Lab 7 Part A

#include <avr/io.h>

#include <avr/interrupt.h>

void CLK\_32MHZ(void);

void ADC(void);

*uint16\_t* adc;

int main(void)

{

CLK\_32MHZ(); //call 32MHZ clock

ADC(); //initialize ADC system

//8 bit unsigned adc with 2.5 V as reference. v=(1/102)adc

while(1) {

ADCA\_CTRLA=ADC\_ENABLE\_bm | ADC\_CH0START\_bm;

while((ADCA\_CH0\_INTFLAGS & 0x01)!= 0x01);

adc=ADCA\_CH0\_RES;

ADCA\_CH0\_INTFLAGS=0x01;

}

return 0;

}

void ADC(void) {

ADCA\_REFCTRL=ADC\_REFSEL\_AREFB\_gc; //adc reference as PORTB aref. start scanning on channel 0

ADCA\_PRESCALER=ADC\_PRESCALER\_DIV512\_gc; //512 prescaler or adc clock

ADCA\_CTRLB=ADC\_RESOLUTION\_8BIT\_gc ; //unsigned mode, 8 bit resolution, no free run

PORTA\_DIRCLR= PIN0\_bm; //PA0 as input

ADCA\_CH0\_CTRL=ADC\_CH\_INPUTMODE\_SINGLEENDED\_gc; //single ended mode

ADCA\_CH0\_MUXCTRL=ADC\_CH\_MUXPOS\_PIN0\_gc; //mux control

ADCA\_CTRLA=ADC\_ENABLE\_bm | ADC\_CH0START\_bm;

}

void CLK\_32MHZ(void)

{

OSC\_CTRL=0x02; //select the 32Mhz osciliator

while ( ((OSC\_STATUS) & 0x02) != 0x02 ); //check if 32Mhz oscillator is stable

//if not stable. keep looping

CPU\_CCP= 0xD8; //write IOREG to CPU\_CCP to enable change

CLK\_CTRL= 0x01; //select the 32Mhz oscillator

CPU\_CCP= 0xD8; //write IOREG to CPU\_CCP to enable change

CLK\_PSCTRL= 0x00; //0x00 for the prescaler

}

Part B

#include <avr/io.h>

#include <avr/interrupt.h>

void CLK\_32MHZ(void);

void ADC(void);

void TIMER\_INIT(void);

*uint16\_t* adc;

double event\_timer = ((32000000\*(1/20000))/64);

int main(void)

{

CLK\_32MHZ(); //call 32MHZ clock

ADC(); //initialize ADC system

TIMER\_INIT(); //initialize timer system

//8 bit unsigned adc with 2.5 V as reference. v=(1/102)adc

while(1) {

ADCA\_CTRLA=ADC\_ENABLE\_bm | ADC\_CH0START\_bm;

while((ADCA\_CH0\_INTFLAGS & 0x01)!= 0x01);

adc=ADCA\_CH0\_RES;

ADCA\_CH0\_INTFLAGS=0x01;

}

return 0;

}

void ADC(void) {

ADCA\_REFCTRL=ADC\_REFSEL\_AREFB\_gc; //adc reference as PORTB aref. start scanning on channel 0

ADCA\_PRESCALER=ADC\_PRESCALER\_DIV512\_gc; //512 prescaler or adc clock

ADCA\_CTRLB=ADC\_RESOLUTION\_8BIT\_gc ; //unsigned mode, 8 bit resolution, no free run

PORTA\_DIRCLR= PIN0\_bm; //PA0 as input

ADCA\_CH0\_CTRL=ADC\_CH\_INPUTMODE\_SINGLEENDED\_gc; //single ended mode

ADCA\_CH0\_MUXCTRL=ADC\_CH\_MUXPOS\_PIN0\_gc; //mux control

ADCA\_CTRLA=ADC\_ENABLE\_bm | ADC\_CH0START\_bm;

ADCA\_EVCTRL=ADC\_SWEEP\_0\_gc | ADC\_EVSEL\_0123\_gc | ADC\_EVACT\_CH0\_gc; //only sweep channel 0, 0123 event as selected inputs,

// then furtuhur reduced down to use EVENT0 to

// trigger ADC CHANNEL0

}

void TIMER\_INIT(void) {

TCC0\_CNT=0x00; //set CNT to zero

TCC0\_PER=25; //timer per value to output 1760 Hz sine wave

TCC0\_CTRLA=TC\_CLKSEL\_DIV64\_gc; //

EVSYS\_CH0MUX=EVSYS\_CHMUX\_TCC0\_OVF\_gc; //set TCC0 OVF as the source for CH0 event

