Institutionen för data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA

Tentamen med lösningsförslag

EDA482 Maskinorienterad programmering D
EDA483 Maskinorienterad programmering E
DAT017 Maskinorienterad programmering IT
DIT153 Maskinorienterad programmering GU

Måndag 11 mars 2024, kl. 14.00 - 18.00

Examinatorer
Roger Johansson, tel. 772 57 29
Erik Sintorn, tel. 772 52 29
Kontaktperson under tentamen:
Erik Sintorn, tel. 772 52 29

Tillåtna hjälpmedel

Utgåvor som distribuerats inom ramen för kursen, häftet:

- *Quick Guide, Laborationsdator MD407 med tillbehör*

Inget annat än understrykningar (”överstrykningar”) får vara införda i detta häfte.

Tabellverk eller miniräknare får ej användas.

Lösningar
Anslås senast dagen efter tentamen via kursens hemsida.

Granskning
Tid och plats anges på kursens hemsida.

Allmänt

Uppgifter 1 t.o.m 3 utgörs av flervalsfrågor.
Försök först lösa uppgifterna, så som du lärt dig i kursen, jämför sedan med flervalsalternativen och ange det alternativ du anser är riktigt.

Observera att, bland svarsalternativen finns alternativ som *kan* ge poängavdrag. Om du är mycket osäker på ditt svar bör du därför i stället välja att avstå.

Avge dina flervalssvar på blanketten, längst bak i tentamenstesen. Skicka med denna blankett med dina övriga lösningar.

Svar kan avges på svenska eller engelska.

Siffror inom parentes anger full poäng på uppgiften.

För full poäng krävs att:

- redovisningen av svar och lösningar är läslig och tydlig. Ett lösningsblad får endast innehålla redovisningsdelar som hör ihop med en uppgift.
- lösningen ej är onödigt komplicerad.
- du har motiverat dina val och ställningstaganden
- assemblerprogram är utformade enligt de råd och anvisningar som getts under kursen.
- C-program är utformade enligt de råd och anvisningar som getts under kursen.
  I programtexterna skall raderna dras in så att man tydligt ser programmens struktur.

Betygsättning
För godkänt slutbetyg på kursen fordras att både tentamen och laborationer är godkända.

Maximal poäng är 50 och tentamenspoäng ger slutbetyg enligt:
$20p \leq$ betyg 3 $< 30p \leq$ betyg 4 $< 40p \leq$ betyg 5

Uppgift 1a (-2..6p)

Följande deklarationer är givna på "toppnivå".

```
static unsigned short a,b;
static int index;
static int vi[100];
```

Vi har nu tilldelningen:

```
vi[ a++ ] = index;
```

Vilken av följande implementeringar i ARM v6-kod är korrekt?

A

```
LDR   R0,=index
LDR   R0,[R0]
LDRH  R1,=a
LDR   R2,[R1]
ADD   R3,R2,#1
STRH  R3,[R1]
LSL   R2,R2,#4
LDR   R3,vi
STR   R0,[R3+R2]
```

B

```
LDR   R0,=index
LDR   R0,[R0]
LDR   R1,=a
LDR   R2,[R1]
ADD   R3,R2,#1
STR   R3,[R1]
LSL   R2,R2,#2
LDR   R3,=vi
STR   R0,[R3,R2]
```

C

```
LDR   R0,=index
LDR   R0,[R0]
LDR   R1,=a
LDRH  R2,[R1]
ADD   R3,R2,#1
STRH  R3,[R1]
LSL   R2,R2,#2
LDR   R3,=vi
STR   R0,[R3,R2]
```

D

```
LDR   R0,=index
LDR   R0,[R0]
LDR   R1,=a
LDRH   R2,[R1]
ADD   R3,R2,#1
STRH  R3,[R1]
LSL   R2,R2,#4
LDR   R3,=vi
STR   R0,[R3,R2]
```

E

```
LDR   R0,=index
LDR   R0,[R0]
LDR   R1,=a
LDRH  R2,[R1]
ADD   R3,R2,#1
STRH  R3,[R1]
LSL   R2,R2,#4
LDR   R3,=vi
STR   R0,[R3+R2]
```

