Project #8(Pjt08_uart_stdin_cir_queue)

UART0 RX(input queue)

```
char qi[QI_SIZE], qo[QO_SIZE];
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]);
}
```

Project #8(Pjt08_uart_stdin_cir_queue)

- UART0 RX(output queue)

```
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]);
}
```

Project #8(Pjt08_uart_stdin_cir_queue)

UART0 RX/TX initialization

```
#include <stdio.h>
#include <avr/io.h>
#include <compat/deprecated.h>
int  uart_putchar(char ch, FILE *stream), uart_getchar(FILE *stream);

FILE Mystdout = FDEV_SETUP_STREAM(uart_putchar, NULL,_FDEV_SETUP_WRITE);
FILE Mystdin  = FDEV_SETUP_STREAM(NULL, uart_getchar,_FDEV_SETUP_READ);
uchar  uart_busy;

void  uart_init()
{
    stdin = &Mystdin; stdout = &Mystdout;
    q_init();
    uart_busy = 0;                  // false

    UBRR0H = 0x00; UBRR0L = 0x07; // 115.2Kbps
    sbi(UCSR0A, U2X0);              // 115.2Kbps
    sbi(UCSR0B, TXEN0);             // TX enable
    sbi(UCSR0B, TXCIE0);            // TX complete interrupt enable
    sbi(UCSR0B, RXEN0);             // RX enable
    sbi(UCSR0B, RXCIE0);            // RX complete interrupt enable
}
```

Project #8(Pjt08_uart_stdin_cir_queue)

- Setup Console Device Driver with UART0(Input)

```
#include <stdio.h>
#include <avr/io.h>

#define ETX   0x04   /* ^D : End of Text */

int  uart_getchar(FILE *stream)
{
    char  ch;

    do {
         cli();
         ch = qi_delete();
         sei();
    } while (ch == 0);

    if (ch == ETX) return(-1);
    else           return(ch);
}
```

```
#include <avr/interrupt.h>

ISR(USART0_RX_vect)
{
    char  ch;

    ch = UDR0;
    if (ch != ETX) {
        if (ch == '₩r')
            ch = '₩n';
        uart_echo(ch);
    }
    qi_insert(ch);
}
```

TeraTerm
RX:<CR>
TX:<CR>

```
void  uart_echo(char ch)
{
    if (ch == '₩n') uart_echo('₩r');

    if (!uart_busy) {
        UDR0 = ch;
        uart_busy = 1;
    }
    else
        qo_insert(ch);
}
```

Project #8(Pjt08_uart_stdin_cir_queue)

◈ Setup Console Device Driver with UART0(Output)

```
#include <stdio.h>
#include <avr/io.h>
int  uart_putchar(char ch, FILE *stream)
{
     char    ch;

     if (ch == '₩n') uart_putchar('₩r', stream);

     cli();
     if (!uart_busy) {
         UDR0 = ch;
         uart_busy = 1;
     }
     else {
          while(qo_insert(ch) == 0) {
                 sei();
                 _delay_us(100);
                 cli();
          }
     }
     sei();

     return(1);
}
```

```
#include <avr/interrupt.h>
ISR(USART0_TX_vect)
{
      char   ch;

      if ((ch = qo_delete()) == 0)
           uart_busy = 0;
      else
           UDR0 = ch;
}
```

Project #8(Pjt08_uart_stdin_cir_queue)

```
#include <stdio.h>
#include <string.h>

main()
{
    char   cmd[128], *cp;
    int    n = 0;

    uart_init();
    sei();
    while(1) {
        printf( “$ ” );
        if (fgets(cmd, sizeof(cmd), stdin) == NULL)
            break;
        if ((cp = strtok(cmd,  “₩n₩r₩t  “)) == NULL) continue
        if (!strcmp(cmd,  “prime” )) app_prime(2000);
        else                          printf( “Unknown command...₩n” );
    }
    printf( “logout, good bye !!!₩n” );
    while(1);
 }
```

Project #8(Pjt08_uart_stdin_cir_queue)

```
int is_prime(int  n)
{
  int  i;
  for (i = 2; i <= n/2; i++)
     if ((n % i) == 0)
        return(0);
  return(1);
}
app_prime(int  t)
{
  int  n, count = 0;
  for (n = 2; n <= t; n++) {
     if  (is_prime(n)) {
       count++;
       printf("%d  is a prime number !!!₩n", n);
     }
  }
  printf("count=%d₩n", count);
}
```