RAM–Random Access Memory

Minnekretser	14 s	Mars 2021
Utført av		
Dato		
Godkjent av		

☐ Åpne fila `RAM-1`.

Beskrivelse av koplingen

Klokkesignaler (CLK)

Øverst til venstre ser du impulsbryteren MANUELL. Den brukes til å gi manuelle klokkepulser.

Venderen CLK velger mellom manuelle klokkepulser og klokkepulser fra pulsgeneratoren AUTO. R2 er en PULLUP resistor som sørger for at nivået fra impulsbryteren MANUELL ligger HØY når bryteren ikke er betjent. Uten R2 vil nivået være flytende (hverken HØY eller LAV).

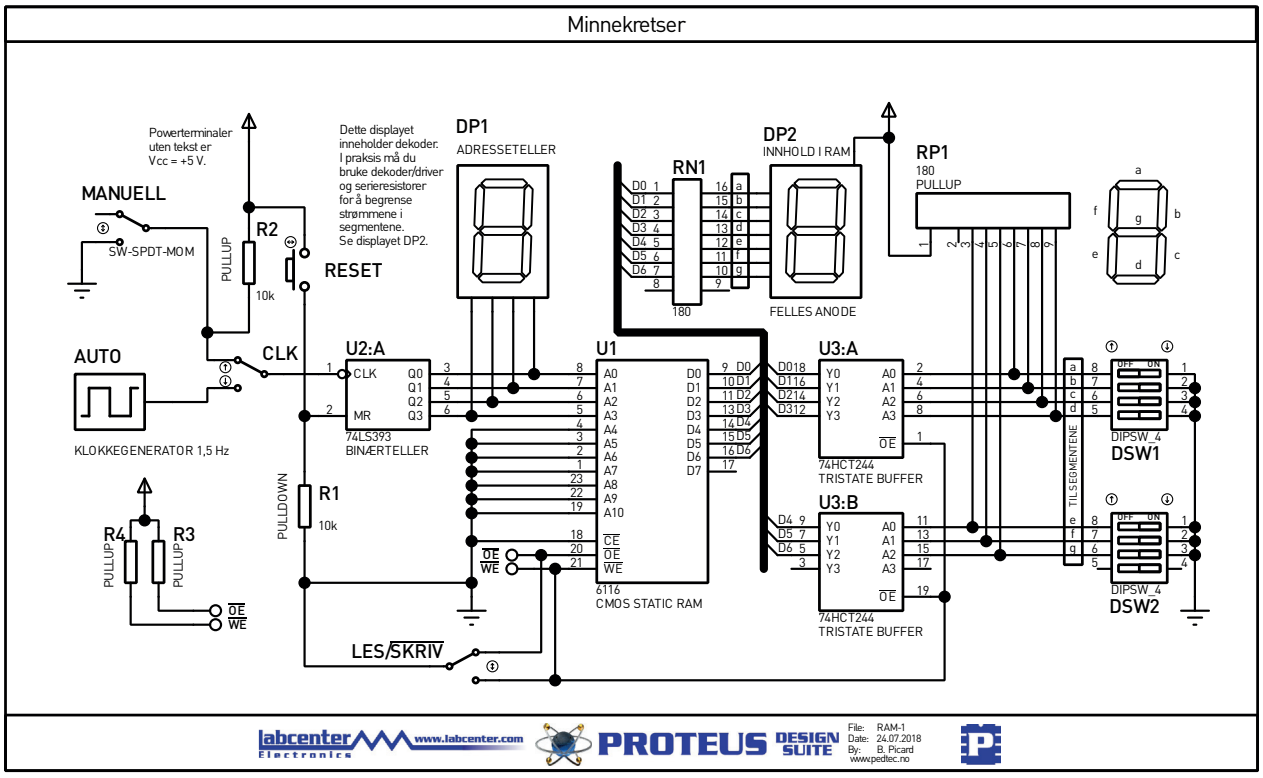
Adresseteller (U2:A)

U2:A er 4 bits binærteller. Den skal generere adresser til RAMet. Impulsbryteren RESET nullstiller telleren. Når RESET ikke er aktiv ligger MR på telleren LAV via PULLDOWN resistoren R1.

❑ Ting du skal utføre vil være merket med en firkant.

☑ Lag en hake i firkanten etter hvert som du går fram, så har du oversikt over hvor langt du er kommet.

Du kan fylle ut direkte i PDF-dokumentet der du ser gule felt.



Adresseteller (DP1)

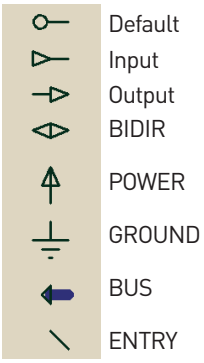
DP1 viser status på RAMets (U1) adresseinngangene A0–A3. Merk at dette displayet kun er ment som et hjelpemiddel for å se adressene.

Displayet har innebygd dekoder og driver og trenger derfor ikke serieresistorer for å begrense strømmen i segmentene. Det finnes displayer med innebygde serieresistorer og dekoder, men de ”koster flesk”.

Skal denne kretsen koples opp, må du bruke dekoder/driver og serieresistorer for å begrense strømmene i segmentene. Se displayet DP2.

Terminaler

TERMINALER med samme navn har elektrisk forbindelse.



