JÖNKÖPING UNIVERSITY
School of Engineering

# FÖRSÄTTSBLAD TENTAMEN

# ENCHIPSDATORER, TEDK18, 7,5HP

**Namn (om tentamen lämnas in):**
Name (*if the examination is handed in*):

.....

| | |
|---|---|
| Datum *Date* | 24 maj 2020 |
| Tid *Time* | 14:00-19:00 |
| Antal sidor inkl. tentamensinformation och bilagor *Number of pages including this page and annexes* | |
| Tillåtna hjälpmedel *Aids permitted to bring* | Bifogade utdrag ur manualer och datablad samt bifogad sammanfattning av HAL |
| Examinator *Examiner* | Andreas Axelsson |
| Examinator besöker salen *The examiner will visit the examination venue* | Nej |
| Examinator nås under tentamen *The examiner may be reached during the examination* | 0709 467760 |
| Betygsättning *Grading* | Tentamen ger maximalt 50p<br>För betyg 3 krävs 20p<br>För betyg 4 krävs 30p<br>För betyg 5 krävs 40p |

**Lösningar redovisas i ett dokument (gärna i Word-format) som laddas upp på pingpong. Alla uppgifter skall vara i ordningsföljd. Om ni har bilder som inte går att infoga i dokumentet skall dessa filer namnges med uppgiftens namn följt av ert användarnamn (kortform). Är det flera filer som ska laddas upp skall de först packas i en zip-fil.**

Lösningarna ska vara renskrivna och lättlästa så det framgår tydligt att tankegången är riktig även om svaret blev fel. Obs! Var noggrann med enheter.

## SKRIV NAMN PÅ ALLA INLÄMNADE BLAD / BIFOGADE FILER.

## LYCKA TILL!

**Uppgift 1.** Besvara följande frågor om enchipsdatorer (14 poäng totalt)

a) Vad står NVIC för? (1p)
*Svar: Nested Vector Interrupt Controller*

b) Vad händer när det sker en interrupt? (2p)
*Svar: Om avbrottet har tillräckligt hög prioritet, avbryter processorn sin nuvarande exekvering och anropar motsvarande avbrottsrutin som pekas ut i avbrottsvektorn. Register som behövs i avbrottsrutinen sparas undan på stacken.*

c) Vad är skillnaden mellan Harvard och von Neuman-arkitektur i en mikrodator? (2p)
*Svar: Harvard har separata bussar för instruktioner och data, medan von Neuman har delad buss för instruktioner och data.*

d) Hur mycket RAM har STM32F401RE? (1p)
*Svar: 96kByte RAM*

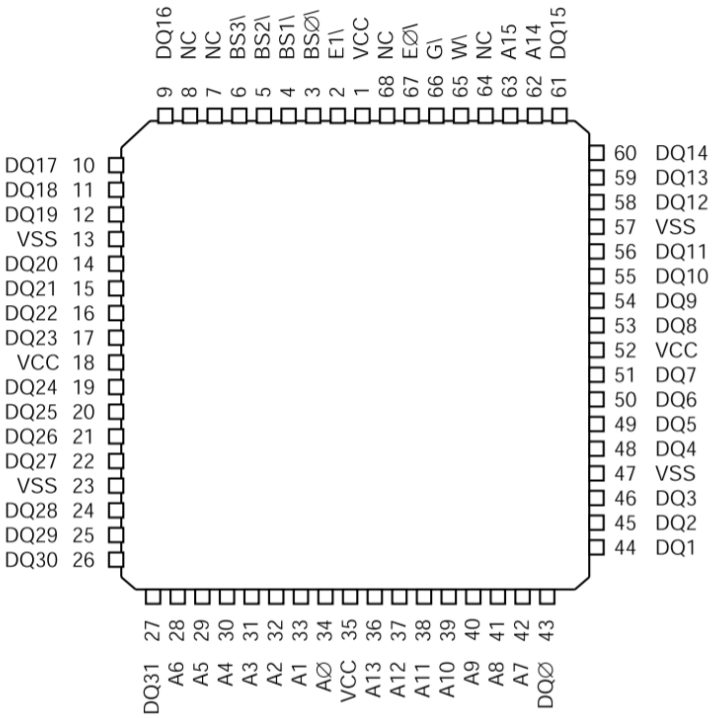
e) Sitter TIM5 på APB1 eller APB2-bussen? Är den 16 eller 32-bitars? (2p)
*Svar: APB1-bussen, 32-bitars.*

f) Nämn tre vanliga periferienheter i en enchipsdator. (1p)
*Svar: Timers, ADC, UART, SPI, RTC, DMA, etc…*

