R/W	-	R/W	R/W	-

Factory default value: 1111 1111b

Bit	Name	Description
7	CBS	CONFIG boot select This bit defines from which block that MCU re-boots after resets except software reset. 1 = MCU will re-boot from APROM after resets except software reset. 0 = MCU will re-boot from LDROM after resets except software reset.

CHPCON – Chip Control (TA protected)

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

Address: 9FH

Reset value: see [Table 6-2. SFR Definitions and Reset Values](#)

Bit	Name	Description
1	BS	Boot select This bit defines from which block that MCU re-boots after all resets. 0 = MCU will re-boot from APROM after all resets. 1 = MCU will re-boot from LDROM after all resets.

[1] BS is initialized by being loaded from the inverted value of CBS bit in CONFIG0.7 after resets except software reset. It keeps unchanged after software reset.

After the MCU is released from reset state, the hardware will always check the BS bit instead of the CBS bit to determine from which block that the device reboots.

25.8 Reset State

The reset state besides power-on reset does not affect the on-chip RAM. The data in the RAM will be preserved during the reset. After the power-on reset the RAM contents will be indeterminate.

After a reset, most of SFRs go to their initial values except bits, which are affected by different reset events. See the notes of [Table 6-2. SFR Definitions and Reset Values](#). The Program Counter is forced to 0000H and held as long as the reset condition is applied. Note that the Stack Pointer is also reset to 07H and thus the stack contents may be effectively lost during the reset event even though the RAM contents are not altered.

After a reset, all peripherals and interrupts are disabled. The I/O port latches resumes FFH and I/O mode input-only.

26. AUXILIARY FEATURES

26.1 Dual DPTRs

The original 8051 contains one DPTR (data pointer) only. With single DPTR, it is difficult to move data from one address to another with wasting code size and low performance. The N76E003 provides two data pointers. Thus, software can load both a source and a destination address when doing a block move. Once loading, the software simply switches between DPTR and DPTR1 by the active data pointer selection DPS (AUXR1.0) bit.

An example of 64 bytes block move with dual DPTRs is illustrated below. By giving source and destination addresses in data pointers and activating cyclic makes block RAM data move more simple and efficient than only one DPTR. The `INC AUXR1` instruction is the shortest (2 bytes) instruction to accomplish DPTR toggling rather than `ORL` or `ANL`. For AUXR1.1 contains a hard-wired 0, it allows toggling of the DPS bit by incrementing AUXR1 without interfering with other bits in the register.

```

MOV    R0, #64           ;number of bytes to move
MOV    DPTR, #D_Addr    ;load destination address
INC    AUXR1             ;change active DPTR
MOV    DPTR, #S_Addr    ;load source address
LOOP:
MOVX   A, @DPTR         ;read source data byte
INC    AUXR1            ;change DPTR to destination
MOVX   @DPTR, A        ;write data to destination
INC    DPTR             ;next destination address
INC    AUXR1            ;change DPTR to source
INC    DPTR             ;next source address
DJNZ   R0, LOOP
INC    AUXR1            ;(optional) restore DPS
    
```

AUXR1 also contains a general purpose flag GF2 in its bit 3 that can be set or cleared by the user via software.

DPL – Data Pointer Low Byte

7	6	5	4	3	2	1	0
DPL[7:0]							
R/W							

Address: 82H

reset value: 0000 0000b

Bit	Name	Description
7:0	DPL[7:0]	<p>Data pointer low byte</p> <p>This is the low byte of 16-bit data pointer. DPL combined with DPH serve as a 16-bit data pointer DPTR to address non-scratch-pad memory or Program Memory. DPS (DPS.0) bit decides which data pointer, DPTR or DPTR1, is activated.</p>

DPH – Data Pointer High Byte

7	6	5	4	3	2	1	0
DPH[7:0]							
R/W							

Address: 83H

reset value: 0000 0000b

Bit	Name	Description
7:0	DPH[7:0]	<p>Data pointer high byte</p> <p>This is the high byte of 16-bit data pointer. DPH combined with DPL serve as a 16-bit data pointer DPTR to address non-scratch-pad memory or Program Memory. DPS (DPS.0) bit decides which data pointer, DPTR or DPTR1, is activated.</p>

AUXR1 – Auxiliary Register 1

7	6	5	4	3	2	1	0
SWRF	RSTPINF	HardF	-	GF2	UART0PX	0	DPS
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
3	GF2	<p>General purpose flag 2</p> <p>The general purpose flag that can be set or cleared by the user via software.</p>
1	0	<p>Reserved</p> <p>This bit is always read as 0.</p>
0	DPS	<p>Data pointer select</p> <p>0 = Data pointer 0 (DPTR) is active by default. 1 = Data pointer 1 (DPTR1) is active. After DPS switches the activated data pointer, the previous inactivated data pointer remains its original value unchanged.</p>

26.2 96-bit UID

Before shipping out, each N76E003 chip was factory pre-programmed with a 96-bit width serial number, which is guaranteed to be unique. The serial number is called UID. The user can read the unique code only by IAP command. Please see [IAP Commands](#).

27. ON-CHIP-DEBUGGER (OCD)

27.1 Functional Description

The N76E003 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 P2.0/ $\overline{\text{P2.0}}$ pin is also necessary for OCD mode entry and exit. The N76E003 supports OCD with Flash Memory control path by ICP writer mode, which shares the same three pins of OCD interface.

The N76E003 uses OCDDA, OCDCK, and P2.0/ $\overline{\text{P2.0}}$ 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 P2.0/ $\overline{\text{P2.0}}$ is configured as external reset pin, it cannot be connected directly to V_{DD} and any external capacitors connected must be removed.
2. If P2.0/ $\overline{\text{P2.0}}$ is configured as input pin P2.0, 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.

27.2 Limitation of OCD

The N76E003 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 P2.0/ $\overline{\text{P2.0}}$ pin needs to be used for OCD mode selection. Therefore, neither P2.0 input nor an external reset source can be emulated.

2. The OCDDA pin is physically located on the same pin P1.6. Therefore, neither its I/O function nor shared multi-functions can be emulated.
3. The OCDCK pin is physically located on the same pin as P0.2. Therefore, neither its I/O function nor shared multi-functions can be emulated.
4. When the system is in Idle or Power-down mode, it is invalid to perform any accesses because parts of the device may not be clocked. A read access could return garbage or a write access might not succeed.
5. HIRC cannot be turned off because OCD uses this clock to monitor its internal status. The instruction that turns off HIRC affects nothing if executing under debug mode. When CPU enters its Power-down mode under debug mode, HIRC keeps turning on.

