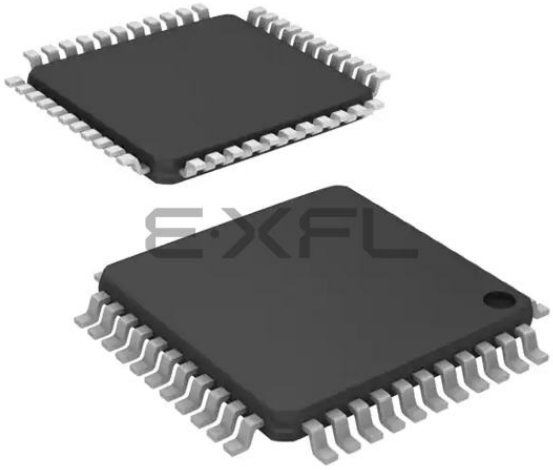


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	PIC
Core Size	8-Bit
Speed	16MHz
Connectivity	UART/USART
Peripherals	POR, PWM, WDT
Number of I/O	33
Program Memory Size	8KB (4K x 16)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	454 x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 6V
Data Converters	-
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	44-TQFP
Supplier Device Package	44-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic17c43t-16i-pt

6.4.1 INDIRECT ADDRESSING REGISTERS

The PIC17C4X has four registers for indirect addressing. These registers are:

- INDF0 and FSR0
- INDF1 and FSR1

Registers INDF0 and INDF1 are not physically implemented. Reading or writing to these registers activates indirect addressing, with the value in the corresponding FSR register being the address of the data. The FSR is an 8-bit register and allows addressing anywhere in the 256-byte data memory address range. For banked memory, the bank of memory accessed is specified by the value in the BSR.

If file INDF0 (or INDF1) itself is read indirectly via an FSR, all '0's are read (Zero bit is set). Similarly, if INDF0 (or INDF1) is written to indirectly, the operation will be equivalent to a NOP, and the status bits are not affected.

6.4.2 INDIRECT ADDRESSING OPERATION

The indirect addressing capability has been enhanced over that of the PIC16CXX family. There are two control bits associated with each FSR register. These two bits configure the FSR register to:

- Auto-decrement the value (address) in the FSR after an indirect access
- Auto-increment the value (address) in the FSR after an indirect access
- No change to the value (address) in the FSR after an indirect access

These control bits are located in the ALUSTA register. The FSR1 register is controlled by the FS3:FS2 bits and FSR0 is controlled by the FS1:FS0 bits.

When using the auto-increment or auto-decrement features, the effect on the FSR is not reflected in the ALUSTA register. For example, if the indirect address causes the FSR to equal '0', the Z bit will not be set.

If the FSR register contains a value of 0h, an indirect read will read 0h (Zero bit is set) while an indirect write will be equivalent to a NOP (status bits are not affected).

Indirect addressing allows single cycle data transfers within the entire data space. This is possible with the use of the MOVPF and MOVFP instructions, where either 'p' or 'f' is specified as INDF0 (or INDF1).

If the source or destination of the indirect address is in banked memory, the location accessed will be determined by the value in the BSR.

A simple program to clear RAM from 20h - FFh is shown in Example 6-1.

EXAMPLE 6-1: INDIRECT ADDRESSING

```
    MOVLW    0x20          ;
    MOVWF    FSR0          ; FSR0 = 20h
    BCF      ALUSTA, FS1   ; Increment FSR
    BSF      ALUSTA, FS0   ; after access
    BCF      ALUSTA, C     ; C = 0
    MOVLW    END_RAM + 1   ;
LP  CLRF     INDF0         ; Addr(FSR) = 0
    CPFSEQ   FSR0          ; FSR0 = END_RAM+1?
    GOTO     LP            ; NO, clear next
    :                      ; YES, All RAM is
    :                      ; cleared
```

6.5 Table Pointer (TBLPTRL and TBLPTRH)

File registers TBLPTRL and TBLPTRH form a 16-bit pointer to address the 64K program memory space. The table pointer is used by instructions TABLWT and TABLRD.

The TABLRD and the TABLWT instructions allow transfer of data between program and data space. The table pointer serves as the 16-bit address of the data word within the program memory. For a more complete description of these registers and the operation of Table Reads and Table Writes, see Section 7.0.

6.6 Table Latch (TBLATH, TBLATL)

The table latch (TBLAT) is a 16-bit register, with TBLATH and TBLATL referring to the high and low bytes of the register. It is not mapped into data or program memory. The table latch is used as a temporary holding latch during data transfer between program and data memory (see descriptions of instructions TABLRD, TABLWT, TLRD and TLWT). For a more complete description of these registers and the operation of Table Reads and Table Writes, see Section 7.0.

7.0 TABLE READS AND TABLE WRITES

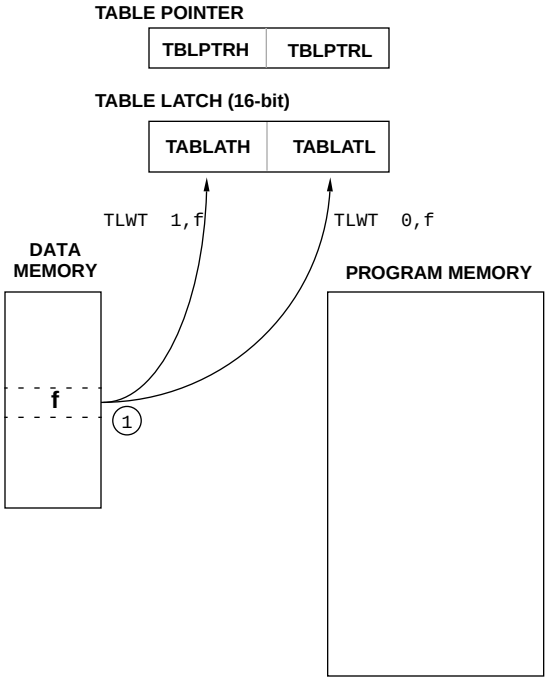
The PIC17C4X has four instructions that allow the processor to move data from the data memory space to the program memory space, and vice versa. Since the program memory space is 16-bits wide and the data memory space is 8-bits wide, two operations are required to move 16-bit values to/from the data memory.

