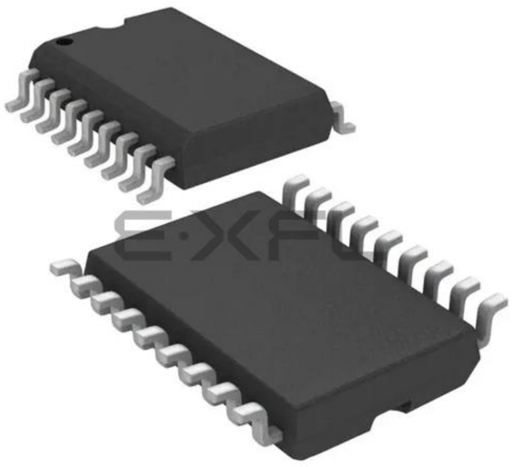


What is "Embedded - Microcontrollers"?

"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	4MHz
Connectivity	-
Peripherals	POR, WDT
Number of I/O	13
Program Memory Size	896B (512 x 14)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	80 x 8
Voltage - Supply (Vcc/Vdd)	2.5V ~ 5.5V
Data Converters	-
Oscillator Type	External
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	18-SOIC (0.295", 7.50mm Width)
Supplier Device Package	18-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16lc554-04e-so

1.0 GENERAL DESCRIPTION

The PIC16C55X are 18, 20 and 28-Pin EPROM-based members of the versatile PIC16CXX family of low cost, high performance, CMOS, fully-static, 8-bit microcontrollers.

All PIC® microcontrollers employ an advanced RISC architecture. The PIC16C55X have enhanced core features, eight-level deep stack, and multiple internal and external interrupt sources. The separate instruction and data buses of the Harvard architecture allow a 14-bit wide instruction word with the separate 8-bit wide data. The two-stage instruction pipeline allows all instructions to execute in a single-cycle, except for program branches (which require two cycles). A total of 35 instructions (reduced instruction set) are available. Additionally, a large register set gives some of the architectural innovations used to achieve a very high performance.

PIC16C55X microcontrollers typically achieve a 2:1 code compression and a 4:1 speed improvement over other 8-bit microcontrollers in their class.

The PIC16C554 has 80 bytes of RAM. The PIC16C557 and PIC16C558 have 128 bytes of RAM. The PIC16C554 and PIC16C558 have 13 I/O pins and an 8-bit timer/counter with an 8-bit programmable prescaler. The PIC16C557 has 22 I/O pins and an 8-bit timer/counter with an 8-bit programmable prescaler.

PIC16C55X devices have special features to reduce external components, thus reducing cost, enhancing system reliability and reducing power consumption. There are four oscillator options, of which the single pin RC oscillator provides a low cost solution, the LP oscillator minimizes power consumption, XT is a standard crystal, and the HS is for high speed crystals. The SLEEP (power-down) mode offers power saving. The user can wake-up the chip from SLEEP through several external and internal interrupts and RESET.

A highly reliable Watchdog Timer, with its own on-chip RC oscillator, provides protection against software lock-up.

A UV-erasable CERDIP packaged version is ideal for code development while the cost effective One-Time Programmable (OTP) version is suitable for production in any volume.

Table 1-1 shows the features of the PIC16C55X mid-range microcontroller families.

A simplified block diagram of the PIC16C55X is shown in Figure 3-1.

The PIC16C55X series fit perfectly in applications ranging from motor control to low power remote sensors. The EPROM technology makes customization of application programs (detection levels, pulse generation, timers, etc.) extremely fast and convenient. The small footprint packages make this microcontroller series perfect for all applications with space limitations. Low cost, low power, high performance, ease of use and I/O flexibility make the PIC16C55X very versatile.

1.1 Family and Upward Compatibility

Users familiar with the family of microcontrollers will realize that this is an enhanced version of the architecture. Please refer to Appendix A for a detailed list of enhancements. Code written for can be easily ported to PIC16C55X family of devices (Appendix B).

The PIC16C55X family fills the niche for users wanting to migrate up from the family and not needing various peripheral features of other members of the PIC16XX mid-range microcontroller family.

1.2 Development Support

The PIC16C55X family is supported by a full-featured macro assembler, a software simulator, an in-circuit emulator, a low cost development programmer and a full-featured programmer.

2.0 PIC16C55X DEVICE VARIETIES

A variety of frequency ranges and packaging options are available. Depending on application and production requirements, the proper device option can be selected using the information in the PIC16C55X Product Identification System section at the end of this data sheet. When placing orders, please use this page of the data sheet to specify the correct part number.

2.1 UV Erasable Devices

The UV erasable version, offered in CERDIP package, is optimal for prototype development and pilot programs. This version can be erased and reprogrammed to any of the oscillator modes.

Microchip's PICSTART® and PROMATE® programmers both support programming of the PIC16C55X.

2.2 One-Time Programmable (OTP) Devices

The availability of OTP devices is especially useful for customers who need the flexibility for frequent code updates and small volume applications. In addition to the program memory, the configuration bits must also be programmed.

2.3 Quick-Turnaround Production (QTP) Devices

Microchip offers a QTP Programming Service for factory production orders. This service is made available for users who choose not to program a medium-to-high quantity of units and whose code patterns have stabilized. The devices are identical to the OTP devices, but with all EPROM locations and configuration options already programmed by the factory. Certain code and prototype verification procedures apply before production shipments are available. Please contact your Microchip Technology sales office for more details.

2.4 Serialized Quick-Turnaround Production (SQTP^{SM}) Devices

Microchip offers a unique programming service where a few user-defined locations in each device are programmed with different serial numbers. The serial numbers may be random, pseudo-random or sequential.

Serial programming allows each device to have a unique number which can serve as an entry code, password or ID number.