F

```
LDR   R0,=index
LDR   R0,[R0]
LDRH  R1,=a
LDR   R2,[R1]
ADD   R3,R2,#1
STRH  R3,[R1]
LSL   R2,R2,#4
LDR   R3,vi
STR   R0,[R3+R2]
```

Uppgift 1b (-2..2p)

Följande deklarationer är givna på "toppnivå".

```
static short a,b,c;
static int index;
static int vi[100];
```

Vilken av följande implementeringar i ARM v6-kod är korrekt?

A

```
.ALIGN 2
a:  .BYTE 2
.ALIGN 2
b:  .BYTE 2
.ALIGN 2
c:  .BYTE 2
.ALIGN 4
index:  .BYTE 4
.ALIGN 100
vi:  .BYTE 100
```

B

```
.ALIGN 2
a:  .SHORT 1
.ALIGN 2
b:  .SHORT 1
.ALIGN 2
c:  .SPACE 2
.ALIGN 4
index:  .WORD 1
.ALIGN 100
vi:  .WORD 100
```

C

```
.ALIGN 2
a:  .WORD 2
.ALIGN 1
b:  .WORD 2
.ALIGN 1
c:  .WORD 2
.ALIGN 2
index:  .SPACE 4
vi:  .SPACE 400
```

D

```
.ALIGN 1
a:  .SPACE 2
b:  .SPACE 2
c:  .SPACE 2
.ALIGN 2
index:  .SPACE 4
vi:  .BYTE 400
```

E

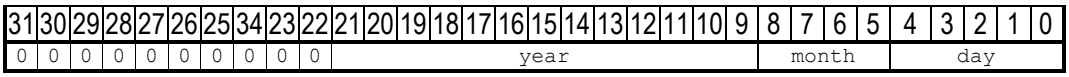
```
.ALIGN 1
a:  .SPACE 2
b:  .SPACE 2
c:  .SPACE 2
.ALIGN 2
index:  .SPACE 4
vi:  .SPACE 400
```

F

```
.ALIGN 1
a:  .SPACE 2
b:  .SPACE 2
c:  .SPACE 2
.ALIGN 2
index:  .SPACE 4
vi:  .SPACE 100
```

Uppgift 2 (-2..6p)

Följande C-funktion `pack` packar födelsedata (år, månad dag) i en `unsigned int`, enligt:




```
int pack( unsigned short year, unsigned short month, unsigned short day)
{
  unsigned int rval = year << 9;
  rval |= month << 5;
  rval |= day;
  return rval;
}
```

Ange det alternativ (A-G) som visar en korrekt implementering (hur funktionen kodas) i ARM v6 assemblerspråk.

A

```
pack:
  SHL  R0,R0,#9
  SHL  R1,R1,#5
  ORR  R0,R0,R2
  ORR  R0,R0,R1
  BX   LR
```

B

```
pack:
  LSL  R0,R0,#9
  LSL  R1,R1,#5
  ORR  R0,R0,R1
  ORR  R0,R0,R2
  BX   LR
```

C

```
pack:
  LSR  R0,R0,#9
  LSR  R1,R1,#5
  ORR  R0,R0,R1
  ORR  R0,R0,R2
  BX   LR
```

D

```
pack:
  LSL  R0,R0,#9
  LSL  R1,R1,#5
  OR   R0,R0,R1
  OR   R0,R0,R2
  BX   LR
```

E

```
pack:
  SHL  R2,R2,#9
  SHL  R1,R1,#5
  ORR  R2,R2,R1
  ORR  R0,R0,R2
  BX   LR
```

F

```
pack:
  LSL  R2,R2,#9
  LSL  R1,R1,#5
  ORR  R2,R2,R1
  ORR  R0,R0,R2
  BX   LR
```

G

```
pack:
  LSR  R2,R2,#9
  LSR  R1,R1,#5
  ORR  R2,R2,R1
  ORR  R0,R0,R2
  BX   LR
```

Uppgift 3 (-2..6p)