The `TLWT t,f` and `TABLWT t,i,f` instructions are used to write data from the data memory space to the program memory space. The `TLRD t,f` and `TABLRD t,i,f` instructions are used to write data from the program memory space to the data memory space.

The program memory can be internal or external. For the program memory access to be external, the device needs to be operating in extended microcontroller or microprocessor mode.

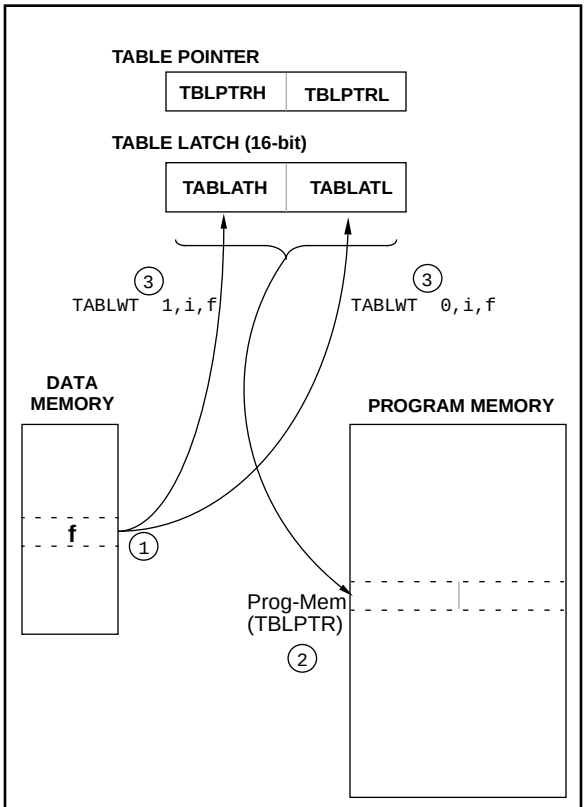
Figure 7-1 through Figure 7-4 show the operation of these four instructions.

FIGURE 7-1: TLWT INSTRUCTION OPERATION



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

FIGURE 7-2: TABLWT INSTRUCTION OPERATION



Note 1: 8-bit value, from register 'f', loaded into the high or low byte in TABLAT (16-bit).
2: 16-bit TABLAT value written to address Program Memory (TBLPTR).
3: If "i" = 1, then TBLPTR = TBLPTR + 1, If "i" = 0, then TBLPTR is unchanged.

7.2 Table Writes to External Memory

Table writes to external memory are always two-cycle instructions. The second cycle writes the data to the external memory location. The sequence of events for an external memory write are the same for an internal write.

Note: If an interrupt is pending or occurs during the TABLWT, the two cycle table write completes. The RA0/INT, TMR0, or T0CKI interrupt flag is automatically cleared or the pending peripheral interrupt is acknowledged.

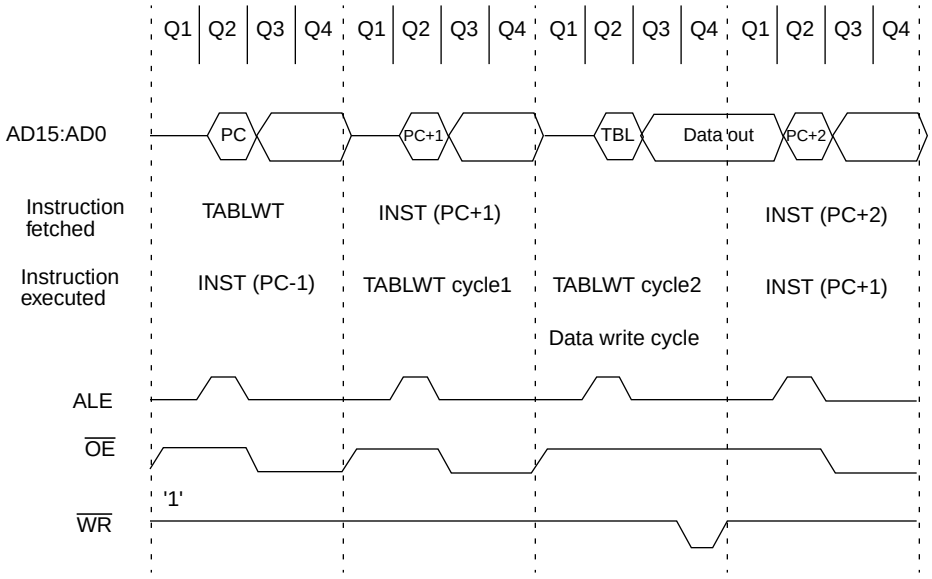
7.2.2 TABLE WRITE CODE

The "i" operand of the TABLWT instruction can specify that the value in the 16-bit TBLPTR register is automatically incremented for the next write. In Example 7-1, the TBLPTR register is not automatically incremented.

