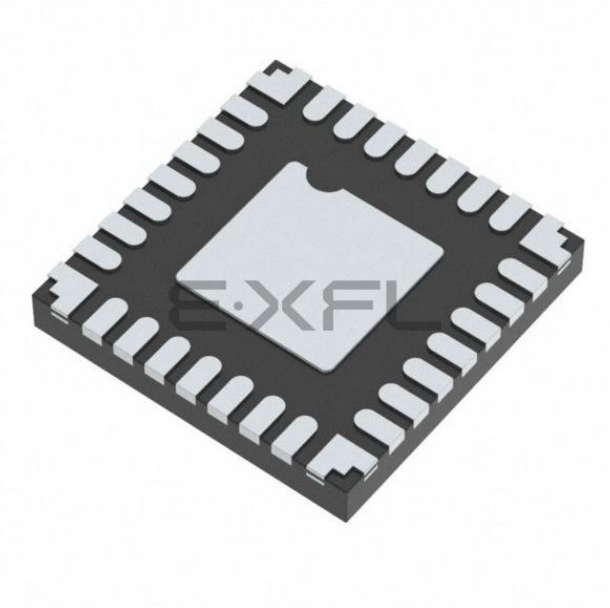


Welcome to E-XFL.COM

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.

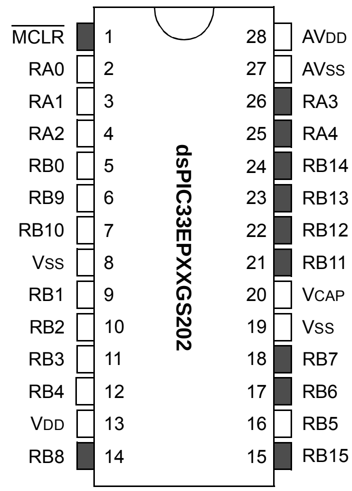
Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	dsPIC
Core Size	16-Bit
Speed	60 MIPs
Connectivity	I²C, IrDA, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	21
Program Memory Size	16KB (16K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 3.6V
Data Converters	A/D 12x12b; D/A 2x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	28-UFQFN Exposed Pad
Supplier Device Package	28-UQFN (4x4)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/dspic33ep16gs202-e-m6