3.1 Clocking Scheme/Instruction Cycle

The clock input (OSC1/CLKIN pin) is internally divided by four to generate four non-overlapping quadrature clocks namely Q1, Q2, Q3 and Q4. Internally, the program counter (PC) is incremented every Q1, the instruction is fetched from the program memory and latched into the instruction register in Q4. The instruction is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow are shown in Figure 3-2.

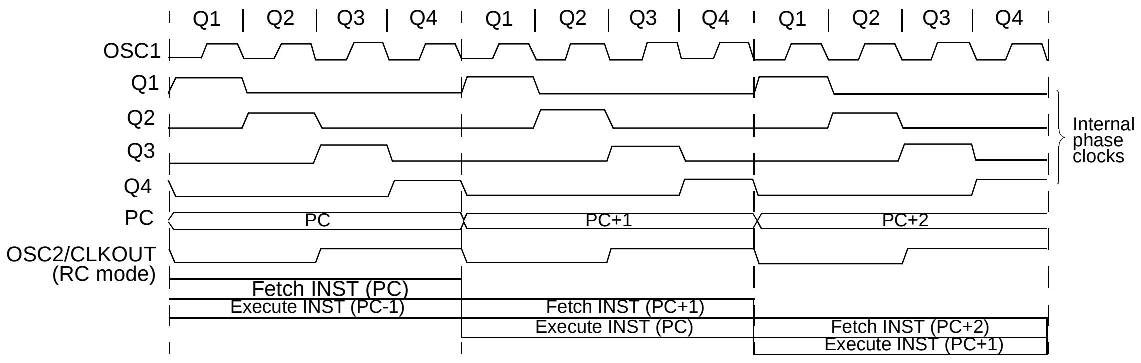
3.2 Instruction Flow/Pipelining

An “Instruction Cycle” consists of four Q cycles (Q1, Q2, Q3 and Q4). The instruction fetch and execute are pipelined such that fetch takes one instruction cycle while decode and execute takes another instruction cycle. However, due to the pipelining, each instruction effectively executes in one cycle. If an instruction causes the program counter to change (e.g., `GOTO`), then two cycles are required to complete the instruction (Example 3-1).

A fetch cycle begins with the program counter (PC) incrementing in Q1.

In the execution cycle, the fetched instruction is latched into the “Instruction Register (IR)” in cycle Q1. This instruction is then decoded and executed during the Q2, Q3, and Q4 cycles. Data memory is read during Q2 (operand read) and written during Q4 (destination write).

FIGURE 3-2: CLOCK/INSTRUCTION CYCLE



EXAMPLE 3-1: INSTRUCTION PIPELINE FLOW

`1. MOVLW 55h`	Fetch 1	Execute 1				
`2. MOVWF PORTB`		Fetch 2	Execute 2			
`3. CALL SUB_1`			Fetch 3	Execute 3		
`4. BSF PORTA, BIT3`				Fetch 4	Flush	
					Fetch SUB_1	Execute SUB_1

All instructions are single cycle, except for any program branches. These take two cycles since the fetch instruction is “flushed” from the pipeline while the new instruction is being fetched and then executed.

4.0 MEMORY ORGANIZATION

4.1 Program Memory Organization

The PIC16C55X has a 13-bit program counter capable of addressing an 8 K x 14 program memory space. Only the first 512 x 14 (0000h - 01FFh) for the PIC16C554 and 2K x 14 (0000h - 07FFh) for the PIC16C557 and PIC16C558 are physically implemented. Accessing a location above these boundaries will cause a wrap-around within the first 512 x 14 spaces in the PIC16C554, or 2K x 14 space of the PIC16C558 and PIC16C557. The RESET vector is at 0000h and the interrupt vector is at 0004h (Figure 4-1, Figure 4-2).

FIGURE 4-1: PROGRAM MEMORY MAP AND STACK FOR THE PIC16C554

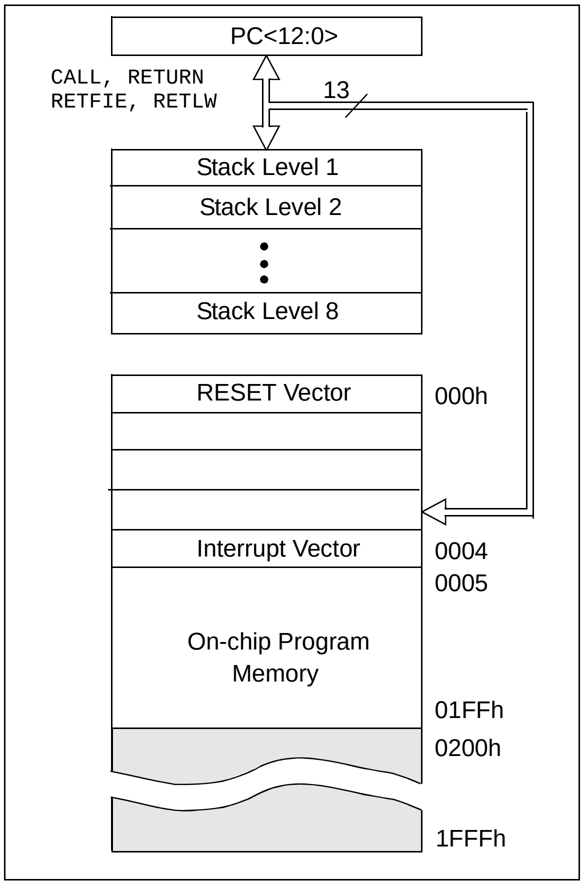
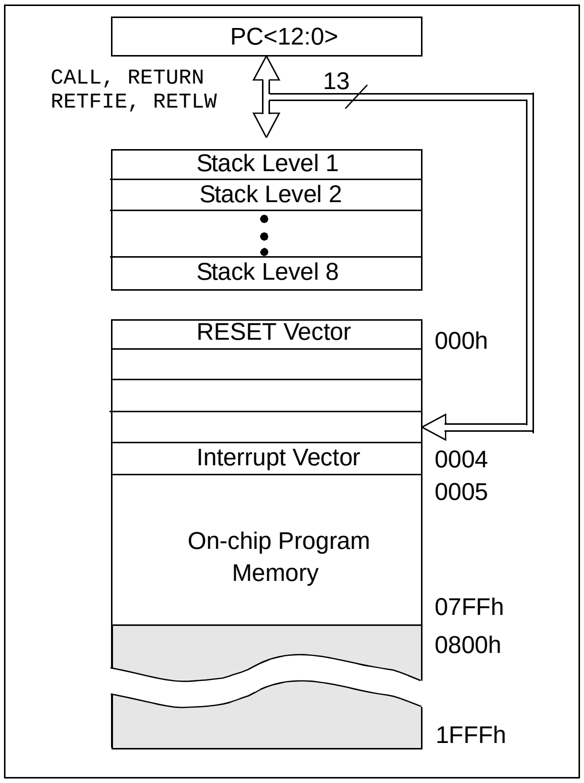


