CW ID-Circuit

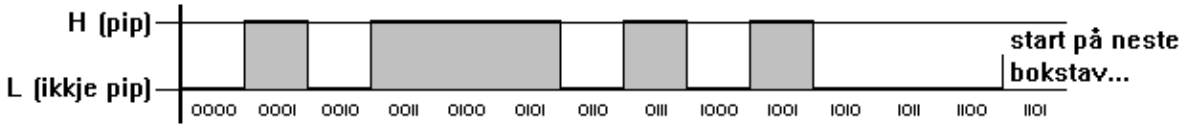
Erik Grindheim
LA9SJA 1997

Dette er eit prosjekt basert på idèen om å lage ein krets som kan morse ut ein radiostasjon sin identifikasjon automatisk. For mitt vedkommande vil det seie at kretsen skal morse ut "LA9SJA" som er mitt kallesignal på radioamatør-banda.

Prosjektets kategori er "eksperimentell forskning", og eg vil her beskrive dei viktigste fasane i prosjektets framdrift. Ein del emner vil også bli forklart i detalj.

Etter ein god del tankearbeid om korleis ein slik krets kunne byggast opp kom eg fram til at eg kunne bruke ein digital teljar til å telje oppover. Verdien til teljaren bestemte i ein dekodar om pipetonen i morsen skulle vere av eller på (L/H).

Figuren viser eksempel på dekodarens utgang i starten av opp-teljinga, første bokstav, L, er vist, og den er ._.. på morse:



For å morse gjennom heile kallesignalet (LA9SJA) må teljaren gå opp til minst 75, dvs den må vere 7-bits.

DEKODAR

Dekodaren måtte altså hatt 7 inputs og ein output. Vha. Karnaughdiagram fann eg likninga som beskreiv dekodaren. Her er kretsens funksjon (forenkla):

B'(E(A'G+C'D'F')+C(D'E'F'+F(D+E))) + B(CF'(D+E)+E'(C'D'F+G)) + G(D'(CE+C'E')+A'F')

...der input A er teljarens MSB og input G er LSB. A-invers = A'

Dekodaren blei bygd opp og testa i *Electronics Workbench*. Problemet var berre det at i praksis ville dekodaren bestå av omtrent 12 stk CMOS ICar i 4000-serien. Kretsen ville blitt enorm...

EPROM

Etter ei tid kom eg til å tenkje på moglegheiten for å bruke ein minnekrets (ROM) til å foreta dekodinga. Funksjonen er jo i utgangspunktet enkel; Teljarens verdi går inn på adresseinngangane på ROM'en, og på dei enkelte adressene ligg utgangens verdi (H/L) lagra.

Strengt tatt kunne altså ROM'ens kapasitet vore på kun 128 bits for å kunne gi ut ein verdi (H/L) for alle teljarens verdiar. (7 inputs, ein output)

Ettersom eg hadde tilgjengeleg mange forskjellige EPROM'ar i 27XXX-serien var det naturleg å prøve å programmere ein slik. Ved feilprogrammering kan desse chipane slettast med UV-lys og programmerast på nytt.

Ein slik EPROM er byte-organisert, dvs. at på kvar i adresse i chipen er det lagra ein byte, dvs *8 bits.* Derfor har chipen 8 utgangar.

Lagringskapasiteten er mange, mange ggr. større enn naudsynt; antal inngangar varierer frå 11 til 15. Sju hadde vore nok. I_0 til I_6 skal brukast, og I_7 og oppover blir lagt låge. På denne måten brukar ein kun dei første 128 ($=2^7$) adressene.

PROGRAMMERING

Utstyret eg brukte til å "brenne" med var ein PC med *Allmax+* software tilkopla ein *Allmax+* universal programmerings-enhet.

Software'en (*Allmax+* programmet) fungerer slik at for kvar einskild adresse skriv ein inn ein tosifra heksadesimal verdi som uttrykkjer H/L-tilstandane som dei åtte bit'a (utgangane) skal ha.

