Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	PIC
Core Size	8-Bit
Speed	33MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	66
Program Memory Size	32KB (16K x 16)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	902 x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 5.5V
Data Converters	A/D 16x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	80-TQFP
Supplier Device Package	80-TQFP (12x12)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic17c766t-33i-pt

TO OUR VALUED CUSTOMERS

It is our intention to provide our valued customers with the best documentation possible to ensure successful use of your Microchip products. To this end, we will continue to improve our publications to better suit your needs. Our publications will be refined and enhanced as new volumes and updates are introduced.

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via E-mail at **docerrors@mail.microchip.com** or fax the **Reader Response Form** in the back of this data sheet to (480) 792-4150. We welcome your feedback.

Most Current Data Sheet

To obtain the most up-to-date version of this data sheet, please register at our Worldwide Web site at:

http://www.microchip.com

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number, (e.g., DS30000A is version A of document DS30000).

Errata

An errata sheet, describing minor operational differences from the data sheet and recommended workarounds, may exist for current devices. As device/documentation issues become known to us, we will publish an errata sheet. The errata will specify the revision of silicon and revision of document to which it applies.

To determine if an errata sheet exists for a particular device, please check with one of the following:

- Microchip's Worldwide Web site; http://www.microchip.com
- Your local Microchip sales office (see last page)
- The Microchip Corporate Literature Center; U.S. FAX: (480) 792-7277

When contacting a sales office or the literature center, please specify which device, revision of silicon and data sheet (include literature number) you are using.

Customer Notification System

Register on our web site at **www.microchip.com/cn** to receive the most current information on all of our products.

FIGURE 3-1: PIC17C752/756A BLOCK DIAGRAM

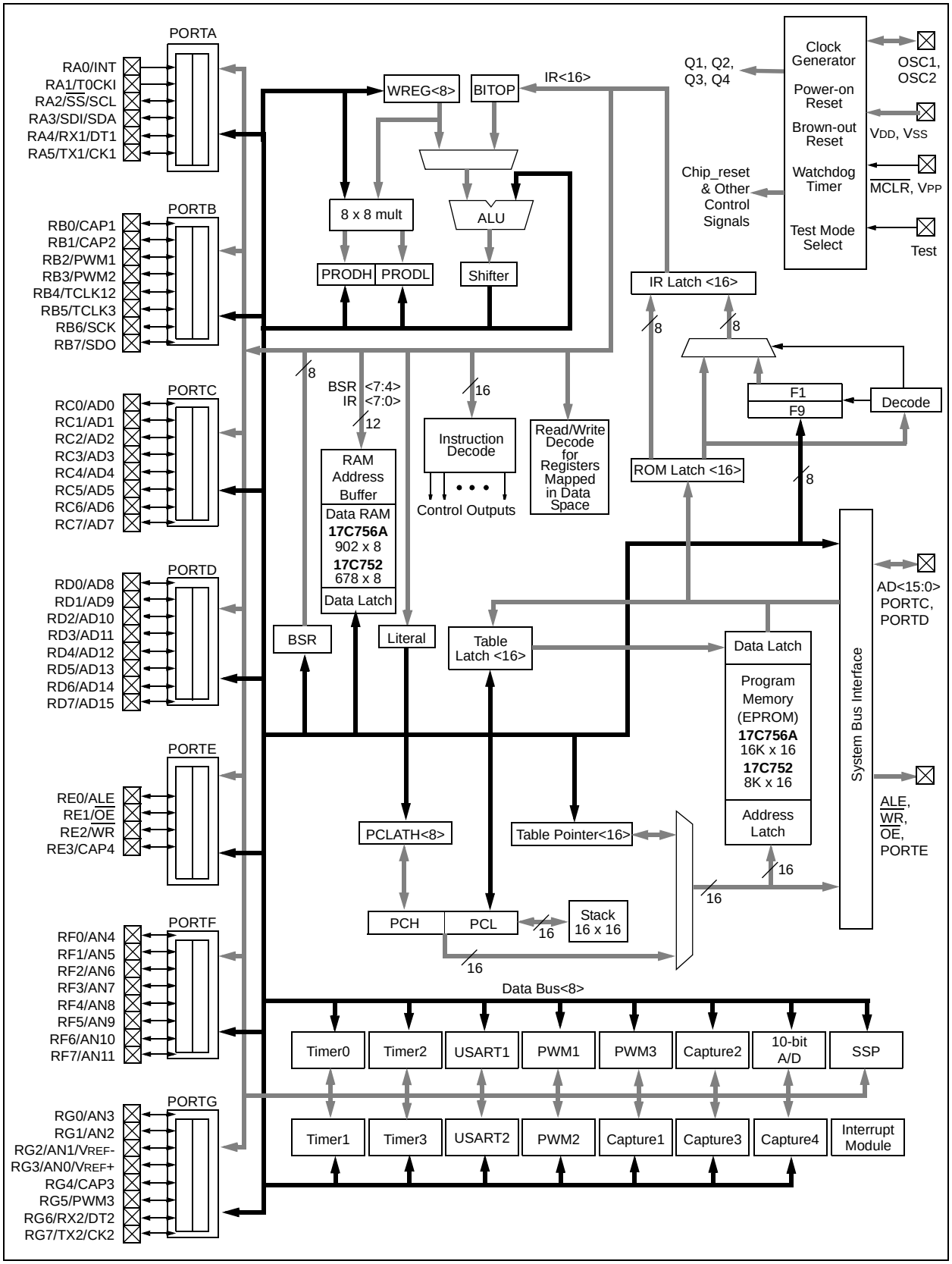


TABLE 5-4: INITIALIZATION CONDITIONS FOR SPECIAL FUNCTION REGISTERS

Register	Address	Power-on Reset Brown-out Reset	$\overline{\text{MCLR}}$ Reset WDT Reset	Wake-up from SLEEP through Interrupt
Unbanked				
INDF0	00h	N/A	N/A	N/A
FSR0	01h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	02h	0000h	0000h	PC + 1(2)
PCLATH	03h	0000 0000	uuuu uuuu	uuuu uuuu
ALUSTA	04h	1111 xxxx	1111 uuuu	1111 uuuu
T0STA	05h	0000 000-	0000 000-	0000 000-
CPUSTA(3)	06h	--11 11qq	--11 qquu	--uu qquu
INTSTA	07h	0000 0000	0000 0000	uuuu uuuu(1)
INDF1	08h	N/A	N/A	N/A
FSR1	09h	xxxx xxxx	uuuu uuuu	uuuu uuuu
WREG	0Ah	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR0L	0Bh	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR0H	0Ch	xxxx xxxx	uuuu uuuu	uuuu uuuu
TBLPTRL	0Dh	0000 0000	0000 0000	uuuu uuuu
TBLPTRH	0Eh	0000 0000	0000 0000	uuuu uuuu
BSR	0Fh	0000 0000	0000 0000	uuuu uuuu
Bank 0				
PORTA(4,6)	10h	0-xx 11xx	0-uu 11uu	u-uu uuuu
DDRB	11h	1111 1111	1111 1111	uuuu uuuu
PORTB(4)	12h	xxxx xxxx	uuuu uuuu	uuuu uuuu
RCSTA1	13h	0000 -00x	0000 -00u	uuuu -uuu
RCREG1	14h	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXSTA1	15h	0000 --1x	0000 --1u	uuuu --uu
TXREG1	16h	xxxx xxxx	uuuu uuuu	uuuu uuuu
SPBRG1	17h	0000 0000	0000 0000	uuuu uuuu

Legend: u = unchanged, x = unknown, - = unimplemented, read as '0', q = value depends on condition

Note 1: One or more bits in INTSTA, PIR1, PIR2 will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GLINTD bit is cleared, the PC is loaded with the interrupt vector.

