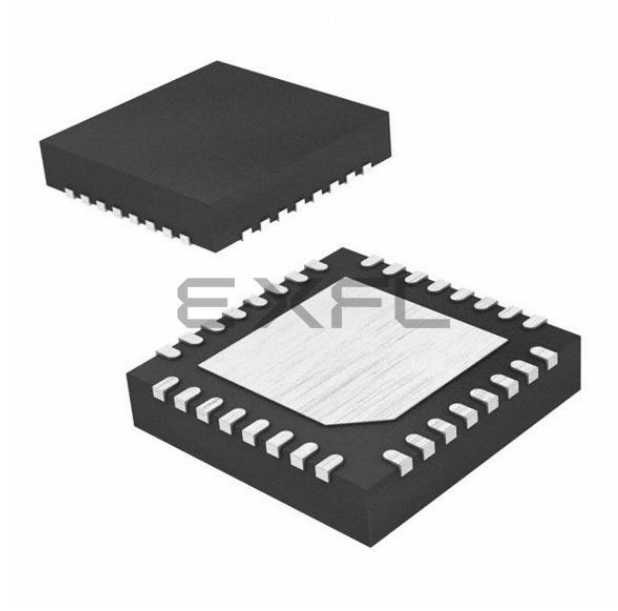


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

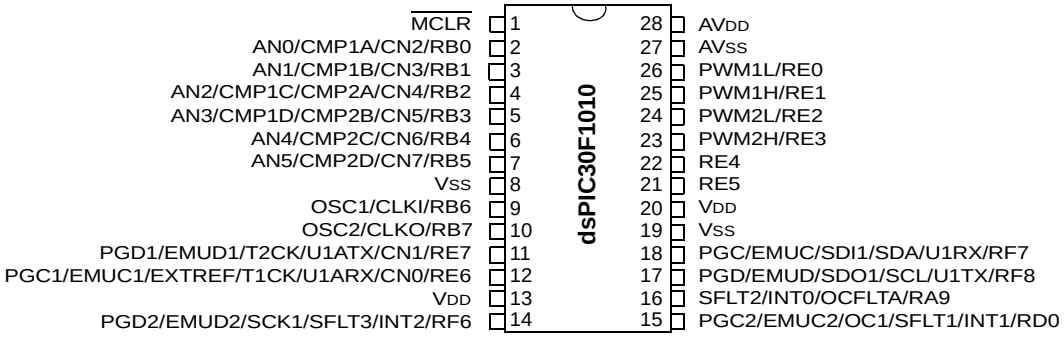
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" 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"

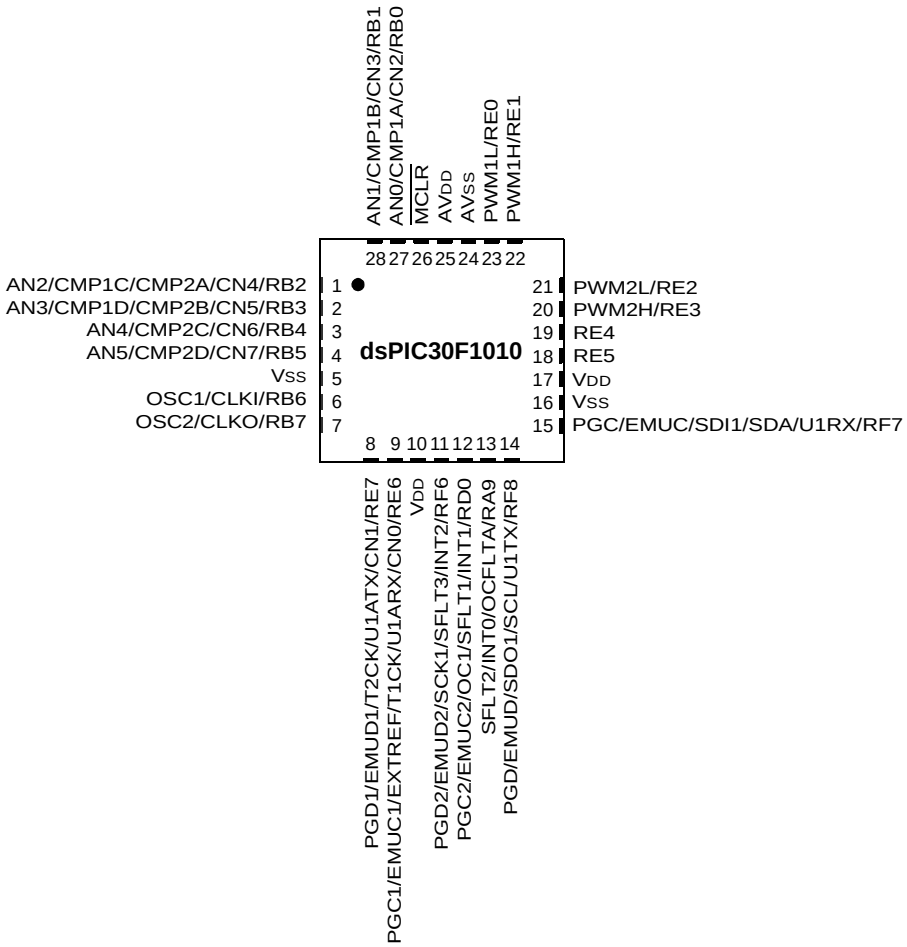
Details	
Product Status	Active
Core Processor	dsPIC
Core Size	16-Bit
Speed	30 MIPs
Connectivity	I²C, IrDA, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	21
Program Memory Size	12KB (4K x 24)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-VQFN Exposed Pad
Supplier Device Package	28-QFN-S (6x6)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/dspic30f2020t-30i-mm