EXAMPLE 7-1: TABLE WRITE

```
CLRWDT                  ; Clear WDT
MOVLW   HIGH (TBL_ADDR) ; Load the Table
MOVWF   TBLPTRH         ;   address
MOVLW   LOW (TBL_ADDR)  ;
MOVWF   TBLPTRL         ;
MOVLW   HIGH (DATA)     ; Load HI byte
TLWT    1, WREG         ;   in TABLATCH
MOVLW   LOW (DATA)      ; Load LO byte
TABLWT  0,0,WREG        ;   in TABLATCH
                        ;   and write to
                        ;   program memory
                        ;   (Ext. SRAM)
```

FIGURE 7-5: TABLWT WRITE TIMING (EXTERNAL MEMORY)



Note: If external write GLINTD = '1', Enable bit = '1', '1' → Flag bit, Do table write. The highest pending interrupt is cleared.

9.4.1 PORTE AND DDRE REGISTER

PORTE is a 3-bit bi-directional port. The corresponding data direction register is DDRE. A '1' in DDRE configures the corresponding port pin as an input. A '0' in the DDRE register configures the corresponding port pin as an output. Reading PORTE reads the status of the pins, whereas writing to it will write to the port latch. PORTE is multiplexed with the system bus. When operating as the system bus, PORTE contains the control signals for the address/data bus (AD15:AD0). These control signals are Address Latch Enable (ALE), Output Enable ($\overline{OE}$), and Write ($\overline{WR}$). The control signals $\overline{OE}$ and $\overline{WR}$ are active low signals. The timing for the system bus is shown in the Electrical Characteristics section.

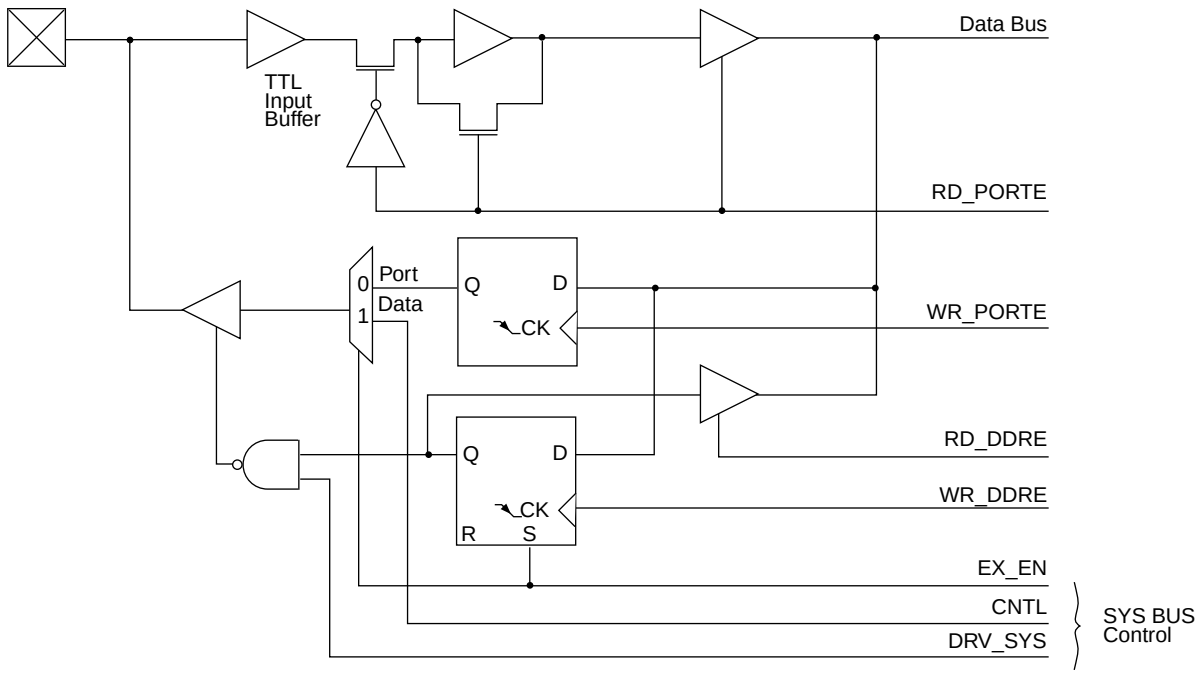
Note:	This port is configured as the system bus when the device's configuration bits are selected to Microprocessor or Extended Microcontroller modes. In the two other microcontroller modes, this port is a general purpose I/O.

Example 9-4 shows the instruction sequence to initialize PORTE. The Bank Select Register (BSR) must be selected to Bank 1 for the port to be initialized.

EXAMPLE 9-4: INITIALIZING PORTE

```
MOVLB  1          ;  Select Bank 1
CLRF   PORTE      ;  Initialize PORTE data
                  ;   latches before setting
                  ;   the data direction
                  ;   register
MOVLW  0x03       ;  Value used to initialize
                  ;   data direction
MOVWF  DDRE       ;  Set RE<1:0> as inputs
                  ;   RE<2> as outputs
                  ;   RE<7:3> are always
                  ;   read as '0'
```

FIGURE 9-8: PORTE BLOCK DIAGRAM (IN I/O PORT MODE)



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

12.1.3.3.1 MAX RESOLUTION/FREQUENCY FOR EXTERNAL CLOCK INPUT

The use of an external clock for the PWM time-base (Timer1 or Timer2) limits the PWM output to a maximum resolution of 8-bits. The PWxDCL<7:6> bits must be kept cleared. Use of any other value will distort the PWM output. All resolutions are supported when internal clock mode is selected. The maximum attainable frequency is also lower. This is a result of the timing requirements of an external clock input for a timer (see the Electrical Specification section). The maximum PWM frequency, when the timers clock source is the RB4/TCLK12 pin, is shown in Table 12-3 (standard resolution mode).