3: See Table 5-3 for RESET value of specific condition.

4: This is the value that will be in the port output latch.

5: When the device is configured for Microprocessor or Extended Microcontroller mode, the operation of this port does not rely on these registers.

6: On any device RESET, these pins are configured as inputs.

6.3 Peripheral Interrupt Request Register1 (PIR1) and Register2 (PIR2)

These registers contains the individual flag bits for the peripheral interrupts.

Note: These bits will be set by the specified condition, even if the corresponding interrupt enable bit is cleared (interrupt disabled), or the GLINTD bit is set (all interrupts disabled). Before enabling an interrupt, the user may wish to clear the interrupt flag to ensure that the program does not immediately branch to the peripheral Interrupt Service Routine.

REGISTER 6-4: PIR1 REGISTER (ADDRESS: 16h, BANK 1)

R/W-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-1	R-0
RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TX1IF	RC1IF
bit 7							bit 0

bit 7 **RBIF**: PORTB Interrupt-on-Change Flag bit
1 = One of the PORTB inputs changed (software must end the mismatch condition)
0 = None of the PORTB inputs have changed

bit 6 **TMR3IF**: TMR3 Interrupt Flag bit

If Capture1 is enabled (CA1/$\overline{PR3}$ = 1):
1 = TMR3 overflowed
0 = TMR3 did not overflow

If Capture1 is disabled (CA1/$\overline{PR3}$ = 0):
1 = TMR3 value has rolled over to 0000h from equalling the period register (PR3H:PR3L) value
0 = TMR3 value has not rolled over to 0000h from equalling the period register (PR3H:PR3L) value

bit 5 **TMR2IF**: TMR2 Interrupt Flag bit
1 = TMR2 value has rolled over to 0000h from equalling the period register (PR2) value
0 = TMR2 value has not rolled over to 0000h from equalling the period register (PR2) value

bit 4 **TMR1IF**: TMR1 Interrupt Flag bit

If TMR1 is in 8-bit mode (T16 = 0):
1 = TMR1 value has rolled over to 0000h from equalling the period register (PR1) value
0 = TMR1 value has not rolled over to 0000h from equalling the period register (PR1) value

If Timer1 is in 16-bit mode (T16 = 1):
1 = TMR2:TMR1 value has rolled over to 0000h from equalling the period register (PR2:PR1) value
0 = TMR2:TMR1 value has not rolled over to 0000h from equalling the period register (PR2:PR1) value

bit 3 **CA2IF**: Capture2 Interrupt Flag bit
1 = Capture event occurred on RB1/CAP2 pin
0 = Capture event did not occur on RB1/CAP2 pin

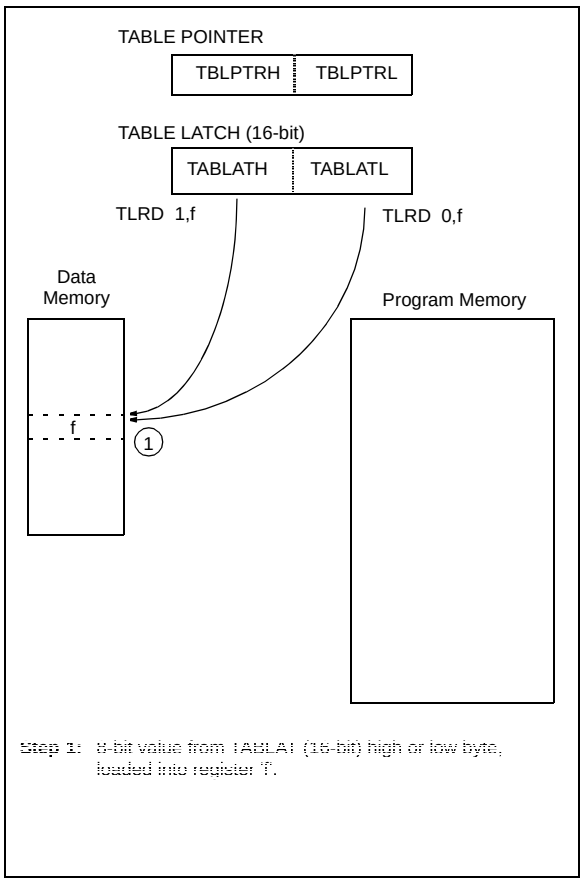
bit 2 **CA1IF**: Capture1 Interrupt Flag bit
1 = Capture event occurred on RB0/CAP1 pin
0 = Capture event did not occur on RB0/CAP1 pin

bit 1 **TX1IF**: USART1 Transmit Interrupt Flag bit (state controlled by hardware)
1 = USART1 Transmit buffer is empty
0 = USART1 Transmit buffer is full

bit 0 **RC1IF**: USART1 Receive Interrupt Flag bit (state controlled by hardware)
1 = USART1 Receive buffer is full
0 = USART1 Receive buffer is empty

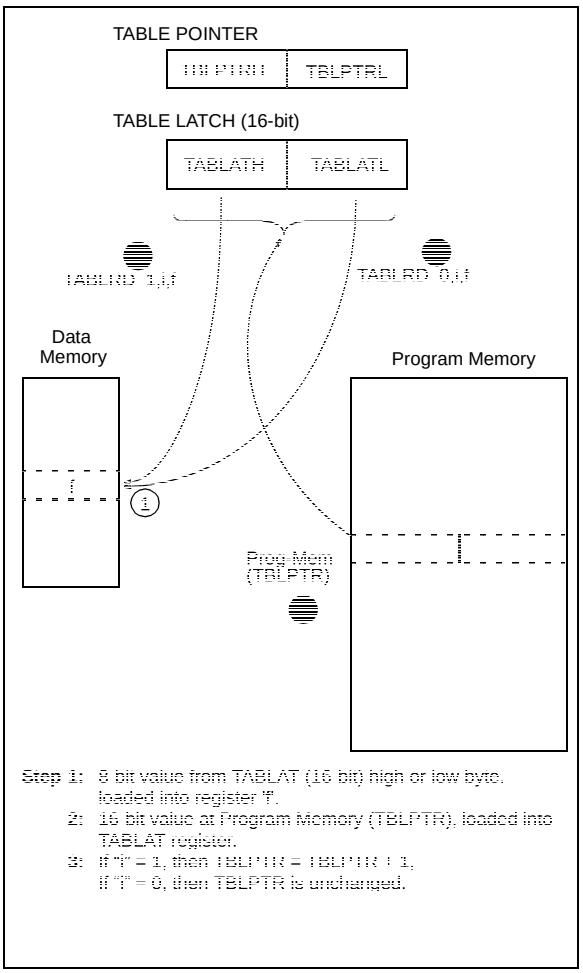
Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR Reset	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

FIGURE 8-3: `TLRD` INSTRUCTION OPERATION



Step 1: 8-bit value from TABLAT (16-bit) high or low byte, loaded into register 'f'.