FIGURE 4-2: PROGRAM MEMORY MAP AND STACK FOR THE PIC16C557 AND PIC16C558



4.2 Data Memory Organization

The data memory (Figure 4-3 through Figure 4-5) is partitioned into two banks which contain the General Purpose Registers (GPR) and the Special Function Registers (SFR). Bank 0 is selected when the RP0 bit (STATUS <5>) is cleared. Bank 1 is selected when the RP0 bit is set. The Special Function Registers are located in the first 32 locations of each Bank. Register locations 20-6Fh (Bank 0) on the PIC16C554 and 20-7Fh (Bank 0) and A0-BFh (Bank 1) on the PIC16C558 and PIC16C557 are General Purpose Registers implemented as static RAM. Some special purpose registers are mapped in Bank 1.

4.2.1 GENERAL PURPOSE REGISTER FILE

The register file is organized as 80 x 8 in the PIC16C554 and 128 x 8 in the PIC16C557 and PIC16C558. Each can be accessed either directly or indirectly through the File Select Register, FSR (Section 4.4).

FIGURE 4-3: DATA MEMORY MAP FOR THE PIC16C554

File Address			File Address
00h	INDF(1)	INDF(1)	80h
01h	TMR0	OPTION	81h
02h	PCL	PCL	82h
03h	STATUS	STATUS	83h
04h	FSR	FSR	84h
05h	PORTA	TRISA	85h
06h	PORTB	TRISB	86h
07h	▨	▨	87h
08h	▨	▨	88h
09h	▨	▨	89h
0Ah	PCLATH	PCLATH	8Ah
0Bh	INTCON	INTCON	8Bh
0Ch	▨	▨	8Ch
0Dh	▨	▨	8Dh
0Eh	▨	PCON	8Eh
0Fh	▨	▨	8Fh
10h	▨	▨	90h
11h	▨	▨	91h
12h	▨	▨	92h
13h	▨	▨	93h
14h	▨	▨	94h
15h	▨	▨	95h
16h	▨	▨	96h
17h	▨	▨	97h
18h	▨	▨	98h
19h	▨	▨	99h
1Ah	▨	▨	9Ah
1Bh	▨	▨	9Bh
1Ch	▨	▨	9Ch
1Dh	▨	▨	9Dh
1Eh	▨	▨	9Eh
1Fh	▨	▨	9Fh
20h–6Fh	General Purpose Register	▨	A0h–FFh
70h–7Fh	▨	▨	
	Bank 0	Bank 1	

▨ Unimplemented data memory locations, read as '0'.

Note 1: Not a physical register.

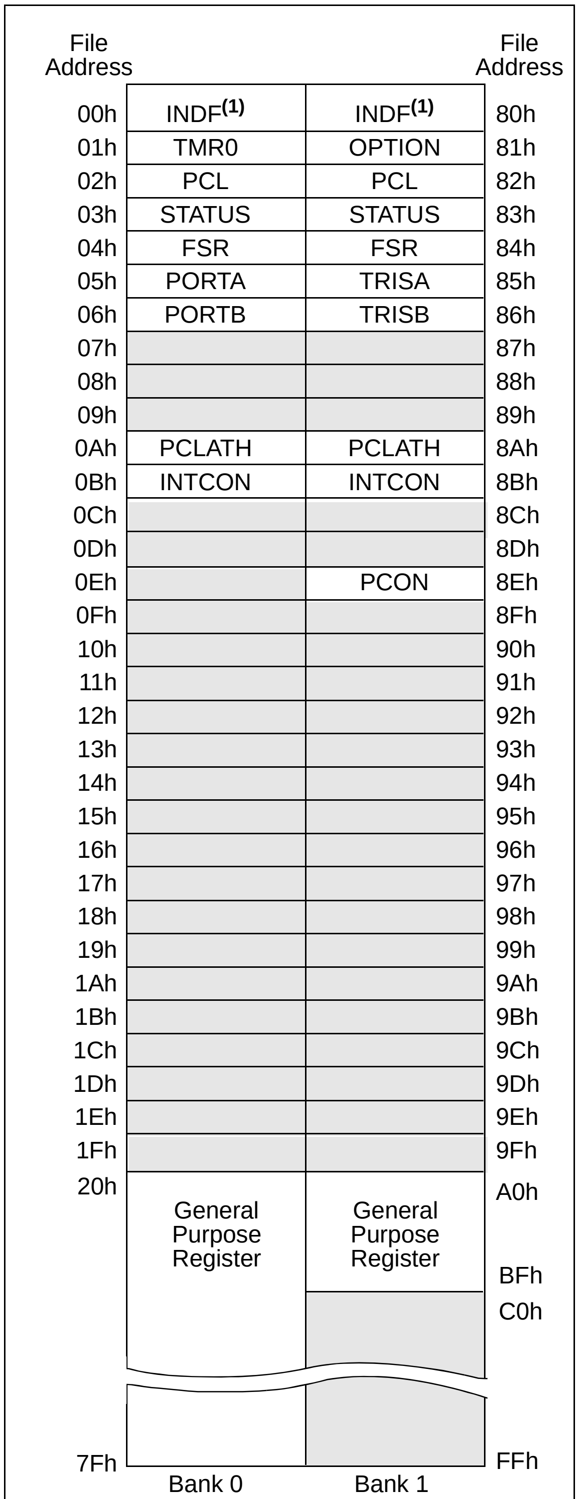
FIGURE 4-4: DATA MEMORY MAP FOR THE PIC16C557