The N76E003 OCD system has another limitation that non-intrusive commands cannot be executed at any time while the user's program is running. Non-intrusive commands allow a user to read or write MCU memory locations or access status and control registers with the debug controller. A reading or writing memory or control register space is allowed only when MCU is under halt condition after a matching of the hardware address breakpoint or a single step running.

CONFIG0

7	6	5	4	3	2	1	0
CBS	-	OCDPWM	OCDEN	-	RPD	LOCK	-
R/W	-	R/W	R/W	-	R/W	R/W	-

Factory default value: 1111 1111b

Bit	Name	Description
5	OCDPWM	<p>PWM output state under OCD halt This bit decides the output state of PWM when OCD halts CPU. 1 = Tri-state pins those are used as PWM outputs. 0 = PWM continues. Note that this bit is valid only when the corresponding PIO bit of PWM channel is set as 1.</p>
4	OCDEN	<p>OCD enable 1 = OCD Disabled. 0 = OCD Enabled.</p> <p>Note: If MCU run in OCD debug mode and OCDEN = 0, hard fault reset will be disabled and only HardF flag be asserted.</p>

28. CONFIG BYTES

The N76E003 has several hardware configuration bytes, called CONFIG, those are used to configure the hardware options such as the security bits, system clock source, and so on. These hardware options can be re-configured through the parallel Writer, In-Circuit-Programming (ICP), or In-Application-Programming (IAP). Several functions, which are defined by certain CONFIG bits are also available to be re-configured by SFR. Therefore, there is a need to load such CONFIG bits into respective SFR bits. Such loading will occur after resets. These SFR bits can be continuously controlled via user's software.

CONFIG bits marked as “-“should always keep un-programmed.

CONFIG0

7	6	5	4	3	2	1	0
CBS	-	OCDPWM	OCDEN	-	RPD	LOCK	-
R/W	-	R/W	R/W	-	R/W	R/W	-

Factory default value: 1111 1111b

Bit	Name	Description
7	CBS	CONFIG boot select This bit defines from which block that MCU re-boots after resets except software reset. 1 = MCU will re-boot from APROM after resets except software reset. 0 = MCU will re-boot from LDROM after resets except software reset.
5	OCDPWM	PWM output state under OCD halt This bit decides the output state of PWM when OCD halts CPU. 1 = Tri-state pins those are used as PWM outputs. 0 = PWM continues. Note that this bit is valid only when the corresponding PIO bit of PWM channel is set as 1.
4	OCDEN	OCD enable 1 = OCD Disabled. 0 = OCD Enabled. Note: If MCU run in OCD debug mode and OCDEN = 0, hard fault reset will be disabled and only Hard F flag be asserted.
3	-	Reserved
2	RPD	Reset pin disable 1 = The reset function of P2.0/nRST pin Enabled. P2.0/nRST functions as the external reset pin. 0 = The reset function of P2.0/nRST pin Disabled. P2.0/nRST functions as an input-only pin P2.0.

Bit	Name	Description
1	LOCK	<p>Chip lock enable 1 = Chip is unlocked. Flash Memory is not locked. Their contents can be read out through a parallel Writer/ICP programmer. 0 = Chip is locked. Whole Flash Memory is locked. Their contents read through a parallel Writer or ICP programmer will be all blank (FFH). Programming to Flash Memory is invalid. Note that CONFIG bytes are always unlocked and can be read. Hence, once the chip is locked, the CONFIG bytes cannot be erased or programmed individually. The only way to disable chip lock is execute "whole chip erase". However, all data within the Flash Memory and CONFIG bits will be erased when this procedure is executed. If the chip is locked, it does not alter the IAP function.</p>

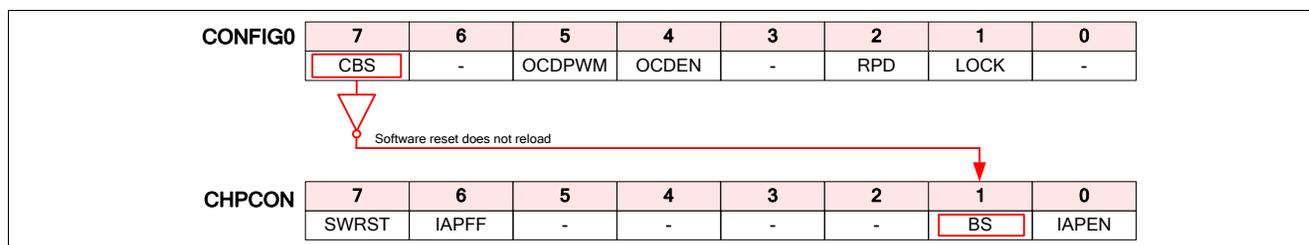


Figure 28-1. CONFIG0 Any Reset Reloading

CONFIG1

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

Factory default value: 1111 1111b

Bit	Name	Description
2:0	LDSIZE[2:0]	<p>LDROM size select Part number is N76E003: 111 = No LDROM. APROM is 18K Bytes. 110 = LDROM is 1K Bytes. APROM is 17K Bytes. 101 = LDROM is 2K Bytes. APROM is 16K Bytes. 100 = LDROM is 3K Bytes. APROM is 15K Bytes. 0xx = LDROM is 4K Bytes. APROM is 14K Bytes.</p>

CONFIG2

7	6	5	4	3	2	1	0
CBODEN	-	CBOV[1:0]		BOIAP	CBORST	-	-
R/W	-	R/W		R/W	R/W	-	-

Factory default value: 1111 1111b

Bit	Name	Description
7	CBODEN	<p>CONFIG brown-out detect enable 1 = Brown-out detection circuit on. 0 = Brown-out detection circuit off.</p>
6	-	Reserved

Bit	Name	Description
5:4	CBOV[1:0]	CONFIG brown-out voltage select 11 = V _{BOD} is 2.2V. 10 = V _{BOD} is 2.7V. 01 = V _{BOD} is 3.7V. 00 = V _{BOD} is 4.4V.
3	BOIAP	Brown-out inhibiting IAP This bit decides whether IAP erasing or programming is inhibited by brown-out status. This bit is valid only when brown-out detection is enabled. 1 = IAP erasing or programming is inhibited if V _{DD} is lower than V _{BOD} . 0 = IAP erasing or programming is allowed under any workable V _{DD} .
2	CBORST	CONFIG brown-out reset enable This bit decides whether a brown-out reset is caused by a power drop below V _{BOD} . 1 = Brown-out reset Enabled. 0 = Brown-out reset Disabled.

