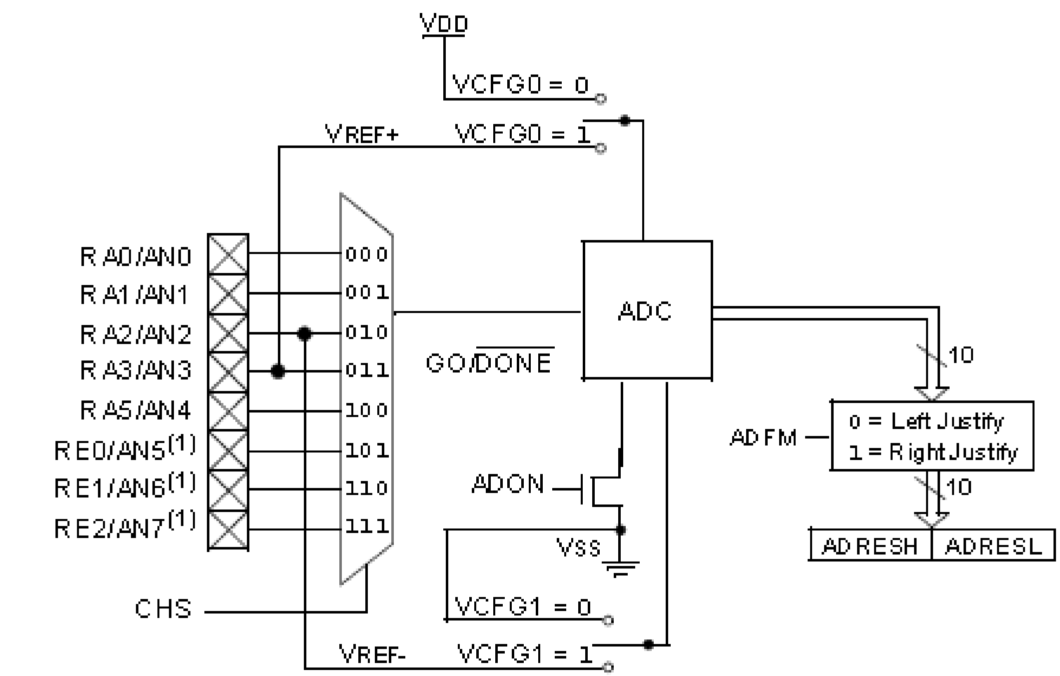
Analog to Digital Converter

EET 250

Block Diagram

FIGURE 12-1: ADC BLOCK DIAGRAM



Note 1: These channels are only available on PIC16F914/917 and PIC16F946 devices.

Description

- The Analog-to-Digital Converter (ADC) allows conversion of an analog input signal to a 10-bit binary representation of that signal. This device uses analog
  - Inputs, which are multiplexed into a single sample and hold circuit.
  - The output of the sample and hold is connected to the input of the converter.
  - The converter generates a 10-bit binary result via successive approximation and stores the conversion result into the ADC result registers (ADRESL and ADRESH).
  - The ADC voltage reference is software selectable to be either internally generated or externally supplied.
  - The ADC can generate an interrupt upon completion of a conversion.
- Interrupts can be used to wake-up the device from Sleep.

ADC Configuration

- When configuring and using the ADC the following functions must be considered:
- Port configuration
- Channel selection
- ADC voltage reference selection
- ADC conversion clock source
- Interrupt control
- Results formatting

ADC Configuration

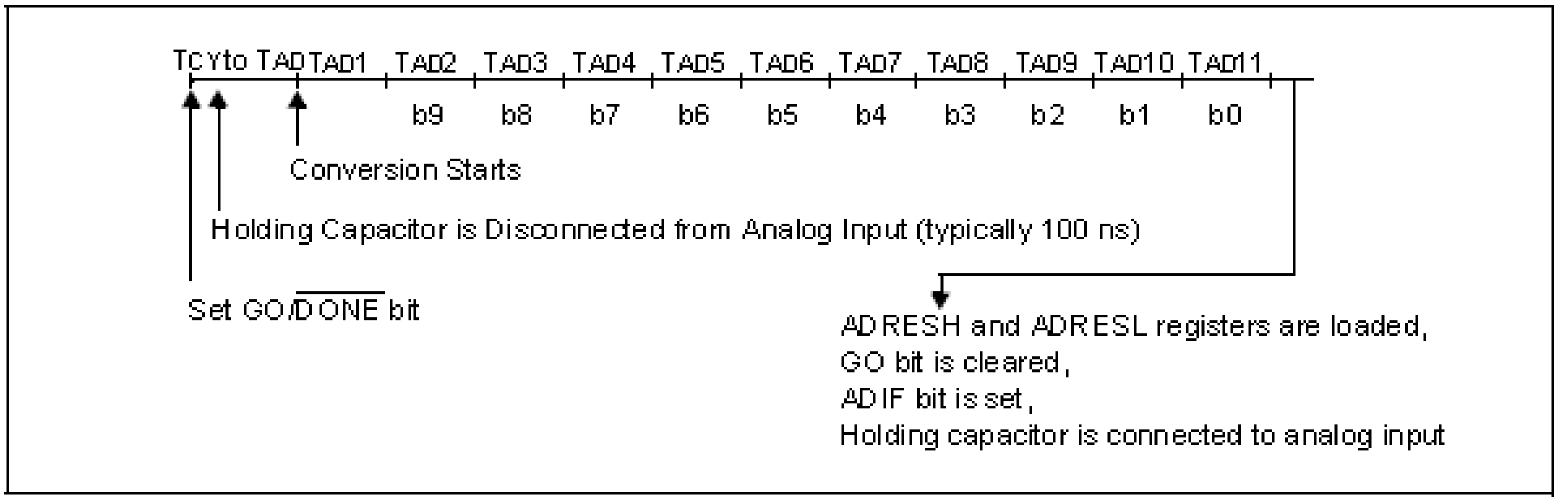
- Port Configuration
  - The ADC can be used to convert both analog and digital signals.
  - When converting analog signals, the I/O pin should be configured for analog by setting the associated TRIS and ANSEL bits.
- Channel selection
  - The CHS bits of the ADCON0 register determine which channel is connected to the sample and hold circuit.
  - When changing channels, a delay is required before starting the next conversion.

ADC Configuration

- ADC VOLTAGE REFERENCE
  - The VCFG bits of the ADCON0 register provide independent control of the positive and negative voltage references.
    - The positive voltage reference can be either VDD or an external voltage source.
    - Likewise, the negative voltage reference can be either VSS or an external voltage source.
- CONVERSION CLOCK
  - The source of the conversion clock is software selectable via the ADCS bits of the ADCON1 register. There are seven possible clock options:
    - FOSC/2
    - FOSC/4
    - FOSC/8
    - FOSC/16
    - FOSC/32
    - FOSC/64
    - FRC (dedicated internal oscillator)
  - The time to complete one bit conversion is defined as TAD. One full 10-bit conversion requires 11 TAD periods

Conversion Process

FIGURE 12-2: ANALOG-TO-DIGITAL CONVERSION TAD CYCLES



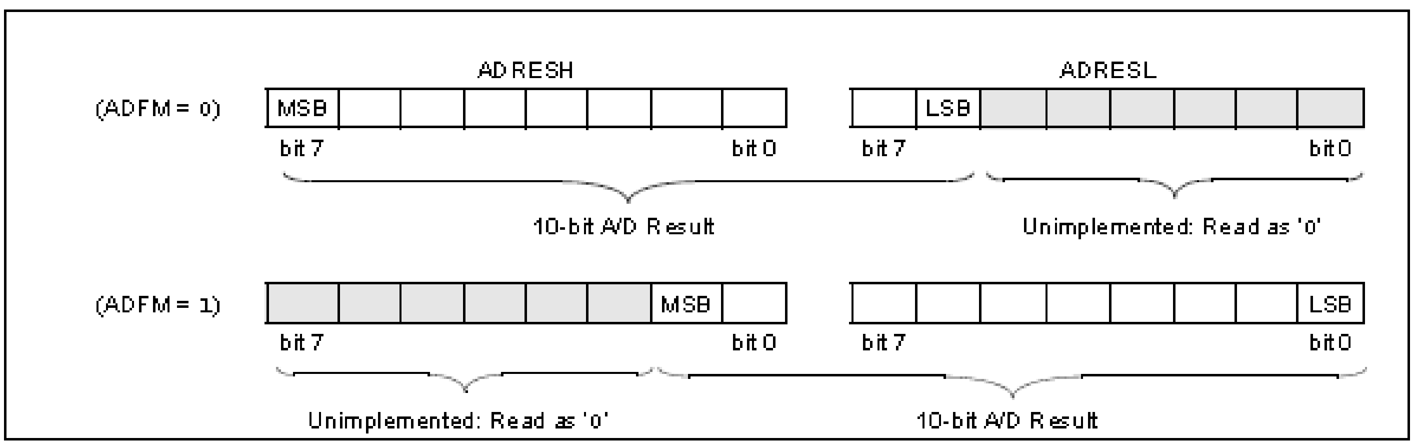
Interrupts

- The ADC module allows for the ability to generate an interrupt upon completion of an Analog-to-Digital conversion.
  - The ADC interrupt flag is the ADIF bit in the PIR1 register.
  - The ADC interrupt enable is the ADIE bit
  - in the PIE1 register.
  - The ADIF bit must be cleared in software

Result Formatting

- The 10-bit A/D conversion result can be supplied in two formats, left justified or right justified.
- The ADFM bit of the ADCON0 register controls the output format.

FIGURE 12-3: 10-BIT A/D CONVERSION RESULT FORMAT



ADC Operation

- STARTING A CONVERSION
  - To enable the ADC module, the ADON bit of the ADCON0 register must be set to a ‘1’.
  - Setting the GO/DONE bit of the ADCON0 register to a ‘1’ will start the Analog-to-Digital conversion.
- COMPLETION OF A CONVERSION
- When the conversion is complete, the ADC module will:
  - Clear the GO/DONE bit
  - Set the ADIF flag bit
  - Update the ADRESH:ADRESL registers with new conversion result

A/D Conversion Procedure

- This is an example procedure for using the ADC to perform an Analog-to-Digital conversion:

  1. Configure Port:
     - Disable pin output driver (See TRIS register)
     - Configure pin as analog
  2. Configure the ADC module:
     - Select ADC conversion clock
     - Configure voltage reference
     - Select ADC input channel
     - Select result format
     - Turn on ADC module
  3. Configure ADC interrupt (optional):
     - Clear ADC interrupt flag
     - Enable ADC interrupt
     - Enable peripheral interrupt
     - Enable global interrupt**(1)**
  4. Wait the required acquisition time
  5. Start conversion by setting the GO/DONE bit.
  6. Wait for ADC conversion to complete by one of the following:
     - Polling the GO/DONE bit
     - Waiting for the ADC interrupt (interrupts enabled)
  7. Read ADC Result
  8. Clear the ADC interrupt flag (required if interrupt is enabled).

Code Snippet

EXAMPLE 12-1: A/D CONVERSION

```
;This code block configures the ADC
;for polling, Vdd reference, Frc clock
;and AN0 input.
;
;Conversion start & polling for completion
; are included.
;
BANKSEL  ADCON1       ;
MOVLW    B'01110000'  ;ADC Frc clock
MOVWF    ADCON1       ;
BANKSEL  TRISA        ;
BSF      TRISA,0      ;Set RA0 to input
BANKSEL  ANSEL        ;
BSF      ANSEL,0      ;Set RA0 to analog
BANKSEL  ADCON0       ;
MOVLW    B'10000001'  ;Right justify,
MOVWF    ADCON0       ;Vdd Vref, AN0, On
CALL     SampleTime   ;Acquisiton delay
BSF      ADCON0,GO    ;Start conversion
BTFSC    ADCON0,GO    ;Is conversion done?
GOTO     $-1          ;No, test again
BANKSEL  ADRESH       ;
MOVF     ADRESH,W     ;Read upper 2 bits
MOVWF    RESULTHI     ;store in GPR space
BANKSEL  ADRESL       ;
MOVF     ADRESL,W     ;Read lower 8 bits
MOVWF    RESULTLO     ;Store in GPR space
```

REGISTER 12-1: ADCON0: A/D CONTROL REGISTER 0

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	VCFG1	VCFG0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON
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 **ADFM:** A/D Conversion Result Format Select bit
1 = Right justified
0 = Left justified

bit 6 **VCFG1:** Voltage Reference bit
1 = V_{REF-} pin
0 = V_{SS}

bit 5 **VCFG0:** Voltage Reference bit
1 = V_{REF+} pin
0 = V_{SS}

bit 4-2 **CHS<2:0>:** Analog Channel Select bits
000 = AN0
001 = AN1
010 = AN2
011 = AN3
100 = AN4
101 = AN5(1)
110 = AN6(1)
111 = AN7(1)

bit 1 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit
1 = A/D conversion cycle in progress. Setting this bit starts an A/D conversion cycle.
This bit is automatically cleared by hardware when the A/D conversion has completed.
0 = A/D conversion completed/not in progress

bit 0 **ADON:** ADC Enable bit
1 = ADC is enabled
0 = ADC is disabled and consumes no operating current

REGISTER 12-2: ADCON1: A/D CONTROL REGISTER 1

U-0	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0
—	ADCS2	ADCS1	ADCS0	—	—	—	—
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 **Unimplemented:** Read as '0'

bit 6-4 **ADCS<2:0>:** A/D Conversion Clock Select bits

000 = $F_{OSC}/2$
001 = $F_{OSC}/8$
010 = $F_{OSC}/32$
x11 = F_{RC} (clock derived from a dedicated internal oscillator = 500 kHz max.)
100 = $F_{OSC}/4$
101 = $F_{OSC}/16$
110 = $F_{OSC}/64$

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

REGISTER 3-1: ANSEL: ANALOG SELECT 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
ANS7(2)	ANS6(2)	ANS5(2)	ANS4	ANS3	ANS2	ANS1	ANS0
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 7-0 **ANS<7:0>**: Analog Select bits
Analog select between analog or digital function on pins AN<7:0>, respectively.
1 = Analog input. Pin is assigned as analog input(1).
0 = Digital I/O. Pin is assigned to port or special function.

Note 1: Setting a pin to an analog input automatically disables the digital input circuitry, weak pull-ups, and interrupt-on-change if available. The corresponding TRIS bit must be set to Input mode in order to allow external control of the voltage on the pin.

2: PIC16F914/PIC16F917/PIC16F946 only.

REGISTER 12-3: ADRESH: ADC RESULT REGISTER HIGH (ADRESH) ADFM = 0

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
ADRES9	ADRES8	ADRES7	ADRES6	ADRES5	ADRES4	ADRES3	ADRES2
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 7-0 **ADRES<9:2>**: ADC Result Register bits
Upper 8 bits of 10-bit conversion result

REGISTER 12-4: ADRESL: ADC RESULT REGISTER LOW (ADRESL) ADFM = 0

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
ADRES1	ADRES0	—	—	—	—	—	—
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 7-6 **ADRES<1:0>**: ADC Result Register bits
Lower 2 bits of 10-bit conversion result

bit 5-0 **Reserved**: Do not use.

REGISTER 12-5: ADRESH: ADC RESULT REGISTER HIGH (ADRESH) ADFM = 1

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	—	—	—	ADRES9	ADRES8
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 7-2 **Reserved**: Do not use.

bit 1-0 **ADRES<9:8>**: ADC Result Register bits
Upper 2 bits of 10-bit conversion result

REGISTER 12-6: ADRESL: ADC RESULT REGISTER LOW (ADRESL) ADFM = 1

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
ADRES7	ADRES6	ADRES5	ADRES4	ADRES3	ADRES2	ADRES1	ADRES0
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 7-0 **ADRES<7:0>**: ADC Result Register bits
Lower 8 bits of 10-bit conversion result

FIGURE 12-5: ADC TRANSFER FUNCTION

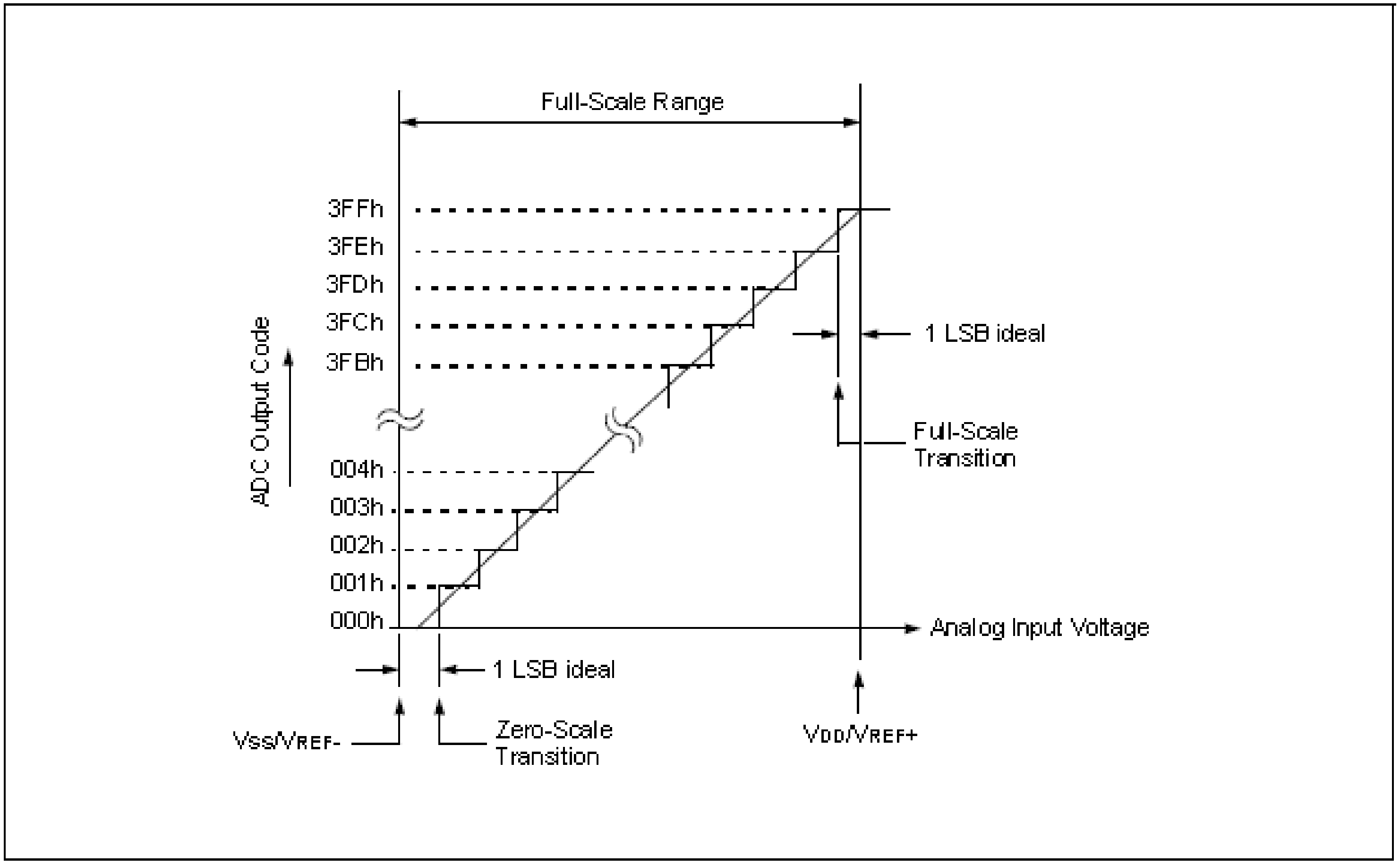


TABLE 12-2: SUMMARY OF ASSOCIATED ADC REGISTERS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
ADCON0	ADFM	VCFG1	VCFG0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON	0000 0000	0000 0000
ADCON1	—	ADCS2	ADCS1	ADCS0	—	—	—	—	-000 ----	-000 ----
ANSEL	ANS7	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0	1111 1111	1111 1111
ADRESH	A/D Result Register High Byte								xxxx xxxx	uuuu uuuu
ADRESL	A/D Result Register Low Byte								xxxx xxxx	uuuu uuuu
INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000x
LCDCON	LCDEN	$\overline{\text{SLPEN}}$	WERR	VLCDEN	CS1	CS0	LMUX1	LMUX0	0001 0011	0001 0011
LCDSE0	SE7	SE6	SE5	SE4	SE3	SE2	SE1	SE0	0000 0000	0000 0000
LCDSE1	SE15	SE14	SE13	SE12	SE11	SE10	SE9	SE8	0000 0000	0000 0000
LCDSE2(1)	SE23	SE22	SE21	SE20	SE19	SE18	SE17	SE16	0000 0000	0000 0000
PIE1	EEIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
PIR1	EEIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0	xxxx xxxx	uuuu uuuu
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
PORTE	RE7	RE6	RE5	RE4	RE3	RE2	RE1	RE0	xxxx xxxx	uuuu uuuu
TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	1111 1111	1111 1111
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 ----	1111 ----
TRISE	TRISE7	TRISE6	TRISE5	TRISE4	TRISE3	TRISE2	TRISE1	TRISE0	1111 1111	1111 1111

Legend: x = unknown, u = unchanged, — = unimplemented read as '0'. Shaded cells are not used for ADC module.

ADC Lab

EET 250

Objective

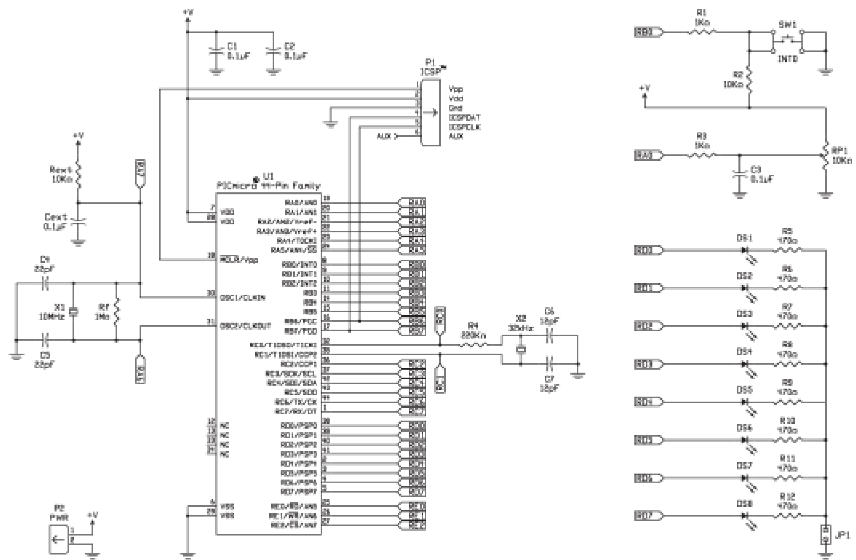
- Become familiar with ADC operation within PIC16917
  - Digitize number of different inputs
  - On board pot, Thermometer Chip, Photocell
  - Verify transfer function of converter using Pot
  - Build a light switch example
- Note ADC function cannot be simulated—all debug with occur with PICKIT2

To Do

- Open ADC Folder and launch ADC.MCP
- Examine ADC Code using Text Editor
- Explain settings in TRISA, TRISD and ANSEL
  - What is PORTA RA0 configured as?
  - What is PORTD configured as?
- Examine ADCON1 and ADCON0
  - What is Output format for ADRESH and ADRESL?
  - What is the conversion clock setting?
  - What is Channel selection?
  - What is the Voltage reference?
- Build code
- Select Debugger -> PICKIT2
- Program , and RUN
- Vary pot and note change of LED with pot settings
- What do the pot setting represent?

Experiment 1 Hardware
44 pin demo board

- Pot wiper is wired to RA0
- Ends of pot are wired form +5V to GND
- Place voltage meter to measure voltage to pin RA0 to ground
- Vary pot setting from fully Counter clock wise position to fully clock wise position making sure to achieve voltages required by table (next slide)
- For each voltage also record LED output
- Plot LED output versus voltage to verify transfer function



Experiment 1 Table

Voltage at RA0	LED READING	
+5v		
+4.5V		
4.25V		
4.00 V		
3.75 V		
3.50 V		
3.25V		
3.00 V		
2.75V		
2.50V		
2.25 V		
2.00 V		
1.75V		
1.50V		
1.25V		
1.00V		
.75V		
.50V		
.25V		
0V		

Plot results using
10 squares per
inch graph paper
Led reading
versus voltage
Verify that plot
matches
published
transfer function

Experiment 2 Thermometer

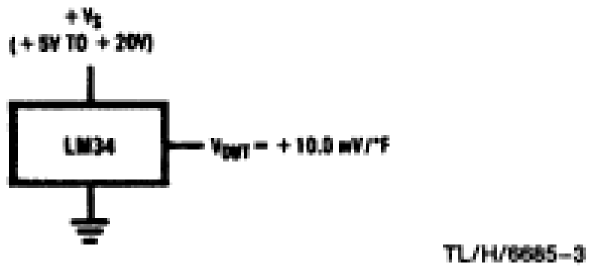


FIGURE 1. Basic Fahrenheit Temperature Sensor (+5° to +300°F)

Features

- Calibrated directly in degrees Fahrenheit
- Linear +10.0 mV/°F scale factor
- 1.0°F accuracy guaranteed (at +77°F)
- Rated for full −50° to +300°F range
- Suitable for remote applications
- Low cost due to wafer-level trimming
- Operates from 5 to 30 volts
- Less than 90 μA current drain
- Low self-heating, 0.18°F in still air
- Nonlinearity only ±0.5°F typical
- Low-impedance output, 0.4Ω for 1 mA load

To DO:

1. **Wire the sensor to the RA1 of the 44 pin DEMO Board**
2. **Use prototype board and three jumper +5V (VDD) , GND, and signal out**
3. **Use adcthermo.mcp lab**
4. **Build, debugger -> PICKIT2, program part and run**
5. **verify operation by examining LED output while touching sensor**

Experiment 2 Thermometer

- Measure room temperature.
- Using the following formula

A/D output each bit represents = +5V/1024 or .00488 volts or approx .005 volts/bit

Thermometer is .010 volts/ degree F

- Let LED reading settle and then Capture LED output 8 bits –this should be room temperature
- Take LED output as binary count (it is multiplied) by 2. This should each binary value for temperature
- Covert to decimal
- What is the reading? Does it make sense?
- Repeat the above process touching the sensor (for body temperature)– What is the reading?. Does it make sense?

Experiment 3 Photocell

- Separate handout will be supplied " Interfacing a Photocell with PIC"
- Follow instructions in that handout
- Photocell is also a linear output device
- Use adcphoto.mcp