



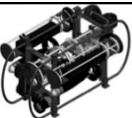
## IMDL Software Series

# Atmel AVR Xmega Code using GPIO from the Atmel ASF for the ADC on DAQ Boards

A. A. Arroyo

University of Florida, EEL-5666  
© Drs. A. A. Arroyo & E. M. Schwartz

1



## IMDL Software Series ADC

The Analog to Digital Converters (ADC) is used to sample and digitize analog signals. These devices are simple to communicate with and are quicker to set up than a USART.

### Typical Applications:

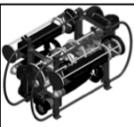
- Audio Input
- Checking battery voltage
- Sensor - i.e., IR range finder

XMEGA A1 devices have two ADCs. These two modules can be operated simultaneously, individually or synchronized.

In the DAQ board they convert a 0 to 5v range into a 12-bit range (0 to 4095), about a ~.025% increment, e.g., 0 = 0v; 0xFF=4095 ≈ 5v; 2047 ≈ 2.5v; 511 ≈ 0.625v, etc.

University of Florida, EEL-5666  
© Drs. A. A. Arroyo & E. M. Schwartz

2



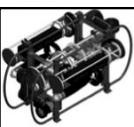
# IMDL Software Series ADC

## 28.1 Features

- 12-bit resolution
- Up to two million samples per second
  - Two inputs can be sampled simultaneously using ADC and 1x gain stage
  - Four inputs can be sampled within 1.5 $\mu$ s
    - Down to 2.5 $\mu$ s conversion time with 8-bit resolution
    - Down to 3.5 $\mu$ s conversion time with 12-bit resolution
- Differential and single-ended input
  - Up to 16 single-ended inputs
  - 16x4 differential inputs without gain
  - 8x4 differential input with gain
- Built-in differential gain stage
  - 1/2x, 1x, 2x, 4x, 8x, 16x, 32x, and 64x gain options
- Single, continuous and scan conversion options
- Four internal inputs
  - Internal temperature sensor
  - DAC output
  - V<sub>CC</sub> voltage divided by 10
  - 1.1V bandgap voltage
- Four conversion channels with individual input control and result registers
  - Enable four parallel configurations and results
- Internal and external reference options
- Compare function for accurate monitoring of user defined thresholds
- Optional event triggered conversion for accurate timing
- Optional DMA transfer of conversion results
- Optional interrupt/event on compare result

University of Florida, EEL-5666  
© Drs. A. A. Arroyo & E. M. Schwartz

3



# IMDL Software Series ADC

Figure 28-1. ADC overview.

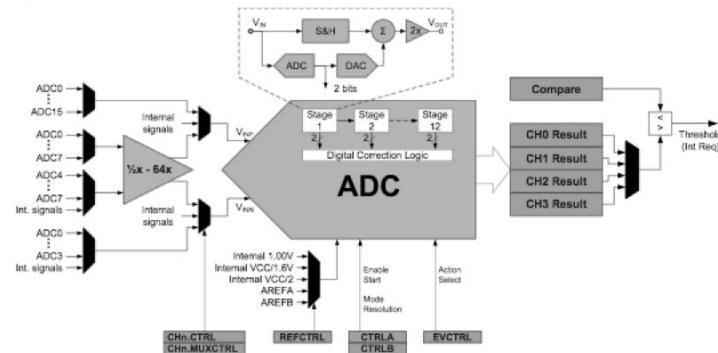
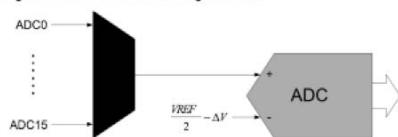
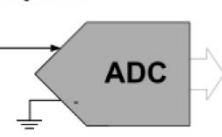


Figure 28-5. Single-ended measurement in unsigned mode.

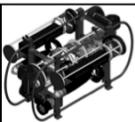


University of Florida, EEL-5666  
© Drs. A. A. Arroyo & E. M. Schwartz

Figure 28-4. Single-ended measurement in signed mode.



