Part A

.include "ATxmega128A1Udef.inc" ;include the file

.list ;list it

.org 0x0000 ;start our program here

rjmp MAIN ;jump to main

.equ stack\_init=0x3FFF ;initialize stack pointer

.equ BSELHIGH=(((8)\*((32000000/(16\*115200))-1))>>8)

.equ BSEL=((8)\*((32000000/(16\*115200))-1))

.org 0x100

MAIN:

ldi r23, 0x00 ;setting for 32MHZ subroutine

rcall CLK

ldi YL, low(stack\_init) ;Load 0xFF to YL

out CPU\_SPL, YL ;transfer to CPU\_SPL

ldi YL, high(stack\_init) ;Load 0x3F to YH

out CPU\_SPH, YL ;transfer to CPU\_SPH

ldi r17, 0x55 ;ASCII hex code for “U". Used in the OUT\_CHAR subroutine

rcall USART ;call subroutine to set up USART system

LOOP:

rcall OUT\_CHAR ;call OUT\_CHAR subroutine

rjmp LOOP ;infinite loop to output "U"

USART:

push r16 ;push r16

push r18

ldi r16, 0x08 ;load r16 with 0x08

sts PORTD\_DIRSET, r16 ;set receiver as output

ldi r16, 0x04 ;load r16 with 0x04

sts PORTD\_DIRCLR, r16 ;set transmitter as input

ldi r16, 0x18

sts USARTD0\_CTRLB, r16 ;enable receiver and transmitter

ldi r16, 0x33 ;USART asynchronous, 8 data bit, odd parity, 1 stop bit

sts USARTD0\_CTRLC, r16

ldi r16, low(BSEL) ;8 bit BSEL value

sts USARTD0\_BAUDCTRLA, r16

ldi r16, low(BSELHIGH)

ldi r18, 0xD0

or r16, r18 ;BSCALE of -3, ignoring highest 4 bit

sts USARTD0\_BAUDCTRLB, r16 ;load BAUDCTRLB with BSCALE and highest 4 bits of BSEL

pop r18

pop r16

ret

OUT\_CHAR: ;OUT\_CHAR subroutine

push r16

/\*

WAIT:

lds r16, USARTD0\_STATUS

bst r16, 6 ;check TXCIF (bit 6, TXCIF:Transmit Complete Interrupt Flag) to see if there is any ongoing transmission

brts COMPLETE

brtc WAIT

\*/

COMPLETE:

lds r16, USARTD0\_STATUS

bst r16, 5 ;check the DREIF (Data register empty flag)

brts LOAD

brtc COMPLETE

LOAD:

sts USARTD0\_DATA, r17 ;transit "U" to the data register

pop r16

ret

CLK: ;take in a r17 value for prescaler. 32MHZ = 0x00 for prescale

push r16 ;push r16

ldi r16, 0b00000010 ;bit 1 is the 32Mhz oscillator

sts OSC\_CTRL, r16 ;store r16 into the OSC\_CTRL

NSTABLE:

lds r16, OSC\_STATUS ;load oscillator status into r16

bst r16, 1 ;check if 32Mhz oscillator is stable

brts STABLE ;branch if stable

brtc NSTABLE ;loop again if non-stable

STABLE:

ldi r16, 0xD8 ;writing IOREG to r16

sts CPU\_CCP, r16 ;write IOREG to CPU\_CCP to enable change

ldi r16, 0b00000001 ;write this to r16. corresponds to 32Mhz oscillator

sts CLK\_CTRL, r16 ;select the 32Mhz oscillator

ldi r16, 0xD8 ;writing IOREG for prescaler

sts CPU\_CCP, r16 ;for prescaler

sts CLK\_PSCTRL, r23 ;r23 will be initialized outside the subroutine for prescale. 32/8=4MHZ

pop r16 ;pop r16

ret ;return to main routine

Part C

.include "ATxmega128A1Udef.inc" ;include the file

.list ;list it

.org 0x0000 ;start our program here

rjmp MAIN ;jump to main

.equ stack\_init=0x3FFF ;initialize stack pointer

.equ BSELHIGH=(((8)\*((32000000/(16\*115200))-1))>>8)

.equ BSEL=((8)\*((32000000/(16\*115200))-1))

.org 0x100

Table :.db 'P', 'e', 'n', 'g', 'z', 'h', 'a', 'o', ' ', 'Z', 'h', 'u', 0x00

.org 0x200

MAIN:

ldi r23, 0x00 ;setting for 32MHZ subroutine

rcall CLK

ldi YL, low(stack\_init) ;Load 0xFF to YL

out CPU\_SPL, YL ;transfer to CPU\_SPL

ldi YL, high(stack\_init) ;Load 0x3F to YH

out CPU\_SPH, YL ;transfer to CPU\_SPH

ldi ZL, low(Table << 1) ;load lower byte of table address

ldi ZH, high(Table << 1) ;load higher byte of table address

rcall USART ;call subroutine to set up USART system

rcall OUT\_STRING

DONE:

rjmp DONE

USART:

push r16 ;push r16

push r18

ldi r16, 0x08 ;load r16 with 0x08

sts PORTD\_DIRSET, r16 ;set receiver as output

ldi r16, 0x04 ;load r16 with 0x04

sts PORTD\_DIRCLR, r16 ;set transmitter as input

ldi r16, 0x18

sts USARTD0\_CTRLB, r16 ;enable receiver and transmitter

ldi r16, 0x33 ;USART asynchronous, 8 data bit, odd parity, 1 stop bit

sts USARTD0\_CTRLC, r16

ldi r16, low(BSEL) ;8 bit BSEL value

sts USARTD0\_BAUDCTRLA, r16

ldi r16, low(BSELHIGH)

ldi r18, 0xD0

or r16, r18 ;BSCALE of -3, ignoring highest 4 bit

sts USARTD0\_BAUDCTRLB, r16 ;load BAUDCTRLB with BSCALE and highest 4 bits of BSEL

pop r18

pop r16

ret

OUT\_CHAR: ;OUT\_CHAR subroutine

push r16

/\*

WAIT:

lds r16, USARTD0\_STATUS

bst r16, 6 ;check TXCIF (bit 6, TXCIF:Transmit Complete Interrupt Flag) to see if there is any ongoing transmission

brts COMPLETE

brtc WAIT

\*/

COMPLETE:

lds r16, USARTD0\_STATUS ;check the DREIF (Data register empty flag)

bst r16, 5

brts LOAD

brtc COMPLETE

LOAD:

sts USARTD0\_DATA, r17 ;transit "U" to the data register

pop r16

ret

OUT\_STRING:

push r17

CONTINUE:

elpm r17, Z+ ;load value in Z to r16. Post increment Z

cpi r17,0 ;check if the value is the null character

breq RETURN ;if it is the null character. prepare to return from subroutine

rcall OUT\_CHAR ;call OUT\_CHAR subroutine

rjmp CONTINUE ;Loop until the null character has been detected.

RETURN:

pop r17

ret

CLK: ;take in a r17 value for prescaler. 32MHZ = 0x00 for prescale

push r16 ;push r16

ldi r16, 0b00000010 ;bit 1 is the 32Mhz oscillator

sts OSC\_CTRL, r16 ;store r16 into the OSC\_CTRL

NSTABLE:

lds r16, OSC\_STATUS ;load oscillator status into r16

bst r16, 1 ;check if 32Mhz oscillator is stable

brts STABLE ;branch if stable

brtc NSTABLE ;loop again if non-stable

STABLE:

ldi r16, 0xD8 ;writing IOREG to r16

sts CPU\_CCP, r16 ;write IOREG to CPU\_CCP to enable change

ldi r16, 0b00000001 ;write this to r16. corresponds to 32Mhz oscillator

sts CLK\_CTRL, r16 ;select the 32Mhz oscillator

ldi r16, 0xD8 ;writing IOREG for prescaler

sts CPU\_CCP, r16 ;for prescaler

sts CLK\_PSCTRL, r23 ;r23 will be initialized outside the subroutine for prescale. 32/8=4MHZ

pop r16 ;pop r16

ret ;return to main routine

Part D

.include "ATxmega128A1Udef.inc" ;include the file

.list ;list it

.org 0x0000 ;start our program here

rjmp MAIN ;jump to main

.equ stack\_init=0x3FFF ;initialize stack pointer

.equ BSELHIGH=(((8)\*((32000000/(16\*115200))-1))>>8)

.equ BSEL=((8)\*((32000000/(16\*115200))-1))

.org 0x100

Table :.db 'P', 'e', 'n', 'g', 'z', 'h', 'a', 'o', ' ', 'Z', 'h', 'u', 0x00

.org 0x200

MAIN:

ldi r23, 0x00 ;setting for 32MHZ subroutine

rcall CLK

ldi YL, low(stack\_init) ;Load 0xFF to YL

out CPU\_SPL, YL ;transfer to CPU\_SPL

ldi YL, high(stack\_init) ;Load 0x3F to YH

out CPU\_SPH, YL ;transfer to CPU\_SPH

ldi ZL, low(Table << 1) ;load lower byte of table address

ldi ZH, high(Table << 1) ;load higher byte of table address

rcall USART ;call subroutine to set up USART system

rcall IN\_CHAR

LOOP:

rcall OUT\_CHAR

rjmp LOOP

USART:

push r16 ;push r16

push r18

ldi r16, 0x08 ;load r16 with 0x08

sts PORTD\_DIRSET, r16 ;set receiver as output

ldi r16, 0x04 ;load r16 with 0x04

sts PORTD\_DIRCLR, r16 ;set transmitter as input

ldi r16, 0x18

sts USARTD0\_CTRLB, r16 ;enable receiver and transmitter

ldi r16, 0x33 ;USART asynchronous, 8 data bit, odd parity, 1 stop bit

sts USARTD0\_CTRLC, r16

ldi r16, low(BSEL) ;8 bit BSEL value

sts USARTD0\_BAUDCTRLA, r16

ldi r16, low(BSELHIGH)

ldi r18, 0xD0

or r16, r18 ;BSCALE of -3, ignoring highest 4 bit

sts USARTD0\_BAUDCTRLB, r16 ;load BAUDCTRLB with BSCALE and highest 4 bits of BSEL

pop r18

pop r16

ret

OUT\_CHAR: ;OUT\_CHAR subroutine

push r16

/\*

WAIT:

lds r16, USARTD0\_STATUS

bst r16, 6 ;check TXCIF (bit 6, TXCIF:Transmit Complete Interrupt Flag) to see if there is any ongoing transmission

brts COMPLETE

brtc WAIT

\*/

COMPLETE:

lds r16, USARTD0\_STATUS ;check the DREIF (Data register empty flag)

bst r16, 5

brts LOAD

brtc COMPLETE

LOAD:

sts USARTD0\_DATA, r17 ;transmit information typed on keypad

pop r16

ret

OUT\_STRING:

push r17

CONTINUE:

elpm r17, Z+ ;load value in Z to r16. Post increment Z

cpi r17,0 ;check if the value is the null character

breq RETURN ;if it is the null character. prepare to return from subroutine

rcall OUT\_CHAR ;call OUT\_CHAR subroutine

rjmp CONTINUE ;Loop until the null character has been detected.

RETURN:

pop r17

ret

IN\_CHAR:

push r16

NOT:

lds r16, USARTD0\_STATUS

bst r16, 7

brts RECEIVE

brtc NOT

RECEIVE:

lds r17, USARTD0\_DATA

pop r16

ret

CLK: ;take in a r17 value for prescaler. 32MHZ = 0x00 for prescale

push r16 ;push r16

ldi r16, 0b00000010 ;bit 1 is the 32Mhz oscillator

sts OSC\_CTRL, r16 ;store r16 into the OSC\_CTRL

NSTABLE:

lds r16, OSC\_STATUS ;load oscillator status into r16

bst r16, 1 ;check if 32Mhz oscillator is stable

brts STABLE ;branch if stable

brtc NSTABLE ;loop again if non-stable

STABLE:

ldi r16, 0xD8 ;writing IOREG to r16

sts CPU\_CCP, r16 ;write IOREG to CPU\_CCP to enable change

ldi r16, 0b00000001 ;write this to r16. corresponds to 32Mhz oscillator

sts CLK\_CTRL, r16 ;select the 32Mhz oscillator

ldi r16, 0xD8 ;writing IOREG for prescaler

sts CPU\_CCP, r16 ;for prescaler

sts CLK\_PSCTRL, r23 ;r23 will be initialized outside the subroutine for prescale. 32/8=4MHZ

pop r16 ;pop r16

ret ;return to main routine

Part E

.include "ATxmega128A1Udef.inc" ;include the file

.list ;list it

.org 0x0000 ;start our program here

rjmp MAIN ;jump to main

.equ stack\_init=0x3FFF ;initialize stack pointer

.equ BSELHIGH=(((8)\*((32000000/(16\*115200))-1))>>8)

.equ BSEL=((8)\*((32000000/(16\*115200))-1))

.equ toggle\_timer= (32000000\*.25)/1024

.org USARTF0\_RXC\_vect

rjmp USART\_ISR

.org 0x100

Table :.db 'P', 'e', 'n', 'g', 'z', 'h', 'a', 'o', ' ', 'Z', 'h', 'u', 0x00

.org 0x200

MAIN:

ldi r23, 0x00 ;setting for 32MHZ subroutine

rcall CLK

ldi YL, low(stack\_init) ;Load 0xFF to YL

out CPU\_SPL, YL ;transfer to CPU\_SPL

ldi YL, high(stack\_init) ;Load 0x3F to YH

out CPU\_SPH, YL ;transfer to CPU\_SPH

ldi ZL, low(Table << 1) ;load lower byte of table address

ldi ZH, high(Table << 1) ;load higher byte of table address

rcall USART ;call subroutine to set up USART system

ldi r16, 0x20 ;load r16 with 0x20

sts PORTD\_DIRSET, r16 ;set GREEN LED as output

ldi r16, low(toggle\_timer) ;set the timer for 5ms to debounce

sts TCD0\_PER, r16 ;need to load low and high byte of PER

ldi r16, high(toggle\_timer)

sts TCD0\_PER+1, r16

ldi r16, 0b00000111 ;Timer clock for clk/1024

sts TCD0\_CTRLA, r16

ldi r16, 0x00 ;setting the CNT back to zero

sts TCD0\_CNT, r16

LOOP:

lds r16, TCD0\_INTFLAGS ;check the intflag

sbrs r16, 0

rjmp LOOP

rjmp TOGGLE ;if intflag set, rjmp to toggle. in other work, if CNT reaches per

TOGGLE:

ldi r16, 0x20 ;use to toggle GREEN LED

sts PORTD\_OUTTGL, r16

ldi r16, 0x00 ;setting the CNT back to zero

sts TCD0\_CNT, r16 ;resetting the timer CNT value

ldi r16, 0x01

sts TCD0\_INTFLAGS, r16 ;clears the interrupt flag

rjmp LOOP ;back to LOOP

USART:

push r16 ;push r16

push r18

ldi r16, 0x08 ;load r16 with 0x08

sts PORTD\_DIRSET, r16 ;set receiver as output

ldi r16, 0x04 ;load r16 with 0x04

sts PORTD\_DIRCLR, r16 ;set transmitter as input

ldi r16, 0x18

sts USARTD0\_CTRLB, r16 ;enable receiver and transmitter

ldi r16, 0x33 ;USART asynchronous, 8 data bit, odd parity, 1 stop bit

sts USARTD0\_CTRLC, r16

ldi r16, low(BSEL) ;8 bit BSEL value

sts USARTD0\_BAUDCTRLA, r16

ldi r16, low(BSELHIGH)

ldi r18, 0xD0

or r16, r18 ;BSCALE of -3, ignoring highest 4 bit

sts USARTD0\_BAUDCTRLB, r16 ;load BAUDCTRLB with BSCALE and highest 4 bits of BSEL

ldi r16, 0x10

sts USARTD0\_CTRLA, r16 ;enable low level interrupt for "receive complete"

ldi r16, 0x01

sts PMIC\_CTRL, r16 ;enable low level interrupt in the PMIC

sei

pop r18

pop r16

ret

USART\_ISR:

push r18

lds r18, CPU\_SREG

push r18

push r18

lds r18, USARTD0\_DATA ;read the information in the receive buffer

WAIT:

lds r16, USARTD0\_STATUS ;check the DREIF (Data register empty flag)

bst r16, 5

brts TRANSMIT

brtc WAIT

TRANSMIT:

sts USARTD0\_DATA, r18 ;transmit received character out

ldi r16, 0x80

sts USARTD0\_STATUS, r16 ;clear the receive complete interrupt flag

pop r16

pop r18

sts CPU\_SREG, r18

pop r18

reti

OUT\_CHAR: ;OUT\_CHAR subroutine

push r16

/\*

WAIT:

lds r16, USARTD0\_STATUS

bst r16, 6 ;check TXCIF (bit 6, TXCIF:Transmit Complete Interrupt Flag) to see if there is any ongoing transmission

brts COMPLETE

brtc WAIT

\*/

KEEPCHECK:

lds r16, USARTD0\_STATUS ;check the DREIF (Data register empty flag)

bst r16, 5

brts LOAD

brtc KEEPCHECK

LOAD:

sts USARTD0\_DATA, r17 ;transmit information typed on keypad

pop r16

ret

OUT\_STRING:

push r17

CONTINUE:

elpm r17, Z+ ;load value in Z to r16. Post increment Z

cpi r17,0 ;check if the value is the null character

breq RETURN ;if it is the null character. prepare to return from subroutine

rcall OUT\_CHAR ;call OUT\_CHAR subroutine

rjmp CONTINUE ;Loop until the null character has been detected.

RETURN:

pop r17

ret

IN\_CHAR:

push r16

NOT:

lds r16, USARTD0\_STATUS

bst r16, 7

brts RECEIVE

brtc NOT

RECEIVE:

lds r17, USARTD0\_DATA

pop r16

ret

CLK: ;take in a r17 value for prescaler. 32MHZ = 0x00 for prescale

push r16 ;push r16

ldi r16, 0b00000010 ;bit 1 is the 32Mhz oscillator

sts OSC\_CTRL, r16 ;store r16 into the OSC\_CTRL

NSTABLE:

lds r16, OSC\_STATUS ;load oscillator status into r16

bst r16, 1 ;check if 32Mhz oscillator is stable

brts STABLE ;branch if stable

brtc NSTABLE ;loop again if non-stable

STABLE:

ldi r16, 0xD8 ;writing IOREG to r16

sts CPU\_CCP, r16 ;write IOREG to CPU\_CCP to enable change

ldi r16, 0b00000001 ;write this to r16. corresponds to 32Mhz oscillator

sts CLK\_CTRL, r16 ;select the 32Mhz oscillator

ldi r16, 0xD8 ;writing IOREG for prescaler

sts CPU\_CCP, r16 ;for prescaler

sts CLK\_PSCTRL, r23 ;r23 will be initialized outside the subroutine for prescale. 32/8=4MHZ

pop r16 ;pop r16

ret ;return to main routine

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_