**Tentamen In2305-ii Embedded Programming**
**Woensdag 17 Januari 2007, 14:00 - 17:00**

Teneinde misverstanden over de syntactische geldigheid van code fragmenten in dit tentamen te voorkomen, zal altijd worden gesproken over pseudo-code hoewel in sommige gevallen er sprake kan zijn van correcte code. Ook gaan we er altijd vanuit dat de benodigde variabelen, semaforen, taken, timers, etc. correct gedeclareerd / geïnitialiseerd zijn.

In dit tentamen gelden de volgende definities, tenzij anders vermeld:

```
void  delay(int ms)
{
   !! do some CPU computation to the amount of ms CPU milliseconds
}

char  getchar()
{
   while (!! UART rx buffer empty) ;
   !! return c from UART rx buffer
}

void  gets(char *s)
{
   !! fill string s using getchar()
}

void  putchar(char c)
{
   while (!! UART tx buffer not empty) ;
   !! send c to UART tx buffer
}

void  puts(char *s)
{
   !! write string s using putchar()
}
```

NB. de rx/tx buffers van de UART kunnen slechts 1 char opslaan,
en de transmissie-snelheid = 100 us / char.
Ook de `printf` functie is geimplementeerd mbv. `putchar/puts`.

In dit tentamen worden de volgende afkortingen gebruikt: RR = Round-Robin, RRI = RR plus interrupts, FQS = function queue scheduling, RTOS = real-time operating system.

Succes met het tentamen.

---

**Vraag 1.**
Gegeven de volgende (pseudo)code:

```
int   done = 0;

void  isr_button(void)  // arrive here when button pressed/released
{
   if (!! button pressed)
      done = 1;
}

void  main(void)
{
   while (! done) {
      printf("Hello World!\r\n");
      delay(10000);
   }
   printf("done\r\n");
}
```

Het programma wordt op de gebruikelijke wijze ge-upload naar het embedded systeem. Welk van de volgende beweringen is het meest waar?

**a.** Het progr. eindigt altijd direct als de button wordt ingedrukt
**b.** Het progr. eindigt soms niet als de button niet ontdenderd is
**c.** Het progr. geeft niet altijd dezelfde output als het herstart wordt zonder opnieuw te uploaden
**d.** Geen van bovenstaande antwoorden.

---

**Vraag 2.**
Welk van de volgende beweringen is het meest waar?

**a.** Een ISR moet kort duren als een snelle responstijd gewenst is
**b.** Een ISR mag geen RTOS aanroepen doen
**c.** Interrupts moeten ge-disabled worden in kritieke secties
**d.** Een RTOS moet altijd weten wanneer een ISR wordt geëxecuteerd

---

Gegeven onderstaande ISR. De (4) buttons zijn allen gemapped op bits 0 .. 3 van het register `buttons`. Elke verandering van het register genereert een IRQ. De buttons zijn *niet* gedebounced.

```
void  isr_buttons(void)
{
   if (buttons & 0x01) c1++;
   ..
   if (buttons & 0x08) c4++;
}
```

**Vraag 3.**
De vier counters (`c1` .. `c4`) staan initieel op 0. Stel dat button #1 wordt ingedrukt. Vervolgens wordt button #4 drie maal ingedrukt (en losgelaten) terwijl button #1 ingedrukt blijft. Welk van de volgende beweringen is het meest waar?

**a.** `c4` = `c1`
**b.** `c4` < `c1`
**c.** `c4` > `c1`
**d.** `c1` = 1 zou een correcte uitkomst kunnen zijn

---

Gegeven de volgende (pseudo)code die de actuele waarde van 4 verschillende buttons leest en daarop acteert. De (4) buttons zijn allen gemapped op bits 0 .. 3 van het register `buttons`. De buttons zijn reeds gedebounced.

```
void  main(void)
{
   while (1) {
      if (buttons & 0x01) {
         f1();  // f1 takes 1 s
      }
      ..
      ..
      if (buttons & 0x08) {
         f4();  // f4 takes 1 s
      }

      f();  // f takes 1 s
   }
}
```

**Vraag 4.**
Geen van de buttons is ooit nog ingedrukt. Men drukt op button #4. Welk van de volgende beweringen is het meest waar?

