Micaz Hardware Overview and ADC Sample Code

Aly El-Osery
Electrical Engineering Department
New Mexico Tech
Socorro, NM 87801

Overview

- Brief overview of Atmega 128L
- Important files
- Digital I/O
- ADC sample code

Atmega128L

- Low-Power AVR 8-bit Microcontroller
- 128K Bytes of In-System Reprogrammable Flash
- Two 8-bit Timer/Counters
- Two 16-bit Timer/Counters
- 8-channel, 10-bit ADC

Micaz

- 51 pin I/O connector.
- 8-channel 10 bit ADC 0-3V input.
- Timers 0,1, and 2 are used for the general purpose timer and for the radio.
- General Purpose I/O, some are tide to specific functions so use with care.
- PW0-PW7 are available general purpose I/O.

iom128.h

- tinyos-1.x/local/avr/include/avr/iom128.h

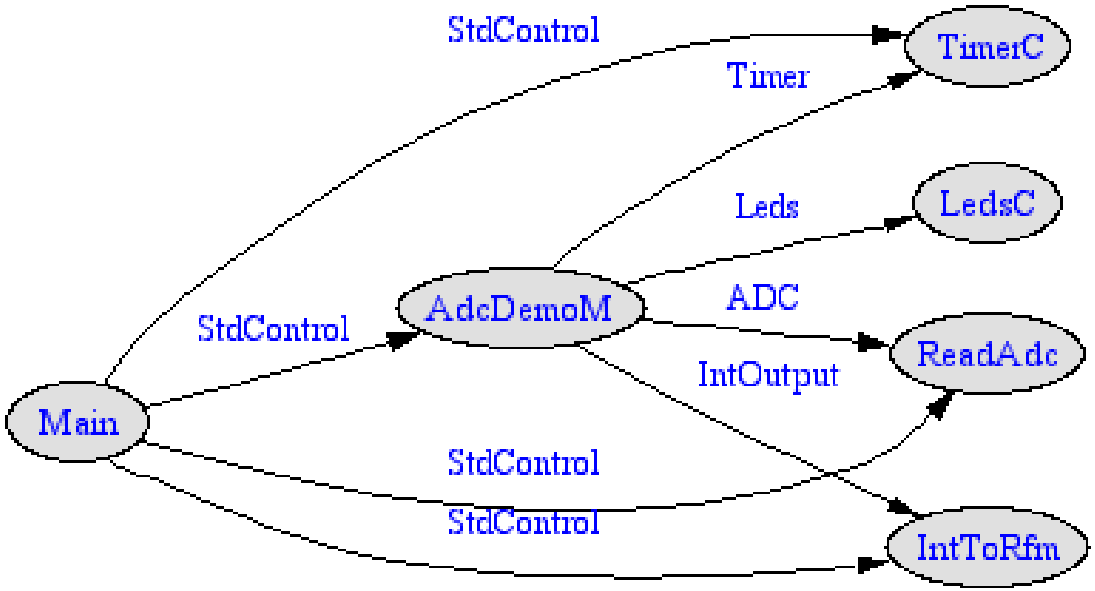
avrhardware.h

```
#define TOSH_ASSIGN_PIN(name, port, bit) \
static inline void TOSH_SET_##name##_PIN() {sbi(PORT##port , bit);} \
static inline void TOSH_CLR_##name##_PIN() {cbi(PORT##port , bit);} \

static inline int TOSH_READ_##name##_PIN() \
  {return (inp(PIN##port) & (1 << bit)) != 0;} \

static inline void TOSH_MAKE_##name##_OUTPUT() {sbi(DDR##port , bit);} \
static inline void TOSH_MAKE_##name##_INPUT() {cbi(DDR##port , bit);}
```

micaz/hardware.h

```
TOSH_ASSIGN_PIN(PW0, C, 0);
TOSH_ASSIGN_PIN(PW1, C, 1);
TOSH_ASSIGN_PIN(PW2, C, 2);
TOSH_ASSIGN_PIN(PW3, C, 3);
TOSH_ASSIGN_PIN(PW4, C, 4);
TOSH_ASSIGN_PIN(PW5, C, 5);
TOSH_ASSIGN_PIN(PW6, C, 6);
TOSH_ASSIGN_PIN(PW7, C, 7);

  TOSH_MAKE_PW7_OUTPUT();
  TOSH_MAKE_PW6_OUTPUT();
  TOSH_MAKE_PW5_OUTPUT();
  TOSH_MAKE_PW4_OUTPUT();
  TOSH_MAKE_PW3_OUTPUT();
  TOSH_MAKE_PW2_OUTPUT();
  TOSH_MAKE_PW1_OUTPUT();
  TOSH_MAKE_PW0_OUTPUT();
```

AdcDemo.nc



AdcDemo.nc

```
configuration AdcDemo {
// this module does not provide any interface
}
implementation
{
  components Main, AdcDemoM, LedsC, TimerC, ReadAdc, IntToRfm;

  Main.StdControl -> TimerC;
  Main.StdControl -> AdcDemoM;
  Main.StdControl -> IntToRfm;

  AdcDemoM.ADC -> ReadAdc;
  AdcDemoM.ADCControl -> ReadAdc;
  AdcDemoM.Leds -> LedsC;
  AdcDemoM.Timer -> TimerC.Timer[unique("Timer")];
  AdcDemoM.IntOutput -> IntToRfm;
}
```

AdcDemoM.nc

```
event result_t Timer.fired() {
  return call ADC.getData();
}

// ADC data ready event handler
async event result_t ADC.dataReady(uint16_t data) {
  display((data>>7) &0x7);
  call IntOutput.output(data);
  return SUCCESS;
}
```

ReadAdc.nc

ReadAdc.nc

```
includes sensorboard;
configuration ReadAdc
{
  provides interface ADC as ReadAdcADC;
  provides interface StdControl;
}
implementation
{
  components ReadAdcM, ADCC;

  StdControl = ReadAdcM;
  ReadAdcADC = ADCC.ADC[TOS_ADC_PHOTO_PORT];
  ReadAdcM.ADCControl -> ADCC;
}
```

ReadAdcM.nc

```
includes sensorboard;
module ReadAdcM {
  provides interface StdControl;
  uses {
    interface ADCControl;
  }
}
implementation {
  command result_t StdControl.init() {
    call ADCControl.bindPort(TOS_ADC_PHOTO_PORT,TOSH_ACTUAL_PHOTO_PORT);
    return call ADCControl.init();
  }
  command result_t StdControl.start() {
    return SUCCESS;
  }
  command result_t StdControl.stop() {
    return SUCCESS;
  }
}
```

sensorboard.h

```
enum {
  TOSH_ACTUAL_PHOTO_PORT = 2,
};

enum {
  TOS_ADC_PHOTO_PORT = 2,
};
```

ADC reading

$$ADC_{value} = \frac{V_{input} \cdot 1024}{V_{ref}}$$

Returned value depends on the value of V_ref which depends on the battery value

Battery Monitor

Select channel 30 = TOS_ADC_VOLTAGE_PORT

This will select a fixed reference input to the ADC V_BG= 1.223V

$$V_{Batt} = \frac{V_{BG} \cdot 1024}{ADC_{value}}$$