Pin Diagrams

28-Pin SDIP and SOIC



28-Pin QFN-S



2.2 Programmer's Model

The programmer's model is shown in Figure 2-1 and consists of 16x16-bit working registers (W0 through W15), 2x40-bit accumulators (ACCA and ACCB), STATUS register (SR), Data Table Page register (TBLPAG), Program Space Visibility Page register (PSVPAG), DO and REPEAT registers (DOSTART, DOEND, DCOUNT and RCOUNT), and Program Counter (PC). The working registers can act as data, address or offset registers. All registers are memory mapped. W0 acts as the W register for file register addressing.

Some of these registers have a shadow register associated with each of them, as shown in Figure 2-1. The shadow register is used as a temporary holding register and can transfer its contents to or from its host register upon the occurrence of an event. None of the shadow registers are accessible directly. The following rules apply for transfer of registers into and out of shadows.

- `PUSH.S` and `POP.S`
  W0, W1, W2, W3, SR (DC, N, OV, Z and C bits only) are transferred.
- `DO` instruction
  DOSTART, DOEND, DCOUNT shadows are pushed on loop start, and popped on loop end.

When a byte operation is performed on a working register, only the Least Significant Byte (LSB) of the target register is affected. However, a benefit of memory mapped working registers is that both the Least and Most Significant Bytes (MSBs) can be manipulated through byte wide data memory space accesses.

2.2.1 SOFTWARE STACK POINTER/ FRAME POINTER

The dsPIC® DSC devices contain a software stack. W15 is the dedicated software Stack Pointer (SP), and will be automatically modified by exception processing and subroutine calls and returns. However, W15 can be referenced by any instruction in the same manner as all other W registers. This simplifies the reading, writing and manipulation of the Stack Pointer (e.g., creating stack frames).

Note:	In order to protect against misaligned stack accesses, W15<0> is always clear.

W15 is initialized to 0x0800 during a Reset. The user may reprogram the SP during initialization to any location within data space.

W14 has been dedicated as a Stack Frame Pointer as defined by the `LNK` and `ULNK` instructions. However, W14 can be referenced by any instruction in the same manner as all other W registers.

2.2.2 STATUS REGISTER

The dsPIC DSC core has a 16-bit STATUS Register (SR), the LSB of which is referred to as the SR Low Byte (SRL) and the MSB as the SR High Byte (SRH). See Figure 2-1 for SR layout.

SRL contains all the MCU ALU operation status flags (including the Z bit), as well as the CPU Interrupt Priority Level Status bits, IPL<2:0>, and the REPEAT active Status bit, RA. During exception processing, SRL is concatenated with the MSB of the PC to form a complete word value, which is then stacked.