**a.** Button #4 moet minimaal 1 s ingedrukt worden om `f4()` eenmalig te activeren
**b.** Button #4 moet maximaal 1 s ingedrukt worden om `f4()`eenmalig te activeren
**c.** Button #4 moet minimaal 4 s ingedrukt worden om `f4()`eenmalig te activeren
**d.** Button #4 moet maximaal 4 s ingedrukt worden om `f4()`eenmalig te activeren

**Vraag 5.**
Het systeem is op een gegeven moment in een willekeurige bedrijfssituatie. Op een willekeurig moment drukt men (ook) op button #4. Welk van de volgende beweringen is het meest waar?

**a.** Button #4 moet minimaal 1 s ingedrukt worden om `f4()`te activeren
**b.** Button #4 moet maximaal 1 s ingedrukt worden om `f4()`te activeren
**c.** Button #4 moet minimaal 4 s ingedrukt worden om `f4()`te activeren
**d.** Button #4 moet maximaal 4 s ingedrukt worden om `f4()`te activeren

**Vraag 6.**
Direct nadat button #4 wordt ingedrukt, drukt men tevens op button #1. Welk van de volgende beweringen is het meest waar?

**a.** De volgorde van indrukken biedt geen garantie voor de volgorde van functie-uitvoering.
**b.** Indien beide buttons worden vastgehouden wordt `f4()` altijd eerder geactiveerd dan `f1()`
**c.** Men moet button #1 minimaal 1 s ingedrukt houden om `f1()` eenmaal te activeren
**d.** Geen van bovenstaande antwoorden.

**Vraag 7.**
Stel men wil af van de problematiek mbt. het ingedrukt houden van een button totdat de verlangde functie wordt uitgevoerd. Welk van de volgende beweringen is het meest waar?

**a.** Een RR architectuur biedt een goede oplossing
**b.** Een RRI architectuur biedt een goede oplossing
**c.** Hiervoor is geen oplossing
**d.** Geen van bovenstaande antwoorden.

Stel men kiest voor een RTOS architectuur, met 5 taken en een ISR voor de buttons, als in onderstaande (pseudo)code. Prioriteitsvolgorde (aflopend): `T1`, .., `T4`, `T`.

```
void  isr_buttons(void) // arrive here when button pressed
{
   if (buttons & 0x01) OSSemPost(sem1);
   ..
   if (buttons & 0x08) OSSemPost(sem4);
}

void  T1(void)
{
   while (1) {
      OSSemPend(sem1);
      f1();  // f1 takes 1 s
   }
}

..
..

void  T4(void)
{
   while (1) {
      OSSemPend(sem4);
      f4();  // f4 takes 1 s
   }
}

void  T(void)
{
   while (1) {
      f();  // f takes 1 s
   }
}
```

**Vraag 8.**
Welk van de volgende beweringen is *niet* waar?

**a.** De buttons hoeven niet meer lang te worden ingedrukt
**b.** De responstijd op een button hangt nog steeds af van wat op dat moment wordt uitgevoerd
**c.** De volgorde van indrukken biedt geen garantie voor de volgorde van functie-uitvoering
**d.** De responstijd op een button is maximaal 1 s

---

**Vraag 9.**
Welk van de volgende beweringen is het meest waar?

**a.** Een FQS architectuur heeft een beter responstijd dan een RR architectuur
**b.** De keuze voor een RRI boven een RR architectuur is bedoeld voor processor hogs
**c.** Een RTOS biedt interrupt preemption terwijl een FQS dat niet biedt
**d.** Een FQS biedt task preemption terwijl een RRI dat niet biedt

---

Gegeven de volgende (pseudo)code:

```
void  isr_1(void)       // arrive here on IRQ 1
{
   !! service IRQ1, takes T1 CPU time
}
void  isr_2(void)       // arrive here on IRQ 2
{
   !! service IRQ2, takes T2 CPU time
}
void  task(void)        // run application
{
   while (1) {
      !! do something that takes between T3 .. T4 us CPU time
      !! disable all interrupts
      !! do something that takes between T5 .. T6 us CPU time
      !! enable all interrupts
      !! do something that takes between T7 .. T8 us CPU time
   }
}
```

Stel IRQ 1 heeft prioriteit over IRQ 2. Zij Li, i = 1,2, de interrupt latency tussen IRQ i en de start van ISR i. De tijd die het de CPU kost om een IRQ te processen (overhead) is verwaarloosbaar.