g) I STM32F401RE finns en PLL för klockgenerering. Vad gör den? (2p)
*Svar: PLL står för Phased Locked Loop, och PLL i har till uppgift att kunna generera en programmerbar frekvens mha en fix referensklocka och en varierbar klocka, samt att man kan dela ner dessa klockor och jämföra deras fas. Genom att välja nerdelningen kan man få både högre och lägre frekvenser än referensklockan.*

h) Hur många bytes ryms i nedanstående minne? (2p)
*Svar: Minnet har 16 adresspinnar (A0-A15), dvs den kan adressera 2^16 = 65536 olika adresser. Notera dock att den har 32 datapinnar. Med andra ord innehåller varje adress 32 bitar (4 bytes). Så totalt antal bytes i minnet = 65536*4 = 256 kBytes.*

*i)* Vilken ordlängd har minnet nedan? (1p)
*Svar: 32 bitars ordlängd (minnet har 32 datapinnar, DQ0-DQ31)*



**Uppgift 2.** Var är sant om timers? 1 poäng per rätt svar och -1 per fel svar. (6 poäng totalt)
(Uppgiften kan som lägst ge 0 poäng, obesvarad fråga räknas som -1 poäng)

**Läs noga**:

1. En timer har alltid samma frekvens som CPU-kärnan.
   *Svar: **FALSKT** eftersom man kan dela ner klockfrekvensen innan den går in i timern.*

2. CCRx-registren kan användas både för att fånga timerns räknare vid händelse och för att jämföras mot timerns räknare och göra något om lika.
   *Svar: **SANT** då registret används både som ”capture” och ”compare”.*

3. Alla timers slår om till 0 efter 65535.
   *Svar: **FALSKT** eftersom alla timers inte nödvändigtvis är 16-bitar.*

4. En timer är en “Peripheral”.
   *Svar: **SANT***

5. En timer kan utan CPU-inblandning användas för att räkna pulser.
   *Svar: **SANT**, man kan låta pulsingången vara klocka till timern.*

6. En timer kan inte användas för att trigga t.ex. en AD-omvandling.
   *Svar: **FALSKT**, timern kan användas för att periodiskt trigga andra peripherals.*

**Uppgift 3.** Bengt Bengtsson är besatt av bilbanor. Han har en 10 meter lång bilbana i källaren, den har två parallella spår, två bilar kan alltså tävla mot varandra. Vid start/mål (samma ställe) sitter en fotosensor per spår. Fotosensorerna är av open drain-typ och jordas när en bil är ovanför dem. Bengt vill gärna ha en varvtidsdisplay, den kan visa t.ex. följande:

```
Spår 1: 2.34s
Spår 2: 2.14s
```

Displayen är inkopplad enligt följande:
LCD_RW PB3
LCD_RS PB4
LCD_EN PB5
LCD_D0-D7 PC0-PC7

Sensorerna är inkopplad på PB0 och PB1.

Det finns även en startknapp inkopplad på pinne PA4 och en aktiv summer (den ljuder så fort det finns 3.3V på I/O-pinnen) på pinne PB2.

a) Beskriv i ord hur respektive pinne konfigureras i CubeMX. (4p)
*Svar:*
*PB0 och PB1 konfigureras som EXTI med avbrott på Falling edge. Pga av att sensorerna är open drain behövs även pull-up motstånd för att de normalt ska vara höga när sensorerna inte är aktiva. Finns inte externa pull-up kan man aktivera detta i CubeMX för dessa pinnar.*
*PB3-PB5 och PC0-PC7 sätts som utgångar (push-pull) för att LCD ska funka som tänkt.*
*PB2 sätts som utgång (push-pull) för att kunna styra summern.*
*PA4 kan tänkas vara både som en vanlig ingång eller EXTI beroende på hur man vill använda den i programmet. Beroende på hur man kopplat in knappen kan en pull-up eller pull-down behövas, antingen intern eller extern.*
*För att EXTI-ingångarna ska funka måste NVIC aktiveras för dessa i CubeMX.*

b) Skriv ett program som Bengt Bengtsson kan använda vid tävlingar med sin bilbana. (16p)
När man trycker på startknappen ska summern pipa 3 korta pip och sedan ett långt pip för att indikera att loppet startar. Sedan ska programmet mäta och visa senaste varvtiden. Den skall även hålla reda på antal varv, och när vinnaren har åkt 5 varv ska summern ljuda, och totala körtiden visas på displayen.

En ren struktur och genomtänkt design ger högre poäng. Vi är medvetna om att det är svårt att koda "på papper". Klarar ni inte allt så ange vad ni löst och vad ni inte löst. Om något är oklart och ni tvingas anta något, skriv ner era antaganden!