Eksempel:
På adresse 0110110011101 (3485_D eller $D9D_H$) skal utgangane ha
følgjande verdiar O_7 - O_0: HLHHLHLL
På binærform blir dette: 1011 0100
...og på heksadesimal: B 4

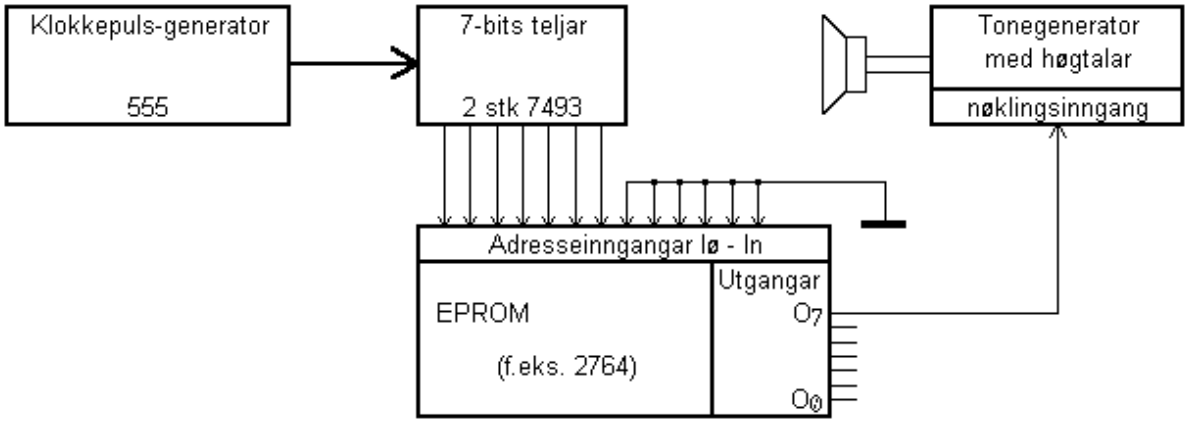
På dei første chipane brente eg alle utgangane like på dei enkelte adressene. Enten var alle "på" eller "av" dvs. 11111111 eller 00000000. HEX-verdiane var altså 00 (av) eller FF (på).

Tabellen under viser programmeringa av L (Lima) som er første bokstav:

Adresse	Utgangane	HEX	Bokstav
0	oooooooo	ØØ	standby
1	ıııııııı	FF	Lima
2	oooooooo	ØØ	Lima
3	ıııııııı	FF	Lima
4	ıııııııı	FF	Lima
5	ıııııııı	FF	Lima
6	oooooooo	ØØ	Lima
7	ıııııııı	FF	Lima
8	oooooooo	ØØ	Lima
9	ıııııııı	FF	Lima
10	oooooooo	ØØ	Mellomrom
11	oooooooo	ØØ	Mellomrom
12	oooooooo	ØØ	Mellomrom
13	start på neste bokstav (Alfa)		

FØRSTE TEST

Chipen (NEC D2764D) var programmert, og blei kopla opp saman med ein 555 (astabil vippe, klokkepuls til teljar), to stk 7493 (samankopla til 7 bits teljar) og tonegenerator (som lot seg nøkle av chipens utgang).



Kretsen fungerte! Teljaren gjekk opp til 128 og starta om att og om att (ringteljar). Frå høgtalaren blei det morsa "LA9SJA (pause) LA9SJA (pause) LA..."
Forresten var det ein liten programmeringsfeil slik at J (Juliett ._ _ _) fekk ein "strek" som var berre 2 ggr. prikk-lengda. Det korrekte er 3 ggr.

prikk-lengda. Det var fort gjort å gå inn i *Allmax+* programmet og rette opp feilen. EPROM'en blei sletta med UV-lys og brent på nytt.

Nå følgde ei tid med ein del eksperimentering. Det blei brent fleire chipar, mellom anna 2716, 2732, 2764, 27256 og 27512. Det gjekk greitt å kople dei inn i kretsen, alle saman fungerte fint, bortsett frå at eg var borti eitt tilfelle der ein chip var øydelagt; den lot seg ikkje programmere.