**Vraag 10.**
Welk van de volgende beweringen is het meest waar?

**a.** Er is altijd interrupt latency (L1 > 0, L2 > 0)
**b.** L1 kan oplopen tot T5, L2 kan oplopen tot T5 + T2
**c.** L1 kan oplopen tot T5, L2 kan oplopen tot T5 + T1
**d.** Geen van bovenstaande antwoorden.

**Vraag 11.**
Stel dat IRQ1 elke DeltaT optreedt als gevolg van een inkomend karakter, waarbij de ISR de karakter inleest mbv `getchar()`. Welk van de volgende beweringen is het meest waar?

**a.** De veilige grens waarop geen karakters gemist kunnen worden ligt bij DeltaT = T6 + T1
**b.** De veilige grens waarop geen karakters gemist kunnen worden ligt bij DeltaT = T6
**c.** De veilige grens waarop geen karakters gemist kunnen worden ligt bij DeltaT = T1
**d.** Er kunnen nooit karakters worden gemist

**Vraag 12.**
Stel men verwijdert de disable/enable interrupt code. Welk van de volgende beweringen is het meest waar?

**a.** De veilige grens waarop geen karakters gemist kunnen worden ligt bij DeltaT = T6 + T1
**b.** De veilige grens waarop geen karakters gemist kunnen worden ligt bij DeltaT = T6
**c.** De veilige grens waarop geen karakters gemist kunnen worden ligt bij DeltaT = T1
**d.** Er kunnen nooit karakters worden gemist

---

Gegeven de volgende (pseudo)code, die tot doel heeft om `f()` *precies* elke 1 s te laten uitvoeren.

```
void  isr_timer(void)  // arrive here every 1 s
{
   OSSemPost(event);
}

void  task(void)
{
   while (1) {
      <S>;
      f(); // takes between 20 and 25 ms
   }
}
```

**Vraag 13.**
Het RTOS heeft een interne clock van 20 ms per tick. Welk van de volgende beweringen tav. de invulling van <S> is het meest waar?

**a.** Tav <S> geldt dat `OSSemPend(event)` even goed is als `delay(1000)`
**b.** Tav <S> geldt dat `OSSemPend(event)` even goed is als `delay(975)`
**c.** Tav <S> geldt dat `OSSemPend(event)` even goed is als `OSTimeDly(50)`
**d.** Tav <S> geldt dat `OSSemPend(event)` het beste is

---

Gegeven de volgende (pseudo)code

```
void  isr_button(void)  // arrive here when button pressed
{
   delay(20);     // wait for debounce
   !! do something -- takes another 10 ms
}

void  task(void)
{
   while (1) {
      delay(100);
      f();
   }
}
```

**Vraag 14.**
Welk van de volgende beweringen is het meest waar?

**a.** Als de button wordt ingedrukt midden in de `delay(100)` call, duurt die 30 ms langer
**b.** Als de button wordt ingedrukt midden in de `delay(100)` call, duurt die niet langer
**c.** Als de button wordt ingedrukt midden in de `f()` call, duurt die niet langer
**d.** Geen van bovenstaande antwoorden.

---

Gegeven de vorige (pseudo)code, nu mbv. een RTOS waarbij `T1` hogere prioriteit heeft:

```
void  T1(void)  // execute on event button pressed
{
   while (1) {
      OSSemPend(event);
      delay(20);
      !! do something -- takes another 10 ms
   }
}

void  T2(void)
{
   while (1) {
      delay(100);
      f();
   }
}
```

**Vraag 15.**
Welk van de volgende beweringen is het meest waar?

**a.** Als de button wordt ingedrukt midden in de `delay(100)` call, duurt die 30 ms langer
**b.** Als de button wordt ingedrukt midden in de `delay(100)` call, duurt die niet langer
**c.** Als de button wordt ingedrukt midden in de `f()` call, duurt die niet langer
**d.** Geen van bovenstaande antwoorden.

