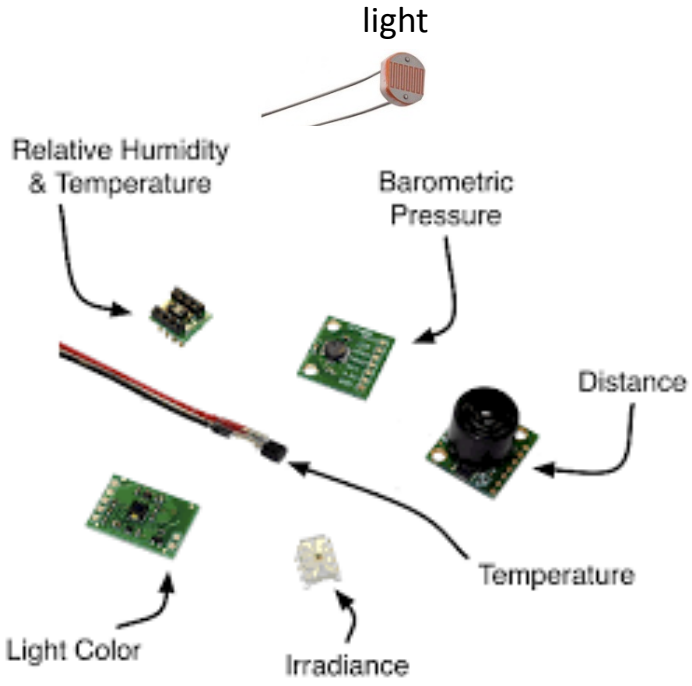
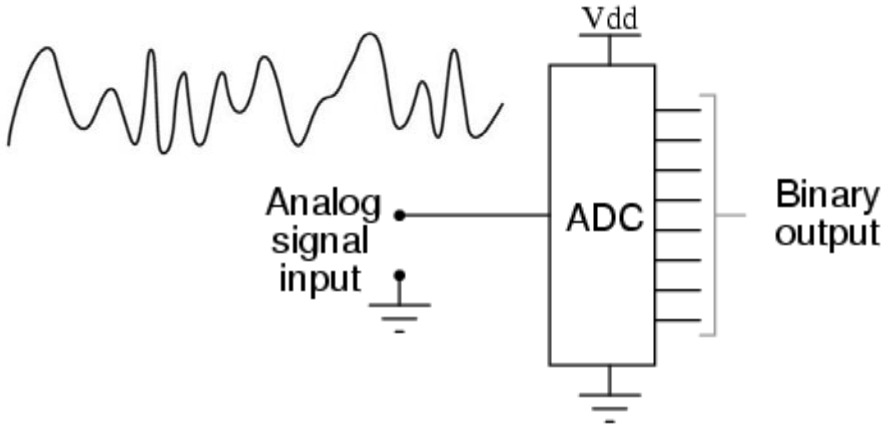
Exercise 3 – Analog/Digital Conversion Prep

Circuit Basics
PIC 24 A/D Conversion
Library ADC Functions
More C Basics

Purpose

- A/D Conversion
  - Real world is analog
  - Analog sensors, analog voltage measurements





Exercise Overview

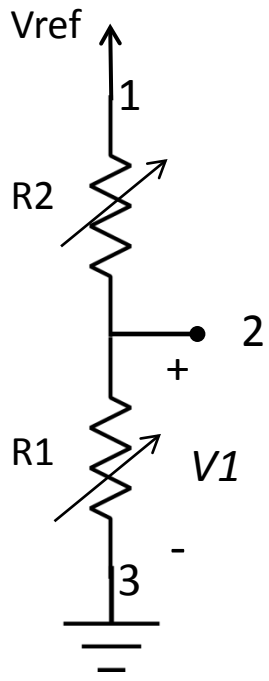
- Set up voltage divider circuit for analog input
- Write program to convert analog voltage to digital using PIC24 ADC
- Run program and record values
- Write your own function and use in main code.

New Interfacing Concepts

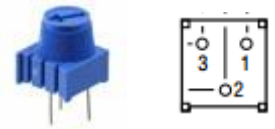
- Using potentiometer to generate analog voltage
- Configuring PIC24 pins to be analog inputs
- PIC24 ADC1 peripheral

Use Ohm's Law to find V1

$$V1 = \left(\frac{R1}{R1+R2}\right)\text{Vref}$$



Voltage Divider Circuit

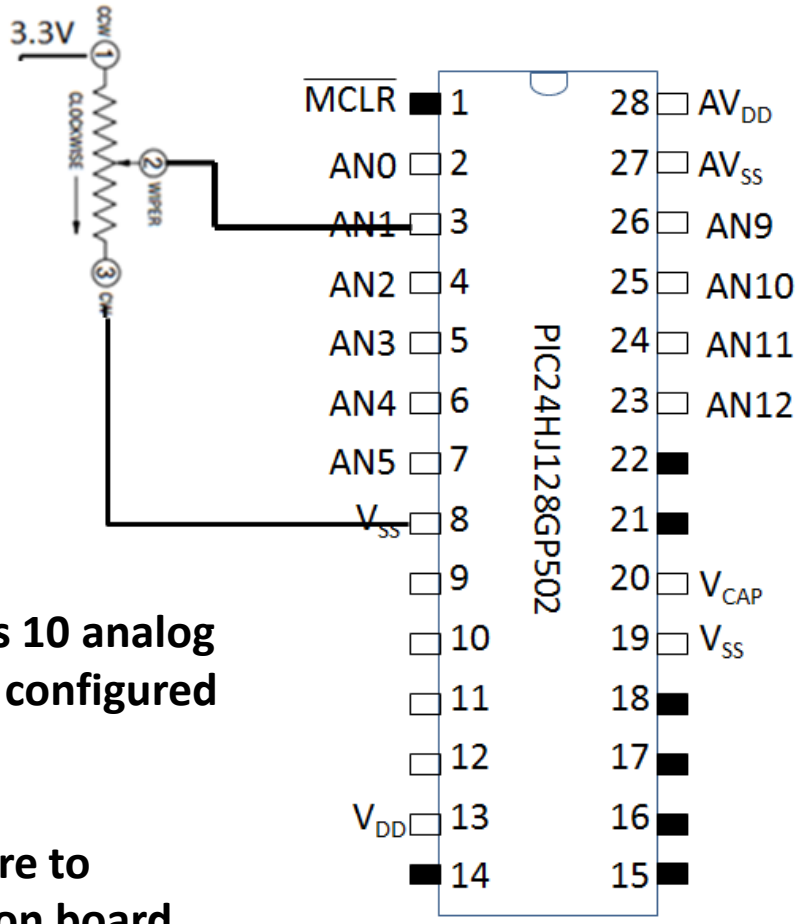


10 KΩ Potentiometer

Clockwise:
R2 increases
R1 decreases
V1?

Counter CW:
R2 decreases
R1 increases
V1?

Hardware Setup

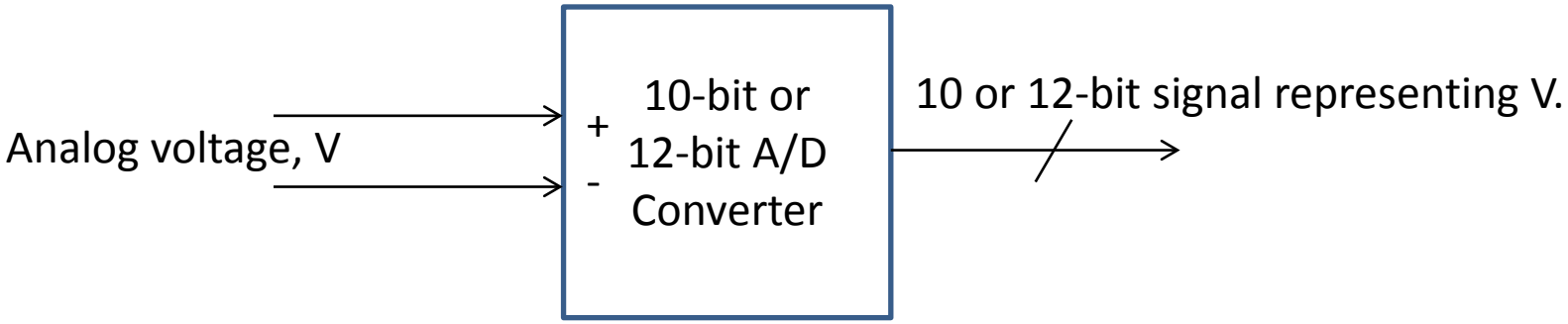


Note that our PIC24 has 10 analog pins, which can also be configured to be digital.

When using AN0, be sure to remove jumper to LED on board.

PIC24 A/D Conversion

Black box view of A/D Converter



Analog range: 0-3.3V

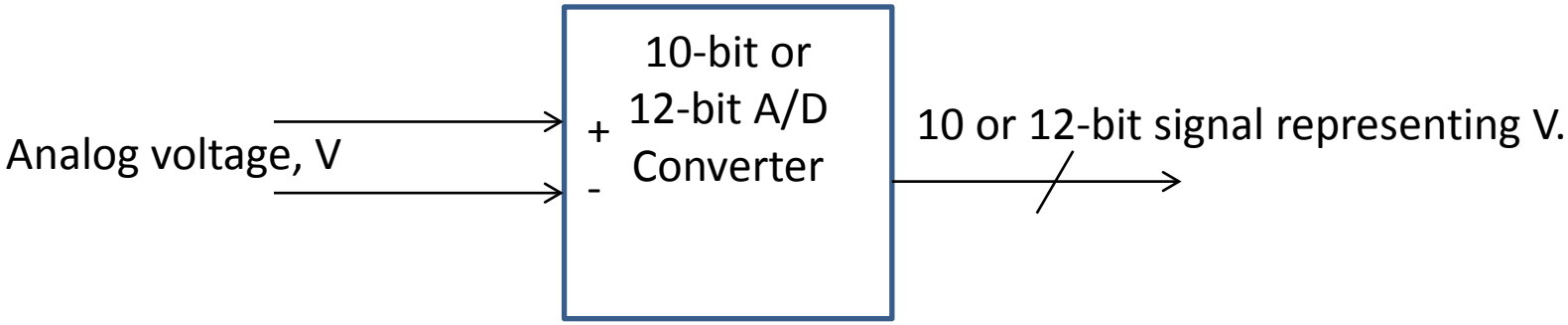
Digital range:

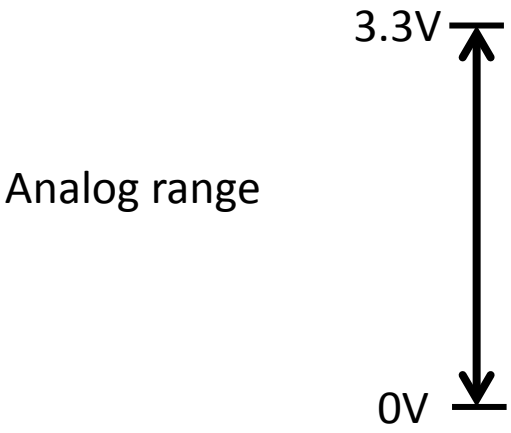
10-bit 0-1023
12-bit 0-4095

Conversion: 10-bit Analog = (Digital * 3.3V)/1024
12-bit Analog = (Digital * 3.3V)/4096

PIC24 A/D Conversion

Black box view of A/D Converter





10-bit Conversion:

$2^{10} = 1024$

1 bit = 3.3V/1024

1 bit = 3.222mV

12-bit Conversion:

$2^{12} = 4096$

1 bit = 3.3V/4096

1 bit = 0.8057mV

"resolution"

PIC24 ADC

- Call ADC1, but there is only ONE ADC on our PIC
- MANY options to select and specify
  - Which of several Analog input channels
  - 10 or 12-bit results
  - Clock input choice
  - Differential voltage input, or single-ended (ref to gnd)
  - Internal or External voltage reference
  - Sequential or simultaneous sampling
  - many others….

Special Function Registers

- Used to set parameters for peripherals.
- ADC Special Function Registers:
  - AD1PCFGL – ADC1 Port ConFiGuration register Low
  - ADC1CON1 – ADC1 Control Register 1
  - ADC1CON2 – ADC1 Control Register 2
  - AD1CHS123 – ADC1 Channel 1, 2, 3 Select Register
  - AD1CHS0 – ADC1 Input Channel 0 Select Register

Step 1 – Setting pins as Analog/Digital

AD1PCFGL – ADC1 Port ConFiGuration register Low

REGISTER 20-8: AD1PCFGL: ADC1 PORT CONFIGURATION REGISTER LOW(1,2,3)

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	PCFG12	PCFG11	PCFG10	PCFG9	PCFG8
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
PCFG7	PCFG6	PCFG5	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0
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 12-0 **PCFG<12:0>:** ADC Port Configuration Control bits

1 = Port pin in Digital mode, port read input enabled, ADC input multiplexer connected to AVSS

0 = Port pin in Analog mode, port read input disabled, ADC samples pin voltage

Note 1: On devices without 13 analog inputs, all PCFG bits are R/W by user. However, PCFG bits are ignored on ports without a corresponding input on device.

2: PCFGx = ANx, where x = 0 through 12.

3: PCFGX bits have no effect if ADC module is disabled by setting ADXMD bit in the PMDX register. In this case, all port pins multiplexed with ANX will be in Digital mode.

PCFGx = 1 – Digital
PCFGx = 0 - Analog

For our chip, only pins 0-5 and 9-12 are available for analog.

To Specify Analog Pins

- Directly using AD1PCFGL register:

  0=analog, 1=digital

  _PCFG0 = 0; // Sets AN0 (pin 2) to analog

- Using textbook library function (see macros defined in pic24_ports_config.h):

  CONFIG_RA0_AS_ANALOG(); // Sets AN0 to analog

To Configure ADC (datasheet)

20.2 ADC Initialization

The following configuration steps should be performed.

1. Configure the ADC module:
   a) Select port pins as analog inputs (AD1PCFGH<15:0> or AD1PCFGL<15:0>)
   b) Select voltage reference source to match expected range on analog inputs (AD1CON2<15:13>)
   c) Select the analog conversion clock to match desired data rate with processor clock (AD1CON3<7:0>)
   d) Determine how many S/H channels are used (AD1CON2<9:8> and AD1PCFGH<15:0> or AD1PCFGL<15:0>)
   e) Select the appropriate sample/conversion sequence (AD1CON1<7:5> and AD1CON3<12:8>)
   f) Select how conversion results are presented in the buffer (AD1CON1<9:8>)
   g) Turn on ADC module (AD1CON1<15>)
2. Configure ADC interrupt (if required):
   a) Clear the AD1IF bit
   b) Select ADC interrupt priority

Textbook library has configuration functions that implement these steps.

New Software Concepts

- Library functions to configure and use ADC
- Using C statements to implement conversion formulas
  - Specifying integer and floating point values
  - Expressions with mixed data types
- Converting algorithm to C function
- Defining and using arrays

Textbook ADC Library Functions

pic24_adc.h

Functions

uint16_t	**convertADC1** (void)
void	**configADC1_ManualCH0** (uint16_t u16_Ch0PositiveMask, **uint8_t** u8_autoSampleTime, **uint8_t** u8_Use12bits)
void	**configADC1_AutoScanIrqCH0** (uint16_t u16_ch0ScanMask, **uint8_t** u8_autoSampleTime, **uint8_t** u8_12bit)
void	**configADC1_AutoHalfScanIrqCH0** (uint16_t u16_ch0ScanMask, **uint8_t** u8_autoSampleTime, **uint8_t** u8_12bit)
void	**configADC1_Simul4ChanIrq** (**uint8_t** u8_ch0Select, uint16_t u16_ch123SelectMask, uint16_t u16_numTcyMask)
static void	**WAIT_UNTIL_CONVERSION_COMPLETE_ADC1** ()

configADC1_ManualCH0(pin, sample_time, 10/12)

```
configADC1_ManualCH0(RA1_AN, 31, 0);
```

RA1 in analog mode

Sample time (31 is max, and safe)

0=10 bit
1=12 bit

Textbook ADC Library Functions

pic24_adc.h

Functions

uint16_t	**convertADC1** (void)
void	**configADC1_ManualCH0** (uint16_t u16_Ch0PositiveMask, **uint8_t** u8_autoSampleTime, **uint8_t** u8_Use12bits)
void	**configADC1_AutoScanIrqCH0** (uint16_t u16_ch0ScanMask, **uint8_t** u8_autoSampleTime, **uint8_t** u8_12bit)
void	**configADC1_AutoHalfScanIrqCH0** (uint16_t u16_ch0ScanMask, **uint8_t** u8_autoSampleTime, **uint8_t** u8_12bit)
void	**configADC1_Simul4ChanIrq** (**uint8_t** u8_ch0Select, uint16_t u16_ch123SelectMask, uint16_t u16_numTcyMask)
static void	**WAIT_UNTIL_CONVERSION_COMPLETE_ADC1** ()

convertADC1(void)

```
u16_var = convertADC1();   //do manual conversion
                           // and put result in
                           // variable
```

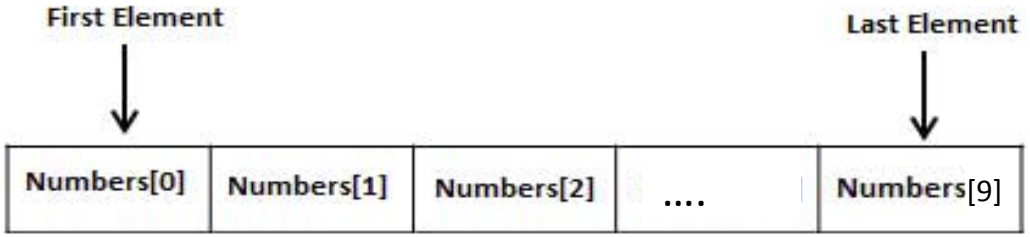
C Array Types - Review

- Indexed collection of elements of the same type.
- Declaring arrays:

  type array_name [array_size];

  char Numbers[10];

  uint8_t Numbers[10];



- Initializing arrays:

  char Numbers[] = {‘0’, ‘1’, 0x32, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’, 0x38, 57};

  // Numbers = “0123456789”

Reference for ADC

- PIC24 Datasheet – Chapter 16 – ADC manual (on Nexus)