För detta svar har följande antaganden gjorts:
(Ni kan ha gjort andra antaganden och löst uppgiften på ett annat sätt utan att det är fel).

1. Bilarna startar före start-/mållinjen, d.v.s. sensorn är inte aktiverad, utan aktiveras så fort bilarna passerar startlinjen efter start. På så vis kan man t.ex. utöka sitt program att detektera tjuvstart.

2. Tidtagningen startar samtidigt som den långa ljudsignalen ljuder, reaktionstiden är en del av tävlingen.

3. Varvtiden som visas är den faktiska varvtiden, från passering till passering. Totaltiden som visas på slutet inkluderar reaktionstiden. (Tiden från ljudsignal till att startlinjen passeras (troligen en väldigt kort tid))

```c
// Include files
#include "lcd.h"

// Global variables
uint32_t current_lap_1;
uint32_t current_lap_2;
uint32_t lap_clock_1[6];
uint32_t lap_clock_2[6];
uint32_t start_time;
uint32_t lap_time_1;
uint32_t lap_time_2;
uint32_t race_running;
TextLCDType lcd1;

// Callback for external GPIO interrupts
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    if (GPIO_Pin == GPIO_Pin_0)
    {
        if (current_lap_1 <= 5)
        {
            lap_clock_1[current_lap_1] = HAL_GetTick();
        }
        current_lap_1++;
    }
    if (GPIO_Pin == GPIO_Pin_1)
    {
        if (current_lap_2 <= 5)
        {
            lap_clock_2[current_lap_2] = HAL_GetTick();
        }
        current_lap_2++;
    }
}

// Function to activate buzzer. Input - duration in milliseconds
void Beep(uint32_t duration)
{
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_2, GPIO_PIN_SET); // Activate summer
    HAL_Delay(duration);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_2, GPIO_PIN_RESET); // Deactivate summer
}

// Main entry point to our program
int main(void)
{

     … Init code … (CubeMX HAL_Init(); and set up GPIO etc according to 3a, as well as
                    LCD driver init)

    while (1)
    {

        // Wait on start button
        while(HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_4)) // PA4 active low, internal pull-up.
            ;

        // Reset lap variable for car 1 and 2
        current_lap_1 = 0;
        current_lap_2 = 0;

        for (uint8_t i = 0; i < 3; i++) // Generate 3 short sound beeps
        {
            Beep(200);
            HAL_Delay(500);
        }

        // Here the race starts. Read the clock for the start event and generate a long beep
        start_time = HAL_GetTick();
        Beep(1000);
```

```
        // Loop while
        race_running = 1;
        while (race_running)
        {
            if (current_lap_1 > 1)
            {
                // Update lap time with latest time
                lap_time_1 = lap_clock_1[current_lap_1 - 1] - lap_clock_1[current_lap_1 - 2];

                if(current_lap_1 > 5)  // Finished!
                {
                    // Reuse lap_time_1 for total time.
                    lap_time_1 = lap_clock_1[5] - start_time;
                }
            }

            if (current_lap_2 > 1)
            {
                // Update lap time with latest time
                lap_time_2 = lap_clock_2[current_lap_2 - 1] - lap_clock_2[current_lap_2 - 2];

                if(current_lap_2 > 5)  // Finished!
                {
                    // Reuse lap_time_2 for total time.
                    lap_time_2 = lap_clock_2[5] - start_time;
                }
             }

            TextLCD_Position(&lcd1, 0,0);
            if(current_lap_1 <= 5)
            {
                TextLCD_Printf("Track 1: %.2fs  ", lap_time_1/1000.0);
            }
            else
            {
                TextLCD_Printf("Total 1: %.2fs  ", lap_time_1/1000.0);
                Beep(2000);
                race_running = 0;
            }

            TextLCD_Position(&lcd1, 0,1);
            if(current_lap_2 <= 5)
            {
                TextLCD_Position(&lcd1, 0,1);
                TextLCD_Printf("Track 2: %.2fs  ", lap_time_2/1000.0);
            }
            else
            {
                TextLCD_Printf("Total 2: %.2fs  ", lap_time_2/1000.0);
                Beep(2000);
                race_running = 0;
            }
        } /* while (race_running) */
    } /* while (1) */
}
```

**Uppgift 4.** Du har fått i uppgift att hjälpa till med en konstinstallation där en stor RGB-LED ska användas som ska kunna visa alla möjliga färger. Du ska med hjälp av en timer / PWM styra intensiteten för röd, grön och blå oberoende av varandra.
Konfigurera så att PWM-räknaren har 256 steg (0-255). Uppdateringsfrekvensen ska vara ca 100 Hz. Ange eventuella antaganden.