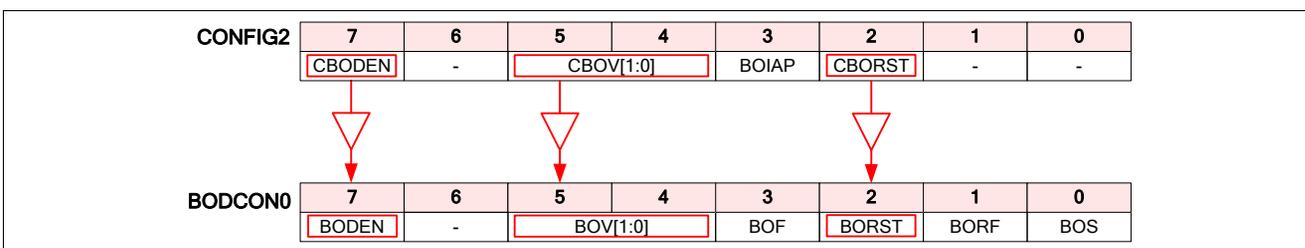


Figure 28-2. CONFIG2 Power-On Reset Reloading

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 This field configures the WDT behavior after MCU execution. 1111 = WDT is Disabled. WDT can be used as a general purpose timer via software control. 0101 = WDT is Enabled as a time-out reset timer and it stops running during Idle or Power-down mode. Others = WDT is Enabled as a time-out reset timer and it keeps running during Idle or Power-down mode.
3:0	-	Reserved

29. IN-CIRCUIT-PROGRAMMING (ICP)

The Flash Memory can be programmed by “n-Circuit-Programming” (P). If the product is just under development or the end product needs firmware updating in the hand of an end customer, the hardware programming mode will make repeated programming difficult and inconvenient. ICP method makes it easy and possible without removing the microcontroller from the system. ICP mode also allows customers to manufacture circuit boards with un-programmed devices. Programming can be done after the assembly process allowing the device to be programmed with the most recent firmware or a customized firmware.

There are three signal pins, \overline{ICPDA} , ICPDA, and ICPCK, involved in ICP function. \overline{ICPDA} is used to enter or exit ICP mode. ICPDA is the data input and output pin. ICPCK is the clock input pin, which synchronizes the data shifted in to or out from MCU under programming. User should leave these three pins plus VDD and GND pins on the circuit board to make ICP possible.

Nuvoton provides ICP tool for N76E003, which enables user to easily perform ICP through Nuvoton ICP programmer. The ICP programmer developed by Nuvoton has been optimized according to the electric characteristics of MCU. It also satisfies the stability and efficiency during production progress. For more details, please visit Nuvoton 8-bit Microcontroller website: [Nuvoton 80C51 Microcontroller Technical Support](#).

30. INSTRUCTION SET

The N76E003 executes all the instructions of the standard 80C51 family fully compatible with MCS-51. However, the timing of each instruction is different for it uses high performance 1T 8051 core. The architecture eliminates redundant bus states and implements parallel execution of fetching, decode, and execution phases. The N76E003 uses one clock per machine-cycle. It leads to performance improvement of rate 8.1 (in terms of MIPS) with respect to traditional 12T 80C51 device working at the same clock frequency. However, the real speed improvement seen in any system will depend on the instruction mix.

All instructions are coded within an 8-bit field called an OPCODE. This single byte should be fetched from Program Memory. The OPCODE is decoded by the CPU. It determines what action the CPU will take and whether more operation data is needed from memory. If no other data is needed, then only one byte was required. Thus the instruction is called a one byte instruction. In some cases, more data is needed, which is two or three byte instructions.

[Table 30-1](#) lists all instructions for details. The note of the instruction set and addressing modes are shown below.

Rn (n = 0~7)	Register R0 to R7 of the currently selected Register Bank.
Direct location	8-bit internal data location's address. It could be an internal data RAM (00H to 7FH) or an SFR (80H to FFH).
@RI (I = 0, 1) through	8-bit internal data RAM location (00H to FFH) addressed indirectly through register R0 or R1.
#data	8-bit constant included in the instruction.
#data16	16-bit constant included in the instruction.
Addr16 be	16-bit destination address. Used by LCALL and LJMP. A branch can be anywhere within the Program Memory address space.
Addr11 be the	11-bit destination address. Used by ACALL and AJMP. The branch will be within the same 2K-Byte page of Program Memory as the first byte of the following instruction.
Rel conditional following	signed ('s complement) 8-bit offset Byte. Used by SJMP and all branches. The range is -128 to +127 Bytes relative to first byte of the following instruction.
Bit	Direct addressed bit in internal data RAM or SFR.

Table 30-1. Instruction Set

Instruction	OPCODE	Bytes	Clock Cycles	N76E003 V.S. Tradition 80C51 Speed Ratio
NOP	00	1	1	12
ADD A, Rn	28~2F	1	2	6
ADD A, direct	25	2	3	4
ADD A, @Ri	26, 27	1	4	3
ADD A, #data	24	2	2	6
ADDC A, Rn	38~3F	1	2	6
ADDC A, direct	35	2	3	4
ADDC A, @Ri	36, 37	1	4	3
ADDC A, #data	34	2	2	6
SUBB A, Rn	98~9F	1	2	6
SUBB A, direct	95	2	3	4
SUBB A, @Ri	96, 97	1	4	3
SUBB A, #data	94	2	2	6
INC A	04	1	1	12
INC Rn	08~0F	1	3	4
INC direct	05	2	4	3
INC @Ri	06, 07	1	5	2.4
INC DPTR	A3	1	1	24
DEC A	14	1	1	12
DEC Rn	18~1F	1	3	4
DEC direct	15	2	4	3
DEC @Ri	16, 17	1	5	2.4
MUL AB	A4	1	4	12
DIV AB	84	1	4	12
DA A	D4	1	1	12
ANL A, Rn	58~5F	1	2	6
ANL A, direct	55	2	3	4
ANL A, @Ri	56, 57	1	4	3
ANL A, #data	54	2	2	6
ANL direct, A	52	2	4	3
ANL direct, #data	53	3	4	6
ORL A, Rn	48~4F	1	2	6
ORL A, direct	45	2	3	4
ORL A, @Ri	46, 47	1	4	3
ORL A, #data	44	2	2	6
ORL direct, A	42	2	4	3
ORL direct, #data	43	3	4	6
XRL A, Rn	68~6F	1	2	6
XRL A, direct	65	2	3	4
XRL A, @Ri	66, 67	1	4	3