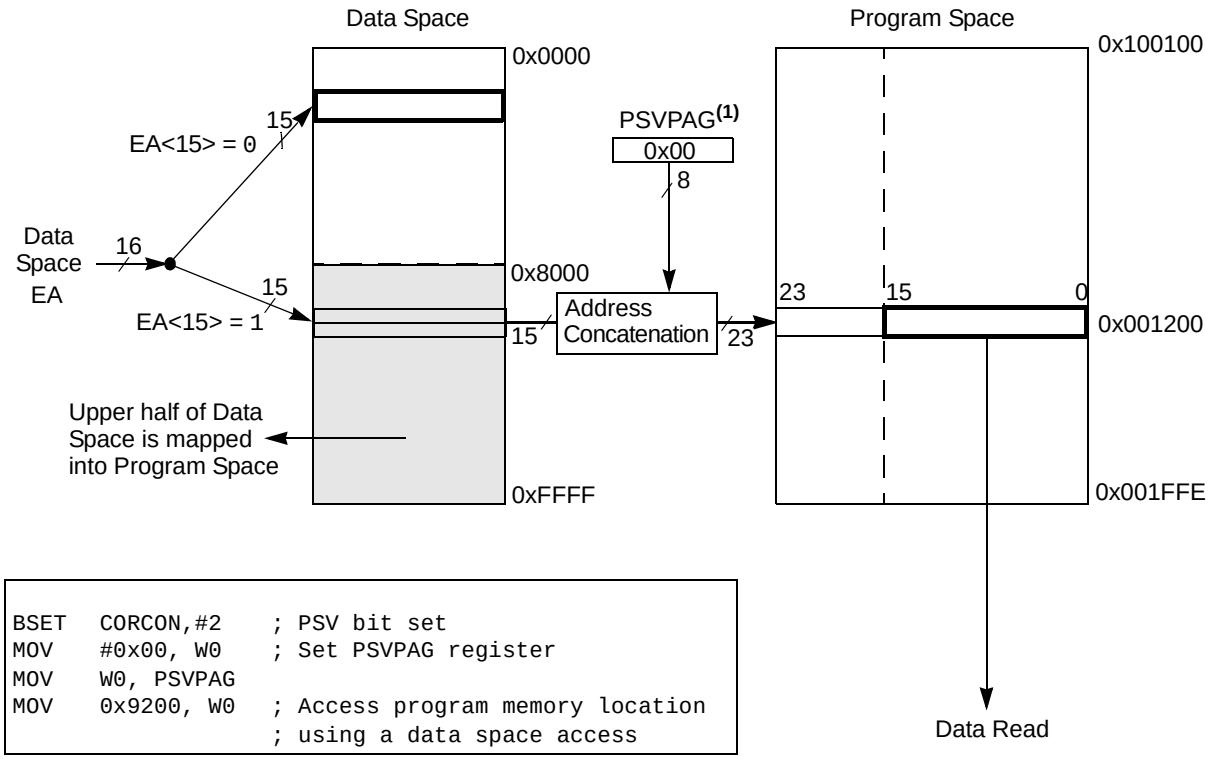
The upper byte of the STATUS register contains the DSP Adder/Subtracter status bits, the DO Loop Active bit (DA) and the Digit Carry (DC) Status bit.

2.2.3 PROGRAM COUNTER

The Program Counter is 23 bits wide. Bit 0 is always clear. Therefore, the PC can address up to 4M instruction words.

NOTES:

FIGURE 3-5: DATA SPACE WINDOW INTO PROGRAM SPACE OPERATION



Note: PSVPAG is an 8-bit register, containing bits <22:15> of the program space address (i.e., it defines the page in program space to which the upper half of data space is being mapped).

3.2 Data Address Space

The core has two data spaces. The data spaces can be considered either separate (for some DSP instructions), or as one unified linear address range (for MCU instructions). The data spaces are accessed using two Address Generation Units (AGUs) and separate data paths.

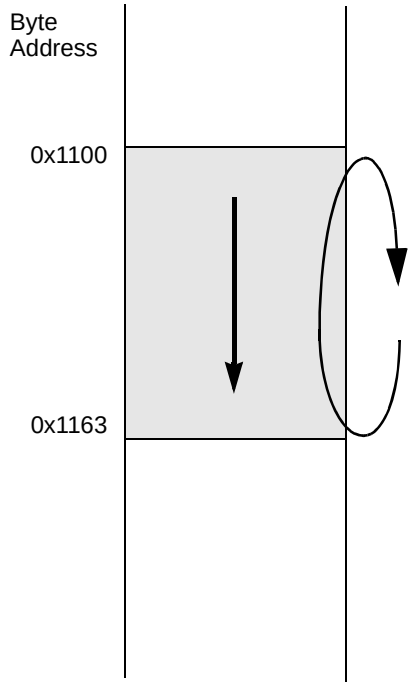
3.2.1 DATA SPACE MEMORY MAP

The data space memory is split into two blocks, X and Y data space. A key element of this architecture is that Y space is a subset of X space, and is fully contained within X space. In order to provide an apparent linear addressing space, X and Y spaces have contiguous addresses.

When executing any instruction other than one of the MAC class of instructions, the X block consists of the 256 byte data address space (including all Y addresses). When executing one of the MAC class of instructions, the X block consists of the 256 bytes data address space excluding the Y address block (for data reads only). In other words, all other instructions regard the entire data memory as one composite address space. The MAC class instructions extract the Y address space from data space and address it using EAs sourced from W10 and W11. The remaining X data space is addressed using W8 and W9. Both address spaces are concurrently accessed only with the MAC class instructions.

A data space memory map is shown in Figure 3-6.

FIGURE 4-1: MODULO ADDRESSING OPERATION EXAMPLE




```
MOV     #0x1100,W0
MOV     W0, XMODSRT     ;set modulo start address
MOV     #0x1163,W0
MOV     W0,MODEND       ;set modulo end address
MOV     #0x8001,W0
MOV     W0,MODCON       ;enable W1, X AGU for modulo
MOV     #0x0000,W0      ;W0 holds buffer fill value
MOV     #0x1110,W1      ;point W1 to buffer
DO      AGAIN,#0x31     ;fill the 50 buffer locations
MOV     W0, [W1++]      ;fill the next location
AGAIN:  INC     W0,W0   ;increment the fill value
```

REGISTER 5-3: IFS0: INTERRUPT FLAG STATUS REGISTER 0 (CONTINUED)

bit 2 **OC1IF:** Output Compare Channel 1 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 1 **IC1IF:** Input Capture Channel 1 Interrupt Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

bit 0 **INT0IF:** External Interrupt 0 Flag Status bit
1 = Interrupt request has occurred
0 = Interrupt request has not occurred

REGISTER 5-14: IPC5: INTERRUPT PRIORITY CONTROL REGISTER 5

U-0	U-0	U-0	U-0	U-0	R/W-1	R/W-0	R/W-0
—	—	—	—	—	PWM4IP<2:0>		
bit 15							bit 8

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	PWM3IP<2:0>			—	PWM2IP<2:0>		
bit 7							bit 0

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

bit 15-11 **Unimplemented:** Read as '0'

bit 10-8 **PWM4IP<2:0>:** PWM Generator #4 Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

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

bit 6-4 **PWM3IP<2:0>:** PWM Generator #3 Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

bit 3 **Unimplemented:** Read as '0'

bit 2-0 **PWM2IP<2:0>:** PWM Generator #2 Interrupt Priority bits
111 = Interrupt is priority 7 (highest priority interrupt)
•
•
•
001 = Interrupt is priority 1
000 = Interrupt source is disabled

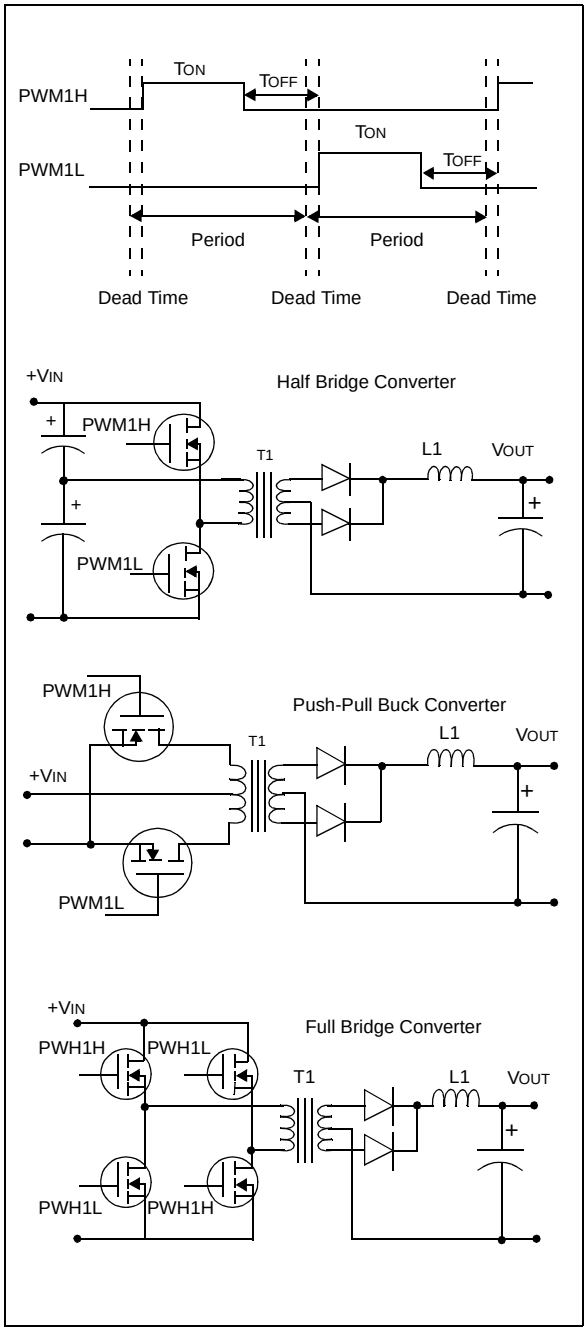
NOTES:

NOTES:

12.34.3 APPLICATION OF PUSH-PULL PWM MODE

Push-Pull PWM mode is typically used in transformer coupled circuits to ensure that no net DC currents flow through the transformer. Push-Pull mode ensures that the same duty cycle PWM pulse is applied to the transformer windings in alternate directions, as shown in Figure 12-24.

FIGURE 12-24: APPLICATIONS OF PUSH-PULL PWM MODE



12.34.4 APPLICATION OF MULTI-PHASE PWM MODE

Multi-Phase PWM mode is often used in DC/DC converters that must handle very fast load current transients and fit into tight spaces. A multi-phase converter is essentially a parallel array of buck converters that are operated slightly out of phase of each other, as shown in Figure 12-25. The multiple phases create an effective switching speed equal to the sum of the individual converters. If a single phase is operating with a 333 KHz PWM frequency, then the effective switching frequency for the circuit is 1 MHz. This high switching frequency greatly reduces output capacitor size requirements and improves load transient response.

