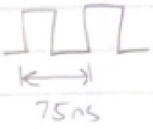
Total A/D sequence time = T_{smp} + T_{CNV}
= Sample time + A/D conversion time (SAR)

- A/D conversion requirements
  TAD: A/D clock period (min 75ns)
  ADCS <7:0> in AD1CON3



$$T_{AD} = \boxed{k} \cdot T_{cy} \geq 75ns$$

$f_{cy} = 16MHz$
$T_{cy} = 62.5ns$

K = 1 $\quad 1 \cdot T_{cy} = 62.5 \geq 75ns$
min k = 2 $\quad \boxed{2 \cdot T_{cy}} = 125 \geq 75ns$ (✓)
if k ≥ 3 $\quad k \cdot T_{cy}$ - Conversion takes longer

$$\boxed{\begin{array}{l} T_{AD} = T_{cy} \cdot (ADCS + 1) \\ ADCS = \frac{T_{AD}}{T_{cy}} - 1 \end{array}}$$

T_{CNV} : conversion time $(12 \cdot T_{AD})$
T_{smp} : Sampling time $T_{smp} = SAMC<4:0> \cdot T_{AD}$

E.g. Find values of ADCS and SAMC
- Fastest conversion

$f_{cy} = 16MHz$
$T_{smp} \geq 2.5$ us

$ADCS = \frac{125ns}{62.5ns} - 1 \qquad SAMC = \frac{2.5us}{2 \cdot 62.5ns}$

$\frac{1}{k_{samples}}$ = Sampling time

ADCS = 1 $\qquad$ SAMC = 20

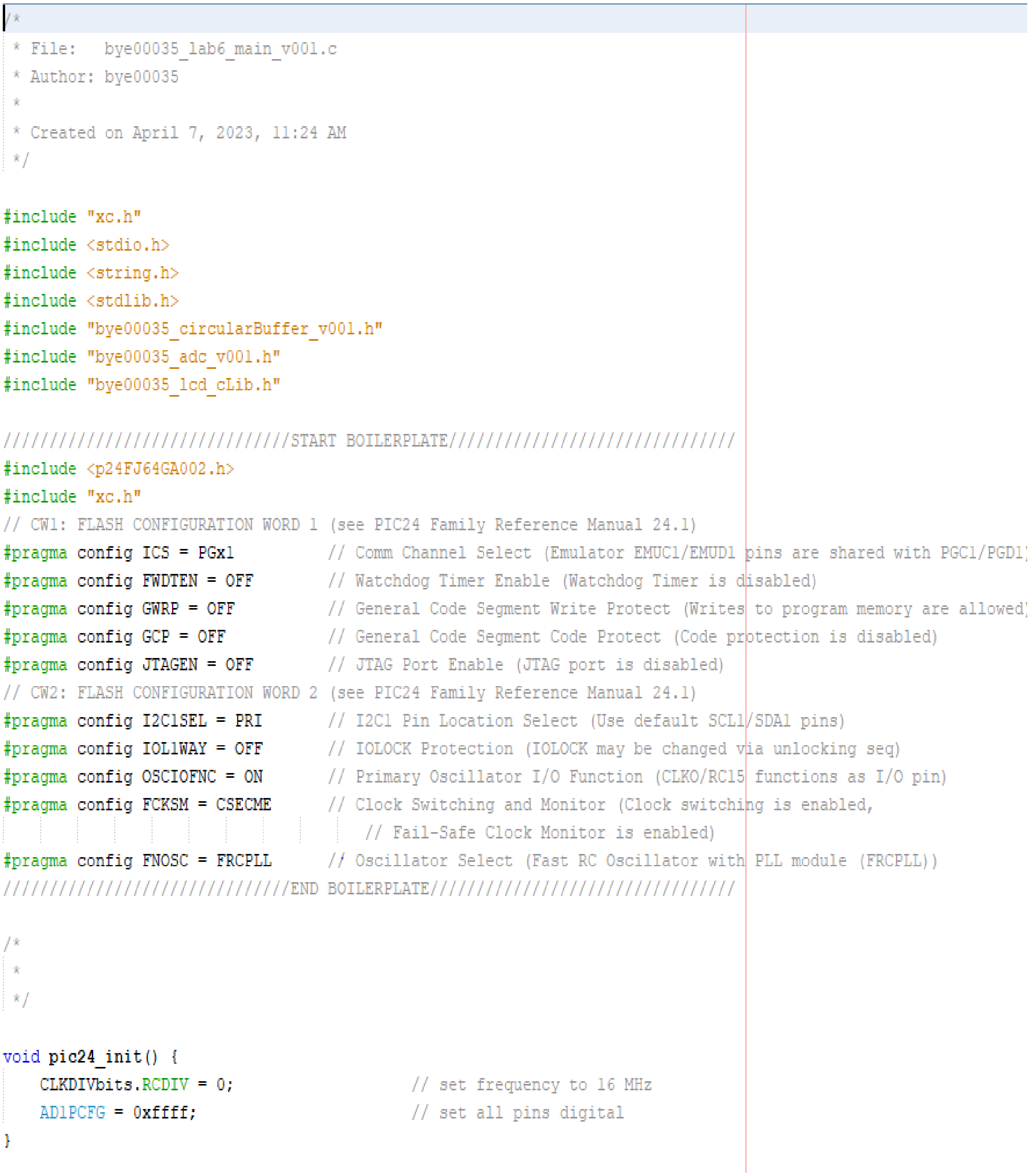
$$\boxed{\frac{1}{4us} = 250 \text{ samples per ms}}$$