Kontrollinngangene $\overline{OE}$ (20) og $\overline{WE}$ (21) på RAMet (U1) er
koplet til terminaler som er gitt samme navn.

Nederst til venstre i skjemet ser du R3 og R4 er koplet til terminaler med navnene $\overline{OE}$ og $\overline{WE}$.

Det er altså forbindelse mellom R3 og $\overline{OE}$ (20) og mellom R4 og $\overline{WE}$ (21).

Terminaler

TERMINALER med samme navn har elektrisk forbindelse.

Bruk terminaler i skjemaene. Det gir bedre oversikt og det er lettere å gjøre endringer.

TERMINALS er *ikke* fysiske terminaler. Skal du for eksempel kople en rekkeklemme til GND og VCC må du hente rekkeklemme fra biblioteket.

RAM (U1)

Adresseinngangene A4–A10 skal ikke brukes og er derfor jordet.

Kontrollsignaler er $\overline{CE}$, $\overline{OE}$ og $\overline{WE}$. $\overline{CE}$ (Chip Enable) ligger fast LAV (aktivt nivå).

På høyre side av RAMet har pinnene navnene D0–D7. Dette er både data inn- og og data-utganger.

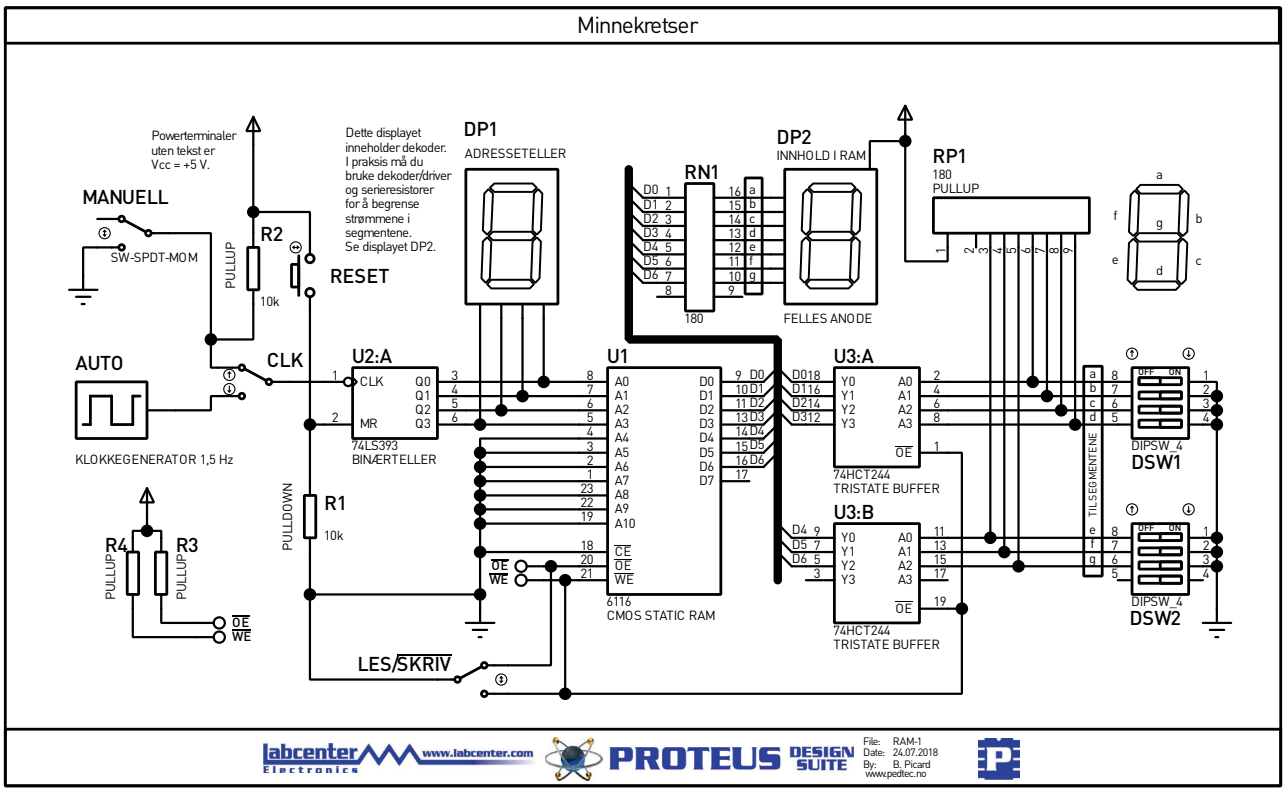
$\overline{OE}$ og $\overline{WE}$ er koplet til impulsbryteren LES/$\overline{SKRIV}$. Den har to stillinger: ubetjent og betjent. Når den ikke er betjent ligger $\overline{OE}$ LAV og $\overline{WE}$ HØY. D0–D7 på RAMet er da utganger.

Impulsbryteren er koplet til PULLUP-resistorene R3 og R4 for å sikre et definert logisk nivå, HØY eller LAV.

Når $\overline{WE}$ er ubetjent er D0–D7 data-utganger ($\overline{OE}$ er aktiv). Når $\overline{WE}$ er betjent, er D0–D7 data innganger ($\overline{OE}$ er inaktiv). Da kan vi skrive data inn i RAMet.

TRISTATE BUFFER (U3:A og U3:B)

Data som skal skrives inn i RAMet legges på inngangene av U3 med DIP-switchene DSW1 og DSW2.