FIGURE 12-25: APPLICATIONS OF MULTI-PHASE PWM MODE

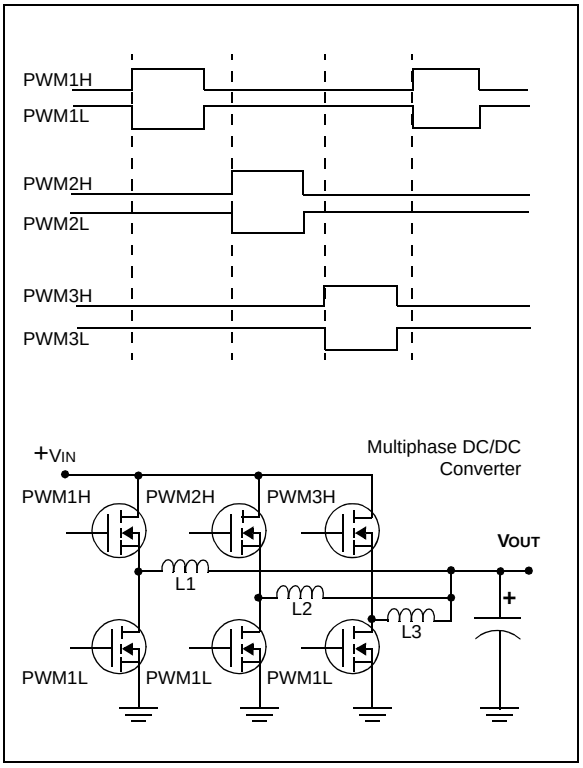


TABLE 12-4: POWER SUPPLY PWM REGISTER MAP

File Name	ADR	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
PTCON	0400	PTEN	—	PTSIDL	SESTAT	SEIEN	EIPU	SYNCPOL	SYNCOEN	SYNCEN	SYNCSRC<2:0>			SEVTPS<3:0>				0000
PTPER	0402	PTPER<15:3>													—	—	—	FFF0
MDC	0404	MDC<15:0>																0000
SEVTCMP	0406	SEVTCMP<15:3>													—	—	—	0000
PWMCON1	0408	FLTSTAT	CLSTAT	TRGSTAT	FLTIEN	CLIEN	TRGIEN	ITB	MDCS	DTC<1:0>		—	—	—	—	XPRES	IUE	0000
IOCON1	040A	PENH	PENL	POLH	POLL	PMOD<1:0>		OVRENH	OVRENL	OVRDAT<1:0>		FLTDAT<1:0>		CLDAT<1:0>		—	OSYNC	0000
FCLCON1	040C	—	—	—	CLSRC<3:0>				CLPOL	CLMOD	FLTSRC<3:0>				FLTPOL	FLTMOD<1:0>		0000
PDC1	040E	PDC1<15:0>																0000
PHASE1	0410	PHASE1<15:2>														—	—	0000
DTR1	0412	—	—	DTR1<13:2>												—	—	0000
ALTDTR1	0414	—	—	ALTDTR1<13:2>												—	—	0000
TRIG1	0416	TRIG<15:3>													—	—	—	0000
TRGCON1	0418	TRGDIV<2:0>			—-	—-	—-	—-	—-	—-	—-	TRGSTRT<5:0>						0000
LEBCON1	041A	PHR	PHF	PLR	PLF	FLTLEBEN	CLLEBEN	LEB<9:3>							—	—	—	0000
PWMCON2	041C	FLTSTAT	CLSTAT	TRGSTAT	FLTIEN	CLIEN	TRGIEN	ITB	MDCS	DTC<1:0>		—	—	—	—	XPRES	IUE	0000
IOCON2	041E	PENH	PENL	POLH	POLL	PMOD<1:0>		OVRENH	OVRENL	OVRDAT<1:0>		FLTDAT<1:0>		CLDAT<1:0>		—	OSYNC	0000
FCLCON2	0420	—	—	—	CLSRC<3:0>				CLPOL	CLMOD	FLTSRC<3:0>				FLTPOL	FLTMOD<1:0>		0000
PDC2	0422	PDC2<15:0>																0000
PHASE2	0424	PHASE2<15:2>														—	—	0000
DTR2	0426	—	—	DTR2<13:2>												—	—	0000
ALTDTR2	0428	—	—	ALTDTR2<13:2>												—	—	0000
TRIG2	042A	TRIG<15:3>													—	—	—	0000
TRGCON2	042C	TRGDIV<2:0>			—-	—-	—-	—-	—-	—-	—-	TRGSTRT<5:0>						0000
LEBCON2	042E	PHR	PHF	PLR	PLF	FLTLEBEN	CLLEBEN	LEB<9:3>							—	—	—	0000
PWMCON3	0430	FLTSTAT	CLSTAT	TRGSTAT	FLTIEN	CLIEN	TRGIEN	ITB	MDCS	DTC<1:0>		—	—	—	—	XPRES	IUE	0000
IOCON3	0432	PENH	PENL	POLH	POLL	PMOD<1:0>		OVRENH	OVRENL	OVRDAT<1:0>		FLTDAT<1:0>		CLDAT<1:0>		—	OSYNC	0000
FCLCON3	0434	—	—	—	CLSRC<3:0>				CLPOL	CLMOD	FLTSRC<3:0>				FLTPOL	FLTMOD<1:0>		0000
PDC3	0436	PDC3<15:0>																0000
PHASE3	0438	PHASE3<15:2>														—	—	0000
DTR3	043A	—	—	DTR3<13:2>												—	—	0000
ALTDTR3	043C	—	—	ALTDTR3<13:2>												—	—	0000
TRIG3	043E	TRIG<15:3>													—	—	—	0000
TRGCON3	0440	TRGDIV<2:0>			—-	—-	—-	—-	—-	—-	—-	TRGSTRT<5:0>						0000
LEBCON3	0442	PHR	PHF	PLR	PLF	FLTLEBEN	CLLEBEN	LEB<9:3>							—	—	—	0000
PWMCON4	0444	FLTSTAT	CLSTAT	TRGSTAT	FLTIEN	CLIEN	TRGIEN	ITB	MDCS	DTC<1:0>		—	—	—	—	XPRES	IUE	0000
IOCON4	0446	PENH	PENL	POLH	POLL	PMOD<1:0>		OVRENH	OVRENL	OVRDAT<1:0>		FLTDAT<1:0>		CLDAT<1:0>		—	OSYNC	0000