How long the whole A/D sequence time?
$T_{ADC} = T_{smp} + T_{CNV}$
$20 \cdot T_{AD} + 12 \cdot T_{AD}$
$T_{ADC} = 4$ usec

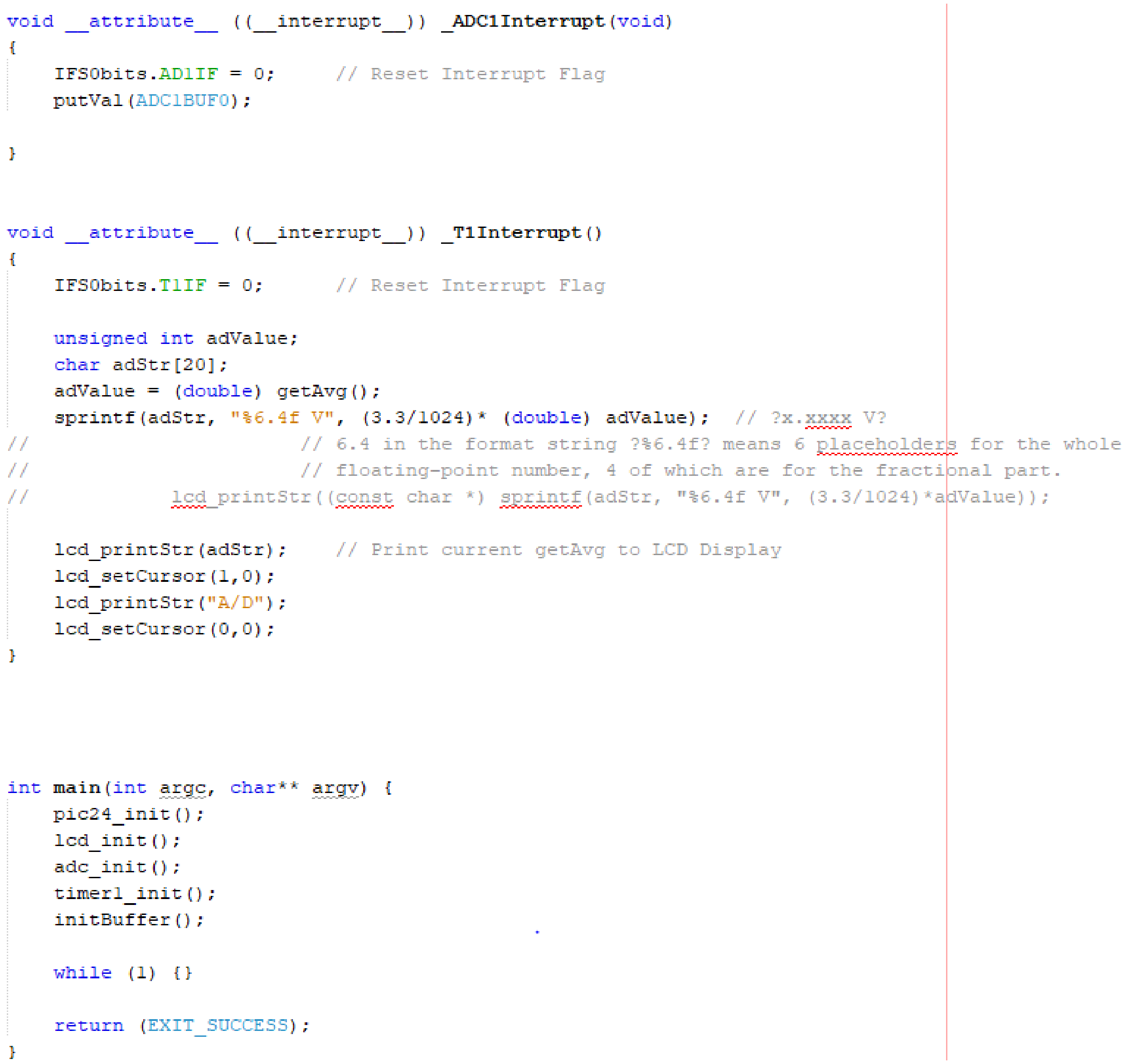
k is inversely related to sampling time

Max sampling rate is 250 samples per ms, in a sampling time of 4 us, these properties are inversely proportional in that the increase in sampling time (an increase in k) will lead to an decrease in sampling rate and vice versa. When our sample rate increases, we are able to paint a more accurate picture on our display since we are collecting precise values.

Main

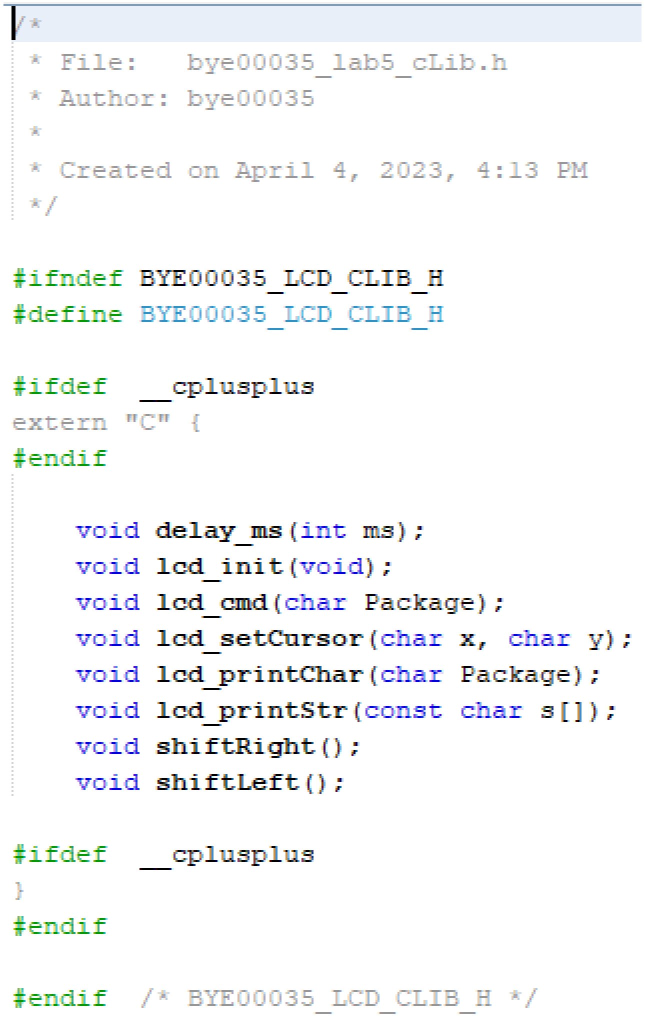


```c
/*
 * File:   bye00035_lab6_main_v001.c
 * Author: bye00035
 *
 * Created on April 7, 2023, 11:24 AM
 */

#include "xc.h"
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "bye00035_circularBuffer_v001.h"
#include "bye00035_adc_v001.h"
#include "bye00035_lcd_cLib.h"

///////////////////////////////START BOILERPLATE/////////////////////////////
#include <p24FJ64GA002.h>
#include "xc.h"
// CW1: FLASH CONFIGURATION WORD 1 (see PIC24 Family Reference Manual 24.1)
#pragma config ICS = PGx1          // Comm Channel Select (Emulator EMUC1/EMUD1 pins are shared with PGC1/PGD1)
