Microchip Technology - PIC16LC558-04I/SS Datasheet

Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	4MHz
Connectivity	-
Peripherals	POR, WDT
Number of I/O	13
Program Memory Size	3.5KB (2K x 14)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	128 x 8
Voltage - Supply (Vcc/Vdd)	2.5V ~ 5.5V
Data Converters	-
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	20-SSOP (0.209", 5.30mm Width)
Supplier Device Package	20-SSOP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16lc558-04i-ss

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℠) 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.

4.2.2.4 PCON Register

The PCON register contains a flag bit to differentiate between a Power-on Reset, an external $\overline{MCLR}$ Reset or WDT Reset. See Section 6.3 and Section 6.4 for detailed RESET operation.

REGISTER 4-4: PCON REGISTER (ADDRESS 8Eh)

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	U-0
—	—	—	—	—	—	$\overline{POR}$	—
bit7							bit0

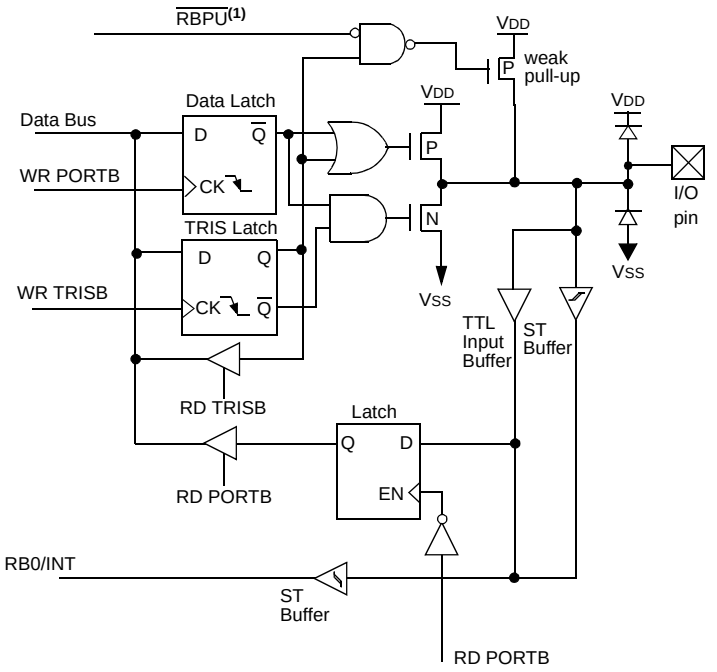
bit 7-2 **Unimplemented:** Read as '0'

bit 1 **$\overline{POR}$**: Power-on Reset status bit
1 = No Power-on Reset occurred
0 = Power-on Reset occurred

bit 0 **Unimplemented:** Read as '0'

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 5-4: BLOCK DIAGRAM OF RB3:RB0 PINS



Note 1: TRISB = 1 enables weak pull-up if $\overline{RBPU}$ = '0' (OPTION<7>).

TABLE 5-3: PORTB FUNCTIONS

Name	Bit #	Buffer Type	Function
RB0/INT	Bit 0	TTL/ST(1)	Bi-directional I/O port. Internal software programmable weak pull-up.
RB1	Bit 1	TTL	Bi-directional I/O port. Internal software programmable weak pull-up.
RB2	Bit 2	TTL	Bi-directional I/O port. Internal software programmable weak pull-up.
RB3	Bit 3	TTL	Bi-directional I/O port. Internal software programmable weak pull-up.
RB4	Bit 4	TTL	Bi-directional I/O port (with interrupt-on-change). Internal software programmable weak pull-up.
RB5	Bit 5	TTL	Bi-directional I/O port (with interrupt-on-change). Internal software programmable weak pull-up.
RB6	Bit 6	TTL/ST(2)	Bi-directional I/O port (with interrupt-on-change). Internal software programmable weak pull-up. Serial programming clock pin.
RB7	Bit 7	TTL/ST(2)	Bi-directional I/O port (with interrupt-on-change). Internal software programmable weak pull-up. Serial programming data pin.

Legend: ST = Schmitt Trigger, TTL = TTL input
Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.

TABLE 5-4: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB AND TRISB

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR	Value on All Other RESETS
06h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
86h	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111	1111 1111
81h	OPTION	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111
0BH, 8BH	INTCON	GIE	Reserved	T0IE	INTE	BRIE	T0IF	INTF	RBIF	0000 000x	0000 000x

Legend: x = unknown, u = unchanged
Note 1: Shaded bits are not used by PORTB.