REGISTER 13-3: SPIxCON2: SPIx CONTROL REGISTER 2

R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0
FRMEN	SPIFSD	FRMPOL	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	U-0
—	—	—	—	—	—	FRMDLY	—
bit 7							bit 0

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

bit 15 **FRMEN**: Framed SPIx Support bit
1 = Framed SPIx support enabled ($\overline{SS}$x pin used as frame sync pulse input/output)
0 = Framed SPIx support disabled

bit 14 **SPIFSD**: Frame Sync Pulse Direction Control bit
1 = Frame sync pulse input (slave)
0 = Frame sync pulse output (master)

bit 13 **FRMPOL**: Frame Sync Pulse Polarity bit
1 = Frame sync pulse is active-high
0 = Frame sync pulse is active-low

bit 12-2 **Unimplemented:** Read as ‘0’

bit 1 **FRMDLY**: Frame Sync Pulse Edge Select bit
1 = Frame sync pulse coincides with first bit clock
0 = Frame sync pulse precedes first bit clock

bit 0 **Unimplemented:** This bit must not be set to ‘1’ by the user application.

REGISTER 16-2: A/D STATUS REGISTER (ADSTAT)

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	R/C-0 H-S	R/C-0 H-S	R/C-0 H-S	R/C-0 H-S	R/C-0 H-S	R/C-0 H-S
—	—	P5RDY	P4RDY	P3RDY	P2RDY	P1RDY	P0RDY
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown
C = Clear in software	H-S = Set by hardware		

bit 15-6 **Unimplemented:** Read as '0'

bit 5 **P5RDY:** Conversion Data for Pair #5 Ready bit
Bit set when data is ready in buffer, cleared when a '0' is written to this bit.

bit 4 **P4RDY:** Conversion Data for Pair #4 Ready bit
Bit set when data is ready in buffer, cleared when a '0' is written to this bit.

bit 3 **P3RDY:** Conversion Data for Pair #3 Ready bit
Bit set when data is ready in buffer, cleared when a '0' is written to this bit.

bit 2 **P2RDY:** Conversion Data for Pair #2 Ready bit
Bit set when data is ready in buffer, cleared when a '0' is written to this bit.

bit 1 **P1RDY:** Conversion Data for Pair #1 Ready bit
Bit set when data is ready in buffer, cleared when a '0' is written to this bit.

bit 0 **P0RDY:** Conversion Data for Pair #0 Ready bit
Bit set when data is ready in buffer, cleared when a '0' is written to this bit.