**Vraag 16.**
Stel we vervangen de `delay` statements door `OSTimeDly` statements (respectievelijk met argumenten 1 en 5 bij een tick van 20 ms). Welk van de volgende beweringen is het meest waar?

**a.** Als de button wordt ingedrukt midden in de `OSTimeDly(5)` call, duurt die 30 ms langer
**b.** Als de button wordt ingedrukt midden in de `OSTimeDly(5)` call, duurt die call niet langer
**c.** Als de button wordt ingedrukt midden in de `f()` call, duurt die call niet langer
**d.** Geen van bovenstaande antwoorden.

---

Gegeven de volgende RTOS (pseudo)code, waarbij `T1` hogere prioriteit heeft:

```
void  T1(void) {
   while (1) {
      puts("1 ");
      delay(20);
   }
}

void  T2(void) {
   while (1) {
      puts("2 ");
      delay(40);
   }
}
```

**Vraag 17.**
Welk van de volgende beweringen is het meest waar?

**a.** De uitvoer is 1 2 1 1 2 1 1 ..
**b.** De uitvoer is 1 2 1 2 1 1 2 ..
**c.** De uitvoer is 1 1 1 1 1 1 1 ..
**d.** Geen van bovenstaande antwoorden.

**Vraag 18.**
Stel we vervangen de `delay` aanroepen door `OSTimeDly` aanroepen (1 tick = 20 ms). Welk van de volgende beweringen is het meest waar?

**a.** De uitvoer is 1 2 1 1 2 1 1 ..
**b.** De uitvoer is 1 2 1 2 1 1 2 ..
**c.** De uitvoer is 1 1 1 1 1 1 1 ..
**d.** Geen van bovenstaande antwoorden.

---

Gegeven de volgende (pseudo)code waarbij `T1` prioriteit heeft over `T2`:

```
void  isr_button(void)  // debounced button
{
   OSIntEnter();
   OSSemPost(request);
   OSIntExit();
}

void  T1(void)
{
   while (1) {
      OSSemPend(request);
      puts(status);
   }
}

void  T2(void)
{
   !! compute system status and update string 'status'
}
```

**Vraag 19.**
Welk van de volgende beweringen is het meest waar?

**a.** Als de button tijdens het printen opnieuw wordt ingedrukt wordt maar 1x geprint
**b.** Het uitrekenen van de system status in `T2` gaat sneller als weinig gedrukt wordt
**c.** Het gebruik van `OSIntEnter`/`OSIntExit` in `isr_button` is hier noodzakelijk
**d.** Geen van bovenstaande antwoorden.

---

Gegeven de volgende (pseudo)code waarbij `T1` prioriteit heeft over `T2`:

```
void  isr(void)
{
   OSIntEnter();
   <S1>;
   OSSemPost(event);
   <S2>;
   OSIntExit();
}

void  T1(void)
{
   OSSemPend(event);
   <S3>;
}

void  T2(void)
{
   !! do something
}
```

**Vraag 20.**
Welk van de volgende beweringen is het meest waar?

**a.** Zonder `OSIntEnter`/`OSIntExit` bestaat de kans op executievolgorde `<S1>,<S2>,<S3>`
**b.** Zonder `OSIntEnter`/`OSIntExit` bestaat de kans op executievolgorde `<S1>,<S3>,<S2>`
**c.** Zonder `OSIntEnter`/`OSIntExit` bestaat de kans dat `<S2>` niet wordt uitgevoerd
**d.** Het gebruik van `OSIntEnter`/`OSIntExit` is in dit programma niet noodzakelijk

---

Gegeven de volgende (pseudo)code van een embedded programma dat de frequentie van een binair signaal meet (frequentie = aantal 0-1 flanken per s). De prioriteitsvolgorde is (aflopend): `isr_timer, isr_signal, isr_button.`

```
void  isr_timer(void)   // arrive here every 1 s
{
   !! put count/2 on display   // takes 3 us
   count = 0;                  // takes 1 us
}

void  isr_signal(void)  // arrive here on any signal edge
{
   count = count + 1;          // takes 10 us
}

void  isr_button(void)  // arrive here when button pressed
{
   exit(0);
}
```

**Vraag 21.**
Er wordt een signaal van 30 kHz aangeboden. De resolutie van het display is beperkt zodat slechts het aantal kHz kan worden afgebeeld. Mbt. `count` is er een potentieel shared data probleem. Welk van de volgende beweringen is het meest waar?