Table 30-1. Instruction Set

Instruction	OPCODE	Bytes	Clock Cycles	N76E003 V.S. Tradition 80C51 Speed Ratio
XRL A, #data	64	2	2	6
XRL direct, A	62	2	4	3
XRL direct, #data	63	3	4	6
CLR A	E4	1	1	12
CPL A	F4	1	1	12
RL A	23	1	1	12
RLC A	33	1	1	12
RR A	03	1	1	12
RRC A	13	1	1	12
SWAP A	C4	1	1	12
MOV A, Rn	E8~EF	1	1	12
MOV A, direct	E5	2	3	4
MOV A, @Ri	E6, E7	1	4	3
MOV A, #data	74	2	2	6
MOV Rn, A	F8~FF	1	1	12
MOV Rn, direct	A8~AF	2	4	6
MOV Rn, #data	78~7F	2	2	6
MOV direct, A	F5	2	2	6
MOV direct, Rn	88~8F	2	3	8
MOV direct, direct	85	3	4	6
MOV direct, @Ri	86, 87	2	5	4.8
MOV direct, #data	75	3	3	8
MOV @Ri, A	F6, F7	1	3	4
MOV @Ri, direct	A6, A7	2	4	6
MOV @Ri, #data	76, 77	2	3	6
MOV DPTR, #data16	90	3	3	8
MOVC A, @A+DPTR	93	1	4	6
MOVC A, @A+PC	83	1	4	6
MOVX A, @Ri ^[1]	E2, E3	1	5	4.8
MOVX A, @DPTR ^[1]	E0	1	4	6
MOVX @Ri, A ^[1]	F2, F3	1	6	4
MOVX @DPTR, A ^[1]	F0	1	5	4.8
PUSH direct	C0	2	4	6
POP direct	D0	2	3	8
XCH A, Rn	C8~CF	1	2	6
XCH A, direct	C5	2	3	4
XCH A, @Ri	C6, C7	1	4	3
XCHD A, @Ri	D6, D7	1	5	2.4
CLR C	C3	1	1	12
CLR bit	C2	2	4	3

Table 30-1. Instruction Set

Instruction	OPCODE	Bytes	Clock Cycles	N76E003 V.S. Tradition 80C51 Speed Ratio
SETB C	D3	1	1	12
SETB bit	D2	2	4	3
CPL C	B3	1	1	12
CPL bit	B2	2	4	3
ANL C, bit	82	2	3	8
ANL C, /bit	B0	2	3	8
ORL C, bit	72	2	3	8
ORL C, /bit	A0	2	3	8
MOV C, bit	A2	2	3	4
MOV bit, C	92	2	4	6
ACALL addr11	11, 31, 51, 71, 91, B1, D1, F1 ^[2]	2	4	6
LCALL addr16	12	3	4	6
RET	22	1	5	4.8
RETI	32	1	5	4.8
AJMP addr11	01, 21, 41, 61, 81, A1, C1, E1 ^[3]	2	3	8
LJMP addr16	02	3	4	6
SJMP rel	80	2	3	8
JMP @A+DPTR	73	1	3	8
JZ rel	60	2	3	8
JNZ rel	70	2	3	8
JC rel	40	2	3	8
JNC rel	50	2	3	8
JB bit, rel	20	3	5	4.8
JNB bit, rel	30	3	5	4.8
JBC bit, rel	10	3	5	4.8
CJNE A, direct, rel	B5	3	5	4.8
CJNE A, #data, rel	B4	3	4	6
CJNE Rn, #data, rel	B8~BF	3	4	6
CJNE @Ri, #data, rel	B6, B7	3	6	4
DJNZ Rn, rel	D8~DF	2	4	6
DJNZ direct, rel	D5	3	5	4.8

[1] The N76E003 does not have external memory bus. MOVX instructions are used to access internal XRAM.

[2] The most three significant bits in the 11-bit address [A10:A8] decide the ACALL hex code. The code will be [A10,A9,A8,1,0,0,0,1].

[3] The most three significant bits in the 11-bit address [A10:A8] decide the AJMP hex code. The code will be [A10,A9,A8,0,0,0,0,1].

31. ELECTRICAL CHARACTERISTICS

31.1 Absolute Maximum Ratings

Parameter	Rating	Unit
Operating temperature under bias (T_A)	-40 to +105	°C
Storage temperature range	-55 to +150	°C
Voltage on VDD pin to GND pin	-0.3 to +6.3	V
Voltage on any other pin to GND pin	-0.3 to ($V_{DD}+0.3$)	V

Stresses at or above those listed under “Absolute maximum ratings” may cause permanent damage to the device. It is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions may affect device reliability.

31.2 D.C. Electrical Characteristics

Table 31-1. D.C. Electrical Characteristics

($V_{DD} - V_{SS} = 2.4 \sim 5.5$ V, $T_A = 25$ °C)

Symbol	Parameter	Condition	Min.	Typ.	Max.	Unit
Supply voltage						
V_{DD}	Operating voltage	F = 0 to 16 MHz	2.4	-	5.5	V
I/O						
V_{IL}	Input low voltage (I/O with TTL input)		$V_{SS}-0.3$	-	$0.2V_{DD}-0.1$	V
V_{IL1}	Input low voltage (I/O with Schmitt trigger input, \overline{XIN} , and XIN)		$V_{SS}-0.3$	-	$0.3V_{DD}$	V
V_{IH}	Input high voltage (I/O with TTL input)		$0.2V_{DD}+0.9$	-	$V_{DD}+0.3$	V
V_{IH1}	Input high voltage (I/O with Schmitt trigger input and XIN)		$0.7V_{DD}$	-	$V_{DD}+0.3$	V
V_{IH2}	Input high voltage (\overline{XIN})		$0.8V_{DD}$	-	$V_{DD}+0.3$	V
V_{OL}	Output low voltage ^[1] (Normal sink current strength, all modes except input-only)	$V_{DD} = 5.5$ V, $I_{OL} = 15$ mA	-	-	0.4	V
		$V_{DD} = 4.5$ V, $I_{OL} = 13$ mA	-	-	0.4	
		$V_{DD} = 3.0$ V, $I_{OL} = 9$ mA	-	-	0.4	
		$V_{DD} = 2.4$ V, $I_{OL} = 7$ mA	-	-	0.4	

