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	48MHz
Connectivity	I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LCD, LVD, POR, PWM, WDT
Number of I/O	51
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	3.8K x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 3.6V
Data Converters	A/D 12x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	80-TQFP
Supplier Device Package	80-TQFP (12x12)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f86j72-i-pt

TABLE 1-2: PIC18F8XJ72 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	TQFP			
				PORTE is a bidirectional I/O port.
RE0/LCDBIAS1	4			
RE0		I/O	ST	Digital I/O.
LCDBIAS1		I	Analog	BIAS1 input for LCD.
RE1/LCDBIAS2	3			
RE1		I/O	ST	Digital I/O.
LCDBIAS2		I	Analog	BIAS2 input for LCD.
RE2/LCDBIAS3	80			
RE2		I/O	ST	Digital I/O.
LCDBIAS3		I	Analog	BIAS3 input for LCD.
RE3/COM0	79			
RE3		I/O	ST	Digital I/O.
COM0		O	Analog	COM0 output for LCD.
RE4/COM1	78			
RE4		I/O	ST	Digital I/O.
COM1		O	Analog	COM1 output for LCD.
RE5/COM2	77			
RE5		I/O	ST	Digital I/O.
COM2		O	Analog	COM2 output for LCD.
RE6/COM3	76			
RE6		I/O	ST	Digital I/O.
COM3		O	Analog	COM3 output for LCD.
RE7/CCP2/SEG31	75			
RE7		I/O	ST	Digital I/O.
CCP2(2)		I/O	ST	Capture 2 input/Compare 2 output/PWM2 output.
SEG31		O	Analog	SEG31 output for LCD.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; Analog = Analog input; I^2C = I^2C/SMBus compatible input; OD = Open-Drain (no P diode to V_{DD}); I = Input; O = Output; P = Power

Note 1: Default assignment for CCP2 when the CCP2MX Configuration bit is set.

2: Alternate assignment for CCP2 when the CCP2MX Configuration bit is cleared.

6.3 Data Addressing Modes

Note: The execution of some instructions in the core PIC18 instruction set are changed when the PIC18 extended instruction set is enabled. See **Section 6.5 "Data Memory and the Extended Instruction Set"** for more information.

While the program memory can be addressed in only one way – through the program counter – information in the data memory space can be addressed in several ways. For most instructions, the addressing mode is fixed. Other instructions may use up to three modes, depending on which operands are used and whether or not the extended instruction set is enabled.

The addressing modes are:

- Inherent
- Literal
- Direct
- Indirect

An additional addressing mode, Indexed Literal Offset, is available when the extended instruction set is enabled (XINST Configuration bit = 1). Its operation is discussed in greater detail in **Section 6.5.1 "Indexed Addressing with Literal Offset"**.

6.3.1 INHERENT AND LITERAL ADDRESSING

Many PIC18 control instructions do not need any argument at all; they either perform an operation that globally affects the device, or they operate implicitly on one register. This addressing mode is known as Inherent Addressing. Examples include SLEEP, RESET and DAW.

Other instructions work in a similar way, but require an additional explicit argument in the opcode. This is known as Literal Addressing mode, because they require some literal value as an argument. Examples include ADDLW and MOVLW, which respectively, add or move a literal value to the W register. Other examples include CALL and GOTO, which include a 20-bit program memory address.

6.3.2 DIRECT ADDRESSING

Direct Addressing specifies all or part of the source and/or destination address of the operation within the opcode itself. The options are specified by the arguments accompanying the instruction.

In the core PIC18 instruction set, bit-oriented and byte-oriented instructions use some version of Direct Addressing by default. All of these instructions include some 8-bit literal address as their Least Significant Byte. This address specifies either a register address in one of the banks of data RAM (**Section 6.3.3 "General Purpose Register File"**) or a location in the Access Bank (**Section 6.3.2 "Access Bank"**) as the data source for the instruction.

The Access RAM bit, 'a', determines how the address is interpreted. When 'a' is '1', the contents of the BSR (**Section 6.3.1 "Bank Select Register"**) are used with the address to determine the complete 12-bit address of the register. When 'a' is '0', the address is interpreted as being a register in the Access Bank. Addressing that uses the Access RAM is sometimes also known as Direct Forced Addressing mode.

A few instructions, such as MOVFF, include the entire 12-bit address (either source or destination) in their opcodes. In these cases, the BSR is ignored entirely.

The destination of the operation's results is determined by the destination bit, 'd'. When 'd' is '1', the results are stored back in the source register, overwriting its original contents. When 'd' is '0', the results are stored in the W register. Instructions without the 'd' argument have a destination that is implicit in the instruction; their destination is either the target register being operated on or the W register.

6.3.3 INDIRECT ADDRESSING

Indirect Addressing allows the user to access a location in data memory without giving a fixed address in the instruction. This is done by using File Select Registers (FSRs) as pointers to the locations to be read or written to. Since the FSRs are themselves located in RAM as Special Function Registers, they can also be directly manipulated under program control. This makes FSRs very useful in implementing data structures such as tables and arrays in data memory.

The registers for Indirect Addressing are also implemented with Indirect File Operands (INDFs) that permit automatic manipulation of the pointer value with auto-incrementing, auto-decrementing or offsetting with another value. This allows for efficient code using loops, such as the example of clearing an entire RAM bank in Example 6-5. It also enables users to perform Indexed Addressing and other Stack Pointer operations for program memory in data memory.

EXAMPLE 6-5: HOW TO CLEAR RAM (BANK 1) USING INDIRECT ADDRESSING

```
        LFSR    FSR0, 100h ;
NEXT    CLRF    POSTINC0   ; Clear INDF
                           ; register then
                           ; inc pointer
        BTFSS   FSR0H, 1   ; All done with
                           ; Bank1?
        BRA     NEXT       ; NO, clear next
CONTINUE                   ; YES, continue
```

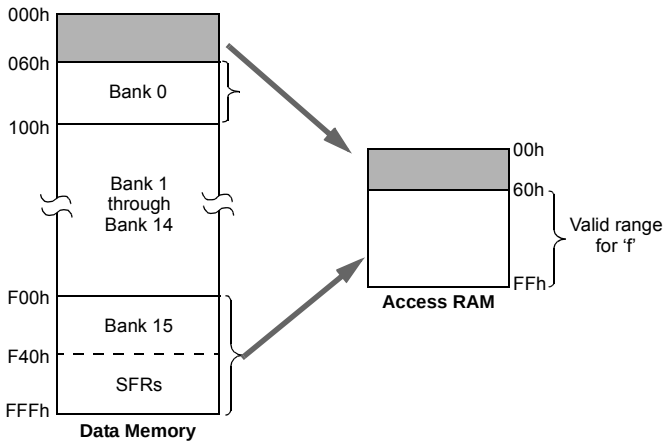
FIGURE 6-2: COMPARING ADDRESSING OPTIONS FOR BIT-ORIENTED AND BYTE-ORIENTED INSTRUCTIONS (EXTENDED INSTRUCTION SET ENABLED)

EXAMPLE INSTRUCTION: `ADDWF, f, d, a` (Opcode: `0010 01da ffff ffff`)

When a = 0 and f ≥ 60h:

The instruction executes in Direct Forced mode. 'f' is interpreted as a location in the Access RAM between 060h and FFFh. This is the same as locations, F60h to FFFh (Bank 15), of data memory.

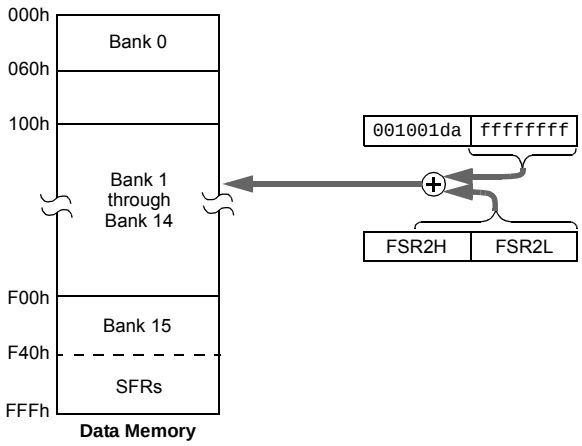
Locations below 060h are not available in this addressing mode.



When a = 0 and f ≤ 5Fh:

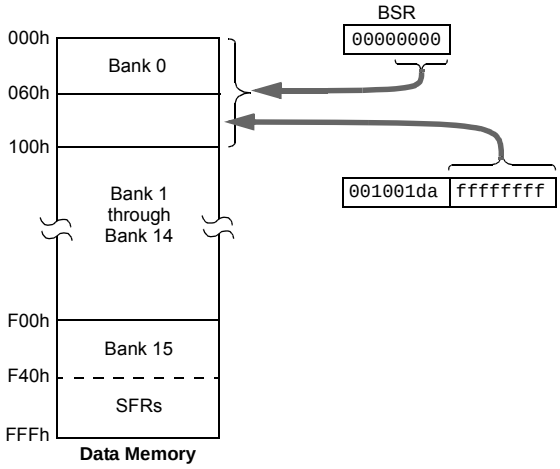
The instruction executes in Indexed Literal Offset mode. 'f' is interpreted as an offset to the address value in FSR2. The two are added together to obtain the address of the target register for the instruction. The address can be anywhere in the data memory space.

Note that in this mode, the correct syntax is now:
`ADDWF [k], d`
where 'k' is the same as 'f'.



When a = 1 (all values of f):

The instruction executes in Direct mode (also known as Direct Long mode). 'f' is interpreted as a location in one of the 16 banks of the data memory space. The bank is designated by the Bank Select Register (BSR). The address can be in any implemented bank in the data memory space.



16.3 Compare Mode

In Compare mode, the 16-bit CCPR2 register value is constantly compared against either the TMR1 or TMR3 register pair value. When a match occurs, the CCP2 pin can be:

- driven high
- driven low
- toggled (high-to-low or low-to-high)
- remain unchanged (that is, reflects the state of the I/O latch)

The action on the pin is based on the value of the mode select bits (CCP2M<3:0>). At the same time, the interrupt flag bit, CCP2IF, is set.

16.3.1 CCP PIN CONFIGURATION

The user must configure the CCPx pin as an output by clearing the appropriate TRIS bit.

Note:	Clearing the CCP2CON register will force the RC1 or RE7 compare output latch (depending on device configuration) to the default low level. This is not the PORTC or PORTE I/O data latch.

16.3.2 TIMER1/TIMER3 MODE SELECTION

Timer1 and/or Timer3 must be running in Timer mode, or Synchronized Counter mode, if the CCP module is using the compare feature. In Asynchronous Counter mode, the compare operation may not work.

16.3.3 SOFTWARE INTERRUPT MODE

When the Generate Software Interrupt mode is chosen (CCP2M<3:0> = 1010), the CCP2 pin is not affected. Only a CCP interrupt is generated, if enabled, and the CCP2IE bit is set.

16.3.4 SPECIAL EVENT TRIGGER

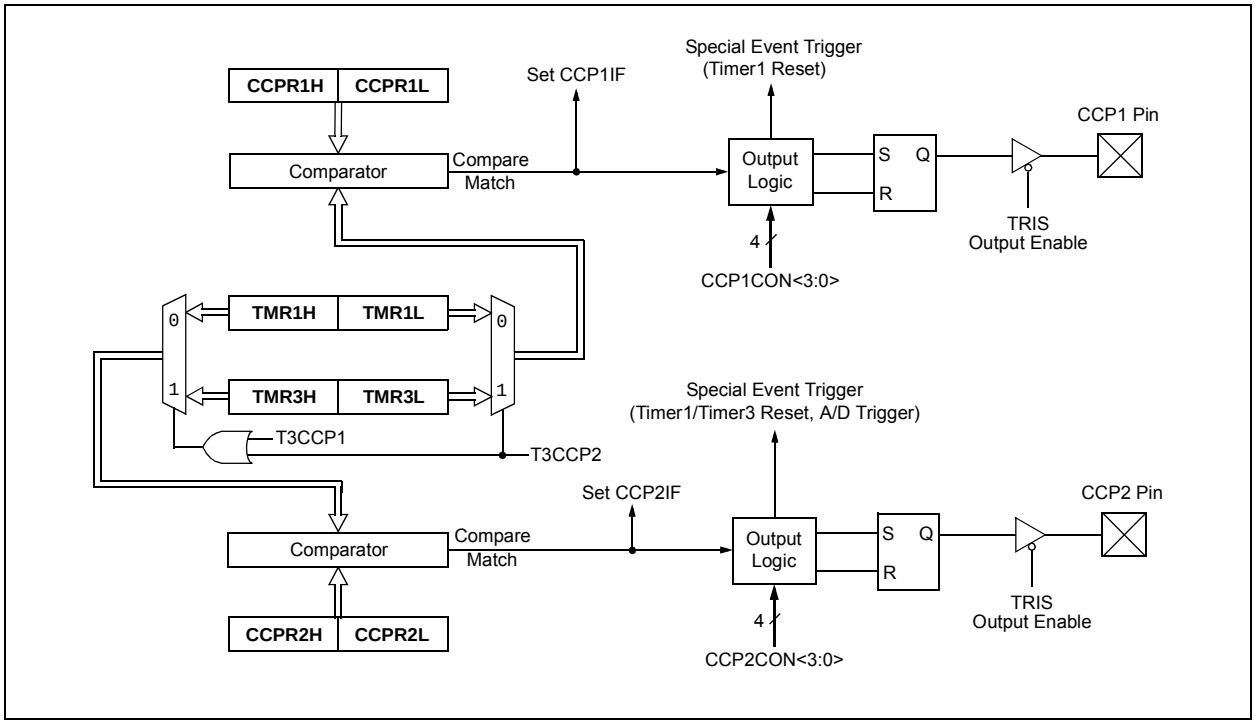
Both CCP modules are equipped with a Special Event Trigger. This is an internal hardware signal generated in Compare mode to trigger actions by other modules. The Special Event Trigger is enabled by selecting the Compare Special Event Trigger mode (CCP2M<3:0> = 1011).

For either CCP module, the Special Event Trigger resets the Timer register pair for whichever timer resource is currently assigned as the module's time base. This allows the CCPRx registers to serve as a programmable period register for either timer.

The Special Event Trigger for CCP2 can also start an A/D conversion. In order to do this, the A/D Converter must already be enabled.

Note:	The Special Event Trigger of CCP1 only resets Timer1/Timer3 and cannot start an A/D conversion even when the A/D Converter is enabled.

FIGURE 16-3: COMPARE MODE OPERATION BLOCK DIAGRAM



16.4.2 PWM DUTY CYCLE

The PWM duty cycle is specified by writing to the CCPR2L register and to the CCP2CON<5:4> bits. Up to 10-bit resolution is available. The CCPR2L contains the eight MSbs and the CCP2CON<5:4> bits contain the two LSbs. This 10-bit value is represented by CCPR2L:CCP2CON<5:4>. The following equation is used to calculate the PWM duty cycle in time:

EQUATION 16-2:

$$\text{PWM Duty Cycle} = (\text{CCPR2L:CCP2CON<5:4>}) \bullet T_{OSC} \bullet (\text{TMR2 Prescale Value})$$

CCPR2L and CCP2CON<5:4> can be written to at any time, but the duty cycle value is not latched into CCPR2H until after a match between PR2 and TMR2 occurs (i.e., the period is complete). In PWM mode, CCPR2H is a read-only register.

The CCPR2H register and a 2-bit internal latch are used to double-buffer the PWM duty cycle. This double-buffering is essential for glitchless PWM operation.

When the CCPR2H and 2-bit latch match TMR2, concatenated with an internal 2-bit Q clock or two bits of the TMR2 prescaler, the CCP2 pin is cleared.

The maximum PWM resolution (bits) for a given PWM frequency is given by the equation:

EQUATION 16-3:

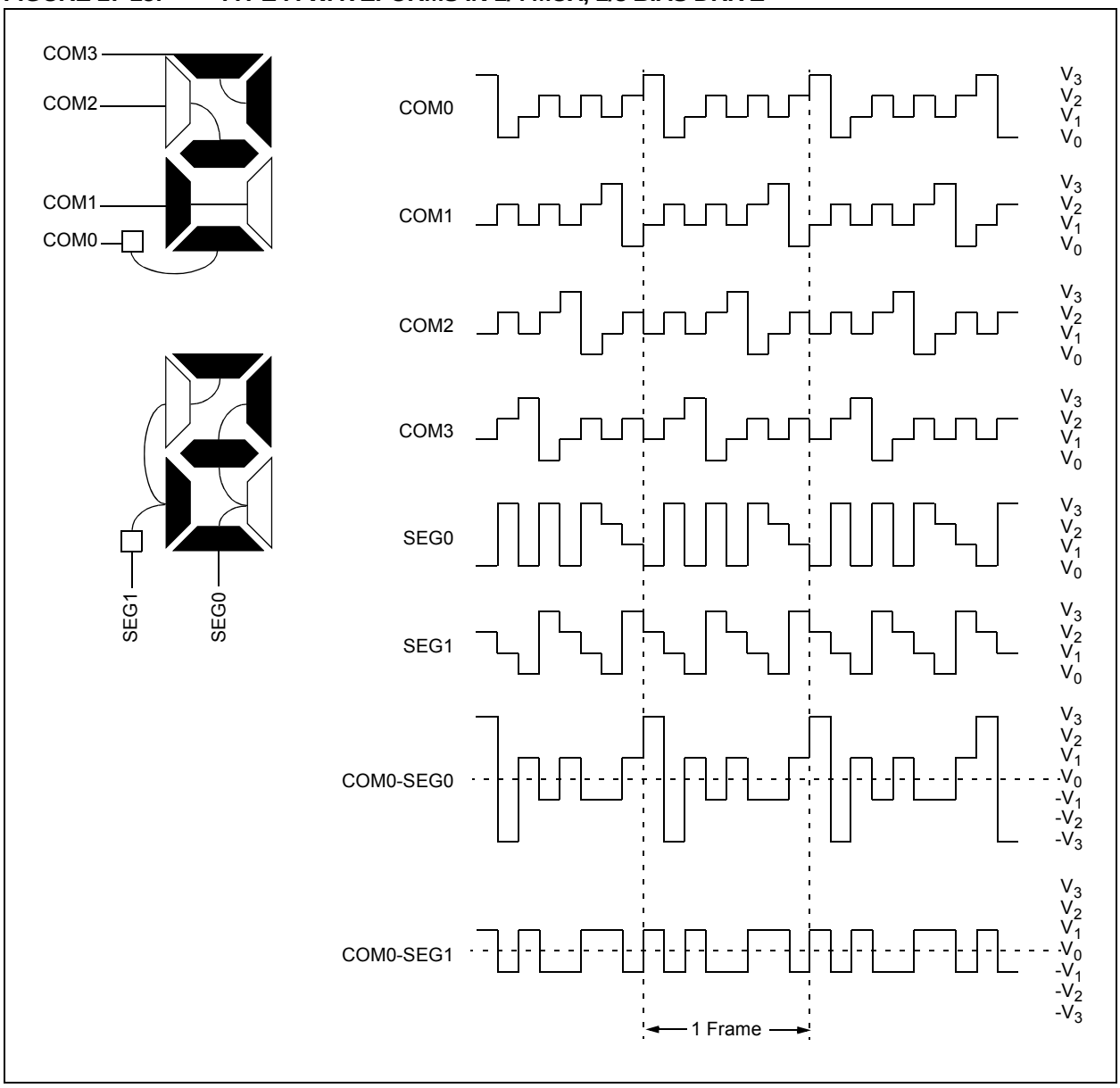
$$\text{PWM Resolution (max)} = \frac{\log\left(\frac{F_{OSC}}{F_{PWM}}\right)}{\log(2)} \text{bits}$$

Note: If the PWM duty cycle value is longer than the PWM period, the CCP2 pin will not be cleared.

TABLE 16-4: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS AT 40 MHz

PWM Frequency	2.44 kHz	9.77 kHz	39.06 kHz	156.25 kHz	312.50 kHz	416.67 kHz
Timer Prescaler (1, 4, 16)	16	4	1	1	1	1
PR2 Value	FFh	FFh	FFh	3Fh	1Fh	17h
Maximum Resolution (bits)	14	12	10	8	7	6.58

FIGURE 17-15: TYPE-A WAVEFORMS IN 1/4 MUX, 1/3 BIAS DRIVE



17.11 Configuring the LCD Module

The following is the sequence of steps to configure the LCD module.

1. Select the frame clock prescale using bits, LP<3:0> (LCDPS<3:0>).
2. Configure the appropriate pins to function as segment drivers using the LCDSEx registers.
3. Configure the appropriate pins as inputs using the TRISx registers.
4. Configure the LCD module for the following using the LCDCON register:

   Multiplex and Bias mode (LMUX<1:0>)

   Timing source (CS<1:0>)

   Sleep mode (SLPEN)
5. Write initial values to pixel data registers, LCDDATA0 through LCDDATA23.
6. Configure the LCD regulator:
   a) If M2 or M3 bias configuration is to be used, turn off the regulator by setting CKSEL<1:0> (LCDREG<1:0>) to '00'. Set or clear the CPEN bit (LCDREG<6>) to select Mode 2 or Mode 3, as appropriate.
   b) If M0 or M1 bias generation is to be used:

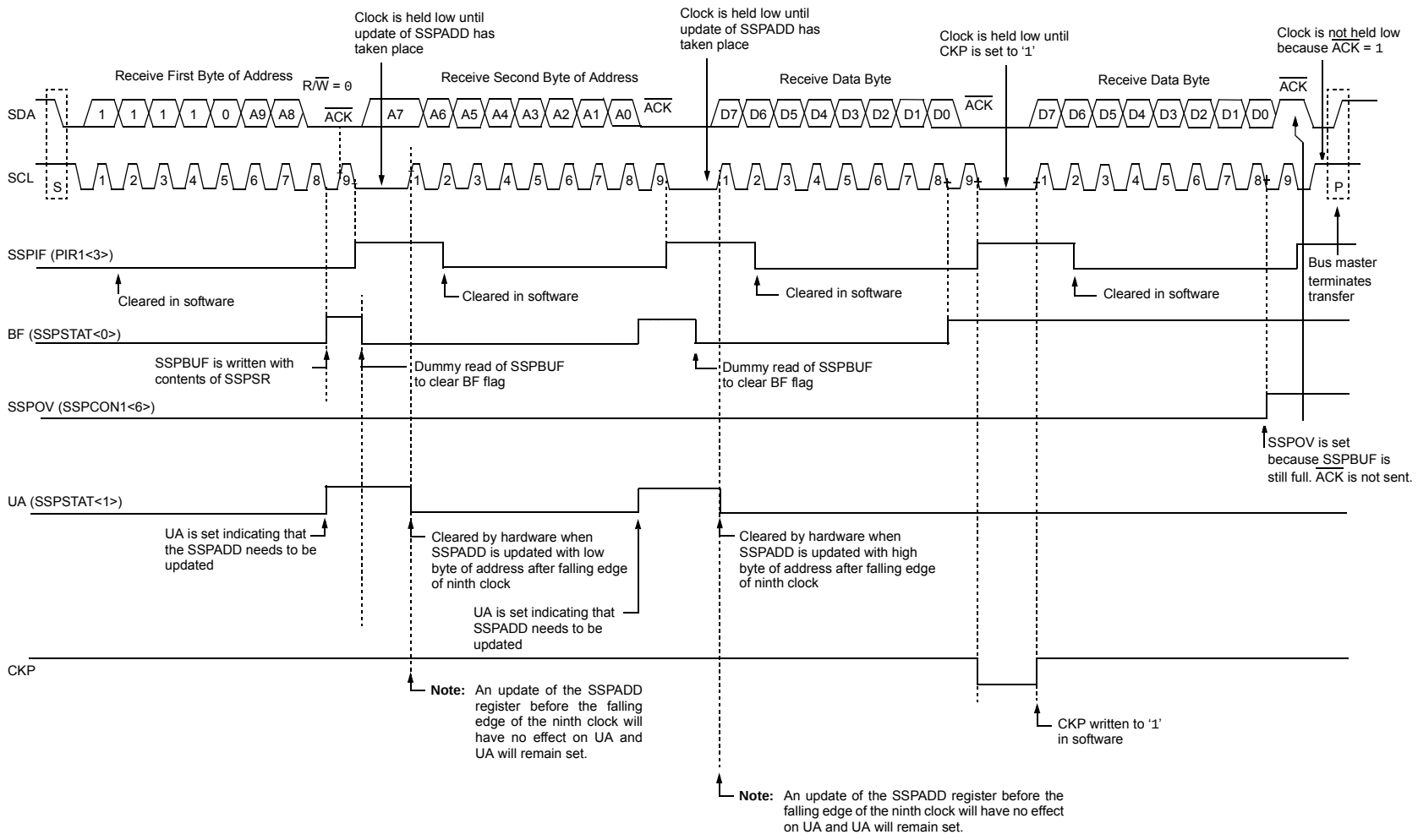
      Set the V_{BIAS} level using the BIAS<2:0> bits (LCDREG<5:3>).

      Set or clear the CPEN bit to enable or disable the charge pump.

      Set or clear the MODE13 bit (LCDREG<2>) to select the Bias mode.

      Select a regulator clock source using the CKSEL<1:0> bits.
7. Clear LCD Interrupt Flag, LCDIF (PIR3<6>), and if desired, enable the interrupt by setting the LCDIE bit (PIE3<6>).
8. Enable the LCD module by setting the LCDEN bit (LCDCON<7>).

FIGURE 18-16: I²C SLAVE MODE TIMING WITH SEN = 1 (RECEPTION, 10-BIT ADDRESSING)



19.4 EUSART Synchronous Master Mode

The Synchronous Master mode is entered by setting the CSRC bit (TXSTA<7>). In this mode, the data is transmitted in a half-duplex manner (i.e., transmission and reception do not occur at the same time). When transmitting data, the reception is inhibited and vice versa. Synchronous mode is entered by setting bit, SYNC (TXSTA<4>). In addition, enable bit, SPEN (RCSTA1<7>), is set in order to configure the TX1 and RX1 pins to CK1 (clock) and DT1 (data) lines, respectively.

The Master mode indicates that the processor transmits the master clock on the CK1 line. Clock polarity is selected with the TXCKP bit (BAUDCON<4>). Setting TXCKP sets the Idle state on CK1 as high, while clearing the bit sets the Idle state as low. This option is provided to support Microwire devices with this module.

19.4.1 EUSART SYNCHRONOUS MASTER TRANSMISSION

The EUSART transmitter block diagram is shown in Figure 19-3. The heart of the transmitter is the Transmit (Serial) Shift register (TSR). The Shift register obtains its data from the Read/Write Transmit Buffer register, TXREG1. The TXREG1 register is loaded with data in software. The TSR register is not loaded until the last bit has been transmitted from the previous load. As soon as the last bit is transmitted, the TSR is loaded with new data from the TXREG1 (if available).

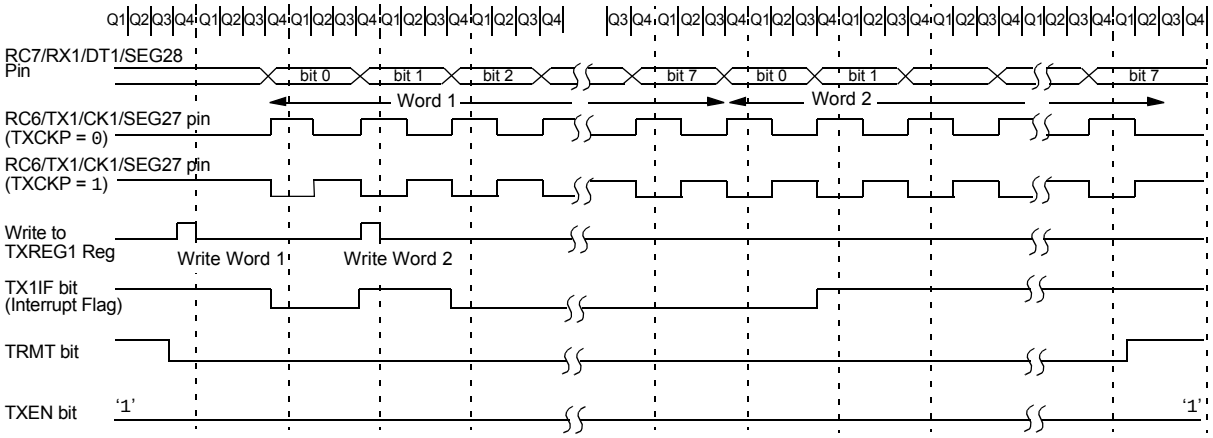
Once the TXREG1 register transfers the data to the TSR register (occurs in one TCYCLE), the TXREG1 is empty and the TX1IF flag bit (PIR1<4>) is set. The interrupt can be enabled or disabled by setting or clearing the interrupt enable bit, TX1IE (PIE1<4>). TX1IF is set regardless of the state of enable bit, TX1IE; it cannot be cleared in software. It will reset only when new data is loaded into the TXREG1 register.

While flag bit, TX1IF, indicates the status of the TXREG1 register, another bit, TRMT (TXSTA<1>), shows the status of the TSR register. TRMT is a read-only bit which is set when the TSR is empty. No interrupt logic is tied to this bit so the user has to poll this bit in order to determine if the TSR register is empty. The TSR is not mapped in data memory so it is not available to the user.

To set up a Synchronous Master Transmission:

1. Initialize the SPBRGH1:SPBRG1 registers for the appropriate baud rate. Set or clear the BRG16 bit, as required, to achieve the desired baud rate.
2. Enable the synchronous master serial port by setting bits, SYNC, SPEN and CSRC.
3. If interrupts are desired, set enable bit, TX1IE.
4. If 9-bit transmission is desired, set bit, TX9.
5. Enable the transmission by setting bit, TXEN.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit, TX9D.
7. Start transmission by loading data to the TXREG1 register.
8. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

FIGURE 19-11: SYNCHRONOUS TRANSMISSION



Note: Sync Master mode, SPBRG1 = 0; continuous transmission of two 8-bit words.

20.3.2 AUSART ASYNCHRONOUS RECEIVER

The receiver block diagram is shown in Figure 20-4. The data is received on the RX2 pin and drives the data recovery block. The data recovery block is actually a high-speed shifter operating at x16 times the baud rate, whereas the main receive serial shifter operates at the bit rate or at FOSC. This mode would typically be used in RS-232 systems.

To set up an Asynchronous Reception:

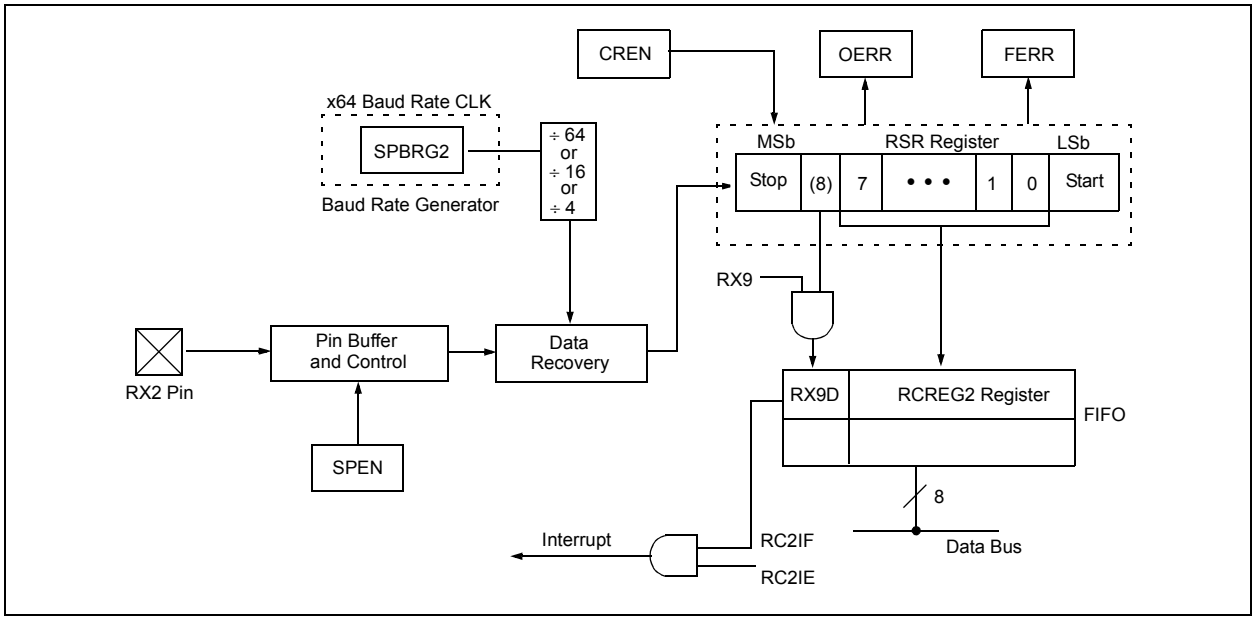
1. Initialize the SPBRG2 register for the appropriate baud rate. Set or clear the BRGH bit, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing bit, SYNC, and setting bit, SPEN.
3. If interrupts are desired, set enable bit, RC2IE.
4. If 9-bit reception is desired, set bit, RX9.
5. Enable the reception by setting bit, CREN.
6. Flag bit, RC2IF, will be set when reception is complete and an interrupt will be generated if enable bit, RC2IE, was set.
7. Read the RCSTA2 register to get the 9th bit (if enabled) and determine if any error occurred during reception.
8. Read the 8-bit received data by reading the RCREG2 register.
9. If any error occurred, clear the error by clearing enable bit, CREN.
10. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

20.3.3 SETTING UP 9-BIT MODE WITH ADDRESS DETECT

This mode would typically be used in RS-485 systems. To set up an Asynchronous Reception with Address Detect Enable:

1. Initialize the SPBRG2 register for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit.
3. If interrupts are required, set the RCEN bit and select the desired priority level with the RC2IP bit.
4. Set the RX9 bit to enable 9-bit reception.
5. Set the ADDEN bit to enable address detect.
6. Enable reception by setting the CREN bit.
7. The RC2IF bit will be set when reception is complete. The interrupt will be Acknowledged if the RC2IE and GIE bits are set.
8. Read the RCSTA2 register to determine if any error occurred during reception, as well as read bit 9 of data (if applicable).
9. Read RCREG2 to determine if the device is being addressed.
10. If any error occurred, clear the CREN bit.
11. If the device has been addressed, clear the ADDEN bit to allow all received data into the receive buffer and interrupt the CPU.

FIGURE 20-4: AUSART RECEIVE BLOCK DIAGRAM



21.5 A/D Conversions

Figure 21-1 shows the operation of the A/D Converter after the GO/$\overline{\text{DONE}}$ bit has been set and the ACQT<2:0> bits are cleared. A conversion is started after the following instruction to allow entry into Sleep mode before the conversion begins.

Figure 21-2 shows the operation of the A/D Converter after the GO/$\overline{\text{DONE}}$ bit has been set. The ACQT<2:0> bits are set to '010' and a 4 TAD acquisition time is selected before the conversion starts.

Clearing the GO/$\overline{\text{DONE}}$ bit during a conversion will abort the current conversion. The A/D Result register pair will NOT be updated with the partially completed A/D conversion sample. This means the ADRESH:ADRESL registers will continue to contain the value of the last completed conversion (or the last value written to the ADRESH:ADRESL registers).

After the A/D conversion is completed or aborted, a 2 TAD wait is required before the next acquisition can be started. After this wait, acquisition on the selected channel is automatically started.

Note: The GO/$\overline{\text{DONE}}$ bit should **NOT** be set in the same instruction that turns on the A/D.

21.6 Use of the CCP2 Trigger

An A/D conversion can be started by the "Special Event Trigger" of the CCP2 module. This requires that the CCP2M<3:0> bits (CCP2CON<3:0>) be programmed as '1011' and that the A/D module is enabled (ADON bit is set). When the trigger occurs, the GO/$\overline{\text{DONE}}$ bit will be set, starting the A/D acquisition and conversion, and the Timer1 (or Timer3) counter will be reset to zero. Timer1 (or Timer3) is reset to automatically repeat the A/D acquisition period with minimal software overhead (moving ADRESH:ADRESL to the desired location). The appropriate analog input channel must be selected and the minimum acquisition period is either timed by the user or an appropriate TACQ time is selected before the Special Event Trigger sets the GO/$\overline{\text{DONE}}$ bit (starts a conversion).

If the A/D module is not enabled (ADON is cleared), the Special Event Trigger will be ignored by the A/D module, but will still reset the Timer1 (or Timer3) counter.

FIGURE 21-1: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 000, TACQ = 0)

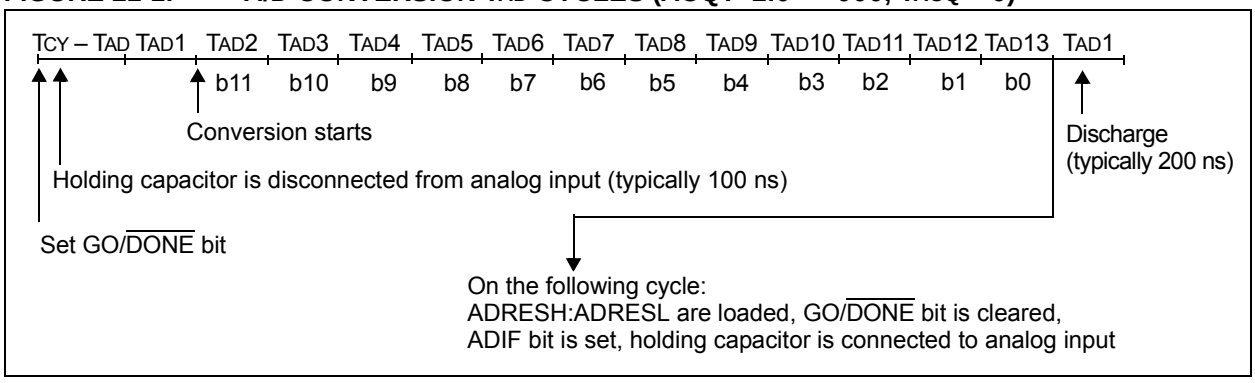
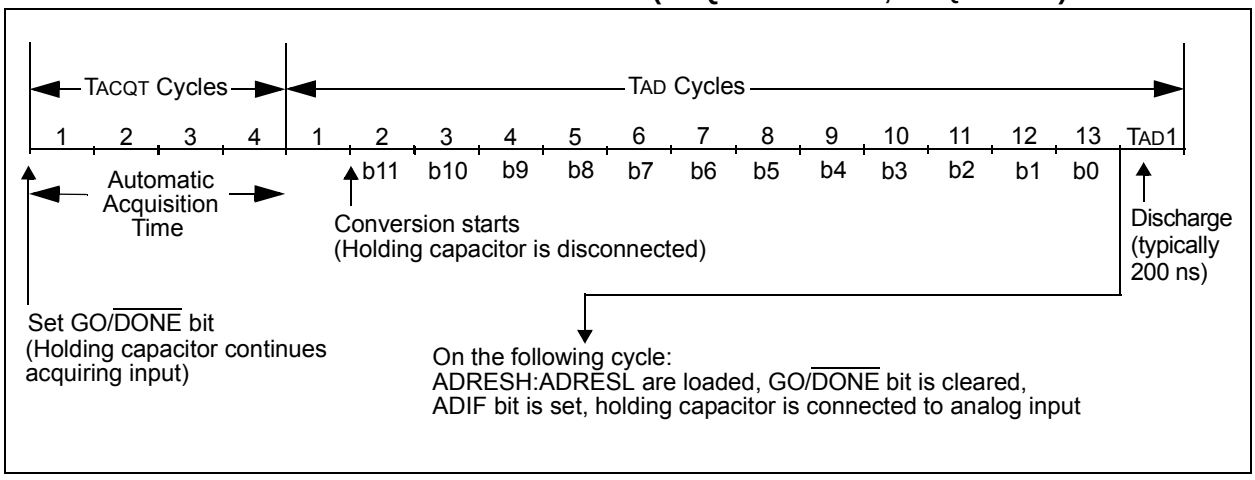


FIGURE 21-2: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 010, TACQ = 4 TAD)



EXAMPLE 22-5: READING DATA FROM AFE DURING INTERRUPT

```
////////////////////////////////////////////////////////////////////////////////////
// STEP 7: Reading AFE results in Interrupt Routine.
// ADC is configured in 16-bit result mode, thus 16-bit result of each Channel can be read.
// In this example DR is connected to INT0; after each convesion, DR issues interrupt to INT0.
// INT0 is configured as high priority interrupt
////////////////////////////////////////////////////////////////////////////////////

#pragma interrupt High_isr_routine
void High_isr_routine(void)
{
char    D_S_ADC_data1=0,D_S_ADC_data2=0,D_S_ADC_data3=0,D_S_ADC_data4=0,Dummy_Read=0;
    if((INTCONbits.INT0IF)&&(INTCONbits.INT0IE))
    {
        //  Disable all Chip selects of other devices connected to SPI
        LATDbits.LATD7=0;         //Chip select enable for Delta Sigma ADC
        SSP1BUF = 0x01;           //Address and Read command for Channel0 result MSB register
        while(!SSPSTATbits.BF);
        Dummy_Read=SSPBUF;        //Dummy read to clear Buffer Full Status bit
        SSPBUF =0x00;
        while(!SSPSTATbits.BF);
        D_S_ADC_data1=SSPBUF;     //Data from Channel0 MSB
        SSPBUF = 0x00;
        while(!SSPSTATbits.BF);
        D_S_ADC_data2=SSPBUF;     //Data from Channel0 LSB, Address automatically incremented
        SSPBUF = 0x00;
        while(!SSPSTATbits.BF);
        D_S_ADC_data3=SSPBUF;     //Data from Channel1 MSB, Address automatically incremented
        SSPBUF = 0x00;
        while(!SSPSTATbits.BF);
        D_S_ADC_data4=SSPBUF;     //Data from Channel1 LSB, Address automatically incremented
        LATDbits.LATD7=1;         //Disable chip select after read/write of registers

        INTCONbits.INT0IF=0;      //Clear INT0IF for next interrupt
    }
}

#pragma code High_isr=0x08
void High_ISR(void)
{
 _asm goto High_isr_routine _endasm

}
```

REGISTER 26-6: CONFIG3H: CONFIGURATION REGISTER 3 HIGH (BYTE ADDRESS 300005h)

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/WO-1
—(1)	—(1)	—(1)	—(1)	—	—	—	CCP2MX
bit 7							bit 0

Legend:		
R = Readable bit	WO = Write-Once bit	U = Unimplemented bit, read as '0'
-n = Value when device is unprogrammed	'1' = Bit is set	'0' = Bit is cleared

bit 7-1 **Unimplemented:** Read as '0'

bit 0 **CCP2MX:** CCP2 MUX bit
1 = CCP2 is multiplexed with RC1
0 = CCP2 is multiplexed with RE7

Note 1: The value of these bits in program memory should always be '1'. This ensures that the location is executed as a NOP if it is accidentally executed.

REGISTER 26-7: DEVID1: DEVICE ID REGISTER 1

R	R	R	R	R	R	R	R
DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0
bit 7							bit 0

Legend:
R = Read-only bit

bit 7-5 **DEV<2:0>:** Device ID bits
011 = PIC18F87J72
010 = PIC18F86J72

bit 4-0 **REV<4:0>:** Revision ID bits
These bits are used to indicate the device revision.

REGISTER 26-8: DEVID2: DEVICE ID REGISTER 2

R	R	R	R	R	R	R	R
DEV10(1)	DEV9(1)	DEV8(1)	DEV7(1)	DEV6(1)	DEV5(1)	DEV4(1)	DEV3(1)
bit 7							bit 0

Legend:
R = Read-only bit

bit 7-0 **DEV<10:3>:** Device ID bits(1)
These bits are used with the DEV<2:0> bits in the Device ID Register 1 to identify the part number.
0101 0000 = PIC18F87J72 family devices

Note 1: The values for DEV<10:3> may be shared with other device families. The specific device is always identified by using the entire DEV<10:0> bit sequence.

TABLE 27-2: PIC18F87J72 FAMILY INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word MSb	LSb	Status Affected	Notes
BYTE-ORIENTED OPERATIONS							
ADDWF	f, d, a	Add WREG and f	1	0010 01da	ffff ffff	C, DC, Z, OV, N	1, 2
ADDWFC	f, d, a	Add WREG and Carry bit to f	1	0010 00da	ffff ffff	C, DC, Z, OV, N	1, 2
ANDWF	f, d, a	AND WREG with f	1	0001 01da	ffff ffff	Z, N	1,2
CLRF	f, a	Clear f	1	0110 101a	ffff ffff	Z	2
COMF	f, d, a	Complement f	1	0001 11da	ffff ffff	Z, N	1, 2
CPFSEQ	f, a	Compare f with WREG, Skip =	1 (2 or	0110 001a	ffff ffff	None	4
CPFSGT	f, a	Compare f with WREG, Skip >	3)	0110 010a	ffff ffff	None	4
CPFSLT	f, a	Compare f with WREG, Skip <	1 (2 or	0110 000a	ffff ffff	None	1, 2
DECF	f, d, a	Decrement f	3)	0000 01da	ffff ffff	C, DC, Z, OV, N	1, 2, 3, 4
DECFSZ	f, d, a	Decrement f, Skip if 0	1 (2 or	0010 11da	ffff ffff	None	1, 2, 3, 4
DCFSNZ	f, d, a	Decrement f, Skip if Not 0	3)	0100 11da	ffff ffff	None	1, 2
INCF	f, d, a	Increment f	1	0010 10da	ffff ffff	C, DC, Z, OV, N	1, 2, 3, 4
INCFSZ	f, d, a	Increment f, Skip if 0	1 (2 or	0011 11da	ffff ffff	None	4
INFSNZ	f, d, a	Increment f, Skip if Not 0	3)	0100 10da	ffff ffff	None	1, 2
IORWF	f, d, a	Inclusive OR WREG with f	1 (2 or	0001 00da	ffff ffff	Z, N	1, 2
MOVF	f, d, a	Move f	3)	0101 00da	ffff ffff	Z, N	1
MOVFF	f_s, f_d	Move f_s (source) to 1st word	1	1100 ffff	ffff ffff	None	
		f_d (destination) 2nd word	1 (2 or	1111 ffff	ffff ffff		
MOVWF	f, a	Move WREG to f	3)	0110 111a	ffff ffff	None	
MULWF	f, a	Multiply WREG with f	1 (2 or	0000 001a	ffff ffff	None	1, 2
NEGF	f, a	Negate f	3)	0110 110a	ffff ffff	C, DC, Z, OV, N	
RLCF	f, d, a	Rotate Left f through Carry	1	0011 01da	ffff ffff	C, Z, N	1, 2
RLNCF	f, d, a	Rotate Left f (No Carry)	1	0100 01da	ffff ffff	Z, N	
RRCF	f, d, a	Rotate Right f through Carry	2	0011 00da	ffff ffff	C, Z, N	
RRNCF	f, d, a	Rotate Right f (No Carry)		0100 00da	ffff ffff	Z, N	
SETF	f, a	Set f	1	0110 100a	ffff ffff	None	1, 2
SUBFWB	f, d, a	Subtract f from WREG with	1	0101 01da	ffff ffff	C, DC, Z, OV, N	
		Borrow	1				
SUBWF	f, d, a	Subtract WREG from f	1	0101 11da	ffff ffff	C, DC, Z, OV, N	1, 2
SUBWFB	f, d, a	Subtract WREG from f with	1	0101 10da	ffff ffff	C, DC, Z, OV, N	
		Borrow	1				
SWAPF	f, d, a	Swap Nibbles in f	1	0011 10da	ffff ffff	None	4
TSTFSZ	f, a	Test f, Skip if 0	1	0110 011a	ffff ffff	None	1, 2
XORWF	f, d, a	Exclusive OR WREG with f	1	0001 10da	ffff ffff	Z, N	
			1				
			1				
			1				
			1 (2 or 3)				
			1				

Note 1: When a PORT register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned.

3: If the Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

4: Some instructions are two-word instructions. The second word of these instructions will be executed as a NOP unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

TBLRD **Table Read**

Syntax: TBLRD (*; *+; *-; +*)

Operands: None

Operation: if TBLRD *,
(Prog Mem (TBLPTR)) → TABLAT;
TBLPTR – No Change
if TBLRD *+,
(Prog Mem (TBLPTR)) → TABLAT;
(TBLPTR) + 1 → TBLPTR
if TBLRD *-,
(Prog Mem (TBLPTR)) → TABLAT;
(TBLPTR) – 1 → TBLPTR
if TBLRD +*,
(TBLPTR) + 1 → TBLPTR;
(Prog Mem (TBLPTR)) → TABLAT

Status Affected: None

Encoding:

0000	0000	0000	10nn nn=0 * =1 *+ =2 *- =3 +*

Description: This instruction is used to read the contents of Program Memory (P.M.). To address the program memory, a pointer called Table Pointer (TBLPTR) is used.

The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-Mbyte address range.

TBLPTR<0> = 0: Least Significant Byte of Program Memory Word

TBLPTR<0> = 1: Most Significant Byte of Program Memory Word

The TBLRD instruction can modify the value of TBLPTR as follows:

- no change
- post-increment
- post-decrement
- pre-increment

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation
No operation	No operation (Read Program Memory)	No operation	No operation (Write TABLAT)

TBLRD **Table Read (Continued)**

Example 1:
```
TBLRD   *+ ;
```
Before Instruction
TABLAT = 55h
TBLPTR = 00A356h
MEMORY(00A356h) = 34h
After Instruction
TABLAT = 34h
TBLPTR = 00A357h

Example 2:
```
TBLRD   +* ;
```
Before Instruction
TABLAT = AAh
TBLPTR = 01A357h
MEMORY(01A357h) = 12h
MEMORY(01A358h) = 34h
After Instruction
TABLAT = 34h
TBLPTR = 01A358h

27.2.3 BYTE-ORIENTED AND BIT-ORIENTED INSTRUCTIONS IN INDEXED LITERAL OFFSET MODE

Note: Enabling the PIC18 instruction set extension may cause legacy applications to behave erratically or fail entirely.

In addition to eight new commands in the extended set, enabling the extended instruction set also enables Indexed Literal Offset Addressing (**Section 6.5.1 "Indexed Addressing with Literal Offset"**). This has a significant impact on the way that many commands of the standard PIC18 instruction set are interpreted.

When the extended set is disabled, addresses embedded in opcodes are treated as literal memory locations: either as a location in the Access Bank (a = 0) or in a GPR bank designated by the BSR (a = 1). When the extended instruction set is enabled and a = 0, however, a file register argument of 5Fh or less is interpreted as an offset from the pointer value in FSR2 and not as a literal address. For practical purposes, this means that all instructions that use the Access RAM bit as an argument – that is, all byte-oriented and bit-oriented instructions, or almost half of the core PIC18 instructions – may behave differently when the extended instruction set is enabled.

When the content of FSR2 is 00h, the boundaries of the Access RAM are essentially remapped to their original values. This may be useful in creating backward-compatible code. If this technique is used, it may be necessary to save the value of FSR2 and restore it when moving back and forth between C and assembly routines in order to preserve the Stack Pointer. Users must also keep in mind the syntax requirements of the extended instruction set (see **Section 27.2.3.1 "Extended Instruction Syntax with Standard PIC18 Commands"**).

Although the Indexed Literal Offset mode can be very useful for dynamic stack and pointer manipulation, it can also be very annoying if a simple arithmetic operation is carried out on the wrong register. Users who are accustomed to the PIC18 programming must keep in mind that, when the extended instruction set is enabled, register addresses of 5Fh or less are used for Indexed Literal Offset Addressing.

Representative examples of typical byte-oriented and bit-oriented instructions in the Indexed Literal Offset mode are provided on the following page to show how execution is affected. The operand conditions shown in the examples are applicable to all instructions of these types.

27.2.3.1 Extended Instruction Syntax with Standard PIC18 Commands

When the extended instruction set is enabled, the file register argument 'f' in the standard byte-oriented and bit-oriented commands is replaced with the literal offset value 'k'. As already noted, this occurs only when 'f' is less than or equal to 5Fh. When an offset value is used, it must be indicated by square brackets ("[ ]"). As with the extended instructions, the use of brackets indicates to the compiler that the value is to be interpreted as an index or an offset. Omitting the brackets, or using a value greater than 5Fh within the brackets, will generate an error in the MPASM™ Assembler.

If the index argument is properly bracketed for Indexed Literal Offset Addressing, the Access RAM argument is never specified; it will automatically be assumed to be '0'. This is in contrast to standard operation (extended instruction set disabled), when 'a' is set on the basis of the target address. Declaring the Access RAM bit in this mode will also generate an error in the MPASM Assembler.

The destination argument 'd' functions as before.

In the latest versions of the MPASM Assembler, language support for the extended instruction set must be explicitly invoked. This is done with either the command line option, /y, or the PE directive in the source listing.

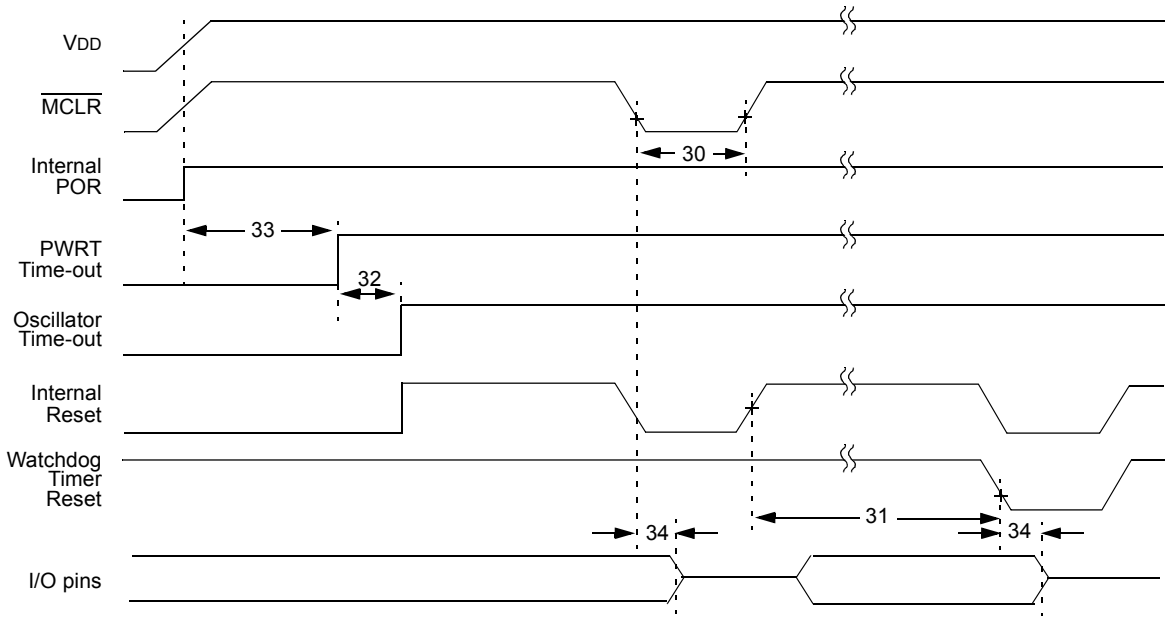
27.2.4 CONSIDERATIONS WHEN ENABLING THE EXTENDED INSTRUCTION SET

It is important to note that the extensions to the instruction set may not be beneficial to all users. In particular, users who are not writing code that uses a software stack may not benefit from using the extensions to the instruction set.

Additionally, the Indexed Literal Offset Addressing mode may create issues with legacy applications written to the PIC18 assembler. This is because instructions in the legacy code may attempt to address registers in the Access Bank below 5Fh. Since these addresses are interpreted as literal offsets to FSR2 when the instruction set extension is enabled, the application may read or write to the wrong data addresses.

When porting an application to the PIC18F87J72 family, it is very important to consider the type of code. A large, re-entrant application that is written in C and would benefit from efficient compilation will do well when using the instruction set extensions. Legacy applications that heavily use the Access Bank will most likely not benefit from using the extended instruction set.

FIGURE 29-6: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING



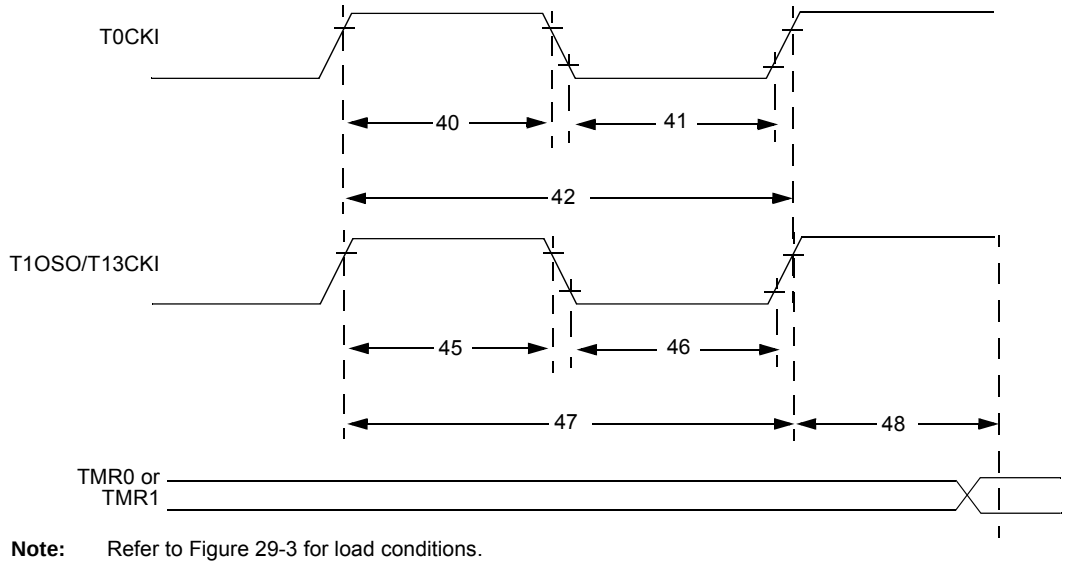
Note: Refer to Figure 29-3 for load conditions.

TABLE 29-2: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER, POWER-UP TIMER AND BROWN-OUT RESET REQUIREMENTS

Param. No.	Symbol	Characteristic	Min.	Typ.	Max.	Units	Conditions
30	TMCL	MCLR Pulse Width (low)	2 TCY	10 TCY	—		**(Note 1)**
31	TWDT	Watchdog Timer Time-out Period (no postscaler)	3.4	4.0	4.6	ms	
32	TOST	Oscillation Start-up Timer Period	1024 TOSC	—	1024 TOSC		TOSC = OSC1 period
33	TPWRT	Power-up Timer Period	45.8	65.5	85.2	ms	
34	TIOZ	I/O High-Impedance from MCLR Low or Watchdog Timer Reset	—	2	—	µs	
38	TCSD	CPU Start-up Time	—	10	—	µs	
				200		µs	Voltage Regulator enabled and put to sleep
39	TIOBST	Time for INTOSC to Stabilize	—	1	—	µs	

Note 1: To ensure device Reset, MCLR must be low for at least 2 TCY or 400 µs, whichever is lower.

FIGURE 29-7: TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS



Note: Refer to Figure 29-3 for load conditions.

TABLE 29-3: TIMER0 AND TIMER1 EXTERNAL CLOCK REQUIREMENTS

Param. No.	Symbol	Characteristic		Min.	Max.	Units	Conditions
40	TT0H	T0CKI High Pulse Width	No prescaler	0.5 TCY + 20	—	ns	
			With prescaler	10	—	ns	
41	TT0L	T0CKI Low Pulse Width	No prescaler	0.5 TCY + 20	—	ns	
			With prescaler	10	—	ns	
42	TT0P	T0CKI Period	No prescaler	TCY + 10	—	ns	
			With prescaler	Greater of: 20 ns or (TCY + 40)/N	—	ns	N = prescale value (1, 2, 4,..., 256)
45	TT1H	T13CKI High Time	Synchronous, no prescaler	0.5 TCY + 20	—	ns	
			Synchronous, with prescaler	10	—	ns	
			Asynchronous	30	—	ns	
46	TT1L	T13CKI Low Time	Synchronous, no prescaler	0.5 TCY + 5	—	ns	
			Synchronous, with prescaler	10	—	ns	
			Asynchronous	30	—	ns	
47	TT1P	T13CKI Input Period	Synchronous	Greater of: 20 ns or (TCY + 40)/N	—	ns	N = prescale value (1, 2, 4, 8)
			Asynchronous	60	—	ns	
	FT1	T13CKI Oscillator Input Frequency Range		DC	50	kHz	
48	TCKE2TMRI	Delay from External T13CKI Clock Edge to Timer Increment		2 TOSC	7 TOSC	—	

B.3 Terminology and Formulas

This section defines the terms and formulas used throughout this data sheet. The following terms are defined:

- MCLK – Master Clock
- AMCLK – Analog Master Clock
- DMCLK – Digital Master Clock
- DRCLK – Data Rate Clock
- OSR – Oversampling Ratio
- Offset Error
- Gain Error
- Integral Non-Linearity Error
- Signal-To-Noise Ratio (SNR)
- Signal-To-Noise Ratio And Distortion (SINAD)
- Total Harmonic Distortion (THD)
- Spurious-Free Dynamic Range (SFDR)
- Idle Tones
- Dithering
- Crosstalk
- PSRR
- CMRR
- ADC Reset Mode
- Hard Reset Mode (ARESET = 0)
- ADC Shutdown Mode
- Full Shutdown Mode

B.3.1 MCLK – MASTER CLOCK

This is the fastest clock present in the device. This is the frequency of the clock input at the CLKIA.

B.3.2 AMCLK – ANALOG MASTER CLOCK

This is the clock frequency that is present on the analog portion of the device, after prescaling has occurred via the CONFIG1 PRESCALE<1:0> register bits. The analog portion includes the PGAs and the two sigma-delta modulators.

EQUATION B-1: ANALOG MASTER CLOCK

$$AMCLK = \frac{MCLK}{PRESCALE}$$

TABLE B-1: OVERSAMPLING RATIO SETTINGS

PRESCALE (CONFIG1<15:14>)		Analog Master Clock Prescale
0	0	AMCLK = MCLK/1 (default)
0	1	AMCLK = MCLK/2
1	0	AMCLK = MCLK/4
1	1	AMCLK = MCLK/8

B.3.3 DMCLK – DIGITAL MASTER CLOCK

This is the clock frequency that is present on the digital portion of the device, after prescaling and division by 4. This is also the sampling frequency, that is the rate at which the modulator outputs are refreshed. Each period of this clock corresponds to one sample and one modulator output.

EQUATION B-2: DIGITAL MASTER CLOCK

$$DMCLK = \frac{AMCLK}{4} = \frac{MCLK}{4 \times PRESCALE}$$

B.3.4 DRCLK – DATA RATE CLOCK

This is the output data rate (i.e., the rate at which the ADCs output new data). Each new data is signaled by a data ready pulse on the $\overline{DR}$ pin.

This data rate is depending on the OSR and the prescaler with the following formula:

EQUATION B-3: DATA RATE CLOCK

$$DRCLK = \frac{DMCLK}{OSR} = \frac{AMCLK}{4 \times OSR} = \frac{MCLK}{4 \times OSR \times PRESCALE}$$

Since this is the output data rate, and since the decimation filter is a SINC (or notch) filter, there is a notch in the filter transfer function at each integer multiple of this rate.

Table B-2 describes the various combinations of OSR and PRESCALE and their associated AMCLK, DMCLK and DRCLK rates.

B.3.5 OSR – OVERSAMPLING RATIO

The ratio of the sampling frequency to the output data rate, OSR = DMCLK/DRCLK. The default OSR is 64, or with MCLK = 4 MHz, PRESCALE = 1, AMCLK = 4 MHz, f_S = 1 MHz, f_D = 15.625 ksps. The following bits in the CONFIG1 register are used to change the oversampling ratio (OSR).

TABLE B-2: OVERSAMPLING RATIO SETTINGS

CONFIG OSR<1:0>		OVERSAMPLING RATIO (OSR)
0	0	32
0	1	64 (default)
1	0	128
1	1	256

B.3.6 OFFSET ERROR

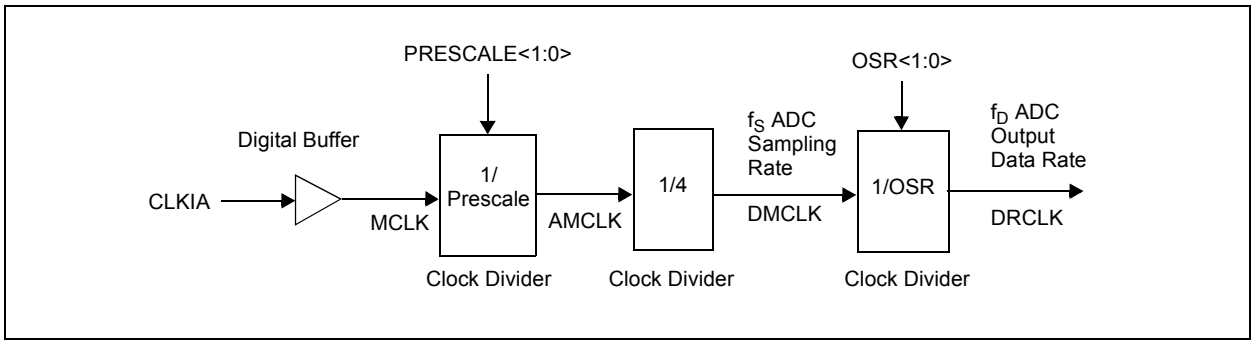
This is the error induced by the ADC when the inputs are shorted together (VIN = 0V). The specification incorporates both PGA and ADC offset contributions.

B.4.10 INTERNAL AFE CLOCK

The AFE uses an external clock signal to operate its internal digital logic. An internal clock generation chain (Figure B-5) is used to produce a range of DRCLK sampling frequencies.

For keeping specified ADC accuracy, AMCLK should be kept between 1 and 5 MHz with BOOST off, or 1 and 8.192 MHz with BOOST on. Larger MCLK frequencies can be used provided the prescaler clock settings allow the AMCLK to respect these ranges.

FIGURE B-5: AFE INTERNAL CLOCK DETAIL



B.5 Serial Interface Description

B.5.1 OVERVIEW

The AFE is accessed for control and data output exclusively through its dedicated Serial Peripheral Interface (SPI). The interface is compatible with SPI Modes 0,0 and 1,1. Data is clocked *out* of the AFE on the *falling* edge of SCK, and data is clocked *in* on the *rising* edge of SCK. In these modes, SCK can Idle either high or low.

Each SPI communication starts with a $\overline{CS}$ falling edge and stops with the $\overline{CS}$ rising edge. Each SPI communication is independent. When $\overline{CS}$ is high, SDO is in high-impedance, transitions on SCK and SDI have no effect. Additional controls pins ($\overline{ARESET}$ and $\overline{DR}$) are also provided on separate pins for advanced communication.

The AFE's SPI interface has a simple command structure. The first byte transmitted is always the control byte and is followed by data bytes that are 8-bit wide. Both ADCs are continuously converting data by default and can be reset or shut down through a CONFIG2 register setting.

Since each ADC data is either 16 or 24 bits (depending on the WIDTH bits), the internal registers can be grouped together with various configurations (through the READ bits) in order to allow easy data retrieval within only one communication. For device reads, the internal address counter can be automatically incremented in order to loop through groups of data within the register map. SDOA will then output the data located at the ADDRESS (A<4:0>) defined in the control byte and then ADDRESS + 1 depending on the READ<1:0> bits, which select the groups of registers. These groups are defined in **Section B.6.1 "ADC Channel Data Output Registers"** (Register Map).

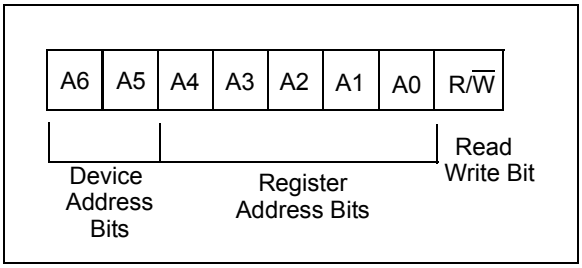
The Data Ready pin ($\overline{DR}$) can be used as an interrupt for a microcontroller and outputs pulses when new ADC channel data is available. The $\overline{ARESET}$ pin acts like a Hard Reset and can reset the AFE to its default power-up configuration, independent of the microcontroller.

B.5.2 CONTROL BYTE

The control byte of the AFE contains two device address bits (A<6:5>), five register address bits (A<4:0>) and a read/write bit (R/$\overline{W}$). The first byte transmitted to the AFE is always the control byte.

The AFE interface is device-addressable (through A<6:5>) so that multiple devices can be present on the same SPI bus with no data bus contention. This functionality enables three-phase power metering systems containing an AFE and two other external AFE-type chips, controlled by a single SPI bus (single $\overline{CS}$, SCK, SDI and SDO pins). The default device address bits are '00'.

FIGURE B-6: CONTROL BYTE



A read on undefined addresses will give an all zeros output on the first and all subsequent transmitted bytes. A write on an undefined address will have no effect and will not increment the address counter either.

The register map is defined in **Section B.6.1 "ADC Channel Data Output Registers"**.