File Address			File Address
00h	INDF(1)	INDF(1)	80h
01h	TMR0	OPTION	81h
02h	PCL	PCL	82h
03h	STATUS	STATUS	83h
04h	FSR	FSR	84h
05h	PORTA	TRISA	85h
06h	PORTB	TRISB	86h
07h	PORTC	TRISC	87h
08h	▨	▨	88h
09h	▨	▨	89h
0Ah	PCLATH	PCLATH	8Ah
0Bh	INTCON	INTCON	8Bh
0Ch	▨	▨	8Ch
0Dh	▨	▨	8Dh
0Eh	▨	PCON	8Eh
0Fh	▨	▨	8Fh
10h	▨	▨	90h
11h	▨	▨	91h
12h	▨	▨	92h
13h	▨	▨	93h
14h	▨	▨	94h
15h	▨	▨	95h
16h	▨	▨	96h
17h	▨	▨	97h
18h	▨	▨	98h
19h	▨	▨	99h
1Ah	▨	▨	9Ah
1Bh	▨	▨	9Bh
1Ch	▨	▨	9Ch
1Dh	▨	▨	9Dh
1Eh	▨	▨	9Eh
1Fh	▨	▨	9Fh
20h–7Fh	General Purpose Register	General Purpose Register	A0h–BFh
		▨	C0h–FFh
	Bank 0	Bank 1	

▨ Unimplemented data memory locations, read as '0'.

Note 1: Not a physical register.

FIGURE 4-5: DATA MEMORY MAP FOR THE PIC16C558



▨ Unimplemented data memory locations, read as '0'.

Note 1: Not a physical register.

4.2.2 SPECIAL FUNCTION REGISTERS

The Special Function Registers are registers used by the CPU and peripheral functions for controlling the desired operation of the device (Table 4-1). These registers are static RAM.

The Special Function Registers can be classified into two sets (core and peripheral). The special function registers associated with the “core” functions are described in this section. Those related to the operation of the peripheral features are described in the section of that peripheral feature.

4.2.2.1 STATUS Register

The STATUS register, shown in Figure 4-2, contains the arithmetic status of the ALU, the RESET status and the bank select bits for data memory.

The STATUS register can be the destination for any instruction, like any other register. If the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. These bits are set or cleared according to the device logic. Furthermore, the $\overline{\text{TO}}$ and $\overline{\text{PD}}$ bits are not writable. Therefore, the result of an instruction with the STATUS register as the destination may be different than intended.

For example, `CLRF STATUS` will clear the upper-three bits and set the Z bit. This leaves the STATUS register as `000uu1uu` (where u = unchanged).

It is recommended, therefore, that only `BCF`, `BSF`, `SWAPF` and `MOVWF` instructions be used to alter the STATUS register because these instructions do not affect any status bits. For other instructions, not affecting any status bits, see the “Instruction Set Summary”.

Note 1: The IRP and RP1 bits (STATUS<7:6>) are not used by the PIC16C55X and should be programmed as ’0'. Use of these bits as general purpose R/W bits is NOT recommended, since this may affect upward compatibility with future products.

2: The C and DC bits operate as a $\overline{\text{Borrow}}$ and Digit Borrow out bit, respectively, in subtraction. See the `SUBLW` and `SUBWF` instructions for examples.

REGISTER 4-1: STATUS REGISTER (ADDRESS 03h OR 83h)

Reserved	Reserved	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{\text{TO}}$	$\overline{\text{PD}}$	Z	DC	C
bit7							bit0

bit 7 **IRP**: Register Bank Select bit (used for Indirect addressing)
`1` = Bank 2, 3 (100h - 1FFh)
`0` = Bank 0, 1 (00h - FFh)
The IRP bit is reserved on the PIC16C55X, always maintain this bit clear

bit 6-5 **RP1:RP0**: Register Bank Select bits (used for Direct addressing)
`11` = Bank 3 (180h - 1FFh)
`10` = Bank 2 (100h - 17Fh)
`01` = Bank 1 (80h - FFh)
`00` = Bank 0 (00h - 7Fh)
Each bank is 128 bytes. The RP1 bit is reserved on the PIC16C55X, always maintain this bit clear.

bit 4 **$\overline{\text{TO}}$**: Timeout bit
`1` = After power-up, `CLRWDT` instruction, or `SLEEP` instruction
`0` = A WDT timeout occurred

bit 3 **$\overline{\text{PD}}$**: Power-down bit
`1` = After power-up or by the `CLRWDT` instruction
`0` = By execution of the `SLEEP` instruction

bit 2 **Z**: Zero bit
`1` = The result of an arithmetic or logic operation is zero
`0` = The result of an arithmetic or logic operation is not zero

bit 1 **DC**: Digit carry/$\overline{\text{borrow}}$ bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions) (for $\overline{\text{borrow}}$ the polarity is reversed)
`1` = A carry-out from the 4th low order bit of the result occurred
`0` = No carry-out from the 4th low order bit of the result

bit 0 **C**: Carry/$\overline{\text{borrow}}$ bit (`ADDWF`, `ADDLW`,`SUBLW`,`SUBWF` instructions)
`1` = A carry-out from the Most Significant bit of the result occurred
`0` = No carry-out from the Most Significant bit of the result occurred

Note 1: For $\overline{\text{borrow}}$ the polarity is reversed. A subtraction is executed by adding the two’s complement of the second operand. For rotate (RRF, RLF) instructions, this bit is loaded with either the high or low order bit of the source register.