a) Vad ska Prescaler och Counter Period (auto-reload) sättas till i CubeMX? (2p)
*Svar: Antag att klockan in till Timer-modulen är 84MHz. Vi vill ha 256 steg på PWM-räknaren, vilket betyder att Counter Period ska vara 256-1. Vi vill ha en uppdateringsfrekvens på ca 100 Hz. Det betyder att Prescaler = 84000000/256/100 - 1 = 3280*

b) Hur kan man styra R, G, och B individuellt mha av en timer? (2p)
*Svar: Man kan ha upp till fyra individuella kanaler per timer (Channel 1 – Channel 4), som används som compare för att ställa duty cycle för PWM. Man kan använda tre av dessa för R, G och B.*

c) Hur många olika unika nyanser kan man totalt få från RGB-dioden med ovan använda inställningar? (1p)
*Svar: Man kan ha 256 olika nyanser per grundfärg (R, G och B). Totalt blir det då 256*256*256 = 16777216 olika nyanser.*

d) Om RGB-lysdioderna drar mer ström än processorpinnarna kan leverera, hur löser man det? (2p)
*Svar: Man använder externa transistorer som man kan slå på och av respektive lysdiod (en transistor var för R, G och B).*

**Uppgift 5.** Besvara följande angående A/D-omvandling i STM32F401RE:

a. Vilken upplösning fås om man använder en 12-bit A/D-omvandlare? (1p)
*Svar: 2^12 olika steg, dvs 4096. Har man t.ex. 3.3V full scale motsvarar det en upplösning på ca 0,8 mV.*

b. Vad innebär det att A/D-omvandlaren är kopplad till en MUX? (1p)
*Svar: Att den kan omvandla spänningen som finns på fler än en ingång till mikrokontrollern. Dock kan den bara omvandla en spänning åt gången, så man får välja med hjälp av en MUX vilken pinne som är framkopplad till A/D-omvandlaren.*

c. Du mäter 1,2V med en 12-bitars ADC, vars VREF är ansluten till 3,3V.
Vad kan du förvänta dig för värde ur ADC:ns resultatregister? (1p)
*Svar: Om VREF är 3.3V (dvs full scale) så motsvarar det 4095 i digitalt värde. Vid 1.2V in så får man då 1.2V/3.3V*4095 = 1489 som digitalt värde. Pga brus mm så får man kanske inte exakt det värdet utan något som ligger i närheten.*

# APPENDIX: LCD-SUBRUTINER

| Gäller PIC: | Gäller ARM: |
|---|---|
| Definitioner för att underlätta läsning av koden: | Definitioner för att underlätta läsning av koden: |
| #define LCD_E RD7<br>#define LCD_RW RD6<br>#define LCD_RS RD5<br>#define LCD_FCN_SET 0x38<br>#define LCD_ENTRY_MODE 0x06<br>#define LCD_DISPLAY_ON 0x0E<br>#define LCD_DISPLAY_CLEAR 0x01<br>#define LCD_CURSOR_HOME 0x80<br>#define LCD_CURSOR_LINE2 0xC0<br>#define LCD_COMMAND 0<br>#define LCD_DATA 1<br>#define HIGH 1<br>#define LOW 0 | `typedef struct {`<br>`uint8_t bits, rows, cols;`<br>`GPIO_TypeDef *controlPort, *dataPort;`<br>`uint16_t rsPin, strbPin, rwPin;`<br>`uint16_t dataPins;`<br><br>`} TextLCDType;` |
| Funktioner: | Funktioner: |
| void **lcd_tkn**(unsigned char t); | **void TextLCD_Strobe**(TextLCDType *lcd); |
| void **display_ascii** ( char *s); | **void TextLCD_Puts** (TextLCDType *lcd, **char** *string); |
| **void itoa(uint8_t tal, char *siffror);** | **void itoa(uint32_t tal, char *siffror);** |
| void **lcd_clear**(void); | **void TextLCD_Clear** (TextLCDType *lcd); |
| void **lcd_home**(void); | **void TextLCD_Home** (TextLCDType *lcd); |
| void **lcd_ini** (void); | **void TextLCD_Init(TextLCDType *lcd, GPIO_TypeDef *controlPort, uint16_t rsPin, uint16_t rwPin, uint16_t enPin, GPIO_TypeDef *dataPort, uint16_t dataPins);** |
| | **void TextLCD_Printf** (TextLCDType *lcd, char *message, ...); |