Så lenge $\overline{OE}$ på U3 er HØY er utgangene høyohmige (verken HØY eller LAV).

Når LES/$\overline{SKRIV}$ legges LAV, går $\overline{OE}$ på U3 LAV og dataene på inngangene A0–A3 legges ut på utgangene Y0–Y3.

$\overline{OE}$ på U3 og $\overline{WE}$ på U1 er sammenkoplet, så når LES/$\overline{SKRIV}$ legges LAV, går $\overline{WE}$ på U1 LAV og innskriving av data på D0–D7 skjer.

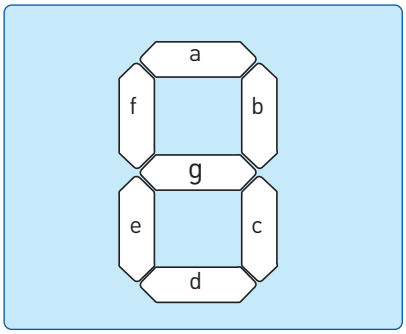
RAMet brukt som sjusegment dekoder

Displayet DP2 er koplet til utgangene på RAMet via resistorpakken RN1. Resistorene i pakken er koplet til hvert segment og fungerer som strømbegrensere. Displayet er av typen felles anode. Det betyr at utgangene på RAMet må ligge til LAV for at det skal gå strøm i segmentene.

Øving 1

I denne øvingen skal vi programmere RAMet slik at det viser hvilken adresse det har på inngangen, altså 0–15_{10}. Heksadesimalt skal 10–15_{10} vises som A-b-C-d-E og F.

Når du skal fylle ut tabellen er det kanskje lettest å finne ut vilke aktive segmenter det er flest eller færrest av. I første rad skal displayet vise tallet 0. Alle segment untatt g skal være lave. I neste rad skal bare b og c lyse.



PLAY STEP PAUSE STOP

Programmering av RAMet

Framgangsmåte

☐ Start animasjonen (klikk på PLAY-knappen) eller trykk på F12.

Du må ikke stoppe animasjonen! Gjør du det mister du programmet.

☐ Sett venderen CLK til manuelle klokkepulser.

☐ Reset telleren hvis den ikke viser 0 på DP1.

Programmeringsunderlag

Adresseteller DP1	Display DP2	DSW1 8 7 6 5	DSW2 8 7 6	
		a b c d	e f g	<-- Segment
0	0	0 0 0 0	0 0 1	
1	1	1 0 0 1	1 1 1	
2	2	0 0 1 0	0 1 0	
3	3			
4	4			
5	5			
6	6			
7	7			
8	8			
9	9			
A	A			
b	b			
C	C			
d	d			
E	E			
F	F			

Display 2 (DP2) skal vise status på adresseteller, altså det samme som vises på DP1.

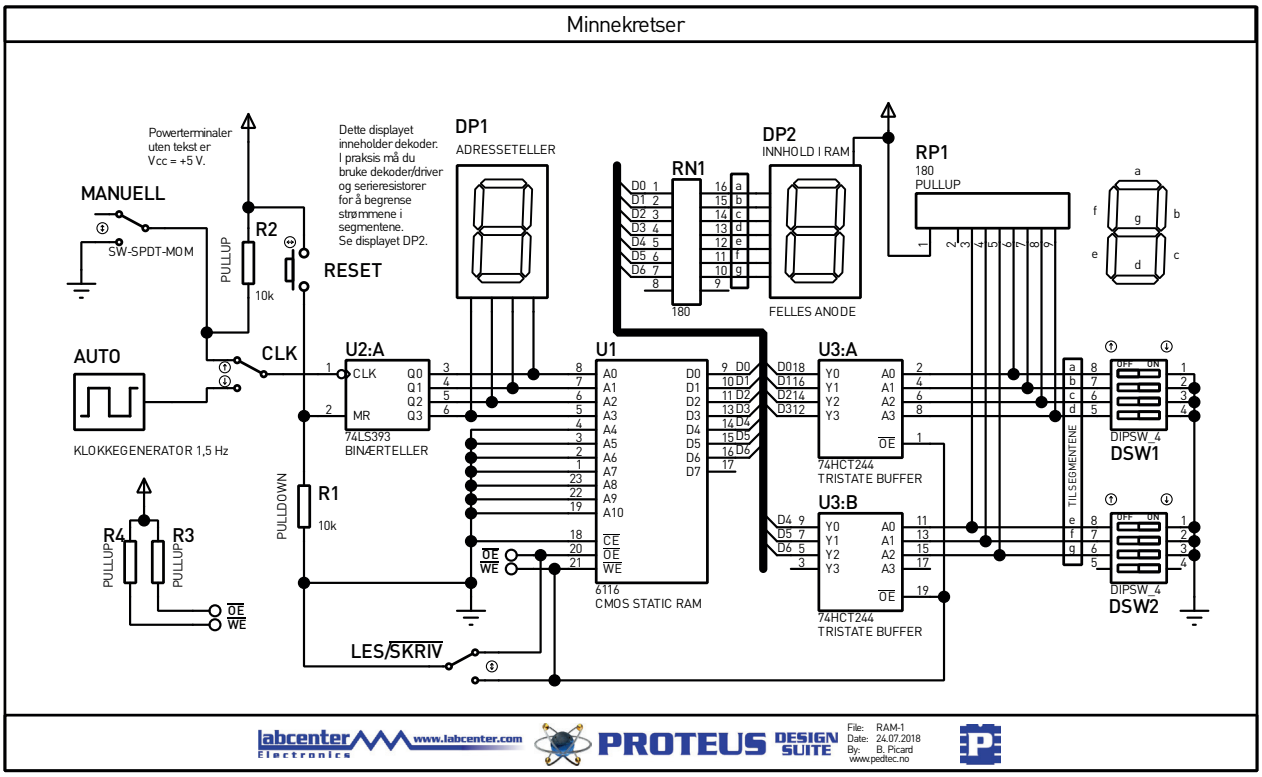
☐ Fyll ut resten av tabellen.