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

EXAMPLE 5-1: READ-MODIFY-WRITE INSTRUCTIONS ON AN I/O PORT

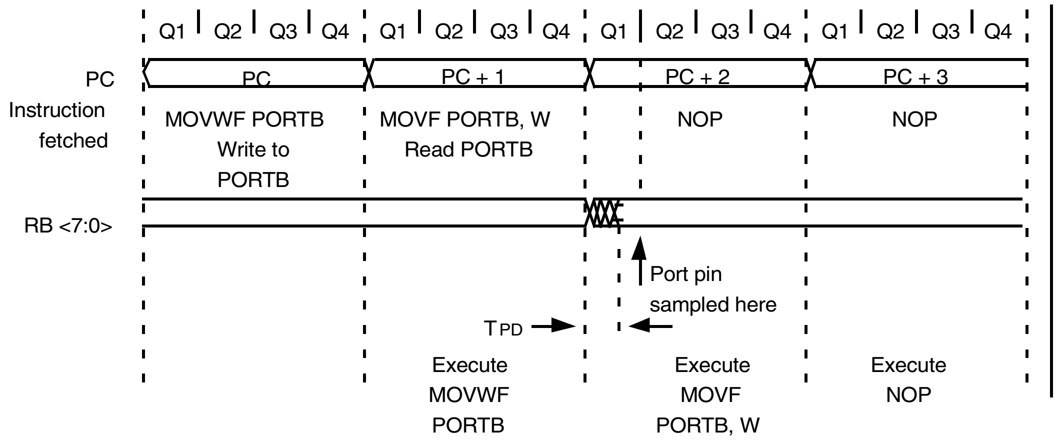
```
; Initial PORT settings: PORTB<7:4> Inputs
;
;                        PORTB<3:0> Outputs
; PORTB<7:6> have external pull-up and are
; not connected to other circuitry
;
;                      PORT latch PORT pins
;                      ---------- ---------
;

     BCF PORTB, 7    ; 01pp pppp  11pp pppp
     BCF PORTB, 6    ; 10pp pppp  11pp pppp
     BSF STATUS, RP0 ;
     BCF TRISB, 7    ; 10pp pppp  11pp pppp
     BCF TRISB, 6    ; 10pp pppp  10pp pppp
```

5.4.2 SUCCESSIVE OPERATIONS ON I/O PORTS

The actual write to an I/O port happens at the end of an instruction cycle, whereas for reading, the data must be valid at the beginning of the instruction cycle, as shown in Figure 5-6. Therefore, care must be exercised if a write followed by a read operation is carried out on the same I/O port. The sequence of instructions should be such to allow the pin voltage to stabilize (load dependent) before the next instruction which causes that file to be read into the CPU is executed. Otherwise, the previous state of that pin may be read into the CPU rather than the new state. When in doubt, it is better to separate these instructions with an `NOP` or another instruction not accessing this I/O port.

FIGURE 5-6: SUCCESSIVE I/O OPERATION



Note 1: This example shows write to PORTB followed by a read from PORTB.

2: Data setup time = (0.25 T_{CY} - T_{PD}) where T_{CY} = instruction cycle and T_{PD} = propagation delay of Q1 cycle to output valid. Therefore, at higher clock frequencies, a write followed by a read may be problematic.

REGISTER 6-1: CONFIGURATION WORD

CP1	CP0	CP1	CP0	CP1	CP0	—	Reserved	CP1	CP0	PWRTE	WDTE	F0SC1	F0SC0
bit 13													bit 0

bit 13-8 bit 5-4	**CP<1:0>**: Code protection bits[(1)] `11` = Program Memory code protection off `10` = 0400h - 07FFh code protected `01` = 0200h - 07FFh code protected `11` = 0000h - 07FFh code protected
bit 7	**Unimplemented**: Read as '1'
bit 6	**Reserved:** Do not use
bit 3	**$\overline{\text{PWRTE}}$**: Power-up Timer Enable bit `1` = PWRT disabled `0` = PWRT enabled
bit 2	**WDTE**: Watchdog Timer Enable bit `1` = WDT enabled `0` = WDT disabled
bit 1-0	**FOSC1:FOSC0**: Oscillator Selection bits `11` = RC oscillator `10` = HS oscillator `01` = XT oscillator `00` = LP oscillator

Note 1: All of the CP1:CP0 pairs have to be given the same value to enable the code protection scheme listed.

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

6.5.1 RB0/INT INTERRUPT

An external interrupt on RB0/INT pin is edge triggered: either rising if INTEDG bit (OPTION<6>) is set, or falling if INTEDG bit is clear. When a valid edge appears on the RB0/INT pin, the INTF bit (INTCON<1>) is set. This interrupt can be disabled by clearing the INTE control bit (INTCON<4>). The INTF bit must be cleared in software in the interrupt service routine before re-enabling this interrupt. The RB0/INT interrupt can wake-up the processor from SLEEP, if the INTE bit was set prior to going into SLEEP. The status of the GIE bit decides whether or not the processor branches to the interrupt vector following wake-up. See Section 6.8 for details on SLEEP and Figure 6-14 for timing of wake-up from SLEEP through RB0/INT interrupt.