6.3 RESET

The PIC16C55X differentiates between various kinds of RESET:

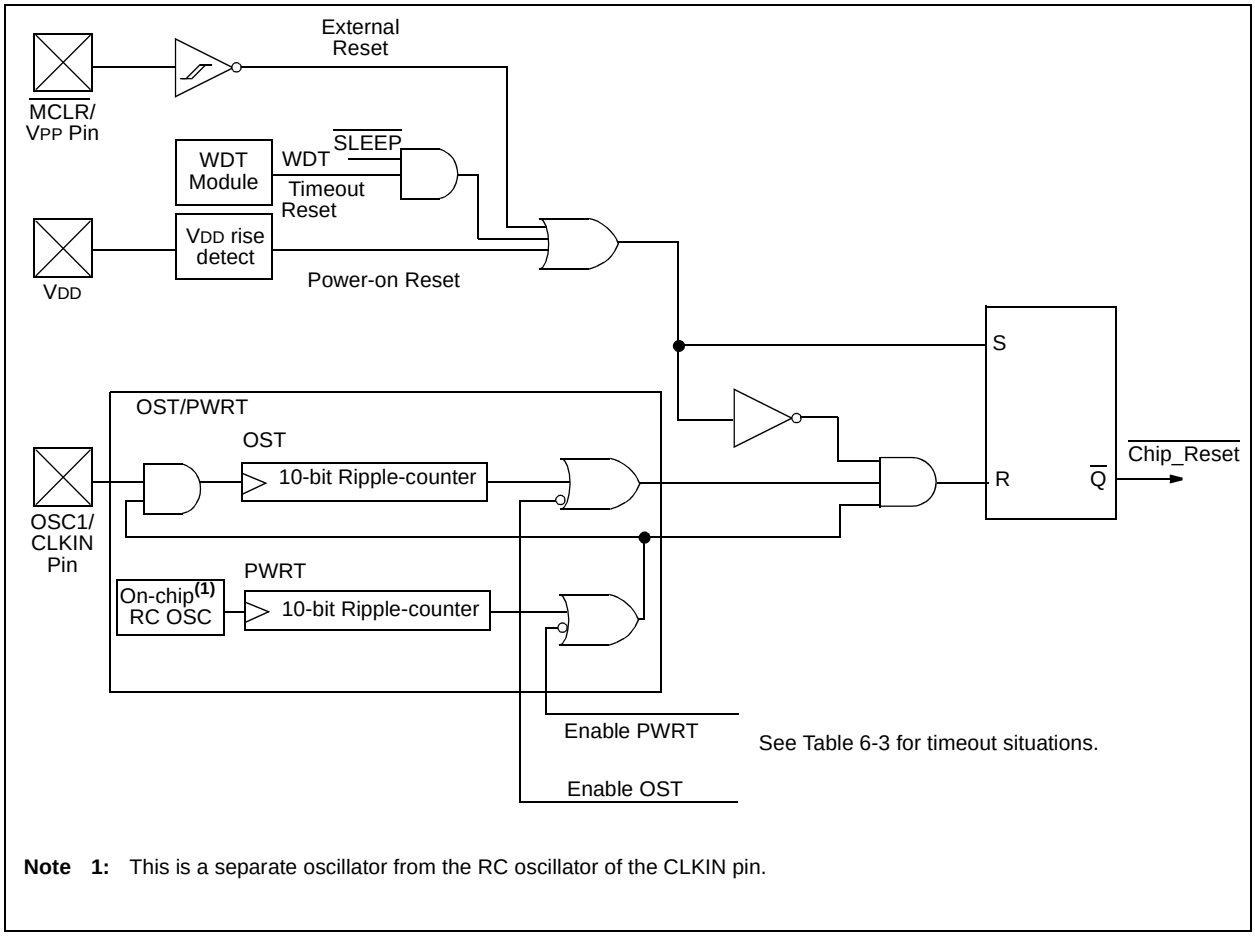
- Power-on Reset (POR)
- $\overline{MCLR}$ Reset during normal operation
- $\overline{MCLR}$ Reset during SLEEP
- WDT Reset (normal operation)
- WDT wake-up (SLEEP)

Some registers are not affected in any RESET condition; their status is unknown on POR and unchanged in any other RESET. Most other registers are reset to a "RESET state" on Power-on Reset, on $\overline{MCLR}$ or WDT Reset and on $\overline{MCLR}$ Reset during SLEEP. They are not affected by a WDT wake-up, since this is viewed as the resumption of normal operation. $\overline{TO}$ and $\overline{PD}$ bits are set or cleared differently in different RESET situations as indicated in Table 6-4. These bits are used in software to determine the nature of the RESET. See Table 6-6 for a full description of RESET states of all registers.

A simplified block diagram of the on-chip RESET circuit is shown in Figure 6-6.