12.2 Timer3

Timer3 is a 16-bit timer consisting of the TMR3H and TMR3L registers. TMR3H is the high byte of the timer and TMR3L is the low byte. This timer has an associated 16-bit period register (PR3H/CA1H:PR3L/CA1L). This period register can be software configured to be a second 16-bit capture register.

When the TMR3CS bit (TCON1<2>) is clear, the timer increments every instruction cycle (Fosc/4). When TMR3CS is set, the timer increments on every falling edge of the RB5/TCLK3 pin. In either mode, the TMR3ON bit must be set for the timer to increment. When TMR3ON is clear, the timer will not increment or set the TMR3IF bit.

Timer3 has two modes of operation, depending on the CA1/$\overline{PR3}$ bit (TCON2<3>). These modes are:

- One capture and one period register mode
- Dual capture register mode

The PIC17C4X has up to two 16-bit capture registers that capture the 16-bit value of TMR3 when events are detected on capture pins. There are two capture pins (RB0/CAP1 and RB1/CAP2), one for each capture register. The capture pins are multiplexed with PORTB pins. An event can be:

- a rising edge
- a falling edge
- every 4th rising edge
- every 16th rising edge

Each 16-bit capture register has an interrupt flag associated with it. The flag is set when a capture is made. The capture module is truly part of the Timer3 block. Figure 12-7 and Figure 12-8 show the block diagrams for the two modes of operation.

TABLE 12-4: REGISTERS/BITS ASSOCIATED WITH PWM

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 (Note1)
16h, Bank 3	TCON1	CA2ED1	CA2ED0	CA1ED1	CA1ED0	T16	TMR3CS	TMR2CS	TMR1CS	0000 0000	0000 0000
17h, Bank 3	TCON2	CA2OVF	CA1OVF	PWM2ON	PWM1ON	CA1/$\overline{PR3}$	TMR3ON	TMR2ON	TMR1ON	0000 0000	0000 0000
10h, Bank 2	TMR1	Timer1 register								xxxx xxxx	uuuu uuuu
11h, Bank 2	TMR2	Timer2 register								xxxx xxxx	uuuu uuuu
16h, Bank 1	PIR	RBIF	TMR3IF	TMR2IF	TMR1IF	CA2IF	CA1IF	TXIF	RCIF	0000 0010	0000 0010
17h, Bank 1	PIE	RBIE	TMR3IE	TMR2IE	TMR1IE	CA2IE	CA1IE	TXIE	RCIE	0000 0000	0000 0000
07h, Unbanked	INTSTA	PEIF	T0CKIF	T0IF	INTF	PEIE	T0CKIE	T0IE	INTE	0000 0000	0000 0000
06h, Unbanked	CPUSTA	—	—	STKAV	GLINTD	$\overline{TO}$	$\overline{PD}$	—	—	--11 11--	--11 qq--
10h, Bank 3	PW1DCL	DC1	DC0	—	—	—	—	—	—	xx-- ----	uu-- ----
11h, Bank 3	PW2DCL	DC1	DC0	TM2PW2	—	—	—	—	—	xx0- ----	uu0- ----
12h, Bank 3	PW1DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	xxxx xxxx	uuuu uuuu
13h, Bank 3	PW2DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	xxxx xxxx	uuuu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented read as '0', q = value depends on conditions, shaded cells are not used by PWM.

14.3 Watchdog Timer (WDT)

The Watchdog Timer's function is to recover from software malfunction. The WDT uses an internal free running on-chip RC oscillator for its clock source. This does not require any external components. This RC oscillator is separate from the RC oscillator of the OSC1/CLKIN pin. That means that the WDT will run, even if the clock on the OSC1/CLKIN and OSC2/CLKOUT pins of the device has been stopped, for example, by execution of a SLEEP instruction. During normal operation and SLEEP mode, a WDT time-out generates a device RESET. The WDT can be permanently disabled by programming the configuration bits WDTPS1:WDTPS0 as '00' (Section 14.1).

Under normal operation, the WDT must be cleared on a regular interval. This time is less the minimum WDT overflow time. Not clearing the WDT in this time frame will cause the WDT to overflow and reset the device.

14.3.1 WDT PERIOD

The WDT has a nominal time-out period of 12 ms, (with postscaler = 1). The time-out periods vary with temperature, VDD and process variations from part to part (see DC specs). If longer time-out periods are desired, a postscaler with a division ratio of up to 1:256 can be assigned to the WDT. Thus, typical time-out periods up to 3.0 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 thus generating a device RESET condition.

The $\overline{TO}$ bit in the CPUSTA register will be cleared upon a WDT time-out.

14.3.2 CLEARING THE WDT AND POSTSCALER

The WDT and postscaler are cleared when:

- The device is in the reset state
- A SLEEP instruction is executed
- A CLRWDT instruction is executed
- Wake-up from SLEEP by an interrupt

The WDT counter/postscaler will start counting on the first edge after the device exits the reset state.

14.3.3 WDT PROGRAMMING CONSIDERATIONS

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

The WDT and postscaler is the Power-up Timer during the Power-on Reset sequence.

