Time in C

```
/*
 * DAT103_lister.c
 *
 * Created: 11.03.2015 08:56:19
 *  Author: geirj
 */


// include files
#include<avr/io.h>
#include<avr/interrupt.h>
#include<stdint.h>
#include<util/atomic.h>
#include<stdlib.h>
#include <assert.h>

// defines
#define F_CPU 4000000   // Define of CPU speed
#define tikk  F_CPU/6400
// Define of how long the timer/counter, counts in one second
//(NB! I have sat it to give tikk every 10 ms for simulation), with prescaler sat to 64.

// Global typedef
typedef struct
{
    volatile uint8_t timer;
    volatile uint8_t min;
    volatile uint8_t sek;
    volatile uint8_t updated;
}   tid_type;



typedef struct
{
    tid_type finish_tid;
    uint8_t loper_nr;
}   loper_type;

typedef struct
{
    volatile tid_type loper_tid;
```

```
    volatile uint8_t loper_nr;
    volatile uint8_t nyloper;
}  ny_event;

//Typedef of node, node is the basic element in the dynamic list,
//and it is a struck of loper_type
typedef struct loper_type_d node;


struct loper_type_d
{
    tid_type finish_tid;
    uint8_t loper_nr;
    node* prior;
    node* next;
};



// Global  variable
tid_type tid = { 0,0,0,0};

ny_event nyEvent;

loper_type loper[255];
loper_type * loper_p[255];
node * head_dyn_list;
node * point_dyn_list;



//prototype of local functions.
static void incTid(tid_type *time_p);
static void finish(ny_event *nyEvent_p);
static void initTimer(void);
static void initPort(void);
static void handelNyEvent(ny_event *nyEvent_p);
static void stopTimer_1(void);
static void soterEtterNummer(void);
static void soterStaticList(void);
static void soterStaticList_p(void);
static void soterDynamicList(void);

// to emulate a run
static void initTimer3(void);
```

```
/*****************************
* Interrupt serves ruiner
* Using timer3 to emulate a test løp!
*  send the runners out with 2 sec between them
*****************************/
ISR(TIMER3_COMPA_vect)
{

    static uint8_t loper=1;
    static uint8_t nummerOfLopere=10;
    PORTE = loper * 100;                      // set loper nummer on portE

    if(PORTD & (1 << 4))                      // test if bit 4 on Port D  is high
    {
       PORTD = ~(1 << 4);                    // sett PortD bit 4 low

       if(loper==nummerOfLopere)             // test if all runners has started?
       {
            PORTD = ~(1 << 0); // last runners has past the finis line, give stop signal

       }
        loper++;                             // Inc runners nummer
    }
    else
    {
        PORTD = 0xff;                        // sett portd bit 4 high
    }

}


/***********************************
* Interrupt serves ruiner for timer 1
***********************************/
ISR(TIMER1_COMPA_vect)
{
    incTid(&tid);
// call incTid with address to tid struct on compare match ( this is a pointer to tid)
}


ISR(TIMER1_CAPT_vect)
{
    finish(&nyEvent);  // Call finish on capture event with address to nyEvent struct.
```

```
}

// Locale functions
/*****************************************
*  InitTimer3
*****************************************
*  In this function all timers that is used are setup
*  and Global interrupt is enabled.
*****************************************/
static void initTimer3(void)
{
    TCCR3B = (1 << WGM12 );                  // Configure timer 1 for CTC mode
    TCCR3B|= ((1 << CS10 ) | (1 << CS11 )) ; // Set up timer at Fcpu /64
    TCNT3  = 0;                              // Sett Timer Counter 1 = 0,
    OCR3A  = tikk;                           // Output Compare Register  A
    ETIMSK  = (1 << OCIE3A );                // Enabler interrupt on  OCR1A match
    //sei();                                 // Enabler  global interrupt
}
/*****************************************
*  InitTimer
*****************************************
*  In this function all timers that is used are setup
*  and Global interrupt is enabled.
