Namn:
Laborationen godkänd:

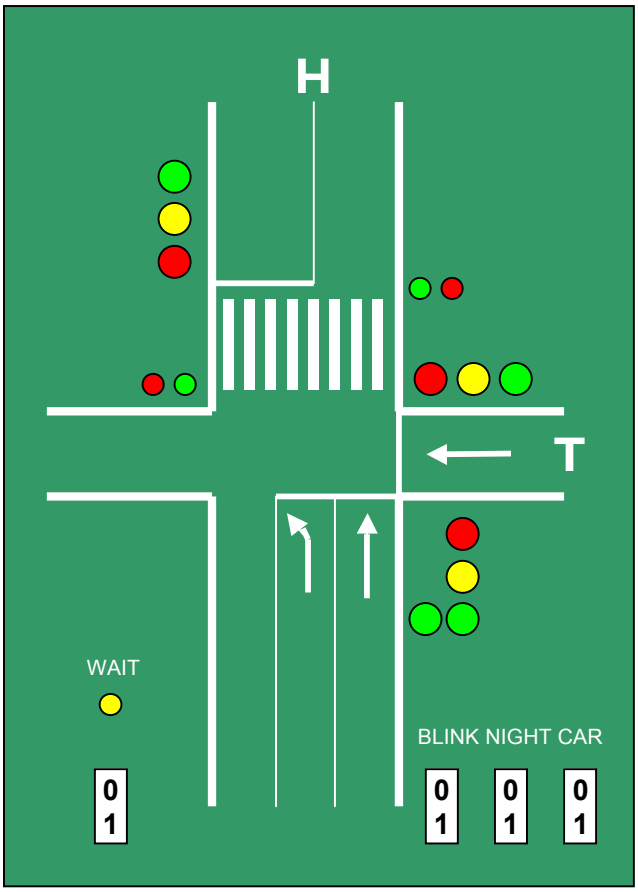
Digitala system 15 p

AVR 5

LUNDS TEKNISKA HÖGSKOLA
Lunds universitet
LTH Ingenjörshögskolan vid Campus Helsingborg

Styrning av trafikljus.

Syftet med laborationen är att styra en trafikkorsning med hjälp av en mikroprocessor. Trafikkorsningen består av en huvudgata (H) och en enkelriktad tvärgata (T). Huvudgatan har ett ljus för skyddad vänstersväng, och tvärgatans trafik kan regleras med så kallad nattkoppling, som ger grönt bara när trafik kommer (simuleras av knappen CAR). Övergångsstället kan styras med en knapp för gående eller helt autonomt i takt med ljussekvensen för trafiken.



Erik Larsson

Förberedelser.

Läs igenom laborationen och gör ett lösningförslag till hemuppgiften. Studera koden som ska användas som bas till ditt program. Koden finns på kurshemsidan.

Hemuppgift, funktionen countDown: "äggklocka".

I programmeringssammanhang används ofta så kallade ***time out***. Vi vill ha en funktion som fungerar precis som en äggklocka (när man kokar ägg). Klockan ställs på ett visst värde, den räknar ner och när tiden gått till noll så stannar den och något händer.
Vi ska skriva en funktion som fungerar så här:

- Räkna ner en 8-bitarsvariabel med ett **om** värdet inte är noll.
- Adressen till variabeln är inparameter till funktionen.

Tips och hjälp:
Om vi låter funktionen heta countDown, så kan den se ut så här:

```
void countDown(char *address){
  ??????????????;
}
```

"char *address" innebär att parametern som tas emot ska vara en "pekare" till en character; alltså en adress till ett 8-bitarstal. I vår processor kommer denna adress att vara ett 16-bitarstal, men det är inget som vi behöver bry oss om. Funktionen ska alltså kunna användas på vilken 8-bitarsvariabel som helst. Finessen med pekare är att man inte behöver hålla reda på vad adressen är för sorts tal; använder man en dator med 32-bitarsadresser så håller kompilatorn reda på det.

Inne i funktionen ska **det värde som `address` pekar på minskas med ett tills det blir noll. Och "det värde som `address` pekar på"** skrivs i C: **`*address`**, alltså en stjärna framför variabelnamnet. När man använder funktionen måste man ha tag i adressen till en variabel (den ska ju vara parameter).
"adressen till variabeln `Value`" skrivs i C: **`&Value`**

- ❒ Skriv färdigt funktionen!

Laborationsuppgifter:

Uppgift 1. Kontroll av den elektriska inkopplingen.

Skapa ett nytt projekt för laborationen. På kurshemsidan finns ett skal till laborationen. Använd skalet som källkodsfil i uppgiften. Skalet kommer senare kompletteras.

Det första att göra är att undersöka vilka bitar i portarna som är kopplade till de olika lamporna. Använd I/O View fönstret för att se hur lamporna styrs. Kryssa i tabellen. Jämför med tidigare laborationer där 7-segment displayen undersöktes.