Pin Diagrams

28-Pin SOIC,
28-Pin SSOP

■ = Pins are up to 5V tolerant

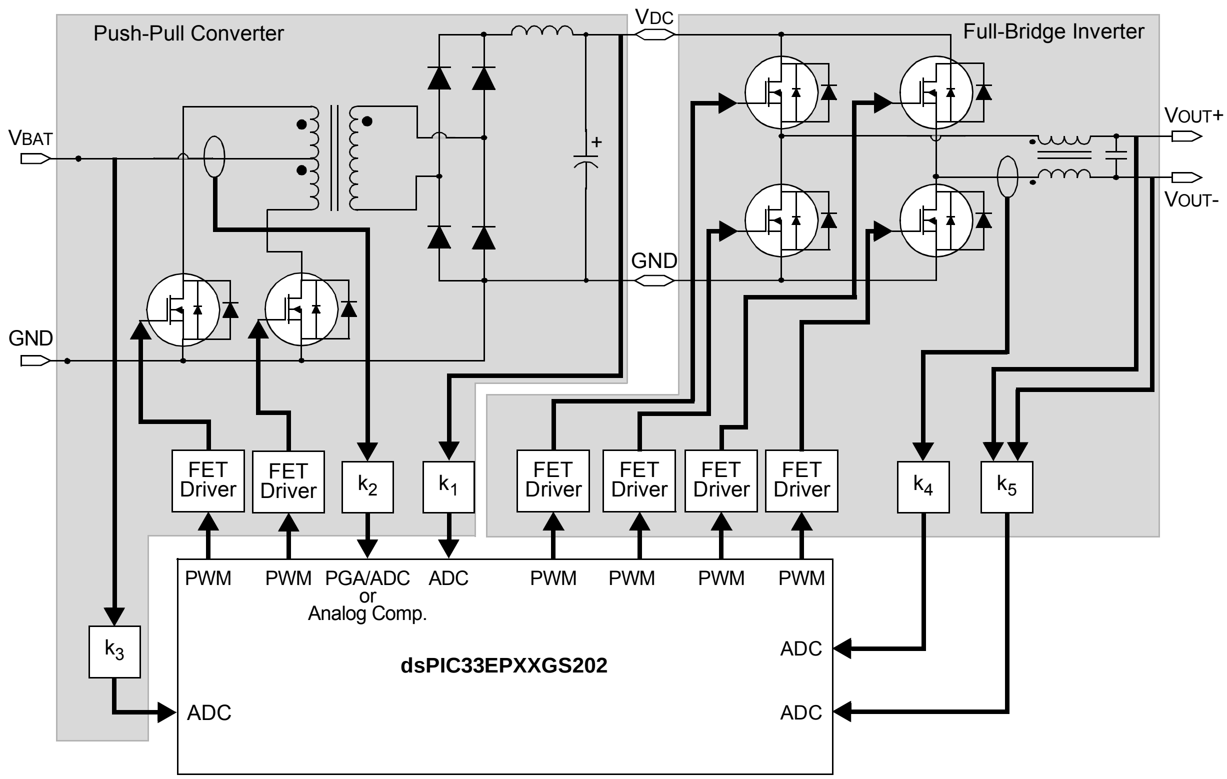


PIN FUNCTION DESCRIPTIONS

Pin	Pin Function	Pin	Pin Function
1	MCLR	15	PGEC3/**RP47**/RB15
2	AN0/PGA1P1/CMP1A/RA0	16	TDO/AN9/PGA2N2/**RP37**/RB5
3	AN1/PGA1P2/PGA2P1/CMP1B/RA1	17	PGED1/TDI/AN10/SCL1/**RP38**/RB6
4	AN2/PGA1P3/PGA2P2/CMP1C/CMP2A/RA2	18	PGEC1/AN11/SDA1/**RP39**/RB7
5	AN3/PGA2P3/CMP1D/CMP2B/**RP32**/RB0	19	VSS
6	AN4/CMP2C/**RP41**/RB9	20	VCAP
7	AN5/CMP2D/**RP42**/RB10	21	TMS/PWM3H/**RP43**/RB11
8	VSS	22	TCK/PWM3L/**RP44**/RB12
9	OSC1/CLKI/AN6/**RP33**/RB1	23	PWM2H/**RP45**/RB13
10	OSC2/CLKO/AN7/PGA1N2/**RP34**/RB2	24	PWM2L/**RP46**/RB14
11	PGED2/AN8/INT0/**RP35**/RB3	25	PWM1H/RA4
12	PGEC2/ADTRG31/**RP36**/RB4	26	PWM1L/RA3
13	VDD	27	AVSS
14	PGED3/**RP40**/RB8	28	AVDD

Legend: Shaded pins are up to 5 VDC tolerant.
Note: **RPn** represents remappable peripheral functions. See Table 10-1 and Table 10-2 for the complete list of remappable sources.

FIGURE 2-6: OFF-LINE UPS



Note: k_1, k_2, k_3, k_4 and k_5 are gains.

3.8 Arithmetic Logic Unit (ALU)

The dsPIC33EPXXGS202 family ALU is 16 bits wide and is capable of addition, subtraction, bit shifts and logic operations. Unless otherwise mentioned, arithmetic operations are two's complement in nature. Depending on the operation, the ALU can affect the values of the Carry (C), Zero (Z), Negative (N), Overflow (OV) and Digit Carry (DC) Status bits in the SR register. The C and DC Status bits operate as Borrow and Digit Borrow bits, respectively, for subtraction operations.

The ALU can perform 8-bit or 16-bit operations, depending on the mode of the instruction that is used. Data for the ALU operation can come from the W register array or data memory, depending on the addressing mode of the instruction. Likewise, output data from the ALU can be written to the W register array or a data memory location.

Refer to the *"16-bit MCU and DSC Programmer's Reference Manual"* (DS70157) for information on the SR bits affected by each instruction.

The core CPU incorporates hardware support for both multiplication and division. This includes a dedicated hardware multiplier and support hardware for 16-bit divisor division.

3.8.1 MULTIPLIER

Using the high-speed 17-bit x 17-bit multiplier, the ALU supports unsigned, signed, or mixed-sign operation in several MCU multiplication modes:

- 16-bit x 16-bit signed
- 16-bit x 16-bit unsigned
- 16-bit signed x 5-bit (literal) unsigned
- 16-bit signed x 16-bit unsigned
- 16-bit unsigned x 5-bit (literal) unsigned
- 16-bit unsigned x 16-bit signed
- 8-bit unsigned x 8-bit unsigned

3.8.2 DIVIDER

The divide block supports 32-bit/16-bit and 16-bit/16-bit signed and unsigned integer divide operations with the following data sizes:

- 32-bit signed/16-bit signed divide
- 32-bit unsigned/16-bit unsigned divide
- 16-bit signed/16-bit signed divide
- 16-bit unsigned/16-bit unsigned divide

The quotient for all divide instructions ends up in W0 and the remainder in W1. Sixteen-bit signed and unsigned `DIV` instructions can specify any W register for both the 16-bit divisor (Wn) and any W register (aligned) pair (W(m + 1):Wm) for the 32-bit dividend. The divide algorithm takes one cycle per bit of divisor, so both 32-bit/16-bit and 16-bit/16-bit instructions take the same number of cycles to execute.

3.9 DSP Engine

The DSP engine consists of a high-speed 17-bit x 17-bit multiplier, a 40-bit barrel shifter and a 40-bit adder/subtracter (with two target accumulators, round and saturation logic).

The DSP engine can also perform inherent accumulator-to-accumulator operations that require no additional data. These instructions are `ADD`, `SUB` and `NEG`.

The DSP engine has options selected through bits in the CPU Core Control register (CORCON), as listed below:

- Fractional or Integer DSP Multiply (IF)
- Signed, unsigned or mixed-sign DSP multiply (USx)
- Conventional or Convergent Rounding (RND)
- Automatic Saturation On/Off for ACCA (SATA)
- Automatic Saturation On/Off for ACCB (SATB)
- Automatic Saturation On/Off for Writes to Data Memory (SATDW)
- Accumulator Saturation mode Selection (ACCSAT)

TABLE 3-2: DSP INSTRUCTIONS SUMMARY

Instruction	Algebraic Operation	ACC Write Back
`CLR`	$A = 0$	Yes
`ED`	$A = (x - y)^2$	No
`EDAC`	$A = A + (x - y)^2$	No
`MAC`	$A = A + (x \bullet y)$	Yes
`MAC`	$A = A + x^2$	No
`MOVSAC`	No change in A	Yes
`MPY`	$A = x \bullet y$	No
`MPY`	$A = x^2$	No
`MPY.N`	$A = - x \bullet y$	No
`MSC`	$A = A - x \bullet y$	Yes

When a PSV page overflow or underflow occurs, EA<15> is cleared as a result of the register indirect EA calculation. An overflow or underflow of the EA in the PSV pages can occur at the page boundaries when:

- The initial address, prior to modification, addresses the PSV page
- The EA calculation uses Pre- or Post-Modified Register Indirect Addressing; however, this does not include Register Offset Addressing

In general, when an overflow is detected, the DSRPAG register is incremented and the EA<15> bit is set to keep the base address within the PSV window. When an underflow is detected, the DSRPAG register is decremented and the EA<15> bit is set to keep the base address within the PSV window. This creates a linear PSV address space, but only when using Register Indirect Addressing modes.