}

void CLK\_32MHZ(void)

{

OSC\_CTRL=0x02; //select the 32Mhz osciliator

while ( ((OSC\_STATUS) & 0x02) != 0x02 ); //check if 32Mhz oscillator is stable

//if not stable. keep looping

CPU\_CCP= 0xD8; //write IOREG to CPU\_CCP to enable change

CLK\_CTRL= 0x01; //select the 32Mhz oscillator

CPU\_CCP= 0xD8; //write IOREG to CPU\_CCP to enable change

CLK\_PSCTRL= 0x00; //0x00 for the prescaler

}

Part C

#include <avr/io.h>

#include <avr/interrupt.h>

void CLK\_32MHZ(void);

void ADC(void);

void TIMER\_INIT(void);

*uint16\_t* adc;

double BSELHIGH = (((4)\*((32000000/(16\*115200))-1))>>8); //bscale of -2

double BSEL= ((4)\*((32000000/(16\*115200))-1)); //bscale of -2

double event\_timer = ((32000000\*(1/20000))/64);

int main(void)

{

CLK\_32MHZ(); //call 32MHZ clock

ADC(); //initialize ADC system

TIMER\_INIT(); //initialize timer system

//8 bit unsigned adc with 2.5 V as reference. v=(1/102)adc

while(1) {

ADCA\_CTRLA=ADC\_ENABLE\_bm | ADC\_CH0START\_bm;

while((ADCA\_CH0\_INTFLAGS & 0x01)!= 0x01);

adc=ADCA\_CH0\_RES;

ADCA\_CH0\_INTFLAGS=0x01;

}

return 0;

}

void ADC(void) {

ADCA\_REFCTRL=ADC\_REFSEL\_AREFB\_gc; //adc reference as PORTB aref. start scanning on channel 0

ADCA\_PRESCALER=ADC\_PRESCALER\_DIV512\_gc; //512 prescaler or adc clock

ADCA\_CTRLB=ADC\_RESOLUTION\_8BIT\_gc ; //unsigned mode, 8 bit resolution, no free run

PORTA\_DIRCLR= PIN0\_bm; //PA0 as input

ADCA\_CH0\_CTRL=ADC\_CH\_INPUTMODE\_SINGLEENDED\_gc; //single ended mode

ADCA\_CH0\_MUXCTRL=ADC\_CH\_MUXPOS\_PIN0\_gc; //mux control

ADCA\_CTRLA=ADC\_ENABLE\_bm | ADC\_CH0START\_bm;

ADCA\_EVCTRL=ADC\_SWEEP\_0\_gc | ADC\_EVSEL\_0123\_gc | ADC\_EVACT\_CH0\_gc; //only sweep channel 0, 0123 event as selected inputs,

// then furtuhur reduced down to use EVENT0 to

// trigger ADC CHANNEL0

}

void TIMER\_INIT(void) {

TCC0\_CNT=0x00; //set CNT to zero

TCC0\_PER=25; //timer per value to output 1760 Hz sine wave

TCC0\_CTRLA=TC\_CLKSEL\_DIV64\_gc; //

EVSYS\_CH0MUX=EVSYS\_CHMUX\_TCC0\_OVF\_gc; //set TCC0 OVF as the source for CH0 event

}

void USARTD0\_init(void)

{

PORTD\_DIRSET=PIN3\_bm; //set transmitter as output

PORTD\_DIRCLR=PIN2\_bm; //set receiver as input

USARTD0\_CTRLB=0x18; //enable receiver and transmitter

USARTD0\_CTRLC= USART\_CHSIZE\_8BIT\_gc | USART\_CMODE\_ASYNCHRONOUS\_gc | USART\_PMODE\_DISABLED\_gc; //USART asynchronous, 8 data bit, no parity, 1 stop bit

USARTD0\_BAUDCTRLA= (*uint8\_t*) BSEL; //load lowest 8 bits of BSEL

USARTD0\_BAUDCTRLB= (((*uint8\_t*) BSELHIGH) | 0xE0); //load BSCALE and upper 4 bits of BSEL. bitwise OR them

PORTD\_OUTSET= PIN3\_bm; //set transit pin idle

}

void CLK\_32MHZ(void)

{

OSC\_CTRL=0x02; //select the 32Mhz osciliator

while ( ((OSC\_STATUS) & 0x02) != 0x02 ); //check if 32Mhz oscillator is stable

//if not stable. keep looping

CPU\_CCP= 0xD8; //write IOREG to CPU\_CCP to enable change

CLK\_CTRL= 0x01; //select the 32Mhz oscillator

CPU\_CCP= 0xD8; //write IOREG to CPU\_CCP to enable change

CLK\_PSCTRL= 0x00; //0x00 for the prescaler

}

Part C

#include <avr/io.h>

#include <avr/interrupt.h>

void CLK\_32MHZ(void);

void ADC(void);

void TIMER\_INIT(void);

void USARTD0\_init(void);

*uint16\_t* adc;

double BSELHIGH = (((4)\*((32000000/(16\*115200))-1))>>8); //bscale of -2

double BSEL= ((4)\*((32000000/(16\*115200))-1)); //bscale of -2

//double event\_timer = ((32000000\*(1/20000))/64); //PER value to trigger event0, which then trigger ADC channel 0 conversion

int main(void)

{

CLK\_32MHZ(); //call 32MHZ clock

ADC(); //initialize ADC system

TIMER\_INIT(); //initialize timer system

//8 bit unsigned adc with 2.5 V as reference. v=(1/102)adc

USARTD0\_init();

while(1) {

ADCA\_CTRLA=ADC\_ENABLE\_bm | ADC\_CH0START\_bm;

while((ADCA\_CH0\_INTFLAGS & 0x01)!= 0x01);

adc=ADCA\_CH0\_RES;

ADCA\_CH0\_INTFLAGS=0x01;

}

return 0;

}

void ADC(void) {

ADCA\_REFCTRL=ADC\_REFSEL\_AREFB\_gc; //adc reference as PORTB aref. start scanning on channel 0

ADCA\_PRESCALER=ADC\_PRESCALER\_DIV512\_gc; //512 prescaler or adc clock

ADCA\_CTRLB=ADC\_RESOLUTION\_8BIT\_gc ; //unsigned mode, 8 bit resolution, no free run

PORTA\_DIRCLR= PIN0\_bm; //PA0 as input

ADCA\_CH0\_CTRL=ADC\_CH\_INPUTMODE\_SINGLEENDED\_gc; //single ended mode

ADCA\_CH0\_MUXCTRL=ADC\_CH\_MUXPOS\_PIN0\_gc; //mux control

ADCA\_CTRLA=ADC\_ENABLE\_bm | ADC\_CH0START\_bm;

ADCA\_EVCTRL=ADC\_SWEEP\_0\_gc | ADC\_EVSEL\_0123\_gc | ADC\_EVACT\_CH0\_gc; //only sweep channel 0, 0123 event as selected inputs,

// then furtuhur reduced down to use EVENT0 to

// trigger ADC CHANNEL0

}

void TIMER\_INIT(void) {

TCC0\_CNT=0x00; //set CNT to zero

TCC0\_PER=25; //timer per value to output 1760 Hz sine wave

TCC0\_CTRLA=TC\_CLKSEL\_DIV64\_gc; //

EVSYS\_CH0MUX=EVSYS\_CHMUX\_TCC0\_OVF\_gc; //set TCC0 OVF as the source for CH0 event

}

void USARTD0\_init(void)

{

PORTD\_DIRSET=PIN3\_bm; //set transmitter as output

PORTD\_DIRCLR=PIN2\_bm; //set receiver as input

USARTD0\_CTRLB=0x18; //enable receiver and transmitter

USARTD0\_CTRLC= USART\_CHSIZE\_8BIT\_gc | USART\_CMODE\_ASYNCHRONOUS\_gc | USART\_PMODE\_DISABLED\_gc; //USART asynchronous, 8 data bit, no parity, 1 stop bit

USARTD0\_BAUDCTRLA= (*uint8\_t*) BSEL; //load lowest 8 bits of BSEL

USARTD0\_BAUDCTRLB= (((*uint8\_t*) BSELHIGH) | 0xE0); //load BSCALE and upper 4 bits of BSEL. bitwise OR them

PORTD\_OUTSET= PIN3\_bm; //set transit pin idle

}

void CLK\_32MHZ(void)

{

OSC\_CTRL=0x02; //select the 32Mhz osciliator

while ( ((OSC\_STATUS) & 0x02) != 0x02 ); //check if 32Mhz oscillator is stable

//if not stable. keep looping

CPU\_CCP= 0xD8; //write IOREG to CPU\_CCP to enable change

CLK\_CTRL= 0x01; //select the 32Mhz oscillator

CPU\_CCP= 0xD8; //write IOREG to CPU\_CCP to enable change

CLK\_PSCTRL= 0x00; //0x00 for the prescaler

}

Part D

#include <avr/io.h>

#include <avr/interrupt.h>

void CLK\_32MHZ(void);

void ADC(void);

void TIMER\_INIT(void);

void USARTD0\_init(void);

void DMA\_INIT(void);

*uint16\_t* adc;

double BSELHIGH = (((4)\*((32000000/(16\*115200))-1))>>8); //bscale of -2

double BSEL= ((4)\*((32000000/(16\*115200))-1)); //bscale of -2

//double event\_timer = ((32000000\*(1/20000))/64); //PER value to trigger event0, which then trigger ADC channel 0 conversion

int main(void)

{

CLK\_32MHZ(); //call 32MHZ clock

ADC(); //initialize ADC system

TIMER\_INIT(); //initialize timer system

//8 bit unsigned adc with 2.5 V as reference. v=(1/102)adc

USARTD0\_init();

DMA\_INIT();

while(1) {

;

}

/\*

while(1) {

ADCA\_CTRLA=ADC\_ENABLE\_bm | ADC\_CH0START\_bm;

while((ADCA\_CH0\_INTFLAGS & 0x01)!= 0x01);

adc=ADCA\_CH0\_RES;

ADCA\_CH0\_INTFLAGS=0x01;

}

\*/

return 0;

}

void DMA\_INIT(void) {

DMA\_CH0\_REPCNT=0x00; //repeat count of 0, which is unlimited repeat

DMA\_CH0\_CTRLA=DMA\_CH\_REPEAT\_bm | DMA\_CH\_SINGLE\_bm; //repeat mode, single shot data transfer

//burst mode defaults to 00=1 byte

DMA\_CH0\_ADDRCTRL=0b10001000; //source address and destination reloaded with initial value at end of each burst

//source and destination does not increment

DMA\_CH0\_TRIGSRC= ( (0x01) + (0x01) ); //trigger source for DMA as event channel 1

DMA\_CH0\_SRCADDR0= 0x10; //source address is ADCA\_CH0RES

DMA\_CH0\_SRCADDR1= 0x02;

DMA\_CH0\_SRCADDR2= 0x00;

DMA\_CH0\_DESTADDR0=0xA0; //destination address is USARTD0\_DATA

DMA\_CH0\_DESTADDR1=0x09;

DMA\_CH0\_DESTADDR2=0x00;

DMA\_CTRL=DMA\_ENABLE\_bm | DMA\_DBUFMODE\_DISABLED\_gc; //enable DMA and disable duffer buffer mode

DMA\_CH0\_CTRLA=DMA\_CH\_ENABLE\_bm; // enable channel 0 of DMC

}

void ADC(void) {

ADCA\_REFCTRL=ADC\_REFSEL\_AREFB\_gc; //adc reference as PORTB aref. start scanning on channel 0

ADCA\_PRESCALER=ADC\_PRESCALER\_DIV512\_gc; //512 prescaler or adc clock

ADCA\_CTRLB=ADC\_RESOLUTION\_8BIT\_gc ; //unsigned mode, 8 bit resolution, no free run

PORTA\_DIRCLR= PIN0\_bm; //PA0 as input

ADCA\_CH0\_CTRL=ADC\_CH\_INPUTMODE\_SINGLEENDED\_gc; //single ended mode

ADCA\_CH0\_MUXCTRL=ADC\_CH\_MUXPOS\_PIN0\_gc; //mux control

ADCA\_CTRLA=ADC\_ENABLE\_bm | ADC\_CH0START\_bm;

ADCA\_EVCTRL=ADC\_SWEEP\_0\_gc | ADC\_EVSEL\_0123\_gc | ADC\_EVACT\_CH0\_gc; //only sweep channel 0, 0123 event as selected inputs,

// then furtuhur reduced down to use EVENT0 to

// trigger ADC CHANNEL0

}

void TIMER\_INIT(void) {

TCC0\_CNT=0x00; //set CNT to zero

TCC0\_PER=25; //timer per value to output 1760 Hz sine wave

TCC0\_CTRLA=TC\_CLKSEL\_DIV64\_gc; //

EVSYS\_CH0MUX=EVSYS\_CHMUX\_TCC0\_OVF\_gc; //set TCC0 OVF as the source for CH0 event

EVSYS\_CH1MUX=EVSYS\_CHMUX\_ADCA\_CH0\_gc; //set ADCA CH0 conversion complete as source for CH1 event

}

void USARTD0\_init(void)

{

PORTD\_DIRSET=PIN3\_bm; //set transmitter as output

PORTD\_DIRCLR=PIN2\_bm; //set receiver as input

USARTD0\_CTRLB=0x18; //enable receiver and transmitter

USARTD0\_CTRLC= USART\_CHSIZE\_8BIT\_gc | USART\_CMODE\_ASYNCHRONOUS\_gc | USART\_PMODE\_DISABLED\_gc; //USART asynchronous, 8 data bit, no parity, 1 stop bit

USARTD0\_BAUDCTRLA= (*uint8\_t*) BSEL; //load lowest 8 bits of BSEL

USARTD0\_BAUDCTRLB= (((*uint8\_t*) BSELHIGH) | 0xE0); //load BSCALE and upper 4 bits of BSEL. bitwise OR them

PORTD\_OUTSET= PIN3\_bm; //set transit pin idle

}

void CLK\_32MHZ(void)

{

OSC\_CTRL=0x02; //select the 32Mhz osciliator

while ( ((OSC\_STATUS) & 0x02) != 0x02 ); //check if 32Mhz oscillator is stable

//if not stable. keep looping

CPU\_CCP= 0xD8; //write IOREG to CPU\_CCP to enable change

CLK\_CTRL= 0x01; //select the 32Mhz oscillator

CPU\_CCP= 0xD8; //write IOREG to CPU\_CCP to enable change

CLK\_PSCTRL= 0x00; //0x00 for the prescaler

}

Part E

#include <avr/io.h>

#include <avr/interrupt.h>

void CLK\_32MHZ(void);

void ADC(void);

void TIMER\_INIT(void);

void USARTD0\_init(void);

void DMA\_INIT(void);

void OUT\_CHAR(*uint8\_t* data);

*uint16\_t* adc;

double BSELHIGH = (((4)\*((32000000/(16\*115200))-1))>>8); //bscale of -2

double BSEL= ((4)\*((32000000/(16\*115200))-1)); //bscale of -2

//double event\_timer = ((32000000\*(1/20000))/64); //PER value to trigger event0, which then trigger ADC channel 0 conversion

int main(void)

{

CLK\_32MHZ(); //call 32MHZ clock

ADC(); //initialize ADC system

TIMER\_INIT(); //initialize timer system

//8 bit unsigned adc with 2.5 V as reference. v=(1/102)adc

USARTD0\_init();

DMA\_INIT();

PORTC\_DIRSET=0x55;

while(1) {

/\*

while((TCC0\_INTFLAGS & 0x01)!= 0x01);

TCC0\_CNT=0x00;

TCC0\_INTFLAGS=0x01;

TCC0\_CTRLA=TC\_CLKSEL\_OFF\_gc;

while((ADCA\_CH0\_INTFLAGS & 0x01)!= 0x01);

ADCA\_CH0\_INTFLAGS=0x01;

TCC0\_CTRLA=TC\_CLKSEL\_DIV64\_gc;

\*/

}

return 0;

}

void DMA\_INIT(void) {

DMA\_CTRL=DMA\_ENABLE\_bm | DMA\_DBUFMODE\_DISABLED\_gc; //enable DMA and disable duffer buffer mode

DMA\_CH0\_REPCNT=0x00; //repeat count of 0, which is unlimited repeat

DMA\_CH0\_ADDRCTRL=0b10001000; //source address and destination reloaded with initial value at end of each burst

//source and destination does not increment

DMA\_CH0\_TRIGSRC= DMA\_CH\_TRIGSRC\_ADCA\_CH0\_gc; //trigger source for DMA as event channel 1

DMA\_CH0\_SRCADDR0= (*uint8\_t*)&ADCA\_CH0\_RES; //source address is ADCA\_CH0RES

DMA\_CH0\_SRCADDR1= ((*uint16\_t*)&ADCA\_CH0\_RES) >> 8;

DMA\_CH0\_SRCADDR2= ((*uint32\_t*)&ADCA\_CH0\_RES) >> 16;

DMA\_CH0\_DESTADDR0=(*uint8\_t*)&USARTD0\_DATA; //destination address is USARTD0\_DATA

DMA\_CH0\_DESTADDR1=((*uint16\_t*)&USARTD0\_DATA) >> 8;

DMA\_CH0\_DESTADDR2=((*uint32\_t*)&USARTD0\_DATA) >> 16;

DMA\_CH0\_CTRLA=DMA\_CH\_ENABLE\_bm | DMA\_CH\_REPEAT\_bm | DMA\_CH\_SINGLE\_bm | DMA\_CH\_BURSTLEN\_1BYTE\_gc;

//repeat mode, single shot data transfer

//burst mode defaults to 00=1 byte

}

void ADC(void) {

ADCA\_REFCTRL=ADC\_REFSEL\_AREFB\_gc; //adc reference as PORTB aref. start scanning on channel 0

ADCA\_PRESCALER=ADC\_PRESCALER\_DIV512\_gc; //512 prescaler or adc clock

ADCA\_CTRLB=ADC\_RESOLUTION\_8BIT\_gc ; //unsigned mode, 8 bit resolution, no free run

PORTA\_DIRCLR= PIN0\_bm; //PA0 as input

ADCA\_CH0\_CTRL=ADC\_CH\_INPUTMODE\_SINGLEENDED\_gc; //single ended mode

ADCA\_CH0\_MUXCTRL=ADC\_CH\_MUXPOS\_PIN0\_gc; //mux control

ADCA\_CTRLA=ADC\_ENABLE\_bm | ADC\_CH0START\_bm;

ADCA\_EVCTRL=ADC\_SWEEP\_0\_gc | ADC\_EVSEL\_0123\_gc | ADC\_EVACT\_CH0\_gc; //only sweep channel 0, 0123 event as selected inputs,

// then furtuhur reduced down to use EVENT0 to

// trigger ADC CHANNEL0

}

void TIMER\_INIT(void) {

TCC0\_CNT=0x00; //set CNT to zero

TCC0\_PER=25; //timer per value to output 1760 Hz sine wave

TCC0\_CTRLA=TC\_CLKSEL\_DIV64\_gc; //

EVSYS\_CH0MUX=EVSYS\_CHMUX\_TCC0\_OVF\_gc; //set TCC0 OVF as the source for CH0 event

EVSYS\_CH1MUX=EVSYS\_CHMUX\_ADCA\_CH0\_gc; //set ADCA CH0 conversion complete as source for CH1 event

}

void USARTD0\_init(void)

{

PORTD\_DIRSET=PIN3\_bm; //set transmitter as output

PORTD\_DIRCLR=PIN2\_bm; //set receiver as input

USARTD0\_CTRLB=0x18; //enable receiver and transmitter

USARTD0\_CTRLC= USART\_CHSIZE\_8BIT\_gc | USART\_CMODE\_ASYNCHRONOUS\_gc | USART\_PMODE\_DISABLED\_gc; //USART asynchronous, 8 data bit, no parity, 1 stop bit

USARTD0\_BAUDCTRLA= (*uint8\_t*) BSEL; //load lowest 8 bits of BSEL

USARTD0\_BAUDCTRLB= (((*uint8\_t*) BSELHIGH) | 0xE0); //load BSCALE and upper 4 bits of BSEL. bitwise OR them

PORTD\_OUTSET= PIN3\_bm; //set transit pin idle

}

void CLK\_32MHZ(void)

{

OSC\_CTRL=0x02; //select the 32Mhz osciliator

while ( ((OSC\_STATUS) & 0x02) != 0x02 ); //check if 32Mhz oscillator is stable

//if not stable. keep looping

CPU\_CCP= 0xD8; //write IOREG to CPU\_CCP to enable change

CLK\_CTRL= 0x01; //select the 32Mhz oscillator

CPU\_CCP= 0xD8; //write IOREG to CPU\_CCP to enable change

CLK\_PSCTRL= 0x00; //0x00 for the prescaler

}

void OUT\_CHAR(*uint8\_t* data) {

while( ((USARTD0\_STATUS) & 0x20) != 0x20); //keep looping if DREIF flag is not set

USARTD0\_DATA= (*uint8\_t*) data;

}