☐ Sett DSW1 og DSW2 til de posisjonene tabell 1 viser.

☐ Klikk på $\overline{WE}$.

☐ Sjekk at displayet DP2 viser som forventet.

☐ Avanser telleren med ny puls på MAN.



☐ Legg inn nye data på DSW1 og DSW2 og klikk på $\overline{\text{WE}}$.

☐ Fortsett til du har programmert for alle 16 adressene.

☐ Klikk på CLK slik at adressetelleren går automatisk.

☐ Sjekk at alt stemmer. Gjør det ikke det, må du stille om til manuelle klokkepulser og programmere på nytt.

☐ Stopp animasjonen og start den igjen.

☐ Forklar hvorfor RAMet nå har fått nytt innhold.

EPROM–Erasable Programmable Read Only Memory

EPROM, eller elektronisk programmerbare read-only minne, er et lagringssystem som brukes i databehandlingen. EPROM minnet kan holde data uten en konstant strømforsyning, og kan omprogrammeres elektrisk.

EPROM lagrer elektroniske data som må være tilgjengelig, selv om strømmen er slått av. EPROM minnet er også brukt i andre elektroniske enheter så som sanntid klokker. I disse enhetene, lagrer EPROM viktig informasjon som kalibreringsdata.

Intel Hex format

Et format som brukes blant annet til å lagre data i EPROMer.

Tabell 1 The 6 fields which comprise a Hex-record are defined as follows:

Intel Hex format

INTRODUCTION

Intel's Hex-record format allows program or data files to be encoded in a printable (ASCII) format. This allows viewing of the object file with standard tools and easy file transfer from one computer to another, or between a host and target. An individual Hex-record is a single line in a file composed of many Hex-records.

HEX-RECORD CONTENT

Hex-Records are character strings made of several fields which specify the record type, record length, memory address, data, and checksum. Each byte of binary data is encoded as a 2-character hexadecimal number: the first ASCII character representing the high-order 4 bits, and the second the low-order 4 bits of the byte.

Field	1	2	3	4	5	6
	Startcode	Bytecount	Address	Type	Data	Checksum
Description	An ASCII colon, ":"	The count of the character pairs in the data field	The 2-byte address at which the data field is to be loaded into memory	00, 01, or 02	From 0 to n bytes of executable code, or memory loadable data. n is normally 20 hex (32 decimal) or less	The least significant byte of the two's complement sum of the values represented by all the pairs of characters in the record except the start code and checksum

Each record may be terminated with a CR/LF/NULL. Accuracy of transmission is ensured by the byte count and checksum fields.