14.3.4 WDT AS NORMAL TIMER

When the WDT is selected as a normal timer, the clock source is the device clock. Neither the WDT nor the postscaler are directly readable or writable. The overflow time is 65536 TOSC cycles. On overflow, the $\overline{TO}$ bit is cleared (device is not reset). The CLRWDT instruction can be used to set the $\overline{TO}$ bit. This allows the WDT to be a simple overflow timer. When in sleep, the WDT does not increment.

Table 15-2 lists the instructions recognized by the MPASM assembler.

Note 1: Any unused opcode is Reserved. Use of any reserved opcode may cause unexpected operation.

Note 2: The shaded instructions are not available in the PIC17C42

All instruction examples use the following format to represent a hexadecimal number:

0xhh

where h signifies a hexadecimal digit.

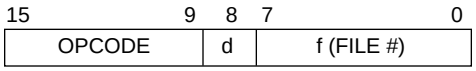
To represent a binary number:

`0000 0100b`

where b signifies a binary string.

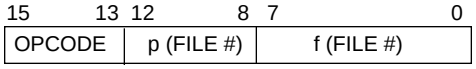
FIGURE 15-1: GENERAL FORMAT FOR INSTRUCTIONS

Byte-oriented file register operations



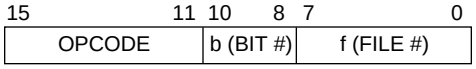
d = 0 for destination WREG
d = 1 for destination f
f = 8-bit file register address

Byte to Byte move operations



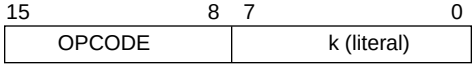
p = peripheral register file address
f = 8-bit file register address

Bit-oriented file register operations



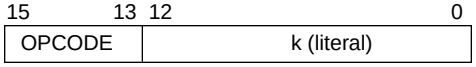
b = 3-bit address
f = 8-bit file register address

Literal and control operations



k = 8-bit immediate value

Call and GOTO operations



k = 13-bit immediate value

15.1 Special Function Registers as Source/Destination

The PIC17C4X's orthogonal instruction set allows read and write of all file registers, including special function registers. There are some special situations the user should be aware of:

15.1.1 ALUSTA AS DESTINATION

If an instruction writes to ALUSTA, the Z, C, DC and OV bits may be set or cleared as a result of the instruction and overwrite the original data bits written. For example, executing `CLRF ALUSTA` will clear register ALUSTA, and then set the Z bit leaving `0000 0100b` in the register.

15.1.2 PCL AS SOURCE OR DESTINATION

Read, write or read-modify-write on PCL may have the following results:

Read PC:	PCH → PCLATH; PCL → dest
Write PCL:	PCLATH → PCH; 8-bit destination value → PCL
Read-Modify-Write:	PCL→ ALU operand PCLATH → PCH; 8-bit result → PCL

Where PCH = program counter high byte (not an addressable register), PCLATH = Program counter high holding latch, dest = destination, WREG or f.

15.1.3 BIT MANIPULATION

All bit manipulation instructions are done by first reading the entire register, operating on the selected bit and writing the result back (read-modify-write). The user should keep this in mind when operating on special function registers, such as ports.

ADDLW **ADD Literal to WREG**

Syntax: [*label*] ADDLW k

Operands: $0 \le k \le 255$

Operation: (WREG) + k → (WREG)

Status Affected: OV, C, DC, Z

Encoding:

1011	0001	kkkk	kkkk

Description: The contents of WREG are added to the 8-bit literal 'k' and the result is placed in WREG.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Execute	Write to WREG

Example: `ADDLW 0x15`

Before Instruction
WREG = 0x10

After Instruction
WREG = 0x25

ADDWF **ADD WREG to f**

Syntax: [*label*] ADDWF f,d

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

Operation: (WREG) + (f) → (dest)

Status Affected: OV, C, DC, Z

Encoding:

0000	111d	ffff	ffff

Description: Add WREG to register 'f'. If 'd' is 0 the result is stored in WREG. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `ADDWF REG, 0`

Before Instruction
WREG = 0x17
REG = 0xC2

After Instruction
WREG = 0xD9
REG = 0xC2

BTFSS **Bit Test, 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:

1001	0bbb	ffff	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 two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Execute	NOP

If skip:

Q1	Q2	Q3	Q4
Forced NOP	NOP	Execute	NOP

Example:
```
HERE    BTFSS   FLAG,1
FALSE   :
TRUE    :
```

Before Instruction
PC = address (HERE)

After Instruction
If FLAG<1> = 0;
PC = address (FALSE)
If FLAG<1> = 1;
PC = address (TRUE)

BTG **Bit Toggle f**

Syntax: [*label*] BTG f,b

Operands: $0 \le f \le 255$
$0 \le b < 7$

Operation: $(\overline{f<b>}) \rightarrow (f<b>)$

Status Affected: None

Encoding:

0011	1bbb	ffff	ffff

Description: Bit 'b' in data memory location 'f' is inverted.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Execute	Write register 'f'

Example:
```
BTG     PORTC,  4
```

Before Instruction:
PORTC = 0111 0101 [0x75]

After Instruction:
PORTC = 0110 0101 [0x65]

CALL **Subroutine Call**

Syntax: [*label*] CALL k

Operands: 0 ≤ k ≤ 4095

Operation: PC+ 1→ TOS, k → PC<12:0>, k<12:8> → PCLATH<4:0>; PC<15:13> → PCLATH<7:5>

Status Affected: None

Encoding:

111k	kkkk	kkkk	kkkk

Description: Subroutine call within 8K page. First, return address (PC+1) is pushed onto the stack. The 13-bit value is loaded into PC bits<12:0>. Then the upper-eight bits of the PC are copied into PCLATH. `Call` is a two-cycle instruction.