Datablad på chipane ligg på Internet i PDF-format (*Acrobat Reader*).

FORBETRING AV KRETSEN

For å gjere kretsen meir anvendeleg laga eg ei SR-vippe til å starte teljaren (klokkepulsgeneratoren) med. For å få reset kretsen (stoppa klokkepulsgeneratoren og reset teljaren) etter at kallesignalet var ferdig morsa programmerte eg ein av utgangane til å vere høg opp til og med adresse 75 og låg over. Denne utgangen styrte Reset-inngangen på SR-vippa.

Når ein nå gav ein puls (manuelt, med trykkbrytar) på Set-inngangen starta klokkepulsgeneratoren, teljaren talde oppover og kallesignalet blei morsa ut. Når kallesignalet var ferdig og teljaren nådde adresse 76 blei SR-vippa reset, klokkepulsgeneratoren stoppa, og teljaren blei reset.

CMOS KOMPONENTAR

For å få ned straumforbruket og for å forenkle kretsen gjekk eg over til å bruke CMOS-kretsar. Teljaren blei ein 4024, EPROM'en blei 27c256 og til klokkepulsgenerator, SR-vippe og tonegenerator valde eg 4093.

BETRE UTNYTTING AV LAGRINGSKAPASITETEN

For å kunne lagre 16 forskjellige kallesignal og kunne velje kva for eit som skal brukast vha. DIP-switch'ar kopla eg 4 av dei ubrukte adresseinngangane til DIP-switchar som valde mellom H og L -nivå. Dei adresseinngangane som blei valde til dette var I_8 tom. I_{11}.

Områdene som dei forskjellige kallesignala lagrast på blir:

Område	*Adresser*	*Bruk av området*
0	000-07F	"LA9SJA"
1	100-17F	"LA9SJA/M"
2	200-27F	"LA9SJA/P"
3	300-37F	"de LA9SJA"
4	400-47F	"de LA9SJA/M"
5	500-57F	"de LA9SJA/P"
6	600-67F	"CODEX"
7	700-77F	"PARIS"
8	800-87F	ikkje i bruk...
9	900-97F	ikkje i bruk...
10	A00-A7F	ikkje i bruk...
11	B00-B7F	ikkje i bruk...
12	C00-C7F	ikkje i bruk...
13	D00-D7F	ikkje i bruk...
14	E00-E7F	ikkje i bruk...
15	F00-F7F	ikkje i bruk...

Dei utgangane som ikkje var i bruk (O_0-O_1 og O_3-O_6) programmerte eg for andre eventuelle framtidige bruksområder, slik at alle bit'a i kvar byte blei utnytta. Her er ein tabell som viser bruken av kvar bit:

Utgang	*Bruksområde*
0	Låg puls etter kvart enkelt tegn (etter L, etter A, etter 9, osv..)
1	Høg puls etter kvart enkelt tegn
2	Låg på reset-verdi og over
3	Høg på reset-verdi og over
4	Høg i aktiv tilstand (når *Reset-verdi > teljar-verdi > 0*)
5	Høg i kvilestilling (invers av O_4)
6	Invers keyout, går låg når det skal pipe
7	Keyout, styrer tonegeneratoren og keyout-kretsen med OC-utg.

