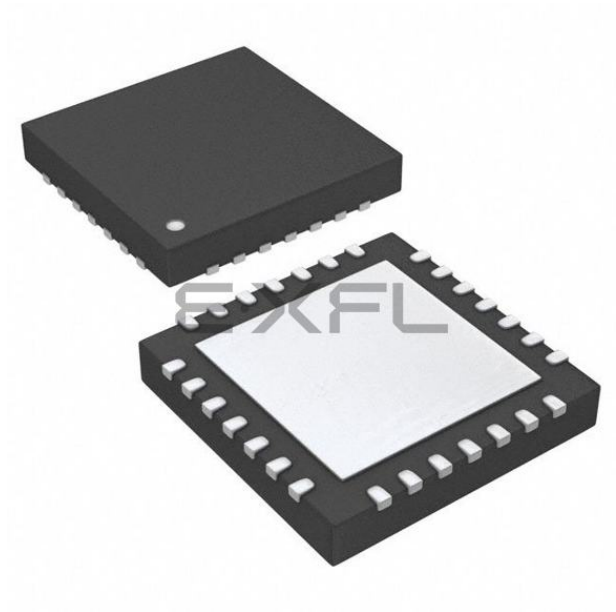


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	64MHz
Connectivity	I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, HLVD, POR, PWM, WDT
Number of I/O	25
Program Memory Size	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	2.3V ~ 5.5V
Data Converters	A/D 24x12b; D/A 1x5b
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 (6x6)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f25k42t-i-ml

3.2.3 ISR PRIORITY > PERIPHERAL PRIORITY > MAIN PRIORITY

In this case, interrupt routines and peripheral operation (DMAx, Scanner) will stall the CPU. Interrupt will preempt peripheral operation. This results in lowest interrupt latency and highest throughput for the peripheral to access the memory.

3.2.4 PERIPHERAL 1 PRIORITY > ISR PRIORITY > MAIN PRIORITY > PERIPHERAL 2 PRIORITY

In this case, the Peripheral 1 will stall the execution of the CPU. However, Peripheral 2 can access the memory in cycles unused by Peripheral 1.

The operation of the System Arbiter is controlled through the following registers:

REGISTER 3-1: ISRPR: INTERRUPT SERVICE ROUTINE PRIORITY REGISTER

U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	—	ISRPR2	ISRPR1	ISRPR0
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
1 = bit is set	0 = bit is cleared	HS = Hardware set

bit 7-3 **Unimplemented:** Read as '0'

bit 2-0 **ISRPR<2:0>**: Interrupt Service Routine Priority Selection bits

REGISTER 3-2: MAINPR: MAIN ROUTINE PRIORITY REGISTER

U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-1/1
—	—	—	—	—	MAINPR2	MAINPR1	MAINPR0
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
1 = bit is set	0 = bit is cleared	HS = Hardware set

bit 7-3 **Unimplemented:** Read as '0'

bit 2-0 **MAINPR<2:0>**: Main Routine Priority Selection bits

REGISTER 3-3: DMA1PR: DMA1 PRIORITY REGISTER

U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-1/1	R/W-0/0
—	—	—	—	—	DMA1PR2	DMA1PR1	DMA1PR0
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
1 = bit is set	0 = bit is cleared	HS = Hardware set

bit 7-3 **Unimplemented:** Read as '0'

bit 2-0 **DMA1PR<2:0>**: DMA1 Priority Selection bits

TABLE 4-2: PROGRAM FLASH MEMORY PARTITION

Region	Address	Partition[3]			
		$\overline{BBEN}$ = 1 SAFEN = 1	$\overline{BBEN}$ = 1 SAFEN = 0	$\overline{BBEN}$ = 0 SAFEN = 1	$\overline{BBEN}$ = 0 SAFEN = 0
Program Flash Memory	00 0000h • • • Last Boot Block Memory Address	APPLICATION BLOCK	APPLICATION BLOCK	BOOT BLOCK	BOOT BLOCK
	Last Boot Block Memory Address[1] + 1 • • • Last Program Memory Address[2] – 100h			APPLICATION BLOCK	APPLICATION BLOCK
	Last Program Memory Address[2] – FEh[4] • • • Last Program Memory Address[2]		STORAGE AREA FLASH		STORAGE AREA FLASH

Note 1: Last Boot Block Memory Address is based on BBSIZE<2:0>, see Table 5-1.

2: For Last Program Memory Address, see Table 5-1.

3: Refer to Register 5-7: Configuration Word 4L for $\overline{BBEN}$ and $\overline{SAFEN}$ definitions.

4: Storage Area Flash is implemented as the last 128 Words of User Flash, if present.

EXAMPLE 4-4: TWO-WORD INSTRUCTIONS

CASE 1:				
Object Code	**Source Code**			
0110 0110 0000 0000	TSTFSZ	REG1	;	is RAM location 0?
1100 0001 0010 0011	MOVFF	REG1, REG2	;	Yes, skip this word
1111 0100 0101 0110			;	Execute this word as a NOP
0010 0100 0000 0000	ADDWF	REG3	;	continue code
CASE 2:				
Object Code	**Source Code**			
0110 0110 0000 0000	TSTFSZ	REG1	;	is RAM location 0?
1100 0001 0010 0011	MOVFF	REG1, REG2	;	No, execute this word
1111 0100 0101 0110			;	2nd word of instruction
0010 0100 0000 0000	ADDWF	REG3	;	continue code

EXAMPLE 4-5: THREE-WORD INSTRUCTIONS

CASE 1:				
Object Code	**Source Code**			
0110 0110 0000 0000	TSTFSZ	REG1	;	is RAM location 0?
0000 0000 0110 0000	MOVFFL	REG1, REG2	;	Yes, skip this word
1111 0100 1000 1100			;	Execute this word as a NOP
1111 0100 0101 0110			;	Execute this word as a NOP
0010 0100 0000 0000	ADDWF	REG3	;	continue code
CASE 2:				
Object Code	**Source Code**			
0110 0110 0000 0000	TSTFSZ	REG1	;	is RAM location 0?
0000 0000 0110 0000	MOVFFL	REG1, REG2	;	No, execute this word
1111 0100 1000 1100			;	2nd word of instruction
1111 0100 0101 0110			;	3rd word of instruction
0010 0100 0000 0000	ADDWF	REG3	;	continue code

11.5 Context Saving

The Interrupt controller supports a two-level deep context saving (Main routine context and Low ISR context). Refer to state machine shown in Figure 11-6 for details.

The Program Counter (PC) is saved on the dedicated device PC stack. CPU registers saved include STATUS, WREG, BSR, FSR0/1/2, PRODL/H and PCLATH/U.

After WREG has been saved to the context registers, the resolved vector number of the interrupt source to be serviced is copied into WREG. Context save and restore operation is completed by the interrupt controller based on current state of the interrupts and the order in which they were sent to the CPU.

Context save/restore works the same way in both states of MVECEN. When IPEN = 0, there is only one level interrupt active. Hence, only the main context is saved when an interrupt is received.

11.5.1 ACCESSING SHADOW REGISTERS

The Interrupt controller automatically saves the context information in the shadow registers available in Bank 56. Both the saved context values (i.e., main routine and low ISR) can be accessed using the same set of shadow registers. By clearing the SHADLO bit in the SHADCON register (Register 11-43), the CPU register values saved for main routine context can accessed, and by setting the SHADLO bit of the CPU register, values saved for low ISR context can accessed. Low ISR context is automatically restored to the CPU registers upon exiting the high ISR. Similarly, the main context is automatically restored to the CPU registers upon exiting the low ISR.

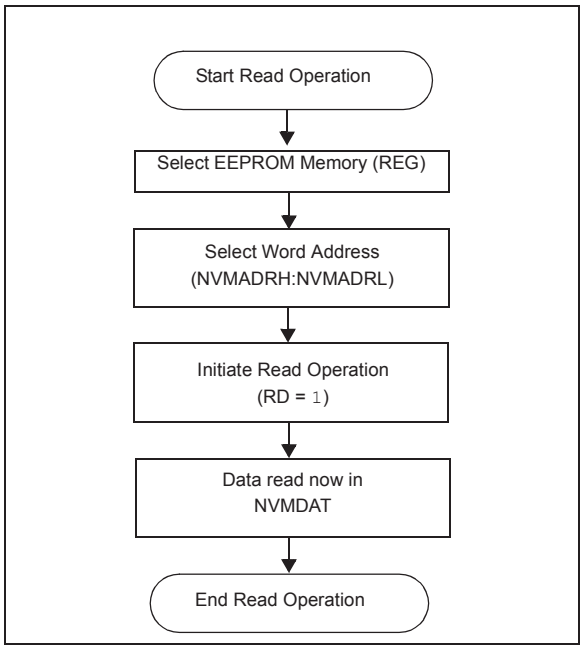
The Shadow registers in Bank 56 are readable and writable, so if the user desires to modify the context then the corresponding shadow register should be modified and the value will be restored when exiting the ISR. Depending on the user's application, other registers may also need to be saved.

15.3.3 READING THE DATA EEPROM MEMORY

To read a data memory location, the user must write the address to the NVMADRL and NVMADRH register pair, clear REG<1:0> control bit in NVMCON1 register to access Data EEPROM locations and then set control bit, RD. The data is available on the very next instruction cycle; therefore, the NVMDAT register can be read by the next instruction. NVMDAT will hold this value until another read operation, or until it is written to by the user (during a write operation).

The basic process is shown in Example 15-5.

FIGURE 15-11: DATA EEPROM READ FLOWCHART



15.3.4 WRITING TO THE DATA EEPROM MEMORY

To write an EEPROM data location, the address must first be written to the NVMADRL and NVMADRH register pair and the data written to the NVMDAT register. The sequence in Example 15-6 must be followed to initiate the write cycle.

The write will not begin if NVM Unlock sequence, described in **Section 15.1.4 "NVM Unlock Sequence"**, is not exactly followed for each byte. It is strongly recommended that interrupts be disabled during this code segment.

Additionally, the WREN bit in NVMCON1 must be set to enable writes. This mechanism prevents accidental writes to data EEPROM due to unexpected code execution (i.e., runaway programs). The WREN bit should be kept clear at all times, except when updating the EEPROM. The WREN bit is not cleared by hardware.

After a write sequence has been initiated, NVMCON1, NVMADRL, NVMADRH and NVMDAT cannot be modified. The WR bit will be inhibited from being set unless the WREN bit is set. Both WR and WREN cannot be set with the same instruction.

After a write sequence has been initiated, clearing the WREN bit will not affect this write cycle. A single Data EEPROM word is written and the operation includes an implicit erase cycle for that word (it is not necessary to set FREE). CPU execution continues in parallel and at the completion of the write cycle, the WR bit is cleared in hardware and the NVM Interrupt Flag bit (NVMIF) is set. The user can either enable this interrupt or poll this bit. NVMIF must be cleared by software.

FIGURE 27-4: GATED TIMER MODE REPEAT ACQUISITION TIMING DIAGRAM

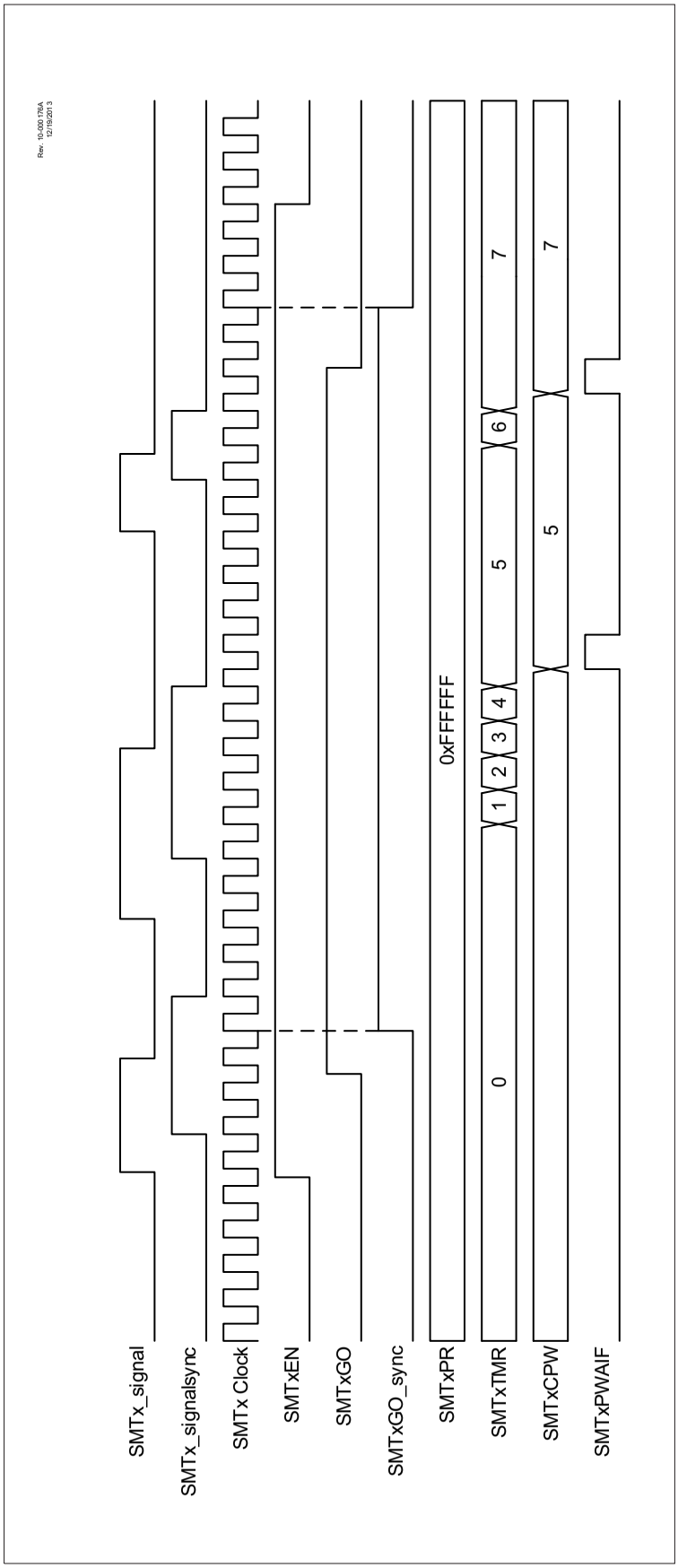


FIGURE 27-13: GATED WINDOWED MEASURE MODE SINGLE ACQUISITION TIMING DIAGRAMS

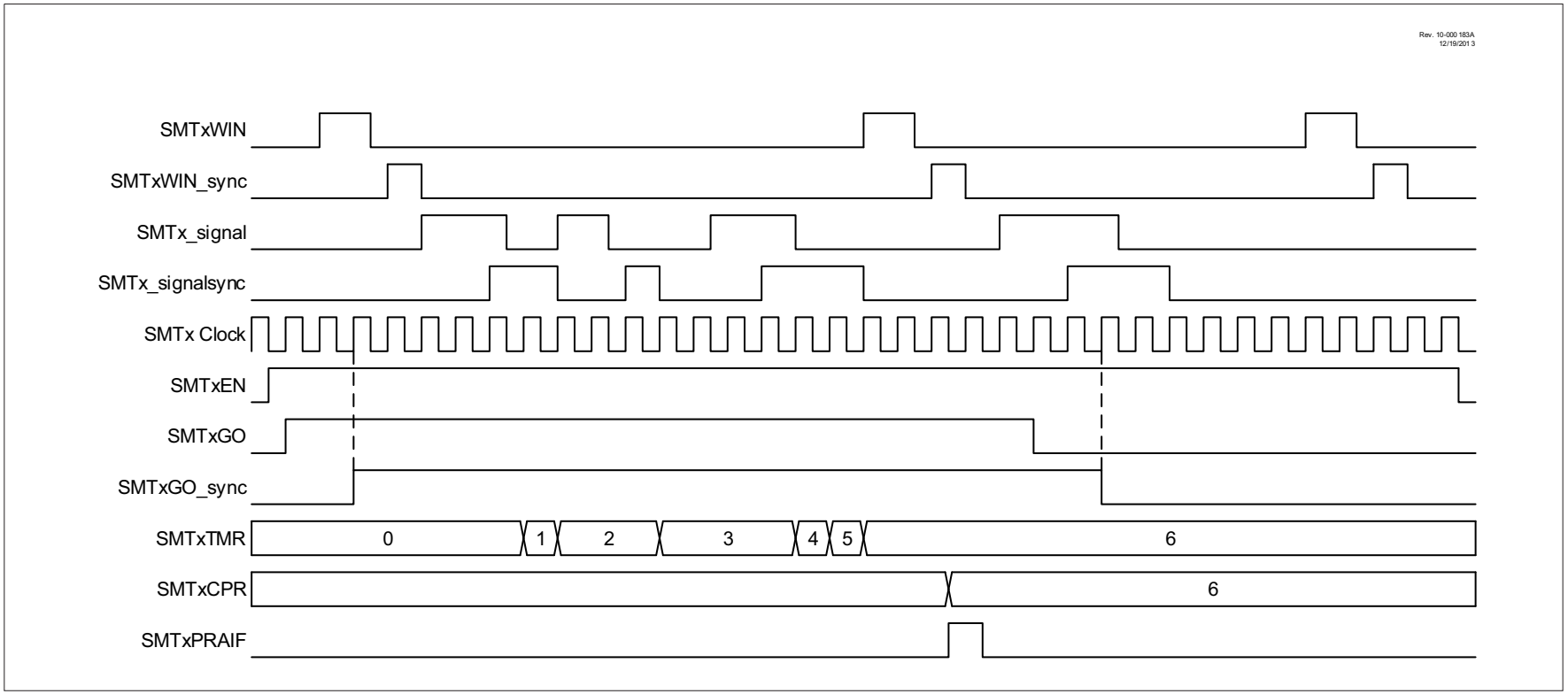
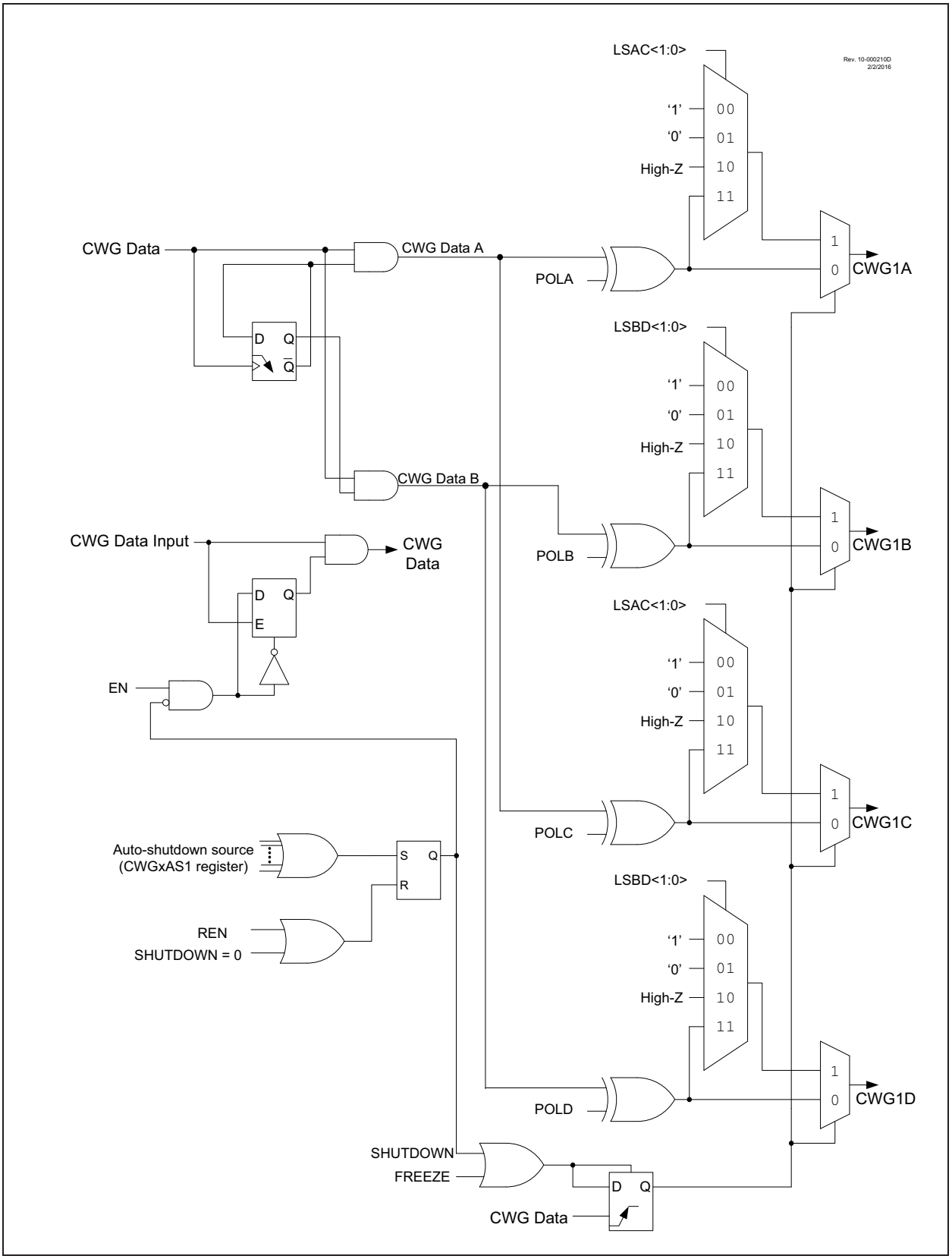


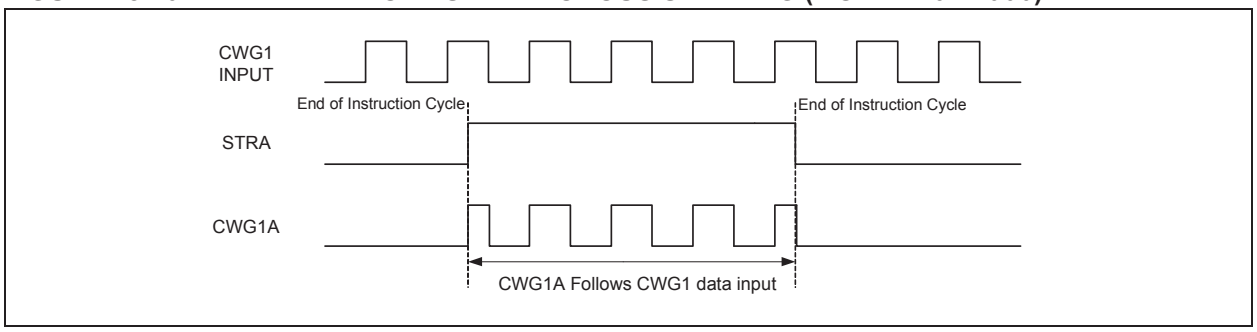
FIGURE 28-3: SIMPLIFIED CWG BLOCK DIAGRAM (PUSH-PULL MODE, MODE<2:0> = 101)



28.2.4.2 Asynchronous Steering Mode

In Asynchronous mode (MODE<2:0> bits = 000, Register 28-1), steering takes effect at the end of the instruction cycle that writes to STR. In Asynchronous Steering mode, the output signal may be an incomplete waveform (Figure 28-10). This operation may be useful when the user firmware needs to immediately remove a signal from the output pin.

FIGURE 28-10: EXAMPLE OF ASYNCHRONOUS STEERING (MODE<2:0>= 000)



28.2.4.3 Start-up Considerations

The application hardware must use the proper external pull-up and/or pull-down resistors on the CWG output pins. This is required because all I/O pins are forced to high-impedance at Reset.

The POLy bits (Register 28-2) allow the user to choose whether the output signals are active-high or active-low.

REGISTER 28-2: CWGxCON1: CWG CONTROL REGISTER 1

U-0	U-0	R-x	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	IN	—	POLD	POLC	POLB	POLA
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	q = Value depends on condition

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

bit 5 **IN:** CWG Input Value bit (read-only)

bit 4 **Unimplemented:** Read as '0'

bit 3 **POLD:** CWGxD Output Polarity bit
1 = Signal output is inverted polarity
0 = Signal output is normal polarity

bit 2 **POLC:** CWGxC Output Polarity bit
1 = Signal output is inverted polarity
0 = Signal output is normal polarity

bit 1 **POLB:** CWGxB Output Polarity bit
1 = Signal output is inverted polarity
0 = Signal output is normal polarity

bit 0 **POLA:** CWGxA Output Polarity bit
1 = Signal output is inverted polarity
0 = Signal output is normal polarity

REGISTER 33-8: UxBRGL: UART BAUD RATE GENERATOR LOW REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
BRG<7:0>							
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

bit 7-0 **BRG<7:0>:** Least Significant Byte of Baud Rate Generator

REGISTER 33-9: UxBRGH: UART BAUD RATE GENERATOR HIGH REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
BRG<15:8>							
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

bit 7-0 **BRG<15:8>:** Most Significant Byte of Baud Rate Generator

Note 1: The UxBRG registers should only be written when ON = 0.

2: Maximum BRG value when MODE = '100x' and BRGS = 1 is 0x7FFE.

3: Maximum BRG value when MODE = '100x' and BRGS = 0 is 0x1FFE.

34.6 Slave Mode

34.6.1 SLAVE MODE TRANSMIT OPTIONS

The SDO output of the SPI module in Slave mode is controlled by the TXR bit of SPIxCON2, the TRIS bit associated with the SDO pin, the Slave Select input, and the current state of the TXFIFO. This control is summarized in Table 34-2. In this table, TRISxn refers to the bit in the TRIS register corresponding to the pin that SDO has been assigned with PPS, TXR is the Transmit Data Required Control bit of SPIxCON2, SS is the state of the Slave Select input, and TXBE is the TXFIFO Buffer Empty bit of SPIxSTATUS.

34.6.1.1 SDO Drive/Tri-state

The TRIS bit associated with the SDO pin controls whether the SDO pin will tri-state. When this TRIS bit is cleared, the pin will always be driving to a level, even when the SPI module is inactive. When the SPI module is inactive (either due to the master not clocking the SCK line or the SS being false), the SDO pin will be driven to the value of the LAT bit associated with the SDO pin. When the SPI module is active, its output is determined by both TXR and whether there is data in the TXFIFO.

When the TRIS bit associated with the SDO pin is set, the pin will only have an output level driven to it when TXR = 1 and the slave select input is true. In all other cases, the pin will be tri-stated.

34.6.1.2 SDO Output Data

The TXR bit controls the nature of the data that is transmitted in Slave mode. When TXR is set, transmitted data is taken from the TXFIFO. If the FIFO is empty, the most recently received data will be transmitted and the TXUIF flag will be set to indicate that a transmit FIFO underflow has occurred.

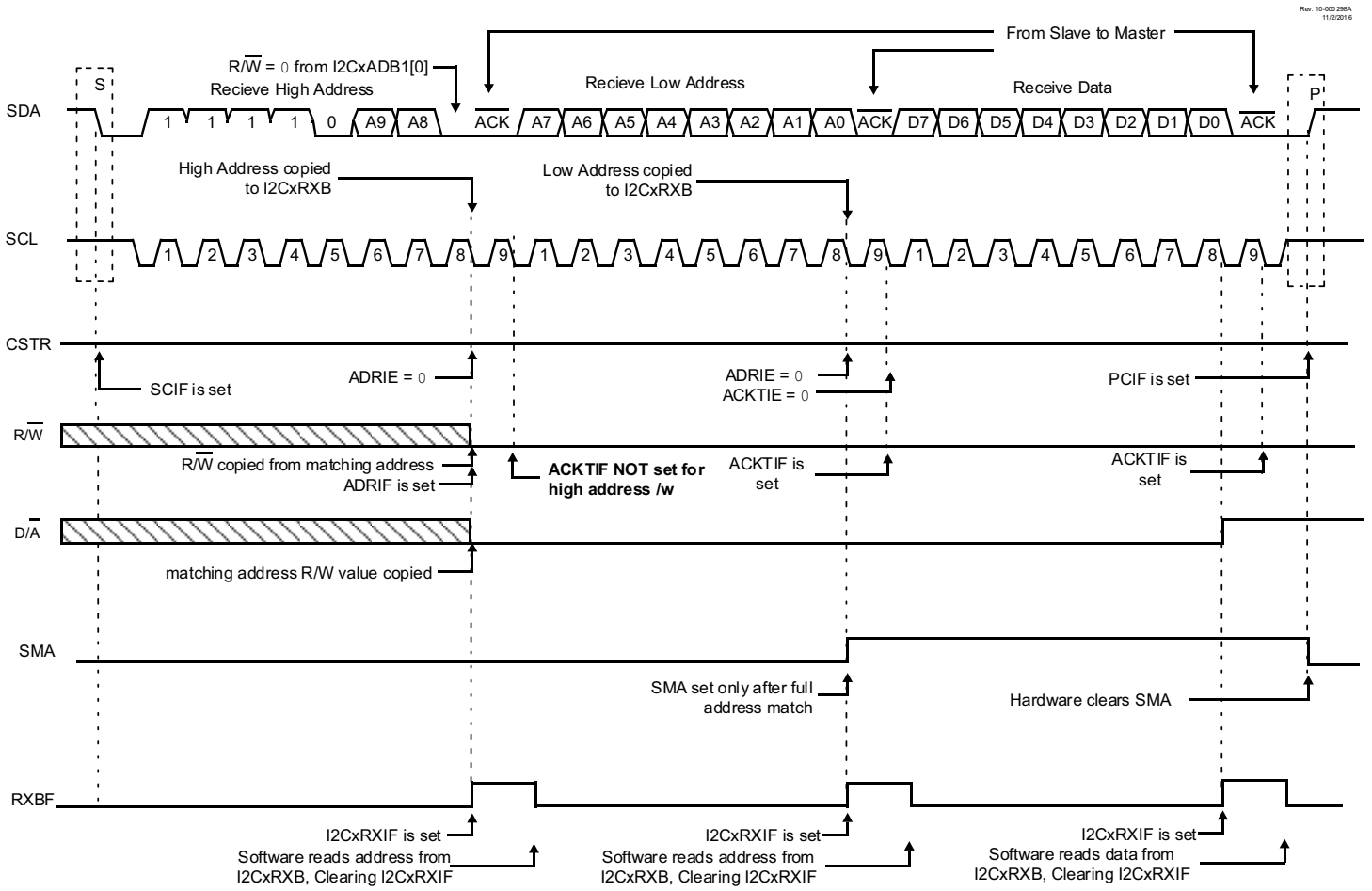
When TXR is cleared, the data will be taken from the TXFIFO, and the TXFIFO occupancy will not decrease. If the TXFIFO is empty, the most recently received data will be transmitted, and the TXUIF bit will not be set. However, if the TRIS bit associated with the SDO pin is set, clearing the TXR bit will cause the SPI module to not output any data to the SDO pin.

TABLE 34-2: SLAVE MODE TRANSMIT

TRISxn(1)	TXR	SS	TXBE	SDO State
0	0	FALSE	0	Drives state determined by LATxn(2)
0	0	FALSE	1	Drives state determined by LATxn(2)
0	0	TRUE	0	Outputs the oldest byte in the TXFIFO Does not remove data from the TXFIFO
0	0	TRUE	1	Outputs the most recently received byte
0	1	FALSE	0	Drives state determined by LATxn(2)
0	1	FALSE	1	Drives state determined by LATxn(2)
0	1	TRUE	0	Outputs the oldest byte in the TXFIFO Removes transmitted byte from the TXFIFO Decrements occupancy of TXFIFO
0	1	TRUE	1	Outputs the most recently received byte Sets the TXUIF bit of SPIxINTF
1	0	FALSE	0	Tri-stated
1	0	FALSE	1	Tri-stated
1	0	TRUE	0	Tri-stated
1	0	TRUE	1	Tri-stated
1	1	FALSE	0	Tri-stated
1	1	FALSE	1	Tri-stated
1	1	TRUE	0	Outputs the oldest byte in the TXFIFO Removes transmitted byte from the TXFIFO Decrements occupancy of TXFIFO
1	1	TRUE	1	Outputs the most recently received byte Sets the TXUIF bit of SPIxINTF

Note 1: TRISxn is the bit in the TRISx register corresponding to the pin that SDO has been assigned with PPS.
2: LATxn is the bit in the LATx register corresponding to the pin that SDO has been assigned with PPS.

FIGURE 35-11: I²C SLAVE, 10-BIT ADDRESS, RECEPTION WITH STOP (ADB = 1)



REGISTER 35-12: I2CxADR0 – I²C ADDRESS 0 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
ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	HS = Hardware set HC = Hardware clear

bit 7-0 **ADR<7-0>:** Address 1 bits

MODE<2:0> = 00x | 11x - 7-bit Slave/Multi-Master Modes

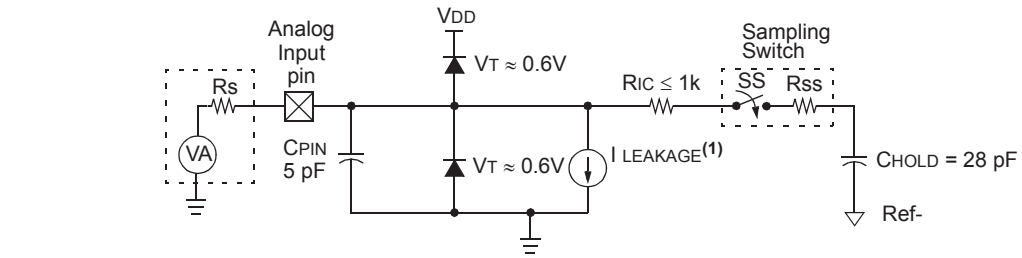
ADR0<7:1>:7-bit Slave Address

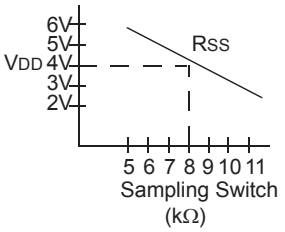
ADR0<0>: Unused in this mode; bit state is a don't care

MODE<2:0> = 01x - 10-bit Slave Modes

ADR0<7:0>: Eight Least Significant bits of 10-bit address 0

FIGURE 38-4: ANALOG INPUT MODEL



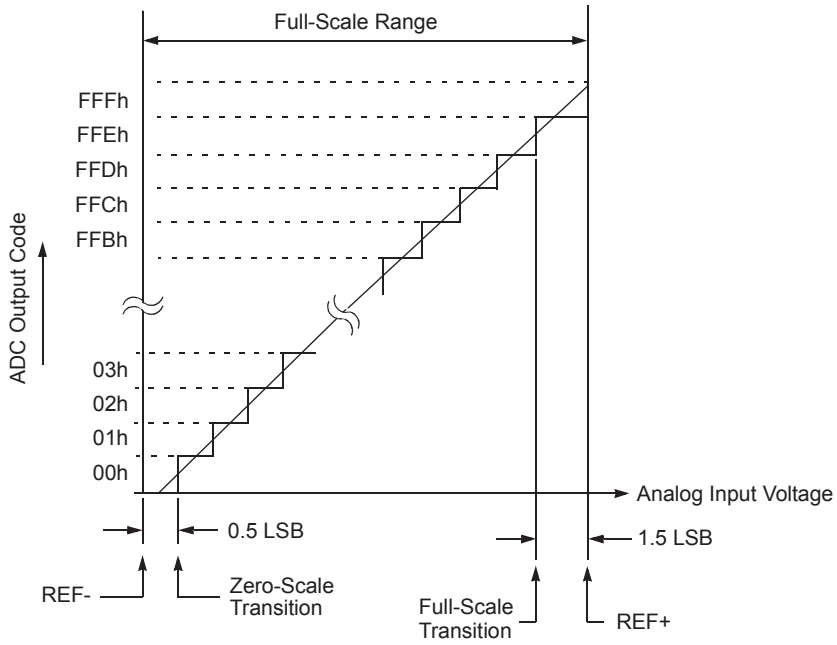


Legend:

CHOLD	=	Sample/Hold Capacitance
CPIN	=	Input Capacitance
I LEAKAGE	=	Leakage current at the pin due to various junctions
RIC	=	Interconnect Resistance
RSS	=	Resistance of Sampling Switch
SS	=	Sampling Switch
VT	=	Threshold Voltage

Note 1: Refer to Table 46-4 (parameter D340 and D341).

FIGURE 38-5: ADC TRANSFER FUNCTION



40.13 Register Definitions: Comparator Control

Long bit name prefixes for the Comparators are shown in Table 40-2. Refer to **Section 1.4.2.2 "Long Bit Names"** for more information.

TABLE 40-2:

Peripheral	Bit Name Prefix
C1	C1
C2	C2

REGISTER 40-1: CMxCON0: COMPARATOR x CONTROL REGISTER 0

R/W-0/0	R-0/0	U-0	R/W-0/0	U-0	U-1	R/W-0/0	R/W-0/0
EN	OUT	—	POL	—	—	HYS	SYNC
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 7 **EN:** Comparator Enable bit
1 = Comparator is enabled
0 = Comparator is disabled and consumes no active power

bit 6 **OUT:** Comparator Output bit
If POL = 0 (noninverted polarity):
1 = CxVP > CxVN
0 = CxVP < CxVN
If POL = 1 (inverted polarity):
1 = CxVP < CxVN
0 = CxVP > CxVN

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

bit 4 **POL:** Comparator Output Polarity Select bit
1 = Comparator output is inverted
0 = Comparator output is not inverted

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

bit 2 **Unimplemented:** Read as '1'

bit 1 **HYS:** Comparator Hysteresis Enable bit
1 = Comparator hysteresis enabled
0 = Comparator hysteresis disabled

bit 0 **SYNC:** Comparator Output Synchronous Mode bit
1 = Comparator output to Timer1/3/5 and I/O pin is synchronous to changes on Timer1 clock source.
0 = Comparator output to Timer1/3/5 and I/O pin is asynchronous
Output updated on the falling edge of Timer1/3/5 clock source.

41.10 Register Definitions: HLVD Control

Long bit name prefixes for the HLVD peripheral is shown in Table 41-1. Refer to **Section 1.4.2.2 "Long Bit Names"** for more information.

TABLE 41-1:

Peripheral	Bit Name Prefix
HLVD	HLVD

REGISTER 41-1: HLVDCON1: LOW-VOLTAGE DETECT CONTROL REGISTER 1

U-0	U-0	U-0	U-0	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u
—	—	—	—	SEL<3:0>			
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	u = Bit is unchanged

bit 7-4 **Unimplemented:** Read as '0'

bit 3-0 **SEL<3:0>:** High/Low Voltage Detection Limit Selection bits

SEL<3:0>	Typical Voltage
1111	Reserved
1110	4.63V
1101	4.32V
1100	4.12V
1011	3.91V
1010	3.71V
1001	3.60V
1000	3.4V
0111	3.09V
0110	2.88V
0101	2.78V
0100	2.57V
0011	2.47V
0010	2.26V
0001	2.06V
0000	1.85V

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

Syntax: CPFSEQ f {,a}

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

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

Status Affected: None

Encoding:

0110	001a	ffff	ffff

Description: Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction.
If 'f' = W, then the fetched instruction is discarded and a NOP is executed instead, making this a 2-cycle instruction.
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 43.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:
```
HERE    CPFSEQ REG, 0
NEQUAL  :
EQUAL   :
```

Before Instruction
PC Address = HERE
W = ?
REG = ?
After Instruction
If REG = W;
PC = Address (EQUAL)
If REG ≠ W;
PC = Address (NEQUAL)

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

Syntax: CPFSGT f {,a}

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

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

Status Affected: None

Encoding:

0110	010a	ffff	ffff

Description: Compares the contents of data memory location 'f' to the contents of the W by performing an unsigned subtraction.
If the contents of 'f' are greater than the contents of WREG, then the fetched instruction is discarded and a NOP is executed instead, making this a 2-cycle instruction.
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 43.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:
```
HERE    CPFSGT REG, 0
NGREATER :
GREATER  :
```

Before Instruction
PC = Address (HERE)
W = ?
After Instruction
If REG > W;
PC = Address (GREATER)
If REG ≤ W;
PC = Address (NGREATER)

CPFSLT **Compare f with W, skip if f < W**

Syntax: CPFSLT f {,a}

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

Operation: (f) – (W),
skip if (f) < (W)
(unsigned comparison)

Status Affected: None

Encoding:

0110	000a	ffff	ffff

Description: Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction.
If the contents of 'f' are less than the contents of W, then the fetched instruction is discarded and a NOP is executed instead, making this a 2-cycle instruction.
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:
```
HERE    CPFSLT REG, 1
NLESS   :
LESS    :
```

Before Instruction
PC = Address (HERE)
W = ?
After Instruction
If REG < W;
PC = Address (LESS)
If REG ≥ W;
PC = Address (NLESS)

DAW **Decimal Adjust W Register**

Syntax: DAW

Operands: None

Operation: If [W<3:0> > 9] or [DC = 1] then
(W<3:0>) + 6 → W<3:0>;
else
(W<3:0>) → W<3:0>;

If [W<7:4> + DC > 9] or [C = 1] then
(W<7:4>) + 6 + DC → W<7:4> ;
else
(W<7:4>) + DC → W<7:4>

Status Affected: C

Encoding:

0000	0000	0000	0111

Description: DAW adjusts the 8-bit value in W, resulting from the earlier addition of two variables (each in packed BCD format) and produces a correct packed BCD result.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register W	Process Data	Write W

Example1:
```
DAW
```
Before Instruction
W = A5h
C = 0
DC = 0
After Instruction
W = 05h
C = 1
DC = 0

Example 2:
Before Instruction
W = CEh
C = 0
DC = 0
After Instruction
W = 34h
C = 1
DC = 0

RRNCF **Rotate Right f (No Carry)**

Syntax: RRNCF f {,d {,a}}

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

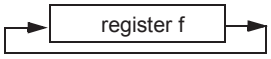
Operation: (f<n>) → dest<n – 1>,
(f<0>) → dest<7>

Status Affected: N, Z

Encoding:

0100	00da	ffff	ffff

Description: The contents of register 'f' are rotated one bit to the right. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).
If 'a' is '0', the Access Bank will be selected (default), overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 43.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.



Words: 1

Cycles: 1

Q Cycle Activity:

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

Example 1: RRNCF REG, 1, 0

Before Instruction
REG = 1101 0111
After Instruction
REG = 1110 1011

Example 2: RRNCF REG, 0, 0

Before Instruction
W = ?
REG = 1101 0111
After Instruction
W = 1110 1011
REG = 1101 0111

SETF **Set f**

Syntax: SETF f {,a}

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

Operation: FFh → f

Status Affected: None

Encoding:

0110	100a	ffff	ffff

Description: The contents of the specified register are set to FFh.
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 43.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example: SETF REG, 1

Before Instruction
REG = 5Ah
After Instruction
REG = FFh