See `LCALL` for calls outside 8K memory space.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'<7:0>	Execute	NOP
Forced NOP	NOP	Execute	NOP

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,s

Operands: 0 ≤ f ≤ 255

Operation: 00h → f, s ∈ [0,1]
00h → dest

Status Affected: None

Encoding:

0010	100s	ffff	ffff

Description: Clears the contents of the specified register(s).
s = 0: Data memory location 'f' and WREG are cleared.
s = 1: Data memory location 'f' is cleared.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `CLRF FLAG_REG`

Before Instruction
FLAG_REG = 0x5A

After Instruction
FLAG_REG = 0x00

CLRWDT **Clear Watchdog Timer**

Syntax: [*label*] CLRWDT

Operands: None

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

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

Encoding:

0000	0000	0000	0100

Description: CLRWDT instruction resets the watchdog timer. It also resets the prescaler of the WDT. Status bits $\overline{TO}$ and $\overline{PD}$ are set.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register ALUSTA	Execute	NOP

Example: CLRWDT

Before Instruction
WDT counter = ?

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

COMF **Complement f**

Syntax: [*label*] COMF f,d

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]

Operation: $(\bar{f})$ → (dest)

Status Affected: Z

Encoding:

0001	001d	ffff	ffff

Description: The contents of register 'f' are complemented. If 'd' is 0 the result is stored in WREG. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Execute	Write register 'f'

Example: COMF REG1,0

Before Instruction
REG1 = 0x13

After Instruction
REG1 = 0x13
WREG = 0xEC

CPFSEQ **Compare f with WREG, skip if f = WREG**

Syntax: [*label*] CPFSEQ f

Operands: $0 \le f \le 255$

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

Status Affected: None

Encoding:

0011	0001	ffff	ffff

Description: Compares the contents of data memory location 'f' to the contents of WREG by performing an unsigned subtraction.

If 'f' = WREG then the fetched instruction is discarded and an NOP is executed instead making this a two-cycle instruction.

Words: 1

Cycles: 1 (2)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Execute	NOP

If skip:

Q1	Q2	Q3	Q4
Forced NOP	NOP	Execute	NOP

Example:

```
HERE    CPFSEQ REG
NEQUAL  :
EQUAL   :
```

Before Instruction
PC Address = HERE
WREG = ?
REG = ?

After Instruction
If REG = WREG;
PC = Address (EQUAL)
If REG ≠ WREG;
PC = Address (NEQUAL)

CPFSGT **Compare f with WREG, skip if f > WREG**

Syntax: [*label*] CPFSGT f

Operands: $0 \le f \le 255$

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

Status Affected: None

Encoding:

0011	0010	ffff	ffff

Description: Compares the contents of data memory location 'f' to the contents of the WREG by performing an unsigned subtraction.

If the contents of 'f' > the contents of WREG then the fetched instruction is discarded and an NOP is executed instead making this a two-cycle instruction.

Words: 1

Cycles: 1 (2)

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Execute	NOP

If skip:

Q1	Q2	Q3	Q4
Forced NOP	NOP	Execute	NOP

Example:

```
HERE     CPFSGT REG
NGREATER :
GREATER  :
```

Before Instruction
PC = Address (HERE)
WREG = ?

After Instruction
If REG > WREG;
PC = Address (GREATER)
If REG ≤ WREG;
PC = Address (NGREATER)

XORLW **Exclusive OR Literal with WREG**

Syntax: [*label*] XORLW k

Operands: $0 \le k \le 255$

Operation: (WREG) .XOR. k → (WREG)

Status Affected: Z

Encoding:

1011	0100	kkkk	kkkk

Description: The contents of WREG are XOR'ed with the 8-bit literal 'k'. The result is placed in WREG.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Execute	Write to WREG

Example: `XORLW 0xAF`

Before Instruction
WREG = 0xB5

After Instruction
WREG = 0x1A

XORWF **Exclusive OR WREG with f**

Syntax: [*label*] XORWF f,d

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

Operation: (WREG) .XOR. (f) → (dest)

Status Affected: Z

Encoding:

0000	110d	ffff	ffff

Description: Exclusive OR the contents of WREG with register 'f'. If 'd' is 0 the result is stored in WREG. If 'd' is 1 the result is stored back in the register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `XORWF REG, 1`

Before Instruction
REG = 0xAF
WREG = 0xB5

After Instruction
REG = 0x1A
WREG = 0xB5

Applicable Devices	42	R42	42A	43	R43	44

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
Operating voltage VDD range as described in Section 17.1

Parameter No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
		Output Low Voltage					
D080	VOL	I/O ports (except RA2 and RA3)	–	–	0.1VDD	V	IOL = 4 mA
D081		with TTL buffer	–	–	0.4	V	IOL = 6 mA, VDD = 4.5V Note 6
D082		RA2 and RA3	–	–	3.0	V	IOL = 60.0 mA, VDD = 5.5V
D083		OSC2/CLKOUT (RC and EC osc modes)	–	–	0.4	V	IOL = 2 mA, VDD = 4.5V
		Output High Voltage (Note 3)					
D090	VOH	I/O ports (except RA2 and RA3)	0.9VDD	–	–	V	IOH = -2 mA
D091		with TTL buffer	2.4	–	–	V	IOH = -6.0 mA, VDD = 4.5V Note 6
D092		RA2 and RA3	–	–	12	V	Pulled-up to externally applied voltage
D093		OSC2/CLKOUT (RC and EC osc modes)	2.4	–	–	V	IOH = -5 mA, VDD = 4.5V
		Capacitive Loading Specs on Output Pins					
D100	COSC2	OSC2 pin	–	–	25 ††	pF	In EC or RC osc modes when OSC2 pin is outputting CLKOUT. External clock is used to drive OSC1.
D101	CIO	All I/O pins and OSC2 (in RC mode)	–	–	50 ††	pF	
D102	CAD	System Interface Bus (PORTC, PORTD and PORTE)	–	–	100 ††	pF	In Microprocessor or Extended Microcontroller mode

* These parameters are characterized but not tested.