STRAUMFORSYNING

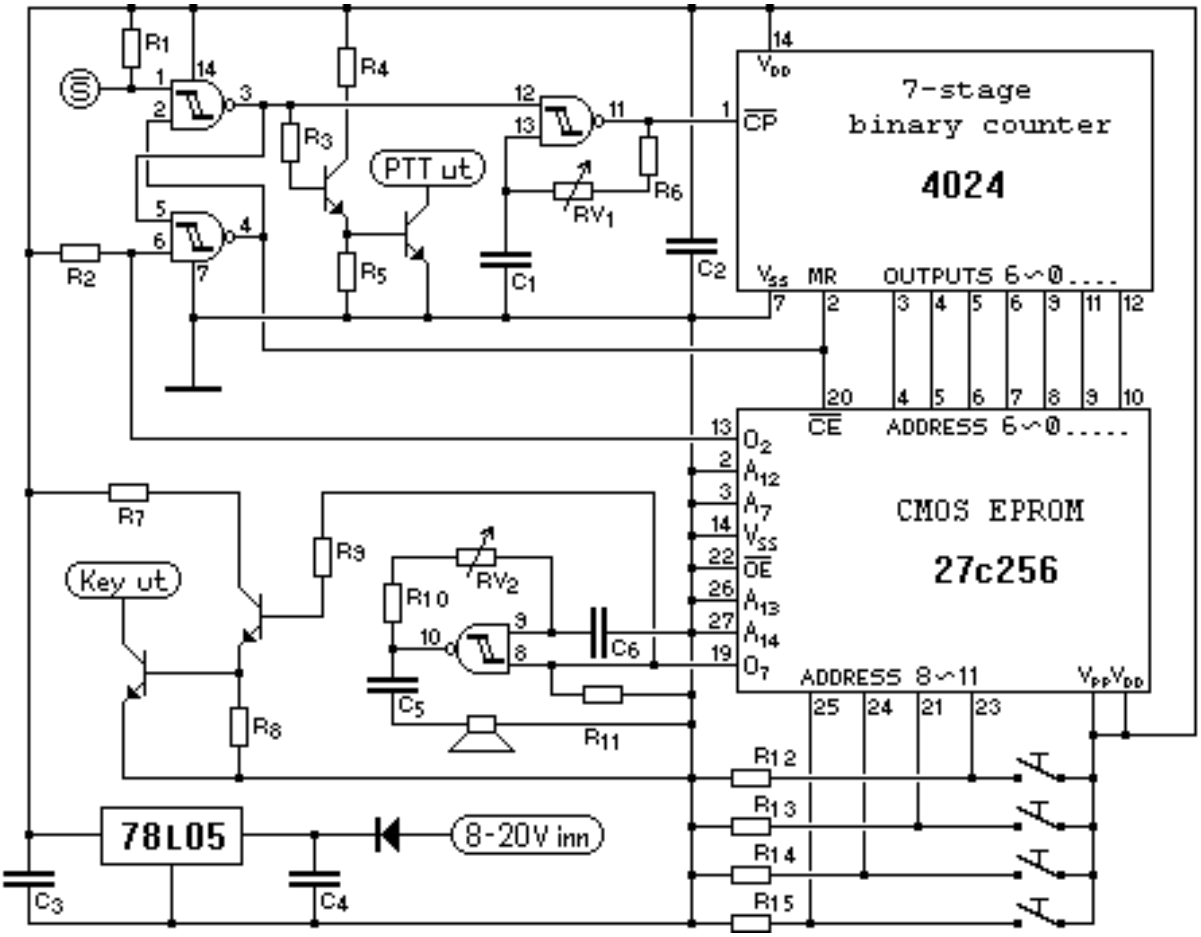
For å gi kretsane ein stabil og passande drivspenning valde eg å bruke ein 5-volts fast regulator. Ved å bruke denne kan kretsen drivast på spenningar frå 8 til 25 volt.

For å få ned straumforbruket i kvilestilling (når SR-vippa og teljaren er reset) tok eg i bruk EPROM'ens *standby* modus. *Chip Enable* -pinnen blei kopla til SR-vippa, slik at når vippa er reset blir EPROM'en "kopla ut". Dvs. straumforbruket går ned (til 100μA) og utgangane blir høgohmige.

Når kretsen ligg i kvilestilling er det totale straumforbruket <5mA.