REGISTER 18-1: OSCCON: OSCILLATOR CONTROL REGISTER

U-0	R-y, HS, HC	R-y, HS, HC	R-y, HS, HC	U-0	R/W-y	R/W-y	R/W-y
—	COSC<2:0>			—	NOSC<2:0>		
bit 15							bit 8

R/W-0	U-0	R-0, HS,HC	R/W-0	R/C-0, HS, HC	R/W-0	U-0	R/W-0, HC
CLKLOCK	—	LOCK	PRCDEN	CF	TSEQEN	—	OSWEN
bit 7							bit 0

Legend:	x = Bit is unknown	
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
-n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared
HC = Cleared by hardware	HS = Set by hardware	-y = Value set from Configuration bits on POR

bit 15 **Unimplemented:** Read as ‘0’

bit 14-12 **COSC<2:0>:** Current Oscillator Group Selection bits (read-only)
000 = Fast RC Oscillator (FRC)
001 = Fast RC Oscillator (FRC) with PLL Module
010 = Primary Oscillator (HS, EC)
011 = Primary Oscillator (HS, EC) with PLL Module
100 = Reserved
101 = Reserved
110 = Reserved
111 = Reserved
This bit is Reset upon:
Set to FRC value (‘000’) on POR
Loaded with NOSC<2:0> at the completion of a successful clock switch
Set to FRC value (‘000’) when FSCM detects a failure and switches clock to FRC

bit 11 **Unimplemented:** Read as ‘0’

bit 10-8 **NOSC<2:0>:** New Oscillator Group Selection bits
000 = Fast RC Oscillator (FRC)
001 = Fast RC Oscillator (FRC) with PLL Module
010 = Primary Oscillator (HS, EC)
011 = Primary Oscillator (HS, EC) with PLL Module
100 = Reserved
101 = Reserved
110 = Reserved
111 = Reserved

bit 7 **CLKLOCK:** Clock Lock Enabled bit
1 = If (FCKSM1 = 1), then clock and PLL configurations are locked
If (FCKSM1 = 0), then clock and PLL configurations may be modified
0 = Clock and PLL selection are not locked, configurations may be modified

Note: Once set, this bit can only be cleared via a Reset.

bit 6 **Unimplemented:** Read as ‘0’

Most single-word instructions are executed in a single instruction cycle, unless a conditional test is true or the Program Counter is changed as a result of the instruction. In these cases, the execution takes two instruction cycles with the additional instruction cycle(s) executed as a NOP. Notable exceptions are the BRA (unconditional/computed branch), indirect CALL/GOTO, all Table Reads and Writes and RETURN/RETFIE instructions, which are single-word instructions, but take two or three cycles. Certain instructions that involve skipping over the subsequent instruction, require either two or three cycles if the skip is performed, depending on whether the instruction being skipped is a single-word or two-word instruction. Moreover, double word moves require two cycles. The double word instructions execute in two instruction cycles.

Note: For more details on the instruction set, refer to the "*dsPIC30F/33F Programmer's Reference Manual*" (DS70157).

TABLE 19-1: SYMBOLS USED IN OPCODE DESCRIPTIONS

Field	Description
#text	Means literal defined by "text"
(text)	Means "content of text"
[text]	Means "the location addressed by text"
{ }	Optional field or operation
<n:m>	Register bit field
.b	Byte mode selection
.d	Double Word mode selection
.S	Shadow register select
.w	Word mode selection (default)
Acc	One of two accumulators {A, B}
AWB	Accumulator write back destination address register ∈ {W13, [W13] + = 2}
bit4	4-bit bit selection field (used in word addressed instructions) ∈ {0...15}
C, DC, N, OV, Z	MCU Status bits: Carry, Digit Carry, Negative, Overflow, Zero
Expr	Absolute address, label or expression (resolved by the linker)
f	File register address ∈ {0x0000...0x1FFF}
lit1	1-bit unsigned literal ∈ {0,1}
lit4	4-bit unsigned literal ∈ {0...15}
lit5	5-bit unsigned literal ∈ {0...31}
lit8	8-bit unsigned literal ∈ {0...255}
lit10	10-bit unsigned literal ∈ {0...255} for Byte mode, {0:1023} for Word mode
lit14	14-bit unsigned literal ∈ {0...16384}
lit16	16-bit unsigned literal ∈ {0...65535}
lit23	23-bit unsigned literal ∈ {0...8388608}; LSB must be '0'
None	Field does not require an entry, may be blank
OA, OB, SA, SB	DSP Status bits: ACCA Overflow, ACCB Overflow, ACCA Saturate, ACCB Saturate
PC	Program Counter
Slit10	10-bit signed literal ∈ {-512...511}
Slit16	16-bit signed literal ∈ {-32768...32767}
Slit6	6-bit signed literal ∈ {-16...16}

20.2 MPLAB XC Compilers

The MPLAB XC Compilers are complete ANSI C compilers for all of Microchip's 8, 16 and 32-bit MCU and DSC devices. These compilers provide powerful integration capabilities, superior code optimization and ease of use. MPLAB XC Compilers run on Windows, Linux or MAC OS X.

For easy source level debugging, the compilers provide debug information that is optimized to the MPLAB X IDE.

The free MPLAB XC Compiler editions support all devices and commands, with no time or memory restrictions, and offer sufficient code optimization for most applications.