**a.** Het display geeft altijd 30 aan
**b.** Het display geeft altijd 60 aan
**c.** Het display geeft bijna altijd 30 aan
**d.** Het display geeft bijna altijd 60 aan

**Vraag 22.**
Stel dat `isr_signal` prioriteit heeft over `isr_timer`. Er wordt een signaal van 30 kHz aangeboden. Welk van de volgende beweringen is het meest waar?

**a.** Het display geeft altijd 30 aan
**b.** Het display geeft altijd 60 aan
**c.** Het display geeft bijna altijd 30 aan
**d.** Het display geeft bijna altijd 60 aan

---

Gegeven onderstaande (pseudo)code.

```
void  T1(void)
{
   while (1) {
      OSSemPend(sem1);  // may unblock at any time
      f(1);
   }
}
void  T2(void)
{
   while (1) {
      OSSemPend(sem2);  // may unblock at any time
      f(-1);
   }
}
void  f(int i)          // increment some global counter
{
   OSSemPend(mutex);
   counter = counter + i;
   OSSemPost(mutex);
}
```

**Vraag 23.**
Welk van de volgende beweringen is het meest waar?

**a.** De initiële waarde van `mutex` moet 0 zijn voor een correct werking van het programma
**b.** De initiële waarde van `mutex` moet 1 zijn voor een correct werking van het programma
**c.** De initiële waarde van `mutex` moet 2 zijn voor een correct werking van het programma
**d.** `f()` is niet reentrant

---

Gegeven onderstaande (pseudo)code, met aflopende prioriteitsvolgorde: T1, T2, T3.

```
void  T1(void)
{
   while (1) {
      OSSemPend(sem1);   // wait until event #1
      OSSemPend(mutex);
      delay(100);
      OSSemPost(mutex);
      while (!! some condition)
            printf("1 ");
   }
}

void  T2(void)
{
   while (1) {
      OSSemPend(sem2);   // wait until event #2
      while (!! some condition)
            printf("2 ");

   }
}
```

```
void  T3(void)
{
   while (1) {
      OSSemPend(sem3);   // wait until event #3
      OSSemPend(mutex);
      delay(100);
      OSSemPost(mutex);
      while (!! some condition)
            printf("3 ");
   }
}
```

**Vraag 24.**
Stel dat om de 10 ms de events #1, #2 en #3 binnenkomen. Welk van de volgende beweringen is het meest waar?

**a.** De uitvoer is `1 1 1 ..`
**b.** De uitvoer is `1 2 3 ..`
**c.** Bij deze volgorde bestaat kans op priority inversion
**d.** Geen van bovenstaande antwoorden.

**Vraag 25.**
Stel dat events binnenkomen op volgorde: #3, #2, #1. Welk van de volgende beweringen is het meest waar?

**a.** De uitvoer is `3 2 1 ..`
**b.** De uitvoer is `3 3 3 ..`
**c.** De uitvoer is `2 2 2 ..`
**d.** Geen van bovenstaande antwoorden.