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

‡ These parameters are for design guidance only and are not tested, nor characterized.

†† Design guidance to attain the AC timing specifications. These loads are not tested.

Note 1: In RC oscillator configuration, the OSC1 pin is a Schmitt Trigger input. It is not recommended that the PIC17CXX devices be driven with external clock in RC mode.

2: The leakage current on the $\overline{MCLR}$ pin is strongly dependent on the 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.

4: These specifications are for the programming of the on-chip program memory EPROM through the use of the table write instructions. The complete programming specifications can be found in: PIC17CXX Programming Specifications (Literature number DS30139).

5: The $\overline{MCLR}$/Vpp pin may be kept in this range at times other than programming, but this is not recommended.

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

Applicable Devices	42	R42	42A	43	R43	44

FIGURE 18-7: TYPICAL IDD vs. FREQUENCY (EXTERNAL CLOCK 25°C)

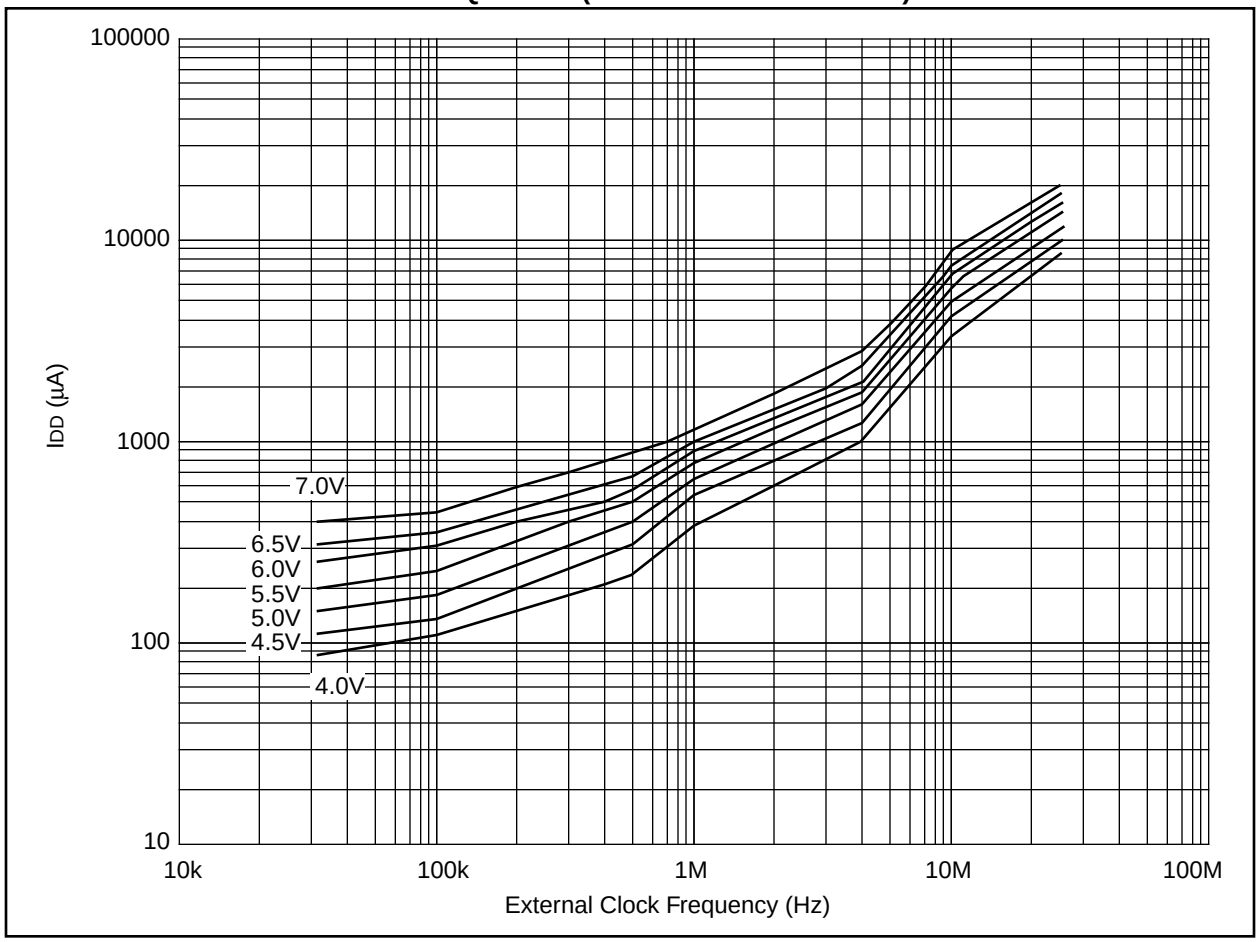
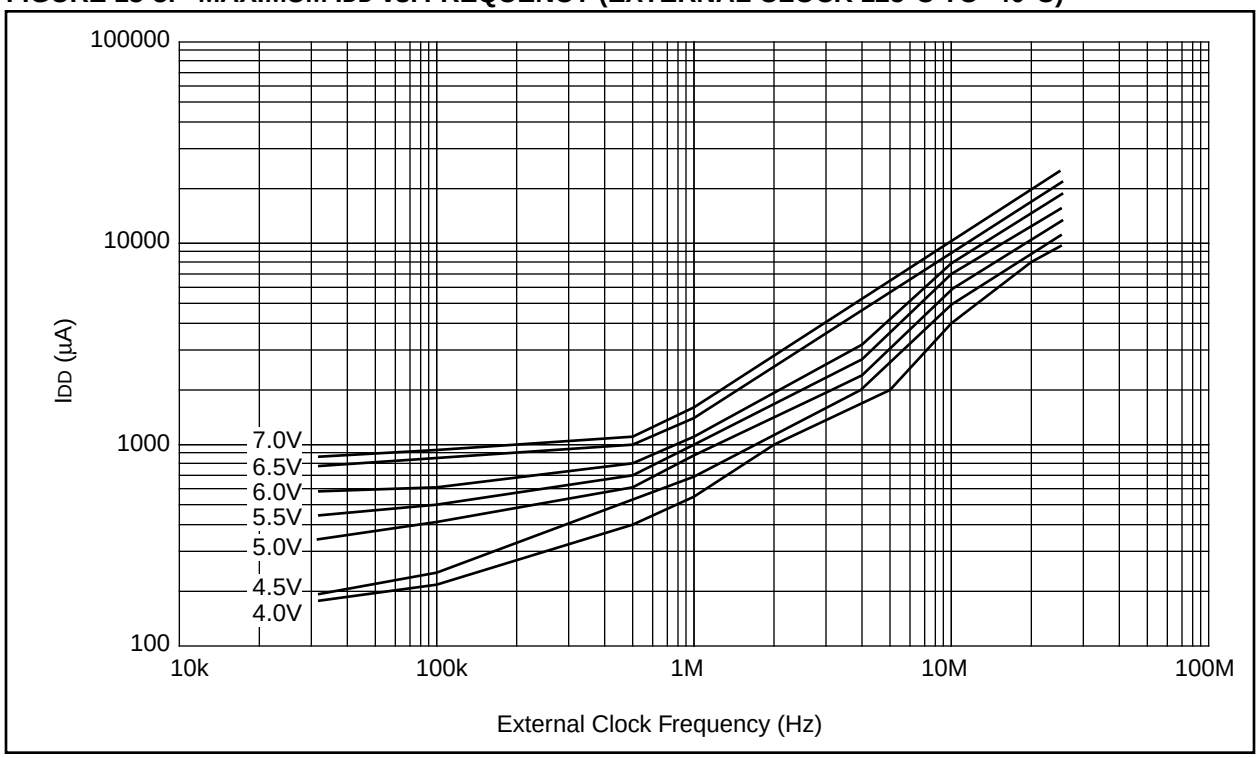