En lista av objekt med typen `unsigned int` skall konverteras till `unsigned short`. Om talet är för stort för att få plats i en `unsigned short` skall det inte vara med i den nya listan. Funktionen `ConvertList` skall returnera pekaren till det första elementet i den nya listan, och parametern `num_elems` skall uppdateras med den nya storleken. Huvudprogrammet till höger skall exempelvis ge utskriften: `345 2333 123.`

```
void main() {
  unsigned short*a;
  unsigned int
  v[]={65536,345,2333,0xFFFFFFFE,123};
  int num_elems = sizeof(v)/sizeof(v[0]);

  a = ConvertList(v, &num_elems);
  for (int i = 0; i < num_elems; i++) {
    printf("%i ", a[i]);
  }
}
```

Sex olika programmerare får i uppgift att implementera funktionen `ConvertList( unsigned int *, int *)`, resultaten blir likartade men bara en programmerare presterar ett helt korrekt resultat. De olika alternativen finns nedan, ange alternativet med den korrekta lösningen.

A

```
unsigned short*
ConvertList(unsigned int* a, int* num_elems)
{
    unsigned short* b = (unsigned short*)&a[0];
    int c = 0;
    for (int i = 0; i < (*num_elems); i++) {
        if (a[i] >= 0xFFFF) { c++; }
        else {
            b = a[i];
            *b += 1;
        }
    }
    *num_elems -= c;
    return b;
}
```

B

```
unsigned short*
ConvertList(unsigned int* a, int* num_elems)
{
    unsigned short* b = (unsigned short*)a;
    int c = 0;
    for (int i = 0; i < (*num_elems); i++) {
        if (a[i] >= (1 << 16)) { c++; }
        else {
            *b = a[i];
            b += 2;
        }
    }
    *num_elems -= c;
    return (unsigned short*)a;
}
```

C

```
unsigned short*
ConvertList(unsigned int* a, int* num_elems)
{
    unsigned short* b = (unsigned short*)&a[0];
    for (int i = 0; i < (*num_elems); i++) {
        if (a[i] >= (1 << 16)) {
            *num_elems -= 1;
        }
        else {
            *b = a[i];
            b += 1;
        }
    }
    return (unsigned short*)a;
}
```

D

```
unsigned short*
ConvertList(unsigned int* a, int* num_elems)
{
    unsigned short* b = (unsigned short*)&a[0];
    int c = 0;
    for (int i = 0; i < (*num_elems); i++) {
        if (a[i] >= (1 << 16)) { c++; }
        else {
            *b = a[i];
            b += 1;
        }
    }
    *num_elems -= c;
    return (unsigned short*)a;
}
```

E

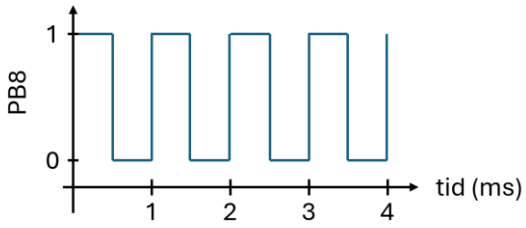
```
unsigned short*
ConvertList(unsigned int* a, int* num_elems)
{
    unsigned short* b = (unsigned short*)a;
    int c = 0;
    for (int i = 0; i < num_elems; i++) {
        if (a[i] >= (1 << 16)) { c++; }
        else {
            *b = a[i];
            b += 1;
        }
    }
    num_elems -= c;
    return (unsigned short*)a;
}
```

F

```
unsigned short*
ConvertList(unsigned int* a, int* num_elems)
{
    unsigned short* b = (unsigned short*)&a[0];
    int c = 0;
    for (int i = 0; i < (*num_elems); i++) {
        if (a[i] >= 0xFFFF) { c++; }
        else { *b++ = *a++; }
    }
    *num_elems -= c;
    return (unsigned short*)a;
}
```

Uppgift 4 (10p)

En fyrkantsvåg med frekvensen 1kHz skall läggas ut på GPIO port B, pinne 8.