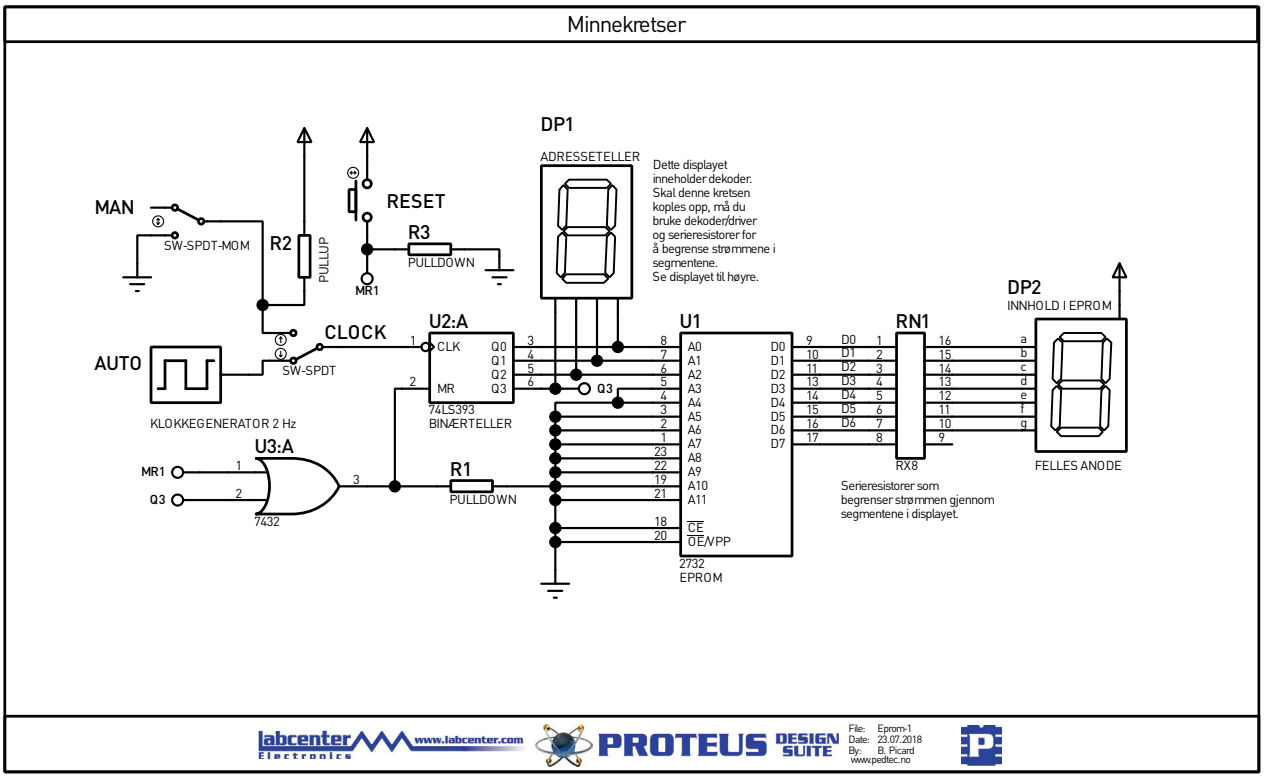
HEX-RECORD TYPES

Tabell 2 There are three possible types of Hex-records.

00	A record containing data and the 2-byte address at which the data is to reside.
01	A termination record for a file of Hex-records. Only one termination record is allowed per file and it must be the last line of the file. There is no data field.
02	A segment base address record. This type of record is ignored by Lucid programmers.

Eksempel 1

☐ Åpne fila `Eprom-1`.



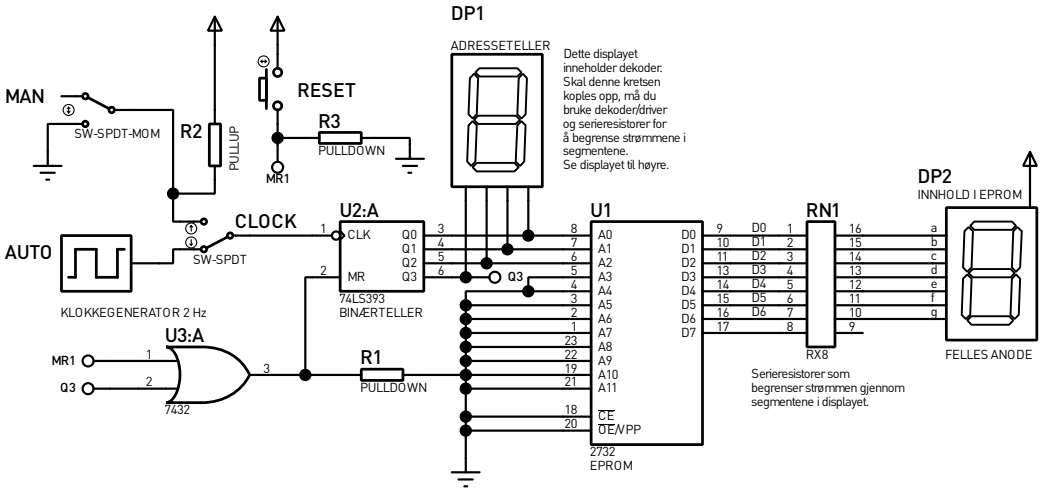
Klokkesignaler (CLOCK)

Øverst til venstre ser du impulsbryteren MAN. Den brukes til å gi manuelle klokkepulser.

Venderen CLOCK velger mellom manuelle klokkepulser og klokkepulser fra pulsgeneratoren AUTO.

Adresseinnganger (DP1)

DP1 viser status på adresseinngangene A0–A3. Merk at dette displayet kun er ment som et hjelpemiddel for å se adressene. Displayet har innebygd dekoder og driver og trenger derfor ikke serieresistorer for å begrense strømmen i segmentene. Skal denne kretsen koples opp, må du bruke dekoder/driver og serieresistorer for å begrense strømmene i segmentene. Se displayet DP2 til høyre i skjemaet. Det finnes displayer med innebygde serieresistorer og dekoder, men de «koster flesk».



DISPLAY DP2

Som du ser er displayet DP2 koplet til utgangene på EPROMen via resistorpakken RN1. Resistorene i pakken er koplet til hvert segment og fungerer som strømbegrensere. Displayet er av typen felles anode. Det betyr at utgangene på EPROMen må ligge til LAV for at det skal gå strøm i segmentene.

ELLER-porten U3:A

Når utgangen på eller-porten går HØY vil telleren U2:A nullstilles.

TELLER U2:A

Utgang Q3 går høy når telleren har avansert til $8_{10} = 1000_2$. Da vil telleren resettes.

Q3 ELLER **MR1** vil resette telleren.

Øving 1

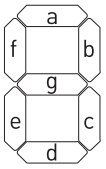
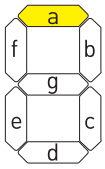
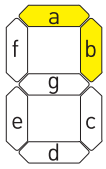
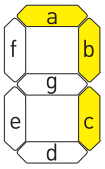
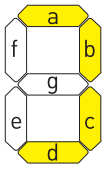
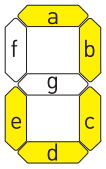
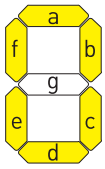
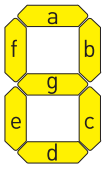
Tellesyklus

Utgangspunktet er at alle segmentene er slukket, Vi ønsker å tenne ett segment av gangen. Segmentene skal forbli tent helt til alle er tent. Deretter skal alle slukke. Se tabell 3.