Symbol	Parameter	Condition	Min.	Typ.	Max.	Unit
V _{OH}	Output high voltage (quasi-bidirectional mode)	V _{DD} = 5.5 V, I _{OH} = - 590 μA	2.4	-	-	V
		V _{DD} = 4.5 V, I _{OH} = - 380 μA	2.4	-	-	
		V _{DD} = 3.0 V, I _{OH} = - 100 μA	2.4	-	-	
		V _{DD} = 2.4 V, I _{OH} = - 40 μA	2.0	-	-	
V _{OH1}	Output high voltage (push-pull mode)	V _{DD} = 5.5 V, I _{OH} = - 20 mA	2.4	-	-	V
		V _{DD} = 4.5 V, I _{OH} = - 13 mA	2.4	-	-	
		V _{DD} = 3.0 V, I _{OH} = - 3.5 mA	2.4	-	-	
		V _{DD} = 2.4 V, I _{OH} = - 2 mA	2.0	-	-	
I _{IL}	Logical 0 input current (quasi-bidirectional mode)	V _{DD} = 5.5 V, V _{IN} = 0.4 V	-	-	-50	μA
I _{TL}	Logical 1-to-0 transition current ^[2] (quasi-bidirectional mode)	V _{DD} = 5.5 V	--	-	-650	μA
I _{LI}	Input leakage current (open-drain or input-only mode)		-	1	±10	μA
R _{RST}	_____ pin internal pull-low resistor		50	-	600	kΩ
Supply current						
I _{DD}	Normal operating current ^[3]	HIRC	-	3.3	3.9	mA
		LIRC	-	170	240	μA
I _{IDL}	Idle mode current	HIRC		2.2	2.6	mA
		LIRC	-	160	230	μA
I _{PD}	Power-down mode current (BOD off)	T _A = 25 °C	-	6	8	μA
		T _A = - 40 °C ~ +105 °C	-	20	40	μA

[1] Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows,

Maximum I_{OL} per port pin: 15mA

Maximum total I_{OL} for all outputs: 100mA

[2] Pins of all ports in quasi-bidirectional mode source a transition current when they are being externally driven from 1 to 0. The transition current reaches its maximum value when V_{IN} is approximately 2V.

[3] t is measured while U keeps in running “ J P \$” loop continuously. All pins of ports are configured as quasi-bidirectional mode.

31.3 A.C. Electrical Characteristics

Table 31-2. I/O Slew Rate A.C. Electrical Characteristics

PxSR.n bit value	Symbol	Parameter	Condition	Min.	Typ.	Max.	Unit
0	F _{OUT}	Maximum output frequency ^[1]	V _{DD} = 5.0 V, C _L = 30 pF	-	34	-	MHz
			V _{DD} = 3.3 V, C _L = 30 pF	-	22.5	-	
			V _{DD} = 2.4 V, C _L = 30 pF	-	12.8	-	
	T _R	Output low to high rising time	V _{DD} = 5.0 V, C _L = 30 pF	-	9.7	-	ns
			V _{DD} = 3.3 V, C _L = 30 pF	-	12.5	-	
			V _{DD} = 2.4 V, C _L = 30 pF	-	16.4	-	
	T _F	Output high to low falling time	V _{DD} = 5.0 V, C _L = 30 pF	-	6.6	-	ns
			V _{DD} = 3.3 V, C _L = 30 pF	-	9.0	-	
			V _{DD} = 2.4 V, C _L = 30 pF	-	12.3	-	
1	F _{OUT}	Maximum output frequency ^[1]	V _{DD} = 5.0 V, C _L = 30 pF	-	39	-	MHz
			V _{DD} = 3.3 V, C _L = 30 pF	-	27.5	-	
			V _{DD} = 2.4 V, C _L = 30 pF	-	17	-	
	T _R	Output low to high rising time	V _{DD} = 5.0 V, C _L = 30 pF	-	9.5	-	ns
			V _{DD} = 3.3 V, C _L = 30 pF	-	12.1	-	
			V _{DD} = 2.4 V, C _L = 30 pF	-	16	-	
	T _F	Output high to low falling time	V _{DD} = 5.0 V, C _L = 30 pF	-	4.7	-	ns
			V _{DD} = 3.3 V, C _L = 30 pF	-	6.2	-	
			V _{DD} = 2.4 V, C _L = 30 pF	-	8.3	-	

[1] Maximum output frequency is achieved if ((T_R + T_F) ≤) and if the duty cycle is 45% to 55%. e e figure below.

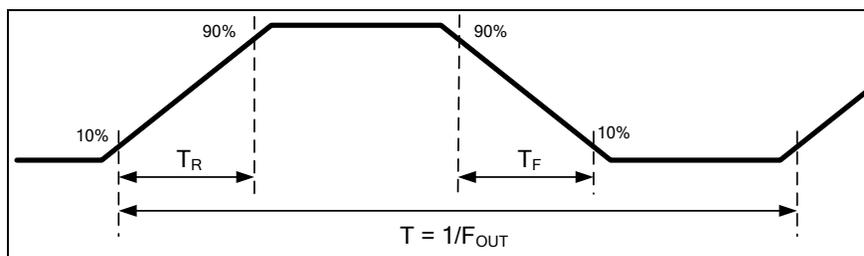


Figure 31-1. I/O A.C. Characteristics Definition

Table 31-3. Internal Oscillator A.C. Electrical Characteristics

Symbol	Parameter	Condition	Frequency Deviation	Min.	Typ.	Max.	Unit
F _{HIRC}	High-speed 16 MHz oscillator frequency (HIRC)	T _A = - 10°C ~ +70°C	± 1 %	15.84	16	16.16	MHz
		T _A = - 40°C ~ +105°C	± 2 %	15.68		16.32	
F _{LIRC}	Low-speed 10 kHz oscillator frequency (LIRC)	T _A = + 25 °C	± 10 %	9	10	11	kHz
		T _A = - 40 °C ~ +105 °C [1]	± 35 %	6.5		13.5	

[1] This value base on characterization results, not from product test.