Exceptions to the operation described above arise when entering and exiting the boundaries of Page 0 and PSV spaces. Table 4-24 lists the effects of overflow and underflow scenarios at different boundaries.

In the following cases, when overflow or underflow occurs, the EA<15> bit is set and the DSRPAG is not modified; therefore, the EA will wrap to the beginning of the current page:

- Register Indirect with Register Offset Addressing
- Modulo Addressing
- Bit-Reversed Addressing

TABLE 4-24: OVERFLOW AND UNDERFLOW SCENARIOS AT PAGE 0 AND PSV SPACE BOUNDARIES(2,3,4)

O/U, R/W	Operation	Before			After		
		DSxPAG	**DS EA<15>**	**Page Description**	**DSxPAG**	**DS EA<15>**	**Page Description**
O, Read	`[++Wn]` or `[Wn++]`	DSRPAG = 0x2FF	`1`	PSV: Last lsw page	DSRPAG = 0x300	`1`	PSV: First MSB page
O, Read		DSRPAG = 0x3FF	`1`	PSV: Last MSB page	DSRPAG = 0x3FF	`0`	See **Note 1**
U, Read	`[--Wn]` or `[Wn--]`	DSRPAG = 0x001	`1`	PSV page	DSRPAG = 0x001	`0`	See **Note 1**
U, Read		DSRPAG = 0x200	`1`	PSV: First lsw page	DSRPAG = 0x200	`0`	See **Note 1**
U, Read		DSRPAG = 0x300	`1`	PSV: First MSB page	DSRPAG = 0x2FF	`1`	PSV: Last lsw page

Legend: O = Overflow, U = Underflow, R = Read, W = Write

Note 1: The Register Indirect Addressing now addresses a location in the base Data Space (0x0000-0x7FFF).

2: An EDS access, when DSRPAG = 0x000, will generate an address error trap.

3: Only reads from PS are supported using DSRPAG.

4: Pseudolinear Addressing is not supported for large offsets.

REGISTER 5-5: NVMSRCADRL: NVM SOURCE DATA ADDRESS LOW REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
NVMSRCADR<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
NVMSRCADR<7:0>							
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 15-0 **NVMSRCADR<15:0>:** Source Data Address bits
The RAM address of the data to be programmed into Flash when the NVMOP<3:0> bits are set to row programming.

REGISTER 5-6: NVMSRCADRH: NVM SOURCE DATA ADDRESS HIGH REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
NVMSRCADR<31:24>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
NVMSRCADR<23:16>							
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 15-0 **NVMSRCADR<31:16>:** Source Data Address bits
The RAM address of the data to be programmed into Flash when the NVMOP<3:0> bits are set to row programming. These bits must be always programmed to zero.

REGISTER 10-9: RPINR18: PERIPHERAL PIN SELECT INPUT REGISTER 18

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
U1CTSR7	U1CTSR6	U1CTSR5	U1CTSR4	U1CTSR3	U1CTSR2	U1CTSR1	U1CTSR0
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
U1RXR7	U1RXR6	U1RXR5	U1RXR4	U1RXR3	U1RXR2	U1RXR1	U1RXR0
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 15-8 **U1CTSR<7:0>:** Assign UART1 Clear-to-Send ($\overline{\text{U1CTS}}$) to the Corresponding RPn Pin bits

`10110101` = Input tied to RP181
`10110100` = Input tied to RP180
•
•
•
`00000001` = Input tied to RP1
`00000000` = Input tied to VSS

bit 7-0 **U1RXR<7:0>:** Assign UART1 Receive (U1RX) to the Corresponding RPn Pin bits

`10110101` = Input tied to RP181
`10110100` = Input tied to RP180
•
•
•
`00000001` = Input tied to RP1
`00000000` = Input tied to VSS

REGISTER 10-14: RPINR42: PERIPHERAL PIN SELECT INPUT REGISTER 42

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
FLT6R7	FLT6R6	FLT6R5	FLT6R4	FLT6R3	FLT6R2	FLT6R1	FLT6R0
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
FLT5R7	FLT5R6	FLT5R5	FLT5R4	FLT5R3	FLT5R2	FLT5R1	FLT5R0
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 15-8 **FLT6R<7:0>:** Assign PWM Fault 6 (FLT6) to the Corresponding RPn Pin bits
10110101 = Input tied to RP181
10110100 = Input tied to RP180
•
•
•
00000001 = Input tied to RP1
00000000 = Input tied to VSS

bit 7-0 **FLT5R<7:0>:** Assign PWM Fault 5 (FLT5) to the Corresponding RPn Pin bits
10110101 = Input tied to RP181
10110100 = Input tied to RP180
•
•
•
00000001 = Input tied to RP1
00000000 = Input tied to VSS

REGISTER 10-18: RPOR2: PERIPHERAL PIN SELECT OUTPUT REGISTER 2

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	RP37R5	RP37R4	RP37R3	RP37R2	RP37R1	RP37R0
bit 15							bit 8

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	RP36R5	RP36R4	RP36R3	RP36R2	RP36R1	RP36R0
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 15-14	**Unimplemented:** Read as ‘0’
bit 13-8	**RP37R<5:0>:** Peripheral Output Function is Assigned to RP37 Output Pin bits (see Table 10-2 for peripheral function numbers)
bit 7-6	**Unimplemented:** Read as ‘0’
bit 5-0	**RP36R<5:0>:** Peripheral Output Function is Assigned to RP36 Output Pin bits (see Table 10-2 for peripheral function numbers)

REGISTER 10-19: RPOR3: PERIPHERAL PIN SELECT OUTPUT REGISTER 3

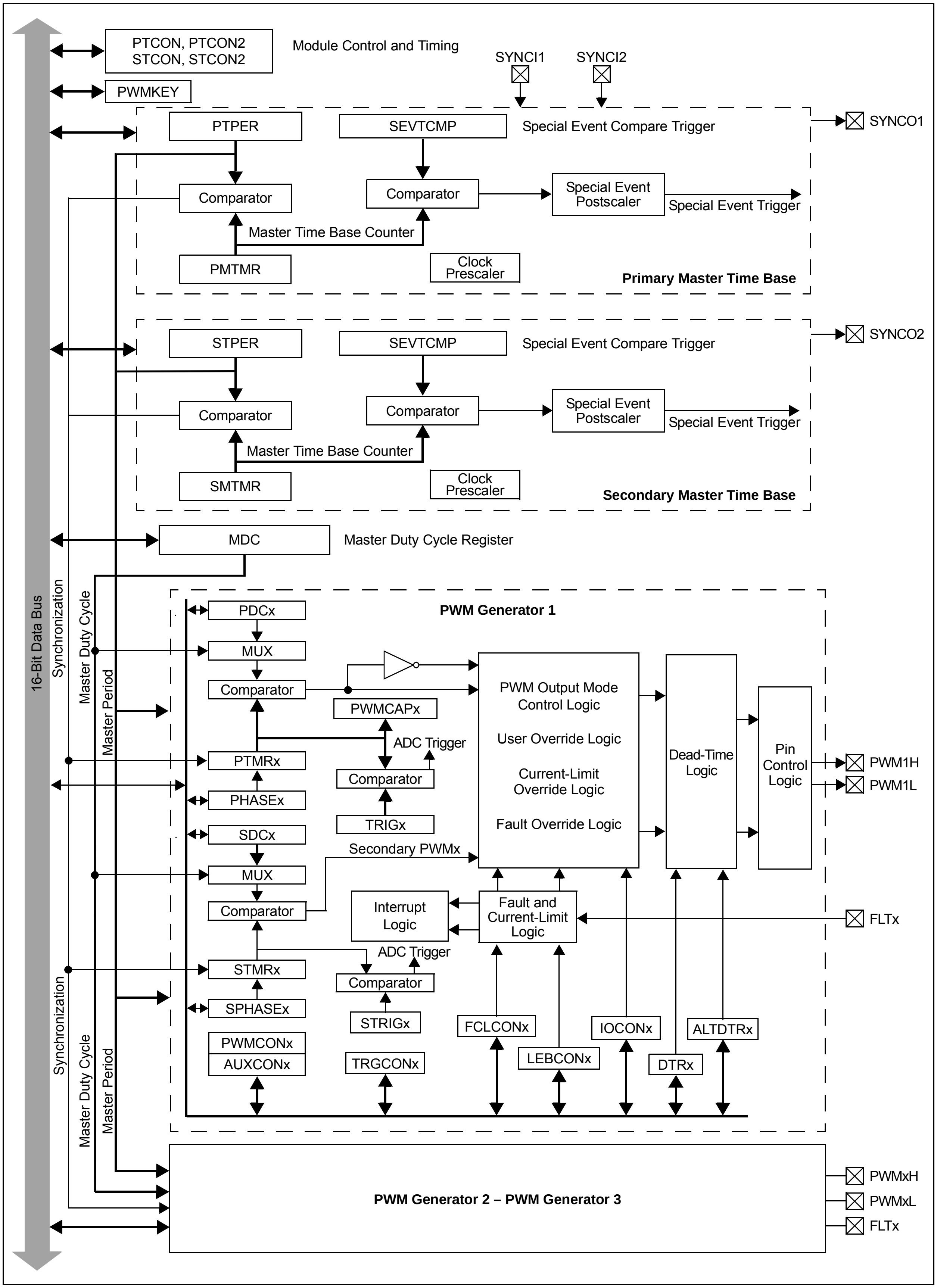
U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	RP39R5	RP39R4	RP39R3	RP39R2	RP39R1	RP39R0
bit 15							bit 8

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	RP38R5	RP38R4	RP38R3	RP38R2	RP38R1	RP38R0
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 15-14	**Unimplemented:** Read as ‘0’
bit 13-8	**RP39R<5:0>:** Peripheral Output Function is Assigned to RP39 Output Pin bits (see Table 10-2 for peripheral function numbers)
bit 7-6	**Unimplemented:** Read as ‘0’
bit 5-0	**RP38R<5:0>:** Peripheral Output Function is Assigned to RP38 Output Pin bits (see Table 10-2 for peripheral function numbers)

FIGURE 15-2: SIMPLIFIED CONCEPTUAL BLOCK DIAGRAM OF THE HIGH-SPEED PWM



REGISTER 15-19: TRGCONx: PWMx TRIGGER CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0
TRGDIV3	TRGDIV2	TRGDIV1	TRGDIV0	—	—	—	—
bit 15							bit 8

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
DTM(1)	—	TRGSTRT5	TRGSTRT4	TRGSTRT3	TRGSTRT2	TRGSTRT1	TRGSTRT0
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 15-12 **TRGDIV<3:0>:** Trigger # Output Divider bits

1111 = Trigger output for every 16th trigger event
1110 = Trigger output for every 15th trigger event
1101 = Trigger output for every 14th trigger event
1100 = Trigger output for every 13th trigger event
1011 = Trigger output for every 12th trigger event
1010 = Trigger output for every 11th trigger event
1001 = Trigger output for every 10th trigger event
1000 = Trigger output for every 9th trigger event
0111 = Trigger output for every 8th trigger event
0110 = Trigger output for every 7th trigger event
0101 = Trigger output for every 6th trigger event
0100 = Trigger output for every 5th trigger event
0011 = Trigger output for every 4th trigger event
0010 = Trigger output for every 3rd trigger event
0001 = Trigger output for every 2nd trigger event
0000 = Trigger output for every trigger event

bit 11-8 **Unimplemented:** Read as ‘0’

bit 7 **DTM:** Dual Trigger Mode bit(1)

1 = Secondary trigger event is combined with the primary trigger event to create a PWM trigger
0 = Secondary trigger event is not combined with the primary trigger event to create a PWM trigger; two separate PWM triggers are generated

bit 6 **Unimplemented:** Read as ‘0’

bit 5-0 **TRGSTRT<5:0>:** Trigger Postscaler Start Enable Select bits

111111 = Wait 63 PWM cycles before generating the first trigger event after the module is enabled
•
•
•
000010 = Wait 2 PWM cycles before generating the first trigger event after the module is enabled
000001 = Wait 1 PWM cycle before generating the first trigger event after the module is enabled
000000 = Wait 0 PWM cycles before generating the first trigger event after the module is enabled

Note 1: The secondary PWMx generator cannot generate PWM trigger interrupts.

REGISTER 15-22: FCLCONx: PWMx FAULT CURRENT-LIMIT CONTROL REGISTER (CONTINUED)

bit 7-3 **FLTSRC<4:0>:** Fault Control Signal Source Select for PWMx Generator # bits
11111 = Reserved
10001 = Reserved
10000 = Reserved
01111 = Reserved
01110 = Analog Comparator 2
01101 = Analog Comparator 1
01100 = Reserved
01011 = Reserved
01010 = Reserved
01001 = Reserved
01000 = Fault 8
00111 = Fault 7
00110 = Fault 6
00101 = Fault 5
00100 = Fault 4
00011 = Fault 3
00010 = Fault 2
00001 = Fault 1
00000 = Reserved

bit 2 **FLTPOL:** Fault Polarity for PWMx Generator # bit(1)
1 = The selected Fault source is active-low
0 = The selected Fault source is active-high

bit 1-0 **FLTMOD<1:0>:** Fault Mode for PWMx Generator # bits
11 = Fault input is disabled
10 = Reserved
01 = The selected Fault source forces the PWMxH, PWMxL pins to the FLTDATx values (cycle)
00 = The selected Fault source forces the PWMxH, PWMxL pins to the FLTDATx values (latched condition)

Note 1: These bits should be changed only when PTEN (PTCON<15>) = 0.

REGISTER 15-23: STRIGx: PWMx SECONDARY TRIGGER COMPARE VALUE REGISTER(1)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
STRGCMP<12:5>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0
STRGCMP<4:0>					—	—	—
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 15-3 **STRGCMP<12:0>:** Secondary Trigger Compare Value bits
When the secondary PWMx functions in the local time base, this register contains the compare values that can trigger the ADC module.

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

Note 1: STRIGx cannot generate the PWMx trigger interrupts.

REGISTER 16-2: SPI1CON1: SPI1 CONTROL REGISTER 1 (CONTINUED)

bit 4-2 **SPRE<2:0>:** Secondary Prescale bits (Master mode)(3)

111 = Secondary prescale 1:1
110 = Secondary prescale 2:1
•
•
•
000 = Secondary prescale 8:1

bit 1-0 **PPRE<1:0>:** Primary Prescale bits (Master mode)(3)

11 = Primary prescale 1:1
10 = Primary prescale 4:1
01 = Primary prescale 16:1
00 = Primary prescale 64:1

Note 1: The CKE bit is not used in Framed SPI modes. Program this bit to ‘0’ for Framed SPI modes (FRMEN = 1).
2: This bit must be cleared when FRMEN = 1.
3: Do not set both primary and secondary prescalers to the value of 1:1.

REGISTER 17-1: I2C1CONL: I2C1 CONTROL REGISTER LOW (CONTINUED)

bit 6	**STREN:** SCL1 Clock Stretch Enable bit (when operating as I^2C slave) Used in conjunction with the SCLREL bit. 1 = Enables software or receives clock stretching 0 = Disables software or receives clock stretching
bit 5	**ACKDT:** Acknowledge Data bit (when operating as I^2C master, applicable during master receive) Value that is transmitted when the software initiates an Acknowledge sequence. 1 = Sends NACK during Acknowledge 0 = Sends ACK during Acknowledge
bit 4	**ACKEN:** Acknowledge Sequence Enable bit (when operating as I^2C master, applicable during master receive) 1 = Initiates Acknowledge sequence on the SDA1 and SCL1 pins and transmits the ACKDT data bit. Hardware clears it at the end of the master Acknowledge sequence. 0 = Acknowledge sequence is not in progress
bit 3	**RCEN:** Receive Enable bit (when operating as I^2C master) 1 = Enables Receive mode for I^2C. Hardware clears it at the end of the eighth bit of the master receive data byte. 0 = Receive sequence is not in progress
bit 2	**PEN:** Stop Condition Enable bit (when operating as I^2C master) 1 = Initiates Stop condition on the SDA1 and SCL1 pins. Hardware clears it at the end of the master Stop sequence. 0 = Stop condition is not in progress
bit 1	**RSEN:** Repeated Start Condition Enable bit (when operating as I^2C master) 1 = Initiates Repeated Start condition on the SDA1 and SCL1 pins. Hardware clears it at the end of the master Repeated Start sequence. 0 = Repeated Start condition is not in progress
bit 0	**SEN:** Start Condition Enable bit (when operating as I^2C master) 1 = Initiates Start condition on the SDA1 and SCL1 pins. Hardware clears it at the end of the master Start sequence. 0 = Start condition is not in progress

REGISTER 19-6: ADCON3H: ADC CONTROL REGISTER 3 HIGH

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CLKSEL1	CLKSEL0	CLKDIV5	CLKDIV4	CLKDIV3	CLKDIV2	CLKDIV1	CLKDIV0
bit 15							bit 8

R/W-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
SHREN	—	—	—	—	—	C1EN	C0EN
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 15-14 **CLKSEL<2:0>:** ADC Module Clock Source Selection bits
11 = APLL
10 = FRC
01 = FOSC (System Clock x 2)
00 = FSYS (System Clock)

bit 13-8 **CLKDIV<5:0>:** ADC Module Clock Source Divider bits
The divider forms a TCORESRC clock used by all ADC cores (shared and dedicated) from the TSRC ADC module clock source selected by the CLKSEL<2:0> bits. Then, each ADC core individually divides the TCORESRC clock to get a core-specific TADCORE clock using the ADCS<6:0> bits in the ADCORExH register or the SHRADCS<6:0> bits in the ADCON2L register.
111111 = 64 Core Source Clock periods
•
•
•
000011 = 4 Core Source Clock periods
000010 = 3 Core Source Clock periods
000001 = 2 Core Source Clock periods
000000 = 1 Core Source Clock period

bit 7 **SHREN:** Shared ADC Core Enable bit
This bit does not disable the core clock and analog bias circuitry.
1 = Shared ADC core is enabled
0 = Shared ADC core is disabled

bit 6-2 **Unimplemented:** Read as ‘0’

bit 1-0 **C1EN:C0EN:** Dedicated ADC Core x Enable bits
This bit does not disable the core clock and analog bias circuitry.
1 = Dedicated ADC Core x is enabled
0 = Dedicated ADC Core x is disabled

REGISTER 19-23: ADCAL1H: ADC CALIBRATION REGISTER 1 HIGH

R/W-0, HS	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
CSHRRDY	—	—	—	CSHRSKIP	CSHRDIFF	CSHREN	CSHRRUN
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 7							bit 0

Legend:	HS = Hardware Settable bit		
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 **CSHRRDY:** Shared ADC Core Calibration Status Flag bit
1 = Shared ADC core calibration is finished
0 = Shared ADC core calibration is in progress

bit 14-12 **Unimplemented:** Read as ‘0’

bit 11 **CSHRSKIP:** Shared ADC Core Calibration Bypass bit
1 = After power-up, the shared ADC core will not be calibrated
0 = After power-up, the shared ADC core will be calibrated

bit 10 **CSHRDIFF:** Shared ADC Core Pseudo-Differential Input Mode Calibration bit
1 = Shared ADC core will be calibrated in Pseudo-Differential Input mode
0 = Shared ADC core will be calibrated in Single-Ended Input mode

bit 9 **CSHREN:** Shared ADC Core Calibration Enable bit
1 = Shared ADC core calibration bits (CSHRRDY, CSHRSKIP, CSHRDIFF and CSHRRUN) can be accessed by software
0 = Shared ADC core calibration bits are disabled

bit 8 **CSHRRUN:** Shared ADC Core Calibration Start bit
1 = If this bit is set by software, the shared ADC core calibration cycle is started; this bit is cleared automatically by hardware
0 = Software can start the next calibration cycle

bit 7-0 **Unimplemented:** Read as ‘0’

23.0 INSTRUCTION SET SUMMARY

> **Note:** This data sheet summarizes the features of the dsPIC33EPXXGS202 family of devices. It is not intended to be a comprehensive reference source. To complement the information in this data sheet, refer to the related section in the *"dsPIC33/PIC24 Family Reference Manual"*, which is available from the Microchip web site (www.microchip.com).

The dsPIC33EP instruction set is almost identical to that of the dsPIC30F and dsPIC33F.

Most instructions are a single program memory word (24 bits). Only three instructions require two program memory locations.

Each single-word instruction is a 24-bit word, divided into an 8-bit opcode, which specifies the instruction type and one or more operands, which further specify the operation of the instruction.

The instruction set is highly orthogonal and is grouped into five basic categories:

- Word or byte-oriented operations
- Bit-oriented operations
- Literal operations
- DSP operations
- Control operations

Table 23-1 lists the general symbols used in describing the instructions.

The dsPIC33EP instruction set summary in Table 23-2 lists all the instructions, along with the status flags affected by each instruction.

Most word or byte-oriented W register instructions (including barrel shift instructions) have three operands:

- The first source operand, which is typically a register ‘Wb’ without any address modifier
- The second source operand, which is typically a register ‘Ws’ with or without an address modifier
- The destination of the result, which is typically a register ‘Wd’ with or without an address modifier

However, word or byte-oriented file register instructions have two operands:

- The file register specified by the value ‘f’
- The destination, which could be either the file register ‘f’ or the W0 register, which is denoted as ‘WREG’

Most bit-oriented instructions (including simple rotate/shift instructions) have two operands:

- The W register (with or without an address modifier) or file register (specified by the value of ‘Ws’ or ‘f’)
- The bit in the W register or file register (specified by a literal value or indirectly by the contents of register ‘Wb’)

The literal instructions that involve data movement can use some of the following operands:

- A literal value to be loaded into a W register or file register (specified by ‘k’)
- The W register or file register where the literal value is to be loaded (specified by ‘Wb’ or ‘f’)

However, literal instructions that involve arithmetic or logical operations use some of the following operands:

- The first source operand, which is a register ‘Wb’ without any address modifier
- The second source operand, which is a literal value
- The destination of the result (only if not the same as the first source operand), which is typically a register ‘Wd’ with or without an address modifier

The `MAC` class of DSP instructions can use some of the following operands:

- The accumulator (A or B) to be used (required operand)
- The W registers to be used as the two operands
- The X and Y address space prefetch operations
- The X and Y address space prefetch destinations
- The accumulator write back destination