4



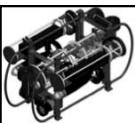
## IMDL Software Series ADC

Before a conversion is started, the desired input source must be selected for one or more ADC channels. An ADC conversion for a ADC channel can either be started by the application software writing to the start conversion bit for the ADC channel, or from any of the events in the Event System. It is possible to write the start conversion bit for several ADC channels at the same time, or to use one event to trigger conversions on several ADC channels at the same time. This makes it possible to scan several or all channels from one event. The scan will start from the lowest channel number.

.

University of Florida, EEL-5666  
© Drs. A. A. Arroyo & E. M. Schwartz

5



## IMDL Software Series ADC

- The ADC is enabled by setting the ADC Enable bit0 in the CTRLA control Register.

Example:

```
ADCA_CTRLA |= 0x01; //Enable ADCA
ADCA_CTRLA |= 0x04; //Start Conversion on ADCA Chn 0
```

CTRLA – Control register A

Bit	7	6	5	4	3	2	1	0
+0x00	DMASEL[1:0]		CHSTART[3:0]			FLUSH		ENABLE
ReadWrite	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- Bit 5:2 – CHSTART[3:0]: Channel Start Single Conversion

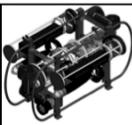
Setting any of these bits will start a conversion on the corresponding ADC channel. Setting several bits at the same time will start conversions on all selected ADC channels, starting with the channel with the lowest number. These bits are cleared by hardware when the conversion has started.

- BIT 0 – ENABLE: Enable

Setting this bit enables the ADC.

University of Florida, EEL-5666  
© Drs. A. A. Arroyo & E. M. Schwartz

6



## IMDL Software Series ADC

- The ADC 12-bit result is placed in the CHnRES Registers, CHnRESH and CHnRESL. By default, the result is presented right adjusted, but can optionally be presented left adjusted.

Example:

**ADCA\_CTRLB = 0x00; //12bit, right adjusted**

CTRLB – ADC Control register B

Bit	7	6	5	4	3	2	1	0
+0x01	IMPMODE	CURRLIMIT[1:0]	CONVMODE	FREERUN	RESOLUTION[1:0]	–	–	–
ReadWrite	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R
Initial Value	0	0	0	0	0	0	0	0

- Bit 3 – FREERUN: Free Running Mode

When the bit is set to one, the ADC is in free running mode and the ADC channels defined in the EVCTRL register are swept repeatedly.

- Bit 2:1 – RESOLUTION[1:0]: Conversion Result Resolution

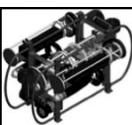
These bits define whether the ADC completes the conversion at 12- or 8-bit result resolution. They also define whether the 12-bit result is left or right adjusted within the 16-bit result registers. See Table 28-4 on page 352 for possible settings.

Table 28-4. ADC conversion result resolution.

RESOLUTION[1:0]	Group configuration	Description
00	12BIT	12-bit result, right adjusted
01	Reserved	
10	8BIT	8-bit result, right adjusted
11	LEFT12BIT	12-bit result, left adjusted

University of Florida, EEL-5666  
© Drs. A. A. Arroyo & E. M. Schwartz

7



## IMDL Software Series ADC

- The ADC reference is selected in the REFCTRL.

Example: On my Epiphany DIY

**ADCA\_REFCTRL = 0x10; //Set to Vref = 5v on ADCA**

**//Set to Vref = Vcc/1.6 ≈ 2.0v on ADCB**

REFCTRL – Reference Control register

Bit	7	6	5	4	3	2	1	0
+0x02	–	–	REFSEL[2:0]	–	–	–	BANDGAP	TEMPREF
ReadWrite	R	R/W	R/W	R/W	R	R	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- Bit 7 – Reserved

This bit is unused and reserved for future use. For compatibility with future devices, always write this bit to zero when this register is written.

- Bits 6:4 – REFSEL[2:0]: Reference Selection

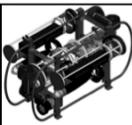
These bits selects the reference for the ADC according to Table 28-5 on page 353.

Table 28-5. ADC reference selection.

REFSEL[2:0]	Group configuration	Description
000	INT1V	10/11 of bandgap (1.0V)
001	INTVCC	V <sub>CC</sub> /1.6
010	AREFA	External reference from AREF pin on PORT A
011	AREFB	External reference from AREF pin on PORT B
100	INTVCC2	V <sub>CC</sub> /2
101 - 111	Reserved	

University of Florida, EEL-5666  
© Drs. A. A. Arroyo & E. M. Schwartz

8



## IMDL Software Series ADC

- Set Pre-Scaler Configuration. Example:  
`ADCA_PRESCALER = 0x03; // sets Prescaler to DIV32`

PRESCALER – Clock Prescaler register

Bit	7	6	5	4	3	2	1	0
+0x04	–	–	–	–	–	–	PRESCALER[2:0]	
Read/Write	R	R	R	R	R	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

• Bit 2:0 – PRESCALER[2:0]: Prescaler Configuration

These bits define the ADC clock relative to the peripheral clock according to Table 28-9 on page 355.

Table 28-9. ADC prescaler settings.

PRESCALER[2:0]	Group configuration	Peripheral clock division factor
000	DIV4	4
001	DIV8	8
010	DIV16	16
011	DIV32	32
100	DIV64	64
101	DIV128	128
110	DIV256	256
111	DIV512	512



## IMDL Software Series ADC

- To start single-ended conversions we use the CTRL register. Example:

`ADCA_CH0_CTRL = 0x01; //set to single-ended`

CTRL – Channel Control register

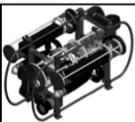
Bit	7	6	5	4	3	2	1	0
+0x10	START	–	–	GAIN[2:0]	–	–	INPUTMODE[1:0]	
Read/Write	R/W	R	R	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

• Bit 7 – START: START Conversion on Channel

Setting this bit will start a conversion on the channel. The bit is cleared by hardware when the conversion has started. Setting this bit when it already is set will have no effect. Writing or reading this bit is equivalent to writing the CH[3:0]START bits in "CTRLA – Control register A" on page 351.

Table 28-11. Channel input modes, CONVMODE=0 (unsigned mode).

INPUTMODE[1:0]	Group configuration	Description
00	INTERNAL	Internal positive input signal
01	SINGLEENDED	Single-ended positive input signal
10		Reserved
11		Reserved



## IMDL Software Series ADC

- The MUX register defines the input source for the channel. Example:

ADCA\_CH0\_MUXCTRL = 0x08; //default to Channel 1

### MUXCTRL – ADC Channel MUX Control registers

The MUXCTRL register defines the input source for the channel.

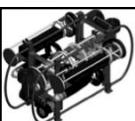
Bit	7	6	5	4	3	2	1	0
+0x01	–	–	MUXPOS[3:0]	–	–	MUXNEG[2:0]	–	–
Read/Write	R	R/W	R/W	R/W	R/W	R	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Table 28-14. ADC MUXPOS configuration when INPUTMODE[1:0] = 01 (single-ended) or INPUTMODE[1:0] = 10 (differential) is used.

MUXPOS[3:0]	Group configuration	Description
0000	PIN0	ADC0 pin
0001	PIN1	ADC1 pin
0010	PIN2	ADC2 pin
0011	PIN3	ADC3 pin
0100	PIN4	ADC4 pin
0101	PIN5	ADC5 pin
0110	PIN6	ADC6 pin
0111	PIN7	ADC7 pin

University of Florida, EEL-5666  
© Drs. A. A. Arroyo & E. M. Schwartz

11



## IMDL Software Series ADC

- Conversions are controlled with the INTCTRL. Example:

ADCA\_CH0\_INTCTRL = 0x00; //set flag at conversion complete. Disable interrupt

### INTCTRL – Channel Interrupt Control registers

Bit	7	6	5	4	3	2	1	0
+0x02	–	–	–	–	INTMODE[1:0]	–	INTLVL[1:0]	–
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- Bit 3:2 – INTMODE: Interrupt Mode

These bits select the interrupt mode for the channel according to Table 28-18.

Table 28-18. ADC channel select.

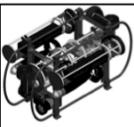
INTMODE[1:0]	Group configuration	Interrupt mode
00	COMPLETE	Conversion complete
01	BELOW	Compare result below threshold
10		Reserved
11	ABOVE	Compare result above threshold

- Bits 1:0 – INTLVL[1:0]: Interrupt Priority Level and Enable

These bits enable the ADC channel interrupt and select the interrupt level, as described in "Interrupts and Programmable Multilevel Interrupt Controller" on page 130. The enabled interrupt will be triggered for conditions when the IF bit in the INTFLAGS register is set.

University of Florida, EEL-5666  
© Drs. A. A. Arroyo & E. M. Schwartz

12



## IMDL Software Series ADC

- Conversions flags are in the INTFLAGS.

Example:

```
// wait for conversion to complete
while ((ADCA_CH0_INTFLAGS & 0x01) != 0x01);
```

### INTFLAGS – ADC Channel Interrupt Flag registers

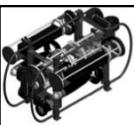
Bit	7	6	5	4	3	2	1	0
+0x03	—	—	—	—	—	—	—	IF
ReadWrite	R	R	R	R	R	R	R	R/W
Initial Value	0	0	0	0	0	0	0	0

- Bit 7:1 – Reserved

These bits are unused and reserved for future use. For compatibility with future devices, always write these bits to zero when this register is written.

- Bit 0 – IF: Channel Interrupt Flag

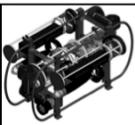
The interrupt flag is set when the ADC conversion is complete. If the channel is configured for compare mode, the flag will be set if the compare condition is met. IF is automatically cleared when the ADC channel interrupt vector is executed. The bit can also be cleared by writing a one to the bit location.



## IMDL Software Series ADC - Epiphany

```
* AAA_Util.c - Notepad2 (Administrator)
File Edit View Settings ?
1 //#####
2 /* Xmega */
3 #####
4 void xmegainit(void) {           // xmegainit: initialize Xmega CPU to 32MHz
5   CCP      = 0xB8;              // Configuration Change Protection: IOREG (0xB8)
6   CLK_PSCTRL = 0x00;            // System Clock Prescaler Reg All prescalers No Division
7   OSC_CTRL  = 0x02;            // Bit 1- RC32MEN: Setup 32 MHz Internal CPU RC oscillator Enable
8   while ((OSC_STATUS & 0x02) == 0); // wait for Bit 1 - RC32MRDY: 32 MHz Internal RC oscillator Ready
9   CCP      = 0xB8;              // Configuration Change Protection: IOREG (0xB8)
10  CLK_CTRL = 0x01;             // Bit 2:0 - SCLKSEL[2:0]: System Clock Selection
11 } // End Xmegainit           // 001 : RC32MHz 32MHz Internal RC oscillator
12 #####
13 /*
14 * ADCA *
15 *****/
16 void ADCAinit(void) {           // ADCAinit: Initialize ADC on PortA pins
17   ADCA_CTRLB = 0x00;            // 12bit, right adjusted
18   ADCA_REFCTRL = 0x10;          // set to Vref = Vcc/1.6 = 2.0V (approx) In Tim's Board 5V
19   ADCA_CH0_CTRL = 0x01;          // set to single-ended
20   ADCA_CH0_INTCTRL = 0x00;        // set flag at conversion complete. Disable interrupt
21   ADCA_CH0_MUXCTRL = 0x08;        // default set to channel ADC1 PortA Pin 1
22   ADCA_PRESCALER = 0x03;          // sets ADC prescaler to DIV32
23   ADCA_CTRLA |= 0x01;            // Enable ADCA
24 } // End ADCAinit
25 #####
26 int analog(int channel) {       // analog(channel[ADCA PortA<0:7>]) Read the PortA ADC channel
27   ADCA_CH0_MUXCTRL = channel*8; // Set to ADCh PortA Pin n where n=channel is <7-0>
28   ADCA_CTRLA |= 0x04;            // Start Conversion on ADCA Channel 0
29   while ((ADCA_CH0_INTFLAGS & 0x01) != 0x01); // wait for conversion to complete
30   _delay_ms(5);                // wait for result to stabilize (may not be needed)
31   int value      = ADCA_CH0_RES; // grab result
32   return value;                // return result
33 } // End analog(channel)
34 #####

```



## IMDL Software Series ADC - DAQ

```
#include "adc.h"
#include "adcIn0.h"

void adcInit0()
{
    //ReadCalibrationByte offset=0x0000_0000_SIGNATURES_1, ADCACALB1 };
    ADC_CALH |= ReadCalibrationByte( offset=0x0000_0000_SIGNATURES_1, ADCACALB1 );
    //ReadCalibrationByte offset=0x0000_0000_SIGNATURES_2, ADCACALB2 };
    ADC_CALH |= ReadCalibrationByte( offset=0x0000_0000_SIGNATURES_2, ADCACALB2 );

    //Digital of analog start up sequence
    NCU_STARTUPOLY0_bn |= NCU_STARTUPOLY01_bn;

    //select the 2.5V external reference
    //ADC_AREFCTRL = {0x00<4>;} // 0b 0011 0000 = 0x38 /* External reference on PORT B */
    ADC_AREFCTRL |= ADC_AREFSEL_ANER9_gc;

    //ADC prescaler setting
    //ADC_PRESCALER = {0x00<8>;} // 0b 0000 0101 = 0x05 /* Divide clock by 128 */
    ADC_PRESCALER = ADC_PRESCALER_DIV128_gc;

    //setup the ADC0 configuration
    //ADC_CTRL0 = {0x00<8>;0x00<3>;} | 0x10 | 0x000 | 0x000<11>; // 0b 1001 1000 = 0x80
    ADC_CTRL0 |= ADC_CTRL0_ANSEL_gc; //high impedance enable, since the board uses opamps for the front end
    ADC_CURRLIMIT_NO_gc |= ADC_CURRLIMIT_NO_gc; //eliminate the current limit in order to max out the ADC speed
    ADC_FREERUN_gc |= ADC_FREERUN_gc; //sets up the ADC for signed mode
    ADC_RESOLUTION_12BIT_gc |= ADC_RESOLUTION_12BIT_gc; //sets adc to have 12 bit resolution with right adjustment (LSB)

    //tell the ADC to sample all channels 0-3 automatically
    ADC_EVCTRL = ADC_SWEEP_0123_gc;

    //*****Channel setup*****
    //presampling gain and input mode selection
    ADC_CH0_CTRL = ADC_CH_GAIN_4X_gc | ADC_CH_INPVTHDE_01PFAGAN_gc; // 0b 0000 1011 = 0x0B
    ADC_CH1_CTRL = ADC_CH_GAIN_4X_gc | ADC_CH_INPVTHDE_01PFAGAN_gc;
    ADC_CH2_CTRL = ADC_CH_GAIN_4X_gc | ADC_CH_INPVTHDE_01PFAGAN_gc;
    ADC_CH3_CTRL = ADC_CH_GAIN_4X_gc | ADC_CH_INPVTHDE_01PFAGAN_gc;
    ADC_CH4_CTRL = ADC_CH_GAIN_4X_gc | ADC_CH_INPVTHDE_01PFAGAN_gc;
    ADC_CH5_CTRL = ADC_CH_GAIN_4X_gc | ADC_CH_INPVTHDE_01PFAGAN_gc;

    //max positions
    //ADC_CH0_MUXCTRL = {0x00<3>;} | {0x00<8>;} // 0b 0001 0000 = 0x10
    ADC_CH0_MUXCTRL |= ADC_CH_MUXPOS_PIN0_gc | ADC_CH_MUXNEG_PIN2_gc;
    //ADC_CH1_MUXCTRL = {0x00<3>;} | {0x00<8>;} // 0b 0000 0000 = 0x00
    ADC_CH1_MUXCTRL |= ADC_CH_MUXPOS_PIN1_gc | ADC_CH_MUXNEG_PIN3_gc;
    //ADC_CH2_MUXCTRL = {0x00<3>;} | {0x00<8>;} // 0b 0001 0000 = 0x10
    ADC_CH2_MUXCTRL |= ADC_CH_MUXPOS_PIN2_gc | ADC_CH_MUXNEG_PIN4_gc;
    //ADC_CH3_MUXCTRL = {0x00<3>;} | {0x00<8>;} // 0b 0001 0000 = 0x10
    ADC_CH3_MUXCTRL |= ADC_CH_MUXPOS_PIN3_gc | ADC_CH_MUXNEG_PIN5_gc;
    //ADC_CH4_MUXCTRL = {0x00<3>;} | {0x00<8>;} // 0b 0001 0000 = 0x10
    ADC_CH4_MUXCTRL |= ADC_CH_MUXPOS_PIN4_gc | ADC_CH_MUXNEG_PIN6_gc;
    //ADC_CH5_MUXCTRL = {0x00<3>;} | {0x00<8>;} // 0b 0001 0000 = 0x10
    ADC_CH5_MUXCTRL |= ADC_CH_MUXPOS_PIN5_gc | ADC_CH_MUXNEG_PIN7_gc;

    //ADC_CH0_INTCTRL = {0x00<8>;1} | {1<0>;}; // 0b 0000 0001 = 0x01
    ADC_CH0_INTCTRL |= ADC_CH0_INTPIN0_gc | ADC_CH0_INTPIN1_gc;
    //ADC_CH1_INTCTRL = {0x00<8>;1} | {1<0>;}; // 0b 0000 0001 = 0x01
    ADC_CH1_INTCTRL |= ADC_CH1_INTPIN0_gc | ADC_CH1_INTPIN1_gc;
    //ADC_CH2_INTCTRL = {0x00<8>;1} | {1<0>;}; // 0b 0000 0001 = 0x01
    ADC_CH2_INTCTRL |= ADC_CH2_INTPIN0_gc | ADC_CH2_INTPIN1_gc;
    //ADC_CH3_INTCTRL = {0x00<8>;1} | {1<0>;}; // 0b 0000 0001 = 0x01
    ADC_CH3_INTCTRL |= ADC_CH3_INTPIN0_gc | ADC_CH3_INTPIN1_gc;
    //ADC_CH4_INTCTRL = {0x00<8>;1} | {1<0>;}; // 0b 0000 0001 = 0x01
    ADC_CH4_INTCTRL |= ADC_CH4_INTPIN0_gc | ADC_CH4_INTPIN1_gc;
    //ADC_CH5_INTCTRL = {0x00<8>;1} | {1<0>;}; // 0b 0000 0001 = 0x01
    ADC_CH5_INTCTRL |= ADC_CH5_INTPIN0_gc | ADC_CH5_INTPIN1_gc;

    //ADC_CH0_CTRL |= 0x00; // 0b 0000 0000 = 0x00
    ADC_CH0_CTRL |= ADC_CH0_START_bm;
    //ADC_CH1_CTRL |= 0x00; // 0b 0000 0000 = 0x00
    ADC_CH1_CTRL |= ADC_CH1_START_bm;
    //ADC_CH2_CTRL |= 0x00; // 0b 0000 0000 = 0x00
    ADC_CH2_CTRL |= ADC_CH2_START_bm;
    //ADC_CH3_CTRL |= 0x00; // 0b 0000 0000 = 0x00
    ADC_CH3_CTRL |= ADC_CH3_START_bm;
    //ADC_CH4_CTRL |= 0x00; // 0b 0000 0000 = 0x00
    ADC_CH4_CTRL |= ADC_ENABLE_bm;
    //ADC_CH5_CTRL |= 0x00; // 0b 0000 0000 = 0x00
    ADC_CH5_CTRL |= ADC_ENABLE_bm;
    //PMIC_CTRL |= 0x01 | 0x00; // 0x01
    PMIC_CTRL |= PMIC_U0_VLEN_bm | PMIC_RHEN_bm;
}
```

University of Florida, EEL-5666  
© Drs. A. A. Arroyo & E. M. Schwartz

15



## IMDL Software Series ADC

- The next example uses the ADC to sample a Cds cell and turn on/off the debug LED on PC1 in the main loop in Tim's DAQ Board.

See project DAQ\_GPIO\_ADC

- If you wish to use Tim's USART functions, then import uart.c and uart.h from Tim's Software

See project DAQ\_GPIO\_ADC  
Change the include in main.c to <uart.h>

University of Florida, EEL-5666  
© Drs. A. A. Arroyo & E. M. Schwartz

16



## IMDL Software Series Arduino ADC

- The example uses the ADC functions in the Arduino MEGA2560 Board to blink internal/external LEDs and uses the serial port to display data and control the LEDs after blinking the set a few times. Serial input  $j$  &  $k$  alternate the two external LEDs on pins 12 & 11 {on, off} and {off, on} respectively. Analog input A0 is connected to a Cds cell that controls the internal DBLED
- See Arduino Sketch: IMDL\_ADC\_Serial\_External\_LED

University of Florida, EEL-5666  
© Drs. A. A. Arroyo & E. M. Schwartz

17



## IMDL Software Series

```
1  /*
2   * EEL-5666 Example 3: Use of Digital Output Pins
3   * Turn on/off External LEDs connected to Pins 11 & 12
4   * alternatively and the internal LED on Pin 13 on/off.
5   * Add a delay in each state of 1/2 sec. Use serial input
6   * to control the LEDs. {i,o} for internal LED {j,k} for
7   * external LEDs. Read Cds cell on Analog A0 and control
8   * the internal LED.
9   * Arroyo January 18, 2016
10 */
11
12
13 int extLED1 = 11; // External LED connected to digital pin 11
14 int extLED2 = 12; // External LED connected to digital pin 12
15 int intLED = 13; // Internal LED connected to digital pin 13
16
17 void setup() {
18     pinMode(extLED1, OUTPUT); // sets digital pin 12 as output
19     pinMode(extLED2, OUTPUT); // sets digital pin 12 as output
20     pinMode(intLED, OUTPUT); // sets digital pin 13 as output
21     Serial.begin(57600); // connect to the serial port
22 }
23
24     int rcvData=0, count=0, ambient, a1;
25 void loop() {
26     if (count < 1) {
27         Serial.println("Arduino MEGA2560 Example");
28         Serial.println("Cds Cell connected on Analog A0");
29         Serial.println("Control the internal LED on pin 13");
30         ambient = analogRead(A0);
31         Serial.print("Ambient: ");
32         Serial.println(ambient);
33
34         for (int j=0; j<5; j++){
35             Serial.print("j= ");
36             Serial.println(j);
37             digitalWrite(13, HIGH); // set the internal LED on
```

University of Florida, EEL-5666  
© Drs. A. A. Arroyo & E. M. Schwartz

Normal text file

length: 2739 lines: 74

ln: 28 Col: 55 Sel: 3 | 0

18



## IMDL Software Series

```
38     digitalWrite(12, LOW); // set the external LED1 off
39     digitalWrite(11, HIGH); // set the external LED2 on
40     delay(500); // wait for a 1/2 second
41     digitalWrite(13, LOW); // set the internal LED off
42     digitalWrite(12, HIGH); // set the external LED on
43     digitalWrite(11, LOW); // set the external LED off
44     delay(500); // wait for a 1/2 second
45   }
46   digitalWrite(13, HIGH); // set the internal LED on
47   digitalWrite(12, HIGH); // set the external LED1 on
48   digitalWrite(11, HIGH); // set the external LED2 on
49 }
50 count=1;
51 Serial.print("Ambient: ");
52 Serial.print(ambient);
53 Serial.print(" A1: ");
54 a1 = analogRead(A0);
55 if (a1 > 1.1*ambient ) digitalWrite(13, LOW); // turn off DBLED
56 else digitalWrite(13, HIGH); // else turn on DBLED
57 Serial.println(a1);
58 delay(100);
59
60 // blink data only when you receive data:
61 if (Serial.available() > 0) {
62 // read the incoming byte:
63 rcvData = Serial.read();
64 // say what you got:
65 Serial.print("I received: ");
66 Serial.write(rcvData);
67 Serial.println();
68 if (rcvData == 'i') digitalWrite(13, HIGH); // internal LED on
69 else if (rcvData == 'o') digitalWrite(13, LOW); // internal LED off
70 else if (rcvData == 'j') (digitalWrite(12,LOW); digitalWrite(11,HIGH););
71 else if (rcvData == 'k') (digitalWrite(12,HIGH); digitalWrite(11,LOW););
72 }
73 }
74 }
```

Normal text file

length:2799 lines:74

Ln:28 Col:55 Sel:3|0

19

University of Florida, EEL-5666  
© Drs. A. A. Arroyo & E. M. Schwartz

University of Florida, EEL-5666  
© Drs. A. A. Arroyo & E. M. Schwartz

# *The End!*

20