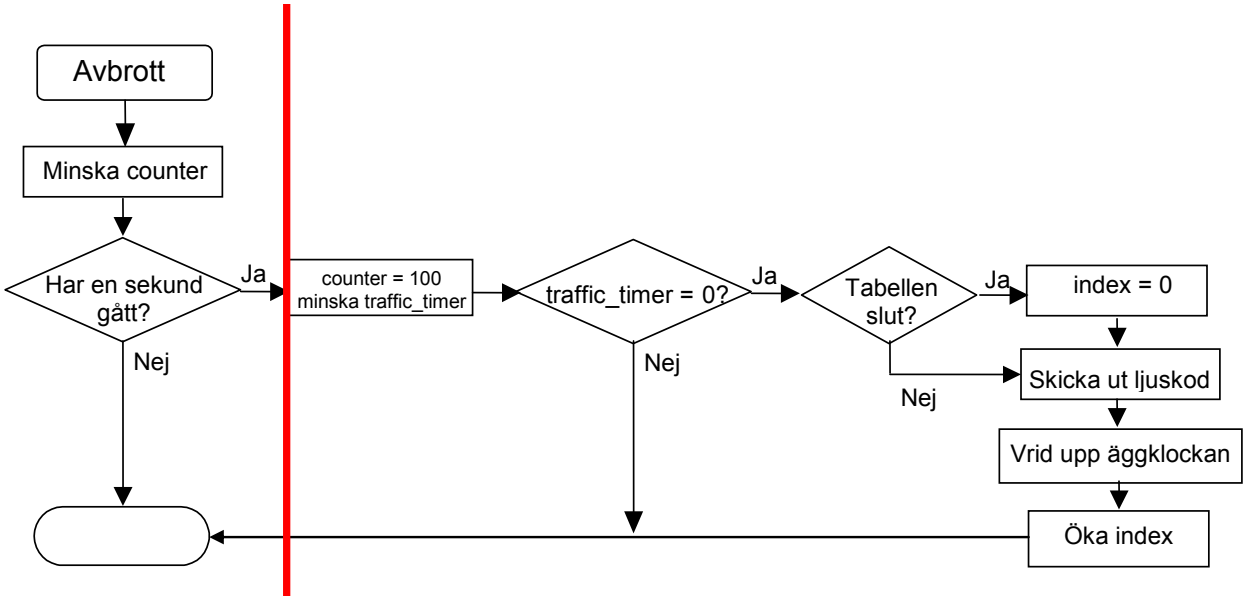
Lampor	PORTA								PORTB								PORTC								PORTD							
H - rött																																
H - gult																																
H - grönt																																
H - vänstersväng																																
T - rött																																
T - gult																																
T - grönt																																
ÖG - rött																																
ÖG - grönt																																
Lampa WAIT																																
Knapp WAIT												x																				
Knapp BLINK													x																			
Knapp NIGHT														x																		
Knapp CAR															x																	

När du fyllt i tabellen ovan, ser du att lamporna är utspridda på flera portar. Det hade varit enklare om, t.ex., alla stora lysdioder varit anslutna till samma port. Då hade vi kunnat uppdatera dem med en och samma utmatning. Att det blir så här är något som man ofta inträffar i konstruktioner. Det är ledningsdragningsproblem som gjort att lamporna blivit inkopplade på detta sätt. När vi styr trafikljusen ska vi ändå lägga ljuskoderna i en tabell av 8-bitarsvärden. Så här organiserar vi bitarna:

	T rött	T gult	T grönt	H vä-sväng	H rött	H gult	H grönt

Att sedan skicka ut bitarna till rätt plats i rätt port kommer vi hantera i uppgift 4.

- ❒ Resultatet av denna uppgift är att du nu kan komplettera funktionen **`initIO`** där portarnas riktningar ställs in.



Avbrottsfunktionen **`SIG_OUTPUT_COMPARE1A`**

Ovan finns en flödesplan för vad som ska hända i avbrottsfunktionen. Det som återfinns till vänster om det röda strecket finns i skalet som du kan ladda ner från hemsidan. Uppgifterna som följer, går ut på att bygga till de delar som saknas.

Uppgift 2. Funktionen countDown.

För att hantera tidhållningen i trafikkorsningen måste funktionen **`countDown`** skrivas.

- ❒ Skriv funktionen enligt hemuppgiften.
- ❒ Anropa den i avbrottsfunktionen med adressen till traffic_timer som parameter.

Avbrottsrutinens nyttiga arbete utgörs nu bara av två funktionsanrop till **`countDown`**.
Testa programmet!

Uppgift 3. En läslig tabell för ljuskoder.

Trafikkorsningen ska under laborationen styras så att följande värden matas ut till ljusen i de olika portarna.
Vi lägger ljuskoderna i en tabell enligt följande format:

	T rött	T gult	T grönt	H vä-sväng	H rött	H gult	H grönt

❒ Fyll i tabellen!

