Tentamen

Enchipsdatorer TEDK18 7,5 hp

Datum:	2020 - 08 - 11
Tid:	14:00 - 19:00
Plats:	
Antal sidor inklusive försättsblad:	7
Bilaga:	Manualutdrag ur Referensmanual och datablad till STM32F401RE
Tillåtna hjälpmedel:	Valfri räknedosa, linjal
Examinator:	Andreas Axelsson
Examinator besöker tentamen:	nej
Examinator nås under tentamen:	0709-467760
Betyg:	Tentamen ger maximalt 50p För betyg 3 krävs 20p För betyg 4 krävs 30p För betyg 5 krävs 40p

OBS: Skriv namn på alla inlämnade blad.

LYCKA TILL!

Denna sida är avsiktligt tom

1) Besvara följande frågor om mikrodatorsystem: (totalt 20p)

a) Vad skiljer en RISC- processor från en CISC-processor? Ange en fördel och en nackdel med RISC-processorn jämfört med CISC-processorn (2p)

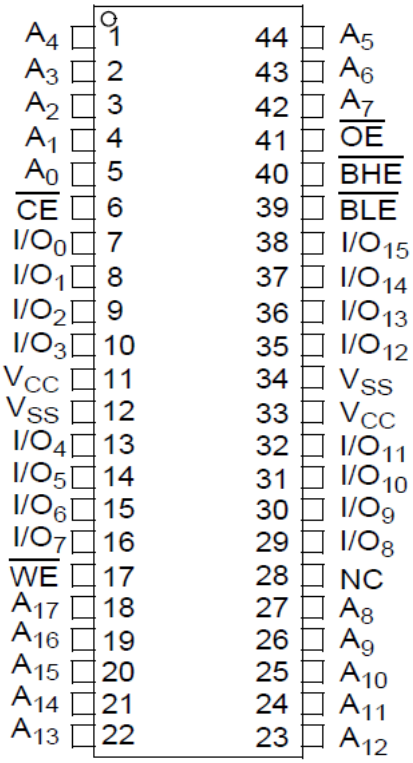
b) Nämn två olika sätt att spara ström vid användning av en mikrodator. Motivera! (2p)

c) Till vad används MODER? (2p)

d) Vad är skillnaden mellan interrupt och att polla en port? (1p)

e) Hur många bytes ryms i nedanstående minne? (2p)

f) Vilken ordlängd har minnet? (1p)



g) Beskriv med ord vad som händer i en ARM-processor då ett interrupt inträffar. Beskriv alltså vad som händer från det att en händelse inträffar på en port tills du får en ”Call-Back” (inte vad du ska göra i ”call-backen”) (3p)

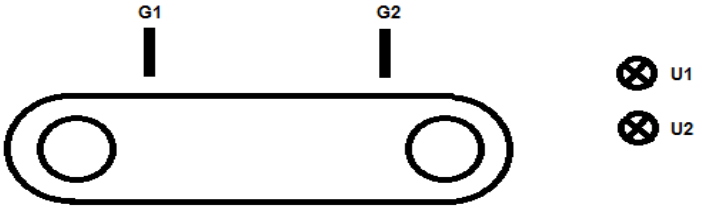
h) Nämn två tillämpningar som du kan använda Timers till (utöver att mäta tid)? (2p)

i) I STM32F401 Används olika register för input och output. Vad kallas dessa register? (1p)

j) GPIO-portarna kan skrivas via ett bit-set/reset-register, BSRR.Skriv kodraden som sätter bit 9 till etta och bit 5 till nolla (Ledning: Manualens avsnitt 8.4.7) (3p)

k) För ST:s processorer behöver du alltid tillgång till tre olika manualer. Vilken information hittar du i programmeringsmanualen? (1p)