FIGURE 8-4: `TABLRD` INSTRUCTION OPERATION



Step 1: 8 bit value from TABLAT (16 bit) high or low byte, loaded into register 'f'.
2: 16 bit value at Program Memory (TBLPTR), loaded into TABLAT register.
3: If "t" = 1, then TBLPTR = TBLPTR + 1,
If "t" = 0, then TBLPTR is unchanged.

10.3 PORTC and DDRC Registers

PORTC is an 8-bit bi-directional port. The corresponding data direction register is DDRC. A '1' in DDRC configures the corresponding port pin as an input. A '0' in the DDRC register configures the corresponding port pin as an output. Reading PORTC reads the status of the pins, whereas writing to PORTC will write to the port latch. PORTC is multiplexed with the system bus. When operating as the system bus, PORTC is the low order byte of the address/data bus (AD7:AD0). The timing for the system bus is shown in the Electrical Specifications section.

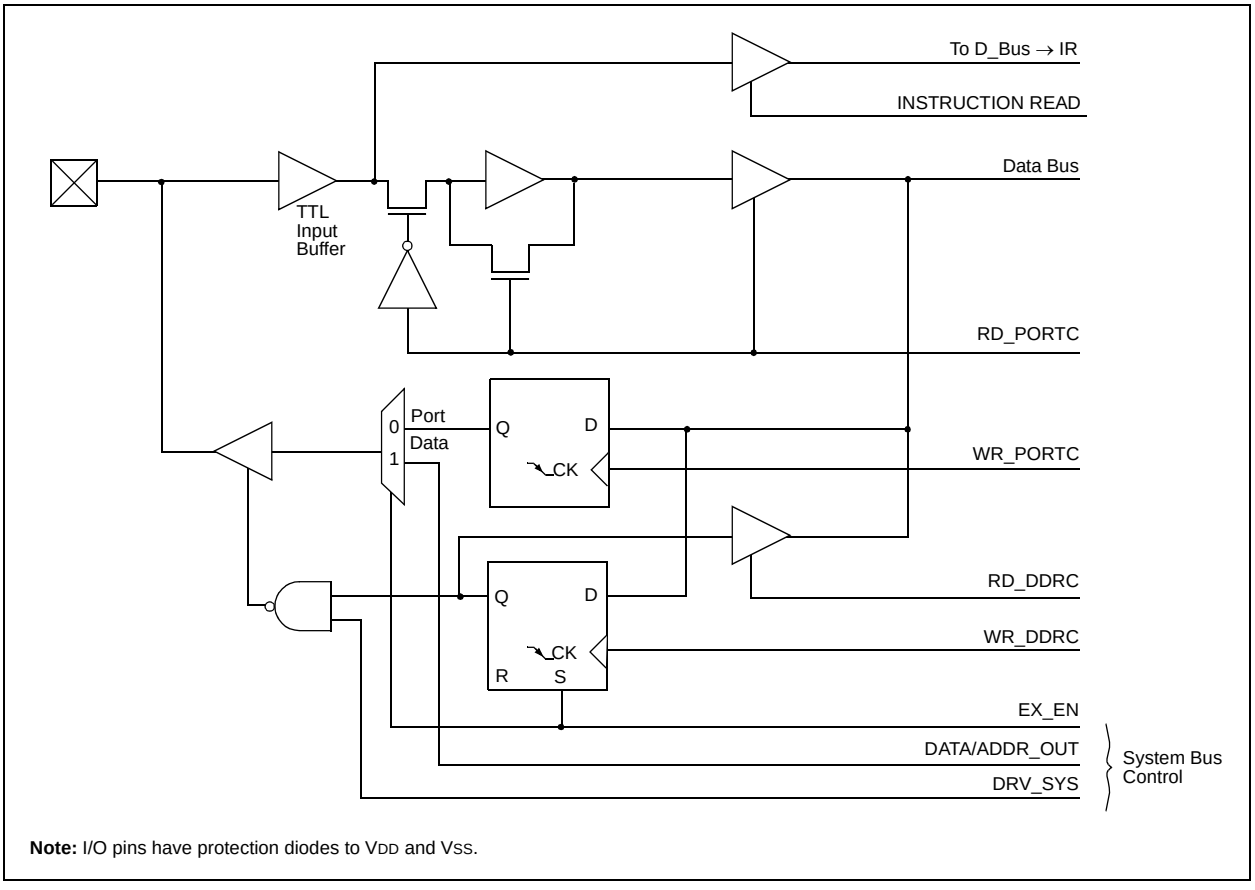
Note:	This port is configured as the system bus when the device's configuration bits are selected to Microprocessor or Extended Microcontroller modes. In the two other microcontroller modes, this port is a general purpose I/O.

Example 10-3 shows an instruction sequence to initialize PORTC. The Bank Select Register (BSR) must be selected to Bank 1 for the port to be initialized. The following example uses the `MOVLB` instruction to load the BSR register for bank selection.

EXAMPLE 10-3: INITIALIZING PORTC

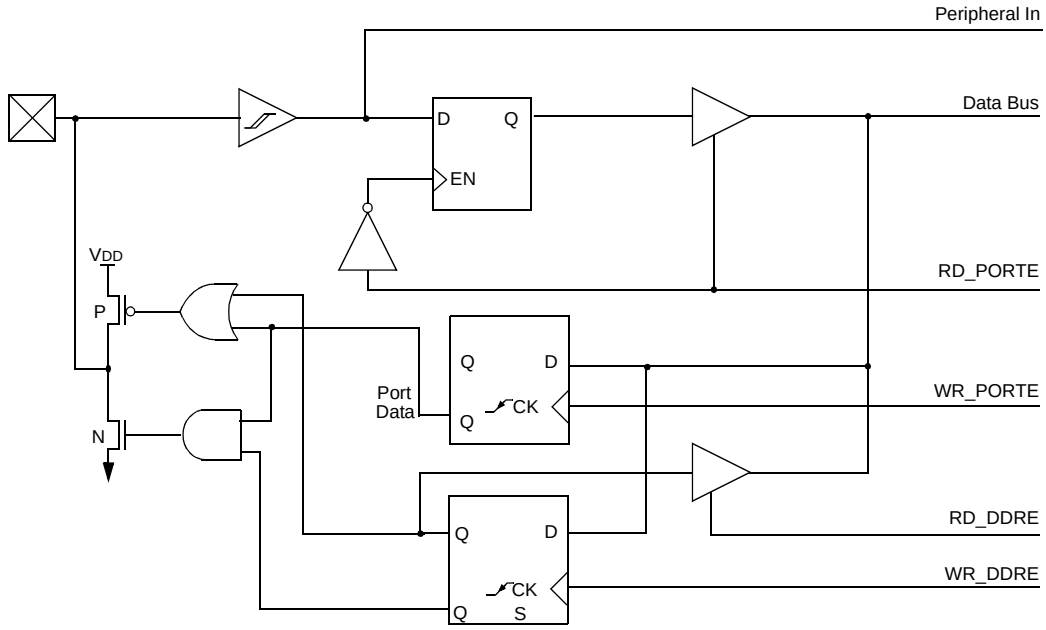
```
MOVLB   1          ; Select Bank 1
CLRF    PORTC, F   ; Initialize PORTC data
                   ; latches before setting
                   ; the data direction reg
MOVLW    0xCF      ; Value used to initialize
                   ; data direction
MOVWF    DDRC      ; Set RC<3:0> as inputs
                   ; RC<5:4> as outputs
                   ; RC<7:6> as inputs
```

FIGURE 10-9: BLOCK DIAGRAM OF RC7:RC0 PORT PINS



Note: I/O pins have protection diodes to VDD and VSS.

