Project #11(Pjt11_timer_task)

- Linked List for Software Callback Timer

```
struct timer {  // node for timer
     int    time;
     struct task     task;
     struct timer   *link;
};
```

```
ISR(TIMER2_OVF_vect)
{
    if (!Thead)
        return;
    if (--Thead->time == 0)
        timer_expire();
}
```

```
void timer_init() {
     TCNT2 = 0;                              // Initialize   Timer/Counter2
     sbi(ASSR, AS2);                         // Asynchronous Timer/Counter2
     sbi(TIMSK2, TOIE2);                     // Timer2 Overflow Int. Enable
     sbi(TCCR2B, CS20); sbi(TCCR2B, CS21); // 32KHz/32 prescaling, Start
}
```

Project #11(Pjt11_timer_task)

- Linked List for Software Callback Timer

```
void
timer_expire(void)
{
   struct timer *tp;

   for( ; Thead != NULL && !Thead->time; ) {
      tp = Thead, Thead = tp->link;

      task_insert(&tp->task);

      free(tp);
   }
}
```

Project #11(Pjt11_timer_task)

Event-driven Task Scheduling-Task Queue

```
#define MAX_TASK  16

struct  task {
    void   (*fun)(void *);
    char    arg[8];
};

struct task  Task_q[MAX_TASK];
volatile int Task_f, Task_r;

void task_init()
{
    Task_f = Task_r = 0;
}
```

```
int task_insert(struct task *tskp)
{
    if ((Task_r + 1) % MAX_TASK == Task_f)
        return(0);    // full
    Task_r = (Task_r + 1) % MAX_TASK;
    Task_q[Task_r] = *tskp;
    return(1);
}

int task_delete(struct task *tskp)
{
    if (Task_r == Task_f)
        return(0);     // empty
     Task_f = (Task_f + 1) % MAX_TASK;
     *tskp = Task_q[Task_f];
     return(1);
}
```

ex) $ prime 2000 ↵
ex) $ timer 3000 prime 2000 ↵

Project #11(Pjt11_timer_task)

Event-driven Task Scheduling-Command Process Task

```
void task_cmd(void *arg)
{
    char  buf[64], *cp0, *cp1, *cp2, *cp3;
    struct task  task;

    if (gets(buf) == NULL)
       { printf( "$ "); return; }

    cp0 = strtok(buf,  " WtWnWr" );
    cp1 = strtok(NULL, " WtWnWr" );
    cp2 = strtok(NULL, " WtWnWr" );
    cp3 = strtok(NULL, " WtWnWr" );

    if (cp0 == NULL) {
        printf( "!!!-111Wn" );
        printf( "$ "); return;
    }
    if (!strcmp(cp0,  "prime" ))
        task_prime(cp1);
    else if (!strcmp(cp0,  "timer" )) {
        if (cp1 == NULL) {
            printf( "!!!-222Wn" );
            printf("$ ");
            return;
        }
        ms = atoi(cp1) / 256;
        if (!strcmp(cp2,  "prime" )) {
            task.fun = task_prime;
            if (cp3) strcpy(task.arg, cp3);
            else     strcpy(task.arg,  "" );
            cli();
            insert_timer(&task, ms);
            sei();
        }
        else printf( "!!!-333Wn" );
    }
    else printf( "!!!-444Wn" );

    printf("$ ");
}
```

Project #11(Pjt11_timer_task)

```
void task_cmd(void *arg);

ISR(USART0_RX_vect)
{
    struct task task;
    char  ch;
    :
    :
    qi_insert(ch);
    if (ch == ETX || ch == 'Wn') {
        task.fun = task_cmd;
        strcpy(task.arg, "");
        task_insert(&task);
    }
}
```

Project #11(Pjt11_timer_task)

- Event-driven Task Scheduling-CPU Scheduler

```
main()
{
    int     tag;
    struct  task  task;

    uart_init();
    task_init();
    timer_init();

    printf("$ ");

    while(1) {
        cli();
        tag = task_delete(&task);
        sei();
        if (tag)
            (*(task.fun))(task.arg);
    }
}
```

Project #11(Pjt11_timer_task)

```
int is_prime(int  n)
{
  int  i;
  for (i = 2; i <= n/2; i++)
     if ((n % i) == 0)
        return(0);
  return(1);
}
void task_prime(char  *ap)
{
  int  n = 2000, count = 0;

  if (ap && *ap) n = atoi(ap);
  for (n = 2; n <= t; n++) {
     if  (is_prime(n)) {
       count++;
       printf("%d  is a prime number !!!₩n", n);
     }
  }
  printf("count=%d₩n", count);
}
```