*****************************************/
static void initTimer(void)
{
    TCCR1B = (1 << WGM12 );                 // Configure timer 1 for CTC mode
    TCCR1B|= ((1 << CS10 ) | (1 << CS11 ) | ( 1 << ICNC1)) ;
    // Set up timer at Fcpu /64 and capture on PortD bit 4
    TCNT1  = 0;                             // Sett Timer Counter 1 = 0,
    OCR1A  = tikk;                          // Output Compare Register  A
    TIMSK  = (1 << OCIE1A | 1 << TICIE1);   // Enabler interrupt on  OCR1A match
    sei();                                  // Enabler  global interrupt
}

/*****************************************
*  StopTimer_1
*****************************************
*  In this  timer 1 is stopped
*****************************************/
static void stopTimer_1(void)
{
    TCCR1B&= ~((1 << CS10 ) | (1 << CS11 )) ; // Set stop time

    TCCR3B&= ~((1 << CS10 ) | (1 << CS11 )) ; // Set stop time
}
```

```
/***************************************
*  InitPort
***************************************
*  In this function all Ports are setup
***************************************/
static void initPort(void)
{
    DDRE = 0xff;
    // sett as output to be able to emulate a test run from timer3  //0;
    // port E to input from start/finish
    DDRD = 0xff;
    // sett as output to be able to emulate a test run from timer3 //0;
    // port D to input  bit 4 from start/finish, and bit 0 from start/stop
    PORTD = 0xff;

}

/*************************************
*  incTid:
*************************************
*    This function increment the time in a tide_type struck with 1 second.
*
* in:  *tide_p :  A pointer to the tide_type struckt that shall be incremented.
* out:  void   :  This function has no output variable!
*************************************/
static void incTid(tid_type *tide_p)
{
    if (tide_p -> sek <59)
         (tide_p -> sek)++;
    else
    {
        (tide_p -> sek) = 0;
        if (tide_p -> min < 59)
            (tide_p -> min)++;
        else
        {
            (tide_p -> min) = 0;
            if (tide_p -> timer < 23)
                 (tide_p -> timer)++;
            else
                 (tide_p -> timer) = 0;
        }
    }
    tide_p ->updated = 1;
}
```

```
/*************************************
*  finish:
*************************************
*    This function update the time and loper nummer in ny_loper_p .
*    And it is called from the ICR interrupt.
*
* in: * ny_loper_p:  A pointer to the ny_event struckt that shall be updated.
* out:  void   :  This function has no output variable!
*************************************/
static void finish(ny_event * ny_loper_p)
{
    ny_loper_p ->loper_tid.min = tid.min;
    ny_loper_p ->loper_tid.sek = tid.sek;
    ny_loper_p ->loper_tid.timer  = tid.timer;
    ny_loper_p ->loper_nr = PINE;
    if(ny_loper_p->loper_nr !=0)
    ny_loper_p->nyloper = 1;
}
```

```
/*************************************************************************************
*  Function handelNyEvent:
*************************************************************************************
* This function Putt the info from the nyEvent into the loper[] ,
 loper_p[] and the dynamic list.
