```
/*


CELL MODULE FOR ATTINY1624

(c)2019 to 2021 Stuart Pittaway

MODIFIED BY
2024 Nidula Gunawardana

COMPILE THIS CODE USING MICROCHIP STUDIO

LICENSE
Attribution-NonCommercial-ShareAlike 2.0 UK: England & Wales (CC BY-NC-SA 2.0
UK)
https://creativecommons.org/licenses/by-nc-sa/2.0/uk/

* Non-Commercial — You may not use the material for commercial purposes.
* Attribution — You must give appropriate credit, provide a link to the license,
  and indicate if changes were made.
  You may do so in any reasonable manner, but not in any way that suggests the
    licensor endorses you or your use.
* ShareAlike — If you remix, transform, or build upon the material, you must
  distribute your
  contributions under the same license as the original.
* No additional restrictions — You may not apply legal terms or technological
  measures
  that legally restrict others from doing anything the license permits.
*/
/*



/*

HARDWARE ABSTRACTION CODE FOR tinyAVR2

*/

#include <avr/io.h>
#include <util/delay.h>

// ATtiny1624 specific register manipulation code

void FlashNotificationLed(size_t times, uint32_t milliseconds) {
    for (size_t i = 0; i < times; i++) {
        PORTA.OUTSET = 0x40; // PA6 = PIN6_bm (0x40) NotificationLedOn
        _delay_ms(milliseconds);
        PORTA.OUTCLR = 0x40; // PA6 = PIN6_bm (0x40) NotificationLedOff
        _delay_ms(milliseconds);
    }
}

void PowerOn_Notification_led() {
    FlashNotificationLed(4, 150);
}
```

```
void double_tap_Notification_led() {
    FlashNotificationLed(2, 50);
}

void ConfigurePorts() {
    // pin out
    // AVR PIN / ARDUINO PIN MAPPING
    // PB0 /7= ENABLE
    // PB1 /6= DUMP LOAD ENABLE
    // PB2 /5= TXD
    // PB3 /4= RXD
    // PA0 /11= RESET
    // PA1 /8= REF_ENABLE
    // PA2 /9= NOT CONNECTED
    // PA3 /10= EXTERNAL TEMP SENSOR (ADC) (ADC0=AIN3)
    // PA4 /0= VOLTAGE INPUT (ADC) (ADC0=AIN4)
    // PA5 /1= VREFERENCE (ADC) (VREFA/ ADC0=AIN5)
    // PA6 /2= NOTIFICATION LED
    // PA7 /3= INTERNAL TEMP SENSOR (ADC)(ADC0=AIN7)

    // Set Port A digital outputs
    PORTA.DIRSET = 0x42; // PIN1_bm | PIN6_bm | PIN2_bm = 0x42

    // Set Port B digital outputs
    PORTB.DIRSET = 0x07; // PIN0_bm | PIN1_bm | PIN2_bm = 0x07
    // Set RX as input
    PORTB.DIRCLR = 0x08; // PIN3_bm = 0x08

    // Set Port A analogue inputs
    PORTA.DIRCLR = 0x88; // PIN3_bm | PIN7_bm = 0x88

    // Disable digital input buffer for unused pins and analog inputs
    PORTA.PIN0CTRL = 0x03; // PORT_ISC_INPUT_DISABLE_gc = 0x03
    PORTA.PIN1CTRL = 0x03;
    PORTA.PIN2CTRL = 0x03;
    PORTA.PIN3CTRL = 0x03;
    PORTA.PIN4CTRL = 0x03;
    PORTA.PIN5CTRL = 0x03;
    PORTA.PIN6CTRL = 0x03;
    PORTA.PIN7CTRL = 0x03;

    PORTB.PIN0CTRL = 0x03;
    PORTB.PIN1CTRL = 0x03;

    // Step 1: Enable ADC
    ADC0.CTRLA = 0x01; // ADC_ENABLE_bm = 0x01
    // PRESC[3:0], DIV16 = 5Mhz/2 = 2500000hz
    ADC0.CTRLB = 0x01; // ADC_PRESC_DIV2_gc = 0x01
    // SAMPDUR[7:0]
    ADC0.CTRLE = 0x80; // 128
    // WINSRC / WINCM[2:0]
    ADC0.CTRLD = 0x00;
    ADC0.PGACTRL = 0x00;

    // Set pins to initial state
```

```
    PORTB.OUTCLR = 0x02; // DumpLoadOff, PIN1_bm = 0x02
    PORTA.OUTCLR = 0x02; // ReferenceVoltageOff, PIN1_bm = 0x02
    PORTA.OUTCLR = 0x80; // TemperatureVoltageOff, PIN7_bm = 0x80
    PORTA.OUTCLR = 0x40; // NotificationLedOff, PIN6_bm = 0x40
}

uint16_t BeginADCReading(uint8_t mode) {
    uint16_t value = 0;

    // Enable ADC
    ADC0.CTRLA = 0x01; // ADC_ENABLE_bm = 0x01

    // TIMEBASE[4:0] / REFSEL[2:0]
    ADC0.CTRLC = 0x04; // TIMEBASE_1US = 0x04 | ADC_REFSEL_VDD_gc = 0x00

    // Take multiple samples (over sample)
    ADC0.COMMAND = 0x88; // ADC_MODE_BURST_SCALING_gc = 0x80 |
      ADC_START_IMMEDIATE_gc = 0x08
    while (!(ADC0.INTFLAGS & 0x01)); // ADC_RESRDY_bm = 0x01
    value = (uint16_t)ADC0.RESULT;

    // Switch off ADC
    ADC0.CTRLA &= ~0x01; // ADC_ENABLE_bm = 0x01

    return value;
}
```