Skrive HEX-fil til EPROMen

For oversiktens skyld gjengir vi tabell 1.

Displayvisning for de forskjellige adressene

Adresse	Desimal	0	1	2	3	4	5	6	7
	Binær	0000	0001	0010	0011	0100	0101	0110	0111
	Hex	00	01	02	03	04	05	06	07
									

Data for de forskjellige adressene

	Adresse	0	1	2	3	4	5	6	7
Data	Binær	111 1111	111 1110	111 1100	111 1000	111 0000	110 0000	100 0000	000 0000
	Hex	7F	7E	7C	78	70	60	40	00

Nedenfor er første linje i programmet (alle segmentene er slukket):

1. **Start Code**: Semikolon "**:**".
2. **Byte count**: Antall tegnpar (hex) i datafeltet "**01**".
3. **Address**: 2 byte (**00 00**) adresse hvor dataene skal lagres.
4. **Record Type: 00** (normale data).
5. **Data: 7F.**
6. **Checksum**: Sum av alle hexpar (untatt Start Code ":"), pluss Checksum skal være 00. I tabell 5 har vi regnet ut checksum.

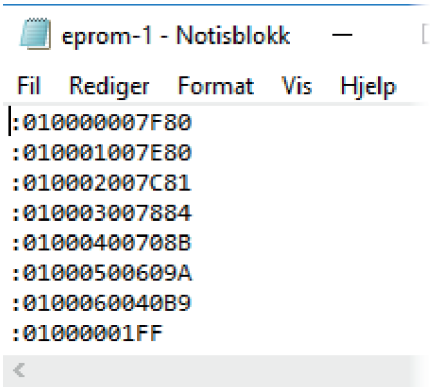
Beregning av Checksum

Field	Address	00	01	02	03	04	05	06	07
2	Byte count	01	01	01	01	01	01	01	00
3	Adress (MS)	+ 00	+ 00	+ 00	+ 00	+ 00	+ 00	+ 00	+ 00
	Address (LS)	+ 00	+ 01	+ 02	+ 03	+ 04	+ 05	+ 06	+ 00
4	Record Type	+ 00	+ 00	+ 00	+ 00	+ 00	+ 00	+ 00	+ 00
5	Data	+ 7F	+ 7E	+ 7C	+ 78	+ 70	+ 60	+ 40	
	Sum	80	80	7F	7C	75	66	47	01
6	Checksum	100 – 80=80	100 – 80=80	100 – 7F = 81	100 – 7C = 84	100 – 75 = 8B	100 – 66 = 9A	100 – 47 = B9	100 – 01 = FF

☐ Åpne Notepad og skriv inn innholdet i tabellen under.
OBS! Det skal ikke være mellomrom mellom tegnene.

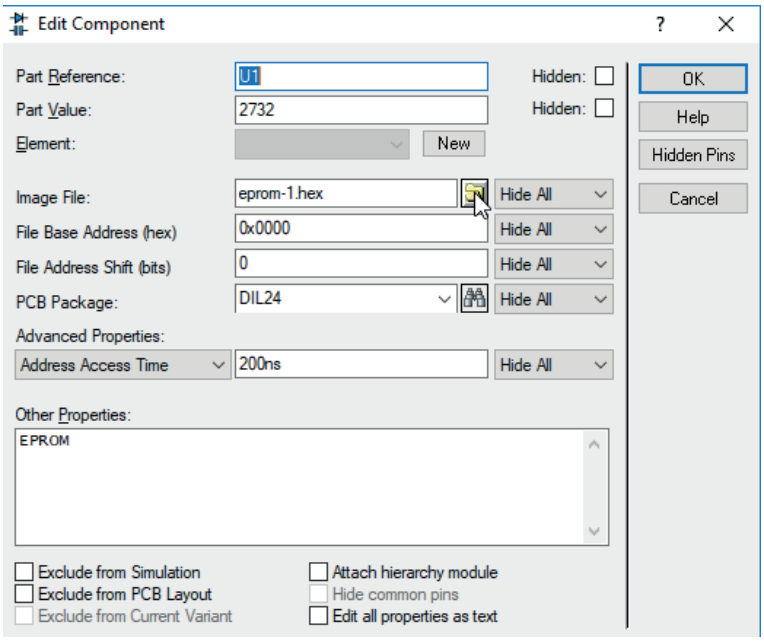
Innhold i EPROM

Startcode	Bytecount	Address		Type	Data	Checksum
		MSB	LSB			
:	01	00	00	00	7F	80
:	01	00	01	00	7E	80
:	01	00	02	00	7C	81
:	01	00	03	00	78	84
:	01	00	04	00	70	8B
:	01	00	05	00	60	9A
:	01	00	06	00	40	B9
:	00	00	00	01		FF



☐ Lagre fila som `eprom-1.hex`.

☐ Dobbeltklikk på U1.



Fasit
`F-Eprom-1.hex`

```
:010000007F80
:010001007E80
:010002007C81
:010003007884
:01000400708B
:01000500609A
:0100060040B9
:01000001FF
```

☐ Klikk på mappesymbolet til høyre for feltet Image File.

