TENTAMEN

KURSNAMN	
PROGRAM: namn **åk / läsperiod**	**REALTIDSSYSTEM** **DAI2 samt EI3**
KURSBETECKNING	**LET 624 0209 (6p)**
EXAMINATOR	**Peter Lundin**
TID FÖR TENTAMEN	**Onsdagen den 19/10 2011 kl 14.00 – 18.00**
HJÄLPMEDEL	**Godkänd räknedosa**
ANSV LÄRARE: namn **telnr** **besöker tentamen kl**	**Peter Lundin (7725726)** **ca 15 samt ca 16.30**
DATUM FÖR **tid och plats för granskning**	**Granskning Onsdagen den 2/11 kl 10.00- 10.30 Sal J121** **Alternativt även på Johanneberg för E3 :** **Torsdagen den 3/11 kl 09.45 sal ES51**
ÖVRIG INFORM.	**Betygsgränser : Max 24 poäng** **3:**11– 15**,5** **4:** 16 – 19,5 **, 5:** 20 -

FORMELSAMLING

RMSA enkel analys
***FÖRUTSÄTTNINGAR*:**

- Varje process i uppsättningen är periodisk
- Deadline (d) och periodtid (p) är lika stora.
- Konstant exekveringstid för processer
- Ingen processkommunikation förekommer
- Alla processer är avbrytbara

n	$n(2^{1/n}-1)$
1	1,000
2	0,828
3	0,780
4	0,757
5	0,743

En uppsättning processer (P_1,P_2,...P_n) är schemaläggningsbar om:

$$\sum_{i=1}^{n} \frac{c_i}{p_i} \leq n\,(2^{1/n}-1)$$

där c_i är är exekveringstiden för P_i och p_i är periodtiden för P_i.

RMSA exact analys (krav som ovan men här räcker det med att d <= p)
Beräkning av svarstiden för process ***i*** under interferens från högre prioriterade processer ***j*** ges av sambandet

$$R_i^{n+1} = c_i + b_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i^n}{p_j} \right\rceil c_j$$

Där b är blockeringsfaktorn , interferens från lägre prioriterade processer.

Svarstidsanalys vid processuppsättningar med fix prioritet och godtycklig deadline.
För en uppsättning processer gäller att maximal svarstid *R*, inom ett fönster *w*, kan beräknas. För blockeringsfaktorn *b* gäller att prioritetstaksprotokoll (max en blockering) är implementerat.

$$w_i^{n+1}(q) = (q+1)c_i + b_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{w_i^n(q) + J_j}{p_j} \right\rceil c_j$$

Iterationen utförs tills : $w_{i,q} <= (q+1)\, p_i$

Svarstiden R ges av:

$$R_i = \left(\max_{q=0,1,2,\ldots} \left(w_i(q) - q p_i \right) \right) + J_i$$

och där:

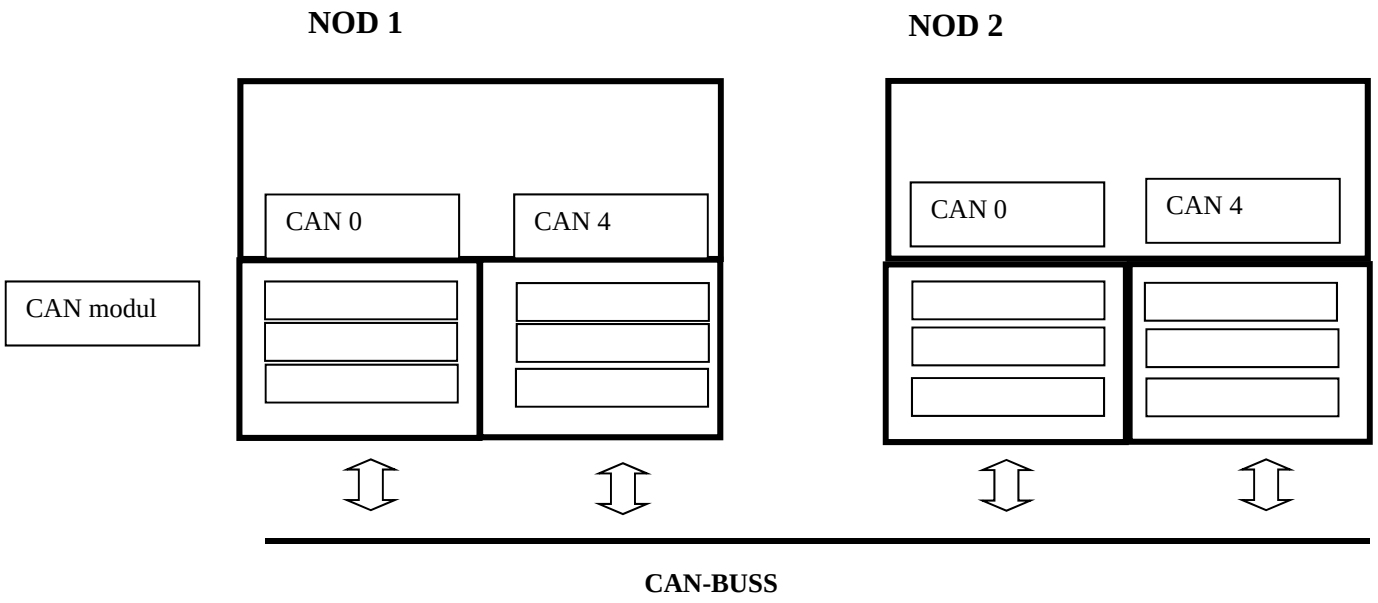
- *w* är det ”fönster” inom vilket vi betraktar svarstider
- *q* är den start, numrerad 0,1,2..., inom fönstret vi betraktar
- *c* är processens exekveringstid
- *b* är blockeringsfaktor för processen
- *p* är processens periodtid
- *J* är maximalt jitter för processen