FIGURE 10-12: BLOCK DIAGRAM OF RE3/CAP4 PORT PIN



Note: I/O pin has protection diodes to VDD and VSS.

TABLE 10-9: PORTE FUNCTIONS

Name	Bit	Buffer Type	Function
RE0/$\overline{ALE}$	bit0	TTL	Input/output or system bus Address Latch Enable ($\overline{ALE}$) control pin.
RE1/$\overline{OE}$	bit1	TTL	Input/output or system bus Output Enable ($\overline{OE}$) control pin.
RE2/$\overline{WR}$	bit2	TTL	Input/output or system bus Write ($\overline{WR}$) control pin.
RE3/CAP4	bit3	ST	Input/output or Capture4 input pin.

Legend: TTL = TTL input, ST = Schmitt Trigger input

TABLE 10-10: REGISTERS/BITS ASSOCIATED WITH PORTE

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	$\overline{MCLR}$, WDT
15h, Bank 1	PORTE	—	—	—	—	RE3/CAP4	RE2/$\overline{WR}$	RE1/$\overline{OE}$	RE0/ALE	---- xxxx	---- uuuu
14h, Bank 1	DDRE	Data Direction Register for PORTE								---- 1111	---- 1111
14h, Bank 7	CA4L	Capture4 Low Byte								xxxx xxxx	uuuu uuuu
15h, Bank 7	CA4H	Capture4 High Byte								xxxx xxxx	uuuu uuuu
16h, Bank 7	TCON3	—	CA4OVF	CA3OVF	CA4ED1	CA4ED0	CA3ED1	CA3ED0	PWM3ON	-000 0000	-000 0000

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by PORTE.

15.2.1.1 Addressing

Once the MSSP module has been enabled, it waits for a START condition to occur. Following the START condition, the 8-bits are shifted into the SSPSR register. All incoming bits are sampled with the rising edge of the clock (SCL) line. The value of register SSPSR<7:1> is compared to the value of the SSPADD register. The address is compared on the falling edge of the eighth clock (SCL) pulse. If the addresses match and the BF and SSPOV bits are clear, the following events occur:

a) The SSPSR register value is loaded into the SSPBUF register on the falling edge of the 8th SCL pulse.
b) The buffer full bit, BF, is set on the falling edge of the 8th SCL pulse.
c) An $\overline{ACK}$ pulse is generated.
d) SSP interrupt flag bit, SSPIF (PIR2<7>), is set (interrupt is generated if enabled) - on the falling edge of the 9th SCL pulse.

In 10-bit address mode, two address bytes need to be received by the slave. The five Most Significant bits (MSbs) of the first address byte specify if this is a 10-bit address. Bit R/$\overline{W}$ (SSPSTAT<2>) must specify a write so the slave device will receive the second address byte. For a 10-bit address, the first byte would equal '1111 0 A9 A8 0', where A9 and A8 are the two MSbs of the address. The sequence of events for a 10-bit address is as follows, with steps 7- 9 for slave-transmitter:

1. Receive first (high) byte of Address (bits SSPIF, BF and bit UA (SSPSTAT<1>) are set).
2. Update the SSPADD register with second (low) byte of Address (clears bit UA and releases the SCL line).
3. Read the SSPBUF register (clears bit BF) and clear flag bit SSPIF.
4. Receive second (low) byte of Address (bits SSPIF, BF and UA are set).
5. Update the SSPADD register with the first (high) byte of Address. This will clear bit UA and release the SCL line.
6. Read the SSPBUF register (clears bit BF) and clear flag bit SSPIF.
7. Receive Repeated Start condition.
8. Receive first (high) byte of Address (bits SSPIF and BF are set).
9. Read the SSPBUF register (clears bit BF) and clear flag bit SSPIF.

Note:	Following the Repeated Start condition (step 7) in 10-bit mode, the user only needs to match the first 7-bit address. The user does not update the SSPADD for the second half of the address.

15.2.1.2 Slave Reception

When the R/$\overline{W}$ bit of the address byte is clear and an address match occurs, the R/$\overline{W}$ bit of the SSPSTAT register is cleared. The received address is loaded into the SSPBUF register.

When the address byte overflow condition exists, then no acknowledge ($\overline{ACK}$) pulse is given. An overflow condition is defined as either bit BF (SSPSTAT<0>) is set, or bit SSPOV (SSPCON1<6>) is set.

An SSP interrupt is generated for each data transfer byte. Flag bit SSPIF (PIR2<7>) must be cleared in software. The SSPSTAT register is used to determine the status of the received byte.

Note:	The SSPBUF will be loaded if the SSPOV bit is set and the BF flag is cleared. If a read of the SSPBUF was performed, but the user did not clear the state of the SSPOV bit before the next receive occurred, the $\overline{ACK}$ is not sent and the SSPBUF is updated.

TABLE 15-2: DATA TRANSFER RECEIVED BYTE ACTIONS

Status Bits as Data Transfer is Received		SSPSR → SSPBUF	Generate $\overline{ACK}$ Pulse	Set bit SSPIF (SSP Interrupt occurs if enabled)
BF	SSPOV			
0	0	Yes	Yes	Yes
1	0	No	No	Yes
1	1	No	No	Yes
0	1	Yes	No	Yes

Note 1: Shaded cells show the conditions where the user software did not properly clear the overflow condition.