a) Visa de makrodefinitioner som behövs för att beskriva SysTick porten. Visa även makrodefinitionen för registret ODR_B_HIGH som motsvarar b8-b15 i GPIO port B. (2p)
b) Skriv funktionen void `start_wave()` som initierar SysTick och startar fyrkantsvågen genom att, med avbrott, utföra avbrottshanteraren `void systick_handler()` var 500:de millisekund. Skriv även koden för avbrottshanteraren. (4p)
c) Ändra avbrottshanteraren så att den även lägger ut en fyrkantsvåg med frekvensen 500Hz på port B, pinne 9. (2p)

d) Två MD407or, **A** och **B**, är ihopkopplade för att överföra data från **A** till **B**. Varje gång en bit skall skickas lägger **A** ut en etta eller nolla på Port D, pinne 0. Sedan flippar den Port D, pinne 1. När **B** märker att pinne 1 byter värde startas en avbrottshanterare som omeddelbart läser av värdet som ligger på pinne 0.

För att spara pengar har man bestämt att överföringen skall ske med bara en sladd från pinne 0 på **A** till pinne 0 på **B**. Båda maskinernas systemklocka är *ungefär, men inte exakt,* 168MHz. Beskriv (i text, ingen kod) hur man tillförlitligt kan överföra datan med bara en sladd. (2p)

Uppgift 5 (10p)

I den här uppgiften ska du implementera en enkel display för att visa ljudvolym. Vi kommer att använda ett numerisk tangentbord med '+' och '-' tangenter (se Figur 2) och en 8-segment bargraph (Figur 3) kopplade till *MD407*-systemet.

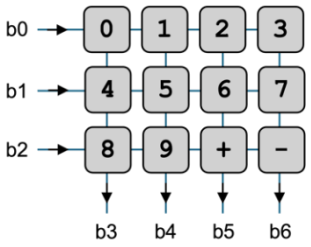
Se Figur 1 som exempel, genom att trycka på '+'-tangenten ökar vi volymen och då skall fler segment lysa. Genom att trycka på '-'-tangenten minskar vi volymen och då skall färre segment lysa på bargraphen.

Från början.

Då man tryckt på '+' tangenten tre gånger.

Då man därefter tryckt på '-' tangenten en gång.

Figur 1



Figur 2

Figur 3

Tangentbordet är anordnat i tre rader och 4 kolumner, som visas i Figur 2, och ansluts till systemet via GPIO-port C. Signaler i stiften som motsvarar bitarna b0-b2 används för att aktivera raderna medan signalen i stiften som motsvarar bitarna b3-b6 används för att identifiera vilken tangent som tryckts ned. Bit b7 används inte. Endast nedtryckning av tangenterna '+' och '-' ska resultera i någon åtgärd i programmet medan nedtryckning av övriga tangenter ignoreras av programmet.

Bargraphen, som ses i Figur 3, ansluts till systemet via GPIO-port D bitar b0-b7.

Alla ingångsstift ska konfigureras som *pull-up* och alla utgångsstift ska konfigureras som *open-drain*.

Implementera ditt program i C-språk och följ stegen nedan:

a) Definera en struct som representerar en GPIO port. Visa sedan hur man refererar MODE-registret i Port D med användning av denna struct. (2p)

Om du väljer att inte göra uppgift a) måste du visa korrekta macrodefinitioner för de register du använder i följande uppgifter.

b) Skriv en funktion `void ports_init(void)` där du gör de initieringar som behövs för att port C och D skall fungera enligt beskrivningen ovan. (2p)