Following shows LIRC value under full temperature condition:

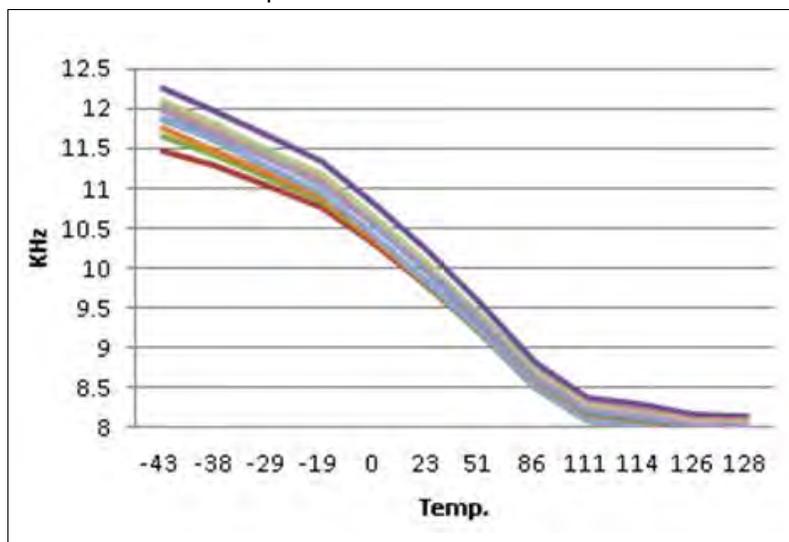


Figure 31-2 LIRC deviation under V_{DD} = 5.5 V

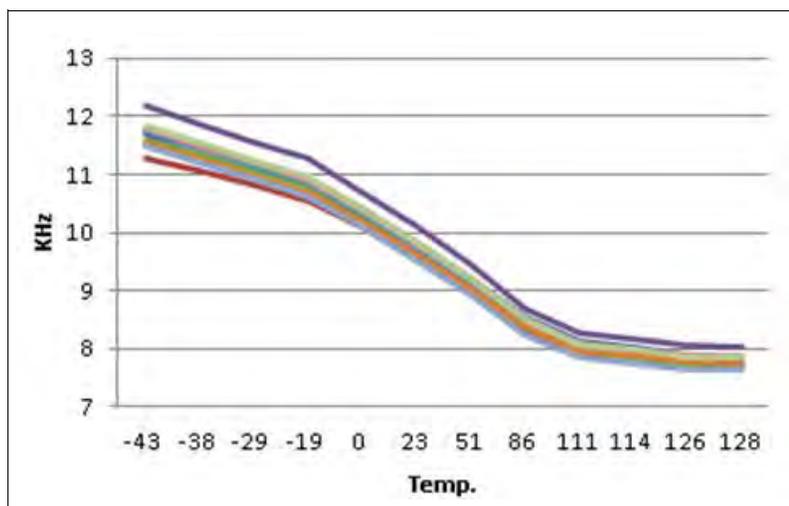


Figure 31-3 LIRC deviation under V_{DD} = 2.4 V

Following shows HIRC accuracy under full temperature condition:

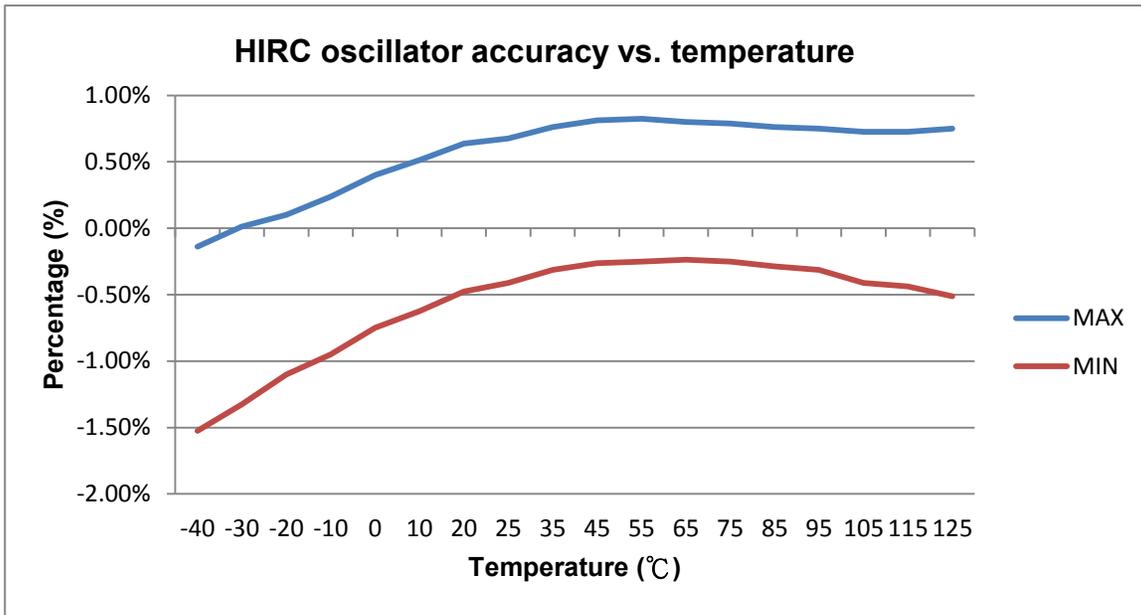


Figure 31-4 HIRC Accuracy vs. Temperature

Table 31-4. Power-Down Wake-Up A.C. Electrical Characteristics

Symbol	Parameter	Condition	Min.	Typ.	Max.	Unit
T _{PDWK}	Power-down wake-up time	HIRC	-	30	-	μs

Table 31-5. External Reset Pin A.C. Electrical Characteristics

Symbol	Parameter	Condition	Min.	Typ.	Max.	Unit
T _{RST}	pin de-bounce timing		-	24/F _{sys}	450	μs

31.4 Analog Electrical Characteristics

Table 31-6. POR Electrical Characteristics

Symbol	Parameter	Condition	Min.	Typ.	Max.	Unit
V _{POR}	Power-on reset voltage		1.3	1.4	1.5	V
T _{PORRD}	Power-on reset release delay		-	5.5	-	ms