FIGURE 15-14: I²C SLAVE-TRANSMITTER (10-BIT ADDRESS)

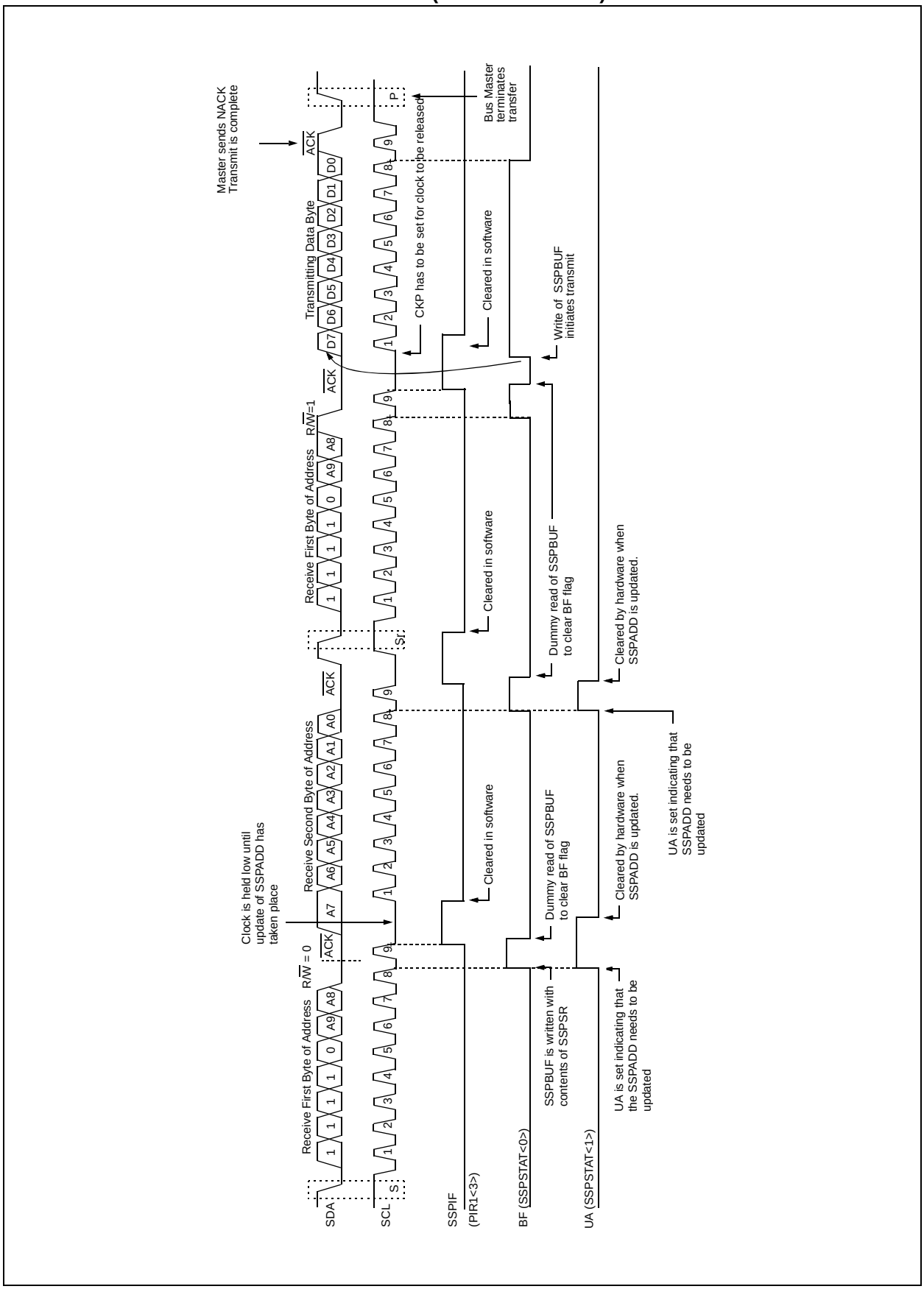


FIGURE 15-36: BUS COLLISION DURING START CONDITION (SCL = 0)

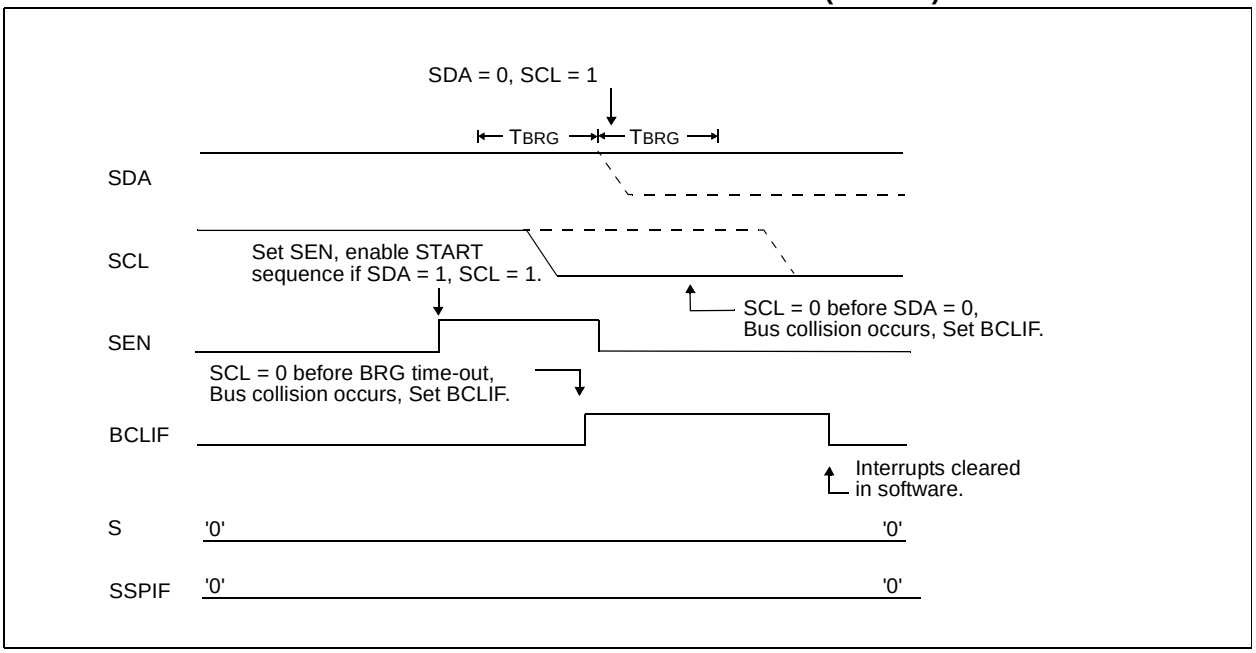
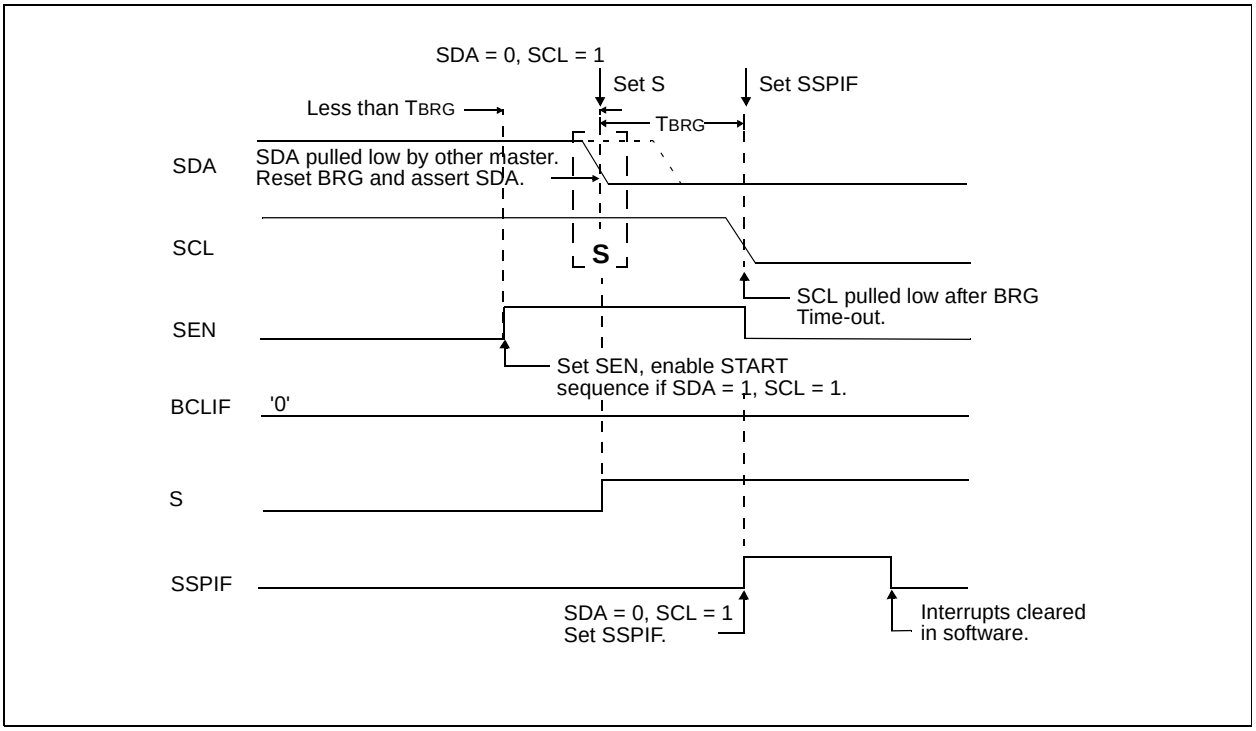


FIGURE 15-37: BRG RESET DUE TO SDA COLLISION DURING START CONDITION



17.4.2 MINIMIZING CURRENT CONSUMPTION

To minimize current consumption, all I/O pins should be either at VDD, or VSS, with no external circuitry drawing current from the I/O pin. I/O pins that are hi-impedance inputs should be pulled high or low externally to avoid switching currents caused by floating inputs. The T0CKI input should be at VDD or VSS. The contributions from on-chip pull-ups on PORTB should also be considered and disabled, when possible.

17.5 Code Protection

The code in the program memory can be protected by selecting the microcontroller in Code Protected mode (PM2:PM0 = '000').

In this mode, instructions that are in the on-chip program memory space, can continue to read or write the program memory. An instruction that is executed outside of the internal program memory range will be inhibited from writing to, or reading from, program memory.

Note:	Microchip does not recommend code protecting windowed devices.

If the code protection bit(s) have not been programmed, the on-chip program memory can be read out for verification purposes.

CPFSLT — **Compare f with WREG, skip if f < WREG**

Syntax: [*label*] CPFSLT f

Operands: $0 \le f \le 255$

Operation: (f) – (WREG), skip if (f) < (WREG) (unsigned comparison)

Status Affected: None

Encoding:

0011	0000	ffff	ffff

Description: Compares the contents of data memory location 'f' to the contents of WREG by performing an unsigned subtraction.
If the contents of 'f' are less than the contents of WREG, then the fetched instruction is discarded and a NOP is executed instead, making this a two-cycle instruction.

Words: 1

Cycles: 1 (2)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

Example:
```
HERE    CPFSLT REG
NLESS   :
LESS    :
```

Before Instruction
PC = Address (HERE)
W = ?

After Instruction
If REG < WREG;
PC = Address (LESS)
If REG ≥ WREG;
PC = Address (NLESS)

DAW — **Decimal Adjust WREG Register**

Syntax: [*label*] DAW f,s

Operands: $0 \le f \le 255$
$s \in [0,1]$

Operation:
If [[WREG<7:4> > 9].OR.[C = 1]].AND. [WREG<3:0> > 9]
then
WREG<7:4> + 7→ f<7:4>, s<7:4>;

If [WREG<7:4> > 9].OR.[C = 1]
then
WREG<7:4> + 6→ f<7:4>, s<7:4>;
else
WREG<7:4>→ f<7:4>, s<7:4>;

If [WREG<3:0> > 9].OR.[DC = 1]
then
WREG<3:0> + 6→ f<3:0>, s<3:0>;
else
WREG<3:0>→ f<3:0>, s<3:0>

Status Affected: C

Encoding:

0010	111s	ffff	ffff

Description: DAW adjusts the eight-bit value in WREG, resulting from the earlier addition of two variables (each in packed BCD format) and produces a correct packed BCD result.
s = 0: Result is placed in Data memory location 'f' and WREG.
s = 1: Result is placed in Data memory location 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f' and other specified register

Example:
```
DAW   REG1, 0
```

Before Instruction
WREG = 0xA5
REG1 = ??
C = 0
DC = 0

After Instruction
WREG = 0x05
REG1 = 0x05
C = 1
DC = 0

MOVFP **Move f to p**

Syntax: [*label*] MOVFP f,p

Operands: $0 \le f \le 255$
$0 \le p \le 31$

Operation: (f) → (p)

Status Affected: None

Encoding:

011p	pppp	ffff	ffff

Description: Move data from data memory location 'f' to data memory location 'p'. Location 'f' can be anywhere in the 256 byte data space (00h to FFh), while 'p' can be 00h to 1Fh.

Either 'p' or 'f' can be WREG (a useful, special situation).

MOVFP is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port). Both 'f' and 'p' can be indirectly addressed.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'p'

Example: MOVFP REG1, REG2

Before Instruction
REG1 = 0x33,
REG2 = 0x11

After Instruction
REG1 = 0x33,
REG2 = 0x33

MOVLB **Move Literal to low nibble in BSR**

Syntax: [*label*] MOVLB k

Operands: $0 \le k \le 15$

Operation: k → (BSR<3:0>)

Status Affected: None

Encoding:

1011	1000	uuuu	kkkk

Description: The four-bit literal 'k' is loaded in the Bank Select Register (BSR). Only the low 4-bits of the Bank Select Register are affected. The upper half of the BSR is unchanged. The assembler will encode the “u” fields as '0'.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write literal 'k' to BSR<3:0>

Example: MOVLB 5

Before Instruction
BSR register = 0x22

After Instruction
BSR register = 0x25 (Bank 5)

RETURN **Return from Subroutine**

Syntax: [*label*] RETURN

Operands: None

Operation: TOS → PC;

Status Affected: None

Encoding:

0000	0000	0000	0010

Description: Return from subroutine. The stack is popped and the top of the stack (TOS) is loaded into the program counter.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	Process Data	POP PC from stack
No operation	No operation	No operation	No operation

Example: `RETURN`

After Interrupt
PC = TOS

RLCF **Rotate Left f through Carry**

Syntax: [*label*] RLCF f,d

Operands: $0 \le f \le 255$
$d \in [0,1]$

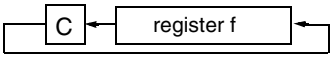
Operation: f<n> → d<n+1>;
f<7> → C;
C → d<0>

Status Affected: C

Encoding:

0001	101d	ffff	ffff

Description: The contents of register 'f' are rotated one bit to the left through the Carry Flag. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is stored back in register 'f'.



Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `RLCF REG,0`

Before Instruction
REG = 1110 0110
C = 0

After Instruction
REG = 1110 0110
WREG = 1100 1100
C = 1

19.8 MPLAB ICD In-Circuit Debugger

Microchip's In-Circuit Debugger, MPLAB ICD, is a powerful, low cost, run-time development tool. This tool is based on the FLASH PIC16F87X and can be used to develop for this and other PIC microcontrollers from the PIC16CXXX family. The MPLAB ICD utilizes the in-circuit debugging capability built into the PIC16F87X. This feature, along with Microchip's In-Circuit Serial Programming™ protocol, offers cost-effective in-circuit FLASH debugging from the graphical user interface of the MPLAB Integrated Development Environment. This enables a designer to develop and debug source code by watching variables, single-stepping and setting break points. Running at full speed enables testing hardware in real-time.

19.9 PRO MATE II Universal Device Programmer

The PRO MATE II universal device programmer is a full-featured programmer, capable of operating in stand-alone mode, as well as PC-hosted mode. The PRO MATE II device programmer is CE compliant.

The PRO MATE II device programmer has programmable V_{DD} and V_{PP} supplies, which allow it to verify programmed memory at V_{DD} min and V_{DD} max for maximum reliability. It has an LCD display for instructions and error messages, keys to enter commands and a modular detachable socket assembly to support various package types. In stand-alone mode, the PRO MATE II device programmer can read, verify, or program PIC MCU devices. It can also set code protection in this mode.

19.10 PICSTART Plus Entry Level Development Programmer

The PICSTART Plus development programmer is an easy-to-use, low cost, prototype programmer. It connects to the PC via a COM (RS-232) port. MPLAB Integrated Development Environment software makes using the programmer simple and efficient.

The PICSTART Plus development programmer supports all PIC devices with up to 40 pins. Larger pin count devices, such as the PIC16C92X and PIC17C76X, may be supported with an adapter socket. The PICSTART Plus development programmer is CE compliant.

19.11 PICDEM 1 Low Cost PIC MCU Demonstration Board

The PICDEM 1 demonstration board is a simple board which demonstrates the capabilities of several of Microchip's microcontrollers. The microcontrollers supported are: PIC16C5X (PIC16C54 to PIC16C58A), PIC16C61, PIC16C62X, PIC16C71, PIC16C8X, PIC17C42, PIC17C43 and PIC17C44. All necessary hardware and software is included to run basic demo programs. The user can program the sample microcontrollers provided with the PICDEM 1 demonstration board on a PRO MATE II device programmer, or a PICSTART Plus development programmer, and easily test firmware. The user can also connect the PICDEM 1 demonstration board to the MPLAB ICE in-circuit emulator and download the firmware to the emulator for testing. A prototype area is available for the user to build some additional hardware and connect it to the microcontroller socket(s). Some of the features include an RS-232 interface, a potentiometer for simulated analog input, push button switches and eight LEDs connected to PORTB.

19.12 PICDEM 2 Low Cost PIC16CXX Demonstration Board

The PICDEM 2 demonstration board is a simple demonstration board that supports the PIC16C62, PIC16C64, PIC16C65, PIC16C73 and PIC16C74 microcontrollers. All the necessary hardware and software is included to run the basic demonstration programs. The user can program the sample microcontrollers provided with the PICDEM 2 demonstration board on a PRO MATE II device programmer, or a PICSTART Plus development programmer, and easily test firmware. The MPLAB ICE in-circuit emulator may also be used with the PICDEM 2 demonstration board to test firmware. A prototype area has been provided to the user for adding additional hardware and connecting it to the microcontroller socket(s). Some of the features include a RS-232 interface, push button switches, a potentiometer for simulated analog input, a serial EEPROM to demonstrate usage of the I^2C™ bus and separate headers for connection to an LCD module and a keypad.

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +125°C for extended -40°C ≤ TA ≤ +85°C for industrial 0°C ≤ TA ≤ +70°C for commercial Operating voltage VDD range as described in Section 20.1				
Param. No.	**Sym**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
D150	VOD	**Open Drain High Voltage**	–	–	8.5	V	RA2 and RA3 pins only pulled up to externally applied voltage
		Capacitive Loading Specs on Output Pins					
D100	COSC2	OSC2/CLKOUT pin	–	–	25	pF	In EC or RC osc modes, when OSC2 pin is outputting CLKOUT. External clock is used to drive OSC1.
D101	CIO	All I/O pins and OSC2 (in RC mode)	–	–	50	pF	
D102	CAD	System Interface Bus (PORTC, PORTD and PORTE)	–	–	50	pF	In Microprocessor or Extended Microcontroller mode
		Internal Program Memory Programming Specs (Note 4)					
D110	VPP	Voltage on MCLR/VPP pin	12.75	–	13.25	V	**(Note 5)**
D111	VDDP	Supply voltage during programming	4.75	5.0	5.25	V	
D112	IPP	Current into MCLR/VPP pin	–	25	50	mA	
D113	IDDP	Supply current during programming	–	–	30	mA	
D114	TPROG	Programming pulse width	100	–	1000	ms	Terminated via internal/ external interrupt or a RESET

† Data in "Typ" column is at 5V, 25°C unless otherwise stated.

Note 1: In RC oscillator configuration, the OSC1/CLKIN pin is a Schmitt Trigger input. It is not recommended that the PIC17CXXX devices be driven with external clock in RC mode.

2: The leakage current on the MCLR pin is strongly dependent on the applied voltage level. Higher leakage current may be measured at different input voltages.

3: Negative current is defined as current sourced by the pin.

4: These specifications are for the programming of the on-chip program memory EPROM through the use of the table write instructions. The complete programming specifications can be found in: PIC17C7XX Programming Specifications (Literature number DS TBD).

5: The MCLR/VPP pin may be kept in this range at times other than programming, but is not recommended.

6: For TTL buffers, the better of the two specifications may be used.

Note 1: When using the Table Write for internal programming, the device temperature must be less than 40°C.

2: For In-Circuit Serial Programming (ICSP™), refer to the device programming specification.