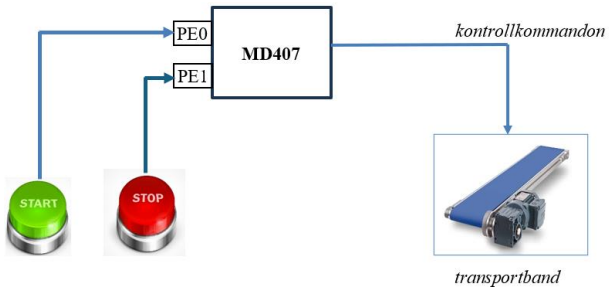
c) Skriv en funktion `int read_key(void)` där du läser av tangentbordet och returnerar ett värde som motsvarar tangenten som har tryckts in. Returvärdena ska vara 0 till 9 som representerar motsvarande numeriska tangenter, 10 för tangenten '+', och 11 för tangenten '-'. (3p)

d) Skriv en funktion `int update_volume(int incdec, int vol)` som tar som inparameter det föregående värdet för volymen (`int vol`) och en flagga som har värdet 1 eller -1 (`int incdec`) beroende på om volymen skall ökas eller minskas med en enhet, och returnerar de nya värdena för volym. Volymvärdena kan vara 0 t.o.m 8. Försök att öka volymen över 8, eller att minska volymen under 0 ska ignoreras. (1p)

e) Gör en huvudfunktion som anropar alla funktionerna ovan för att först initiera portarna och sedan i en oändlig slinga läsa av tangentbordet. Om '+' eller '-' trycks ned uppdateras volymen och motsvarande antal segment skall lysa på bargraphen. (2p)

Uppgift 6 (10p)

Betrakta nedanstående koppling, där en *MD407* har kopplats till ett transportband. Transportbandet kan startas och stoppas från programvara som körs på *MD407*-kortet med hjälp av funktionerna `start_machine()`och `stop_machine()`. Du kan anta att dessa funktioner är implementerade.



Två återfjädrande knappar START och STOP är kopplade till pin PE0 respektive PE1 på MD407-kortet. När respektive knapp trycks ned sätts den tillkopplade pinnen till '1'. När respektive knapp inte är nedtryckt sätts pinnen till '0'.

Målet med denna uppgift är att konfigurera avbrott för pinnarna PE0 respektive PE1, så att avbrott genereras när knapparna trycks ner. Avbrottshanteraren ska anropa anropa funktionen `start_machine()` eller `stop_machine()` beroende på vilken knapp som trycks ned.

Observera, i denna uppgift kan du förutsätta att korrekt typkonverterade macrodefinitioner för pekare till alla register redan finns.

a) Pinnarna PE0 and PE1 ska ställas in som ingångar med pull-down-motstånd. Visa hur detta kan åstadkommas utan att inställningarna för övriga pinnar påverkas. (1p)
b) Visa hur SYSCFG konfigureras så att PE0 and PE1 kopplas till EXTI-modulen i *MD407*. Tänk på att bara PE0 and PE1 ska konfigureras och att övriga EXTI-linor inte får påverkas. (2p)
c) Visa hur EXTI och NVIC konfigureras så att avbrott genereras på positiv flank för de två pinnarna. (3p)
d) Visa en avbrottshanterare `void at_irq (void)` som startar transportbandet genom att anropa `start_machine()` vid positiv flank på PE0 och som stoppar transportbandet genom att anropa `stop_machine()` vid positiv flank på PE1.
Visa också hur avbrottsvektorn i vektortabellen initieras. Systemets vektortabell har tidigare relokerats till adress 0x2001C00. (4p)

Svarsblankett

Anonym kod

Maskinorienterad programmering, tentamen flervalsfrågor.

Kryssa i det alternativ du anser vara korrekt, eller avstå.
Lämna in denna blankett tillsammans med dina övriga lösningar.

Fråga	A	B	C	D	E	F	G	H	I
1a									
1b									
2									
3									

Lösningsförslag

Uppgift 1a (-1..6p)

Svarsalternativ C

```
 LDR   R0,=index
 LDR   R0,[R0]
 LDR   R1,=a
 LDRH  R2,[R1]
 ADD   R3,R2,#1
 STRH  R3,[R1]
 LSL   R2,R2,#2
 LDR   R3,=vi
 STR   R0,[R3,R2]
```

Uppgift 1b (-1..2p)

Svarsalternativ E

```
.ALIGN 1
a:  .SPACE 2
b:  .SPACE 2
c:  .SPACE 2
.ALIGN 2
index:  .SPACE 4
vi:  .SPACE 400
```

