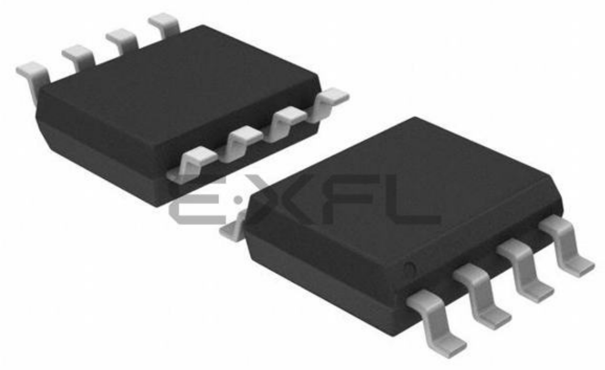


Welcome to E-XFL.COM

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	5
Program Memory Size	768B (512 x 12)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	25 x 8
Voltage - Supply (Vcc/Vdd)	2.5V ~ 5.5V
Data Converters	-
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	8-SOIC (0.154", 3.90mm Width)
Supplier Device Package	8-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic12lc508a-04i-sn

TABLE OF CONTENTS

To Our Valued Customers

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 may exist for current devices, describing minor operational differences (from the data sheet) and recommended workarounds. 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: (602) 786-7277

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.

Corrections to this Data Sheet

We constantly strive to improve the quality of all our products and documentation. We have spent a great deal of time to ensure that this document is correct. However, we realize that we may have missed a few things. If you find any information that is missing or appears in error, please:

- Fill out and mail in the reader response form in the back of this data sheet.
- E-mail us at webmaster@microchip.com.

We appreciate your assistance in making this a better document.

2.0 PIC12C5XX DEVICE VARIETIES

A variety of packaging options are available. Depending on application and production requirements, the proper device option can be selected using the information in this section. When placing orders, please use the PIC12C5XX Product Identification System at the back of this data sheet to specify the correct part number.

2.1 UV Erasable Devices

The UV erasable version, offered in ceramic side brazed package, is optimal for prototype development and pilot programs.

The UV erasable version can be erased and reprogrammed to any of the configuration modes.

Note:	Please note that erasing the device will also erase the pre-programmed internal calibration value for the internal oscillator. The calibration value must be saved prior to erasing the part.

Microchip's PICSTART® PLUS and PRO MATE® programmers all support programming of the PIC12C5XX. Third party programmers also are available; refer to the *Microchip Third Party Guide* for a list of sources.

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 or small volume applications.

The OTP devices, packaged in plastic packages permit the user to program them once. 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 fuse options already programmed by the factory. Certain code and prototype verification procedures do apply before production shipments are available. Please contact your local 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.

2.5 Read Only Memory (ROM) Device

Microchip offers masked ROM to give the customer a low cost option for high volume, mature products.

NOTES:

4.2 Data Memory Organization

Data memory is composed of registers, or bytes of RAM. Therefore, data memory for a device is specified by its register file. The register file is divided into two functional groups: special function registers and general purpose registers.

The special function registers include the TMR0 register, the Program Counter (PC), the Status Register, the I/O registers (ports), and the File Select Register (FSR). In addition, special purpose registers are used to control the I/O port configuration and prescaler options.

The general purpose registers are used for data and control information under command of the instructions.

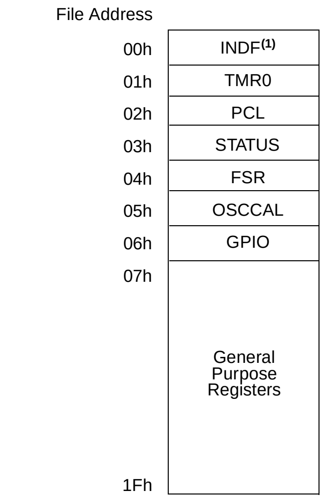
For the PIC12C508, PIC12C508A and PIC12CE518, the register file is composed of 7 special function registers and 25 general purpose registers (Figure 4-2).

For the PIC12C509, PIC12C509A, PIC12CR509A, and PIC12CE519 the register file is composed of 7 special function registers, 25 general purpose registers, and 16 general purpose registers that may be addressed using a banking scheme (Figure 4-3).