2) I en fabrik som tillverkar långa och korta lådor finns ett löpande band med två optiska givare monterade med en meters mellanrum (Brädorna kommer in från vänster). Givare 1 kopplas in på PC1 och givare 2 på PC2. De ger 1 om en bräda passerar, och 0 annars. Beroende på lådlängd kommer således olika sekvenser att genereras.
Motorn för att driva transportbandet är kopplad till en PWM-kanal, PB6 , dvs TIM4_CHANNEL1

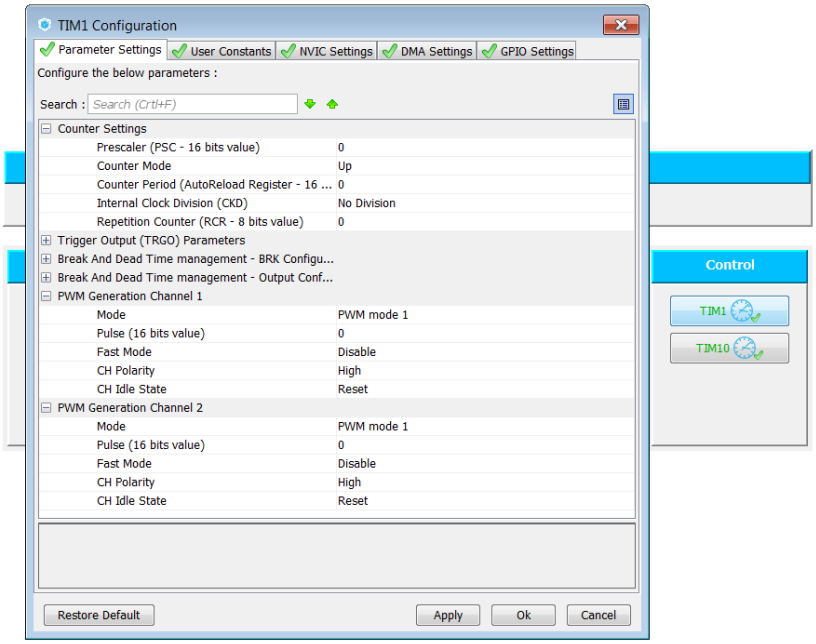


a) Vilka sekvenser kan erhållas? (1p)

b) Skapa en enum-variabel BOX, som kan anta värdena NONE, LONG och SHORT. (2p)

c) Skriv Call-back-funktionen så att du i variabeln BOX kan avgöra om lådan som kommer in är lång eller kort. (5p)

d) För att styra motorn behöver du ställa in TIM4 för PWM. Du vill ha switchfrekvensen 42kHz och din klockfrekvens är 84MHz. Gör lämpliga inställningar i bilden nedan: (2p)



e) För att mjukstarta motorn behöver du skriva en funktion MotorStart() som genererar en duty-cycle som börjar på 10% under ca 0,5s och sedan ökar med 10% i taget tills den når 100%. (Vänta ca 0,5s mellan varje ökning). Skriv funktionen. (3p)

f) Skriv ett program som på din display skriver ut ”Lång låda” eller ”Kort låda”, beroende på om variabeln BOX har värdet LONG eller SHORT. Texten ska vara synlig på skärmen tills nästa låda kommer fram till givaren G1 (4p)
(OBS!) Denna deluppgift kan skrivas även om du inte klarat c, d eller e-uppgiften)

g) Motorerna kräver mer ström än vad mikrodatorn kan ge, dvs du behöver en transistor för att driva motorn. Rita kopplingsschema och utför beräkningar för inkoppling av motorn. Antag att du driver motorn från en 3.3V-utgång och att den arbetar med 12V, 500mA (3p)

3) Du ska ordna en automat för att köpa bensin i en automat. Du ska lösa den som en tillståndsmaskin. Processen börjar med att en person för in sitt kort i automaten. Kortet läses av och om det finns pengar på kortet reserveras 500 kr (Exakt hur detta går till behöver du inte veta utan det funktionen
boolean readCard(void)
returnerar true om beloppet reserverats och false om beloppet inte kunde reserveras.
Om beloppet kunde reserveras fortsätter processen med att starta pumpen, som är kopplad som en utgång på portC (pinne PC5). För att bensin ska komma ut ur pumpen måse sedan dessutom handtaget tryckas in. Bensinen ska flöda tills antingen den som tankar släpper handtaget eller tills en givare känner av att det blivit fullt i tanken. Därefter ska volym, pris och pris/l skrivas ut på skärmen (du behöver inte se till att utskriften uppdateras hela tiden). Efter 40 sekunder ska skärmen återställas och nästa kund ska kunna tanka.