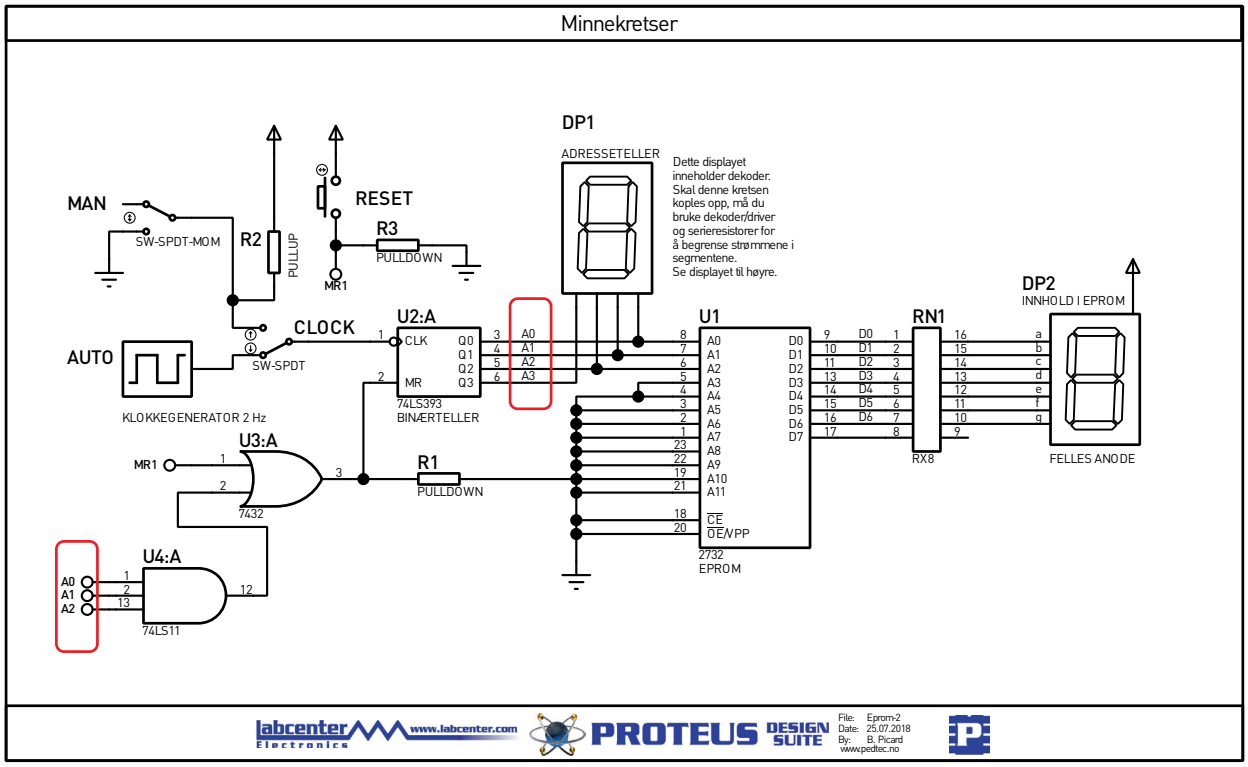
☐ Let fram fila `eprom-1.hex` som du lagret og dobbeltklikk på den.

☐ Lukk dialogboksen og start animasjonen.

☐ Sett CLOCK til AUTO og kontroller at displayet viser riktig.

Øving 2

☐ Åpne Eprom-2.



☐ Telleren U2:A nullstilles fra ELLER-porten U3:A.

☐ Forklar.

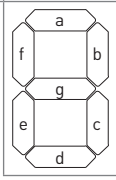
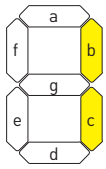
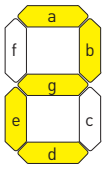
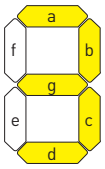
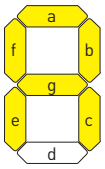
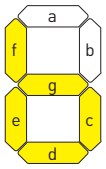
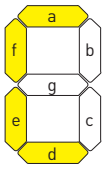
☐ Skriv et program som viser denne rekkefølgen på DP2:

8 (slukket) - 1 - 2 - 3 - A - b - C

Sekvensen skal starte med alle segment avslått.

Se neste side →

Displayvisning for de forskjellige adressene

Adresse	Desimal	0	1	2	3	4	5	6
	Binær	0000	0001	0010	0011	0100	0101	0110
	Hex	00	01	02	03	04	05	06
								

Data for de forskjellige adressene

	Adresse	0	1	2	3	4	5	6
Data	Binær	111 1111	111 1001	010 0100	011 0000	000 1000	000 0011	100 0110
	Hex	7F	79	24	30			

☐ Fyll ut resten av feltene i tabellen over.

Beregning av Checksum

Field	Address	00	01	02	03	04	05	06
2	Byte count	01	01	01	01	01	01	01
3	Address (MS)	+ 00	+ 00	+ 00	+ 00	+ 00	+ 00	+ 00
	Address (LS)	+ 00	+ 01	+ 02	+ 03	+ 04	+ 05	+ 06
4	Record Type	+ 00	+ 00	+ 00	+ 00	+ 00	+ 00	+ 00
5	Data	+ 7F	+ 79	+ 24	+ 30			
	Sum	= 80	= 7B	= 27	= 34			
6	Checksum	100 – 80 = 80	100 – 7B = 85	100 – 27 = D9	100 – 34 = CC			

☐ Fyll ut resten av feltene i tabellen over.