6.5.2 TMR0 INTERRUPT

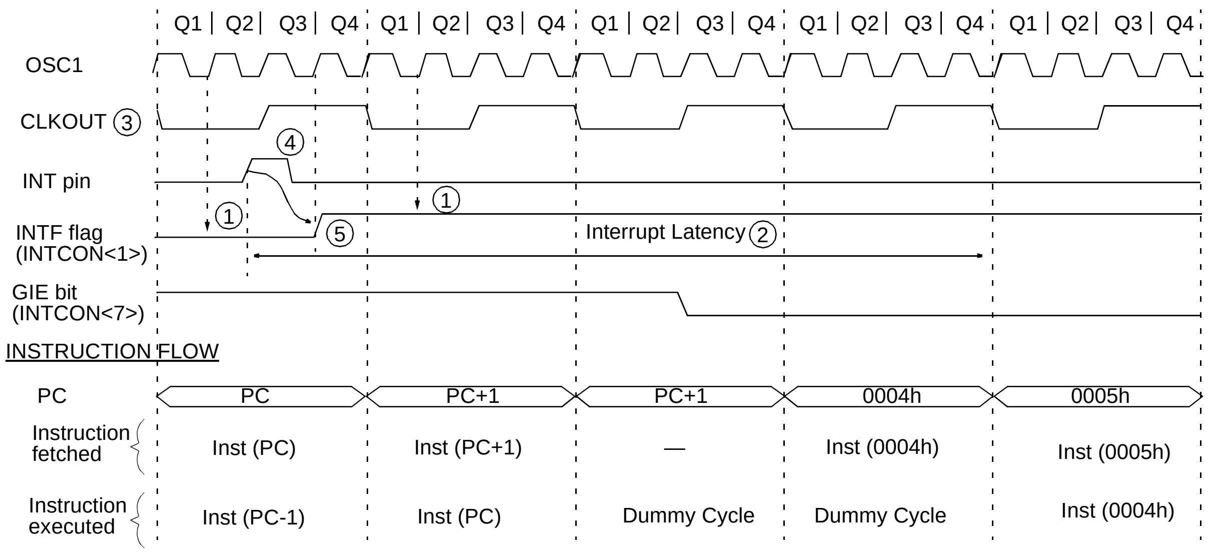
An overflow (FFh $\rightarrow$ 00h) in the TMR0 register will set the T0IF (INTCON<2>) bit. The interrupt can be enabled/disabled by setting/clearing T0IE (INTCON<5>) bit. For operation of the Timer0 module, see Section 7.0.

6.5.3 PORTB INTERRUPT

An input change on PORTB <7:4> sets the RBIF (INTCON<0>) bit. The interrupt can be enabled/disabled by setting/clearing the RBIE (INTCON<4>) bit. For operation of PORTB (Section 5.2).

Note:	If a change on the I/O pin should occur when the read operation is being executed (start of the Q2 cycle), then the RBIF interrupt flag may get set.

FIGURE 6-12: INT PIN INTERRUPT TIMING



Note 1: INTF flag is sampled here (every Q1).
2: Interrupt latency = 3-4 TCY where TCY = instruction cycle time. Latency is the same whether Inst (PC) is a single cycle or a 2-cycle instruction.
3: CLKOUT is available only in RC Oscillator mode.
4: For minimum width of INT pulse, refer to AC specs.
5: INTF is enabled to be set anytime during the Q4-Q1 cycles.

6.6 Context Saving During Interrupts

During an interrupt, only the return PC value is saved on the stack. Typically, users may wish to save key registers during an interrupt (e.g., W register and STATUS register). This will have to be implemented in software.

Example 6-1 stores and restores the STATUS and W registers. The user register, W_TEMP, must be defined in both banks and must be defined at the same offset from the bank base address (i.e., W_TEMP is defined at 0x20 in Bank 0 and it must also be defined at 0xA0 in Bank 1). The user register, STATUS_TEMP, must be defined in Bank 0. The Example 6-1:

- Stores the W register
- Stores the STATUS register in Bank 0
- Executes the ISR code
- Restores the STATUS (and bank select bit register)
- Restores the W register

EXAMPLE 6-1: SAVING THE STATUS AND W REGISTERS IN RAM

```
MOVWF  W_TEMP         ;copy W to TEMP
                      ;register, could be in
                      ;either bank
SWAPF  STATUS,W       ;swap STATUS to be
                      ;saved into W
BCF    STATUS,RP0     ;change to bank0
                      ;regardless of
                      ;current bank
MOVWF  STATUS_TEMP    ;save STATUS to bank0
                      ;register
:
:
:
SWAPF  STATUS_TEMP,W;swap STATUS_TEMP
                      ;register into W, sets
                      ;bank to original state
MOVWF  STATUS         ;move W into STATUS
                      ;register
SWAPF  W_TEMP,F       ;swap W_TEMP
SWAPF  W_TEMP,W       ;swap W_TEMP into W
```

6.7 Watchdog Timer (WDT)

The Watchdog Timer is a free running on-chip RC oscillator which does not require any external components. This RC oscillator is separate from the RC oscillator of the CLKIN pin. That means that the WDT will run, even if the clock on the OSC1 and OSC2 pins of the device has been stopped, for example, by execution of a SLEEP instruction. During normal operation, a WDT timeout generates a device RESET. If the device is in SLEEP mode, a WDT timeout causes the device to wake-up and continue with normal operation. The WDT can be permanently disabled by programming the configuration bit WDTE as clear (Section 6.1).

6.7.1 WDT PERIOD

The WDT has a nominal timeout period of 18 ms, (with no prescaler). The timeout periods vary with temperature, VDD and process variations from part-to-part (see DC specs). If longer timeout periods are desired, a prescaler with a division ratio of up to 1:128 can be assigned to the WDT under software control by writing to the OPTION register. Thus, timeout periods up to 2.3 seconds can be realized.