MPLAB XC Compilers include an assembler, linker and utilities. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. MPLAB XC Compiler uses the assembler to produce its object file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command-line interface
- Rich directive set
- Flexible macro language
- MPLAB X IDE compatibility

20.3 MPASM Assembler

The MPASM Assembler is a full-featured, universal macro assembler for PIC10/12/16/18 MCUs.

The MPASM Assembler generates relocatable object files for the MPLINK Object Linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, absolute LST files that contain source lines and generated machine code, and COFF files for debugging.

The MPASM Assembler features include:

- Integration into MPLAB X IDE projects
- User-defined macros to streamline assembly code
- Conditional assembly for multipurpose source files
- Directives that allow complete control over the assembly process

20.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK Object Linker combines relocatable objects created by the MPASM Assembler. It can link relocatable objects from precompiled libraries, using directives from a linker script.

The MPLIB Object Librarian manages the creation and modification of library files of precompiled code. When a routine from a library is called from a 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 object linker/library features include:

- Efficient linking of single libraries instead of many smaller files
- Enhanced code maintainability by grouping related modules together
- Flexible creation of libraries with easy module listing, replacement, deletion and extraction

20.5 MPLAB Assembler, Linker and Librarian for Various Device Families

MPLAB Assembler produces relocatable machine code from symbolic assembly language for PIC24, PIC32 and dsPIC DSC devices. MPLAB XC Compiler uses the assembler to produce its object file. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command-line interface
- Rich directive set
- Flexible macro language
- MPLAB X IDE compatibility

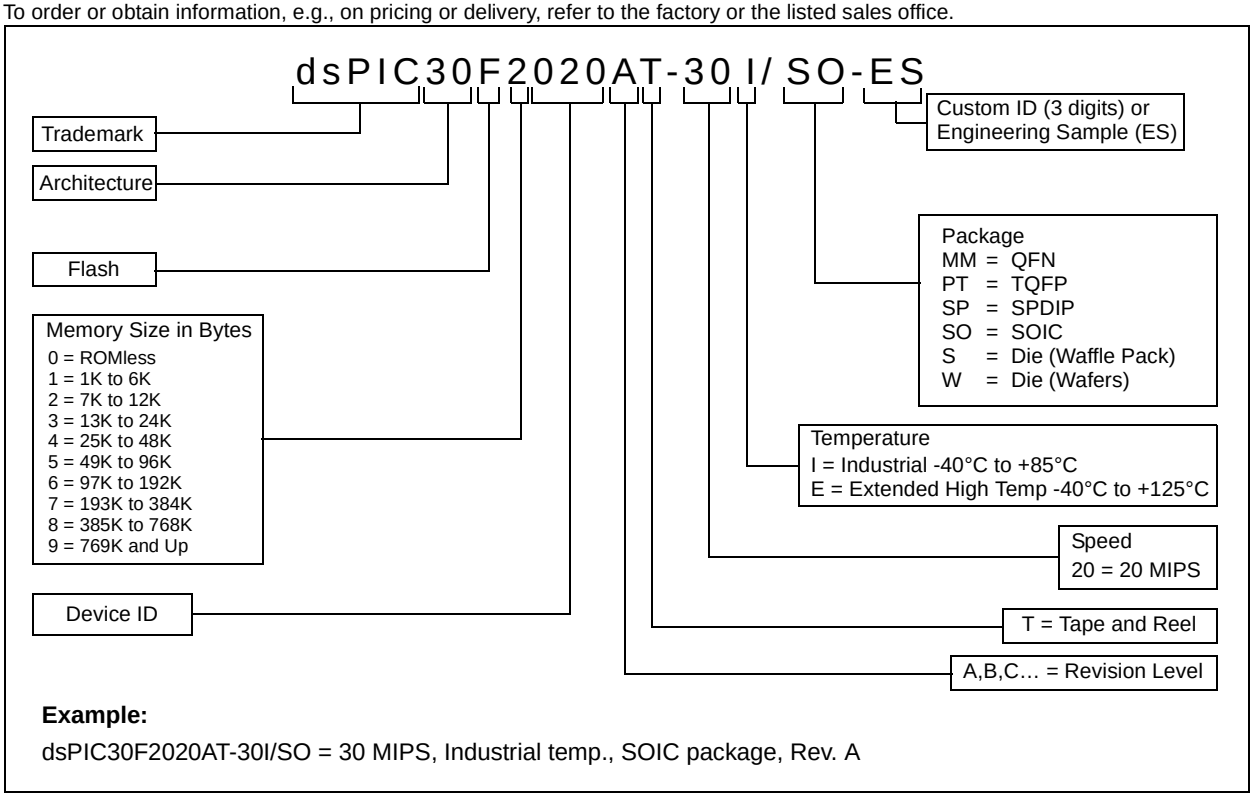
NOTES:

U

W

PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.



Example:

dsPIC30F2020AT-30I/SO = 30 MIPS, Industrial temp., SOIC package, Rev. A

NOTES: