Microchip Technology - DSPIC33CH128MP505-E/PT Datasheet

Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	dsPIC
Core Size	16-Bit Dual-Core
Speed	180MHz, 200MHz
Connectivity	CANbus, I²C, IrDA, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, Motor Control PWM, POR, PWM, QEI, WDT
Number of I/O	39
Program Memory Size	152KB (152K x 8)
Program Memory Type	FLASH, PRAM
EEPROM Size	-
RAM Size	20K x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 3.6V
Data Converters	A/D 31x12b; D/A 4x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	48-TQFP
Supplier Device Package	48-TQFP (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/dspic33ch128mp505-e-pt

TABLE 7: 48-PIN QFN/TQFP/UQFN

Pin #	Master Core	Slave Core
1	**RP46**/PWM1H/RB14	**S1RP46**/S1PWM6L/S1RB14
2	**RP47**/PWM1L/RB15	**S1RP47**/S1PWM6H/S1RB15
3	**RP60**/RC12	**S1RP60**/S1PWM3H/S1RC12
4	**RP61**/RC13	**S1RP61**/S1PWM3L/S1RC13
5	$\overline{\text{MCLR}}$	—
6	RD13	S1ANN0/S1PGA1N2/S1RD13
7	AN12/IBIAS3/**RP48**/RC0	S1AN10/**S1RP48**/S1RC0
8	AN0/CMP1A/RA0	S1RA0
9	AN1/RA1	S1AN15/S1RA1
10	AN2/RA2	S1AN16/S1RA2
11	AN3/IBIAS0/RA3	S1AN0/S1CMP1A/S1PGA1P1/S1RA3
12	AN4/IBIAS1/RA4	$\overline{\text{S1MCLR3}}$/S1AN1/S1CMP2A/S1PGA2P1/S1PGA3P2/S1RA4
13	AVDD	AVDD
14	AVSS	AVSS
15	AN13/ISRC0/**RP49**/RC1	S1ANA1/**S1RP49**/S1RC1
16	AN14/ISRC1/**RP50**/RC2	S1ANA0/**S1RP50**/S1RC2
17	**RP54**/RC6	S1AN11/S1CMP1B/**S1RP54**/S1RC6
18	VDD	VDD
19	VSS	VSS
20	CMP1B/**RP51**/RC3	S1AN8/S1CMP3B/**S1RP51**/S1RC3
21	OSCI/CLKI/AN5/**RP32**/RB0	S1AN5/**S1RP32**/S1RB0
22	OSCO/CLKO/AN6/IBIAS2/**RP33**/RB1	S1AN4/**S1RP33**/S1RB1
23	ISRC3/RD10	S1AN13/S1CMP2B/S1RD10
24	AN15/ISRC2/**RP55**/RC7	S1AN12/**S1RP55**/S1RC7
25	DACOUT/AN7/CMP1D/**RP34**/INT0/RB2	$\overline{\text{S1MCLR2}}$/S1AN3/S1ANC0/S1ANC1/S1CMP1D/S1CMP2D/S1CMP3D/**S1RP34**/S1INT0/S1RB2
26	PGD2/AN8/**RP35**/RB3	S1PGD2/S1AN18/S1CMP3A/S1PGA3P1/**S1RP35**/S1RB3
27	PGC2/**RP36**/RB4	S1PGC2/S1AN9/**S1RP36**/S1PWM5L/S1RB4
28	**RP56**/ASDA1/SCK2/RC8	**S1RP56**/S1ASDA1/S1SCK1/S1RC8
29	**RP57**/ASCL1/SDI2/RC9	**S1RP57**/S1ASCL1/S1SDI1/S1RC9
30	SDO2/PCI19/RD8	S1SDO1/S1PCI19/S1RD8
31	VSS	VSS
32	VDD	VDD
33	PGD3/**RP37**/SDA2/RB5	S1PGD3/**S1RP37**/S1RB5
34	PGC3/**RP38**/SCL2/RB6	S1PGC3/**S1RP38**/S1RB6
35	TDO/AN9/**RP39**/RB7	$\overline{\text{S1MCLR1}}$/S1AN6/**S1RP39**/S1PWM5H/S1RB7
36	PGD1/AN10/**RP40**/SCL1/RB8	S1PGD1/S1AN7/**S1RP40**/S1SCL1/S1RB8
37	PGC1/AN11/**RP41**/SDA1/RB9	S1PGC1/**S1RP41**/S1SDA1/S1RB9
38	**RP52**/RC4	**S1RP52**/S1PWM2H/S1RC4
39	**RP53**/RC5	**S1RP53**/S1PWM2L/S1RC5
40	**RP58**/RC10	**S1RP58**/S1PWM1H/S1RC10
41	**RP59**/RC11	**S1RP59**/S1PWM1L/S1RC11
42	VSS	VSS
43	VDD	VDD
44	**RP65**/RD1	**S1RP65**/S1PWM4H/S1RD1
45	TMS/**RP42**/PWM3H/RB10	**S1RP42**/S1PWM8L/S1RB10
46	TCK/**RP43**/PWM3L/RB11	**S1RP43**/S1PWM8H/S1RB11
47	TDI/**RP44**/PWM2H/RB12	**S1RP44**/S1PWM7L/S1RB12
48	**RP45**/PWM2L/RB13	**S1RP45**/S1PWM7H/S1RB13

Legend: **RPn** and **S1RPn** represent remappable pins for Peripheral Pin Select functions.

3.1.5 PROGRAMMER'S MODEL

The programmer's model for the dsPIC33CH128MP508 family is shown in Figure 3-2. All registers in the programmer's model are memory-mapped and can be manipulated directly by instructions. Table 3-1 lists a description of each register.

In addition to the registers contained in the programmer's model, the dsPIC33CH128MP508 devices contain control registers for Modulo Addressing, Bit-Reversed Addressing and interrupts. These registers are described in subsequent sections of this document.

All registers associated with the programmer's model are memory-mapped, as shown in Figure 3-3 and Figure 3-4.

TABLE 3-1: PROGRAMMER'S MODEL REGISTER DESCRIPTIONS

Register(s) Name	Description
W0 through W15(1)	Working Register Array
W0 through W14(1)	Alternate Working Register Array 1
W0 through W14(1)	Alternate Working Register Array 2
W0 through W14(1)	Alternate Working Register Array 3
W0 through W14(1)	Alternate Working Register Array 4
ACCA, ACCB	40-Bit DSP Accumulators (Additional 4 Alternate Accumulators)
PC	23-Bit Program Counter
SR	ALU and DSP Engine STATUS Register
SPLIM	Stack Pointer Limit Value Register
TBLPAG	Table Memory Page Address Register
DSRPAG	Extended Data Space (EDS) Read Page Register
RCOUNT	REPEAT Loop Counter Register
DCOUNT	DO Loop Counter Register
DOSTARTH, DOSTARTL(2)	DO Loop Start Address Register (High and Low)
DOENDH, DOENDL	DO Loop End Address Register (High and Low)
CORCON	Contains DSP Engine, DO Loop Control and Trap Status bits

Note 1: Memory-mapped W0 through W14 represent the value of the register in the currently active CPU context.

2: The DOSTARTH and DOSTARTL registers are read-only.

3.1.8 ARITHMETIC LOGIC UNIT (ALU)

The dsPIC33CH128MP508 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"* (DS70000157) 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.

3.1.8.1 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

3.1.8.2 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 16-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. There are additional instructions: `DIV2` and `DIVF2`. Divide instructions will complete in six cycles.

3.1.9 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`, `NEG`, `MIN` and `MAX`.

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

REGISTER 3-24: TRISx: OUTPUT ENABLE FOR PORTx REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TRISx<15:8>							
bit 15							bit 8

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TRISx<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 **TRISx<15:0>:** Output Enable for PORTx bits
1 = LATx[n] is not driven on the PORTx[n] pin
0 = LATx[n] is driven on the PORTx[n] pin

REGISTER 3-25: PORTx: INPUT DATA FOR PORTx REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
PORTx<15:8>							
bit 15							bit 8

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
PORTx<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 **PORTx<15:0>:** PORTx Data Input Value bits

REGISTER 3-45: RPINR9: PERIPHERAL PIN SELECT INPUT REGISTER 9

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ICM7R7	ICM7R6	ICM7R5	ICM7R4	ICM7R3	ICM7R2	ICM7R1	ICM7R0
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
TCKI7R7	TCKI7R6	TCKI7R5	TCKI7R4	TCKI7R3	TCKI7R2	TCKI7R1	TCKI7R0
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 **ICM7R<7:0>:** Assign SCCP Capture 7 (ICM7) Input to the Corresponding RPn Pin bits
See Table 3-30.

bit 7-0 **TCKI7R<7:0>:** Assign SCCP Timer7 (TCKI7) Input to the Corresponding RPn Pin bits
See Table 3-30.

REGISTER 3-46: RPINR10: PERIPHERAL PIN SELECT INPUT REGISTER 10

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ICM8R7	ICM8R6	ICM8R5	ICM8R4	ICM8R3	ICM8R2	ICM8R1	ICM8R0
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
TCKI8R7	TCKI8R6	TCKI8R5	TCKI8R4	TCKI8R3	TCKI8R2	TCKI8R1	TCKI8R0
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 **ICM8R<7:0>:** Assign SCCP Capture 8 (ICM8) Input to the Corresponding RPn Pin bits
See Table 3-30.

bit 7-0 **TCKI8R<7:0>:** Assign SCCP Timer8 (TCKI8) Input to the Corresponding RPn Pin bits
See Table 3-30.

REGISTER 3-101: DMTHOLDREG: DMT HOLD 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
UPRCNT<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
UPRCNT<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 **UPRCNT<15:0>:** DMTCNTH Register Value when DMTCNTL and DMTCNTH were Last Read bits

Note 1: The DMTHOLDREG register is initialized to '0' on Reset, and is only loaded when the DMTCNTL and DMTCNTH registers are read.

REGISTER 3-178: ADTRIGnL AND ADTRIGnH: ADC CHANNEL TRIGGER n(x) SELECTION REGISTERS LOW AND HIGH (x = 0 TO 19; n = 0 TO 4)

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	TRGSRC(x+1)4	TRGSRC(x+1)3	TRGSRC(x+1)2	TRGSRC(x+1)1	TRGSRC(x+1)0
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	TRGSRCx4	TRGSRCx3	TRGSRCx2	TRGSRCx1	TRGSRCx0
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 15-13 **Unimplemented:** Read as '0'

bit 12-8 **TRGSRC(x+1)<4:0>:** Trigger Source Selection for Corresponding Analog Input bits (TRGSRC1 to TRGSRC19 – Odd)

11111 = ADTRG31 (PPS input)
11110 = Master PTG
11101 = Slave CLC1
11100 = Master CLC1
11011 = Slave PWM8 Trigger 2
11010 = Slave PWM5 Trigger 2
11001 = Slave PWM3 Trigger 2
11000 = Slave PWM1 Trigger 2
10111 = Master SCCP4 PWM interrupt
10110 = Master SCCP3 PWM interrupt
10101 = Master SCCP2 PWM interrupt
10100 = Master SCCP1 PWM interrupt
10011 = Reserved
10010 = Reserved
10001 = Reserved
10000 = Reserved
01111 = Reserved
01110 = Reserved
01101 = Reserved
01100 = Reserved
01011 = Master PWM4 Trigger 2
01010 = Master PWM4 Trigger 1
01001 = Master PWM3 Trigger 2
01000 = Master PWM3 Trigger 1
00111 = Master PWM2 Trigger 2
00110 = Master PWM2 Trigger 1
00101 = Master PWM1 Trigger 2
00100 = Master PWM1 Trigger 1
00011 = Reserved
00010 = Level software trigger
00001 = Common software trigger
00000 = No trigger is enabled

bit 7-5 **Unimplemented:** Read as '0'

FIGURE 4-2: PROGRAMMER'S MODEL (SLAVE)

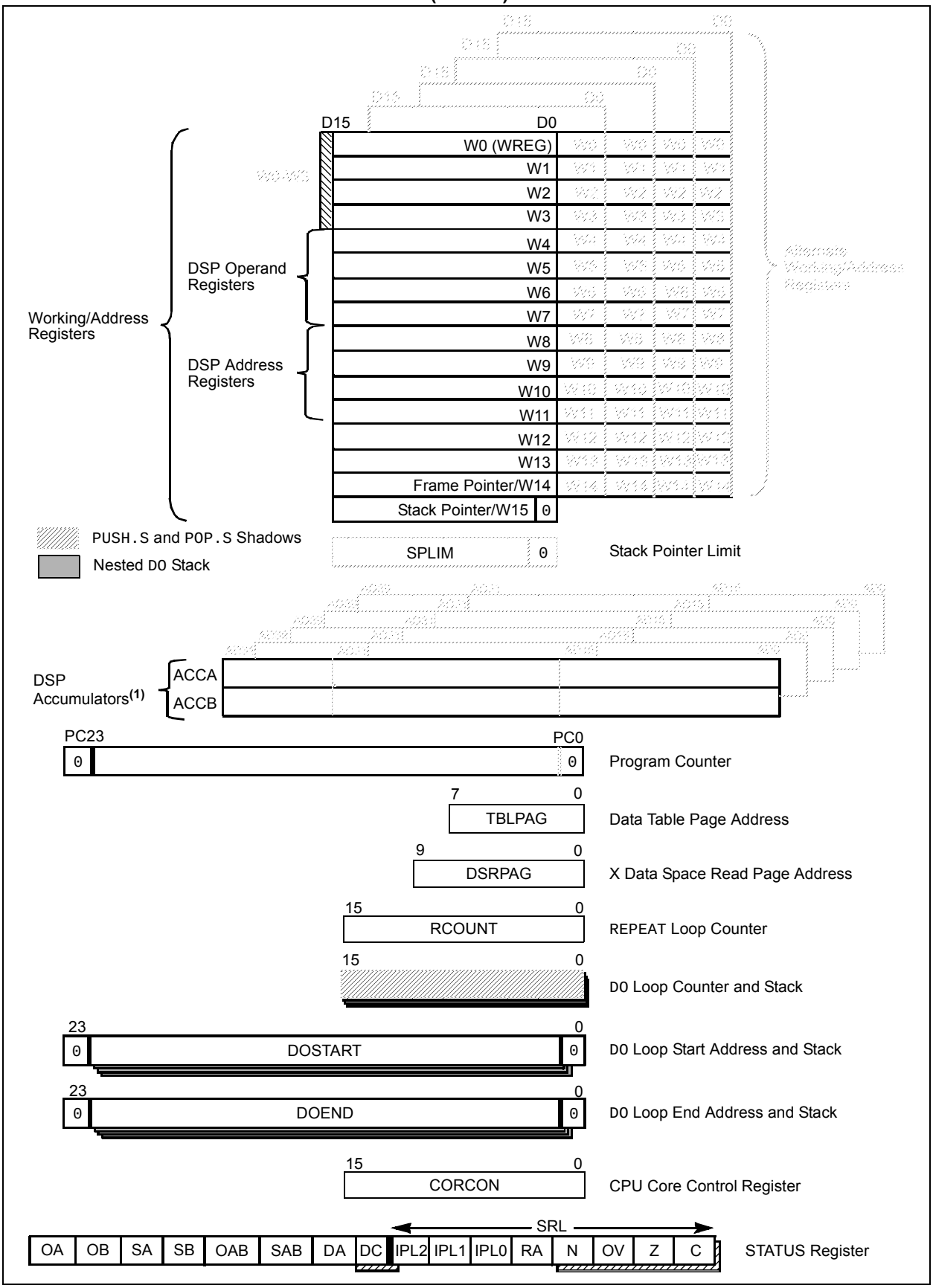
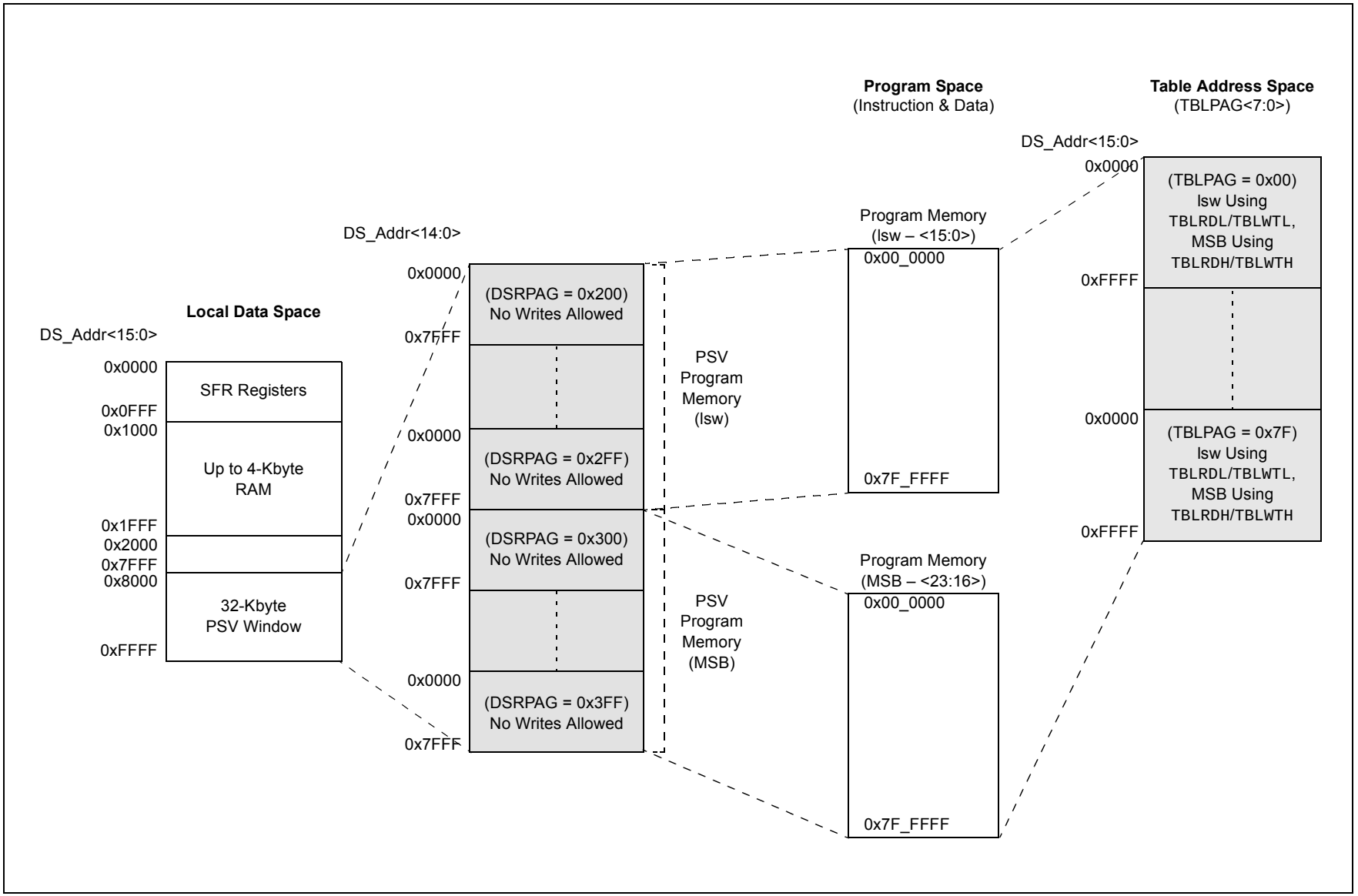


FIGURE 4-7: PAGED DATA MEMORY SPACE



REGISTER 4-49: RPINR20: PERIPHERAL PIN SELECT INPUT REGISTER 20

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SCK1R7	SCK1R6	SCK1R5	SCK1R4	SCK1R3	SCK1R2	SCK1R1	SCK1R0
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
SDI1R7	SDI1R6	SDI1R5	SDI1R4	SDI1R3	SDI1R2	SDI1R1	SDI1R0
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 **SCK1R<7:0>:** Assign SPI1 Clock Input (S1SCK1) to the Corresponding S1RPn Pin bits
See Table 4-27.

bit 7-0 **SDI1R<7:0>:** Assign SPI1 Data Input (S1SDI1) to the Corresponding S1RPn Pin bits
See Table 4-27.

REGISTER 4-50: RPINR21: PERIPHERAL PIN SELECT INPUT REGISTER 21

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
REFOIR7	REFOIR6	REFOIR5	REFOIR4	REFOIR3	REFOIR2	REFOIR1	REFOIR0
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
SS1R7	SS1R6	SS1R5	SS1R4	SS1R3	SS1R2	SS1R1	SS1R0
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 **REFOIR<7:0>:** Assign Reference Clock Input (S1REFOI) to the Corresponding S1RPn Pin bits
See Table 4-27.

bit 7-0 **SS1R<7:0>:** Assign SPI1 Slave Select ($\overline{\text{S1SS1}}$) to the Corresponding S1RPn Pin bits
See Table 4-27.

REGISTER 4-51: RPINR23: PERIPHERAL PIN SELECT INPUT REGISTER 23

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

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-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 15-8 **U1CTSR<7:0>:** Assign UART1 Clear-to-Send ($\overline{\text{S1U1CTS}}$) to the Corresponding S1RPn Pin bits
See Table 4-27.

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

REGISTER 4-52: RPINR37: PERIPHERAL PIN SELECT INPUT REGISTER 37

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PCI17R7	PCI17R6	PCI17R5	PCI17R4	PCI17R3	PCI17R2	PCI17R1	PCI17R0
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:			
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 **PCI17R<7:0>:** Assign PWM Input 17 (S1PCI17) to the Corresponding S1RPn Pin bits
See Table 4-27.

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

REGISTER 4-66: RPOR6: PERIPHERAL PIN SELECT OUTPUT REGISTER 6

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	RP45R5	RP45R4	RP45R3	RP45R2	RP45R1	RP45R0
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
—	—	RP44R5	RP44R4	RP44R3	RP44R2	RP44R1	RP44R0
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 **RP45R<5:0>:** Peripheral Output Function is Assigned to S1RP45 Output Pin bits (see Table 4-31 for peripheral function numbers)

bit 7-6 **Unimplemented:** Read as '0'

bit 5-0 **RP44R<5:0>:** Peripheral Output Function is Assigned to S1RP44 Output Pin bits (see Table 4-31 for peripheral function numbers)

REGISTER 4-67: RPOR7: PERIPHERAL PIN SELECT OUTPUT REGISTER 7

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	RP47R5	RP47R4	RP47R3	RP47R2	RP47R1	RP47R0
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
—	—	RP46R5	RP46R4	RP46R3	RP46R2	RP46R1	RP46R0
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 **RP47R<5:0>:** Peripheral Output Function is Assigned to S1RP47 Output Pin bits (see Table 4-31 for peripheral function numbers)

bit 7-6 **Unimplemented:** Read as '0'

bit 5-0 **RP46R<5:0>:** Peripheral Output Function is Assigned to S1RP46 Output Pin bits (see Table 4-31 for peripheral function numbers)

REGISTER 7-12: PMD4: SLAVE PERIPHERAL MODULE DISABLE 4 CONTROL REGISTER

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

U-0	U-0	U-0	U-0	R/W-0	U-0	U-0	U-0
—	—	—	—	REFOMD	—	—	—
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 15-4 **Unimplemented:** Read as ‘0’

bit 3 **REFOMD:** Reference Clock Module Disable bit
1 = Reference clock module is disabled
0 = Reference clock module is enabled

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

REGISTER 7-15: PMD8: SLAVE PERIPHERAL MODULE DISABLE 8 CONTROL REGISTER

U-0	R/W-0	U-0	U-0	U-0	R/W-0	U-0	U-0
—	PGA3MD	—	—	—	PGA2MD	—	—
bit 15							bit 8

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0
—	—	CLC4MD	CLC3MD	CLC2MD	CLC1MD	—	—
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 **Unimplemented:** Read as '0'

bit 14 **PGA3MD:** PGA3 Module Disable bit
1 = PGA3 module is disabled
0 = PGA3 module is enabled

bit 13-11 **Unimplemented:** Read as '0'

bit 10 **PGA2MD:** PGA2 Module Disable bit
1 = PGA2 module is disabled
0 = PGA2 module is enabled

bit 9-6 **Unimplemented:** Read as '0'

bit 5 **CLC4MD:** CLC4 Module Disable bit
1 = CLC4 module is disabled
0 = CLC4 module is enabled

bit 4 **CLC3MD:** CLC3 Module Disable bit
1 = CLC3 module is disabled
0 = CLC3 module is enabled

bit 3 **CLC2MD:** CLC2 Module Disable bit
1 = CLC2 module is disabled
0 = CLC2 module is enabled

bit 2 **CLC1MD:** CLC1 Module Disable bit
1 = CLC1 module is disabled
0 = CLC1 module is enabled

bit 1-0 **Unimplemented:** Read as '0'

REGISTER 9-32: PGxCAP: PWM GENERATOR x CAPTURE REGISTER

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
PGxCAP<15:8>							
bit 15							bit 8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
PGxCAP<7:0>(1)							
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 **PGxCAP<15:0>:** PGx Time Base Capture bits(1)

Note 1: PGxCAP<1:0> will read as '0' in Standard Resolution mode. PGxCAP<4:0> will read as '0' in High-Resolution mode.

REGISTER 13-13: UxTXCHK: UARTx TRANSMIT CHECKSUM REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
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
TXCHK<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 15-8 **Unimplemented:** Read as '0'

bit 7-0 **TXCHK<7:0>:** Transmit Checksum bits (calculated from TX words)

LIN Modes:
C0EN = 1: Sum of all transmitted data + addition carries, including PID.
C0EN = 0: Sum of all transmitted data + addition carries, excluding PID.

LIN Slave:
Cleared when Break is detected.

LIN Master/Slave:
Cleared when Break is detected.

Other Modes:
C0EN = 1: Sum of every byte transmitted + addition carries.
C0EN = 0: Value remains unchanged.

REGISTER 13-17: UxINT: UARTx INTERRUPT REGISTER

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

HS/R/W-0	HS/R/W-0	U-0	U-0	U-0	R/W-0	U-0	U-0
WUIF	ABDIF	—	—	—	ABDIE	—	—
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 15-8 **Unimplemented:** Read as '0'

bit 7 **WUIF:** Wake-up Interrupt Flag bit
- 1 = Sets when WAKE = 1 and RX makes a '1'-to-'0' transition; triggers event interrupt (must be cleared by software)
- 0 = WAKE is not enabled or WAKE is enabled, but no wake-up event has occurred

bit 6 **ABDIF:** Auto-Baud Completed Interrupt Flag bit
- 1 = Sets when ABD sequence makes the final '1'-to-'0' transition; triggers event interrupt (must be cleared by software)
- 0 = ABAUD is not enabled or ABAUD is enabled but auto-baud has not completed

bit 5-3 **Unimplemented:** Read as '0'

bit 2 **ABDIE:** Auto-Baud Completed Interrupt Enable Flag bit
- 1 = Allows ABDIF to set an event interrupt
- 0 = ABDIF does not set an event interrupt

bit 1-0 **Unimplemented:** Read as '0'

The SPI module has the ability to generate three interrupts reflecting the events that occur during the data communication. The following types of interrupts can be generated:

1. Receive interrupts are signalled by SPIxRXIF. This event occurs when:
   - RX watermark interrupt
   - SPIROV = 1
   - SPIRBF = 1
   - SPIRBE = 1

   provided the respective mask bits are enabled in SPIxIMSKL/H.
2. Transmit interrupts are signalled by SPIxTXIF. This event occurs when:
   - TX watermark interrupt
   - SPITUR = 1
   - SPITBF = 1
   - SPITBE = 1

   provided the respective mask bits are enabled in SPIxIMSKL/H.
3. General interrupts are signalled by SPIxGIF. This event occurs when:
   - FRMERR = 1
   - SPIBUSY = 1
   - SRMT = 1

   provided the respective mask bits are enabled in SPIxIMSKL/H.

Block diagrams of the module in Standard and Enhanced modes are shown in Figure 14-1 and Figure 14-2.

Note:	In this section, the SPI modules are referred to together as SPIx, or separately as SPI1, SPI2 or SPI3. Special Function Registers will follow a similar notation. For example, SPIxCON1 and SPIxCON2 refer to the control registers for any of the three SPI modules.

To set up the SPIx module for the Standard Master mode of operation:

1. If using interrupts:
   a) Clear the interrupt flag bits in the respective IFSx register.
   b) Set the interrupt enable bits in the respective IECx register.
   c) Write the SPIxIP bits in the respective IPCx register to set the interrupt priority.
2. Write the desired settings to the SPIxCON1L and SPIxCON1H registers with the MSTEN bit (SPIxCON1L<5>) = 1.
3. Clear the SPIROV bit (SPIxSTATL<6>).
4. Enable SPIx operation by setting the SPIEN bit (SPIxCON1L<15>).
5. Write the data to be transmitted to the SPIxBUFL and SPIxBUFH registers. Transmission (and reception) will start as soon as data is written to the SPIxBUFL and SPIxBUFH registers.

To set up the SPIx module for the Standard Slave mode of operation:

1. Clear the SPIxBUF registers.
2. If using interrupts:
   a) Clear the SPIxBUFL and SPIxBUFH registers.
   b) Set the interrupt enable bits in the respective IECx register.
   c) Write the SPIxIP bits in the respective IPCx register to set the interrupt priority.
3. Write the desired settings to the SPIxCON1L, SPIxCON1H and SPIxCON2L registers with the MSTEN bit (SPIxCON1L<5>) = 0.
4. Clear the SMP bit.
5. If the CKE bit (SPIxCON1L<8>) is set, then the SSEN bit (SPIxCON1L<7>) must be set to enable the $\overline{SSx}$ pin.
6. Clear the SPIROV bit (SPIxSTATL<6>).
7. Enable SPIx operation by setting the SPIEN bit (SPIxCON1L<15>).

FIGURE 16-1: SENTx MODULE BLOCK DIAGRAM

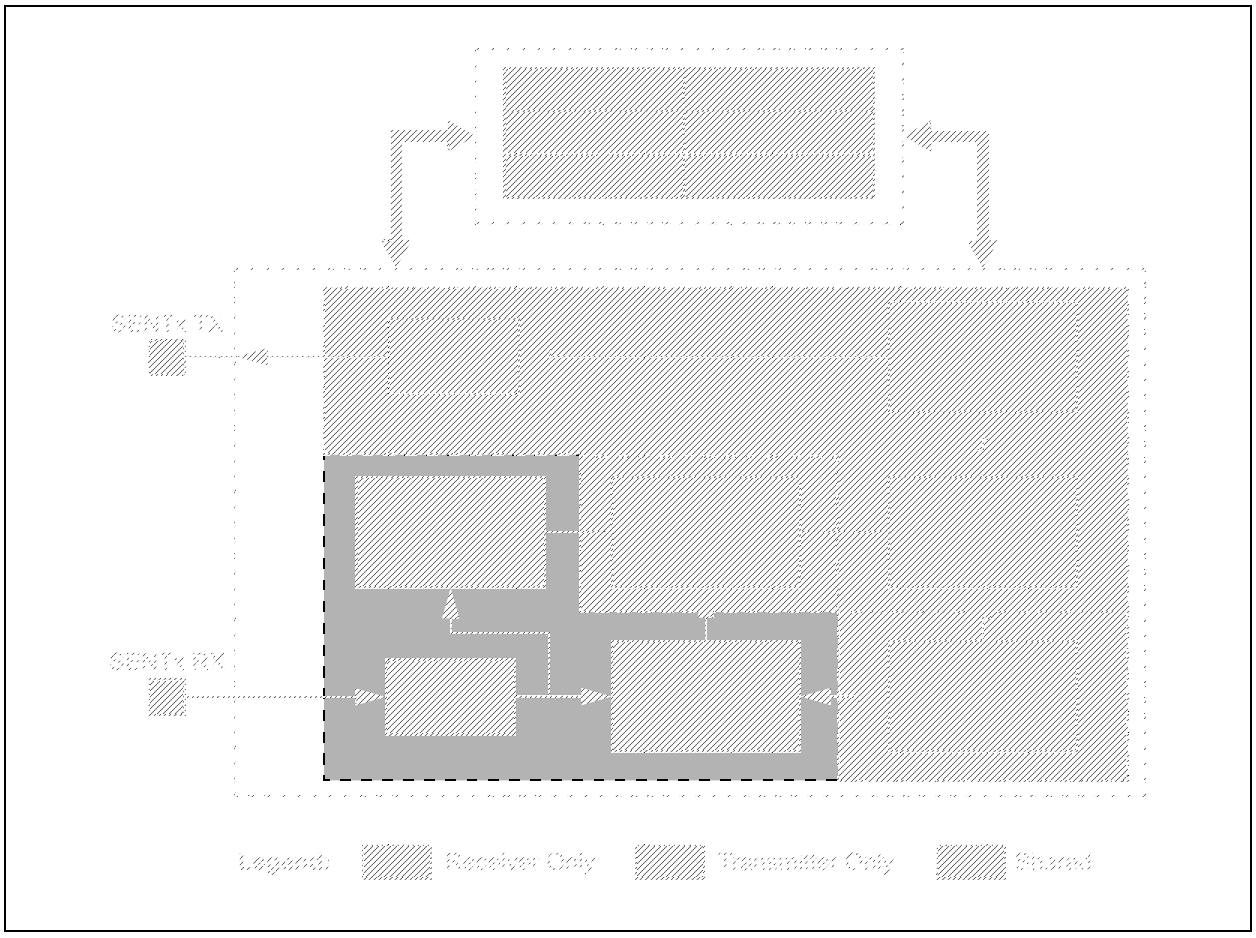
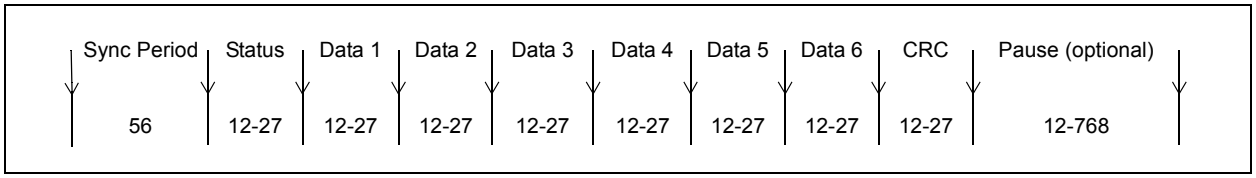


FIGURE 16-2: SENTx PROTOCOL DATA FRAMES



16.2 Transmit Mode

By default, the SENTx module is configured for transmit operation. The module can be configured for continuous asynchronous message frame transmission, or alternatively, for Synchronous mode triggered by software. When enabled, the transmitter will send a Sync, followed by the appropriate number of data nibbles, an optional CRC and optional pause pulse. The tick period used by the SENTx transmitter is set by writing a value to the TICKTIME<15:0> (SENTxCON2<15:0>) bits. The tick period calculations are shown in Equation 16-1.

EQUATION 16-1: TICK PERIOD CALCULATION

$$TICKTIME\langle 15:0\rangle = \frac{T_{TICK}}{T_{CLK}} - 1$$

An optional pause pulse can be used in Asynchronous mode to provide a fixed message frame time period. The frame period used by the SENTx transmitter is set by writing a value to the FRAMETIME<15:0> (SENTxCON3<15:0>) bits. The formulas used to calculate the value of frame time are shown in Equation 16-2.

EQUATION 16-2: FRAME TIME CALCULATIONS

$$FRAMETIME\langle 15:0\rangle = T_{TICK}/T_{FRAME}$$

$$FRAMETIME\langle 15:0\rangle \geq 122 + 27N$$

$$FRAMETIME\langle 15:0\rangle \geq 848 + 12N$$

Where:

T_{FRAME} = Total time of the message from ms

N = The number of data nibbles in message, 1-6

Note: The module will not produce a pause period with less than 12 ticks, regardless of the FRAMETIME<15:0> value. FRAMETIME<15:0> values beyond 2047 will have no effect on the length of a data frame.

16.2.1 TRANSMIT MODE CONFIGURATION

16.2.1.1 Initializing the SENTx Module

Perform the following steps to initialize the module:

1. Write RCVEN (SENTxCON1<11>) = 0 for Transmit mode.
2. Write TXM (SENTxCON1<10>) = 0 for Asynchronous Transmit mode or TXM = 1 for Synchronous mode.
3. Write NIBCNT<2:0> (SENTxCON1<2:0>) for the desired data frame length.
4. Write CRCEN (SENTxCON1<8>) for hardware or software CRC calculation.
5. Write PPP (SENTxCON1<7>) for optional pause pulse.
6. If PPP = 1, write TFRAME to SENTxCON3.
7. Write SENTxCON2 with the appropriate value for the desired tick period.
8. Enable interrupts and set interrupt priority.
9. Write initial status and data values to SENTxDATH/L.
10. If CRCEN = 0, calculate CRC and write the value to CRC<3:0> (SENTxDATL<3:0>).
11. Set the SNTEN (SENTxCON1<15>) bit to enable the module.

User software updates to SENTxDATH/L must be performed after the completion of the CRC and before the next message frame's status nibble. The recommended method is to use the message frame completion interrupt to trigger data writes.