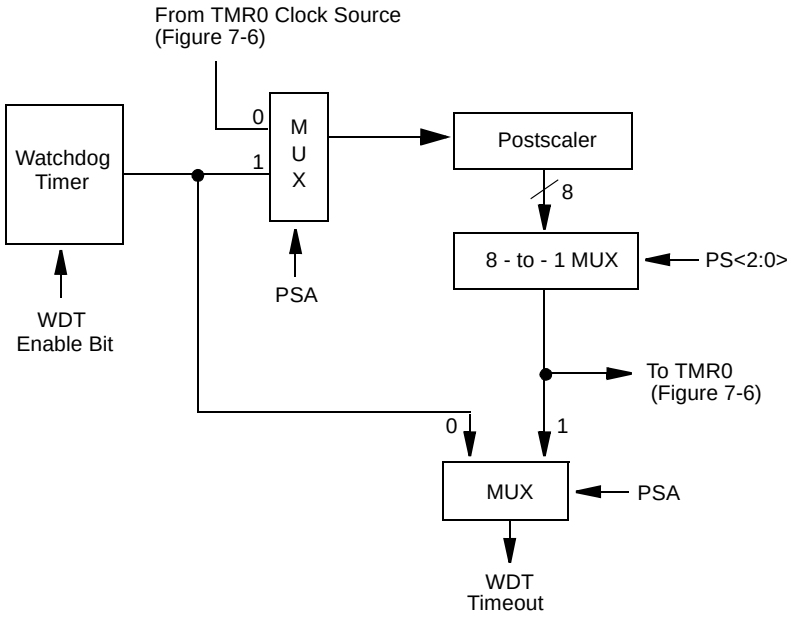
The $\overline{MCLR}$ Reset path has a noise filter to detect and ignore small pulses. See Table 10-3 for pulse width specification.

FIGURE 6-6: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT



Note 1: This is a separate oscillator from the RC oscillator of the CLKIN pin.

FIGURE 6-13: WATCHDOG TIMER BLOCK DIAGRAM



Note 1: T0SE, T0CS, PSA, PS0-PS2 are bits in the OPTION register.

TABLE 6-7: SUMMARY OF WATCHDOG TIMER REGISTERS

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR	Value on all other RESETS
2007h	Config. bits	—	Reserved	CP1	CP0	PWRTE	WDTE	FOSC1	FOSC0		
81h	OPTION	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legend: x = unknown, u = unchanged, q = value depends on condition, — = unimplemented, read as '0'.
Shaded cells are not used by the Watchdog Timer.

FIGURE 7-3: TIMER0 TIMING: INTERNAL CLOCK/PRESCALE 1:2

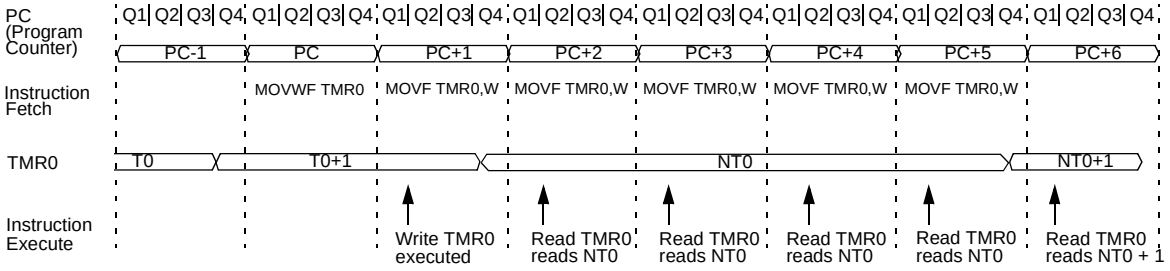
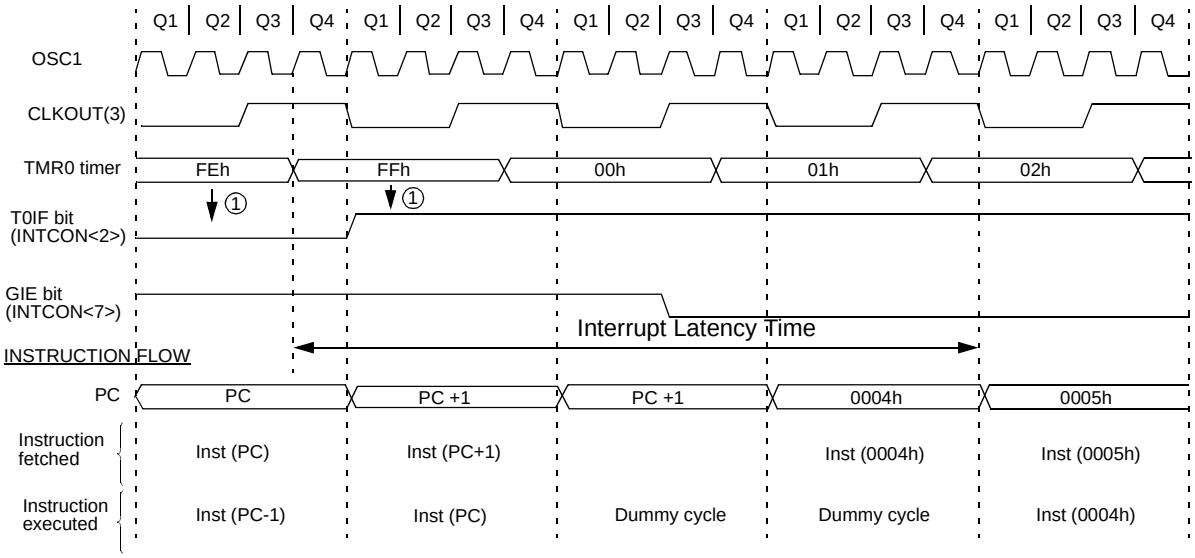