Uppgift 2 (6p):

Svarsalternativ B

```
pack:
  LSL  R0,R0,#9
  LSL  R1,R1,#5
  ORR  R0,R0,R1
  ORR  R0,R0,R2
  BX   LR
```

Uppgift 3 (6p):

Svarsalternativ D

```
unsigned short*
ConvertList(unsigned int* a, int* num_elems)
{
    unsigned short* b = (unsigned short*)&a[0];
    int c = 0;
    for (int i = 0; i < (*num_elems); i++) {
        if (a[i] >= (1 << 16)) { c++; }
        else {
            *b = a[i];
            b += 1;
        }
    }
    *num_elems -= c;
    return (unsigned short*)a;
}
```

Uppgift 4 (10p)

a)

```
#define STK_CTRL            ((volatile unsigned int *)0xE000E010)
#define STK_LOAD            ((volatile unsigned int *)0xE000E014)
#define STK_VAL             ((volatile unsigned int *)0xE000E018)
#define STK_CALIB           ((volatile unsigned int *)0xE000E01C)
#define GPIO_B_ODR_HIGH     ((volatile unsigned char *)0x40020415)
```

b)

Anm: Muntlig information under tentamen fel i tentamenstes: "var 500 millisekund" ska vara "var 500 mikrosekund", dvs 2 ggr per millisekund, annars blir 1 kHz omöjlig.

```
void systick_handler() {
  *GPIO_B_ODRHIGH ^= 0x1;       /* Toggle bit PORTB bit 8 */
}
void start_wave()
{
  *((void (**)(void))(0x2001C000 + 0x3C)) = systick_handler;
  /* Assume 168 MHz stystem frequency */
  *STK_CTRL = 0;                /* Disable systick */
  *STK_LOAD = 168 * 500;        /* 500 microsecs period */
  *STK_CTRL = 7;                /* Enable systick */
}
```

c)

```
void systick_handler() {
  *GPIO_B_ODRHIGH ^= 0x1;       /* 1kHz wave on pin 8 */
  if( *GPIO_B_ODRHIGH & 0x1 )   /* Every other (500 Hz) */
    *GPIO_B_ODRHIGH ^= 0x2;     /* Toggle bit PORTB bit 9 */

}
```

d)

(kort beskrivning av Manchester Kodning, eller ett ram protokoll som RS232, se lärobok) 1-2p

Uppgift 5 (10p)

(a) (2p)

Alternativ 1:

```
typedef volatile struct {
   unsigned int moder;
   unsigned int otyper; // +0x4
   unsigned int ospeedr; // +0x8
   unsigned int pupdr; // +0xC (12)
   unsigned int idr; // +0x10
   unsigned int odr; // +0x14
   unsigned int bsrr; // +0x18
   unsigned int lckr; // +0x1C
   unsigned int afrl; // +0x20
   unsigned int afrh; // +0x24
} GPIO, *PGPIO;
```

Anm: Fler alternativ är möjliga här

```
Alt a:
#define GPIO_C (*((volatile PGPIO) 0x40020800))
#define GPIO_D (*((volatile PGPIO) 0x40020c00))
GPIO_D.moder
Alt b:
#define GPIO_C ((volatile PGPIO) 0x40020800)
#define GPIO_D ((volatile PGPIO) 0x40020c00)
GPIO_D->moder
```

Alternativ 2

```
typedef volatile struct {
    unsigned int moder;
    unsigned short otyper; // +0x4
    unsigned short unused_0;
    unsigned int ospeedr; // +0x8
    unsigned int pupdr; // +0xC (12)
    unsigned short idr; // +0x10
    unsigned short unused_1;
    unsigned short odr; // +0x14
    unsigned short unused_2;
    unsigned int bsrr; // +0x18
    unsigned int lckr; // +0x1C
    unsigned int afrl; // +0x20
    unsigned int afrh; // +0x24
} GPIO;
```