4.2.1 GENERAL PURPOSE REGISTER FILE

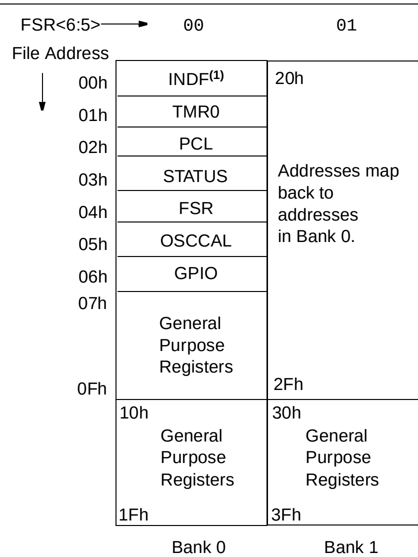
The general purpose register file is accessed either directly or indirectly through the file select register FSR (Section 4.8).

FIGURE 4-2: PIC12C508, PIC12C508A AND PIC12CE518 REGISTER FILE MAP



Note 1: Not a physical register. See Section 4.8

FIGURE 4-3: PIC12C509, PIC12C509A, PIC12CR509A AND PIC12CE519 REGISTER FILE MAP



Note 1: Not a physical register. See Section 4.8

4.8 Indirect Data Addressing; INDF and FSR Registers

The INDF register is not a physical register. Addressing INDF actually addresses the register whose address is contained in the FSR register (FSR is a *pointer*). This is indirect addressing.

EXAMPLE 4-1: INDIRECT ADDRESSING

- Register file 07 contains the value 10h
- Register file 08 contains the value 0Ah
- Load the value 07 into the FSR register
- A read of the INDF register will return the value of 10h
- Increment the value of the FSR register by one (FSR = 08)
- A read of the INDR register now will return the value of 0Ah.

Reading INDF itself indirectly (FSR = 0) will produce 00h. Writing to the INDF register indirectly results in a no-operation (although STATUS bits may be affected).

A simple program to clear RAM locations 10h-1Fh using indirect addressing is shown in Example 4-2.

EXAMPLE 4-2: HOW TO CLEAR RAM USING INDIRECT ADDRESSING

```
            movlw   0x10    ;initialize pointer
            movwf   FSR     ; to RAM
NEXT        clrf    INDF    ;clear INDF register
            incf    FSR,F   ;inc pointer
            btfsc   FSR,4   ;all done?
            goto    NEXT    ;NO, clear next
CONTINUE
            :               ;YES, continue
```

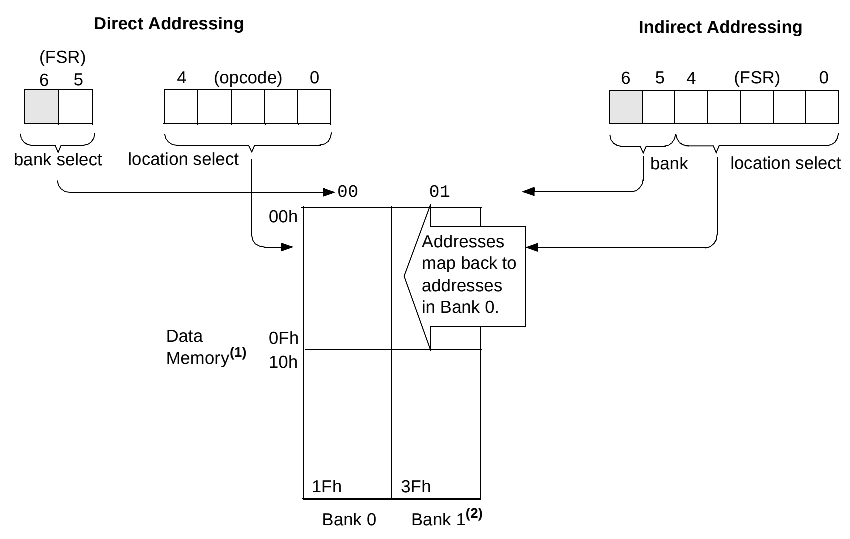
The FSR is a 5-bit wide register. It is used in conjunction with the INDF register to indirectly address the data memory area.

The FSR<4:0> bits are used to select data memory addresses 00h to 1Fh.

PIC12C508/PIC12C508A/PIC12CE518: Does not use banking. FSR<7:5> are unimplemented and read as '1's.

PIC12C509/PIC12C509A/PIC12CR509A/ PIC12CE519: Uses FSR<5>. Selects between bank 0 and bank 1. FSR<7:6> is unimplemented, read as '1' .

FIGURE 4-9: DIRECT/INDIRECT ADDRESSING



Note 1: For register map detail see Section 4.2.

Note 2: PIC12C509, PIC12C509A, PIC12CR509A, PIC12CE519.

5.0 I/O PORT

As with any other register, the I/O register can be written and read under program control. However, read instructions (e.g., `MOVF GPIO,W`) always read the I/O pins independent of the pin's input/output modes. On RESET, all I/O ports are defined as input (inputs are at hi-impedance) since the I/O control registers are all set. See Section 7.0 for SCL and SDA description for PIC12CE5XX.

5.1 GPIO

GPIO is an 8-bit I/O register. Only the low order 6 bits are used (GP5:GP0). Bits 7 and 6 are unimplemented and read as '0's. Please note that GP3 is an input only pin. The configuration word can set several I/O's to alternate functions. When acting as alternate functions the pins will read as '0' during port read. Pins GP0, GP1, and GP3 can be configured with weak pull-ups and also with wake-up on change. The wake-up on change and weak pull-up functions are not pin selectable. If pin 4 is configured as $\overline{\text{MCLR}}$, weak pull-up is always on and wake-up on change for this pin is not enabled.

5.2 TRIS Register

The output driver control register is loaded with the contents of the W register by executing the `TRIS f` instruction. A '1' from a TRIS register bit puts the corresponding output driver in a hi-impedance mode. A '0' puts the contents of the output data latch on the selected pins, enabling the output buffer. The exceptions are GP3 which is input only and GP2 which may be controlled by the option register, see Figure 4-5.

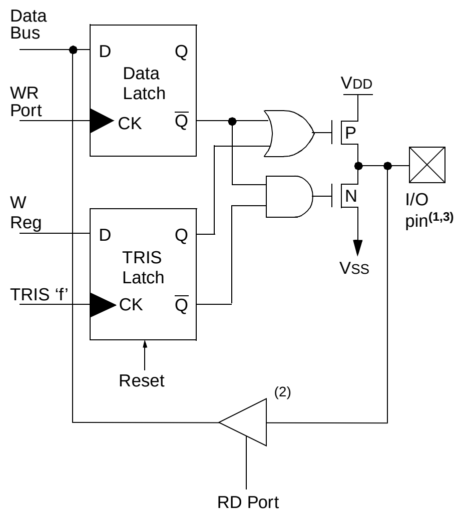
Note:	A read of the ports reads the pins, not the output data latches. That is, if an output driver on a pin is enabled and driven high, but the external system is holding it low, a read of the port will indicate that the pin is low.

The TRIS registers are "write-only" and are set (output drivers disabled) upon RESET.

5.3 I/O Interfacing

The equivalent circuit for an I/O port pin is shown in Figure 5-1. All port pins, except GP3 which is input only, may be used for both input and output operations. For input operations these ports are non-latching. Any input must be present until read by an input instruction (e.g., `MOVF GPIO,W`). The outputs are latched and remain unchanged until the output latch is rewritten. To use a port pin as output, the corresponding direction control bit in TRIS must be cleared (= 0). For use as an input, the corresponding TRIS bit must be set. Any I/O pin (except GP3) can be programmed individually as input or output.

FIGURE 5-1: EQUIVALENT CIRCUIT FOR A SINGLE I/O PIN



Note 1: I/O pins have protection diodes to V_{DD} and V_{SS}.

Note 2: See Table 3-1 for buffer type.

Note 3: See Section 7.0 for SCL and SDA description for PIC12CE5XX

TABLE 5-1: SUMMARY OF PORT REGISTERS

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-On Reset	Value on All Other Resets
N/A	TRIS	—	—							`--11 1111`	`--11 1111`
N/A	OPTION	$\overline{\text{GPWU}}$	$\overline{\text{GPPU}}$	T0CS	T0SE	PSA	PS2	PS1	PS0	`1111 1111`	`1111 1111`
03H	STATUS	GPWUF	—	PAO	$\overline{\text{TO}}$	$\overline{\text{PD}}$	Z	DC	C	`0001 1xxx`	`q00q quuu`(1)
06h	GPIO (PIC12C508/ PIC12C509/ PIC12C508A/ PIC12C509A/ PIC12CR509A)	—	—	GP5	GP4	GP3	GP2	GP1	GP0	`--xx xxxx`	`--uu uuuu`
06h	GPIO (PIC12CE518/ PIC12CE519)	SCL	SDA	GP5	GP4	GP3	GP2	GP1	GP0	`11xx xxxx`	`11uu uuuu`

Legend: Shaded cells not used by Port Registers, read as '0', — = unimplemented, read as '0', x = unknown, u = unchanged, q = see tables in Section 8.7 for possible values.

Note 1: If reset was due to wake-up on change, then bit 7 = 1. All other resets will cause bit 7 = 0.

5.4 I/O Programming Considerations

5.4.1 BI-DIRECTIONAL I/O PORTS

Some instructions operate internally as read followed by write operations. The BCF and BSF instructions, for example, read the entire port into the CPU, execute the bit operation and re-write the result. Caution must be used when these instructions are applied to a port where one or more pins are used as input/outputs. For example, a BSF operation on bit5 of GPIO will cause all eight bits of GPIO to be read into the CPU, bit5 to be set and the GPIO value to be written to the output latches. If another bit of GPIO is used as a bi-directional I/O pin (say bit0) and it is defined as an input at this time, the input signal present on the pin itself would be read into the CPU and rewritten to the data latch of this particular pin, overwriting the previous content. As long as the pin stays in the input mode, no problem occurs. However, if bit0 is switched into output mode later on, the content of the data latch may now be unknown.

Example 5-1 shows the effect of two sequential read-modify-write instructions (e.g., `BCF, BSF`, etc.) on an I/O port.

A pin actively outputting a high or a low should not be driven from external devices at the same time in order to change the level on this pin ("wired-or", "wired-and"). The resulting high output currents may damage the chip.

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

```
;Initial GPIO Settings
; GPIO<5:3> Inputs
; GPIO<2:0> Outputs
;
;                    GPIO latch  GPIO pins
;                    ----------  ----------
  BCF   GPIO, 5   ;--01 -ppp   --11 pppp
  BCF   GPIO, 4   ;--10 -ppp   --11 pppp
  MOVLW 007h       ;
  TRIS  GPIO      ;--10 -ppp   --11 pppp
;
;Note that the user may have expected the pin
;values to be --00 pppp. The 2nd BCF caused
;GP5 to be latched as the pin value (High).
```

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 (Figure 5-2). 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 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 a NOP or another instruction not accessing this I/O port.

8.6.1 WDT PERIOD

The WDT has a nominal time-out period of 18 ms, (with no prescaler). If a longer time-out period is 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, a time-out period of a nominal 2.3 seconds can be realized. These periods vary with temperature, V_{DD} and part-to-part process variations (see DC specs).

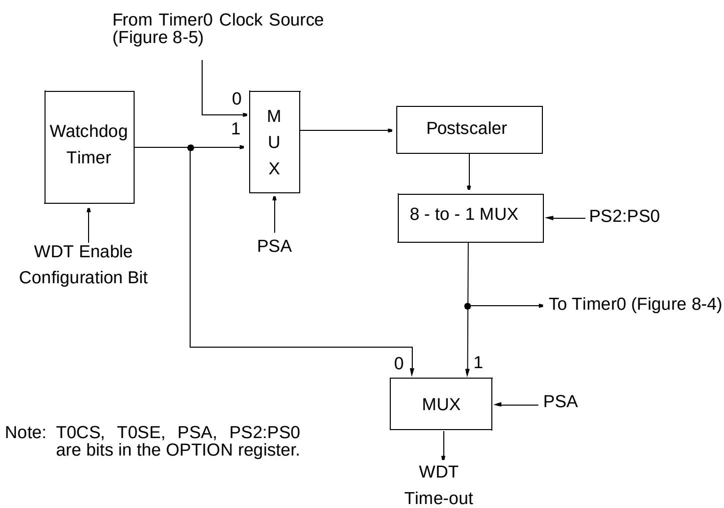
Under worst case conditions (V_{DD} = Min., Temperature = Max., max. WDT prescaler), it may take several seconds before a WDT time-out occurs.

8.6.2 WDT PROGRAMMING CONSIDERATIONS

The `CLRWDT` instruction clears the WDT and the postscaler, if assigned to the WDT, and prevents it from timing out and generating a device RESET.

The `SLEEP` instruction resets the WDT and the postscaler, if assigned to the WDT. This gives the maximum SLEEP time before a WDT wake-up reset.

FIGURE 8-12: WATCHDOG TIMER BLOCK DIAGRAM



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

TABLE 8-6: SUMMARY OF REGISTERS ASSOCIATED WITH THE WATCHDOG TIMER

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-On Reset	Value on All Other Resets
N/A	OPTION	$\overline{\text{GPWU}}$	$\overline{\text{GPPU}}$	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legend: Shaded boxes = Not used by Watchdog Timer, – = unimplemented, read as '0', u = unchanged

9.0 INSTRUCTION SET SUMMARY

Each PIC12C5XX instruction is a 12-bit word divided into an OPCODE, which specifies the instruction type, and one or more operands which further specify the operation of the instruction. The PIC12C5XX instruction set summary in Table 9-2 groups the instructions into byte-oriented, bit-oriented, and literal and control operations. Table 9-1 shows the opcode field descriptions.

For **byte-oriented** instructions, 'f' represents a file register designator and 'd' represents a destination designator. The file register designator is used to specify which one of the 32 file registers is to be used by the instruction.

The destination designator specifies where the result of the operation is to be placed. If 'd' is '0', the result is placed in the W register. If 'd' is '1', the result is placed in the file register specified in the instruction.

For **bit-oriented** instructions, 'b' represents a bit field designator which selects the number of the bit affected by the operation, while 'f' represents the number of the file in which the bit is located.

For **literal and control** operations, 'k' represents an 8 or 9-bit constant or literal value.

TABLE 9-1: OPCODE FIELD DESCRIPTIONS

Field	Description
`f`	Register file address (0x00 to 0x7F)
`W`	Working register (accumulator)
`b`	Bit address within an 8-bit file register
`k`	Literal field, constant data or label
`x`	Don't care location (= 0 or 1) The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
`d`	Destination select; d = 0 (store result in W) d = 1 (store result in file register 'f') Default is d = 1
`label`	Label name
`TOS`	Top of Stack
`PC`	Program Counter
`WDT`	Watchdog Timer Counter
$\overline{\texttt{TO}}$	Time-Out bit
$\overline{\texttt{PD}}$	Power-Down bit
`dest`	Destination, either the W register or the specified register file location
[ ]	Options
( )	Contents
→	Assigned to
< >	Register bit field
∈	In the set of
italics	User defined term (font is courier)

All instructions are executed within a single instruction cycle, unless a conditional test is true or the program counter is changed as a result of an instruction. In this case, the execution takes two instruction cycles. One instruction cycle consists of four oscillator periods. Thus, for an oscillator frequency of 4 MHz, the normal instruction execution time is 1 μs. If a conditional test is true or the program counter is changed as a result of an instruction, the instruction execution time is 2 μs.

Figure 9-1 shows the three general formats that the instructions can have. All examples in the figure use the following format to represent a hexadecimal number:

0xhhh

where 'h' signifies a hexadecimal digit.

FIGURE 9-1: GENERAL FORMAT FOR INSTRUCTIONS

Byte-oriented file register operations

11 ... 6	5	4 ... 0
OPCODE	d	f (FILE #)

d = 0 for destination W
d = 1 for destination f
f = 5-bit file register address

Bit-oriented file register operations

11 ... 8	7 ... 5	4 ... 0
OPCODE	b (BIT #)	f (FILE #)

b = 3-bit bit address
f = 5-bit file register address

Literal and control operations (except `GOTO`)

11 ... 8	7 ... 0
OPCODE	k (literal)

k = 8-bit immediate value

Literal and control operations - `GOTO` instruction

11 ... 9	8 ... 0
OPCODE	k (literal)

k = 9-bit immediate value

ADDWF — Add W and f

Syntax:	[*label*] ADDWF f,d
Operands:	$0 \le f \le 31$ $d \in [0,1]$
Operation:	(W) + (f) → (dest)
Status Affected:	C, DC, Z
Encoding:	`0001` `11df` `ffff`
Description:	Add the contents of the W register and register 'f'. If 'd' is 0 the result is stored in the W register. If '1' the result is stored back in register 'f'.
Words:	1
Cycles:	1
Example:	`ADDWF FSR, 0`

Before Instruction
W = 0x17
FSR = 0xC2

After Instruction
W = 0xD9
FSR = 0xC2

ANDLW — And literal with W

Syntax:	[*label*] ANDLW k
Operands:	$0 \le k \le 255$
Operation:	(W).AND. (k) → (W)
Status Affected:	Z
Encoding:	`1110` `kkkk` `kkkk`
Description:	The contents of the W register are AND'ed with the eight-bit literal 'k'. The result is placed in the W register.
Words:	1
Cycles:	1
Example:	`ANDLW 0x5F`

Before Instruction
W = 0xA3

After Instruction
W = 0x03

ANDWF — AND W with f

Syntax:	[*label*] ANDWF f,d
Operands:	$0 \le f \le 31$ $d \in [0,1]$
Operation:	(W) .AND. (f) → (dest)
Status Affected:	Z
Encoding:	`0001` `01df` `ffff`
Description:	The contents of the W register are AND'ed 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:	`ANDWF FSR, 1`

Before Instruction
W = 0x17
FSR = 0xC2

After Instruction
W = 0x17
FSR = 0x02

BCF — Bit Clear f

Syntax:	[*label*] BCF f,b
Operands:	$0 \le f \le 31$ $0 \le b \le 7$
Operation:	0 → (f<b>)
Status Affected:	None
Encoding:	`0100` `bbbf` `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 ≤ f ≤ 31 0 ≤ b ≤ 7
Operation:	1 → (f<b>)
Status Affected:	None
Encoding:	`0101` `bbbf` `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 f, Skip if Clear

Syntax:	[*label*] BTFSC f,b
Operands:	0 ≤ f ≤ 31 0 ≤ b ≤ 7
Operation:	skip if (f<b>) = 0
Status Affected:	None
Encoding:	`0110` `bbbf` `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 an NOP is executed instead, making this a 2 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 ≤ f ≤ 31 0 ≤ b < 7
Operation:	skip if (f<b>) = 1
Status Affected:	None
Encoding:	`0111` `bbbf` `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 an NOP is executed instead, making this a 2 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 — Subroutine Call

Syntax: [*label*] CALL k

Operands: $0 \leq k \leq 255$

Operation: (PC) + 1→ Top of Stack;
k → PC<7:0>;
(STATUS<6:5>) → PC<10:9>;
0 → PC<8>

Status Affected: None

Encoding:

`1001`	`kkkk`	`kkkk`

Description: Subroutine call. First, return address (PC+1) is pushed onto the stack. The eight bit immediate address is loaded into PC bits <7:0>. The upper bits PC<10:9> are loaded from STATUS<6:5>, PC<8> is cleared. `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: $0 \leq f \leq 31$

Operation: 00h → (f);
1 → Z

Status Affected: Z

Encoding:

`0000`	`011f`	`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

CLRW — Clear W

Syntax: [*label*] CLRW

Operands: None

Operation: 00h → (W);
1 → Z

Status Affected: Z

Encoding:

`0000`	`0100`	`0000`

Description: The W register is cleared. Zero bit (Z) is set.

Words: 1

Cycles: 1

Example: `CLRW`

Before Instruction
W = 0x5A

After Instruction
W = 0x00
Z = 1

CLRWDT — Clear Watchdog Timer

Syntax: [*label*] CLRWDT

Operands: None

Operation: 00h → WDT;
0 → WDT prescaler (if assigned);
1 → $\overline{TO}$;
1 → $\overline{PD}$

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

Encoding:

`0000`	`0000`	`0100`

Description: The `CLRWDT` instruction resets the WDT. It also resets the prescaler, if the prescaler is assigned to the WDT and not Timer0. Status bits $\overline{TO}$ and $\overline{PD}$ are set.

Words: 1

Cycles: 1

Example: `CLRWDT`

Before Instruction
WDT counter = ?

After Instruction
WDT counter = 0x00
WDT prescale = 0
$\overline{TO}$ = 1
$\overline{PD}$ = 1

MOVF — Move f

Syntax:	[*label*] MOVF f,d
Operands:	$0 \le f \le 31$ $d \in [0,1]$
Operation:	$(f) \rightarrow (dest)$
Status Affected:	Z
Encoding:	`0010` `00df` `ffff`
Description:	The contents of register 'f' is moved to destination 'd'. If 'd' is 0, destination is the W register. If 'd' is 1, the destination is file register 'f'. 'd' is 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

MOVLW — Move Literal to W

Syntax:	[*label*] MOVLW k
Operands:	$0 \le k \le 255$
Operation:	$k \rightarrow (W)$
Status Affected:	None
Encoding:	`1100` `kkkk` `kkkk`
Description:	The eight bit literal 'k' is loaded into the W register. The don't cares will assemble as 0s.
Words:	1
Cycles:	1
Example:	`MOVLW 0x5A`

After Instruction
W = 0x5A

MOVWF — Move W to f

Syntax:	[*label*] MOVWF f
Operands:	$0 \le f \le 31$
Operation:	$(W) \rightarrow (f)$
Status Affected:	None
Encoding:	`0000` `001f` `ffff`
Description:	Move data from the W register to register 'f'.
Words:	1
Cycles:	1
Example:	`MOVWF TEMP_REG`

Before Instruction
TEMP_REG = 0xFF
W = 0x4F

After Instruction
TEMP_REG = 0x4F
W = 0x4F

NOP — No Operation

Syntax:	[*label*] NOP
Operands:	None
Operation:	No operation
Status Affected:	None
Encoding:	`0000` `0000` `0000`
Description:	No operation.
Words:	1
Cycles:	1
Example:	`NOP`

SLEEP — Enter SLEEP Mode

Syntax:	[*label*] SLEEP
Operands:	None
Operation:	00h → WDT; 0 → WDT prescaler; 1 → $\overline{TO}$; 0 → $\overline{PD}$
Status Affected:	$\overline{TO}$, $\overline{PD}$, GPWUF
Encoding:	`0000` `0000` `0011`
Description:	Time-out status bit ($\overline{TO}$) is set. The power down status bit ($\overline{PD}$) is cleared. GPWUF is unaffected. The WDT and its prescaler are cleared. The processor is put into SLEEP mode with the oscillator stopped. See section on SLEEP for more details.
Words:	1
Cycles:	1
Example:	`SLEEP`

SUBWF — Subtract W from f

Syntax:	[*label*] SUBWF f,d
Operands:	$0 \le f \le 31$ $d \in [0,1]$
Operation:	(f) – (W) → (dest)
Status Affected:	C, DC, Z
Encoding:	`0000` `10df` `ffff`
Description:	Subtract (2's complement method) the W register from 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 1: `SUBWF REG1, 1`

Before Instruction
REG1 = 3
W = 2
C = ?

After Instruction
REG1 = 1
W = 2
C = 1 ; result is positive

Example 2:

Before Instruction
REG1 = 2
W = 2
C = ?

After Instruction
REG1 = 0
W = 2
C = 1 ; result is zero

Example 3:

Before Instruction
REG1 = 1
W = 2
C = ?

After Instruction
REG1 = FF
W = 2
C = 0 ; result is negative

NOTES:

NOTES:

TABLE 11-5: RESET, WATCHDOG TIMER, AND DEVICE RESET TIMER - PIC12C508/C509

AC Characteristics	Standard Operating Conditions (unless otherwise specified) Operating Temperature $0°C \le T_A \le +70°C$ (commercial) $-40°C \le T_A \le +85°C$ (industrial) $-40°C \le T_A \le +125°C$ (extended) Operating Voltage V_{DD} range is described in Section 11.1						
Parameter No.	**Sym**	**Characteristic**	**Min**	**Typ(1)**	**Max**	**Units**	**Conditions**
30	TmcL	$\overline{MCLR}$ Pulse Width (low)	2000*	—	—	ns	V_{DD} = 5 V
31	Twdt	Watchdog Timer Time-out Period (No Prescaler)	9*	18*	30*	ms	V_{DD} = 5 V (Commercial)
32	T_{DRT}	Device Reset Timer Period(2)	9*	18*	30*	ms	V_{DD} = 5 V (Commercial)
34	Tioz	I/O Hi-impedance from $\overline{MCLR}$ Low	—	—	2000*	ns	

* These parameters are characterized but not tested.

Note 1: Data in the Typical (“Typ”) column is at 5V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 2: See Table 11-6.

TABLE 11-6: DRT (DEVICE RESET TIMER PERIOD - PIC12C508/C509)

Oscillator Configuration	POR Reset	Subsequent Resets
IntRC & ExtRC	18 ms (typical)	300 µs (typical)
XT & LP	18 ms (typical)	18 ms (typical)

FIGURE 14-9: I_{OL} vs. V_{OL}, V_{DD} = 2.5 V

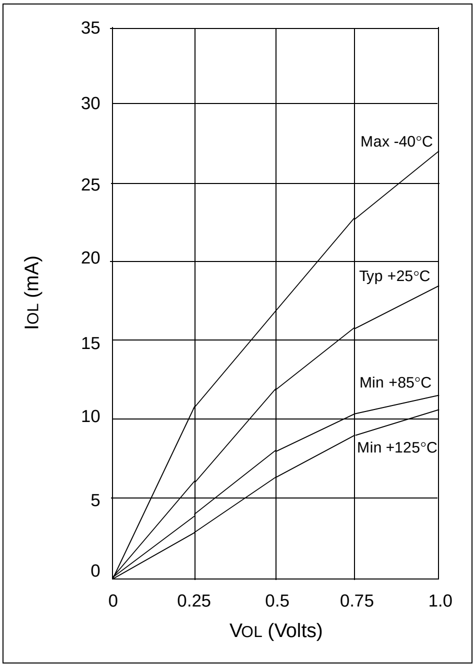


FIGURE 14-10: I_{OL} vs. V_{OL}, V_{DD} = 3.5 V

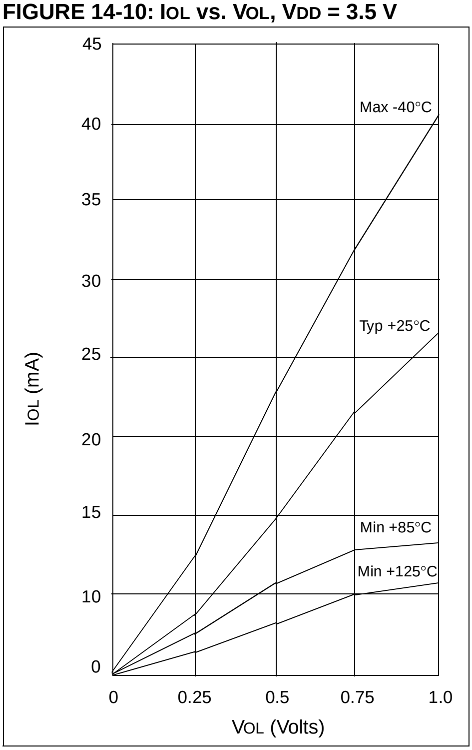


FIGURE 14-11: I_{OH} vs. V_{OH}, V_{DD} = 5.5 V

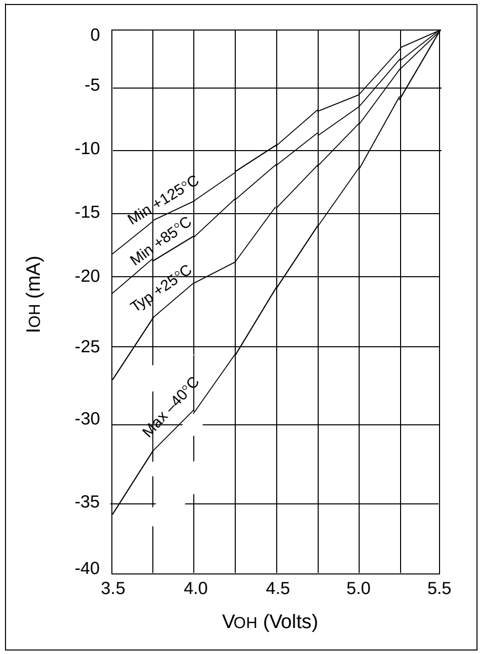


FIGURE 14-12: I_{OL} vs. V_{OL}, V_{DD} = 5.5 V

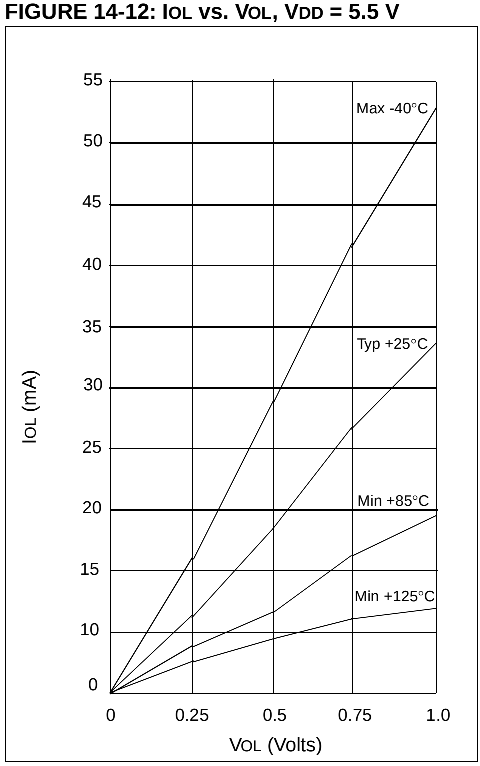


FIGURE 14-13: TYPICAL IPD VS. VDD, WATCHDOG DISABLED (25°C)

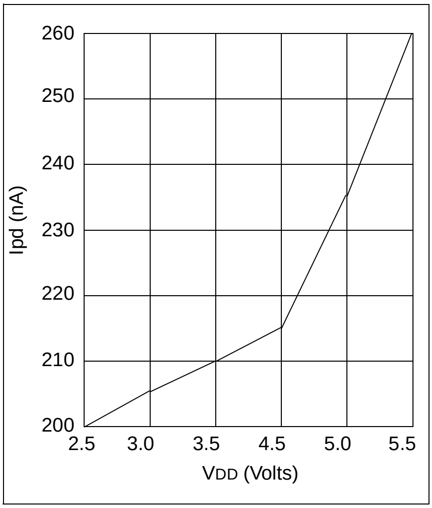
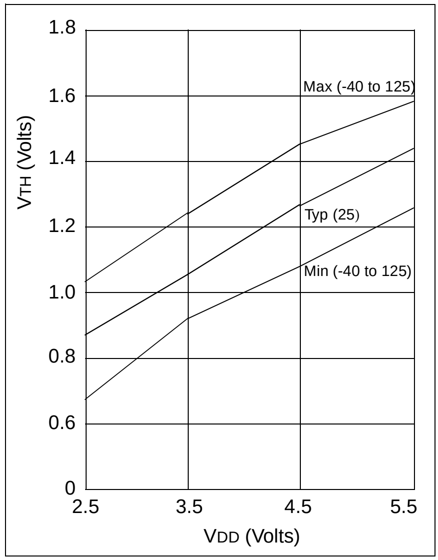


FIGURE 14-14: VTH (INPUT THRESHOLD VOLTAGE) OF GPIO PINS VS. VDD



INDEX