Table 31-7. BOD Electrical Characteristics

Symbol	Parameter	Condition	Min.	Typ.	Max.	Unit
V _{BOD0}	Brown-out threshold 4.4V	BOV[1:0] = [0,0]	4.25	4.4	4.55	V
V _{BOD1}	Brown-out threshold 3.7V	BOV[1:0] = [0,1]	3.55	3.7	3.85	V
V _{BOD2}	Brown-out threshold 2.7V	BOV[1:0] = [1,0]	2.60	2.7	2.80	V
V _{BOD3}	Brown-out threshold 2.2V	BOV[1:0] = [1,1]	2.10	2.2	2.30	V
V _{BODHYS}	Brown-out hysteresis	LPBOD[1:0] = [0,0] only	-	50	200	mV
I _{BOD}	Brown-out quiescent current	V _{DD} = 5V, LPBOD[1:0] = [0,0]	-	147	184	μA
		V _{DD} = 5V, LPBOD[1:0] = [0,1]	-	19	24	
		V _{DD} = 5V, LPBOD[1:0] = [1,0]	-	5	7	
		V _{DD} = 5V, LPBOD[1:0] = [1,1]	-	3	4	
T _{BOD}	Brown-out detect pulse width		See Table 24-2			-
T _{BODEN}	Brown-out enable time		2	-	3	1/F _{LIRC}

Table 31-8. Band-gap Electrical Characteristics

Symbol	Parameter	Condition	Min.	Typ.	Max.	Unit
V _{BG}	Band-gap voltage	TA = + 25 °C (4%)	1.17	1.22	1.30	V
		TA = +10 °C ~ +85 °C ^[1]	1.14		1.33	
T _{BGEN}	Band-gap enable time		1	-	2	1/F _{LIRC}

[1] This value base on characterization results, not from product test.

Table 31-9. ADC Electrical Characteristics

Symbol	Parameter	Condition	Min.	Typ.	Max.	Unit
V _{AVDD}	ADC operating voltage		2.7	-	5.5	V
I _{AVDD}	ADC power supply current	V _{AVDD} = 5V	-	300	-	μA
V _{AIN}	Analog input voltage		0	-	V _{AVDD}	V
N _R	Resolution		12			bit
DNL	Differential non-linearity error		-	4	-	LSB
INL	Integral non-linearity error		-	±3	-	LSB
OE	Offset error		-	±2	-	LSB
TUE	Total un-adjust error		-	8	-	LSB
-	Monotonicity		Guaranteed			-
F _C	Conversion rate		500			ksps
F _S	Sampling rate ^[1]		380 ^[1]			ksps
T _{ADCEEN}	ADC enable time			15		μs
C _{IN}	ADC input equivalent capacitor			3.6		pF

[1] This value base on code polling flag. If call interrupt subroutine need add entry and exit interrupt timing , the sampling rate is about 290 ksp.

31.5 ESD Characteristics

Table 31-10 ESD Characteristics

Symbol	Ratings	Condition	Package	Maximum Value	Unit
V _{ESD}	Electrostatic discharge (Human body mode)	TA = + 25 °C	TSSOP 20 QFN 20	7000	V
	Electrostatic discharge (Machine mode)			400	V
	Electrostatic discharge (Device Charged mode)			1000	V

31.6 EFT Characteristics

Table 31-11 EFT Characteristics

Symbol	Condition		Package	Pass level	Unit
	Fsys	BOD			
	HIRC	Disable / Enable	TSSOP 20 QFN 20	± 4400	V

31.7 Flash DC Electrical Characteristics

Table 31-12 Flash DC Electrical Characteristics

Symbol	Parameter	Min.	Typ.	Max.	Unit	Condition
V _{FLA}	Supply Voltage	1.62	1.8	1.98	V	
N _{ENDUR}	Endurance	100,000	-	-	cycle	
		-	50	-	year	TA = + 25 °C
T _{RET}	Data Retention	-	40	-	year	TA = + 55 °C
			10	-	year	TA = + 85 °C
T _{ERASE}	Page Erase Time	-	5	-	ms	1 page
T _{PROG}	Program Time	-	10	-	us	1 byte
I _{DD1}	Read Current	-	4	-	mA	
I _{DD2}	Program Current	-	4	-	mA	
I _{DD3}	Erase Current	-	2	-	mA	

32. PACKAGE DIMENSIONS

32.1 20-pin TSSOP - 4.4 X 6.5 mm

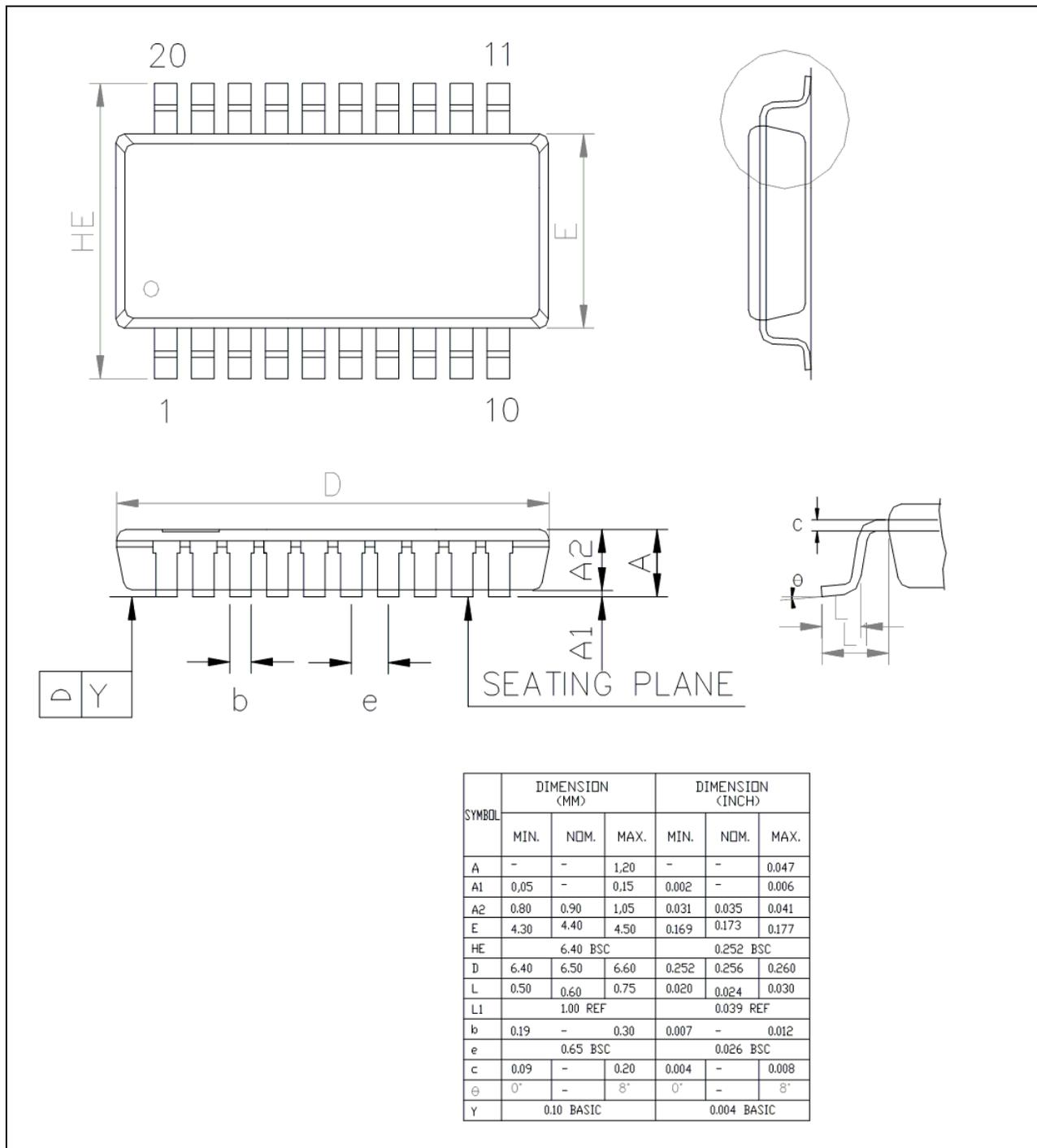


Figure 32-1. TSSOP-20 Package Dimension

32.2 20-pin QFN – 3.0 X 3.0 mm

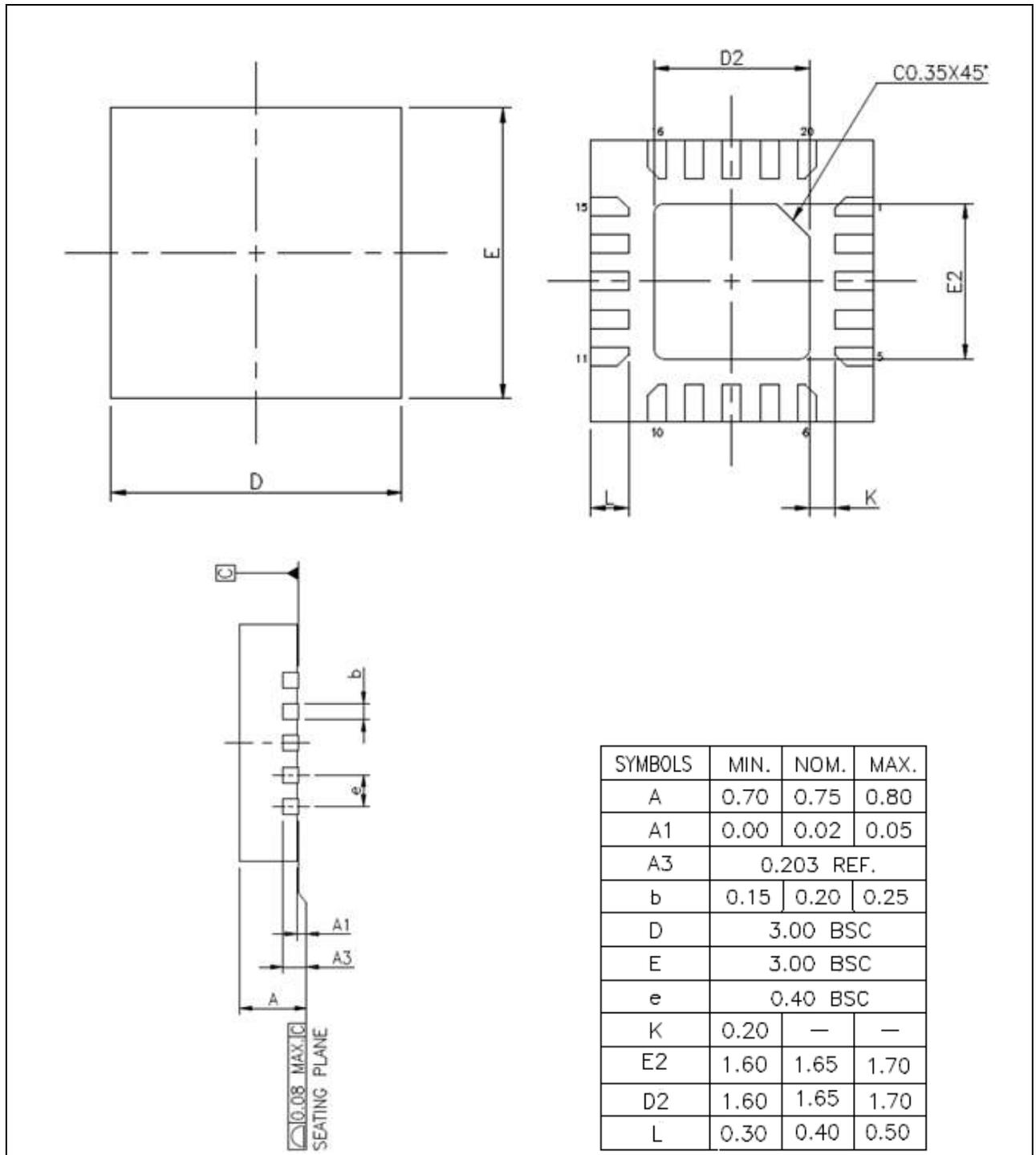


Figure 32-2. QFN-20 Package Dimension

33. DOCUMENT REVISION HISTORY

Revision	Date	Description
V1.00	2016/10/28	Initial release
V1.01	2017/6/12	Chapter 31.3 Added HIRC Accuracy vs. Temperature figure. Chapter 31.4 Modified band-gap max value from 1.27 to 1.30. Chapter 13.5 Modified SFR name to RCTRIM0 and RCTRIM1 for modify HIRC application. Chapter 24.1 Added disable POR function after power on description. Chapter 31.7 Modified data retention value at 25°C condition, added condition at 55°C and 85°C data.
V1.02	2017/6/26	Chapter 18.1.4 Added chapter description of band-gap as ADC input to calculate the VDD value. Chapter 15.3 Added how to clear SI register example code for I ² C transmission issue.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, “ nsecure Usage”.

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All nsecure Usage shall be made at customer’s risk, and in the event that third parties lay claims to Nuvoton as a result of customer’s nsecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

*Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.*