Anm: Fler alternativ är möjliga här

```
Alt a:
#define GPIO_C (*((volatile GPIO *) 0x40020800))
#define GPIO_D (*((volatile GPIO *) 0x40020c00))
GPIO_D.moder // Referens
Alt b:
#define GPIO_C ((volatile GPIO *) 0x40020800)
#define GPIO_D ((volatile GPIO *) 0x40020c00)
GPIO_D->moder // Referens
```

(b) (2p)

```
void ports_init(void) {
   GPIO_C.moder = 0x15; /* b0-b2 out and b3-b6 in */
   GPIO_C.otyper = 0x7; /* b0-b2 open drain */
   GPIO_C.pupdr = 0x1540; /* b3-b6 pull-up */
   GPIO_D.moder = 0x5555; /* b0-b7 out */
   GPIO_D.otyper = 0xFF; /* b0-b7 open drain **/
}
```

(c) (3p)

```
int read_key(void) {
   int row, value;

   /* blocking read */
   while(1) {
      /* activate rows */
      for(row=0; row< 3; row++) {
         GPIO_C.odr = 1 << row; /* activate one row at a time b0, b1, b2 */
         value = GPIO_C.idr >> 3; /* read b3-b6 and shift to 0 position */
         col = 0;
         while(value) {
            if(value & 0x1)
               return (row*4+col); /* convert read value into key value */
            col++;
            value = value >> 1;
         }
      }
   }
}
```

(d) (1p)

```
int update_volume(int incdec, int vol) {
   if((incdec == 1) && (vol < 8))
      vol++;
   else if((incdec == -1) && (vol > 0))
      vol--;
   return vol;
}
```

(e) (2p)

```
int main(void) {
   int key;
   int volume = 0;
   int bars;

   ports_init();
   while(1) {
      key = read_key();
      if(key == 10) /* key '+' */
         volume = update_volume(1, volume);
      else if(key == 11) /* key '-' */
         volume = update_volume(-1, volume);
      bars = 0;
      while(volume) {
         bars = bars << 1;
         bars |= 0x1;
         volume = volume -1 1;
      }
      GPIO_D.odr = bars;
   }
}
```

Uppgift 6 (10p)

(a) (1p)

```
*GPIO_E_MODER &= 0xFFFFFFF0;            /*pin0 and pin1 are inputs*/
*GPIO_E_PUPDR &= 0xFFFFFFF0;
*GPIO_E_PUPDR |= 0x0000000A;            /*pin0 and pin1 are pull down*/
```

(b) (2p)

```
// Port E's fingerprint in 0x4 and we need to write this fingerprint
// in SYSCFG_EXTICR1 register's in first 4 bits for pin 0 and next 4 bits
// for pin 1

void InitSYSCFG() {
*SYSCFG_EXTICR1 &= 0xFF00;
*SYSCFG_EXTICR1 |= 0x0044;
}
```

(c) (2p)

```
void InitEXTI() {

// Enable line 0 and line 1 in the EXTI module's mask register.
// Rising edge will generate interrupts from pins PE0 and PE1.
// Falling edge will NOT generate interrupts from pins PE0 and PE1

*EXTI_IMR |= 0x0003;
*EXTI_RTSR |= 0x0003;
*EXTI_FTSR &= 0xFFFC;

// Enable interrupt sources in NVIC
*NVIC_ISER0 |= ((1<<6) | (1<<7));
}
```

(d) (3p)

```
void at_irq (void)
{
// If an interrupt is generated at the rising edge of PE0, i.e., the
// first bit of port E's IDR registers becomes 1, then START the machine

   if( * GPIO_E_IDRLOW & (0x1)){
        start_machine();
   }

// If an interrupt is generated at the rising edge of PE1, i.e., the second
// bit of port E's IDR registers becomes 1, then STOP the machine
   if( * GPIO_E_IDRLOW & (0x2)){
        stop_machine();
   }

   *EXTI_PR |=0x3;            /* Acknowledge both interrupts*/

}

// Initializing the interrupt vector for PE0
*((void (**)(void) ) 0x2001C058 ) =  at_irq;
// Initializing the interrupt vector for PE1
*((void (**)(void) ) 0x2001C05C ) =  at_irq;
```