**2019 마이크로프로세서 중간 대비 답안지**

1. 128, 256은 프로세서의 메모리의 크기를 의미한다.
2. DDR은 Data Direction Register로 데이터에 입/출력을 설정하는 레지스터이다.

DDRL은 L포트의 입/출력 설정으로 0xff 🡪 출력이 설정된 상태이다.

PORTL은 LED를 담당하는 포트로 8개의 LED를 각 비트로 나타내어 설정한다.

LED는 0인경우 on, 1인경우 off를 의미한다.

이 경우는 전체 LED를 off한 상태이다.

1. UBRR0H, UBRR0L은 Baud Rate Register로써 전송속도를 설정하는 것이다.

상위, 하위로 나눠서 설정한다.

TXEN0는 Transmit Enable을 의미하고, RXEN0는 Receive Enable을 의미한다.

TXCIE는 Transmit Complete Interrupt Enable을 의미한다.

따라서 전송 및 수신이 가능한 상황이며 전송완료시 인터럽트가 발생한다.

// uart.h

void uart\_init();

void uart\_putchar(char ch);

// uart.c

#include <avr/io.h>

#include <avr/interrupt.h>

#include <compat/deprecated.h>

#include <util/delay.h>

#include "uart.h"

#include “queue.h”

int volatile txbusy = 0;

void uart\_putchar(char ch)

{

if (ch == '\n')

uart\_putchar('\r');

cli();

if (!txbusy) {

UDR0 = ch, txbusy = 1;

}

else {

while(qo\_insert(ch) == 0) {

sei();

\_delay\_ms(100);

cli();

}

}

sei();

}

ISR(USART0\_TX\_vect)

{

char ch;

if (!(ch = qo\_delete())

txbusy = 0;

else

UDR0 = ch;

}

void insert\_node\_ascn(struct node \*np)

{

struct node \*cp, \*pp;

if(!Head) {

Head = np;

np->link = NULL;

}

else {

for (cp=Head, pp=NULL; cp != NULL && cp->data < np->data;

pp = cp, cp = cp->link) {

if (pp == NULL) {

np->link = Head;

Head = np;

}

else {

np->link = pp->link;

pp->link = np;

}

}

}

Memory Mapped IO

메모리 주소 영역에 입출력 장치를 대응시켜, load/store 등 메모리 접근 명령에 의해 입출력을 수행한다.

Isolated IO

IO 장치를 식별하기 위한 주소가 메모리 영역과 분리되어, in/out 등 별도의 입출력 명령어에 의해 입출력을 수행한다.

// uart.c

#include <stdio.h>

#include <avr/io.h>

#include <avr/interrupt.h>

#include <compat/deprecated.h>

#include <util/delay.h> // for sbi(), \_delay\_ms()

#include "uart.h"

#include "queue.h"

int uart\_putchar(char ch, FILE \*stream);

FILE Mystdout = FDEV\_SETUP\_STREAM (uart\_putchar, NULL, \_FDEV\_SETUP\_WRITE);

char volatile uart\_busy; // for always access memory

// uart\_busy가 int이던 char이던 상관없음.

void uart\_init()

{

stdout = &Mystdout;

uart\_busy = 0;

q\_init();

UBRR0H = 0x00; UBRR0L = 0x07; // 115.2K

sbi(UCSR0A, U2X0);

sbi(UCSR0B, TXEN0);

sbi(UCSR0B, TXCIE0);

sbi(UCSR0B, RXEN0);

sbi(UCSR0B, RXCIE0);

}

ISR(USART0\_TX\_vect)

{

char ch;

if((ch = qo\_delete()) == 0){

uart\_busy = 0;

}

else

UDR0 = ch;

}

ISR(USART0\_RX\_vect)

{

char ch;

ch = UDR0;

qi\_insert(ch);

}

int uart\_putchar(char ch, FILE \*stream)

{

if(ch == '\n') uart\_putchar('\r', stream);

cli();

if(!uart\_busy){

UDR0 = ch;

uart\_busy = 1;

}

else{

while(qo\_insert(ch) == 0){

sei();

\_delay\_ms(100);

cli();

}

}

sei();

return(1);

}

// uart.h

int uart\_putchar(char ch, FILE \*stream);

void uart\_init(void);

// queue.h

void q\_init();

int qi\_insert(char ch);

int qi\_delete();

int qo\_insert(char ch);

int qo\_delete();

// queue.c

#include "queue.h"

#define QI\_SIZE 128

#define QO\_SIZE 128

static char qi[QI\_SIZE], qo[QO\_SIZE];

static int fi, ri, fo, ro;

void q\_init()

{

fi = ri = fo = ro = 0;

}

int qi\_insert(char ch)

{

if((ri+1) % QI\_SIZE == fi)

return(0); //full

ri = (ri+1) % QI\_SIZE;

qi[ri] = ch;

return(1);

}

int qi\_delete()

{

if(ri == fi)

return(0); //empty

fi = (fi+1) % QI\_SIZE;

return(qi[fi]);

}

int qo\_insert(char ch)

{

if((ro+1) % QO\_SIZE == fo)

return(0); //full

ro = (ro+1) % QO\_SIZE;

qo[ro] = ch;

return(1);

}

int qo\_delete()

{

if(ro == fo)

return(0); //empty

fo = (fo+1) % QO\_SIZE;

return(qo[fo]);

}

// main.c

#include <stdio.h> // for printf(), fputs

#include <avr/interrupt.h> // for sei(), cli()

#include <util/delay.h> // for \_delay\_ms()

#include "uart.h" // for uart\_init()

#include "queue.h" // for qi\_delete()

int main()

{

char cmd[128], ch;

int n = 0, i;

uart\_init();

sei();

printf("$ ");

while(1){

cli();

ch = qi\_delete();

sei();

if(ch){

if(ch == '\r') continue;

if(ch == '\n'){

cmd[n] = 0;

for( i=0; i<n; i++) {

printf("%c", cmd[i]);

}

\_delay\_ms(3000);

n = 0;

printf("\n$ ");

}

else

cmd[n++] = ch;

}

}

}