The CLRWDT and SLEEP instructions clear the WDT and the postscaler, if assigned to the WDT, and prevent it from timing out and generating a device RESET.

The $\overline{TO}$ bit in the STATUS register will be cleared upon a Watchdog Timer timeout.

6.7.2 WDT PROGRAMMING CONSIDERATIONS

It should also be taken in account that under worst case conditions (VDD = Min., Temperature = Max., max. WDT prescaler) it may take several seconds before a WDT timeout occurs.

RETFIE — Return from Interrupt

Syntax:	[*label*] RETFIE
Operands:	None
Operation:	TOS → PC, 1 → GIE
Status Affected:	None
Encoding:	`00` `0000` `0000` `1001`
Description:	Return from Interrupt. Stack is POPed and Top of Stack (TOS) is loaded in the PC. Interrupts are enabled by setting Global Interrupt Enable bit, GIE (INTCON<7>). This is a two-cycle instruction.
Words:	1
Cycles:	2
Example	`RETFIE`

After Interrupt

PC	=	TOS
GIE	=	1

RETURN — Return from Subroutine

Syntax:	[*label*] RETURN
Operands:	None
Operation:	TOS → PC
Status Affected:	None
Encoding:	`00` `0000` `0000` `1000`
Description:	Return from subroutine. The stack is POPed and the top of the stack (TOS) is loaded into the program counter. This is a two-cycle instruction.
Words:	1
Cycles:	2
Example	`RETURN`

After Interrupt

PC	=	TOS

RETLW — Return with Literal in W

Syntax:	[*label*] RETLW k
Operands:	$0 \leq k \leq 255$
Operation:	k → (W); TOS → PC
Status Affected:	None
Encoding:	`11` `01xx` `kkkk` `kkkk`
Description:	The W register is loaded with the eight bit literal 'k'. The program counter is loaded from the top of the stack (the return address). This is a two-cycle instruction.
Words:	1
Cycles:	2

Example

```
      CALL TABLE;W contains table
                ;offset value
      •         ;W now has table
      value
      •
      •
      ADDWF PC   ;W = offset
TABLE RETLW k1   ;Begin table
      RETLW k2   ;
      •
      •
      •
      RETLW kn   ; End of table
```

Before Instruction

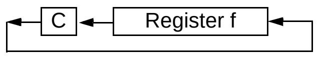
W	=	0x07

After Instruction

W	=	value of k8

RLF — Rotate Left f through Carry

Syntax:	[*label*] RLF f,d
Operands:	$0 \leq f \leq 127$ $d \in [0,1]$
Operation:	See description below
Status Affected:	C
Encoding:	`00` `1101` `dfff` `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 the W register. If 'd' is 1 the result is stored back in register 'f'.



Words:	1
Cycles:	1
Example	`RLF REG1,0`

Before Instruction

REG1	=	1110 0110
C	=	0

After Instruction

REG1	=	1110 0110
W	=	1100 1100
C	=	1

XORLW	Exclusive OR Literal with W
Syntax:	[*label*] XORLW k
Operands:	$0 \le k \le 255$
Operation:	(W) .XOR. k $\rightarrow$ (W)
Status Affected:	Z
Encoding:	`11` | `1010` | `kkkk` | `kkkk`
Description:	The contents of the W register are XOR'ed with the eight bit literal 'k'. The result is placed in the W register.
Words:	1
Cycles:	1

Example:	`XORLW 0xAF`
	Before Instruction
	W = 0xB5
	After Instruction
	W = 0x1A

XORWF	Exclusive OR W with f
Syntax:	[*label*] XORWF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(W) .XOR. (f) $\rightarrow$ (dest)
Status Affected:	Z
Encoding:	`00` | `0110` | `dfff` | `ffff`
Description:	Exclusive OR the contents of the W register with register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.
Words:	1
Cycles:	1

Example	`XORWF REG 1`
	Before Instruction
	REG = 0xAF
	W = 0xB5
	After Instruction
	REG = 0x1A
	W = 0xB5

TABLE 9-1: DEVELOPMENT TOOLS FROM MICROCHIP

		PIC12CXXX	PIC14000	PIC16C5X	PIC16C6X	PIC16CXXX	PIC16F62X	PIC16C7X	PIC16C7XX	PIC16C8X	PIC16F8XX	PIC16C9XX	PIC17C4X	PIC17C7XX	PIC18CXX2	PIC18FXXX	24CXX/ 25CXX/ 93CXX	HCSXXX	MCRFXXX	MCP2510
Software Tools	MPLAB® Integrated Development Environment	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓				
	MPLAB® C17 C Compiler												✓	✓						
	MPLAB® C18 C Compiler														✓	✓				
	MPASM™ Assembler/ MPLINK™ Object Linker	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
Emulators	MPLAB® ICE In-Circuit Emulator	✓	✓	✓	✓	✓	✓**	✓	✓	✓	✓	✓	✓	✓	✓	✓				
	ICEPIC™ In-Circuit Emulator	✓		✓	✓	✓		✓	✓	✓		✓								
Debugger	MPLAB® ICD In-Circuit Debugger				✓*			✓*			✓					✓				
Programmers	PICSTART® Plus Entry Level Development Programmer	✓	✓	✓	✓	✓	✓**	✓	✓	✓	✓	✓	✓	✓	✓	✓				
	PRO MATE® II Universal Device Programmer	✓	✓	✓	✓	✓	✓**	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
Demo Boards and Eval Kits	PICDEM™ 1 Demonstration Board			✓		✓		✓†		✓			✓							
	PICDEM™ 2 Demonstration Board				✓†			✓†							✓	✓				
	PICDEM™ 3 Demonstration Board											✓								
	PICDEM™ 14A Demonstration Board		✓																	
	PICDEM™ 17 Demonstration Board													✓						
	KEELOQ® Evaluation Kit																	✓		
	KEELOQ® Transponder Kit																	✓		
	microID™ Programmer's Kit																		✓	
	125 kHz microID™ Developer's Kit																		✓	
	125 kHz Anticollision microID™ Developer's Kit																		✓	
	13.56 MHz Anticollision microID™ Developer's Kit																		✓	
	MCP2510 CAN Developer's Kit																			✓