#pragma config FWDTEN = OFF        // Watchdog Timer Enable (Watchdog Timer is disabled)
#pragma config GWRP = OFF          // General Code Segment Write Protect (Writes to program memory are allowed)
#pragma config GCP = OFF           // General Code Segment Code Protect (Code protection is disabled)
#pragma config JTAGEN = OFF        // JTAG Port Enable (JTAG port is disabled)
// CW2: FLASH CONFIGURATION WORD 2 (see PIC24 Family Reference Manual 24.1)
#pragma config I2C1SEL = PRI       // I2C1 Pin Location Select (Use default SCL1/SDA1 pins)
#pragma config IOL1WAY = OFF       // IOLOCK Protection (IOLOCK may be changed via unlocking seq)
#pragma config OSCIOFNC = ON       // Primary Oscillator I/O Function (CLKO/RC15 functions as I/O pin)
#pragma config FCKSM = CSECME      // Clock Switching and Monitor (Clock switching is enabled,
                                       // Fail-Safe Clock Monitor is enabled)
#pragma config FNOSC = FRCPLL      // Oscillator Select (Fast RC Oscillator with PLL module (FRCPLL))
///////////////////////////////END BOILERPLATE/////////////////////////////////

/*
 *
 */

void pic24_init() {
    CLKDIVbits.RCDIV = 0;                   // set frequency to 16 MHz
    AD1PCFG = 0xffff;                       // set all pins digital
}
```

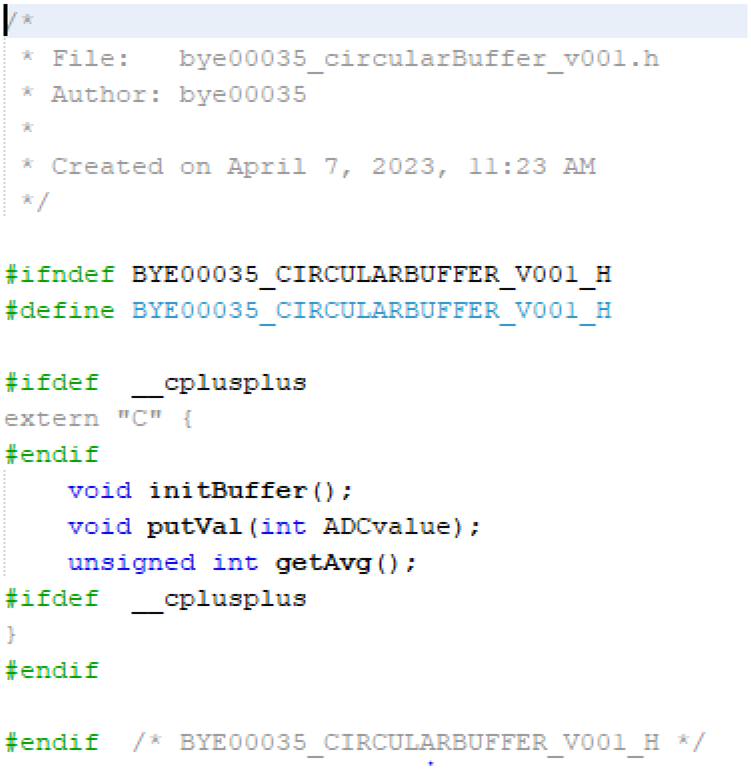


```
void __attribute__ ((__interrupt__)) _ADC1Interrupt(void)
{
    IFS0bits.AD1IF = 0;     // Reset Interrupt Flag
    putVal(ADC1BUF0);

}


void __attribute__ ((__interrupt__)) _T1Interrupt()
{
    IFS0bits.T1IF = 0;      // Reset Interrupt Flag

    unsigned int adValue;
    char adStr[20];
    adValue = (double) getAvg();
    sprintf(adStr, "%6.4f V", (3.3/1024)* (double) adValue);  // ?x.xxxx V?
//                       // 6.4 in the format string ?%6.4f? means 6 placeholders for the whole
//                       // floating-point number, 4 of which are for the fractional part.
//            lcd_printStr((const char *) sprintf(adStr, "%6.4f V", (3.3/1024)*adValue));

    lcd_printStr(adStr);    // Print current getAvg to LCD Display
    lcd_setCursor(1,0);
    lcd_printStr("A/D");
    lcd_setCursor(0,0);
}



int main(int argc, char** argv) {
    pic24_init();
    lcd_init();
    adc_init();
    timer1_init();
    initBuffer();

    while (1) {}

    return (EXIT_SUCCESS);
}
```

Header File

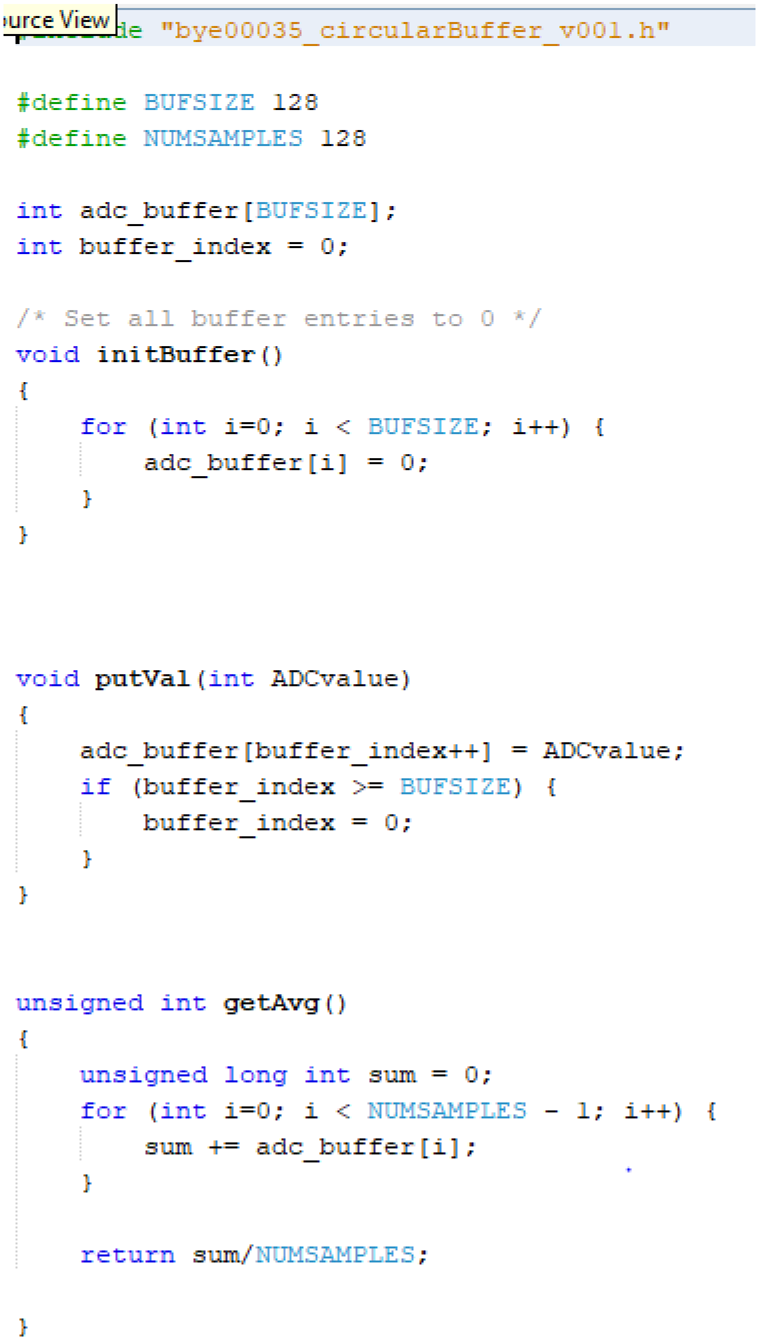


```c
/*
 * File:   bye00035_lab5_cLib.h
 * Author: bye00035
 *
 * Created on April 4, 2023, 4:13 PM
 */

#ifndef BYE00035_LCD_CLIB_H
#define BYE00035_LCD_CLIB_H

#ifdef  __cplusplus
extern "C" {
#endif

    void delay_ms(int ms);
    void lcd_init(void);
    void lcd_cmd(char Package);
    void lcd_setCursor(char x, char y);
    void lcd_printChar(char Package);
    void lcd_printStr(const char s[]);
    void shiftRight();
    void shiftLeft();

#ifdef  __cplusplus
}
#endif

#endif  /* BYE00035_LCD_CLIB_H */
```

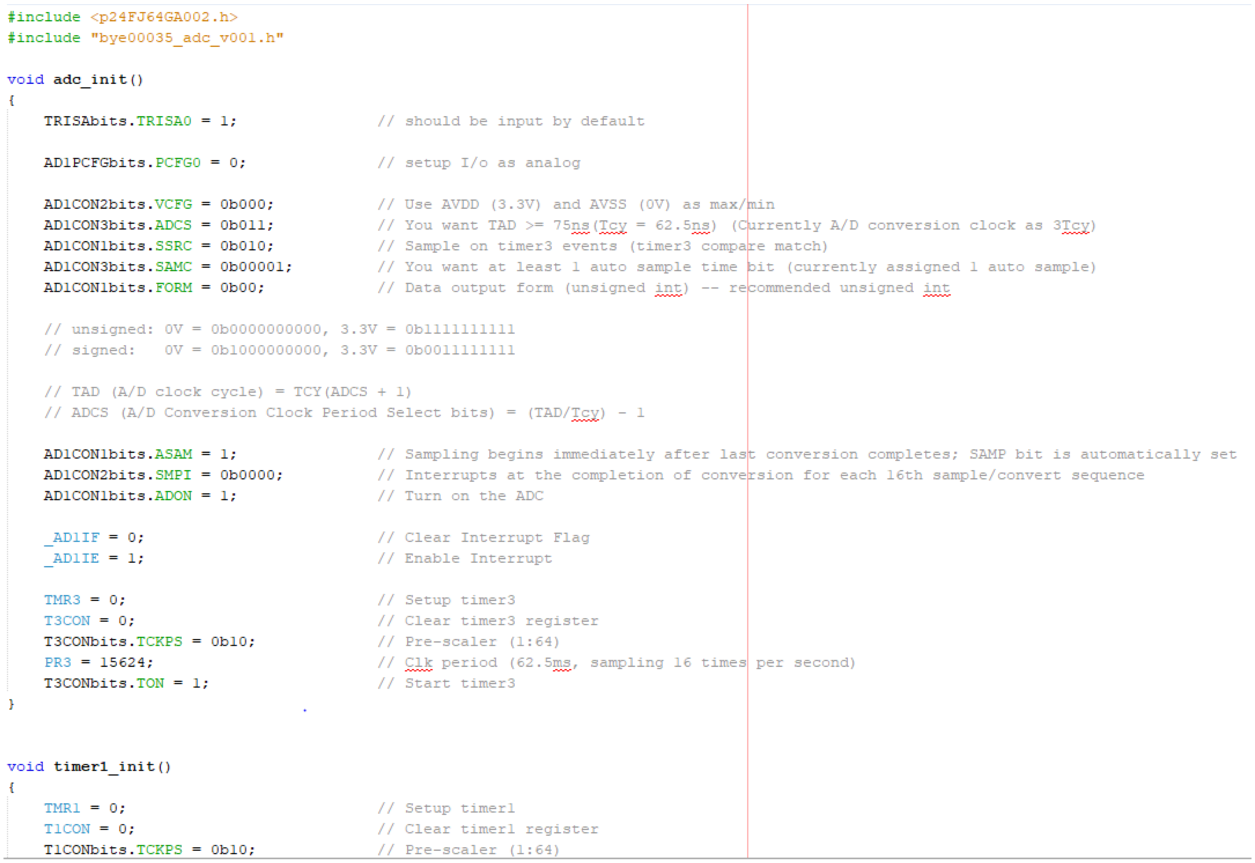


```
/*
 * File:   bye00035_circularBuffer_v001.h
 * Author: bye00035
 *
 * Created on April 7, 2023, 11:23 AM
 */

#ifndef BYE00035_CIRCULARBUFFER_V001_H
#define BYE00035_CIRCULARBUFFER_V001_H

#ifdef  __cplusplus
extern "C" {
#endif
    void initBuffer();
    void putVal(int ADCvalue);
    unsigned int getAvg();
#ifdef  __cplusplus
}
#endif

#endif  /* BYE00035_CIRCULARBUFFER_V001_H */
```

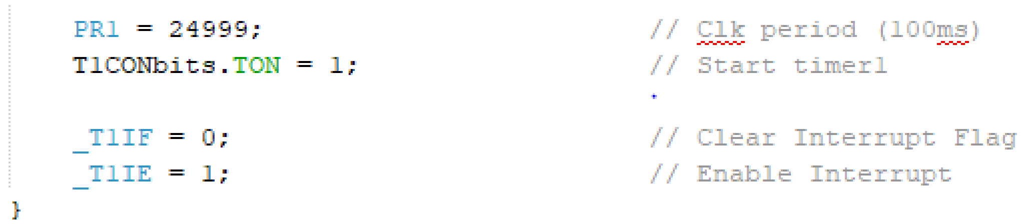
Library File



```
de "bye00035_circularBuffer_v001.h"

#define BUFSIZE 128
#define NUMSAMPLES 128

int adc_buffer[BUFSIZE];
int buffer_index = 0;

/* Set all buffer entries to 0 */
void initBuffer()
{
    for (int i=0; i < BUFSIZE; i++) {
        adc_buffer[i] = 0;
    }
}



void putVal(int ADCvalue)
{
    adc_buffer[buffer_index++] = ADCvalue;
    if (buffer_index >= BUFSIZE) {
        buffer_index = 0;
    }
}


unsigned int getAvg()
{
    unsigned long int sum = 0;
    for (int i=0; i < NUMSAMPLES - 1; i++) {
        sum += adc_buffer[i];
    }

    return sum/NUMSAMPLES;

}
```



```
#include <p24FJ64GA002.h>
#include "bye00035_adc_v001.h"

void adc_init()
{
    TRISAbits.TRISA0 = 1;               // should be input by default

    AD1PCFGbits.PCFG0 = 0;              // setup I/o as analog

    AD1CON2bits.VCFG = 0b000;           // Use AVDD (3.3V) and AVSS (0V) as max/min
    AD1CON3bits.ADCS = 0b011;           // You want TAD >= 75ns(Tcy = 62.5ns) (Currently A/D conversion clock as 3Tcy)
    AD1CON1bits.SSRC = 0b010;           // Sample on timer3 events (timer3 compare match)
    AD1CON3bits.SAMC = 0b00001;         // You want at least 1 auto sample time bit (currently assigned 1 auto sample)
    AD1CON1bits.FORM = 0b00;            // Data output form (unsigned int) -- recommended unsigned int

    // unsigned: 0V = 0b0000000000, 3.3V = 0b1111111111
    // signed:   0V = 0b1000000000, 3.3V = 0b0011111111

    // TAD (A/D clock cycle) = TCY(ADCS + 1)
    // ADCS (A/D Conversion Clock Period Select bits) = (TAD/Tcy) - 1

    AD1CON1bits.ASAM = 1;               // Sampling begins immediately after last conversion completes; SAMP bit is automatically set
    AD1CON2bits.SMPI = 0b0000;          // Interrupts at the completion of conversion for each 16th sample/convert sequence
    AD1CON1bits.ADON = 1;               // Turn on the ADC

    _AD1IF = 0;                         // Clear Interrupt Flag
    _AD1IE = 1;                         // Enable Interrupt

    TMR3 = 0;                           // Setup timer3
    T3CON = 0;                          // Clear timer3 register
    T3CONbits.TCKPS = 0b10;             // Pre-scaler (1:64)
    PR3 = 15624;                        // Clk period (62.5ms, sampling 16 times per second)
    T3CONbits.TON = 1;                  // Start timer3
}


void timer1_init()
{
    TMR1 = 0;                           // Setup timer1
    T1CON = 0;                          // Clear timer1 register
    T1CONbits.TCKPS = 0b10;             // Pre-scaler (1:64)
```



```
    PR1 = 24999;                        // Clk period (100ms)
    T1CONbits.TON = 1;                  // Start timer1

    _T1IF = 0;                          // Clear Interrupt Flag
    _T1IE = 1;                          // Enable Interrupt
}
```

Same LCD library from Lab 5