Tvärgata	Huvudgata	Tid	Utmatat värde i binär form							
			bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Rött	Gult	1s								
Rött	Rött	1s								
Rött,gult	Rött	1s								
Grönt	Rött	5s								
Gult	Rött	1s								
Rött	Rött	1s								
Rött	Rött, gult	1s								
Rött	Grönt	10s								

Skapa en tabell som innehåller dessa värden. Den kan se ut så här:

```
const char light_table[][2] = { {  1, main_yellow     },
                                {  1, red_red         },
                                {  1, cross_redyellow },
                                {  5, cross_green     },
                                {  1, cross_yellow    },
                                {  1, red_red         },
                                {  1, main_redyellow  },
                                { 10, main_green      },
                                {  0, 0               }
                              };
```

Tabellen består av en kolumn med tider och en kolumn med ljuskoder. Dessa koder är angivna med förkortningar för att det ska vara lättare att förstå.

```
#define main_yellow 0b01000010
#define red_red           // Komplettera!
#define cross_redyellow   // Komplettera!
#define cross_green       // Komplettera!
#define cross_yellow      // Komplettera!
#define main_redyellow    // Komplettera!
#define main_green        // Komplettera!
```

❒ Komplettera definitionerna och skriv in dem före tabellen.

Uppgift 4. Skicka ut koden till rätt bitar i rätt portar.

De tio stora lysdioderna som ska föreställa biltrafikens ljus är inkopplade till sju bitar i ***olika*** portar. Här behövs alltså en funktion som ordnar upp så att rätt bit kommer till rätt port

T.ex. ska biten H grönt som i tabellen är placerad i den allra lägsta positionen, bit 0, skickas till bit 2 i PORTD.

- ❒ Skriv funktionen **`skicka_ut`** som gör detta. Funktionen ska ta emot ett argument som innehåller koden från tabellen och skicka ut den till rätt ställen.

```
void skicka_ut(char ut){
    PORTA = ;
    PORTC = ;
    PORTD =                                 ;
}
```

Uppgift 5. Trafikstyrningen

Trafikstyrningen ska ligga i avbrottsfunktionen enligt flödesplanen på sid. 4.
Vi behöver använda oss av:

1. Funktionen **`countDown`**.
2. Värdet i **`traffic_timer`** (variabel som är global eller static)
3. Tabellen **`light_table`**
4. Index för aktuell rad i tabellen (variabel som är global eller static)
5. Funktionen **`skicka_ut`**

- ❒ Skriv klart avbrottsfunktionen.

- ❒ Testkör! Använd gärna ensekundsintervall för alla koder till att börja med.

Uppgift 6. Övergångsstället.

På laborationskortet finns en knapp och en lysdiod under texten ”WAIT”. Det är tänkt att de ska fungera så här:

1. **Om (**någon trycker på knappen WAIT**)** så ska lampan WAIT tändas.

2. **Om (**trafikljuset **inte** är grönt på tvärgatan**)** så ska övergångsställets röda lampor tändas och de gröna släckas.

3. **Om ((**trafikljusen precis har växlat till grönt på tvärgatan**) och (**lysdioden WAIT lyser**))** så ska övergångsstället visa grönt och WAIT släckas.

- Formulera de tre punkterna som **if**-satser. Översätta från svenska till C utan att krångla till det. Tänk ut hur t.ex. sanningshalten av **`(någon trycker på knappen WAIT)`** ska avgöras. Hur formuleras ett villkor som är sant om någon trycker på knappen WAIT? Hur formuleras ett villkor som är sant om den gröna lampan på tvärgatan är släckt? (Punkt 2)
- Punkt 1och 2 placeras lämpligen i avbrottsfunktionen på ett ställe dit man kommer ofta och som inte begränsas av några andra villkor.
- Punkt 3 placeras där ljuskoden precis har växlats.

Rita in dessa tre villkorsfigurer i flödesplanen på sidan 4. Markera tydligt var de ska komma in.

Testa!

Uppgift 7. Nattkoppling.

På nätterna är det inte mycket trafik på tvärgatan. Därför vill man att huvudgatans trafik bara skall kunna avbrytas av fotgängare eller om en bil kommer på tvärgatan. Nattkoppling ställs in med knappen NIGHT (etta neråt). En bil simuleras med knappen CAR. Komplettera programmet så att det testar knapparna NIGHT och CAR samt lampan WAIT på följande sätt:

> ***Om*** *((NIGHT = 1)* ***och*** *(CAR=0)* ***och*** *(ljusen visar grönt för huvudgatan)* ***och*** *(lysdioden WAIT är släckt)) så ska sekvensen* ***stoppas****.*

Vid alla andra kombinationer ska den normala sekvensen (i uppgift 6) genomlöpas normalt. Lägg märke till att formuleringen ovan kan göras om med de Morgans lag:

> ***Om*** *((NIGHT = 0)* ***eller*** *(CAR=1)* ***eller*** *(huvudgatans gröna ljus är släckt)* ***eller*** *(lysdioden WAIT är tänd)) så ska sekvensen* ***köras****.*

Det är ungefär lika lätt att förstå båda formuleringarna. Fundera igenom båda så du är med på att det är riktiga sätt att bestämma när trafikkorsningssekvensen ska köras och stoppas.

Testa!