Lös följande uppgifter:

a) Rita kopplingsschema för att ansluta kontakten i handtaget till processorn (görs på samma sätt som tryckknappar som vi använt på laborationerna. (2p)
b) Rita ett tillståndsdiagram för tillståndsmaskinen (state machine). (4p)
c) Skriv koden för tillståndsmaskinen. Antag att interruptservicerutinen sätter variabeln Handtag =1 (du behöver inte skriva interruptservicerutinen). Du måste använda ett timerinterrupt för tidräkningen. För detta ändamål finns en variabel SekTick, som räknas upp en gång för varje sekund och som slår om till 0 efter att ha räknat till 59. (Även denna sköter sig automatiskt, så du behöver bara använda variabeln SekTick) (4p)

APPENDIX: LCD-SUBRUTINER

<table>
<tr><th>Gäller PIC:</th><th>Gäller ARM:</th></tr>
<tr><td colspan="2">Definitioner för att underlätta läsning av koden:</td></tr>
<tr><td>#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</td><td><pre>typedef struct {
     uint8_t bits, rows, cols;
     GPIO_TypeDef *controlPort,
*dataPort;
     uint16_t rsPin, strbPin, rwPin;
     uint16_t dataPins;

} TextLCDType;</pre></td></tr>
<tr><td colspan="2">Funktioner:</td></tr>
<tr><td>void lcd_tkn(unsigned char t);</td><td>void TextLCD_Strobe(TextLCDType *lcd);</td></tr>
<tr><td>void display_ascii (char *s);</td><td>void TextLCD_Puts (TextLCDType *lcd, char *string);</td></tr>
<tr><td>void itoa(uint8_t tal, char *siffror);</td><td>void itoa(uint32_t tal, char *siffror);</td></tr>
<tr><td>void lcd_clear(void);</td><td>void TextLCD_Clear (TextLCDType *lcd);</td></tr>
<tr><td>void lcd_home(void);</td><td>void TextLCD_Home (TextLCDType *lcd);</td></tr>
<tr><td>void lcd_ini (void);</td><td>void TextLCD_Init(TextLCDType *lcd, GPIO_TypeDef *controlPort, uint16_t rsPin, uint16_t rwPin, uint16_t enPin, GPIO_TypeDef *dataPort, uint16_t dataPins);</td></tr>
<tr><td></td><td>void TextLCD_Printf (TextLCDType *lcd, char *message, ...);</td></tr>
</table>

STM32 HAL-sammanfattning.

- `void HAL_Delay(uint32_t Delay)`
- `uint32_t HAL_GetTick( void )`
- `void HAL_GPIO_WritePin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState)`
- `GPIO_PinState HAL_GPIO_ReadPin( GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)`
- `void HAL_GPIO_TogglePin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)`
- `void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)`
- `HAL_StatusTypeDef HAL_TIM_Base_Start_IT(TIM_HandleTypeDef *htim)`
- `HAL_StatusTypeDef HAL_TIM_PWM_Start(TIM_HandleTypeDef *htim, uint32_t Channel)`
- `__HAL_TIM_SET_COMPARE(__HANDLE__, __CHANNEL__, __COMPARE__)`
- `void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim):`
- `void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)`
- `__HAL_ADC_GET_FLAG(__HANDLE__, __FLAG__)`
- `uint32_t HAL_ADC_GetValue(ADC_HandleTypeDef* hadc)`
- `HAL_StatusTypeDef HAL_ADC_Start_IT(ADC_HandelTypeDef* hadc)`
- `void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc):`