---

Gegeven de volgende (pseudo)code, die een onbemand verkenningsvliegtuig bestuurt. T1 voert de besturings-applicatie uit terwijl T2 zorg draagt voor de ontvangst van besturingsgegevens (100 bytes per boodschap) van het grondstation en het terugzenden van telemetrie-informatie (100 bytes per boodschap) naar het grondstation. Voor het voorbeeld mag de UART (rx, tx) verbonden worden beschouwd met een duplex radio link die met het grondstation communiceert. Het communicatieprotocol is synchroon, dwz. er wordt achtereenvolgend ontvangen en gezonden.

```
void  T1(void)
{
   while (1) {
      !! translate pilot commands into airplane control data
      !! control the airplane and compute telemetry data
   }
}
void  T2(void)
{
   while (1) {
      gets(command);    // input commands from ground station
      puts(telemetry);  // output telemetry data to ground station
   }
}
```

**Vraag 26.**
Stel dat `T1` en `T2` worden geëxecuteerd dmv. time slicing (round-robin context switching mbv. een timer, timer interrupt periode 1 ms). De bandbreedte van de communicatie verbinding (UART) is 10 kB/s. Welk van de volgende beweringen is het meest waar?

**a.** Als het grondstation communiceert met `T2` gaat dat ten koste van de snelheid van `T1`
**b.** Een boodschap van `T2` van 100 bytes kan ongeveer 10 ms kosten
**c.** Een boodschap van `T2` van 100 bytes kan ongeveer 20 ms kosten
**d.** Geen van bovenstaande antwoorden.

**Vraag 27.**
Stel dat `T1` en `T2` onder een RTOS worden uitgevoerd waarbij `T2` een hogere prioriteit heeft dan `T1` (de andere gegevens blijven indentiek). Welk van de volgende beweringen is het meest waar?

**a.** Als het grondstation communiceert met `T2` gaat dat ten koste van de snelheid van `T1`
**b.** Een boodschap van `T2` van 100 bytes kan ongeveer 20 ms kosten
**c.** Het vliegtuig is onbestuurbaar
**d.** Geen van bovenstaande antwoorden.

Stel nu dat de invoer en uitvoer via de UART worden aangepast dmv. ISR's volgens

```
void  isr_rx(void)                  // service incoming char
{
   OSSemPost(rx_sem);
}

char  getchar(void)
{
   OSSemPend(rx_sem);
   !! return c from UART rx buffer
}

void  isr_tx(void)                  // ready to transmit new char
{
   OSSemPost(tx_sem);
}

void  putchar(char c)
{
   OSSemPend(tx_sem);
   !! send c to UART tx buffer
}
```

**Vraag 28.**
Welk van de volgende beweringen is het meest waar?

**a.** Als het grondstation communiceert met `T2` gaat dat ten koste van de snelheid van `T1`
**b.** Een boodschap van `T2` van 100 bytes kan ongeveer 20 ms kosten
**c.** Het vliegtuig is onbestuurbaar
**d.** Geen van bovenstaande antwoorden.

Stel dat men het embedded programma verandert volgens onderstaande (pseudo)code, waarbij de bedoeling is om het communicatiekanaal optimaal te benutten dmv een asynchroon protocol (ontvanger en zender communiceren tegelijkertijd). De RTOS prioriteitsvolgorde is (aflopend): `T2`, `T3`, `T1`.

```
void  T1(void)
{
   while (1) {
      !! translate pilot commands into airplane control data
      !! control the airplane and compute telemetry data
   }
}

void  T2(void)
{
   while (1) {
      gets(command);    // input commands from ground station
}


void  T3(void)
{
   while (1) {
      puts(telemetry);  // output telemetry data to ground station
   }
}
```

**Vraag 29.**
Stel dat we de oorspronkelijke (busy-wait) implementatie van `getchar` en `putchar` zouden gebruiken. Welk van de volgende beweringen is het meest waar?

**a.** Als het grondstation communiceert met `T2` gaat dat ten koste van de snelheid van `T1`
**b.** Een boodschap van `T2` van 100 bytes kan ongeveer 20 ms kosten
**c.** Het vliegtuig is onbestuurbaar
**d.** Geen van bovenstaande antwoorden.

**Vraag 30.**
Stel dat we de ISR-implementatie van `getchar` en `putchar` zouden gebruiken. Welk van de volgende beweringen is het meest waar?

**a.** Als het grondstation communiceert met `T2` gaat dat ten koste van de snelheid van `T1`
**b.** Een boodschap van `T2` van 100 bytes kan ongeveer 20 ms kosten
**c.** Het vliegtuig is onbestuurbaar
**d.** Geen van bovenstaande antwoorden.

---

**Einde van het tentamen**