Uppgift 1. (3p)

a) För CAN-protokollet finns en beskrivning av ramformatet, dvs en beskrivning av vad varje fält i ramen har för funktion . Det första fältet i ramen består av den s.k. arbitreringskoden. Beskriv vilka viktiga funktioner som koden har i samband med en sändning.

b)

Antag att vi har ett system bestående av två datornoder bestående av MC12 kort av samma typ som använts vid våra laborationer. Noderna är anslutna till varandra via en CAN-buss enligt figuren nedan. I varje nod finns ett flertal CAN-moduler av vilka vi i denna uppgift använder oss av modulerna 0 och 4.



Vi har tillgång till ett program som initierar nod1 och sänder 2 meddelanden respektive läser ett av de sända meddelande inom den egna noden enligt programlistan nedan.

Skriv ett program för Nod 2 som läser in och skriver ut båda de sändande texter i terminalfönstret till systemet. Vi kan förutsätta att nod 2:s program har samma inledning som nod1 till nedan rad med undantag för att strängarna sendtxt[] respektive sendtxt2[] ej behövs.

// ---------- Skriv program till Nod 2 från denna rad ------

Program Nod 1

```
void main(void){
    int length=0, i;
    char sendtxt[]={"Send_1\0"};
    char sendtxt2[]={“From_1\0”};
    char intxt[8];
    unsigned long identifier;
    REG8(DDRB)=0xFF;                    //configure CAN0 & CAN4
    config_CAN_port(CAN0PM01,CAN4PM67);
    init_CAN_pointers();
```

```
//default initialize CAN0 o CAN4
 default_CAN(ptrCAN0_init, ptrCAN0);
 default_CAN(ptrCAN4_init, ptrCAN4);

  //initialize CAN0/4 for transmission/ reception
 init_CAN(ptrCAN0_init,ptrCAN0, CANreceive);
 init_CAN(ptrCAN4_init,ptrCAN4, CANreceive);

 // --------------------------------  Skriv program till Nod 2 från denna rad ------
 //Acceptence Reg 1o2
  ptrCAN4->CAN_AR_MR_ID[0]=0x0AAAAAAC;   // ID acc reg
  ptrCAN4->CAN_AR_MR_ID[1]=0x0AAAAAAC;   // ID acc reg
  ptrCAN4->CAN_AR_MR_ID[2]=0xF0000000;       // ID mask reg

 //---------------------- Huvudloop ----------------------------------------------------------
   while(1){
        //send message_1
        //transmission address
         ptrCAN0->CAN_AR_MR_ID[4]=0x0AAAAAAC;
         for(i=0; i<7;i++)
                 ptrCAN_trans_message->byte[i]=sendtxt[i];

          ptrCAN_trans_message->length=7;
          transmit_CAN(ptrCAN0,ptrCAN_trans_message,TXE0);     / /send message

          //send message_2
        //transmission address
         ptrCAN0->CAN_AR_MR_ID[4]=0x0AAAAAAF;
         for(i=0; i<7;i++)
                 ptrCAN_trans_message->byte[i]=sendtxt2[i];

          ptrCAN_trans_message->length=7;
          transmit_CAN(ptrCAN0,ptrCAN_trans_message,TXE0);     / /send message

         if(receive_CAN(ptrCAN4,ptrCAN_rec_message)==1){       //if received message
                length=ptrCAN_rec_message->length;                    // antal databytes
                for ( i=0;i<length;i++){
                          intxt[i]=(ptrCAN_rec_message->byte[i]);
                }
                intxt[i]='\0';                                                  // fixa strängslut
                puts(intxt);
        }else
                  puts("no message\n\r");

  }// End Huvudloop
}// End Main
```

Uppgift 2 (3p).

I ett realtidssystem med stöd av en realtidskärna kan man exekvera flera processer ’samtidigt’via tidsdelning. I ett sådant system är principerna för schemaläggning av central betydelse och i samband med detta finns ett antal olika begrepp. Försök så gott det går att kort förklara innebörden av följande begrepp :

a) Pre-emptive respektive non pre-emptive process.
b) Round robin.
c) Tidsdelad respektive seriell exekvering.
d) Statisk schemaläggning respektive dynamisk schemaläggning.

Uppgift 3 (3p)

Antag att vi har ett system med tre processer som exekveras ’samtidigt’ i ett system med stöd av en realtidskärna . Realtidskärnan stödjer användande av semaforer. Två av processerna delar på en global buffert med variabler av typen heltal där varje process kan läsa respektive skriva data till bufferten enligt kodexemplet nedan. I exemplet finns det kritiska regioner för vilka man vill garantera ömsesidig uteslutning med hjälp av en gemensam Semafor **S1**.

```
PROCESS nr_1 {

        while(1){
                …
                // kritisk region 1
                // ta ut ett tal från bufferten
                if n>0{
                        tal=buf[n];
                        n=n-1;
                }
                ….
                ….
        }
}

PROCESS nr_2{
        while(1){
                ….
                // kritisk region 2
                // lägga in et tal i bufferten
                if n<MAX{
                        n=n+1;
                        buf[n]=intal;
                }
                …
        }
}
```

a) Beskriv kort vad en semafor är och vilka operationer man kan utföra på semaforen. Beskriv även hur operationen påverkar själva semaforen.

b) Beskriv vad man menar med :
- Kritisk region i ett program.
- Odelbar region i ett program.
- Ömsesidig uteslutning .

c) Ange hur man i processerna ovan bör använda sig av semaforen S1 för att garantera ömsesidig uteslutning mellan de kritiska regionerna i processerna ovan.

Uppgift 4 (2p)

I en realtidskärna utan stöd för ‘ömsesidig uteslutning’ på maskinnära nivå har Dekker visat att man kan implementera ömsesidig uteslutning mellan kritiska regioner i två processer, Dekkers algoritm.

Dekker beskriver sin algoritm genom att först beskriva ett antal enklare algoritmer (Dekker 1 – 4) som visar ett antal begränsningar i form av deadlock och svält. Utifrån de fyra icke fungerande lösningarna så tar Dekker fram den slutliga lösningen för två processer.

```
int i1=FALSE ; int i2=FALSE;
PROCESS P1() {
  while(1){
    // icke kritisk region
     ……
    while( i2==TRUE);
    i1=TRUE;
    // Kritisk region
    ……
    …….
    i1=FALSE;
  }
}
```

```
PROCESS P2() {
  while(1){
    // icke kritisk region
     ……
    while( i1==TRUE);
    i2=TRUE;
    // Kritisk region
    ……
    …….
    i2=FALSE;
  }
}
```

a) Ovan processer beskriver en av Dekkers algoritmer. Visa med ett scenario, dvs beskriv en följd av processbyten mellan processerna som ger upphov till problem. Beskriv även vilket problem som uppstår.

b) Beskriv kort begreppen : deadlock respektive svält

Uppgift 5 (3p)

Antag att vi har en uppsättning av processer med data enligt tabellen nedan.
Processerna skall schemaläggas med ett statiskt schema. Processerna får avbrytas.

a) För ett antal periodiska processer med periodtiderna p_n kan man för periodtiderna beräkna ett sk LCM-tal. Justera periodtiderna för nedan system av processer så att ett för statisk schemaläggning lämplig LCM-tal för periodtiderna kan beräknas. Beräkna därefter LCM-talet för periodtiderna både före och efter justeringen.
b) Rita ett möjligt statiskt schema för processerna med den eller de justerade periodtiderna.

Process	p	c	d
P1	6	2	4
P2	14	2	5
P3	18	5	10

Uppgift 6 (3)

Antag att villkoren för dynamisk schemaläggning enligt RMSA är uppfyllda för nedan uppsättning av processer. Processerna är uppställda i fallande prioritetsordning där prioriteten för P1 är högst.

Process	p (ms)	c (ms)	d(ms)
P1	18	4	10
P2	34	6	20
P3	68	12	30
P4	68	16	60

a) Visa att processerna är schemaläggningsbara.
b) Antag att processerna delar användande av en semafor S1 enligt följande beskrivning:
P2 : Använder S1 i 3 ms dvs $cs_{P2,S1}$ = 2 ms
P4 : Använder S1 i 5 ms dvs $cs_{P4,S1}$ = 5 ms

Beräkna maximala svarstiden för P3 med hänsyn till de nya förutsättningarna.

Uppgift 7 (2p)

Antag att vi har ett system av processer där vi inte har några krav på svarstiderna men har krav på en viss prioritetsordning . En av processerna P_2 är dessutom av den karaktären att den har ett startjitter +/- 0,5 ms. Processernas data ges av tabellen nedan där de är listade i prioritetsordning.

Beräkna maximala svarstiden för P3 enligt metoden för godtycklig deadline och med prioritering enligt tabellen.

Proc	Prioritet	p (ms)	c (ms)	J
P1	0	20	5	-
P2	1	25	6	1
P3	2	8	3	-

Uppgift 8 (2p)

I en process finns nedanstående rutin som läser in ett tal från en inport i systemet. Man har för avsikt att teoretiskt uppskatta exekveringstiden för programdelen.
(**Not:** Uppgiften och svar är justerad efter tentamenstillfället pga av felaktigheter)

```
#define ReadInport *((unsigned char*) 0x9C00)
    int antal;
    unsigned char inkod, temp;

1   antal=0;
2   inkod= ReadInport;
3   temp=inkod & 0xF;
4    If(temp){
5      while( !(temp & 0x01 ) && (antal < 4 ) ){
6
7         temp = temp >> 1;
8         antal++;
9      }
10  }
```

a) Rita en programflödesgraf för raderna 1 – 8
b) Beräkna exekveringstiden [min,max] för programdelen om följande uppskattningar gäller:

BB1= [5 ,5]
BB2= [5,5]
BB3= [20,30]
BB4= [5 ,5]
BB5= [25,35]
BB6= [0,0,]
BB7= [5,8]
BB8= [5 ,8]

Uppgift 9 (3p)

Studera nedan programexempel. Programmet skapar en länkad lista med poster av typen REGTYP.

Uppgifter:
a) Skriv en funktion med funktionsprototypen : REGTYP* add_first(REGTYP* temp, int data);
Som lägger till en ny post först i listan och tilldelar fältet ***tal*** värdet enligt inparametern ***data***.
Funktionen skall returnera en pekare till den nya första posten i listan.

b) Ange hur anropet av funktionen ser ut från huvudprogrammet. Se avsedd plats i programlistans huvudprogram.

```
/*********************************************
    Programexempel 9 tentamen i realtidssystem    **
    2011-10-19  / Peter Lundin                    **
*********************************************/

#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>

//#### Konstanter #####
#define MAX 5 // ( alternativt 1 )

// ##### Typdeklaretioner ####
typedef struct q{
      int tal;
      struct q *next;
      struct q *prev;
} REGTYP;

// ##### Funktionsprototyper #####

REGTYP* slumpa_lista();
REGTYP* add_first(REGTYP* temp, int data);

//###### Huvudprogram #######
int main(int argc, char *argv[])
{
   int nr=0;
   int intal;
   REGTYP *akt_post , *head=NULL;

   srand(254);
   head=slumpa_lista();
   akt_post=head;
   while( akt_post!=NULL){
        printf("\n Post nr %d : %d" , nr++, akt_post->tal);
        akt_post=akt_post->next;
   }
  //  Programdel där heltalet intal tilldelas ett visst värde , dvs intal=xxx;
  //  Skapa en ny post först i listan i vilken intal skrivs in.
  // Här anropas den nya funktionen.
   ……
   ……
```

```
 // --- Frigör minnet
  while((akt_post=head)!=NULL){
     head=akt_post->next;
     free(akt_post);
  }

  //------------------
  system("PAUSE");
  return 0;
}
// ====     Slut main   ======================================
REGTYP* slumpa_lista(){
    int nr,i=0;
    REGTYP *top, *old, *item;
    item=(REGTYP*) malloc(sizeof(REGTYP));
    top=item;
    item->tal=0;
    item->next=NULL;
    item->prev=NULL;
    old=item;
    while(i<MAX){

                item=(REGTYP*) malloc(sizeof(REGTYP));
                nr=rand()%100;
                item->tal=nr;
                item->next=NULL;
                old->next=item;
                item->prev=old;
                old=item;
                i++;
    }
    return(top);
}
 //=========================================================
REGTYP* add_first(REGTYP* temp, int data){
// Funktion som lägger till ett element först i den länkade listan samt
// lägger in talet data i elementets fält tal.




}
```

Lycka till önskar Peter Lundin !