KOPLINGSSKJEMA

Her er den endelege kretsen:



R_1 = 47 kΩ
R_2 = 800-1200 kΩ
R_3 = 180 kΩ
R_4 = 1,8 kΩ
R_5 = 1 kΩ
R_6 = 47 kΩ
R_7 = 1,8 kΩ
R_8 = 1 kΩ
R_9 = 180 kΩ
R_{10} = 40-70 kΩ
R_{11} = 800-1200 kΩ
R_{12} = 56-68 kΩ
R_{13} = 56-68 kΩ
R_{14} = 56-68 kΩ
R_{15} = 56-68 kΩ

RV_1 = 220kΩ
RV_2 = 220kΩ

C_1 = 1 µF
C_2 = 100 nF
C_3 = 100 nF
C_4 = 1 µF
C_5 = Over 400 nF
C_6 = 10 nF

$Speed_{CW}$ = 40-220 cpm

Komponentverdiane kan i mange tilfelle endrast. Dei komponentane som styrer farten på morsen og tonefrekvensen på lyden er R_6, R_{10}, RV_1, RV_2, C_1 og C_6. Her er formlane:

$$Speed_{CW} = (1 / (0{,}48(RV_1+R_6)C_1)) \text{ wpm}$$

$$f_{tone} = (1 / (0{,}58(RV_2+R_{10})C_6)) \text{ Hz}$$

PRINTPLATE

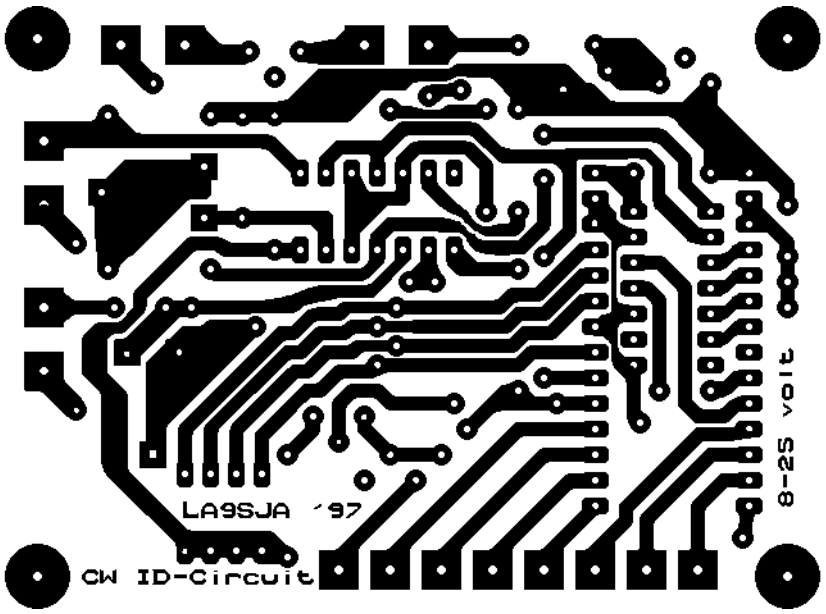
Nå som kretsen var 100% ok var det på tide å lage PCB. Til det arbeidet brukte eg eit gammalt DOS-basert CAE-program.

For å klare å komprimere plata ned til ca 6 8 cm brukte eg sokkel på EPROM'en og plasserte teljaren (4024) inni sokkelen. Printplata er tosidig, der den eine sida kun blir brukt som jordplan, common. Alle komponentbein som skal til jord blir lodda rett på oversida av plata.

R_5 sitt eine bein blir lodda fast til jordplanet saman med 4093'ens bein 7. Det andre beinet blir lodda saman med emitter og basis på transistorane i PTT-nøklingskretsen *oppå* printplata.

På undersida av plata skal det loddast to leidningar. Den eine går frå EPROM’ens pinne 8 til teljarens pinne 9. Den andre går frå EPROM’ens pinne 13 til 4093'ens pinne 6.

Slik ser printplata ut: *(Holavstand: 53,3 74,3 mm)*



MIKROKONTROLLER

Ein gang i framtida kan det bli aktuelt å prøve å programmere ein mikrokontroller for å realisere denne kretsen. Miniatyriseringa vil då vere komen svært langt i høve til den opprinnelege kretsen med 12 logiske kretsar for å kunne morse ut eit kallesignal.

Prosjektet blei avslutta i desember 1997.

Erik Grindheim