20.3 Timing Parameter Symbology

The timing parameter symbols have been created following one of the following formats:

1. TppS2ppS
2. TppS
3. TCC:ST (I²C specifications only)
4. Ts (I²C specifications only)

T			
F	Frequency	T	Time

Lowercase symbols (pp) and their meanings:

pp			
ad	Address/Data	ost	Oscillator Start-Up Timer
al	ALE	pwrt	Power-Up Timer
cc	Capture1 and Capture2	rb	PORTB
ck	CLKOUT or clock	rd	$\overline{RD}$
dt	Data in	rw	$\overline{RD}$ or $\overline{WR}$
in	INT pin	t0	T0CKI
io	I/O port	t123	TCLK12 and TCLK3
mc	$\overline{MCLR}$	wdt	Watchdog Timer
oe	$\overline{OE}$	wr	$\overline{WR}$
os	OSC1		

Uppercase symbols and their meanings:

S			
D	Driven	L	Low
E	Edge	P	Period
F	Fall	R	Rise
H	High	V	Valid
I	Invalid (Hi-impedance)	Z	Hi-impedance

FIGURE 21-11: TYPICAL AND MAXIMUM IPD vs. VDD (SLEEP MODE, ALL PERIPHERALS DISABLED, -40°C to +125°C)

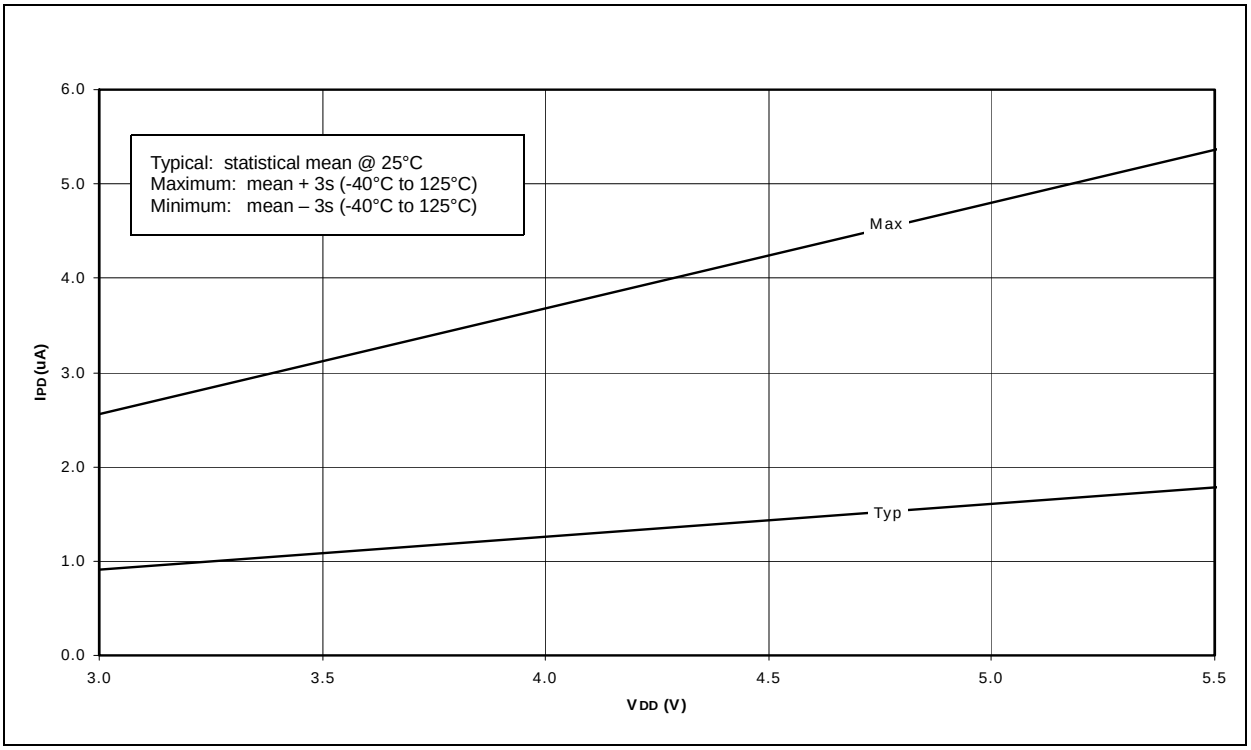
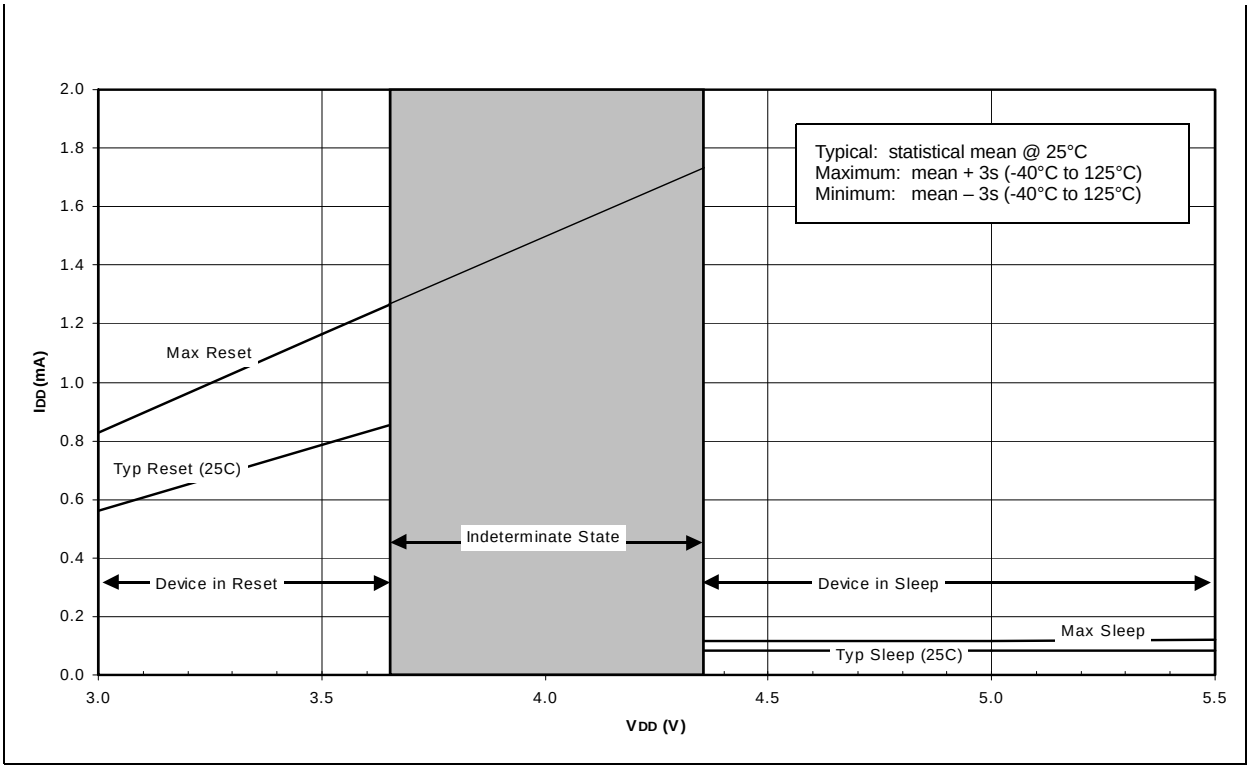


FIGURE 21-12: TYPICAL AND MAXIMUM IPD vs. VDD (SLEEP MODE, BOR ENABLED, -40°C to +125°C)



APPENDIX C: WHAT'S NEW

This is a new Data Sheet for the Following Devices:

- PIC17C752
- PIC17C756A
- PIC17C762
- PIC17C766

This Data Sheet is based on the PIC17C75X Data Sheet (DS30246A).

APPENDIX D: WHAT'S CHANGED

Clarified the TAD vs. device maximum operating frequency tables in Section 16.2.

Added device characteristic graphs and charts in Section 21.

Removed the "Preliminary" status from the entire document.

Revision C (January 2013)

Added a note to each package outline drawing.

INDEX