Innhold i EPROM

Start-code	Byte-count	Address		Type	Data	Checksum
:	01	00	00	00	7F	80
:	01	00	01	00	79	85
:	01	00	02	00	24	D9
:	01	00	03	00	30	CC
:	01	00	04			
:	01	00	05			
:	01	00	05			
:	00	00	00			

☐ Fyll ut resten av feltene i tabellen over.

☐ Lagre fila som `Eprom-2.hex`.

☐ Dobbeltklikk på U1.

☐ Klikk på mappesymbolet til høyre for feltet Image File.

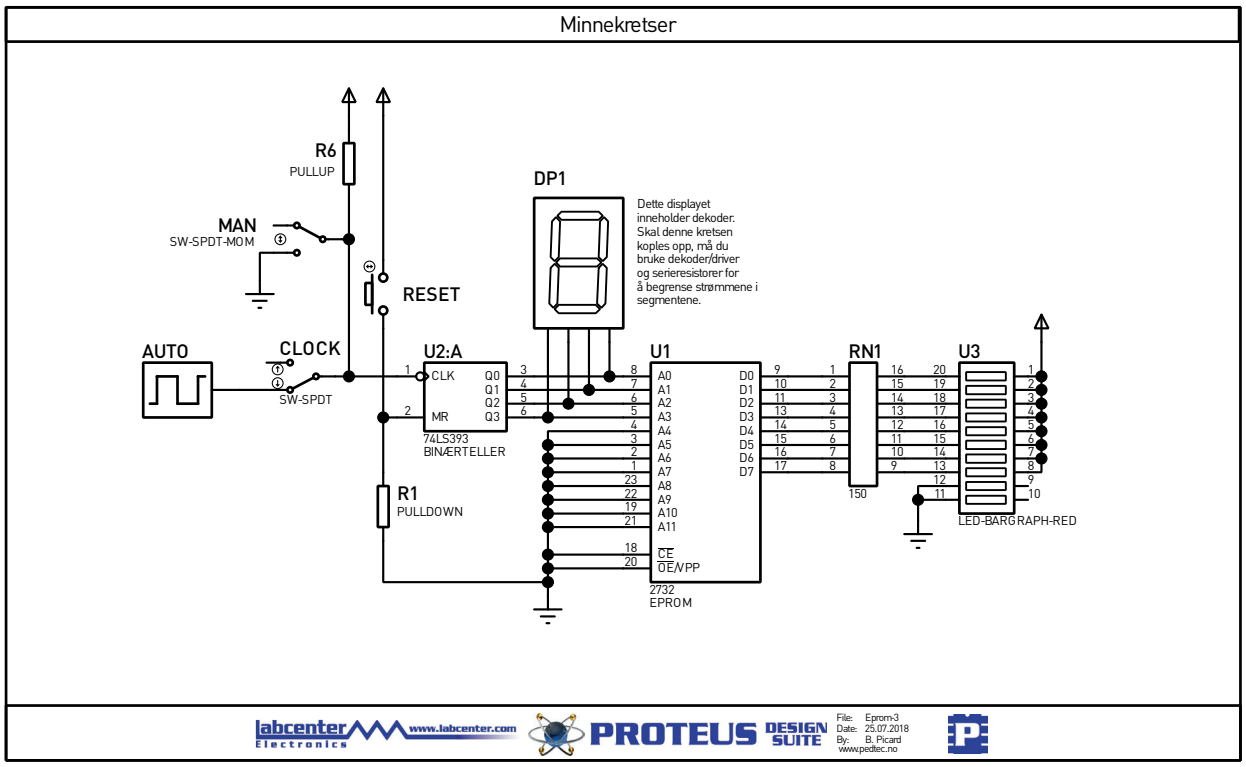
☐ Let fram fila `Eprom-2.hex` som du lagret og dobbeltklikk på den.

☐ Lukk dialogboksen og start animasjonen.

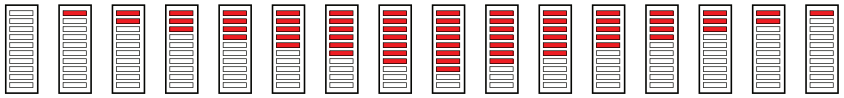
☐ Sett CLOCK til AUTO og kontroller at displayet viser riktig.

Øving 3

☐ Åpne `Eprom-3`.



☐ I denne øvingen skal du programmere epromen slik at BAR-GRAPH-displayet fungerer som løpelys.



Se neste side . . .

Løpelys

Adresse (16)	Bargraph	Adresse (16)	Bargraph	Adresse (16)	Bargraph	Adresse (16)	Bargraph
00		04		08		0C	
01		05		09		0D	
02		06		0A		0E	
03		07		0B		0F	

☐ Fyll ut tabellen under.

Program for løpelys

Start-code	Byte-count	Address		Type	Data	Checksum
			00			
			01			
			02			
			03			
			04			
			05			
			06			
			07			
			08			
			09			
			0A			
			0B			
			0C			
			0D			
			0E			
			0F			
			00			

☐ Skriv programmet og lagre som `eprom-3.hex`.

☐ Start animasjonen og test.