The other DSP instructions do not involve any multiplication and can include:

- The accumulator to be used (required)
- The source or destination operand (designated as Wso or Wdo, respectively) with or without an address modifier
- The amount of shift specified by a W register ‘Wn’ or a literal value

The control instructions can use some of the following operands:

- A program memory address
- The mode of the Table Read and Table Write instructions

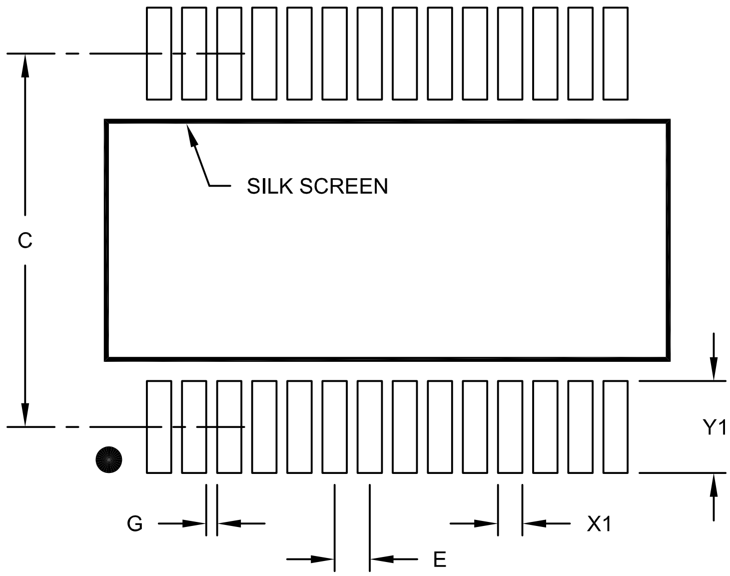
TABLE 23-2: INSTRUCTION SET OVERVIEW (CONTINUED)

Base Instr #	Assembly Mnemonic	Assembly Syntax	Description	# of Words	# of Cycles	Status Flags Affected
47	MAC	MAC Wm*Wn,Acc,Wx,Wxd,Wy,Wyd,AWB	Multiply and Accumulate	1	1	OA,OB,OAB, SA,SB,SAB
		MAC Wm*Wm,Acc,Wx,Wxd,Wy,Wyd	Square and Accumulate	1	1	OA,OB,OAB, SA,SB,SAB
48	MOV	MOV f,Wn	Move f to Wn	1	1	None
		MOV f	Move f to f	1	1	None
		MOV f,WREG	Move f to WREG	1	1	None
		MOV #lit16,Wn	Move 16-bit literal to Wn	1	1	None
		MOV.b #lit8,Wn	Move 8-bit literal to Wn	1	1	None
		MOV Wn,f	Move Wn to f	1	1	None
		MOV Wso,Wdo	Move Ws to Wd	1	1	None
		MOV WREG,f	Move WREG to f	1	1	None
		MOV.D Wns,Wd	Move Double from W(ns):W(ns + 1) to Wd	1	2	None
		MOV.D Ws,Wnd	Move Double from Ws to W(nd + 1):W(nd)	1	2	None
49	MOVPAG	MOVPAG #lit10,DSRPAG	Move 10-bit literal to DSRPAG	1	1	None
		MOVPAG #lit8,TBLPAG	Move 8-bit literal to TBLPAG	1	1	None
		MOVPAGW Ws, DSRPAG	Move Ws<9:0> to DSRPAG	1	1	None
		MOVPAGW Ws, TBLPAG	Move Ws<7:0> to TBLPAG	1	1	None
50	MOVSAC	MOVSAC Acc,Wx,Wxd,Wy,Wyd,AWB	Prefetch and store accumulator	1	1	None
51	MPY	MPY Wm*Wn,Acc,Wx,Wxd,Wy,Wyd	Multiply Wm by Wn to Accumulator	1	1	OA,OB,OAB, SA,SB,SAB
		MPY Wm*Wm,Acc,Wx,Wxd,Wy,Wyd	Square Wm to Accumulator	1	1	OA,OB,OAB, SA,SB,SAB
52	MPY.N	MPY.N Wm*Wn,Acc,Wx,Wxd,Wy,Wyd	-(Multiply Wm by Wn) to Accumulator	1	1	None
53	MSC	MSC Wm*Wm,Acc,Wx,Wxd,Wy,Wyd,AWB	Multiply and Subtract from Accumulator	1	1	OA,OB,OAB, SA,SB,SAB
54	MUL	MUL.SS Wb,Ws,Wnd	{Wnd + 1, Wnd} = signed(Wb) * signed(Ws)	1	1	None
		MUL.SS Wb,Ws,Acc	Accumulator = signed(Wb) * signed(Ws)	1	1	None
		MUL.SU Wb,Ws,Wnd	{Wnd + 1, Wnd} = signed(Wb) * unsigned(Ws)	1	1	None
		MUL.SU Wb,Ws,Acc	Accumulator = signed(Wb) * unsigned(Ws)	1	1	None
		MUL.SU Wb,#lit5,Acc	Accumulator = signed(Wb) * unsigned(lit5)	1	1	None
		MUL.US Wb,Ws,Wnd	{Wnd + 1, Wnd} = unsigned(Wb) * signed(Ws)	1	1	None
		MUL.US Wb,Ws,Acc	Accumulator = unsigned(Wb) * signed(Ws)	1	1	None
		MUL.UU Wb,Ws,Wnd	{Wnd + 1, Wnd} = unsigned(Wb) * unsigned(Ws)	1	1	None
		MUL.UU Wb,#lit5,Acc	Accumulator = unsigned(Wb) * unsigned(lit5)	1	1	None
		MUL.UU Wb,Ws,Acc	Accumulator = unsigned(Wb) * unsigned(Ws)	1	1	None
		MULW.SS Wb,Ws,Wnd	Wnd = signed(Wb) * signed(Ws)	1	1	None
		MULW.SU Wb,Ws,Wnd	Wnd = signed(Wb) * unsigned(Ws)	1	1	None
		MULW.US Wb,Ws,Wnd	Wnd = unsigned(Wb) * signed(Ws)	1	1	None
		MULW.UU Wb,Ws,Wnd	Wnd = unsigned(Wb) * unsigned(Ws)	1	1	None
		MUL.SU Wb,#lit5,Wnd	{Wnd + 1, Wnd} = signed(Wb) * unsigned(lit5)	1	1	None
		MUL.SU Wb,#lit5,Wnd	Wnd = signed(Wb) * unsigned(lit5)	1	1	None
		MUL.UU Wb,#lit5,Wnd	{Wnd + 1, Wnd} = unsigned(Wb) * unsigned(lit5)	1	1	None
		MUL.UU Wb,#lit5,Wnd	Wnd = unsigned(Wb) * unsigned(lit5)	1	1	None
		MUL f	W3:W2 = f * WREG	1	1	None