*
*************************************************************************************/
static void handelNyEvent(ny_event *nyEvent)
{

    static uint8_t nummer = 0;
    loper_type * temp;
    node * tempD;

    // add nyEvent to statistic list
        {
            loper[nummer].finish_tid.min = nyEvent->loper_tid.min;
            loper[nummer].finish_tid.sek = nyEvent->loper_tid.sek;
            loper[nummer].finish_tid.timer= nyEvent->loper_tid.timer;
            loper[nummer].loper_nr = nyEvent->loper_nr;
            //nummer++;
        }
```

```
    // add nyEvent to statistic list med pointers
      {
            temp=malloc(sizeof(loper_type));
            if(temp == NULL)
            {
                //noe er galt
                assert(0);  // stops the program
            }
            temp->finish_tid.min = nyEvent->loper_tid.min;
            temp->finish_tid.sek = nyEvent->loper_tid.sek;
            temp->finish_tid.timer= nyEvent->loper_tid.timer;
            temp->loper_nr = nyEvent->loper_nr;
            loper_p[nummer]=temp;
            //nummer++;
       }

        // add nyEvent to dynamick list
         {
             tempD=( node*)malloc(sizeof( node));
             if(tempD == NULL)
             {
                 //noe er galt
                 assert(0);  // stops the program
             }
             tempD->finish_tid.min = nyEvent->loper_tid.min;
             tempD->finish_tid.sek = nyEvent->loper_tid.sek;
             tempD->finish_tid.timer= nyEvent->loper_tid.timer;
             tempD->loper_nr = nyEvent->loper_nr;
             tempD->prior=point_dyn_list;
             point_dyn_list->next=tempD;
             point_dyn_list=tempD;
             nummer++;
         }
}

/*****************************************************************************************
*  Function soterEtterNummer:
*****************************************************************************************
*         This function reorder the runners after nummer.
*
*****************************************************************************************/
static void soterEtterNummer(void)
{

    soterStaticList();
```

```
    soterStaticList_p();
    soterDynamicList();

}


/*****************************************************************************************
*  Function soterEtterNummer:
*****************************************************************************************
*         This function reorder the runners after nummer.
*
*****************************************************************************************/
static void soterStaticList(void)
{


    loper_type Nyloper[255];
    uint8_t i;
    uint8_t nyPlassToPlace = 0;
    uint8_t minLoperNummer = 254;
    uint8_t minLoperNummerI;
    while(nyPlassToPlace <=253)
    {
        for(i=0; i <= 254; i++)         //  What had been wrong with writing < 255 ?
        {

           if(loper[i].loper_nr != 0)
           if(loper[i].loper_nr < minLoperNummer)
           {
              minLoperNummer =loper[i].loper_nr;
              minLoperNummerI = i;
           }
        }
        Nyloper[nyPlassToPlace].finish_tid.min=loper[minLoperNummerI].finish_tid.min;
        Nyloper[nyPlassToPlace].finish_tid.sek=loper[minLoperNummerI].finish_tid.sek;
        Nyloper[nyPlassToPlace].finish_tid.timer=loper[minLoperNummerI].finish_tid.timer;
        Nyloper[nyPlassToPlace].finish_tid.updated=loper[minLoperNummerI].finish_tid.updated
        Nyloper[nyPlassToPlace].loper_nr=loper[minLoperNummerI].loper_nr;
        loper[minLoperNummerI].loper_nr=0;
        minLoperNummer = 254;
        nyPlassToPlace++;
    }
}

/*****************************************************************************************
*  Function soterEtterNummer:
```

```
*****************************************************************************************
*         This function reorder the runners after nummer.
*
*****************************************************************************************/
static void soterStaticList_p(void)
{
    uint8_t i;
    uint8_t nyPlassToPlace = 0;
    uint8_t minLoperNummer = 254;
    uint8_t minLoperNummerI;
    loper_type * Nyloper_p[255];
    loper_type allReadyMoved;
    allReadyMoved.loper_nr = 0;
    nyPlassToPlace = 0;
    while(nyPlassToPlace <=254)
    {
        for(i=0; i <= 254; i++)         //  What had been wrong with writing < 255 ?
        {

           if(loper_p[i]->loper_nr != 0)
           if(loper_p[i]->loper_nr < minLoperNummer)
           {
              minLoperNummer =loper_p[i]->loper_nr;
              minLoperNummerI = i;
           }
        }
        Nyloper_p[nyPlassToPlace] = loper_p[minLoperNummerI];
        loper_p[minLoperNummerI] = &allReadyMoved;

        minLoperNummer = 254;
        nyPlassToPlace++;
    }
}


/*****************************************************************************************
*  Function soterEtterNummer:
*****************************************************************************************
*         This function reorder the runners after nummer.
*
*****************************************************************************************/
static void soterDynamicList(void)
{
    node NyHead;          // node for the start point of the sorted dynamic list
    node * nyHead_p;      // pointer to the new head
    node * nyPoini_p;     // pointer to use for looking threw the list
```

```
    node * temp_p;       // temp_p pointer to use when moving node in list
    uint8_t nyNode;
    // variable to indicant that we need to fetch a new node from the list
    NyHead.next = NULL;
    // set the NyHead.next to NULL, at the start the only node is the NyHead
    NyHead.prior= NULL;
    // set the NyHead.prior= NULL, the NyHead is the first node in this list
    nyHead_p = &NyHead; // set the nyHead_p to point to the NyHead node
    nyPoini_p = nyHead_p; // set nyPoint_p to point at the start node.

    while(point_dyn_list->prior != NULL)
// test that the global point_dyn_list-> next !=Null,
//this test test for when we reatch point_dyn_list == head_dyn_list !!
    {
        temp_p = point_dyn_list;
// set temp_p to point to the last node in the list that shall be sorted.
        point_dyn_list->prior->next= NULL;
// take out the node from the list that shall be sorted.
        point_dyn_list = point_dyn_list->prior;
//move the point_dyn_list to its prior node.
        nyNode=0;
// we have now a node that shall be placed in the sorted list, temp_p points to it.
        nyPoini_p = nyHead_p;
// we set the nyPoini_p to point to the nyHead_p, this is the start of the new sorted list
        while(nyNode==0)
// while we have not placed the node in its right place in the list
        {

            if(nyPoini_p->next == NULL)
// test for the end of the new sorted list
            {
                temp_p->next = NULL;
// If we are at the end, place the node at the end.
                temp_p->prior = nyPoini_p;
                nyPoini_p->next = temp_p;
                nyPoini_p = temp_p;
                nyNode=1;
//set nyNode= 1 to break out of the while loop to
//fetch a new node from the list that shall be sorted.
            }
            else if(nyPoini_p->next->loper_nr == 0);
// test for the not legal runner nummer
            else if( nyPoini_p->next->loper_nr > temp_p->loper_nr)
//If the next node has greater number, the we are at the right place,
//putt the node in to the list her.
            {
```

```
                nyPoini_p->next->prior = temp_p;
// change the next nodes prior pointer to point to temp_p
                nyPoini_p->prior->next = temp_p;
// change the prior nodes next pointer to point to temp_p
                temp_p->next = nyPoini_p->next;
// change the temp_p next pointer to point to the next node in the sorted list
                temp_p->prior = nyPoini_p;
// change the temp_p prior pointer to point to the prior node in the sorted list.
                nyNode=1;
// set nyNode=1 to break out of the while loop to fetch a new node
//from the list that shall be sorted.
            }
            else
            {
                nyPoini_p = nyPoini_p->next;
//We are still not at the right place to place the temp_p node,
//step to the next node in the sorted list
            }

        }



    }



}
/*********************************
* main,
*********************************
*   Remember to keep this short
*********************************/
int main(void)
{
    node head; // make the head node for the dynamic list start point

    head.prior=NULL;
    head.next=NULL;
    head_dyn_list=&head; // set the global head_dyn_list to point to start node

    point_dyn_list=head_dyn_list; //set point_dyn_list to point to start node

    initPort();
    while((0 == ((PIND) & (1 << 0))) ); // wait for PIND0 to go high
    initTimer3(); // Init timer3 this is used for emulating a test run.
```

```
    initTimer();  // timer1 is used for the second counter.
    while((1==((PIND) & (1 << 0)))) // run until PINDO go low
    {
        if(nyEvent.nyloper == 1)
        // this variable is used to signal that a new event has happened
        {
           handelNyEvent(&nyEvent);
           nyEvent.nyloper = 0; // reset the new event variable
        }

    }
    stopTimer_1();
    soterEtterNummer(); // TODO this is the function that you need to write!!
    return 0;
}
```