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*	9 januari 2021
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 För betyg 3 krävs 20p För betyg 4 krävs 30p För betyg 5 krävs 40p

Alla uppgifter skall vara i ordningsföljd.

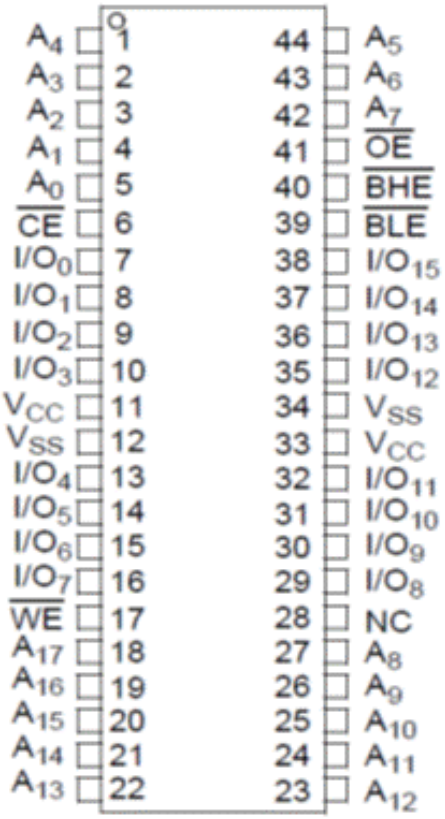
Lösningarna ska vara renskrivna och lättlästa så det framgår tydligt att tankegången är riktig även om svaret blev fel. Obs! Var noggrann med enheter.

SKRIV NAMN PÅ ALLA INLÄMNADE BLAD / BIFOGADE FILER.

LYCKA TILL!

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

a) Nämn tre vanliga periferienheter i en enchipsdator. (1p)
b) Vad står NVIC för? (1p)
c) Vad står RTC för, och vad används den till? (2p)
d) Vad är skillnaden mellan Harvard och von Neuman-arkitektur i en mikrodator? (2p)
e) Hur mycket FLASH-minne har STM32F401RE? (1p)
f) Sitter TIM5 på APB1 eller APB2-bussen? Är den 16 eller 32-bitars? (2p)
g) I STM32F401RE finns en PLL för klockgenerering. Vad gör den? (2p)
h) Hur många bytes ryms i nedanstående minne? (2p)
i) Vilken ordlängd har minnet nedan? (1p)



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 är en "Peripheral".
2. En timer har alltid samma frekvens som CPU-kärnan.
3. CCRx-registren kan användas både för att fånga timerns räknare vid händelse och för att jämföras mot timerns räknare och göra något om lika.
4. TIM2 i STM32F401RE har en 16 bitars räknare.
5. En timer kan utan CPU-inblandning användas för att räkna pulser.
6. En timer kan inte användas för att trigga t.ex. en AD-omvandling.

Uppgift 3. Besvara följande angående A/D-omvandling i STM32F401RE:

a. Vilken upplösning fås om man använder en 12-bit A/D-omvandlare? (1p)

b. Vad innebär det att A/D-omvandlaren är kopplad till en MUX? (1p)

c. Vilken typ av A/D-omvandlare är vanligast i mikrodatorer? (1p)

d. Du mäter 2,1V med en 12-bitars ADC, vars VREF är ansluten till 3,3V. Vad kan du förvänta dig för värde ur ADC:ns resultatregister? (1p)

Uppgift 4. I många apparater använder man en s.k. roterande pulsgivare som användaren kan vrida på, istället för att använda ett tangentbord. En normal pulsgivare kan vridas åt både höger och vänster, och varje helt varv är uppdelat på t.ex. 256 steg med tydliga "klick-steg". Pulsgivaren har tre anslutningsben. Ett av benen är jord, och de övriga två är signalben (A och B). När givarens axel vrids genereras korta pulser på signalutgångarna. Om axeln vrids åt vänster kommer signal B att gå hög en liten tid efter signal A. Om axeln vrids åt höger kommer signal B att gå hög en liten tid före signal A, dvs. omvänd situation jämfört med vänstervridning. Signalutgångarna (A och B) är att betrakta som rent mekaniska switchar och måste förses med pullup-motstånd och avstudsas för att ha rena signaler ut. Lös följande uppgifter:

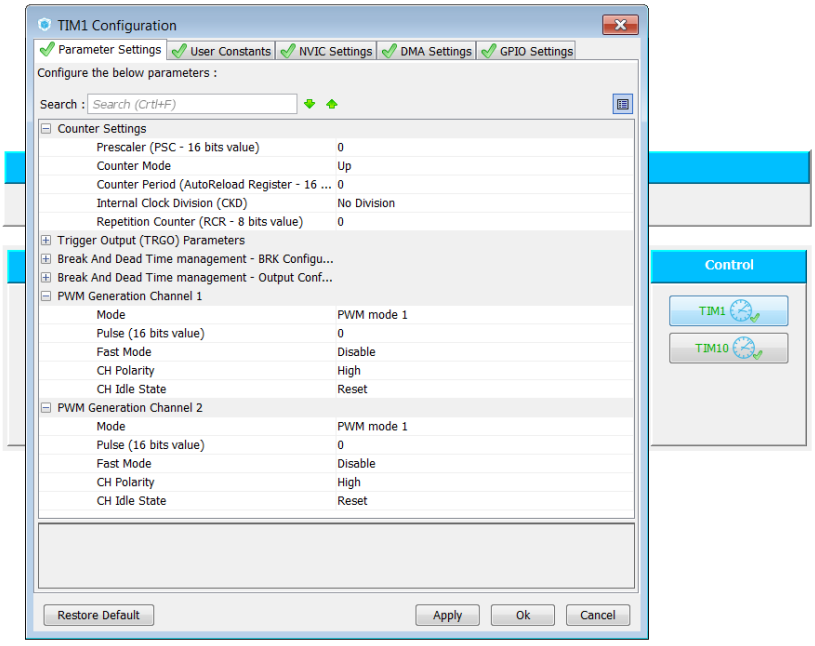
a) Rita kopplingsschema för att ansluta pulsgivaren till processorn. (2p)
b) Hur gör du för att erhålla avbrott (interrupt) varje gång axeln vrids åt något håll? (Beskriv med ord – skriv inte kod) (3p)
c) Antag att man har en variabel i sitt system kallad MENU. Denna variabel återspeglar axelns position genom att innehålla ett tal mellan 0 och 255. Skriv en avbrottsrutin som detekterar vridningsriktning och uppdaterar variabeln MENU genom att addera 1 för varje detekterad högervridning och subtraherar 1 för varje detekterad vänstervridning (denna variabel skulle senare kunna vara användbar för att visa olika menyer i en applikation). (5p)
d) Du vill skriva ut variabeln MENU på din lcd-display och har tillgång till färdiga funktioner enligt appendix. Skriv kodraderna som behövs för att initiera lcd-displayen och skriva ut: Meny val: 23 (Skriv inte själva menyhanteraren) (3p)

Uppgift 5. GPIO-portarna kan skrivas via ett bit-set/reset-register, BSRR.Skriv kodraden som sätter bit 4 till etta och bit 5 till nolla (Ledning: Manualens avsnitt 8.4.7) (3p)

Uppgift 6. I en applikation skall du kunna styra varvtalet på en likströmsmotor. Enklast görs detta med en PWM-kanal från enchipsdatorn. När man trycker på PA4 skall motorn ”mjukstartas” från stillastående till full hastighet på några sekunder. När man trycker på PA5 skall motorn ”mjukstoppas” på ung. samma tid. Den exakta tiden är inte så kritisk. Anta att mikrokontrollerns klockfrekvens är 84 MHz.

e) Du behöver koppla dina tryckknappar så att du inte får kontaktstudsar. Rita schema för inkopplingen.
(3p)

f) Visa hur du skall ställa in PWM-enheten för 1kHz switchfrekvens. Redovisa beräkningar på ett tydligt sätt. Fyll i figuren nedan
(3p)



c) Skriv program som anropar subrutin **void motorStart(void)** som mjukstartar motorn om PA4 trycks ned, och subrutin **void motorStop(void)** om PA5 trycks. För full poäng skall huvudloop, och subrutinerna motorStart och motorStop skrivas. (4p)

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
#define LCD_RW RD6
#define LCD_RS RD5
#define LCD_FCN_SET 0x38
#define LCD_ENTRY_MODE 0x06
#define LCD_DISPLAY_ON 0x0E
#define LCD_DISPLAY_CLEAR 0x01
#define LCD_CURSOR_HOME 0x80
#define LCD_CURSOR_LINE2 0xC0
#define LCD_COMMAND 0
#define LCD_DATA 1
#define HIGH 1
#define LOW 0
	`typedef struct {`
	`uint8_t bits, rows, cols;`
	`GPIO_TypeDef *controlPort, *dataPort;`
	`uint16_t rsPin, strbPin, rwPin;`
	`uint16_t dataPins;`
	
	`} 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, ...);