Note: Read and Read-Modify-Write (e.g., bit operations and logical operations) on non-CPU SFRs incur an additional instruction cycle.

28-Lead Plastic Shrink Small Outline (SS) - 5.30 mm Body [SSOP]

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



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E	0.65 BSC		
Contact Pad Spacing	C		7.20	
Contact Pad Width (X28)	X1			0.45
Contact Pad Length (X28)	Y1			1.75
Distance Between Pads	G	0.20		

Notes:

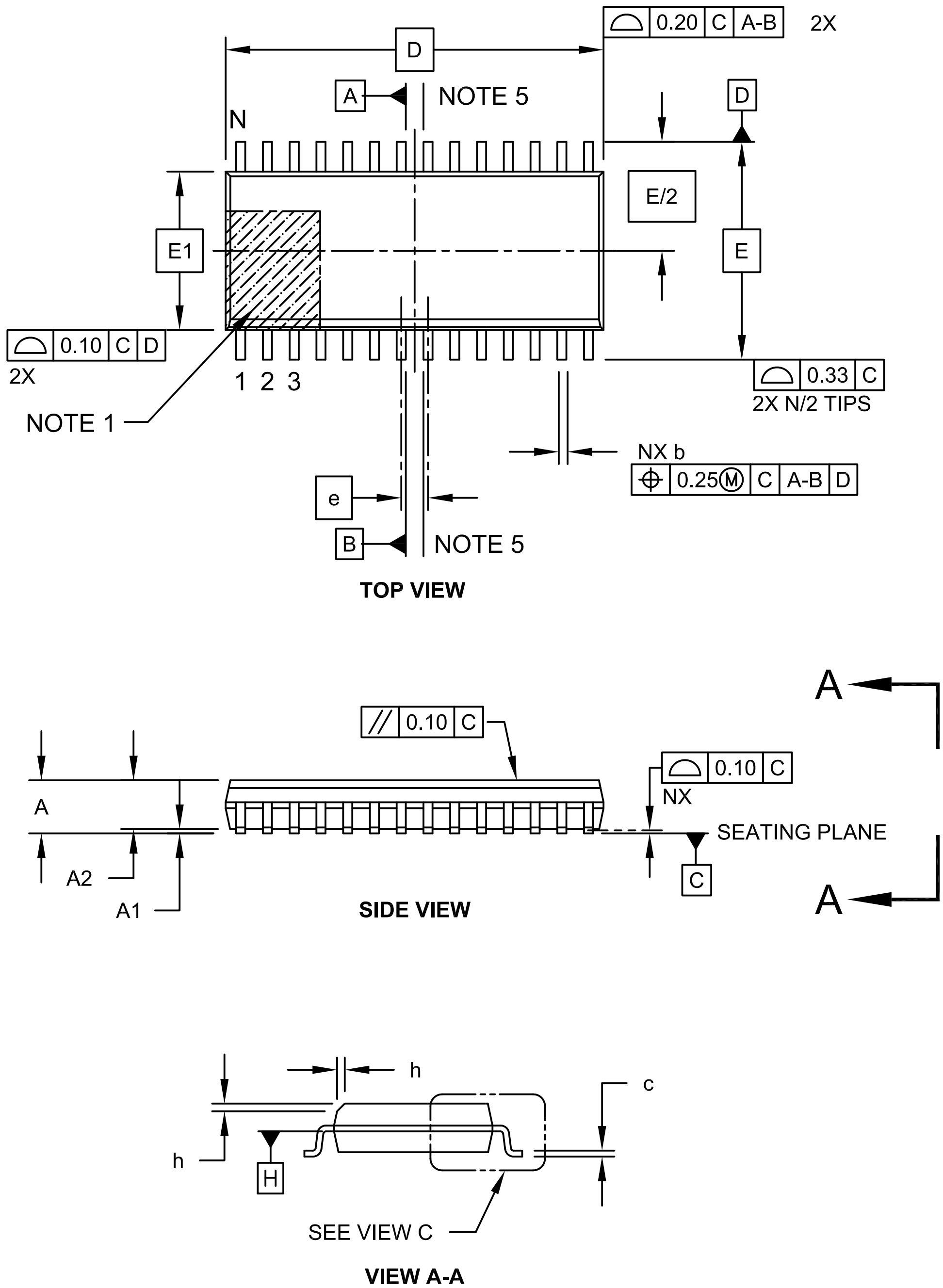
1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2073A

28-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

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



Microchip Technology Drawing C04-052C Sheet 1 of 2

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
═ ISO/TS 16949 ═

ISBN: 978-1-5224-0578-8