FIGURE 7-4: TIMER0 INTERRUPT TIMING



Note 1: T0IF interrupt flag is sampled here (every Q1).
2: Interrupt latency = 4 TCY, where TCY = instruction cycle time.
3: CLKOUT is available only in RC Oscillator mode.

BCF **Bit Clear f**

Syntax: [*label*] BCF f,b

Operands: $0 \le f \le 127$
$0 \le b \le 7$

Operation: $0 \rightarrow$ (f<b>)

Status Affected: None

Encoding:

01	00bb	bfff	ffff

Description: Bit 'b' in register 'f' is cleared.

Words: 1

Cycles: 1

Example
```
BCF     FLAG_REG, 7
```
Before Instruction
FLAG_REG = 0xC7
After Instruction
FLAG_REG = 0x47

BSF **Bit Set f**

Syntax: [*label*] BSF f,b

Operands: $0 \le f \le 127$
$0 \le b \le 7$

Operation: $1 \rightarrow$ (f<b>)

Status Affected: None

Encoding:

01	01bb	bfff	ffff

Description: Bit 'b' in register 'f' is set.

Words: 1

Cycles: 1

Example
```
BSF     FLAG_REG,   7
```
Before Instruction
FLAG_REG = 0x0A
After Instruction
FLAG_REG = 0x8A

BTFSC **Bit Test, Skip if Clear**

Syntax: [*label*] BTFSC f,b

Operands: $0 \le f \le 127$
$0 \le b \le 7$

Operation: skip if (f<b>) = 0

Status Affected: None

Encoding:

01	10bb	bfff	ffff

Description: If bit 'b' in register 'f' is '0' then the next instruction is skipped. If bit 'b' is '0' then the next instruction fetched during the current instruction execution is discarded, and a `NOP` is executed instead, making this a two-cycle instruction.

Words: 1

Cycles: 1(2)

Example
```
HERE    BTFSC   FLAG,1
FALSE   GOTO    PROCESS_CODE
TRUE    •
        •
        •
```
Before Instruction
PC = address `HERE`
After Instruction
if FLAG<1> = 0,
PC = address `TRUE`
if FLAG<1> = 1,
PC = address `FALSE`

BTFSS **Bit Test f, Skip if Set**

Syntax: [*label*] BTFSS f,b

Operands: $0 \le f \le 127$
$0 \le b < 7$

Operation: skip if (f<b>) = 1

Status Affected: None

Encoding:

01	11bb	bfff	ffff

Description: If bit 'b' in register 'f' is '1' then the next instruction is skipped.
If bit 'b' is '1', then the next instruction fetched during the current instruction execution, is discarded and a NOP is executed instead, making this a two-cycle instruction.

Words: 1

Cycles: 1(2)

Example

```
HERE    BTFSS   FLAG,1
FALSE   GOTO    PROCESS_CODE
TRUE    •
        •
        •
```

Before Instruction
PC = address HERE
After Instruction
if FLAG<1> = 0,
PC = address FALSE
if FLAG<1> = 1,
PC = address TRUE

CALL **Call Subroutine**

Syntax: [*label*] CALL k

Operands: $0 \le k \le 2047$

Operation: (PC)+ 1→ TOS,
k → PC<10:0>,
(PCLATH<4:3>) → PC<12:11>

Status Affected: None

Encoding:

10	0kkk	kkkk	kkkk

Description: Call Subroutine. First, return address (PC+1) is pushed onto the stack. The eleven bit immediate address is loaded into PC bits <10:0>. The upper bits of the PC are loaded from PCLATH. CALL is a two-cycle instruction.

Words: 1

Cycles: 2

Example

```
HERE    CALL    THERE
```

Before Instruction
PC = Address HERE
After Instruction
PC = Address THERE
TOS = Address HERE+1

CLRF **Clear f**

Syntax: [*label*] CLRF f

Operands: $0 \le f \le 127$

Operation: 00h → (f)
1 → Z

Status Affected: Z

Encoding:

00	0001	1fff	ffff

Description: The contents of register 'f' are cleared and the Z bit is set.

Words: 1

Cycles: 1

Example

```
CLRF    FLAG_REG
```

Before Instruction
FLAG_REG=0x5A
After Instruction
FLAG_REG=0x00
Z =1

INCFSZ **Increment f, Skip if 0**

Syntax: [*label*] INCFSZ f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) + 1 → (dest), skip if result = 0

Status Affected: None

Encoding:

00	1111	dfff	ffff

Description: The contents of register 'f' are incremented. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.
If the result is 0, the next instruction, which is already fetched, is discarded. A NOP is executed instead making it a two-cycle instruction.

Words: 1

Cycles: 1(2)

Example

```
HERE     INCFSZ     CNT,  1
         GOTO       LOOP
CONTINUE •
         •
         •
```

Before Instruction
PC = address HERE
After Instruction
CNT = CNT + 1
if CNT = 0,
PC = address CONTINUE
if CNT ≠ 0,
PC = address HERE +1

IORWF **Inclusive OR W with f**

Syntax: [*label*] IORWF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (W) .OR. (f) → (dest)

Status Affected: Z

Encoding:

00	0100	dfff	ffff

Description: Inclusive OR the W register with register 'f'. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.

Words: 1

Cycles: 1

Example

```
IORWF          RESULT, 0
```

Before Instruction
RESULT = 0x13
W = 0x91
After Instruction
RESULT = 0x13
W = 0x93
Z = 1

IORLW **Inclusive OR Literal with W**

Syntax: [*label*] IORLW k

Operands: $0 \le k \le 255$

Operation: (W) .OR. k → (W)

Status Affected: Z

Encoding:

11	1000	kkkk	kkkk

Description: The contents of the W register is OR'ed with the eight bit literal 'k'. The result is placed in the W register.

Words: 1

Cycles: 1

Example

```
IORLW   0x35
```

Before Instruction
W = 0x9A
After Instruction
W = 0xBF
Z = 1

MOVLW **Move Literal to W**

Syntax: [*label*] MOVLW k

Operands: $0 \le k \le 255$

Operation: k → (W)

Status Affected: None

Encoding:

11	00xx	kkkk	kkkk

Description: The eight bit literal 'k' is loaded into W register. The don't cares will assemble as 0's.

Words: 1

Cycles: 1

Example

```
MOVLW   0x5A
```

After Instruction
W = 0x5A

MOVF **Move f**

Syntax: [*label*] MOVF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) → (dest)

Status Affected: Z

Encoding:

00	1000	dfff	ffff

Description: The contents of register f is moved to a destination dependant upon the status of d. If d = 0, destination is W register. If d = 1, the destination is file register f itself. d = 1 is useful to test a file register since status flag Z is affected.

Words: 1

Cycles: 1

Example
```
MOVF    FSR,   0
```
After Instruction
W = value in FSR register
Z = 1

MOVWF **Move W to f**

Syntax: [*label*] MOVWF f

Operands: $0 \le f \le 127$

Operation: (W) → (f)

Status Affected: None

Encoding:

00	0000	1fff	ffff

Description: Move data from W register to register 'f'.

Words: 1

Cycles: 1

Example
```
MOVWF   OPTION
```
Before Instruction
OPTION = 0xFF
W = 0x4F
After Instruction
OPTION = 0x4F
W = 0x4F

NOP **No Operation**

Syntax: [*label*] NOP

Operands: None

Operation: No operation

Status Affected: None

Encoding:

00	0000	0xx0	0000

Description: No operation.

Words: 1

Cycles: 1

Example
```
NOP
```

OPTION **Load Option Register**

Syntax: [*label*] OPTION

Operands: None

Operation: (W) → OPTION

Status Affected: None

Encoding:

00	0000	0110	0010

Description: The contents of the W register are loaded in the OPTION register. This instruction is supported for code compatibility with PIC16C5X products. Since OPTION is a readable/writable register, the user can directly address it.

Words: 1

Cycles: 1

Example

To maintain upward compatibility with future PIC MCU products, do not use this instruction.

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 ≤ k ≤ 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 ≤ f ≤ 127
d ∈ [0,1]

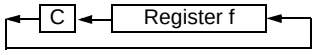
Operation: See description below

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

RRF **Rotate Right f through Carry**

Syntax: [*label*] RRF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

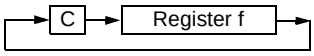
Operation: See description below

Status Affected: C

Encoding:

00	1100	dfff	ffff

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



Words: 1

Cycles: 1

Example
```
RRF             REG1,0
```
Before Instruction
```
REG1 = 1110 0110
C    = 0
```
After Instruction
```
REG1 = 1110 0110
W    = 0111 0011
C    = 0
```

SLEEP

Syntax: [*label* SLEEP]

Operands: None

Operation: 00h → WDT,
0 → WDT prescaler,
1 → $\overline{TO}$,
0 → $\overline{PD}$

Status Affected: $\overline{TO}$, $\overline{PD}$

Encoding:

00	0000	0110	0011

Description: The power-down status bit, $\overline{PD}$ is cleared. Timeout status bit, $\overline{TO}$ is set. Watchdog Timer and its prescaler are cleared.
The processor is put into SLEEP mode with the oscillator stopped. See Section 6.8 for more details.

Words: 1

Cycles: 1

Example:
```
SLEEP
```

SUBLW **Subtract W from Literal**

Syntax: [*label*] SUBLW k

Operands: $0 \le k \le 255$

Operation: k - (W) → (W)

Status Affected: C, DC, Z

Encoding:

11	110x	kkkk	kkkk

Description: The W register is subtracted (2's complement method) from the eight bit literal 'k'. The result is placed in the W register.

Words: 1

Cycles: 1

Example 1:
```
SUBLW   0x02
```
Before Instruction
```
W = 1
C = ?
```
After Instruction
```
W = 1
C = 1; result is positive
```

Example 2: Before Instruction
```
W = 2
C = ?
```
After Instruction
```
W = 0
C = 1; result is zero
```

Example 3: Before Instruction
```
W = 3
C = ?
```
After Instruction
```
W = 0xFF
C = 0; result is negative
```

9.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK object linker combines relocatable objects created by the MPASM assembler and the MPLAB C17 and MPLAB C18 C compilers. It can also link relocatable objects from pre-compiled libraries, using directives from a linker script.

The MPLIB object librarian is a librarian for pre-compiled code to be used with the MPLINK object linker. When a routine from a library is called from another source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications. The MPLIB object librarian manages the creation and modification of library files.

The MPLINK object linker features include:

- Integration with MPASM assembler and MPLAB C17 and MPLAB C18 C compilers.
- Allows all memory areas to be defined as sections to provide link-time flexibility.

The MPLIB object librarian features include:

- Easier linking because single libraries can be included instead of many smaller files.
- Helps keep code maintainable by grouping related modules together.
- Allows libraries to be created and modules to be added, listed, replaced, deleted or extracted.

9.5 MPLAB SIM Software Simulator

The MPLAB SIM software simulator allows code development in a PC-hosted environment by simulating the PIC series microcontrollers on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a file, or user-defined key press, to any of the pins. The execution can be performed in single step, execute until break, or Trace mode.

The MPLAB SIM simulator fully supports symbolic debugging using the MPLAB C17 and the MPLAB C18 C compilers and the MPASM assembler. The software simulator offers the flexibility to develop and debug code outside of the laboratory environment, making it an excellent multi-project software development tool.

9.6 MPLAB ICE High Performance Universal In-Circuit Emulator with MPLAB IDE

The MPLAB ICE universal in-circuit emulator is intended to provide the product development engineer with a complete microcontroller design tool set for PIC microcontrollers (MCUs). Software control of the MPLAB ICE in-circuit emulator is provided by the MPLAB Integrated Development Environment (IDE), which allows editing, building, downloading and source debugging from a single environment.

The MPLAB ICE 2000 is a full-featured emulator system with enhanced trace, trigger and data monitoring features. Interchangeable processor modules allow the system to be easily re configured for emulation of different processors. The universal architecture of the MPLAB ICE in-circuit emulator allows expansion to support new PIC microcontrollers.

The MPLAB ICE in-circuit emulator system has been designed as a real-time emulation system, with advanced features that are generally found on more expensive development tools. The PC platform and Microsoft® Windows environment were chosen to best make these features available to you, the end user.

9.7 ICEPIC In-Circuit Emulator

The ICEPIC low cost, in-circuit emulator is a solution for the Microchip Technology PIC16C5X, PIC16C6X, PIC16C7X and PIC16CXXX families of 8-bit One-Time-Programmable (OTP) microcontrollers. The modular system can support different subsets of PIC16C5X or PIC16CXXX products through the use of interchangeable personality modules, or daughter boards. The emulator is capable of emulating without target application circuitry being present.

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	COSC2	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 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.

10.4 Timing Diagrams and Specifications

FIGURE 10-6: EXTERNAL CLOCK TIMING

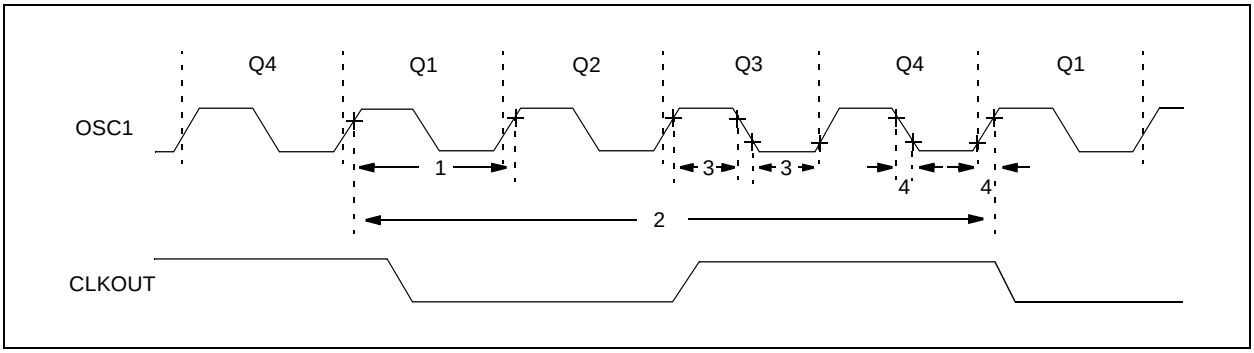


TABLE 10-1: EXTERNAL CLOCK TIMING REQUIREMENTS

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
	Fos	External CLKIN Frequency(1)	DC	—	4	MHz	XT and RC osc mode, VDD=5.0V
			DC	—	20	MHz	HS osc mode
			DC	—	200	kHz	LP osc mode
		Oscillator Frequency(1)	DC	—	4	MHz	RC osc mode, VDD=5.0V
			0.1	—	4	MHz	XT osc mode
			1	—	20	MHz	HS osc mode
			DC	–	200	kHz	LP osc mode
1	Tosc	External CLKIN Period(1)	250	—	—	ns	XT and RC osc mode
			50	—	—	ns	HS osc mode
			5	—	—	µs	LP osc mode
		Oscillator Period(1)	250	—	—	ns	RC osc mode
			250	—	10,000	ns	XT osc mode
			50	—	1,000	ns	HS osc mode
			5	—	—	µs	LP osc mode
2	Tcy	Instruction Cycle Time(1)	1.0	Fos/4	DC	µs	TCY=FOS/4
3*	TosL, TosH	External Clock in (OSC1) High or Low Time	100*	—	—	ns	XT osc mode
			2*	—	—	µs	LP osc mode
			20*	—	—	ns	HS osc mode
4*	TosR, TosF	External Clock in (OSC1) Rise or Fall Time	25*	—	—	ns	XT osc mode
			50*	—	—	ns	LP osc mode
			15*	—	—	ns	HS osc mode

* These parameters are characterized but not tested.

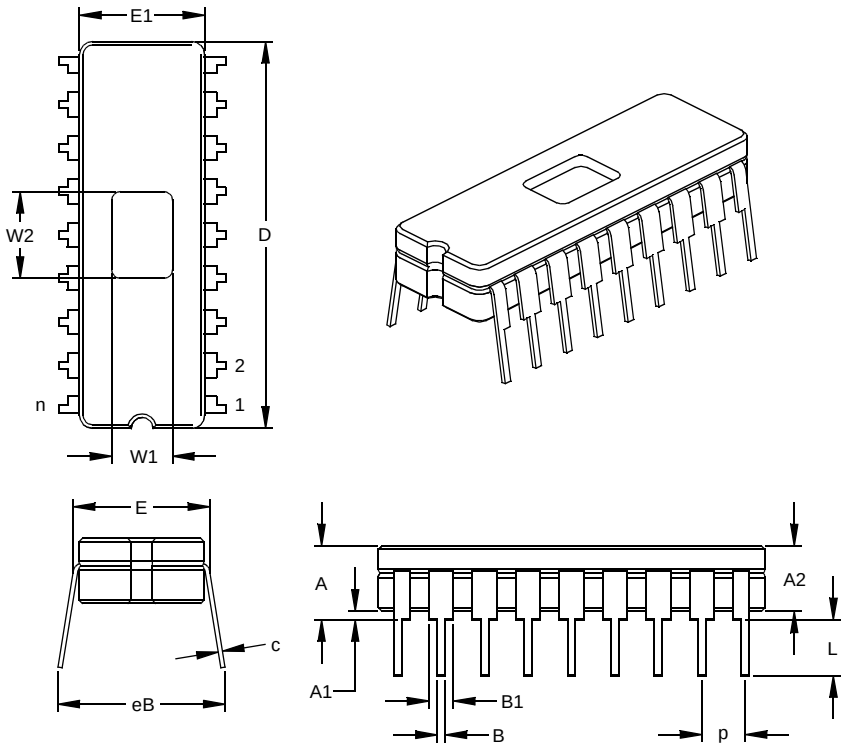
† Data in "Typ" column is at 5.0 V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: Instruction cycle period (TCY) equals four times the input oscillator time-base period. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at "min." values with an external clock applied to the OSC1 pin. When an external clock input is used, the "Max." cycle time limit is "DC" (no clock) for all devices.

NOTES:

18-Lead Ceramic Dual In-line with Window (JW) – 300 mil (CERDIP)

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging

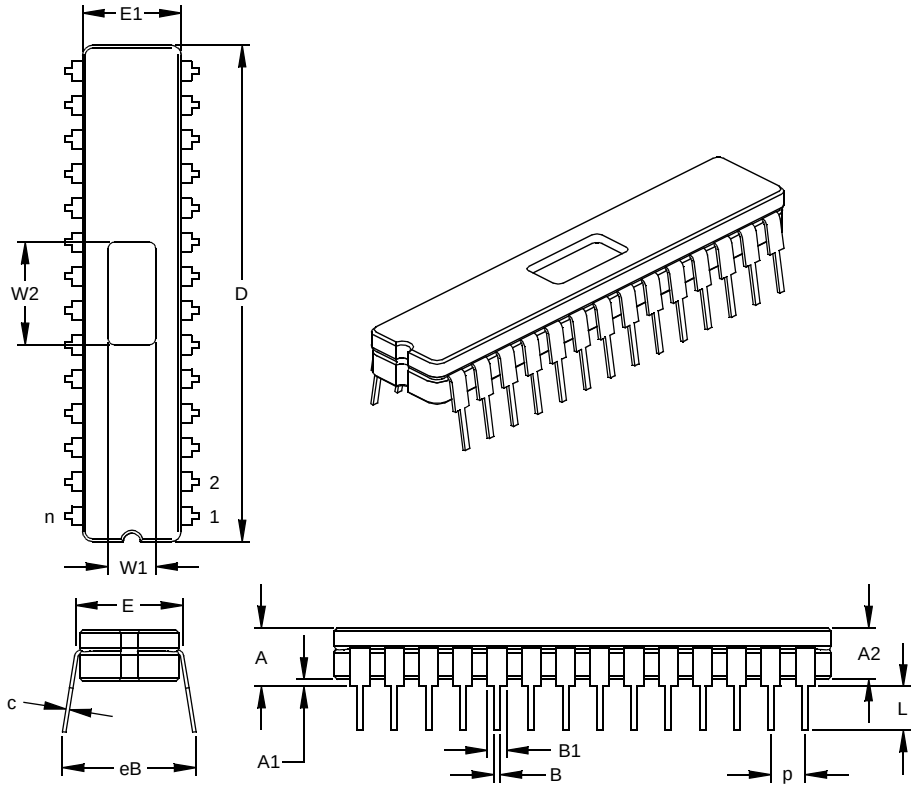


Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		18			18	
Pitch	p		.100			2.54	
Top to Seating Plane	A	.170	.183	.195	4.32	4.64	4.95
Ceramic Package Height	A2	.155	.160	.165	3.94	4.06	4.19
Standoff	A1	.015	.023	.030	0.38	0.57	0.76
Shoulder to Shoulder Width	E	.300	.313	.325	7.62	7.94	8.26
Ceramic Pkg. Width	E1	.285	.290	.295	7.24	7.37	7.49
Overall Length	D	.880	.900	.920	22.35	22.86	23.37
Tip to Seating Plane	L	.125	.138	.150	3.18	3.49	3.81
Lead Thickness	c	.008	.010	.012	0.20	0.25	0.30
Upper Lead Width	B1	.050	.055	.060	1.27	1.40	1.52
Lower Lead Width	B	.016	.019	.021	0.41	0.47	0.53
Overall Row Spacing §	eB	.345	.385	.425	8.76	9.78	10.80
Window Width	W1	.130	.140	.150	3.30	3.56	3.81
Window Length	W2	.190	.200	.210	4.83	5.08	5.33

* Controlling Parameter
§ Significant Characteristic
JEDEC Equivalent: MO-036
Drawing No. C04-010

28-Lead Ceramic Dual In-line with Window (JW) – 300 mil (CERDIP)

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		28			28	
Pitch	p		.100			2.54	
Top to Seating Plane	A	.170	.183	.195	4.32	4.64	4.95
Ceramic Package Height	A2	.155	.160	.165	3.94	4.06	4.19
Standoff	A1	.015	.023	.030	0.38	0.57	0.76
Shoulder to Shoulder Width	E	.300	.313	.325	7.62	7.94	8.26
Ceramic Pkg. Width	E1	.285	.290	.295	7.24	7.37	7.49
Overall Length	D	1.430	1.458	1.485	36.32	37.02	37.72
Tip to Seating Plane	L	.135	.140	.145	3.43	3.56	3.68
Lead Thickness	c	.008	.010	.012	0.20	0.25	0.30
Upper Lead Width	B1	.050	.058	.065	1.27	1.46	1.65
Lower Lead Width	B	.016	.019	.021	0.41	0.47	0.53
Overall Row Spacing §	eB	.345	.385	.425	8.76	9.78	10.80
Window Width	W1	.130	.140	.150	3.30	3.56	3.81
Window Length	W2	.290	.300	.310	7.37	7.62	7.87

* Controlling Parameter
§ Significant Characteristic
JEDEC Equivalent: MO-058
Drawing No. C04-080

Q

R

S

T

W

X

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 Total Pages Sent _____

RE: Reader Response

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?