* Contact the Microchip Technology Inc. web site at www.microchip.com for information on how to use the MPLAB® ICD In-Circuit Debugger (DV164001) with PIC16C62, 63, 64, 65, 72, 73, 74, 76, 77.
** Contact Microchip Technology Inc. for availability date.
†

NOTES:

10.1 DC Characteristics: PIC16C55X-04 (Commercial, Industrial, Extended) PIC16C55X-20 (Commercial, Industrial, Extended) HCS1365-04 (Commercial, Industrial, Extended)

DC Characteristics			**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ T_A ≤ +85°C for industrial and 0°C ≤ T_A ≤ +70°C for commercial and -40°C ≤ T_A ≤ +125°C for extended				
Param No.	**Sym**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
D020	I_{PD}	**Power-Down Current**[3]					
		16LC55X	—	0.7	2	μA	V_{DD} = 3.0V, WDT disabled
		16C55X	—	1.0	2.5 15	μA μA	V_{DD} = 4.0V, WDT disabled (+85°C to +125°C)
	ΔI_{WDT}	**WDT Current**[5]					
		16LC55X	—	6.0	15	μA	V_{DD} = 3.0V
		16C55X	—	6.0	20	μA	V_{DD} = 4.0V (+85°C to +125°C)

* These parameters are characterized but not tested.

† Data is "Typ" column is at 5V, 25°C, unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: This is the limit to which V_{DD} can be lowered in SLEEP mode without losing RAM data.

2: The supply current is mainly a function of the operating voltage and frequency. Other factors such as I/O pin loading and switching rate, oscillator type, internal code execution pattern, and temperature also have an impact on the current consumption.
The test conditions for all I_{DD} measurements in active Operation mode are:
OSC1 = external square wave, from rail to rail; all I/O pins configured as input, pulled to V_{DD},
$\overline{\text{MCLR}}$ = V_{DD}; WDT enabled/disabled as specified.

3: The power-down current in SLEEP mode does not depend on the oscillator type. Power-down current is measured with the part in SLEEP mode, with all I/O pins configured as input and tied to V_{DD} or V_{SS}.

4: For RC osc configuration, current through R_{EXT} is not included. The current through the resistor can be estimated by the formula Ir = $V_{DD}/2R_{EXT}$ (mA) with R_{EXT} in kΩ.

5: The Δ current is the additional current consumed when this peripheral is enabled. This current should be added to the base I_{DD} or I_{PD} measurement.

10.2 DC Characteristics: PIC16C55X (Commercial, Industrial, Extended) PIC16LC55X(Commercial, Industrial, Extended) (Continued)

DC Characteristics	Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial and 0°C ≤ TA ≤ +70°C for commercial and -40°C ≤ TA ≤ +125°C for automotive Operating voltage VDD range as described in DC spec Table 10-1						
Param. No.	**Sym**	**Characteristic**	**Min**	**Typ†**	**Max**	**Unit**	**Conditions**
			VDD-0.7	—	—	V	IOH=-2.5 mA, VDD=4.5V, +125°C
D092		OSC2/CLKOUT	VDD-0.7	—	—	V	IOH=-1.3 mA, VDD=4.5V, -40° to +85°C
		(RC only)	VDD-0.7	—	—	V	IOH=-1.0 mA, VDD=4.5V, +125°C
*	VOD	**Open-Drain High Voltage**			10*	V	RA4 pin
		Capacitive Loading Specs on Output Pins					
D100	COSC 2	OSC2 pin			15	pF	In XT, HS and LP modes when external clock used to drive OSC1.
D101	CIO	All I/O pins/OSC2 (in RC mode)			50	pF	

* These parameters are characterized but not tested.

† Data in “Typ” column is at 5.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: In RC oscillator configuration, the OSC1 pin is a Schmitt Trigger input. It is not recommended that the PIC16C55X be driven with external clock in RC mode.

2: The leakage current on the $\overline{\text{MCLR}}$ pin is strongly dependent on applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

3: Negative current is defined as coming out of the pin.

NOTES:

NOTES:

READER RESPONSE

It is our intention to provide you with the best documentation possible to ensure successful use of your Microchip product. If you wish to provide your comments on organization, clarity, subject matter, and ways in which our documentation can better serve you, please FAX your comments to the Technical Publications Manager at (480) 792-4150.

Please list the following information, and use this outline to provide us with your comments about this document.

TO: Technical Publications Manager

RE: Reader Response

Total Pages Sent ________

From: Name ________________________________

Company ________________________________

Address ________________________________

City / State / ZIP / Country ________________________________

Telephone: (_______) _________ - _________ FAX: (______) _________ - _________

Application (optional):

Would you like a reply?___Y ___N

Device: Literature Number: DS40143E

Questions:

1. What are the best features of this document?

____

____

2. How does this document meet your hardware and software development needs?

____

____

3. Do you find the organization of this document easy to follow? If not, why?

____

____

4. What additions to the document do you think would enhance the structure and subject?

____

____

5. What deletions from the document could be made without affecting the overall usefulness?

____

____

6. Is there any incorrect or misleading information (what and where)?

____

____

7. How would you improve this document?

____

____