FIGURE 18-8: MAXIMUM IDD vs. FREQUENCY (EXTERNAL CLOCK 125°C TO -40°C)



Applicable Devices	42	R42	42A	43	R43	44

19.1 DC CHARACTERISTICS: PIC17CR42/42A/43/R43/44-16 (Commercial, Industrial) PIC17CR42/42A/43/R43/44-25 (Commercial, Industrial) PIC17CR42/42A/43/R43/44-33 (Commercial, Industrial)

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				
Parameter No.	**Sym**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
D001	VDD	Supply Voltage	4.5	–	6.0	V	
D002	VDR	RAM Data Retention Voltage (Note 1)	1.5 *	–	–	V	Device in SLEEP mode
D003	VPOR	VDD start voltage to ensure internal Power-on Reset signal	–	VSS	–	V	See section on Power-on Reset for details
D004	SVDD	VDD rise rate to ensure internal Power-on Reset signal	0.060 *	–	–	mV/ms	See section on Power-on Reset for details
D010	IDD	Supply Current (Note 2)	–	3	6	mA	FOSC = 4 MHz (Note 4)
D011			–	6	12 *	mA	FOSC = 8 MHz
D012			–	11	24 *	mA	FOSC = 16 MHz
D013			–	19	38	mA	FOSC = 25 MHz
D015			–	25	50	mA	FOSC = 33 MHz
D014			–	95	150	µA	FOSC = 32 kHz, WDT enabled (EC osc configuration)
D020	IPD	Power-down Current (Note 3)	–	10	40	µA	VDD = 5.5V, WDT enabled
D021			–	< 1	5	µA	VDD = 5.5V, WDT disabled

* These parameters are characterized but not tested.

† Data in "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 VDD 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 IDD measurements in active operation mode are:
OSC1 = external square wave, from rail to rail; all I/O pins tristated, pulled to VDD or VSS, T0CKI = VDD, $\overline{MCLR}$ = VDD; WDT enabled/disabled as specified.
Current consumed from the oscillator and I/O's driving external capacitive or resistive loads needs to be considered.
For the RC oscillator, the current through the external pull-up resistor (R) can be estimated as: VDD / (2 • R).
For capacitive loads, the current can be estimated (for an individual I/O pin) as (CL • VDD) • f
CL = Total capacitive load on the I/O pin; f = average frequency the I/O pin switches.
The capacitive currents are most significant when the device is configured for external execution (includes extended microcontroller mode).

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 in hi-impedance state and tied to VDD and VSS.

4: For RC osc configuration, current through Rext is not included. The current through the resistor can be estimated by the formula IR = VDD/2Rext (mA) with Rext in kOhm.

NOTES:

L

M

U

W

NOTES: