Name: Junzhe Wu NetID: junzhew3

Q4:

1. Sending eight bits of data per character with the required start and stop bits will take up 10 times the bit time for ach byte sent. At 9600 baud, the bit time is about 1/9600 microseconds which makes each character sent take 10/9600 milliseconds.
2. X = 4
3. X = 13
4. X = 941
5. X = 59
6. X = 832
7. X = 64

Q5:

char, int, float, double, void

Q6:

int: 32 bits

long int: 40 bits

unsigned int: 32 bits

short int: 16 bits

char: 8 bits

double: 64 bits

float: 32 bits

Q7:

int: 16 bits

long int: 32 bits

unsigned int: 16 bits

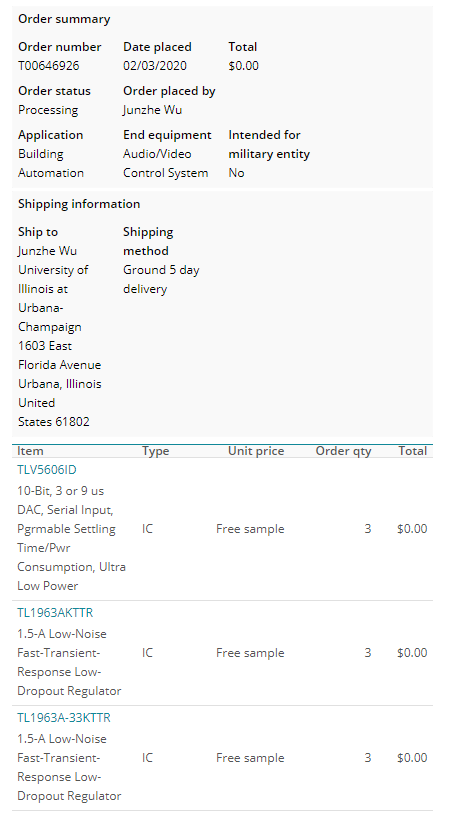
short int: 16 bits

char: 8 bits

double: 32 bits

float: 32 bits

Q8:



Q9:

**#include** "msp430g2553.h"

**#include** "UART.h"

**void** **print\_every**(**int** rate);

**char** **get\_switchstate**(**void**);

**char** newprint = 0;

**long** NumOn = 0;

**long** NumOff = 0;

**int** statevar = 1;

**int** timecheck = 0;

**int** x = 0;

**int** state = 0;

**void** **main**(**void**) {

WDTCTL = WDTPW + WDTHOLD; // Stop WDT

**if** (CALBC1\_16MHZ ==0xFF || CALDCO\_16MHZ == 0xFF) **while**(1);

DCOCTL = CALDCO\_16MHZ; // Set uC to run at approximately 16 Mhz

BCSCTL1 = CALBC1\_16MHZ;

// Initialize Port 1

P1SEL &= ~0x01;

P1SEL2 &= ~0x01;

P2SEL &= ~0xc0;

P2SEL2 &= ~0xc0;

P1REN = 0x0; // No resistors enabled for Port 1

P1DIR |= 0xf0; // Set P1,4,P1.5, P1.6, and P1.7 to output

P2DIR &= ~0xc0; //Set P2.6 and P2.7 to input

P2REN |= 0xc0; // P2.6 and P2.7 Resistor enabled

P2OUT |= 0xc0; // P2.6 and P2.7 Pullup Resistor selected

// Timer A Config

TACCTL0 = CCIE; // Enable Periodic interrupt

TACCR0 = 16000; // period = 1ms

TACTL = TASSEL\_2 + MC\_1; // source SMCLK, up mode

**Init\_UART**(115200,1);

\_BIS\_SR(GIE); // Enable global interrupt

/\*

x = 495 & ( 0x5 << 4);

UART\_printf("%d\n\r",x);

\*/

**while**(1) { // Low priority Slow computation items go inside this while loop. Very few (if anyt) items in the HWs will go inside this while loop

// for use if you want to use a method of receiving a string of chars over the UART see USCI0RX\_ISR below

// if(newmsg) {

// newmsg = 0;

// }

// The newprint variable is set to 1 inside the function "print\_every(rate)" at the given rate

**if** ( (newprint == 1) && (senddone == 1) ) { // senddone is set to 1 after UART transmission is complete

// only one UART\_printf can be called every 15ms

//UART\_printf("St%d On %ld Off %ld\n\r",statevar,NumOn,NumOff);

**UART\_printf**("State: %d \n\r",statevar);

newprint = 0;

}

}

}

// Timer A0 interrupt service routine

//Q9

**#pragma** vector=TIMER0\_A0\_VECTOR

**\_\_interrupt** **void** **Timer\_A** (**void**)

{

timecheck++; // Keep track of time for main while loop.

print\_every(500);

**switch** (statevar) {

**case** 1: //LED1 ON

P1OUT = 0x10;

**if** (timecheck == 500) {

timecheck = 0;

statevar = 2; // Next Timer\_A call go to state 2

} **else**

**break**;

**case** 2: //LED2 ON

P1OUT = 0x20;

**if** (timecheck == 500) {

timecheck = 0;

statevar = 3;

} **else**

**break**;

**case** 3: //LED3 ON

P1OUT = 0x40;

**if** (timecheck == 500) {

timecheck = 0;

statevar = 4;

} **else**

**break**;

**case** 4: //LED4 ON

P1OUT = 0x80;

**if** (timecheck == 500) {

timecheck = 0;

statevar = 5;

} **else**

**break**;

**case** 5: //LED3 ON

P1OUT = 0x40;

**if** (timecheck == 500) {

timecheck = 0;

statevar = 6;

}

**break**;

**case** 6: //LED2 ON

P1OUT = 0x20;

**if** (timecheck == 500) {

timecheck = 0;

statevar = 1;

} **else**

**break**;

}

}

/\*

// ADC 10 ISR - Called when a sequence of conversions (A7-A0) have completed

#pragma vector=ADC10\_VECTOR

\_\_interrupt void ADC10\_ISR(void) {

}

\*/

// USCI Transmit ISR - Called when TXBUF is empty (ready to accept another character)

**#pragma** vector=USCIAB0TX\_VECTOR

**\_\_interrupt** **void** **USCI0TX\_ISR**(**void**) {

**if**(IFG2&UCA0TXIFG) { // USCI\_A0 requested TX interrupt

**if**(printf\_flag) {

**if** (currentindex == txcount) {

senddone = 1;

printf\_flag = 0;

IFG2 &= ~UCA0TXIFG;

} **else** {

UCA0TXBUF = printbuff[currentindex];

currentindex++;

}

} **else** **if**(UART\_flag) {

**if**(!donesending) {

UCA0TXBUF = txbuff[txindex];

**if**(txbuff[txindex] == 255) {

donesending = 1;

txindex = 0;

}

**else** txindex++;

}

} **else** { // interrupt after sendchar call so just set senddone flag since only one char is sent

senddone = 1;

}

IFG2 &= ~UCA0TXIFG;

}

**if**(IFG2&UCB0TXIFG) { // USCI\_B0 requested TX interrupt (UCB0TXBUF is empty)

IFG2 &= ~UCB0TXIFG; // clear IFG

}

}

// USCI Receive ISR - Called when shift register has been transferred to RXBUF

// Indicates completion of TX/RX operation

**#pragma** vector=USCIAB0RX\_VECTOR

**\_\_interrupt** **void** **USCI0RX\_ISR**(**void**) {

**if**(IFG2&UCB0RXIFG) { // USCI\_B0 requested RX interrupt (UCB0RXBUF is full)

IFG2 &= ~UCB0RXIFG; // clear IFG

}

**if**(IFG2&UCA0RXIFG) { // USCI\_A0 requested RX interrupt (UCA0RXBUF is full)

// Uncomment this block of code if you would like to use this COM protocol that uses 253 as STARTCHAR and 255 as STOPCHAR

/\* if(!started) { // Haven't started a message yet

if(UCA0RXBUF == 253) {

started = 1;

newmsg = 0;

}

}

else { // In process of receiving a message

if((UCA0RXBUF != 255) && (msgindex < (MAX\_NUM\_FLOATS\*5))) {

rxbuff[msgindex] = UCA0RXBUF;

msgindex++;

} else { // Stop char received or too much data received

if(UCA0RXBUF == 255) { // Message completed

newmsg = 1;

rxbuff[msgindex] = 255; // "Null"-terminate the array

}

started = 0;

msgindex = 0;

}

}

\*/

IFG2 &= ~UCA0RXIFG;

}

}

// This function takes care of all the timing for printing to UART

// Rate determined by how often the function is called in Timer ISR

**int** print\_timecheck = 0;

**void** **print\_every**(**int** rate) {

**if** (rate < 15) {

rate = 15;

}

**if** (rate > 10000) {

rate = 10000;

}

print\_timecheck++;

**if** (print\_timecheck == rate) {

print\_timecheck = 0;

newprint = 1;

}

}

**char** **get\_switchstate**(**void**){

**char** state1 = 1;

**char** P6 = 0;

**char** P7 = 0;

P6 = 0x40 & P2IN;

P7 = 0x80 & P2IN;

**if** ((P6 == 64) && (P7 == 128)){

state1 = 0;

}

**if** ((P6 == 0) && (P7 == 128)){

state1 = 1;

}

**if** ((P6 == 64) && (P7 == 0)){

state1 = 2;

}

**if** ((P6 == 0) && (P7 == 0)){

state1 = 3;

}

**return** state1;

}

Q10:

**#include** "msp430g2553.h"

**#include** "UART.h"

**void** **print\_every**(**int** rate);

**char** **get\_switchstate**(**void**);

**char** newprint = 0;

**long** NumOn = 0;

**long** NumOff = 0;

**int** statevar = 1;

**int** timecheck = 0;

**int** x = 0;

**int** state = 0;

**void** **main**(**void**) {

WDTCTL = WDTPW + WDTHOLD; // Stop WDT

**if** (CALBC1\_16MHZ ==0xFF || CALDCO\_16MHZ == 0xFF) **while**(1);

DCOCTL = CALDCO\_16MHZ; // Set uC to run at approximately 16 Mhz

BCSCTL1 = CALBC1\_16MHZ;

// Initialize Port 1

P1SEL &= ~0x01;

P1SEL2 &= ~0x01;

P2SEL &= ~0xc0;

P2SEL2 &= ~0xc0;

P1REN = 0x0; // No resistors enabled for Port 1

P1DIR |= 0xf0; // Set P1,4,P1.5, P1.6, and P1.7 to output

P2DIR &= ~0xc0; //Set P2.6 and P2.7 to input

P2REN |= 0xc0; // P2.6 and P2.7 Resistor enabled

P2OUT |= 0xc0; // P2.6 and P2.7 Pullup Resistor selected

// Timer A Config

TACCTL0 = CCIE; // Enable Periodic interrupt

TACCR0 = 16000; // period = 1ms

TACTL = TASSEL\_2 + MC\_1; // source SMCLK, up mode

**Init\_UART**(115200,1);

\_BIS\_SR(GIE); // Enable global interrupt

/\*

x = 495 & ( 0x5 << 4);

UART\_printf("%d\n\r",x);

\*/

**while**(1) { // Low priority Slow computation items go inside this while loop. Very few (if anyt) items in the HWs will go inside this while loop

// for use if you want to use a method of receiving a string of chars over the UART see USCI0RX\_ISR below

// if(newmsg) {

// newmsg = 0;

// }

// The newprint variable is set to 1 inside the function "print\_every(rate)" at the given rate

**if** ( (newprint == 1) && (senddone == 1) ) { // senddone is set to 1 after UART transmission is complete

// only one UART\_printf can be called every 15ms

//UART\_printf("St%d On %ld Off %ld\n\r",statevar,NumOn,NumOff);

**UART\_printf**("State: %d \n\r",state);

newprint = 0;

}

}

}

//Q10

**#pragma** vector=TIMER0\_A0\_VECTOR

**\_\_interrupt** **void** **Timer\_A** (**void**)

{

timecheck++; // Keep track of time for main while loop.

print\_every(500); // units determined by the rate Timer\_A ISR is called, print every "rate" calls to this function

state = get\_switchstate();

**switch** (state) {

**case** 0: //LED1 Blinking

**if** (timecheck == 1) {

P1OUT = 0x10;

}

**else** **if** (timecheck == 500) {

P1OUT = 0x00;

}

**else** **if** (timecheck == 1000) {

timecheck = 0;

}

**break**;

**case** 1: //LED2 Blinking

**if** (timecheck == 1) {

P1OUT = 0x20;

}

**else** **if** (timecheck == 500) {

P1OUT = 0x00;

}

**else** **if** (timecheck == 1000) {

timecheck = 0;

}

**break**;

**case** 2: //LED3 Blinking

**if** (timecheck == 1) {

P1OUT = 0x40;

}

**else** **if** (timecheck == 500) {

P1OUT = 0x00;

}

**else** **if** (timecheck == 1000) {

timecheck = 0;

}

**break**;

**case** 3: //LED4 Blinking

**if** (timecheck == 1) {

P1OUT = 0x80;

}

**else** **if** (timecheck == 500) {

P1OUT = 0x00;

}

**else** **if** (timecheck == 1000) {

timecheck = 0;

}

**break**;

}

}

/\*

// ADC 10 ISR - Called when a sequence of conversions (A7-A0) have completed

#pragma vector=ADC10\_VECTOR

\_\_interrupt void ADC10\_ISR(void) {

}

\*/

// USCI Transmit ISR - Called when TXBUF is empty (ready to accept another character)

**#pragma** vector=USCIAB0TX\_VECTOR

**\_\_interrupt** **void** **USCI0TX\_ISR**(**void**) {

**if**(IFG2&UCA0TXIFG) { // USCI\_A0 requested TX interrupt

**if**(printf\_flag) {

**if** (currentindex == txcount) {

senddone = 1;

printf\_flag = 0;

IFG2 &= ~UCA0TXIFG;

} **else** {

UCA0TXBUF = printbuff[currentindex];

currentindex++;

}

} **else** **if**(UART\_flag) {

**if**(!donesending) {

UCA0TXBUF = txbuff[txindex];

**if**(txbuff[txindex] == 255) {

donesending = 1;

txindex = 0;

}

**else** txindex++;

}

} **else** { // interrupt after sendchar call so just set senddone flag since only one char is sent

senddone = 1;

}

IFG2 &= ~UCA0TXIFG;

}

**if**(IFG2&UCB0TXIFG) { // USCI\_B0 requested TX interrupt (UCB0TXBUF is empty)

IFG2 &= ~UCB0TXIFG; // clear IFG

}

}

// USCI Receive ISR - Called when shift register has been transferred to RXBUF

// Indicates completion of TX/RX operation

**#pragma** vector=USCIAB0RX\_VECTOR

**\_\_interrupt** **void** **USCI0RX\_ISR**(**void**) {

**if**(IFG2&UCB0RXIFG) { // USCI\_B0 requested RX interrupt (UCB0RXBUF is full)

IFG2 &= ~UCB0RXIFG; // clear IFG

}

**if**(IFG2&UCA0RXIFG) { // USCI\_A0 requested RX interrupt (UCA0RXBUF is full)

// Uncomment this block of code if you would like to use this COM protocol that uses 253 as STARTCHAR and 255 as STOPCHAR

/\* if(!started) { // Haven't started a message yet

if(UCA0RXBUF == 253) {

started = 1;

newmsg = 0;

}

}

else { // In process of receiving a message

if((UCA0RXBUF != 255) && (msgindex < (MAX\_NUM\_FLOATS\*5))) {

rxbuff[msgindex] = UCA0RXBUF;

msgindex++;

} else { // Stop char received or too much data received

if(UCA0RXBUF == 255) { // Message completed

newmsg = 1;

rxbuff[msgindex] = 255; // "Null"-terminate the array

}

started = 0;

msgindex = 0;

}

}

\*/

IFG2 &= ~UCA0RXIFG;

}

}

// This function takes care of all the timing for printing to UART

// Rate determined by how often the function is called in Timer ISR

**int** print\_timecheck = 0;

**void** **print\_every**(**int** rate) {

**if** (rate < 15) {

rate = 15;

}

**if** (rate > 10000) {

rate = 10000;

}

print\_timecheck++;

**if** (print\_timecheck == rate) {

print\_timecheck = 0;

newprint = 1;

}

}

**char** **get\_switchstate**(**void**){

**char** state1 = 1;

**char** P6 = 0;

**char** P7 = 0;

P6 = 0x40 & P2IN;

P7 = 0x80 & P2IN;

**if** ((P6 == 64) && (P7 == 128)){

state1 = 0;

}

**if** ((P6 == 0) && (P7 == 128)){

state1 = 1;

}

**if** ((P6 == 64) && (P7 == 0)){

state1 = 2;

}

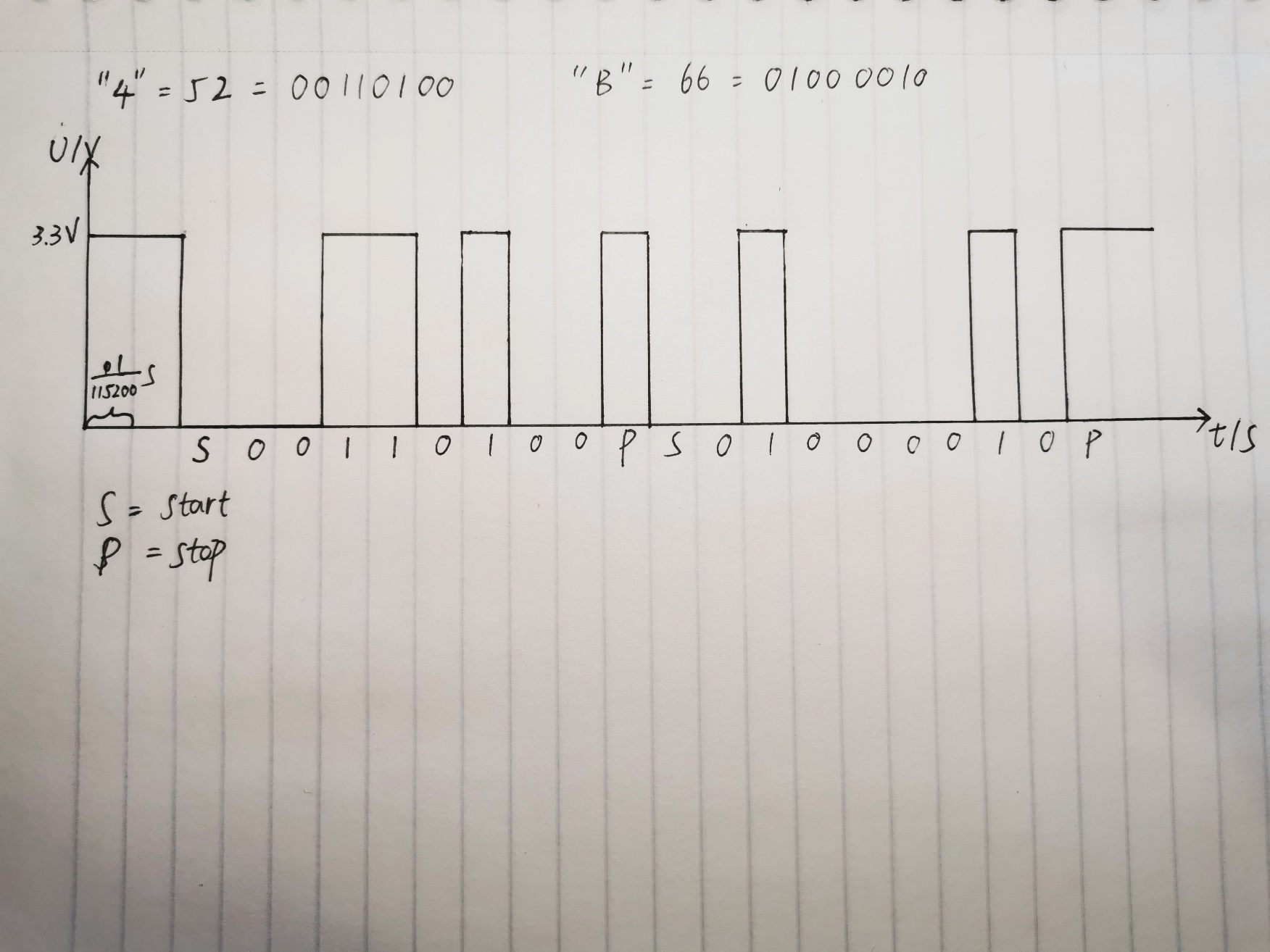
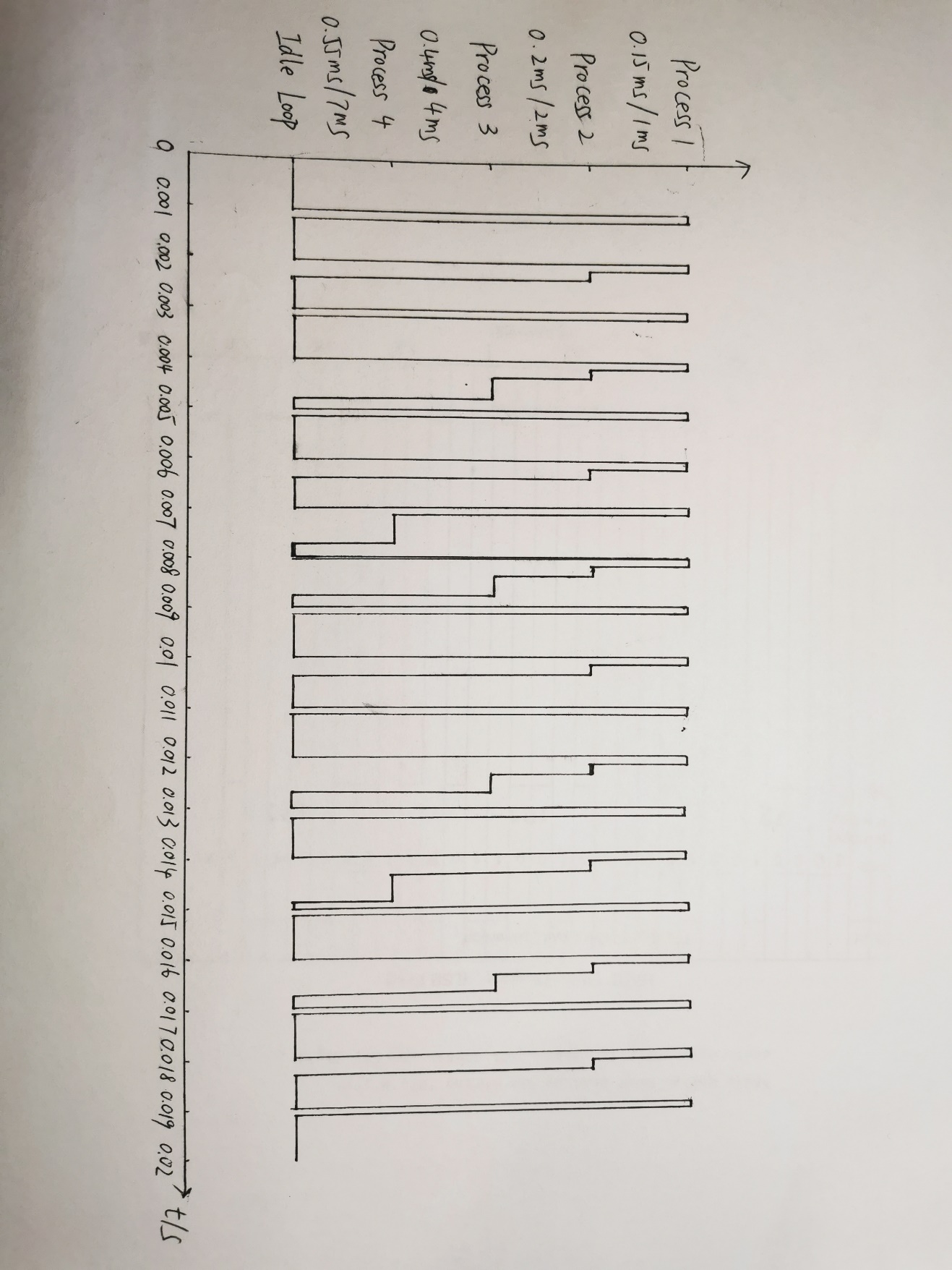
**if** ((P6 == 0) && (P7 == 0)){

state1 = 3;

}

**return** state1;

}

Q11:

Q12:

Q13:

