SUPERMAX HARDWARE

Technical Manual
for the
CPU MODULE 100
October 1984

Appendix 2. Beskrivelse af PAL´er.

PAL C10. Tilstandsdiagram.
PAL C10. Programmeringstekst.
PAL C20. Tilstandsdiagram.
PAL C20. Programmeringstekst.
Lagerstyring. Tilstandsdiagram.
PAL C30. Programmeringstekst.
PAL C41. Programmeringstekst.
PAL C50. Programmeringstekst.
PAL C60. Tilstandsdiagram.
PAL C60. Programmeringstekst.
PAL C70. Programmeringstekst.
PAL C80. Tilstandsdiagram.
PAL C80. Programmeringstekst.
PAL C90. Tilstandsdiagram.
PAL C90. Programmeringstekst.
PAL C100. Tilstandsdiagram.
PAL C100. Programmeringstekst.
PAL C110. Tilstandsdiagram.
PAL C110. Programmeringstekst.
PAL C120. Tilstandsdiagram.
PAL C120. Programmeringstekst.
PAL C130. Tilstandsdiagram.
PAL C130. Programmeringstekst.
PAL C140. Programmeringstekst.
PAL C151. Programmeringstekst.
PAL C173. Programmeringstekst.

1. Indledning.

Formålet med denne manual er at give en hardware beskrivelse af CPU kortet 100. Den detaljerede beskrivelse af kortet findes i diagrammerne til kortet samt i programmeringsteksterne til PAL´erne på kortet. CPU100 er endvidere beskrevet i Programmers Manual for CPU MODULE 100. I den følgende beskrivelse forudsættes denne manual bekendt.

CPU100 er opbygget omkring en 68000. Kortet indeholder denne mikroprocessor samt interface til SUPERMAX IO BUS og kontrol logik til de dynamiske lagerkort.

I denne tekst refereres ofte til forskellige kredsløb på CPU100. Der anvendes følgende betegnelser:

U3-11 pp7 betyder: Pin nummer 11 på pakken med navn U3. Signalet kan findes i diagrammet side 7. U3´s placering på CPU100 fremgår af komponentlisten til CPU100.

PAL C30 F7 pp7 betyder: Pakken med navn F7 er en PAL programmeret med indholdet C30. Den er vist i diagrammet side 7. F7´s placering på CPU100 fremgår af komponentlisten til CPU100.

2. 68000 Hardware beskrivelse.

68000 er en 16 bit mikroprocessor. Den har en 16 bit data bus og en 23 bit adresse bus. Adressen på adressebussen er en ordadresse. 68000 drives af en klok og et antal klokcycles udgør en CPU cycle. 68000 kan udføre følgende cycles: READ, WRITE, READ-MODIFY-WRITE, samt INTERRUPT ACKNOWLEDGE.

2.1. Styresignaler.

AS: ADDRESS STROBE, aktiv lav. Signalet angiver, at der er en adresse på adresse bussen, og at CPU´en er i færd med at udføre en cycle.

R/W: READ/WRITE. Signalet er højt, når 68000 udfører en læse cycle, og lavt, når 68000 udfører en skrive cycle.

UDS,LDS: UPPER DATA STROBE, LOWER DATA STROBE, aktiv lave signaler. Disse signaler styrer sammen med R/W data bussen.

I en skrive cycle angiver UDS og LDS, at 68000 har sat data på data bussen. I en læse cycle angiver UDS og LDS hvilke(n) byte(s) 68000 ønsker at læse i det ord, der er adresseret af adressen på adresse bussen.

Læsning/skrivning af ord: både UDS og LDS er aktiveret.

Læsning/skrivning af byte:

UDS aktiveret, LDS deaktiveret: byte adresse er lige. Data på DATA(15:8)

UDS deaktiveret, LDS aktiveret: byte adress er ulige. Data på DATA(7:0)

Bemærk at 68000 anvender memory mapped IO. Der findes altså ingen specielle instruktioner eller styresignaler til input output. Om en cycle er en lager access eller en IO access skal styres af ekstern adresse dekodning.

2.2. Afslutning af CPU cycle.

En CPU cycle afsluttes normalt, når den adresserede enhed, IO eller lager, aktiverer et af følgende signaler:

DTACK: Data Transfer Acknowledge, aktivt lavt signal. Aktiveres af den adresserede enhed, når denne har placeret data på databussen i en læse cycle eller modtaget data i en skrive cycle.

BERR: Bus Error, aktivt lavt signal. Aktiveres af den adresserede enhed, hvis denne ikke er i stand til at udføre den pågældende cycle, f.eks. hvis der ved læsning optræder en paritetsfejl.

Hvis ingen enhed aktiverer DTACK eller BERR afsluttes cyclen aldrig, 68000 går i stå. Normalt er maskiner basseret på 68000 derfor forsynet med et timeout kredsløb, som aktiverer BERR, hvis en cycle varer mere end en vis tid. Dette er også tilfældet for CPU100.

Der eksisterer en undtagelse fra reglen om afslutning af cycles: hvis 68000 detekterer en ADDRESS ERROR, d.v.s access til et ord på en ulige byte adresse, afslutter 68000 cyclen uden at vente på DTACK eller BERR.

I enhver cycle placerer 68000 en funktionskode på FCT(2:0). Af de 8 mulige funktionskoder er 3 ikke defineret, mens de sidste 5 skelner mellem user data, user program, supervisor data, supervisor program og interrupt acknowledge.

Read cycle timing er vist fig. 1.

Write cycle timing er vist fig. 2.

2.3. Read modify write.

Denne cycle består af en byte læsning og en skrivning af samme byte. Cyclen adskiller sig fra en normal læsning efterfulgt af en normal skrivning, ved at AS er konstant aktiveret under hele READ-MODIFY-WRITE cyclen. Den eneste instruktion, der anvender READ-MODIFY-WRITE, er Test and Set, TAS. Operativsystemet anvender denne instruktion til forskellige former for reservation.

2.4. Interrupt.

68000 anvender 3 aktivt lave input, IPL(2:0), til interrupt.

Hvis alle signaler er deaktiverede, høje, er der intet interrupt request. Hvis et eller flere af signalerne er aktiverede, angiver koden på IPL(2:0) niveauet på interrupt requestet. Der er således 7 interrupt niveauer, af hvilke 7 er det højst prioriterede. 68000 honorerer interrupt request ved at udføre

en READ CYCLE med funktionskoden, FCT(2:0)= INTERRUPT ACKNOWLEDGE. CPU100 vil aktivere input signalet, VPA, til 68000, når kortet detekterer en INTERRUPT ACKNOWLEDGE cycle. 68000 vil afslutte cyclen og anvende "auto vectoring", d.v.s 68000 vil selv finde interrupt vektoren, som så kun kan afhænge af interrupt niveauet.

3. Cycles på CPU100.

CPU100 udfører cycles. Initiativet til disse cycles kan komme fra to kilder: 68000 og IO bussen.

68000 kan udføre følgnede cycles:

1. Lager access.
2. IO access.
3. Access til IO bus. CPU100 er den aktive enhed.
4. Interrupt acknowledge cycle.

IO bussen kan udføre følgende cycles:

1. Lager access. CPU100 er den passive enhed.
2. IO access til test formål. CPU100 er den passive enhed.

Da både IO bussen og 68000 kan tage initativet til en cycle, må der prioriteres mellem de to initiativtagere. Dele af CPU 100 kender ikke forskel på cycles startet af 68000 og cycles startet af bussen. Dette gælder f.eks. lagerstyringen. CPU100 udfører en cycle ad gangen. Prioritering af den følgende cycle finder sted, når den nuværende er afsluttet. Der findes en situation, hvor dette ikke er tilfældet: CPU100 har startet en cycle for 68000. Denne cycle er en access til IO bussen. IO bussen er optaget af en cycle, der har CPU100 som den passive enhed. Hermed er situationen låst: IO bussen er optaget og venter på CPU100. CPU100 er optaget og venter på IO bussen. Denne situation detekteres, og CPU100 opgiver accessen til IO bussen. Der foretages en ny prioritering, så cyclen fra IO bussen bliver udført, herefter genoptages cyclen fra 68000 til IO bussen.

Til alle cycles, undtaget interrupt acknowledge, hører en adresse. Adressen leveres af initiativtageren til cyclen. Denne adresse betegnes den logiske adresse. Den logiske adresse oversættes til en fysisk adresse, der anvendes til adressering af lageret. Oversættelsen foretages af MMU´en, der er beskrevet i afsnit 8. Ved adressering af IO enheder på CPU100 anvendes den logiske adresse altid direkte uden oversættelse i MMU.

Cycles klassificeres og overvåges ved dekodning af cycle status beskrevet i afsnit 9. På grund af beskyttelssesmekanismerne på CPU100 er enhver cycle ikke tilladt i enhver tilstand. Hvis beskyttelsesmekanismerne overtrædes genererer CPU100 en bus error til initiativtageren til cyclen.

Normalt afsluttes en cycle på CPU100, ved at CPU100 genererer DTACK eller BERR til 68000 eller IO bussen, men CPU100 er konstrueret, så den tåler korte eller afbrudte cycles. Hvis initiativtageren til en cycle afbryder cyclen og evnt. starter en ny cycle, før CPU100 har givet DTACK eller BERR, fungerer CPU100 rigtigt. Denne facilitet har kompliceret konstruktionen betydeligt, men har været nødvendig blandt andet p.g.a. 68000 opførsel ved ADDRESS ERROR.

Generering af DTACK/BUS ERROR er beskrevet i afsnit 10.

4. Prioritering af cycles.

Cycles fra 68000 og IO bussen er prioriteret ens, sådan at efter en cycle fra IO bussen har 68000 den højeste prioritet, og efter en cycle fra 68000 har IO bussen den højeste prioritet.

Kontrollogikken på CPU100 er så vidt muligt udført med synkrone sekvensmaskiner implementeret ved hjælp af PAL´er. Dette gælder også prioriteringen af cycles.

Genkendelse af unit nummer foretages af PAL C17X B8 pp3. Når unit nummeret på bussen matcher nummeret kodet i PAL´en, og

ASB er aktiveret, er B8-19, /UNIT, aktiveret, d.v.s lav. /UNIT klokkes ind i en flip-flop , S3 pp6, på busklokken, CPB. Udgangen fra denne flip-flop, S3-05, synkroniseres til CPL i S0 pp6. Udgangen ,S0-06, er address strobe fra bussen synkroniseret til CPL. Signalet kaldes AS2 og angiver, når det er højt, at der er en access til CPU100 fra IO bussen.

På tilsvarende måde synkroniseres address strobe fra 68000 til CPL i S1 pp6. Den synkroniserede address strobe fra 68000 kaldes AS1.

AS1 og AS2 er de fundamentale signaler i prioriteringen af cycles på CPU100.

Når CPU100 skal udføre en access til IO bussen, sættes signalet REQ pp3 på forskellig måder, som beskrives i afsnit 14. Signalet findes i to versioner: synkroniseret til bus klokken, CPB, S3-09, og synkroniseret til lagerklokken, CPL, S0-09.

REQ, S0-09, indgår også i prioriteringen af cycles på CPU100, idet den tidligere omtalte konflikt situation indtræffer, når AS2*REQ=1.

Prioriteringen af cycles sker af PAL C10 position F1 pp6 og PAL C20 position C1 pp3.

4.1. PAL C10.

PAL C10´s vigtigste funktioner er:

Prioritering af cycles : Når F1-16, /CON, er lav udføres en cycle fra IO bussen.

Start af lager styring: Når F1-17, /CS, er lav, skal lagerstyringen startes.

Detektion af read modify write: Når F1-15, /MW, og /CS begge er lave, skal lagerstyringen starte write delen af en read modify write.

Detektion af korte cycles:

Hvis 68000 udfører en kort cycle, skifter F1-19, IC, fra høj til lav og forbliver lav, indtil CPU100 har afsluttet cyclen.

Hvis IO bussen udfører en kort cycle, skifter F1-18, IB, fra høj til lav og forbliver lav, indtil CPU100 har afsluttet cyclen.

4.2. PAL C20.

PAL C20´s vigtigste funktioner:

Prioritering af cycles: Når C1-19, /CON, er lav, udføres en cycle fra IO bussen.

Generering af chip select til MMU.
C1-18, CSMMUA, chip selest til MMU i cycle fra 68000.
C1-17, CSMMUB, chip select til MMU i cycle fra IO bus.

Generering af fundamentale timing signaler.

C1-16, /T1, lav i første klokperiode efter AS1 eller AS2.
C1-15, /T2, lav i anden klokperiode efter AS1 eller AS2.
C1-14, /T3, intern variabel i PAL C20. Anvendes ikke på CPU100
C1-13, /T4, lav i fjerde klokperiode efter AS1 eller AS2.
/T4 holdes lav til AS1 eller AS2 forsvinder. Se også fig. 6.

Detektion af konflikt situationer:

C1-12, /NA0, lav i en klokperiode, hvis AS2*REQ=1.

/NA0 er endvidere lav i en klokperiode, hvis 68000 eller IO bus udfører en kort cycle. NA0 er er input til PAL C10.

En detaljeret beskribelse af PAL C10 og C20 findes i appendix 2.

5. Styresignaler til lagerstyring.

Lagerstyringen kan udføre cycles startet af 68000 eller IO bussen. Der skal derfor ske en multipleksning mellem de to sæt af styresignaler, pp6. Flip flop´ene S1-09, S2-05, S2-09 sættes lave af den negativt gående flanke af styresignalerne fra 68000, når address strobe fra 68000, ASC, er aktiv, og når IC er høj. IC høj betyder at lagerstyringen er parat til at modtage styresignaler fra 68000. Hvis 68000 udfører en kort cycle, vil først ASC blive deaktiveret og dernæst IC, sådan at styresignalerne ikke ændres, selv om 68000 når at starte en ny cycle, før CPU100 har afsluttet den gamle. Styresignalerne kan kun fjernes, sættes høje, af signalet /COMP1, som angiver at CPU100 har afsluttet cyclen.

Styresignalerne fra IO bussen latches på helt tilsvarende måde i S4-05, S4-09, S5-05 og S5-09.

De to sæt styresignaler multiplekses af multiplekseren F1, som styres af signalet /CON. De multipleksede styresignaler synkroniseres i F3. De multipleksede styresignaler anvendes af lagerstyringen og af styringen af IO enheder på CPU100.

Ud fra de synkroniserede styresignaler genereres en 2 bit funktionskode, F1 og F0, med følgende betydning:

	F1	F0
READ	0	0
WRITE BYTE	0	1
WRITE WORD	1	1
MODIFY WRITE	1	0

Funktionskoden angiver de fire fundamentale cycles, lagerstyringen kan udføre. De multipleksede og synkroniserede styresignaler indeholder detaljer til disse cycles som f.eks. byteadressen. Se fig. 3.

6. Lagerstyring.

Det er lagerstyringens opgave at styre det dynamiske lager, der er tilsluttet CPU100. Det dynamiske lager anvender en

fejlkorrigerende kode. Til et 16 bit ord er der tilføjet 6 ECC bit, som muliggør korrektion af enkelt bit fejl og detektion af alle dobbelt bit fejl. Da den fejlkorrigerende kode genereres for et helt ord bliver byte skrivning kompliceret. Ordet, som indeholder den byte, der skal ændres, må først læses fra lageret, den nye byte maskes ind i ordet, ECC bit genereres for det nye ord, hvorefter dette kan skrives i lageret. Lagerstyringen styrer alt vedrørende fejlkorrektion og laver desuden refresh af det dynamiske lager. Se fig. 7.

Forbindelsen mellem CPU100 og det dynamiske lager består af en lokal bus, MB(43:0), som løber i SUPERMAX´ens motherboard. Denne bus anvendes multiplekset til overførsel af både adresse og data til det dynamiske lager.

Den følgnede beskrivelse af lagerstyringen omfatter styringen af dataveje, registre mm, som er nødvendige p.g.a. fejlkorrektionen. Styringen af adressen til lageret er beskrevet i afsnit 7.

Input til lagerstyringen er CS, CYCLE START, som genereres af PAL C10 pp6., funktionskoden F1, F0, som er beskrevet ovenfor, samt signalet NA, No Access, U0-08 pp7. I starten af enhver cycle aktiveres CS. Hvis lageret ikke skal startes f.eks. fordi cyclen er en IO cycle, eller fordi lagerbeskyttelsen er overtrådt, aktiveres NA, hvorved lagercyclen standses. NA genereres først og fremmest ved dekodning af cycle status, se afsnit 9.

Lagerstyringen er implementeret som to sekvensmaskiner efter hinanden. CS er input til den første sekvensmaskine, som styrer busser, fejlkorrektion og den anden sekvensmaskine. Den første sekvensmaskine er implementeret med PAL C30, C40 og C51 i positionerne F7, F6 og F5 henholdsvis. Den anden sekvensmaskine er implementeret i PAL C60 position F8 pp6. Se fig. 3 og fig. 4.

PAL C60 genererer følgende signaler til lagerkortet:
RAS, ROWEN, CAS, WP, REF alle aktivt lave.

RAS, CAS og WP er direkte styresignaler til de dynamiske lagerkredse 4164.

ROWEN og REF styrer multipleksningen af adressen på lagerkortet til de dynamiske lagerkredse.

Når REF er aktiveret, lav, vælger lagerkortet refresh adressen, som kommer fra en tæller placeret på lagerkortet.

Når REF er deaktiveret, vælger lagerkortet row adressen, når ROWEN er deaktiveret, høj, og column adressen, når ROWEN er aktiveret, lav.

PAL C60 er altså i stand til at udføre de tre fundamentale lager cycles:

Læs ord.
Skriv ord.
Refresh.

Input til PAL C60 er

START, WRITE, som generes af den første sekvensmaskine, PAL C30.

NA, som tidligere er omtalt.

RQ, Refresh Request, som er høj, når tiden er inde til at udføre en refresh cycle. RQ genereres ved hjælp af to flip flop, G1-05 og G1-08 pp7 samt en klok, RCLK med en periodetid på 13 us.

PAL C60 styrer læsning og skrivning af ord i lageret efter ordre fra PAL C30. Da C60 desuden udfører refresh cycles, som er ukendte for C30 er det nødvendigt med en tilbagemelding fra C60 til C30. Denne tilbagemelding sker med signalet ACK, som aktiveres, sættes lav, på et vist tidspunkt i ord læsningen eller ord skrivningen. En detaljeret beskrivelse af PAL C60 findes i appendix 2.

Den første sekvensmaskine er implementeret med PAL C30, C40 og C51. C30 er hjertet i denne sekvensmaskine, idet den genererer tilstandsvariable Y3, Y2, Y1 og Y0, som er input til C40 og C51. C40 og C51 genererer kun udgangssignaler.

6.1. Dataveje i lagerstyringen.

Datavejene i lagerstyringen består af to næsten identiske dele på hver 16 bit. Se også fig. 5. Den ene del anvendes ved byte og ord accesser, mens den anden del anvendes sammen med den første ved 32 bit accesser. Den første del er vist i diagrammet pp8, den anden i diagrammet pp9. I det følgende beskrives den del, der anvendes ved byte og ord accesser pp8.

Interface mod 68000 består af to 16 bit registre, som kaldes C. Det ene register bestående af G2 og G3 anvendes ved skrivning i lageret, mens det andet bestående af G4 og G5 anvendes ved læsning i lageret. Registeret C kan betragtes som et tovejs register, der sidder mellem bussen LD(21:0) og databussen på CPU100, DATA(15:0). På LD(21:0) sidder også registeret R(15:0), G7 og H2, samt kredsen til fejlkorrektion, G8. R(15:0) anvendes ved byte skrivninger, og når der korrigeres fejl. Forbindelsen mellem LD(21:0) og lagerbussen MB(21:0) består af tovejs drivere, G9, H0, H1. På LD(21:0) sidder også et register, G6, hvori syndrom bit gemmes, når der korrigeres fejl. Syndrom bittene udpeger den bit, der fejlede.

Kredsen, G8, der anvendes til fejlkorrektion er en 74LS630.

Den udfører fire funktioner styret af input signalerne S1 og S0:

S1 S0

L L :INPUT DATA, OUTPUT CHECKBIT. SKRIVNING
L H :INPUT DATA, INPUT CHECBIT. LÆSNING
H H :LATCH DATA OG CHECKBIT. FLAG ERRORS. KONTROL AF DATA
H L :OUTPUT KORRIGERET DATA OG SYNDROM BIT. FEJLKORREKTION

I det følgende gennemgås de forskellige lagercycles. Der henvises til tilstandsdiagrammet for lagerstyringen i appendix 2.

6.2. Read cycle.

PAL C30 venter i T0, til CS aktiv. Når CS aktiv og F1,F0 = READ, hoppes til T1. I T1 startes en læsning i lageret, START aktiveres, og WRITE deaktiveres til PAL C60. C30 venter i T1, til C60 aktiverer ACK.

I T4 og T5 er datavejene sat op til at modtage data fra lageret. Fejlkorrektionskredsen er i tilstanden input data og checkbit. Ved overgangen mellem T5 og T6 latches data i registrene R og C samt i fejlkorrektionskredsen, som i tilstand T6 og T7 er i tilstand flag errors. Hvis der ikke er fejl, genererer lagerstyringen DTACK i T7 og hopper til T12, hvor den venter, til CS deaktiveres. Hvis der i T7 konstateres fejl, går lagerstyringen til T8. I T8 og T9 korrigeres data. Fejlkorrektionskredsen er i tilstand output korrigeret data og syndrom bit. På overgangen mellem T9 og T10 latches data i R og C og syndrombittene i G6. I T11 startes skrivningen af de korrigerede data i lageret. Data kommer fra R. Fejlkorrektionskredsen er i tilstand input data output check bit. Lagerstyringen forbliver i T11 til PAL C60 aktiverer ACK. Dernæst går lagerstyringen til T12, hvor der sendes DTACK og ventes, til CS deaktiveres. Efter en READ cycle er data fra lageret latchet i C og i R. Data i R anvendes ved en eventuel efterfølgende modify write cycle. Fig. 6 viser et timing diagram for en read cycle startet af 68000.

6.3. Modify write cycle.

Denne lagercycle er write delen af en read modify write cycle. Lagerstyringen går til T11, hvor START og WRITE aktiveres, så PAL C60 starter en skrive cycle. I T11 sættes data vejene op, så den adresserede byte i ordet kommer fra C, mens den ikke adresserede byte kommer fra R, hvor ordet blev indlæst i den foregående read cycle. Fejlkorrektionskredsen er i tilstand input data output checkbit. Lagerstyringen forbliver i T11,

indtil PAL C60 aktiverer ACK, hvorefter den går til T12, hvor der sendes DTACK og ventes, til CS deaktiveres.

6.4. Write word cycle.

Lagerstyringen går til T1, når CS er aktiveret og F1, F0 er WRITE WORD eller WRITE BYTE. I T1 aktiveres START, og WRITE deaktiveres. Der startes altså en læse cycle i lageret. Data vejene er sat op, så data fra C driver LD(15:0), og fejlkorrektionskredsen er i tilstand input data output checkbit. Hvis funktionskoden er write byte, forbliver lagerstyringen i denne tilstand, til ACK aktiveres. Hvis funktionskoden er write word, går lagerstyringen til T2, hvor WRITE aktiveres, og write data til lageret enables. I T11 forbliver datavejene sat op, så data fra C sendes til lageret, mens fejlkorrektionskredsen genererer checkbit. Lagerstyringen forbliver i denne tilstand, til PAL C60 aktiverer ACK, hvorefter den går til T12, hvor der sendes DTACK og ventes, til CS deaktiveres.

6.5. Write byte cycle.

Lagerstyringen går til T1, når CS er aktiveret, og funktions koden er write byte. I T1 aktiveres START, og WRITE deaktiveres, så der startes en læsning i lageret. I T1 er datavejene sat op til skrivning af data fra C, mens der altså er startet en lager læsning. Lagerstyringen forbliver i denne tilstand, indtil enten funktionskoden ændres til WRITE WORD, eller PAL C60 sender ACK. Hvis PAL C60 sender ACK, og funktionskoden stadig er WRITE BYTE, går lagerstyrnigen til T3, hvor datavejene ændres til læsning. Herefter gennemføres en læsning i tilstandene T4, T5, T6, T7, samt en eventuel fejlkorrektion i tilstandene T8, T9, T10 som beskrevet under READ CYCLE. Der sendes dog ikke DTACK i T7. Når lagerstyringen går til T11, er det udlæste ord latchet i R. I T11 aktiverer lagerstyringen START og WRITE, så PAL C60 starter en skrivning. Den adresserede byte i C driver den ene halvdel af LD(15:0), OECA, mens den ikke adresserede byte i R driver den anden halvdel, OERB. Fejlkorrektionskredsen er i tilstand input data output checkbit. Lagerstyringen bliver i T11, indtil C60 aktiverer ACK,

hvorefter lagerstyringen går til T12, hvor DTACK aktiveres og der ventes, til CS deaktiveres.

6.6. Styresignaler fra lagerstyring.

Styresignalerne fra PAL C30, C40 og C51 gates sammen med DWMPX, som er aktiveret, når det er en 32 bit access, og styrer de to 16 bit dele, som datavejene i lagerstyringen er opdelt i, pp7.

Lokal bussen til lageret MB(43:0) anvendes både til adresse og data til lagerkortene. Multipleksningen sker ved hjælp af signalet ROWEN. Når ROWEN er deaktiveret, anvendes bussen til adresse; når ROWEN er aktiveret, anvendes MB(43:0) til data.

Output enable af læs data fra lagerstyringen til databussen på CPU100 styres af lagerstyringen og initiativtageren til read cyclen i skøn forening. Styringen sker ved hjælp af flip flop S8 pp8. Når lagerstyringen skriver i C, sættes S8-09, hvis cyclen er fra 68000, og hvis cyclen ikke er afbrudt, IC aktiv. Herved enables output fra C til databussen, hvis address strobe fra 68000, ASC, er aktiv, og cyclen er en læse cycle, RC aktiv.

7. Adresse til lager.

I det foregående er styringen af datavejene til lageret beskrevet. I det følgende beskrives adresseringen af lageret. Ved adresseringen af lageret anvendes en Memory Management Unit, MMU, som oversætter fra logiske til fysiske adresser. Den logiske adressebus på CPU100 betegnes ADR(23:1). ADR(23:1) drives af en af to kilder: 68000 eller IO bussen styret af signaler CON. Se pp1, pp2 og pp3. Da CPU100 kan deltage i IO bus cycles som den aktive part, d.v.s drive adresse delen af IO bussen, modtages den logiske adresse fra IO bussen ved hjælp af to vejs drivere, A2, A3, A4, A5 pp1. Generering af enable signaler er vist på pp3. Den logiske adresse oversættes i MMU'en eller anvendes direkte til adresseringen af lageret. Valget mellem logisk og fysisk adresse sker ved hjælp af sig-

nalet NAT pp12. NAT genereres ved dekodning af cycle status beskrevet i afsnit 9. NAT gates sammen med ROWEN på pp7 og danner OEPHA, output enable physical address, og OELADR, output enable logical address. Disse signaler styrer enabling af adresser til lagerkortet, pp5. Logiske adresser latches i E2, E6, E7, og fysiske adresser latches i E2, E3 og E4. Det er nødvendigt at latche adresser af hensyn til korte eller afbrudte cycles. Da bussen til lageret, MB(43:0), er multiplekset, latches adressen igen på lagerkortene. Se også fig. 5.

8. Memory Management Unit. MMU.

MMU´en oversætter fra logiske til fysiske adresser ved at addere et offset til den logiske adresse. En detaljeret beskrivelse af MMU´ens funktion kan findes i "Programmers Manual". Her skal kun gives en kort beskrivelse til forståelse af hardware. CPU100 arbejder med 64 logiske adresse rum. Valget af adresse rum styres af et Address Space Number. Se fig. 8. Hvert logisk adresserum er på 16 MB og inddelt i 16 logiske segmenter af 1 MB. Der eksisterer altså totalt 1024 logiske segmenter. Til hver logisk segment hører en 32 bit Segment Descriptor. En segment descriptor indeholder en access kode, en segment længde samt et offset. Segment descriptors opbevares i MMU´ens lager, som er et hurtigt statisk lager på 1K32, pakke D0, D1, D2, D3, D4, D5, D6, D7 pp3. MMU´ens lager er en ydre enhed for CPU100. Der der derfor tre kilder til MMU´ens adresse, MAR(9:0):

1. IO bussen. Pakke A6, A7 pp1
2. 68000. Pakke A7 pp2, C2 pp4.
3. ADR(23:1). Pakke C4, C5 pp4.

Den første kilde anvendes, når IO bussen accesser CPU100.

Når 68000 accesser lageret, kommer Address Space Nummeret, ASN, fra et register kaldet Address Space Register, ASR, C2 pp4.

Den tredje kilde anvendes, når segment descriptors skrives

eller læses i MMU´ens lager. I denne situation er MMU´en en IO enhed på CPU100.

Data ind/udgangene fra MMU´ens lager, MD(31:0), er sluttet til data bussen på CPU100, DATA(15:0), ved hjælp af tovejs drivere C6, C7, C8 og C9 pp4. Disse drivere anvendes ved læsning og skrivning af segment descrioptors i MMU´ens lager.

Offset feltet i segment descriptor, MD(15:0), er ført til den ene indgang på en 16 bit adder, mens ADR(19:8) er ført til den anden indgang på adderen. Adderen består af pakkerne D8, D9, E0, E1 pp5. Udgangen fra adderen går til latch E3 og E4, som driver lagerbussen MB(43:0). ADR(7:1) deltager ikke i additionen, men er ført direkte til en adresselatch, E2 pp5. Bemærk, at Double Word Data Strobe, DWMPX, overføres til lagerkortet via samme latch.

Length feltet i segment descriptor, MD(27:16), er ført til den ene indgang til en 12 bit comparator, mens ADR(19:8) er ført til den anden indgang. Comparatoren består af pakkerne E8, E9 og F0. Udgangen fra comparatoren, Too Long, TL indgår i dekodningen af cycle status sammen med access koden ACC(3:0), hvilket er MD(31:28). Se fig. 9.

Udgangen fra MMU´en er yderligere ført til A8 og A9 pp1, da dele af segment descriptoren, MD(13:0), anvendes som adresse til IO bussen ved en lang access til denne. Se afsnit 14.2.

Når MMU´en anvendes til oversættelse af adresser genereres chip select til lagerkredsene af PAL C20 pp3. CSMMUA, C1-18, er chip select, når 68000 udfører en cycle, og CSMMUB, C1-17, er chip select, når IO bussen udfører en cycle.

Disse chip select gates sammen med chip select fra IO styringen af MMU´en i Y6 pp3.

9. Dekodning af cycle status.

CPU100 udfører cycles af forskellige typer. De vigtigste cycles er:

1. Access til lager fra 68000.
2. Access til lager fra IO bus.
3. Access til IO units på CPU100 fra 68000.
4. Access til IO units på CPU100 fra IO bus.
5. Long access til IO bus fra 68000.
6. Short access til IO bus fra 68000.
7. Interrupt acknowledge cycle fra 68000.

Formålet med dekodning af cycles er at skelne mellem de forskellige cycles og kontrollere, at den påbegyndte cycle er lovlig. Af hensyn til hastigheden antages altid at cyclen er en lager access. Lageret startes altså, og starten sker parallelt med dekodning af cyclen. Hvis dekodningen viser, at cyclen ikke er en lager access, standses lagercyclen med signalet NA pp7.

I dekodningen af cycles, vist pp7, indgår bit fra kommando registeret, som enabler/disabler:

oversættelse af adresse,
test på access code,
user mode tilladt i SEG1,
adgang til IO units på CPU100 fra IO bus.

Endvidere indgår ADR(23:20), som dekodes til 16 logiske segmenter, hvoraf SEG0 og SEG1 er specielle. SEG0 anvendes normalt kun af operativsystemet. SEG0 kan normalt kun accesses i supervisor mode og oversættelse af adresser anvendes aldrig.

SEG1 anvendes til memory mapped IO på CPU100 og kan normalt kun accesses i Supervisor mode. Styret af kommandoregisteret kan IO units dog også accesses i User mode og fra IO bussen.

Signalet CON sammen med FCT(2:0) fra 68000 giver 4 forskellige typer af cycles:

1. Hvis CON er aktiv er cyclen fra IO bussen og FCT(2:0) er uden betydning.

2. Hvis CON er inaktiv er cyclen fra 68000 og der skelnes mellem 3 cycles ved hjælp af FCT(2:0):

2A. Cycle i Supervisor mode.
2B. Cycle i User mode
2C. Interrupt acknowledge cycle.

I dekodningen indgår altså tre segment typer:

1. Segment 0
2. Segment 1
3. Segment 2 til Segment 15

samt de fire typer af cycles nævnt ovenfor.

Denne dekodning sker ved hjælp af PAL C140 position M9 vist i diagrammet pp7.

Output fra PAL´en har følgende betydning:

Pin 19: Hvis høj, skal adressen ikke oversættes.
Pin 18: Hvis lav, skal lageret ikke accesses, lager cyclen skal standses.
Pin 17: Hvis lav, er cyclen en ulovlig cycle i user mode.
Pin 16: Hvis høj, er cyclen en interrupt acknowledge cycle.
Pin 15: S, hvis høj, er cyclen i supervisor mode
Pin 12: U, hvis høj, er cyclen i user mode.
Pin 13: T, hvis høj, skal access kode testes.

Access koden testes i PAL C150 position N0.

De vigtigste input til denne PAL er S, U og T fra PAL C140, Access koden fra MMU´en, skrive signalet fra multiplekseren pp6, samt output fra komperatoren, TL.

Output fra PAL´en har følgende betydning:

Pin 19: Hvis lav, ingen lager access, da beskyttelse er overtrådt.

Pin 18: Hvis høj, illegal user.
Pin 17: Hvis høj, illegal bus
Pin 16: Hvis høj, illegal write i en normal skrive cycle.
Pin 15: Hvis høj, segment too long.
Pin 12: Hvis høj, cyclen er en lang access til IO bussen.
PAL C140 og C150 er beskrivet i appendix 2.

Output fra PAL C140 og C150 klokkes ind i et register på bagkanten af T2, hvor informationen bevares resten af cyclen, indtil registeret resettes af CLCYST. Registeret består af N1, N2 og N3.

En read modify write cycle opfattes som en cycle på CPU100, men bestående af 2 lagercycles. Dette betyder, at MMU'en kun udlæses en gang i starten af cyclen svarende til read cyclen. ACC(2), som angiver, at skrivning er tilladt/forbudt, gemmes derfor i N3 til en eventuel skrive cycle. Testen, om skrivning er tilladt i en read modify write cycle, sker i Y0 pp12.

Indholdet af registeret bestående af N1, N2 og N3 anvendes til:

1. At standse en startet lager cycle ved hjælp af signalet NA pp6.

2. At generere bus error, hvis cyclen overtræder beskyttelsesmekanismerne på CPU100 pp14.

3. At gemme information i status registeret om årsagen til en eventuel bus error.

4. At generere VPA til 68000 i en interrupt acknowledge cycle.

5. At drive lysdioderne i front pannel display user/supervisor mode, pp16.

10. Generering af DTACK og BUS ERROR.

En cycle afsluttes normalt, når der genereres DTACK eller BUS

ERROR. Forskellige enheder på CPU100 generer selv DTACK, når cyclen er udført. Alle disse forskellige kilder til DTACK samles i en stor NAND gate, N9 pp14. Når udgangen fra N9 skifter fra lav til høj indlæses værdien af IC og IB i henholdsvis 02 og 03 pp14. Hvis IC er høj, sendes DTACK til 68000. Hvis IB er høj, sendes DTACK til IO bussen. Hvis IC og IB begge er lave er cyclen "kort", og der sendes ingen DTACK.

DTACK resettes af data strobe 02-01 eller 03-01.

DTACK fra lagerstyringen er DTX og DT2. DT2 genereres, når der ikke er fejl i lageret, Y9-11 pp14. DTX genereres, hvis fejlkorrektionen har korrigeret en enkelt fejl, Y8-06.

Hvis der er dobbelt fejl i lageret, løber lagerstyringen gennem de samme tilstande som ved enkelt fejl, men i stedet for DTX generes en bus error, BER1, Y8-11.

Bus error detekteres ved dekodning af cycle status. De forskellige kilder til bus error samles i en stor NAND gate 00 pp. 14. Udgangen fra 00 anvendes til generering af bus error på samme måde som udgangen af N9 anvendes til generering af DTACK.

Ved access til IO bussen, når CPU100 er den aktive part i en buscycle, kommer DTACK eller BUS ERROR fra IO bussen.

DTACK fra IO bussen genererer DT9, Y1-11, når ASBO er aktiv.

Bus error fra IO bussen genererer ERB, Y1-08, når ASBO er aktiv.

DTACK/BUS ERROR er afslutningssignalet til den, der har startet en cycle på CPU100, d.v.s til 68000 eller IO bussen. Afslutningssignalet til styringen på CPU100, COMP1 og COMP2, genereres særskilt ud fra DTACK/BUS ERROR ved hjælp af et to bit skifteregister, 01 pp14. CPU100 vil først starte en ny cycle, når initiativtageren til cyclen har afsluttet sin cycle, address strobe inaktiv, og når CPU100 har afsluttet cyclen, COMP1/COMP2 aktiv.

11. Styring af ydre enheder.

68000 anvender memory mapped IO. Ydre enheder på CPU100 er placeret i logisk SEG1 d.v.s fra adresse 1 Mb til 2 Mb. Adresse dekodning samt styring af ydre enheder er kombineret i en række PAL´er pp11.

PAL C70 position M0 dekoder funktionskoden, en del af den logiske adresse samt styrebit fra kommandoregisteret og genererer et aktivt lavt signal I1. Det andet output signal fra PAL´en, /BS, angiver at der er en kort access til IO bussen. Se afsnit 14.1.

M1 og M2 detekterer ADR(14:6) = 0 og generer et aktive lavt signal I2. PAL C80 position M3 dekoder andre adresse bit og gater resultatet af dekodningen sammen med I1 og I2. Output fra PAL C80 er 6 aktivt lave enable signaler, /E0, /E1,, /E5.

Disse enablesignaler, timingsignalet T4, styresignaler fra multiplekseren pp6 samt ADR(2:1) er input til følgende PAL´er:

PAL C90 position M4:
Læsning og skrivning i MMU lager.

PAL C100 position M5:
Skrivning i kommando register.
Læsning af status register.
Reset af status register efter læsning.
Læsning af Error Address Register
Læsning af Syndrom Register.

PAL C110 position M6:
Skrivning i maske register.
Læsning af maske register.
Læsning af interrupts.
Programmeret set/reset af eksterne interrupts.

PAL C120 position M7:
Skrivning af kommando og data i service USART.
Læsning af status og data i service USART.

PAL C130 position M8:
Skrivning i display register.
Skrivning i Address Space Register.
Læsning af Address Space Register.
Læsning af UNIT number.

Tilstands diagrammer for disse PAL´er er vist i appendix 2.

Bemærk at DTACK for IO ikke genereres centralt, men af de enkelte PAL´er. Fig. 10 indeholder en liste over ydre enheder på CPU100 samt deres adresser.

11.1. Kommando register.

Kommando registeret enabler og disabler forskellige funktioner på CPU100. Registeret er et simpelt register, hvori 68000 kan skrive. Efter reset er kommando registeret i en sådan tilstand, at CPU100 er så simpel som muligt. Hvis 68000 går i HALT, resettes kommando registeret. Se også fig. 11.

11.2. Status Register.

Status register er vist pp13. Se også fig. 12. Status registeret indeholder to typer af informationer:

1. Oplysninger om årsagen til en bus error.

2. Oplysninger om tilstande eller hændelser, der bør give anledning til et interrupt.

11.2.1. Oplysninger om buserror.

En buserror kan skyldes følgende:
Illegal user, detekteres ved dekodning af cycle status.
Illegal bus, detekteres ved dekodning af cycle status.
Illegal write, detekteres ved dekodning af cycle status.
Segment too long, detekteres ved dekodning af cycle status.

Time out, hvis der ikke genereres DTACK inden en vis tid, afslutte cyclen med en bus error forårsaget af time out. pp17.

No memory, hvis en lagercycle adresserer et ikke installeret lagerområde, afsluttes cyclen med en buserror. Når et lagerkort adresseres, aktiverer det signalet NM, sætter det lavt. Hvis NM ikke er lav, når lagerstyringen aktiverer CAS, genereres denne bus error, pp7.

Buserror fra IO bussen. Hvis den passive enhed svarer med bus error i stedet for DTACK, genereres en bus error på CPU100. Detaljerede oplysninger om årsagen til denne bus error findes kun i den passive enhed.

Double fault in memory. Hvis fejlkorrektionen detekterer en dobbelt fejl, genereres en bus error. pp14.

11.2.2. Oplysninger om interrupt.

Single fault in memory. Hvis fejlkorrektionen korrigerer en enkelt fejl sættes en bit i status registeret. Denne bit sættes af lagerstyringen, samtidig med at syndrom bittene gemmes, WSB.

Timer interrupt. T1-09 sættes, når klokindgangen T1-11 skifter fra lav til høj. Klokken kan strappes til forskellige frekvenser pp17. Udgangen, T1-09, kan strappes til at give et internt interrupt, Z5.

Error er et signal i IO bussen, hvis tilstand kan læses i status registeret. Signalet kan strappes til at give interrupt.

Power int er et signal i IO bussen, hvis tilstand kan læses i status registeret. Signalet kan strappes til at give interrupt, Z5.

Signalerne TxRDY og RxRDY fra service porten kan læses i status registeret. De kan strappes til at give interrupt, Z5.

Status registeret kan læses af 68000. Dette sker ved hjælp af driverne N6 og N7. Når status registeret læses styret af PAL

C100 vil PAL´en resette status registeret, når læsningen har fundet sted.

Hvis en cycle fra IO bussen medfører en bus error, sendes bus error til den aktive enhed, de relevante bit i status registeret sættes, og 68000 lades uberørt. Dette forhold er årsagen til, at bit i status registeret, som indeholder oplysning om buserror, også kan strappes til at give interrupt.

11.3. Service Port.

Service porten består af en USART, 8251A, position P2 pp16. Service porten er forsynet V24 drivere, Z7, og receivere, Z8. V24 udgangene er modificeret, idet udgangene drives lavt af pull down modstande og ikke af V24 driverne. Herved opnås, at service porte fra flere enheder kan kobles parallelt.

11.4. Display register.

Display registeret er et 16 bit register, P4 og P5 pp16. Udgangene fra registeret driver åben collector drivere P6 og P7. Disse drivere driver front panel display´et.

12. Interrupt enhed.

Interrupt enheden er vist på pp10 i diagrammet. Interrupt enheden tillader 8 eksterne interrupts og 8 interne interrupt.

Da interrupt request til 68000 er kodet på tre signaler, IPL(2:0), og da koden 0, alle inaktive, betyder intet interrupt, er der kun 7 mulige interrupt niveauer. Af disse syv anvendes niveau 7, det højst prioriterede, til en trap switch, der kan monteres på CPU100.

12.1. Interne interrupts.

Interne interrupts kommer fra en strap sokkel, Z5 pp13, inverteres i L4 pp10 til aktiv høje signaler og gates sammen med interrupt masken i L0 og L1. Masken findes i register K4. Se fig. 13.

12.2. Eksterne interrupts.

Eksterne interrupts genereres, når IO bussen skriver i visse dele af lageret. Eksterne interrupts kan også sættes og resettes programmeret. De eksterne interrupts sættes i en adresserbar latch, L5. Skrivesignalet til latchen generes ved dekodning af adressen, ppl1. Multiplekseren L6 vælger en tre bit adresse til latchen og en bit data. Multiplekseren styres af CON. Når cyclen er en cycle fra IO bussen, vælges ADR(3:1). Når cyclen er fra CPU'en, vælges DATA(3:0). Udgangen fra latchen gates sammen med de tilsvarende adresse bit i L2, L3. Masken findes i registeret K5.

De maskede interne interrupts er forbundet sammen med de tilsvarende maskede eksterne interrupts, åben kollektor, og klokkes ind i register L7 på den inverterede CPU klok. Udgangen af L7 er forbundet til en prioritets enkoder L8, hvis udgange igen klokkes i L9 på den inverterede CPU klok. De to registre L7 og L9 sikrer, at 68000 ikke får falske interrupts.

Masken, K4 og K5, kan læses ved hjælp af drivere K6 og K7. De ikke maskede interrupts kan læses ved hjælp af drivere K8 og K9.

13.1. BOOT PROM.

Boot PROM'en består af to stk. 2764, O8 og O9 vist pp15.

Boot PROM'en accesses ved læsecycles fra 68000, når ADR(23:14)=0, og når PROM'en er enabled, COM(6)=0.

Boot PROM'en kan også accesses fra IO bussen, når COM(7)=1.

Dekodning af access til boot PROM sker i O5. Når der er access til Boot PROM, genereres NA4, som standser lagerstyringen, O6. DT8, DTACK, genereres ved hjælp af multivibratorer, O7. P0 og P1 er drivere, der driver data bussen.

13.2. Klokgeneratorer.

CPU100 kan generere klok til forskellige dele:
68000
IO bussen
Intern logik på CPU100.

CPU100 er forsynet med tre klokgeneratorer på 12MHz, 16 MHz og 20 Mhz. Disse frekvenser neddeles, så man har 6, 8, 10, 12, 16 og 20 MHz til rådighed i klokstrappen, hvor klokken til de tre ovennævnte enheder strappes.

Normal strapning er:

CPU klok, CPCPU: 10 MHz.
IO bus klok, CPB: 8 MHz.
Intern logik, CPL: 16 MHz.

16 MHz klokken neddeles til timer interrupt og baud rate klok for service porten, ved hjælp af en række tællere. Timer interrupt frekvens og baud rate strappes i Z4. Se fig. 14 og fig. 15.

13.3. RESET.

CPU100 driver RESET i IO bussen. Dette signal er et åben kollektor signal, V9-12 ppl7. RESET generes ud fra signalet KEY i IO bussen. Når KEY er lav holdes tællerne R3 og R4 samt flip flop'en T5 resat. Når KEY går høj, frigøres tællerne R3 og R4, og T5 sættes, når R4-06 skifter fra lav til høj efter 1280 ms. Hvis KEY ikke anvendes, giver RC leddet bestående af RE7 og CA4 power up reset.

13.4. Timeout.

Hvis en cycle varer så længe, at signalet T4 samples aktiv to gange af en 100 us klok, R2-13, genereres en time out bus error. Sampling af signalet T4 sker i flip flop T6 ppl7.

13.5. HALT/RESET til 68000.

For at resette 68000 skal såvel RESET som HALT pin til 68000 aktiveres. Reset til 68000 tages fra IO bussen, V7-11. Når RESET i bussen er aktiv lav, aktiveres RESET til 68000, V8-06 og HALT V8-08.

Hvis 68000 går i HALT p.g.a. en dobbelt bus error, vil den drive HALT pinnen lav, hvilket medfører at kommando registeret resettes, X0-10, U9-08.

Hvis programmellet skriver i kommando registeret og sætter COM(8)=0 sættes T3-08, hvorved HALT til 68000 aktiveres. Selvmord. Herved resettes hele kommandoregisteret.

Hvis COM(8)=0, og DIAGNOSE i IO bussen aktiveres, sættes lav, fjernes reset fra kommando registeret, U9-08, og visse bit i kommando registeret sættes af V8-03, så CPU100 kan testes fra IO bussen.

Hvis COM(8) = 0, ERROR IN UNIT aktiv, og hvis ERROR RESET aktiveres, genereres igen RESET og HALT til 68000, V8-05 og V8-10.

14. Access til IO bussen.

CPU100 kan accesse IO bussen på to måder, Long Access to IO Bus og Short Access to IO Bus. Se fig. 16.

14.1. Short access to IO bus.

Ved denne access kan kun en del af IO bussens adresse rum accesses. En fjerdedel ef SEG1 er reserveret til access af IO bussen, ialt 256 kb. Dette lagerrum er fordelt som 64 kb i 16 enheder. Den nødvendige adressedekodning sker i PAL C70 position M0 pp11. Når M0-19, /BS, er lav, svarer adressen med videre til en kort access til IO bussen. Signalet BS indlæses på bagkanten af T2 i flip flop R9 pp3. Når flip flop R9-05 er lav, er cyclen en kort access til IO bussen. Når den korte access til IO bussen udføres, drives adresse bussen i IO bussen af A0, A1, A2, A4 og A5 pp1. Enable signalerne til adresse bus driverne genereres af X4 og Y6 pp3.

14.2. Long access to IO bus.

Ved denne access driver CPU100 hele adresse bussen. Unit number, Address space number, og segment number kommer fra den aktuelle segment descriptor i MMU´en. Den resteredne del af adressen kommer fra den logiske adresse fra 68000.

Hvis ACC(3)=1 i den aktuelle segment descriptor, er cyclen en long access til IO bussen. Denne dekodning sker i PAL C151 position N0 pp12. N0-12, BL, indlæses i flip flop R9 pp3 på bagkanten af T2. Hvis R9-08 er lav, er cyclen en lang access til IO bussen. De relevante dele af segment descriptoren latches i A8 og A9 pp1 til tiden T2 af signalet TPC, Y7-08 pp3. Når den lange access udføres drives adressebussen i IO bussen af A8, A9, A3, A4 og A5. Generering af enable signaler til adresse bus driverne er vist pp3.

14.3. Konflikt situationer.

Når R9-05 eller R9-08 er lav er cyclen en access til IO bussen. REQuest til IO bussen synkroniseres til bus klokken, CPB, i S3-09 og til den interne klok i S0-09. S0-09 anvendes til detektion af konflikt situationer, hvor 68000 har startet en cycle, der er en access til IO bussen, mens IO bussen er optaget af en cycle, der er en access til CPU100. Denne konflikt situation detekteres af PAL C20 position C1 pp3. Konflikten løses, idet cyclen fra IO bussen til CPU100 gennemføres, og cyclen fra 68000 genstartes. Når konflikten detekteres, generer PAL C10 /NA0. NA0 sætter C10-05, som synkroniseres til busklokken i C10-09. Herved vil prioritets PAL´en se ASB, B9-05, aktiv, tro bussen er optaget af en data transfer cycle og først starte den genoptagne cycle til bussen, når CPU100 er klar d.v.s, når C10-05 er nulstillet på bagkanten af den genoptagne cycles T2.

REQ synkroniseret til bus klokken, S3-09, sendes til prioriterings PAL position B9. Prioriteringen af cycles på IO bussen foretages i samarbejde af alle prioriterings PAL´er i SUPERMAX´en. Når CPU100 får IO bussen, aktiveres B0-12, /ASB0

sættes lav. ASBO anvendes til at styre enabling af busdrivere mod IO bussen.

Cyclen til bussen afsluttes, når address strobe fra 68000, ASC, går inaktiv og sætter R9-05 og R9-08. Så snart det synkroniserede REQ forsvinder, fjernes address strobe fra bussen, ASBO og cyclen er afsluttet.

fig. 1.

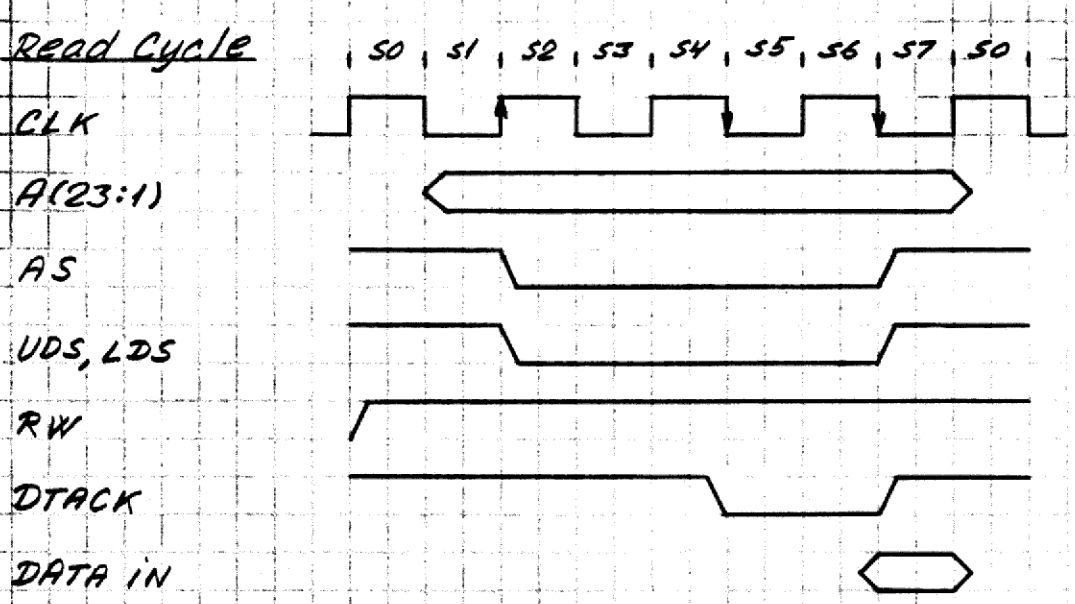
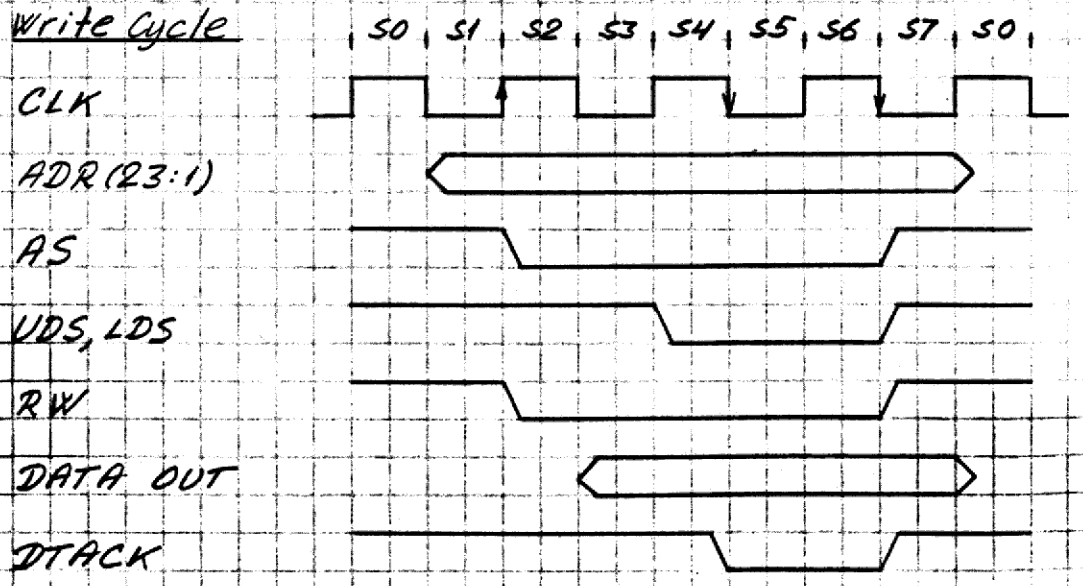

Read Cycle
S0 S1 S2 S3 S4 S5 S6 S7 S0
CLK
A(23:1)
AS
UDS, LDS
RW
DTACK
DATA IN

fig. 2.


Write Cycle
S0 S1 S2 S3 S4 S5 S6 S7 S0
CLK
ADR(23:1)
AS
UDS, LDS
RW
DATA OUT
DTACK

Lagerstyring. Struktur

Initialer/dato	Side
KAN 841009	
Revideret	Projekt

fig. 3.

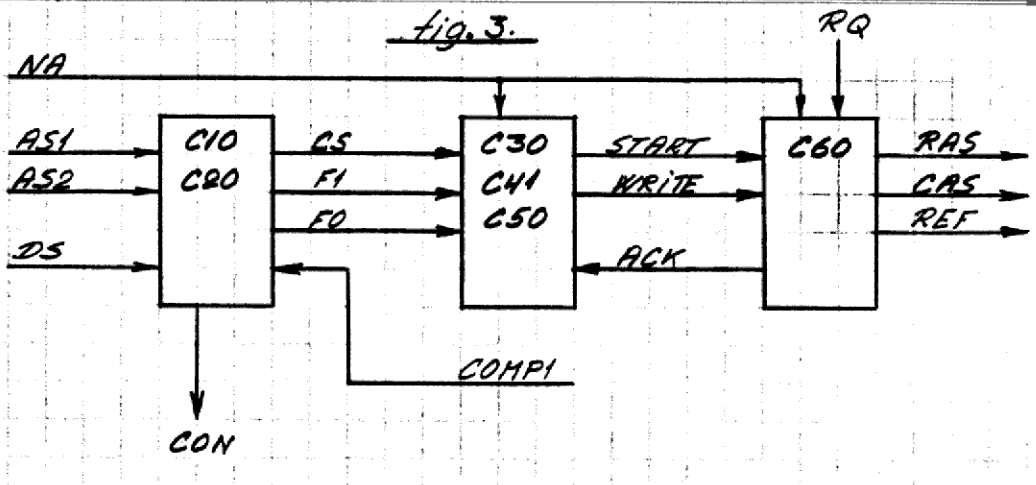
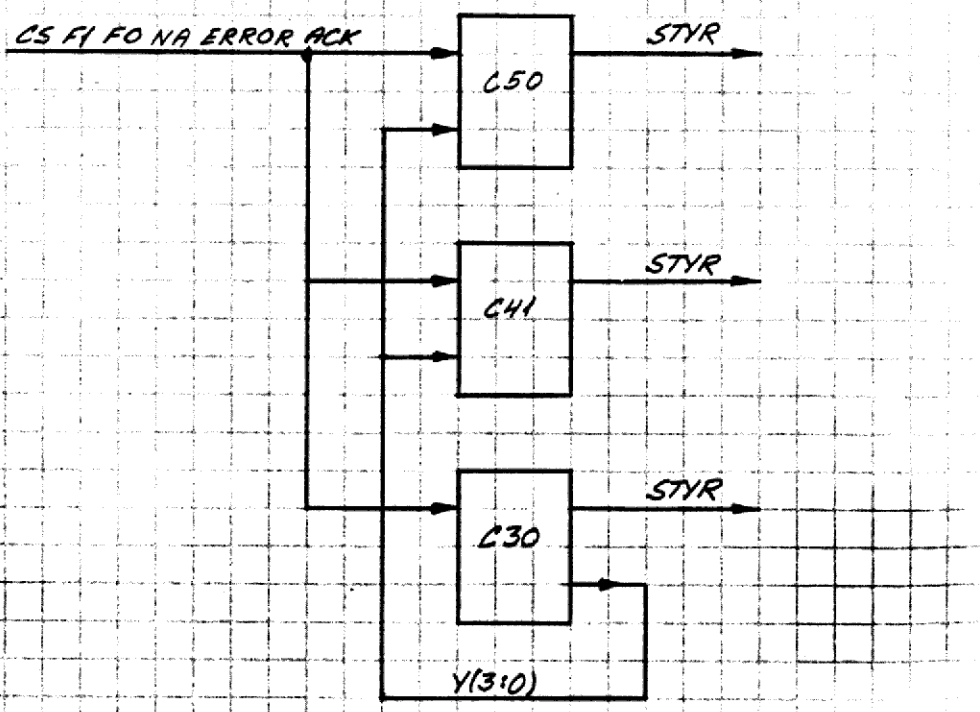


fig. 4.



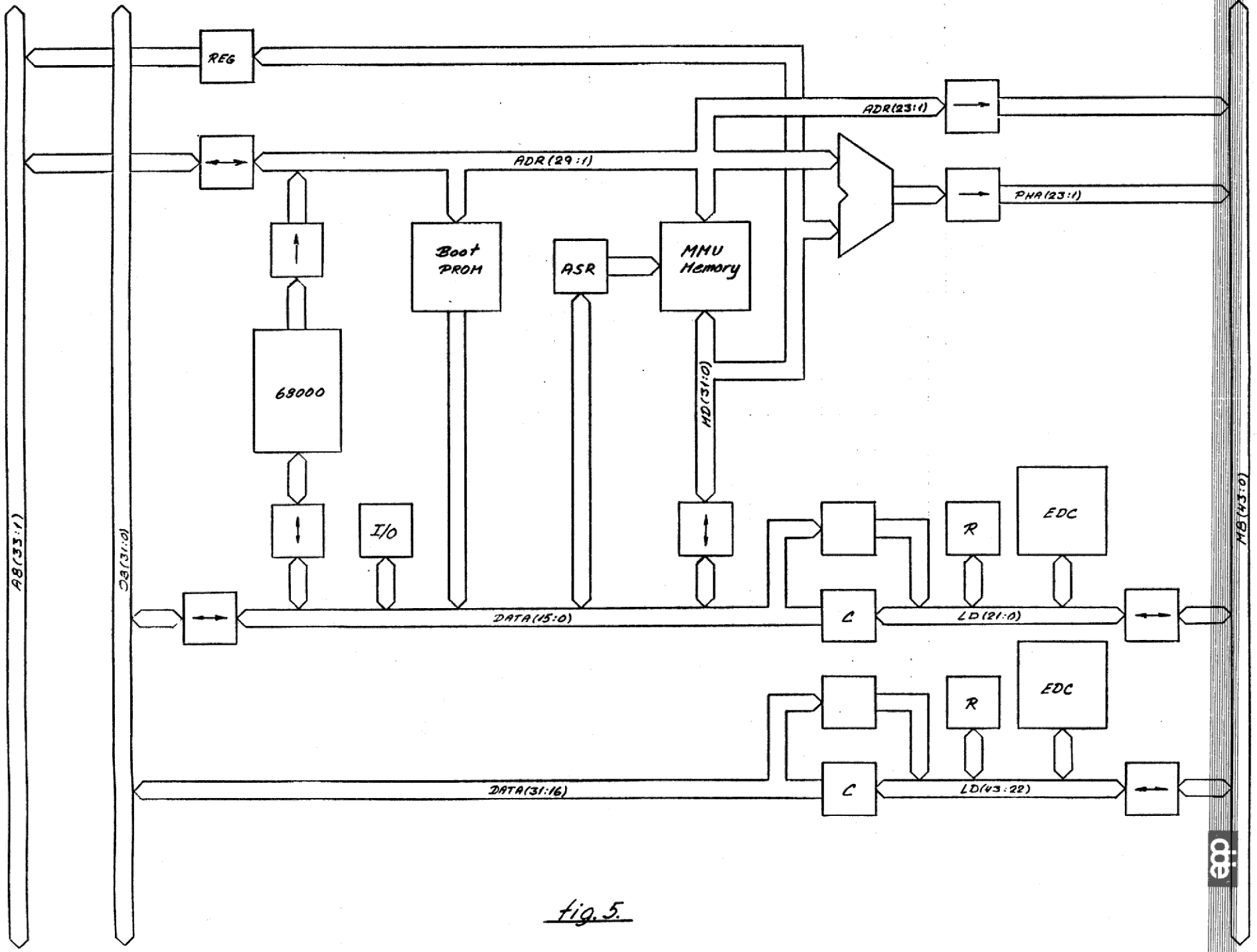

REG
ADR(23:1)
ADR(29:1)
PHA(23:1)
Boot PROM
ASR
MMU Memory
68000
HD(31:0)
I/O
R
EDC
C
LD(21:0)
DATA(15:0)
EDC
R
C
LD(43:22)
DATA(31:16)
AB(33:1)
DB(31:0)
MB(43:0)

fig. 5.

dansk data elektronik a/s

68000 Memory Read

Initialer/dato	Side
KAN 841009	
Revideret	Projekt

fig. 6

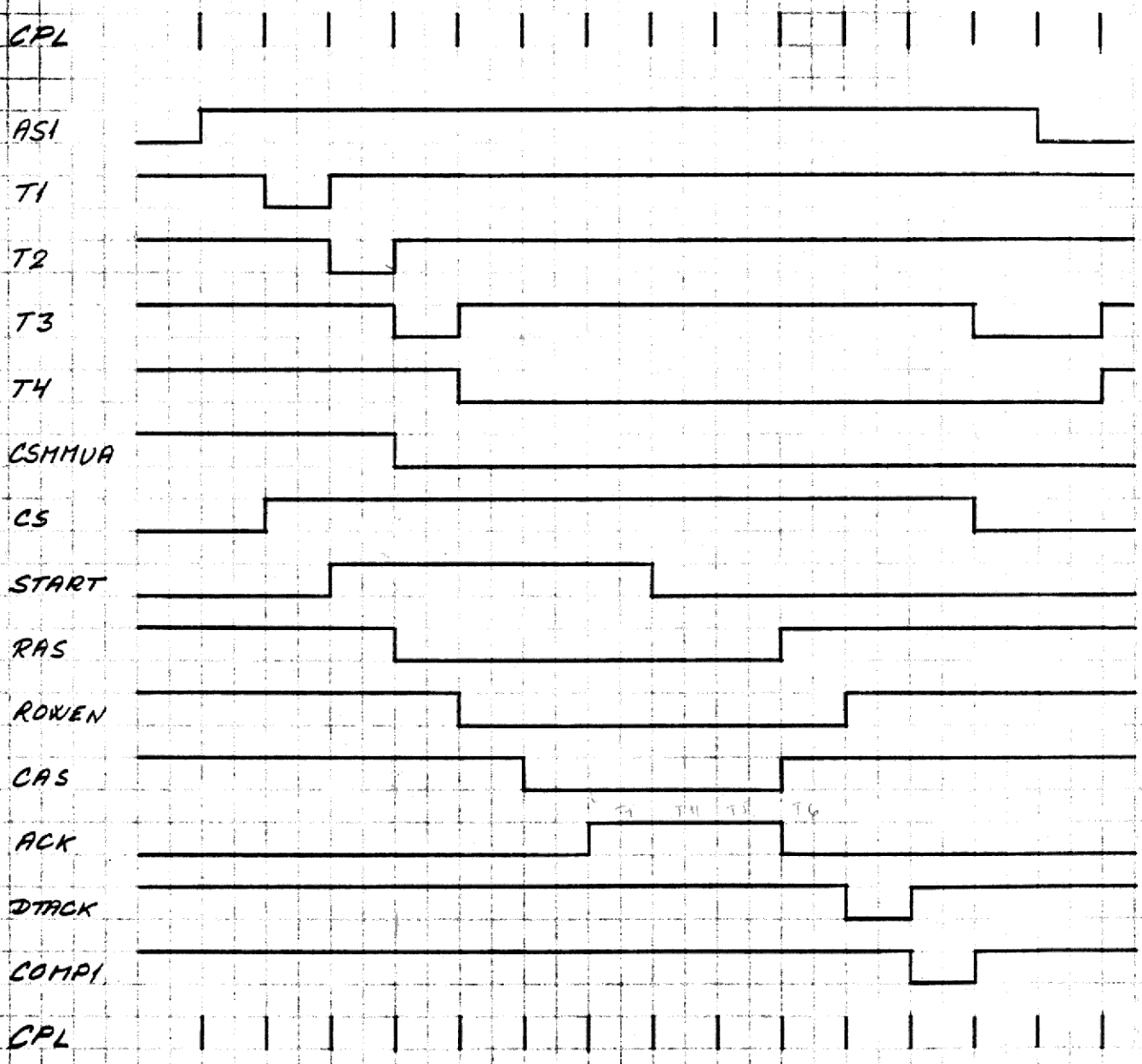
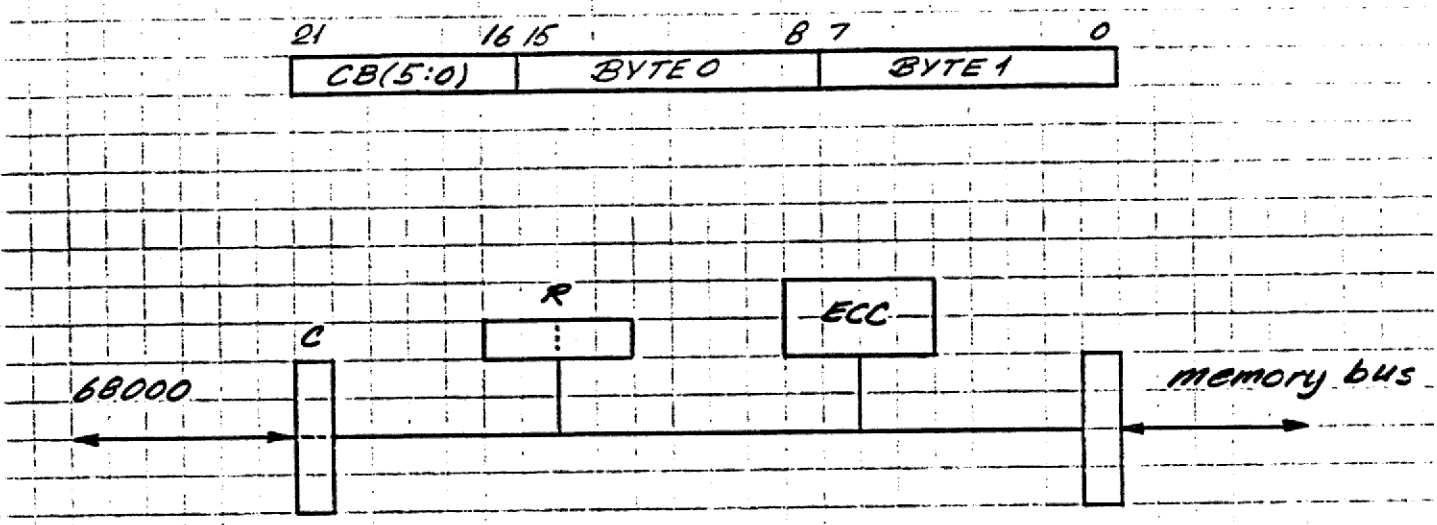


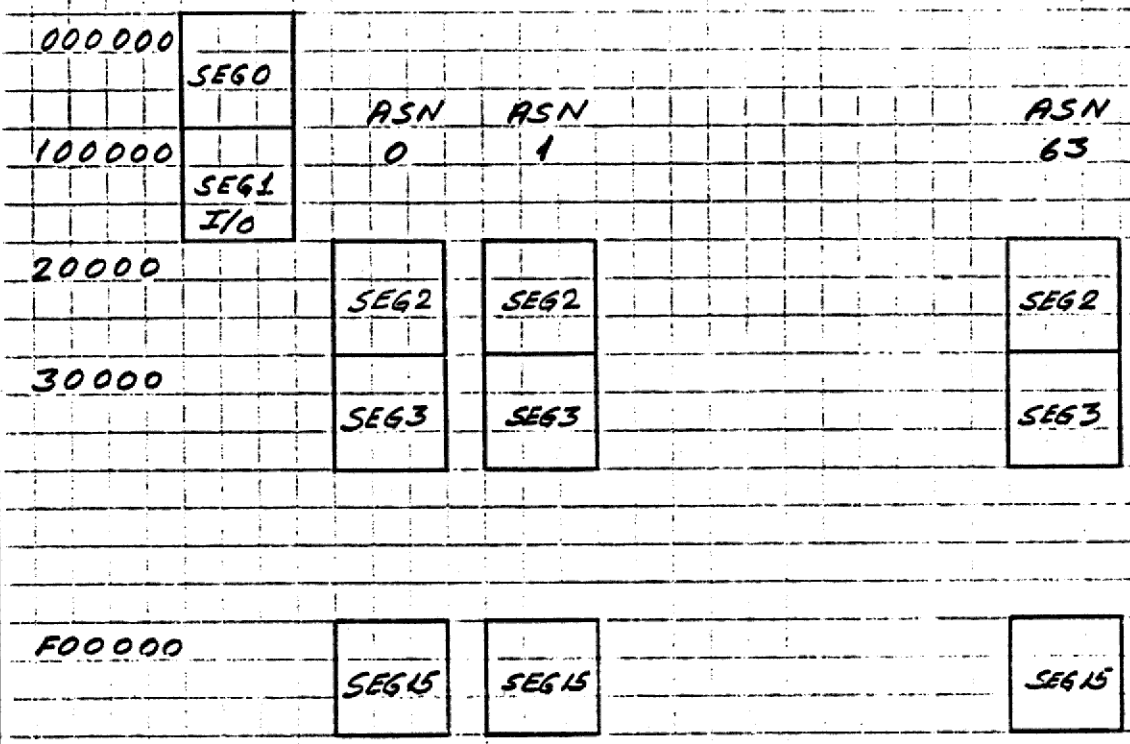
fig. 7



Byte write

1. Latch new byte in C.
2. Read word from memory
3. Latch word in ECC and R.
   If error
   Correct data
   Latch corrected data in R.
4. Output new byte from C
5. Output old byte from R
6. Generate check bit
7. Write word in memory.

Logical memory in SUPERMAX



64 Address Space Numbers ASN

14 logical segments pr. ASN

Variable physical segment size : 256b - 1Mb

User programs may use the same logical starting address.

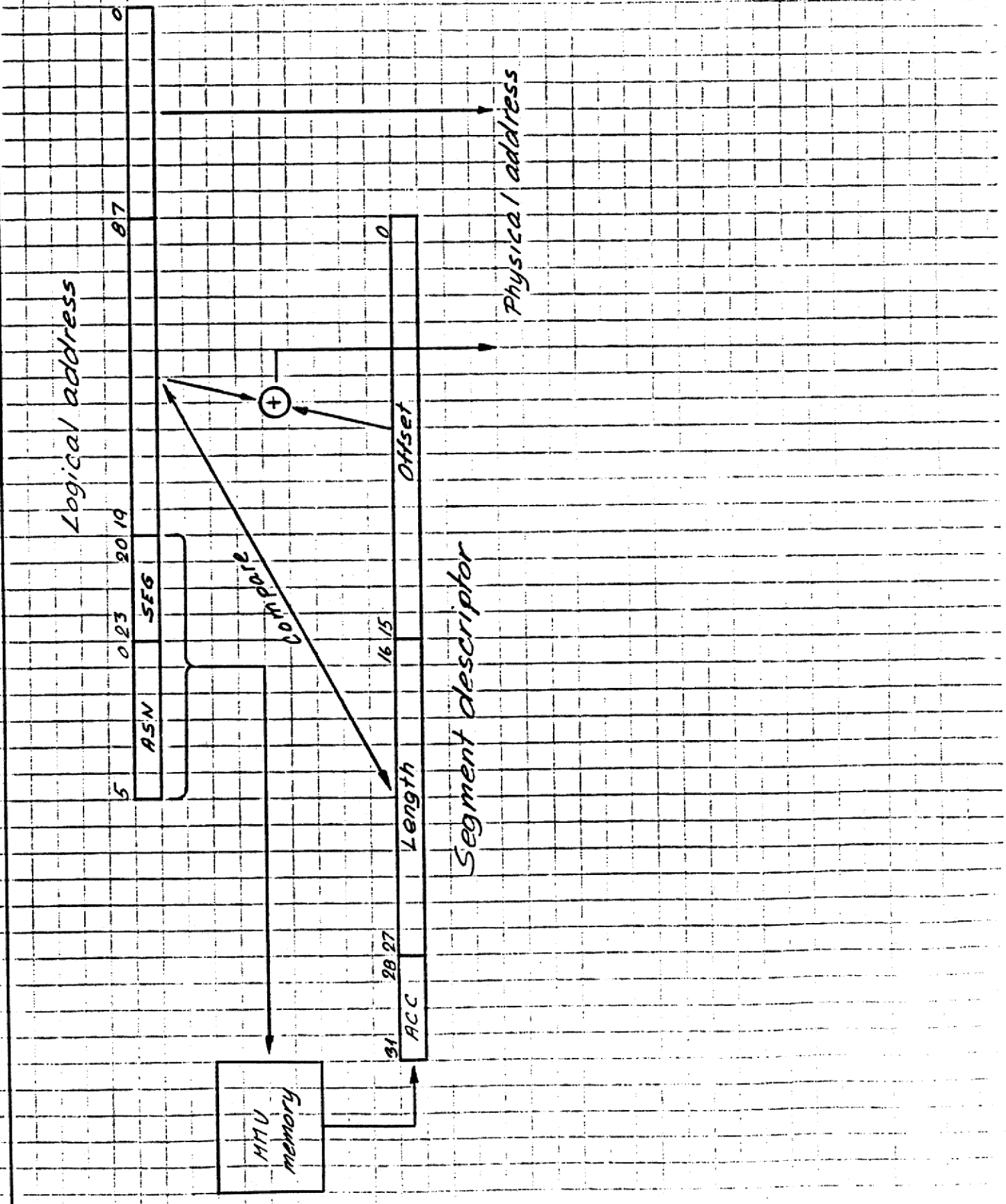
Different logical segments can be mapped to the same physical addresses.

Memory Management Unit MMU

Translates logical to physical addresses

Protects the main memory.

fig. 9.


Logical address
0
07
20 19
0 23
SEG
ASN
5
Compare
Physical address
0
Offset
16 15
Length
Segment descriptor
28 27
ACC
31
MMU
Memory

IO UNITS ON THE CPU MODULE.

fig. 10.

ADDRESS		
100XXX:	MMU	16 RW
101000:	Command Register	16 W
101002:	Status Register	16 R
101005:	Error Address	8 R
101006:	Syndrome Bits	16 R
101008:	Interrupt Mask	16 RW
10100A:	Interrupt Pending	16 R
10100D:	Ext. Interrupt Set/Reset.	8 W
101011:	USART Data Register	8 RW
101013:	USART Status/Command	8 RW
101014:	Display Register	16 W
101017:	Address Space Register	8 RW
10102X:	Unit Number	8 bytes read Odd addresses only

COMMAND REGISTER.

fig. 11.

BIT

15: Not used
14: Not used
13: Not used
12: Not used
11: Not used
10: Error Reset Out. (OC signal in IO bus)
9: Diagnose out. (OC signal in IO bus)
8: No Error in Unit.(OC signal in IO bus)
7: Enable Access to Registers from IO Bus
6: Disable Boot Loader PROM
5: Enable Protection against User in SEG 0 and SEG 1.
4: Enable Error Detection.
3: Enable Access Code Test
2: Enable Address Mapping.
1: Enable Access from IO Bus
0: Enable Access to IO Bus

IO UNITS ON THE CPU MODULE.

fig. 10.

ADDRESS		
100XXX:	MMU	16 RW
101000:	Command Register	16 W
101002:	Status Register	16 R
101005:	Error Address	8 R
101006:	Syndrome Bits	16 R
101008:	Interrupt Mask	16 RW
10100A:	Interrupt Pending	16 R
10100D:	Ext. Interrupt Set/Reset.	8 W
101011:	USART Data Register	8 RW
101013:	USART Status/Command	8 RW
101014:	Display Register	16 W
101017:	Address Space Register	8 RW
10102X:	Unit Number	8 bytes read Odd addresses only

COMMAND REGISTER.

fig. 11.

BIT	
15:	Not used
14:	Not used
13:	Not used
12:	Not used
11:	Not used
10:	Error Reset Out. (OC signal in IO bus)
9:	Diagnose out. (OC signal in IO bus)
8:	No Error in Unit.(OC signal in IO bus)
7:	Enable Access to Registers from IO Bus
6:	Disable Boot Loader PROM
5:	Enable Protection against User in SEG 0 and SEG 1.
4:	Enable Error Detection.
3:	Enable Access Code Test
2:	Enable Address Mapping.
1:	Enable Access from IO Bus
0:	Enable Access to IO Bus

STATUS REGISTER.

fig. 12

BIT	
15:	Not used
14:	Not used
13:	TxRDY Service USART
12:	RxRDY Service USART
11:	Power Loss
10:	Error in Unit
9:	Timer Interrupt
8:	Single Fault in Memory
7:	Double Fault in Memory
6:	Bus Error from IO Bus
5:	No Memory
4:	Time out
3:	Segment too Long
2:	Illegal Write
1:	Illegal Bus
0:	Illegal User

CPU MODULE INTERRUPT STRAP.

fig. 13

POSITION: Z5

PIN 1: POWER FAILURE	PIN 16: Not used
PIN 2: ERROR IN UNIT	PIN 15: Interrupt level 6
PIN 3: MEMORY VIOLATION	PIN 14: Interrupt level 5
PIN 4: FAULT IN MEMORY	PIN 13: Interrupt level 4
PIN 5: TIMER INTERRUPT	PIN 12: Interrupt level 3
PIN 6: SERVICE PORT	PIN 11: Interrupt level 2
PIN 7: Not used	PIN 10: Interrupt level 1
PIN 8: Not used	PIN 9 : Not used.

CPU MODULE CLOCK STRAP.

fig. 14.

POSITION: Z6

PIN 1:	6 Mhz output	PIN 16:	CPU clock input
PIN 2:	8 Mhz output	PIN 15:	CPU clock input
PIN 3:	10 Mhz output	PIN 14:	CPU clock input
PIN 4:	10 Mhz output	PIN 13:	BUS clock input
PIN 5:	12 Mhz output	PIN 12:	BUS clock input
PIN 6:	16 Mhz output	PIN 11:	BUS clock input
PIN 7:	16 Mhz output	PIN 10:	INTERNAL clock input
PIN 8:	20 Mhz output	PIN 9 :	INTERNAL clock input

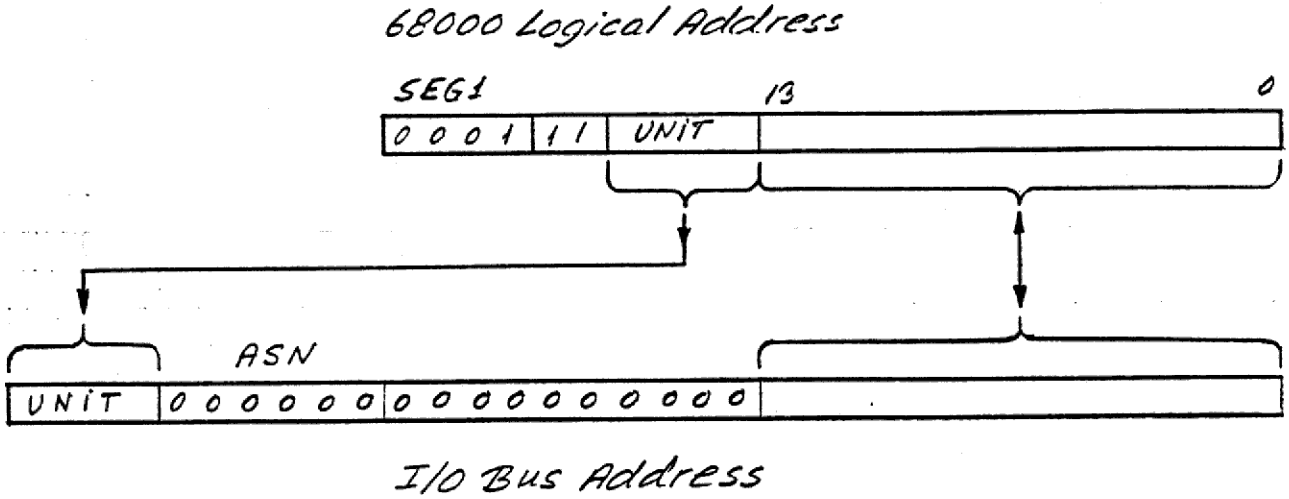
CPU MODULE TIMER STRAP.

fig. 15

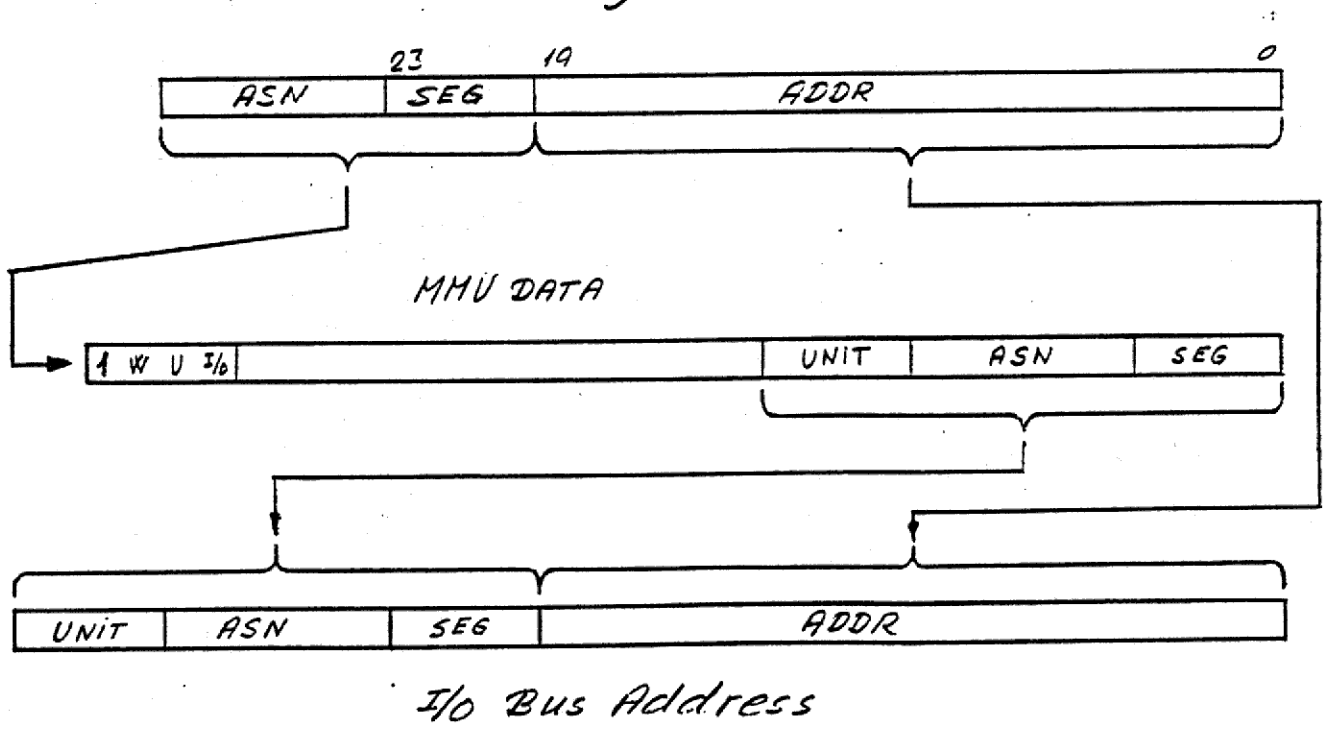
POSITION: Z4

PIN 1: BAUD rate input	PIN 16: 5 ms output
PIN 2: 9600 baud rate output	PIN 15: Interrupt input
PIN 3: 4800 baud rate output	PIN 14: 10 ms output
PIN 4: 2400 baud rate output	PIN 13: 20 ms output
PIN 5: 1200 baud rate output	PIN 12: Interrupt input
PIN 6: 600 baud rate output	PIN 11: 40 ms output
PIN 7: 300 baud rate output	PIN 10: 80 ms output
PIN 8: Not used	PIN 9 : Interrupt input

Short access to I/o bus:



Long access to I/o bus:



CPU MODULE INSTALLATION. fig. 17

The FIRST CPU module (Unit number 3):

This module drives the BUS CLOCK.

This module contains pull up resistors for various bus signals.

Check CPU TIMER strap position Z4:

PIN 1 and PIN 2 connected: Service USART 9600 baud.
PIN 11 and PIN 12 connected: Timer interrupt 40 ms.

Check CPU INTERRUPT strap position Z5:

PIN 5 and PIN 15 connected: Timer interrupt level 6.

Check CPU CLOCK strap position Z6:

PIN 3 and PIN 14 connceted: CPU clock 10 Mhz.
PIN 2 and PIN 13 connected: BUS clock 8 Mhz.
PIN 7 and PIN 10 connected: INTERNAL clock 16 Mhz.

Check SIL resistors:

Position SI25: 150 ohm
Position SI26: 150 ohm
Position SI27: 1.0 Kohm

Check unit and priority PAL:

Unit PAL position B8: C173
Priority PAL position B9: C163

CPU MODULE INSTALLATION. fig. 18.

NOT THE FIRST CPU MODULE.

Check CPU TIMER strap position Z4:

PIN 1 and PIN 2 connected: Service USART 9600 baud.
PIN 11 and PIN 12 connected: Timer interrupt 40 ms.

Check CPU INTERRUPT strap position Z5:

PIN 5 and PIN 15 connected: Timer interrupt level 6.

Check CPU CLOCK strap position Z6:

PIN 3 and PIN 14 connceted: CPU clock 10 Mhz.
PIN 7 and PIN 10 connected: INTERNAL clock 16 Mhz.

Check SIL resistors:

Position SI25: Not installed.
Position SI26: Not installed.
Position SI27: Not installed.

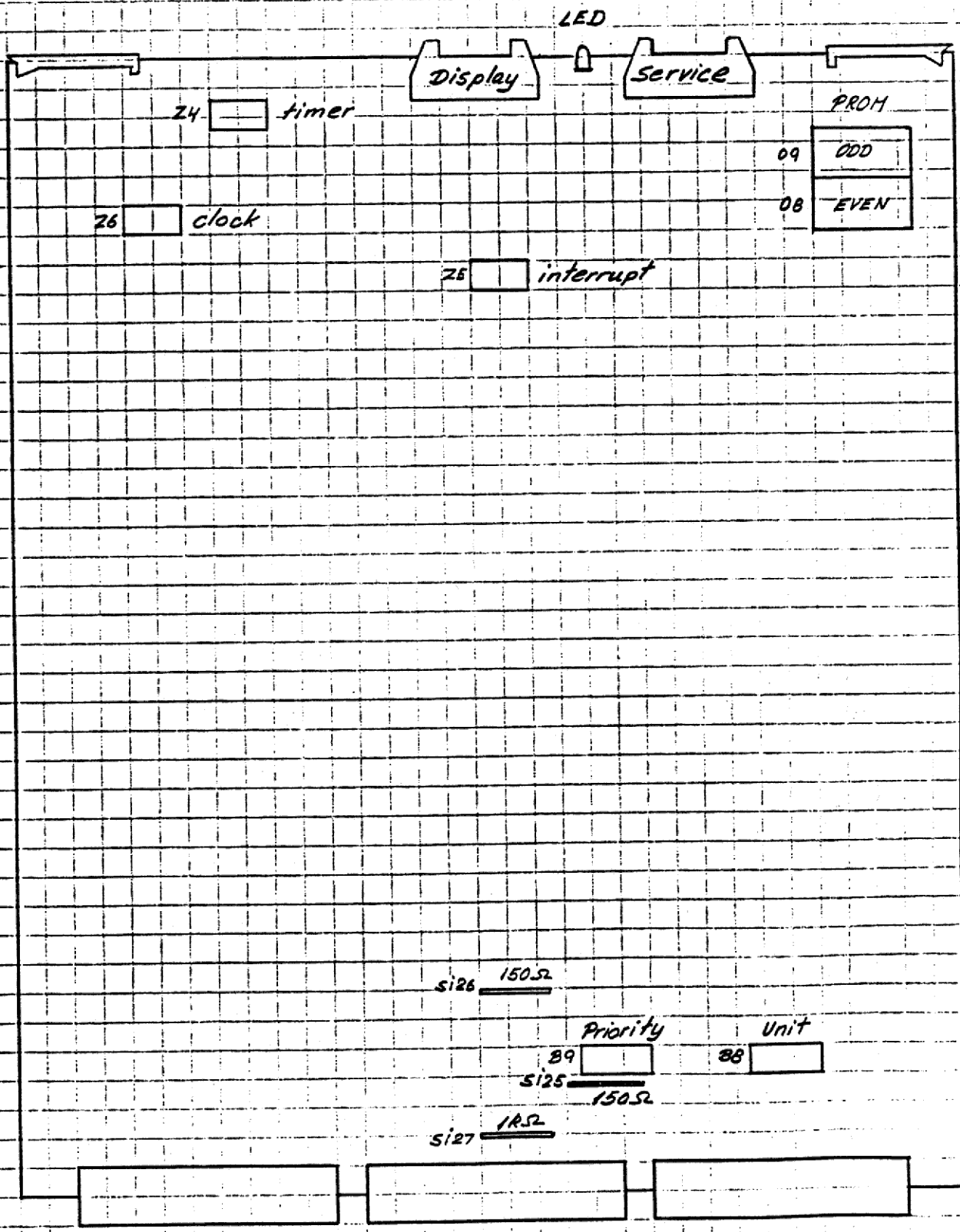
Check unit and priority PAL:

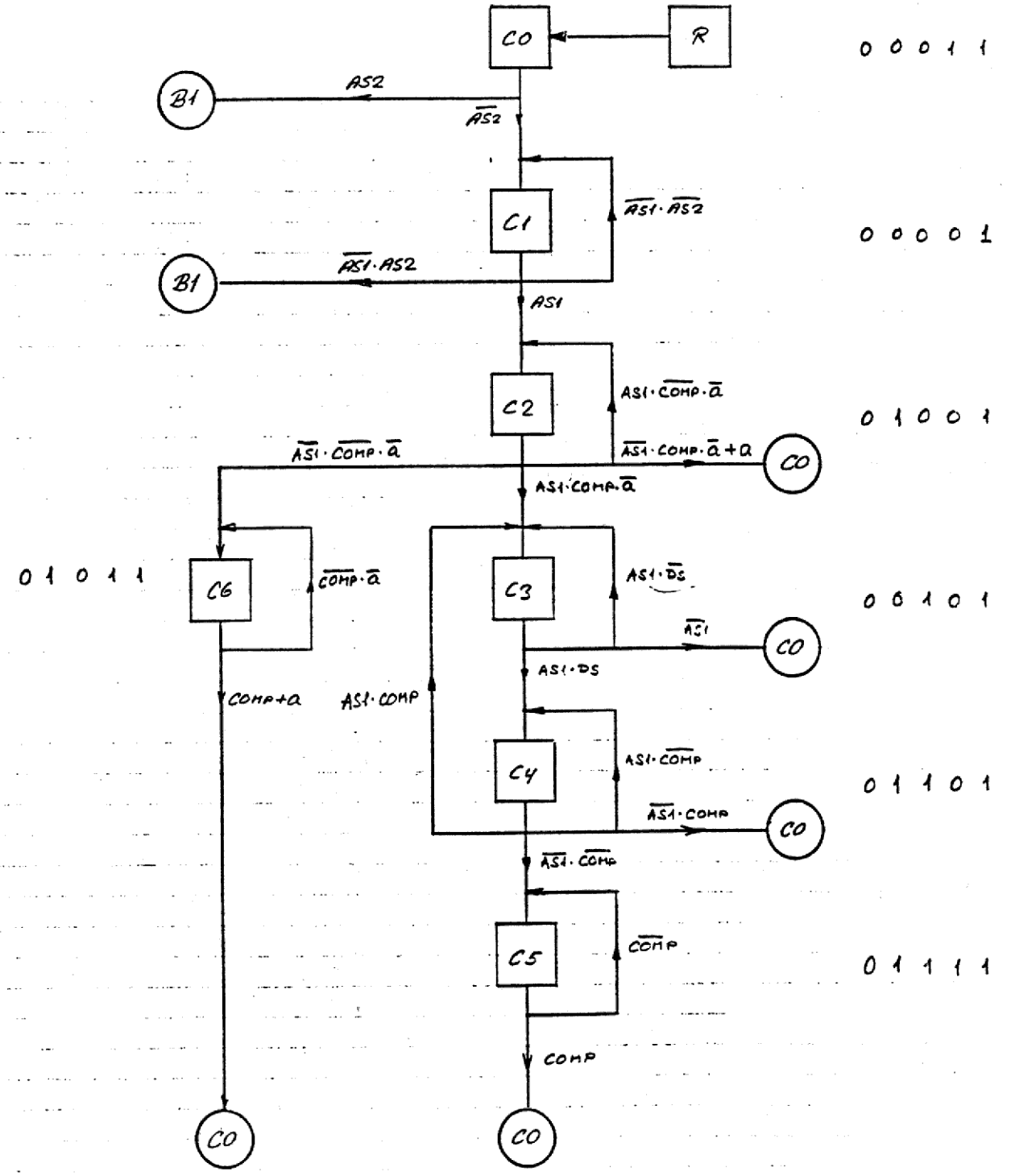
Unit PAL position B8: C170 to C175. See list.
Priority PAL position B9: C160 to C165. See list.

dansk data elektronik a/s

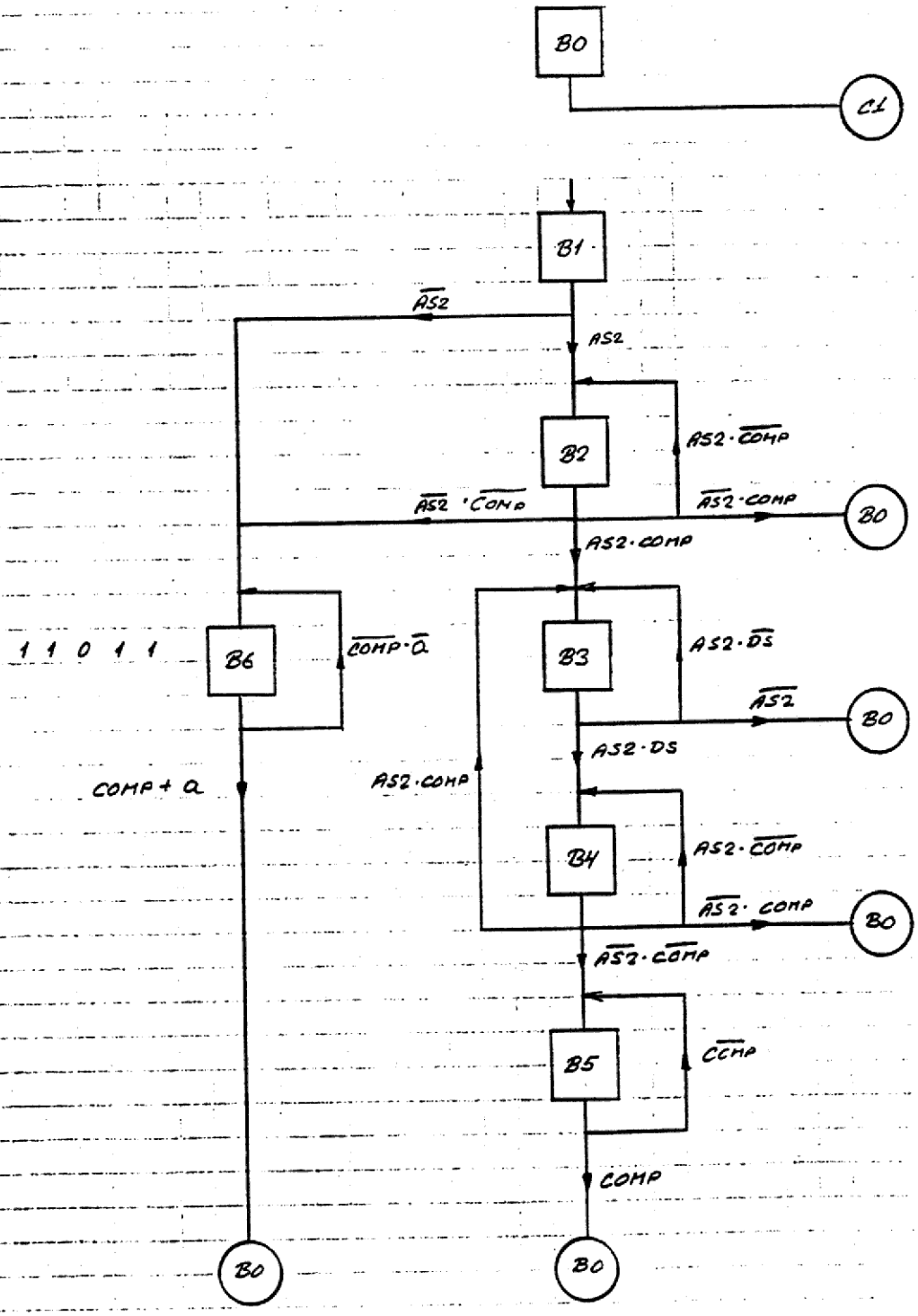
CPU Module fig19

Initialer/dato: KAN-840120
Revideret:
Side:
Projekt:




C0
R
0 0 0 1 1
B1
AS2
AS2
C1
AS1 · AS2
0 0 0 0 1
B1
AS1 · AS2
AS1
C2
AS1 · COMP · a
0 1 0 0 1
AS1 · COMP · a
AS1 · COMP · a + a
C0
AS1 · COMP · a
0 1 0 1 1
C6
COMP · a
COMP + a
C3
AS1 · DS
0 0 1 0 1
AS1
C0
AS1 · DS
AS1 · COMP
C4
AS1 · COMP
0 1 1 0 1
AS1 · COMP
C0
AS1 · COMP
C5
COMP
0 1 1 1 1
COMP
C0
C0

CON CS HW IC IB



```
t10

c0 = C2*/AS1*COMP*/A + C2*A + C3*/AS1 + C4*/AS1*COMP + C5*COMP
   + C6*COMP + C6*A + R

c1 = C0*/AS2 + C1*/AS1*/AS2 + B0

c2 = C1*AS1 + C2*AS1*/COMP*/A

c3 = C2*AS1*COMP*/A + C3*AS1*/DS + C4*AS1*COMP

c4 = C3*AS1*DS + C4*AS1*/COMP

c5 = C4*/AS1*/COMP + C5*/COMP

c6 = C2*/AS1*/COMP*/A + C6*/COMP*/A

b0 = B2*/AS2*COMP + B3*/AS2 + B4*/AS2*COMP + B5*COMP + B6*COMP
   + B6*A

b1 = C0*AS2 + C1*/AS1*AS2

b2 = B1*AS2 + B2*AS2*/COMP

b3 = B2*AS2*COMP + B3*AS2*/DS + B4*AS2*COMP

b4 = B3*AS2*DS + B4*AS2*/COMP

b5 = B4*/AS2*/COMP + B5*/COMP

b6 = B1*/AS2 + B2*/AS2*/COMP + B6*/COMP*/A
```

CODE		C O N	C S	M W	I C	I B
	C0:	0	0	0	1	1
	C1:	0	0	0	0	1
	C2:	0	1	0	0	1
	C3:	0	0	1	0	1
	C4:	0	1	1	0	1
	C5:	0	1	1	1	1
	C6:	0	1	0	1	1
	B0:	1	0	0	1	1
	B1:	1	0	0	1	0
	B2:	1	1	0	1	0
	B3:	1	0	1	1	0
	B4:	1	1	1	1	0
	B5:	1	1	1	1	1
	B6:	1	1	0	1	1

```
CON = b0 + b1 + b2 + b3 + b4 + b5 + b6

CON = C0*AS2 + C1*/AS1*AS2 + (B1+B2+B3+B4) + (B5+B6)

CS = c2 + c4 + c5 + c6 + b2 + b4 + b5 + b6

CS = C1*AS1 + C2*/COMP*/A + C3*AS1*DS + (C4+C5+B4+B5)*/COMP
   + (C6+B6)*/COMP*/A + B1 + B2*/COMP + B3*AS2*DS

MW = c3 + c4 + c5 + b3 + b4 + b5

MW = C2*AS1*COMP*/A + (C3+C4)*AS1 + (C4+C5+B4+B5)*/COMP
   + B2*AS2*COMP + (B3+B4)*AS2

IC = c0 + c5 + c6 + b0 + b1 + b2 + b3 + b4 + b5 + b6

IC = C0*AS2 + C1*/AS1*AS2 + C2*/AS1 + C2*A + (C3+C4)*/AS1
   + (C5+C6+B5+B6) + (B1+B2+B3+B4) +R
```

```
IB = c0 + c1 + c2 + c3 + c4 + c5 + c6 + b0 + b5 + b6

IB = (C0+C1)*/AS2 + C1*AS1 + C2 + (C3+C4) + (C5+C6+B5+B6)
   + B0 + (B1+B2+B3+B4)*/AS2 + R

CLCYST = c1 + b1

CLCYST = C0 + B0 + C1*/AS1
```

```
PAL16R8
SEKVENSMASKINE SEKA TIL STYRING AF LAGER
CPAL10: CPU PAL NR. 1 VERSION 0. POSITION F1
KAN 821101
CPL    /COMP  REQ     T4       AS1    AS2    /DS     /A      /RES    GND
GND     NC     NC     /CLCYST  /MW    /CON    /CS     /IB     /IC     VCC


CON :=/CON * /CS * /MW *  IC *  IB *  AS2 *  /RES
    + /CON * /CS * /MW * /IC *  IB * /AS1 *  AS2 *  /RES
    +  CON              *  IC * /IB *  /RES
    +  CON *  CS *         IC *  IB *  /RES


CS := /CON * /CS * /MW * /IC *  IB *  AS1 *  /RES
    + /CON *  CS * /MW * /IC *  IB * /COMP * /A *  /RES
    + /CON * /CS *  MW * /IC *  IB *  AS1 *  DS *  /RES
    +          CS *  MW *              /COMP *  /RES
    +          CS * /MW *  IC *  IB * /COMP * /A *  /RES
    +  CON * /CS * /MW *  IC * /IB *  /RES
    +  CON *  CS * /MW *  IC * /IB * /COMP *  /RES
    +  CON * /CS *  MW *  IC * /IB *  AS2 *DS *  /RES


MW := /CON *  CS * /MW * /IC *  IB *  AS1 *  COMP * /A *  /RES
    + /CON *        MW * /IC *  IB *  AS1 *  /RES
    + /CON *  CS *  MW *         IB * /COMP *  /RES
    +  CON *  CS *  MW *  IC *        /COMP *  /RES
    +  CON *  CS * /MW *  IC * /IB *  AS2 * COMP *  /RES
    +  CON *        MW *  IC * /IB *  AS2 *  /RES


IC := /CON * /CS * /MW *  IC *  IB *  AS2 *  /RES
    + /CON * /CS * /MW * /IC *  IB * /AS1 *  AS2 *  /RES
    + /CON *  CS * /MW * /IC *  IB * /AS1 *  /RES
    + /CON *  CS * /MW * /IC *  IB *  A *  /RES
    + /CON *        MW * /IC *  IB * /AS1 *  /RES
    +          CS *        IC *  IB *  /RES
    +  CON *                IC * /IB *  /RES
    + /CON * /CS * /MW * /IC * /IB *  /RES
```

```
IB := /CON * /CS * /MW *        IB * /AS2 *  /RES
    + /CON * /CS * /MW * /IC *  IB *  AS1 *  /RES
    + /CON *  CS * /MW * /IC *  IB *  /RES
    + /CON *       MW * /IC *  IB *  /RES
    +         CS *       IC *  IB *  /RES
    +  CON * /CS * /MW *  IC *  IB *  /RES
    +  CON *              IC * /IB * /AS2 *  /RES
    + /CON * /CS * /MW * /IC * /IB * /RES


CLCYST:=
      /CON * /CS * /MW *  IC *  IB *  /RES
    +  CON * /CS * /MW *  IC *  IB *  /RES
    + /CON * /CS * /MW * /IC *  IB * /AS1 *  /RES

FUNCTION TABLE:

CPL RES AS1 AS2 DS A COMP CLCYST CON CS MW IC IB
-----------------------------------------------
CHXXXXXL LLLLL R
CHXXXXXL LLLLL R
CLXXXXXL LLLHH C0
CLXLXXXH LLLLH C1
CLLLXXXH LLLLH C1
CLHXXXXL LHLLH C2
CLHXXLLL LHLLH C2
CLLXXLHL LLLHH C0
CLXLXXXH LLLLH C1
CLHXXXXL LHLLH C2
CLXXXHXL LLLHH C0
CLXLXXXH LLLLH C1
CLHXXXXL LHLLH C2
CLLXXLLL LHLHH C6
CLXXXLLL LHLHH C6
CLXXXHXL LLLHH C0
CLXLXXXH LLLLH C1
CLHXXXXL LHLLH C2
CLLXXLLL LHLHH C6
CLXXXXHL LLLHH C0
```

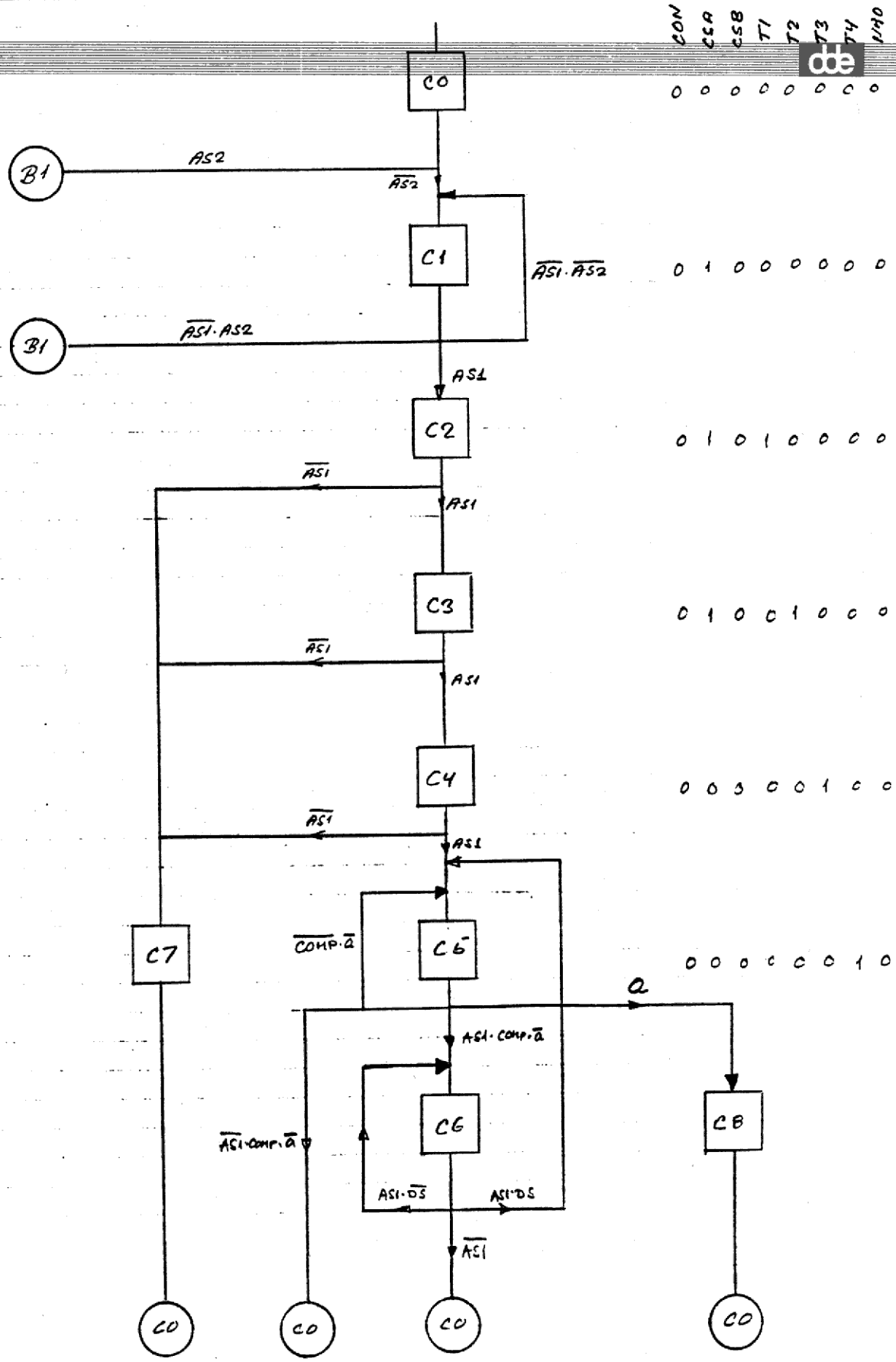
```
CLXLXXXH LLLLH C1
CLHXXXXL LHLLH C2
CLHXXLHL LLHLH C3
CLHXLXXL LLHLH C3
CLLXXXXL LLLHH C0
CLXLXXXH LLLLH C1
CLHXXXXL LHLLH C2
CLHXXLHL LLHLH C3
CLHXHXXL LHHLH C4
CLHXXXLL LHHLH C4
CLHXXXHL LLHLH C3
CLHXHXXL LHHLH C4
CLLXXXHL LLLHH C0
CLXLXXXH LLLLH C1
CLHXXXXL LHLLH C2
CLHXXLHL LLHLH C3
CLHXHXXL LHHLH C4
CLLXXXLL LHHHH C5
CLXXXXLL LHHHH C5
CLXXXXHL LLLHH C0
CLXHXXXH HLLHL B1
CLXLXXXL HHLHH B6
CLXXXLLL HHLHH B6
CLXXXHXL HLLHH B0
CLXXXXXH LLLLH C1
CLLHXXXH HLLHL B1
CLXLXXXL HHLHH B6
CLXXXXHL HLLHH B0
CLXXXXXH LLLLH C1
CLLHXXXH HLLHL B1
CLXHXXXL HHLHL B2
CLXHXXLL HHLHL B2
CLXLXXLL HHLHH B6
CLXXXXHL HLLHH B0
CLXXXXXH LLLLH C1
CLLHXXXH HLLHL B1
CLXHXXXL HHLHL B2
CLXLXXHL HLLHH B0
CLXXXXXH LLLLH C1
```

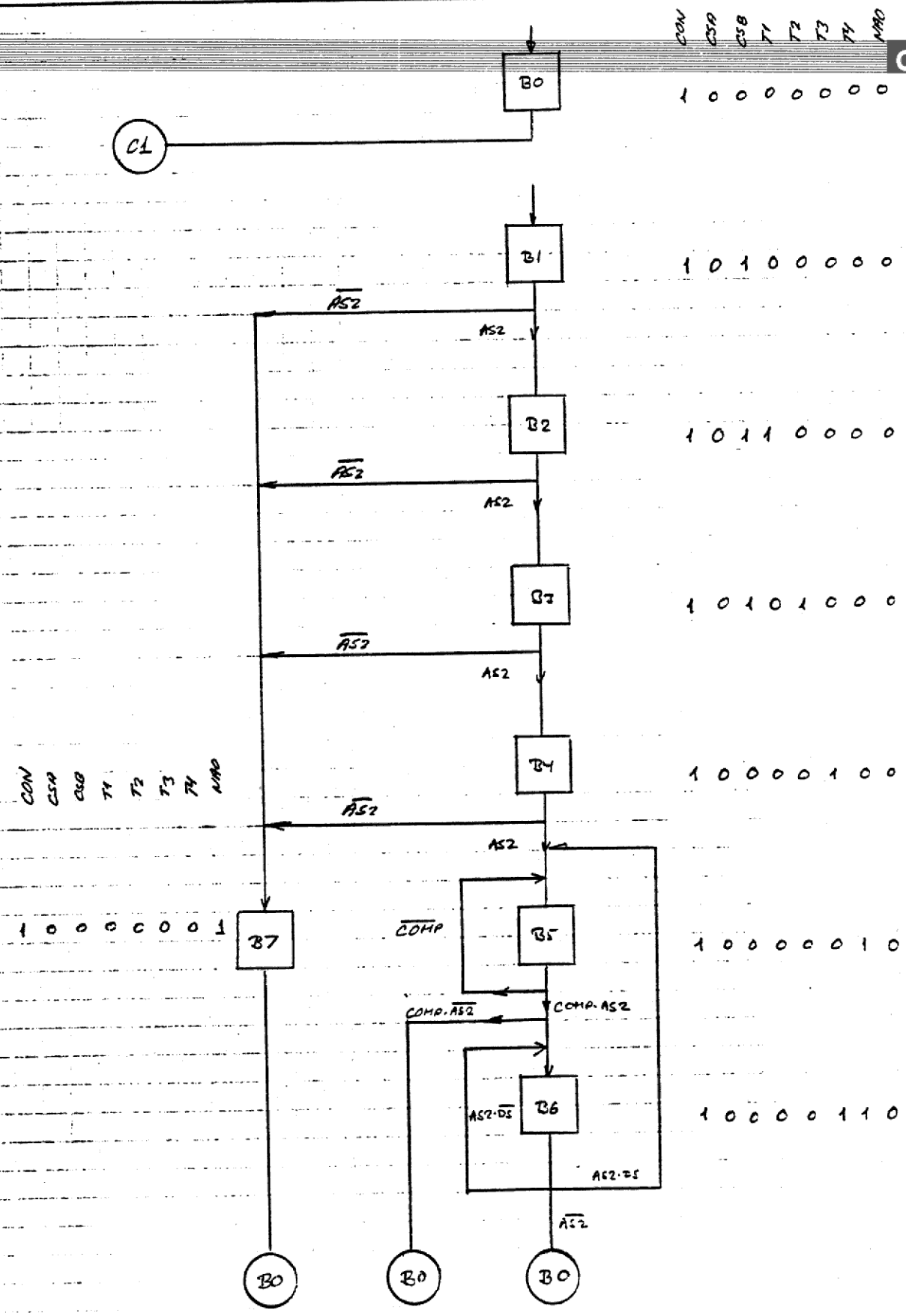
```
CLLHXXXH HLLHL B1
CLXHXXXL HHLHL B2
CLXHXXHL HLHHL B3
CLXHLXXL HLHHL B3
CLXLXXXL HLLHH B0
CLXXXXXH LLLLH C1
CLLHXXXH HLLHL B1
CLXHXXXL HHLHL B2
CLXHXXHL HLHHL B3
CLXHHXXL HHHHL B4
CLXHXXLL HHHHL B4
CLXHXXHL HLHHL B3
CLXHHXXL HHHHL B4
CLXLXXHL HLLHH B0
CLXXXXXH LLLLH C1
CLLHXXXH HLLHL B1
CLXHXXXL HHLHL B2
CLXHXXHL HLHHL B3
CLXHHXXL HHHHL B4
CLXLXXLL HHHHH B5
CLXXXXLL HHHHH B5
CLXXXXHL HLLHH B0
CLXXXXXH LLLLH C1
CLLHXXXH HLLHL B1
CLXHXXXL HHLHL B2
CLXHXXHL HLHHL B3
CLXHHXXL HHHHL B4
CLXLXXLL HHHHH B5
CHXXXXXL LLLLL R
-----------------
DESCRIPTION

;BLA BLA
```

dansk data elektronik aps

Styring af MMU CPAL20

Initialer/dato	Side
KAN 821029	1
Revideret	Projekt




CON CSA CSB T1 T2 T3 T4 UMD
B0
1 0 0 0 0 0 0 0
C1
B1
1 0 1 0 0 0 0 0
AS2
AS2
B2
1 0 1 1 0 0 0 0
AS2
AS2
B3
1 0 1 0 1 0 0 0
AS2
AS2
B4
1 0 0 0 0 1 0 0
AS2
AS2
B7
1 0 0 0 0 0 0 1
COMP
B5
1 0 0 0 0 0 1 0
COMP·AS2
COMP·AS2
AS2·DS
B6
1 0 0 0 0 1 1 0
AS2·DS
AS2
B0
B0
B0

```
t20

A  = AS2*REQ

c0 = C5*/AS1*COMP*/A + C6*/AS1 + C7 + C8

c1 = C0*/AS2 + C1*/AS1*/AS2 + B0

c2 = C1*AS1

c3 = C2*AS1

c4 = C3*AS1

c5 = C4*AS1 + C5*/COMP*/A + C6*AS1*DS

c6 = C5*AS1*COMP*/A + C6*AS1*/DS

c7 = C2*/AS1 + C3*/AS1 + C4*/AS1

c8 = C5*A

b0 = B5*COMP*/AS2 + B6*/AS2 + B7

b1 = C0*AS2 + C1*/AS1*AS2

b2 = B1*AS2

b3 = B2*AS2

b4 = B3*AS2

b5 = B4*AS2 + B5*/COMP + B6*AS2*DS

b6 = B5*COMP*AS2 + B6*AS2*/DS

b7 = B1*/AS2 + B2*/AS2 + B3*/AS2 + B4*/AS2
```

CODE

	C O N	C S A	C S B	T 1	T 2	T 3	T 4	N A 0
C0:	0	0	0	0	0	0	0	0
C1:	0	1	0	0	0	0	0	0
C2:	0	1	0	1	0	0	0	0
C3:	0	1	0	0	1	0	0	0
C4:	0	0	0	0	0	1	0	0
C5:	0	0	0	0	0	0	1	0
C6:	0	0	0	0	0	1	1	0
C7:	0	0	0	0	0	0	0	1
C8:	0	0	0	0	0	0	1	1
B0:	1	0	0	0	0	0	0	0
B1:	1	0	1	0	0	0	0	0
B2:	1	0	1	1	0	0	0	0
B3:	1	0	1	0	1	0	0	0
B4:	1	0	0	0	0	1	0	0
B5:	1	0	0	0	0	0	1	0
B6:	1	0	0	0	0	1	1	0
B7:	1	0	0	0	0	0	0	1

$$CON = b0 + b1 + b2 + b3 + b4 + b5 + b6 + b7$$

$$CON = C0*AS2 + C1*/AS1*AS2 + (B1+B2) + B3 + B4 + (B5+B6) + B7$$

$$CSA = c1 + c2 + c3$$

$$CSA = C0*/AS2 + C1*/AS1*/AS2 + B0 + C1*AS1 + C2*AS1$$

$$CSB = b1 + b2 + b3$$

$$CSB = C0*AS2 + C1*/AS1*AS2 + B1*AS2 + B2*AS2$$

$$T1 = c2 + b2$$

$$T1 = C1*AS1 + B1*AS2$$

$$T2 = c3 + b3$$

$$T2 = C2*AS1 + B2*AS2$$

$$T3 = c4 + c6 + b4 + b6$$

$$T3 = C3*AS1 + C5*AS1*COMP*/REQ + C5*AS1*COMP*/AS2 + C6*AS1*/DS + B3*AS2 + B5*AS2*COMP + B6*AS2*/DS$$

$$T4 = c5 + c6 + c8 + b5 + b6$$

$$T4 = C4*AS1 + (C5+C6)*AS1 + C5*/COMP + C5*AS2*REQ + B4*AS2 + (B5+B6)*AS2 + B5*/COMP$$

$$NA0 = c7 + c8 + b7$$

$$NA0 = C2*/AS1 + C3*/AS1 + C4*/AS1 + C5*AS2*REQ + (B1+B2)*/AS2 + B3*/AS2 + B4*/AS2$$

```
PAL16R8
SEKVENSMASKINE TIL STYRING AF MMU
CPAL20: CPU PAL NR. 2 VERSION 0. POSITION C1.
KAN 821101
CPL A1 A2 REQ /COMP NC /DS NC /R GND
GND /N /T4 /T3 /T2 /T1 /CB /CA /CN VCC

CN:=/R*/CN*/CA*/CB*/T1*/T2*/T3*/T4*/N
*A2
+/R*/CN*CA*/CB*/T1*/T2*/T3*/T4*/N
*/A1*A2
+/R*CN*/CA*CB*/T2*/T3*/T4*/N
+/R*CN*/CA*CB*/T1*T2*/T3*/T4*/N
+/R*CN*/CA*/CB*/T1*/T2*T3*/T4*/N
+/R*CN*/CA*/CB*/T1*/T2*T4*/N
+/R*CN*/CA*/CB*/T1*/T2*/T3*/T4*N



CA:=/R*/CN*/CA*/CB*/T1*/T2*/T3*/T4*/N
*/A2
+/R*/CN*CA*/CB*/T1*/T2*/T3*/T4*/N
*/A1*/A2
+/R*CN*/CA*/CB*/T1*/T2*/T3*/T4*/N
+/R*/CN*CA*/CB*/T1*/T2*/T3*/T4*/N
*A1
+/R*/CN*CA*/CB*T1*/T2*/T3*/T4*/N
*A1

CB:=/R*/CN*/CA*/CB*/T1*/T2*/T3*/T4*/N
*A2
+/R*/CN*CA*/CB*/T1*/T2*/T3*/T4*/N
*/A1*A2
+/R*CN*/CA*CB*/T1*/T2*/T3*/T4*/N
*A2
+/R*CN*/CA*CB*T1*/T2*/T3*/T4*/N
*A2
```

```
T1:=/R*/CN*CA*/CB*/T1*/T2*/T3*/T4*/N
*A1
+/R*CN*/CA*CB*/T1*/T2*/T3*/T4*/N
*A2

T2:=/R*/CN*CA*/CB*T1*/T2*/T3*/T4*/N
*A1
+/R*CN*/CA*CB*T1*/T2*/T3*/T4*/N
*A2

T3:=/R*/CN*CA*/CB*/T1*T2*/T3*/T4*/N
*A1
+/R*/CN*/CA*/CB*/T1*/T2*/T3*T4*/N
*A1*/REQ*COMP
+/R*/CN*/CA*/CB*/T1*/T2*/T3*T4*/N
*A1*/A2*COMP
+/R*/CN*/CA*/CB*/T1*/T2*T3*T4*/N
*A1*/DS
+/R*CN*/CA*CB*/T1*T2*/T3*/T4*/N
*A2
+/R*CN*/CA*/CB*/T1*/T2*/T3*T4*/N
*A2*COMP
+/R*CN*/CA*/CB*/T1*/T2*T3*T4*/N
*A2*/DS


T4:=/R*/CN*/CA*/CB*/T1*/T2*T3*/T4*/N
*A1
+/R*/CN*/CA*/CB*/T1*/T2*T4*/N
*A1
+/R*/CN*/CA*/CB*/T1*/T2*/T3*T4*/N
*/COMP
+/R*/CN*/CA*/CB*/T1*/T2*/T3*T4*/N
*A2*REQ
+/R*CN*/CA*/CB*/T1*/T2*T3*/T4*/N
*A2
+/R*CN*/CA*/CB*/T1*/T2*T4*/N
*A2
+/R*CN*/CA*/CB*/T1*/T2*/T3*T4*/N
*/COMP
```

```
N:=/R*/CN*CA*/CB*T1*/T2*/T3*/T4*/N
*/A1
+/R*/CN*CA*/CB*/T1*T2*/T3*/T4*/N
*/A1
+/R*/CN*/CA*/CB*/T1*/T2*T3*/T4*/N
*/A1
+/R*/CN*/CA*/CB*/T1*/T2*/T3*T4*/N
*A2*REQ
+/R*CN*/CA*CB*/T2*/T3*/T4*/N
*/A2
+/R*CN*/CA*CB*/T1*T2*/T3*/T4*/N
*/A2
+/R*CN*/CA*/CB*/T1*/T2*T3*/T4*/N
*/A2

FUNCTION TABLE:
CPL A1 A2 REQ COMP DS R CN CA CB T1 T2 T3 T4 N
-----------------------------------------------
CXXXXXHLLLLLLLLC0
CHHXXHHLLLLLLLLC0
CXLXXXLLHLLLLLLC1
CLLXXXLLHLLLLLLC1
CHXXXXLLHLHLLLLC2
CLXXXXLLLLLLLLHC7
CXXXXXXLLLLLLLLC0
CXLXXXLLHLLLLLLC1
CHXXXXLLHLHLLLLC2
CHXXXXLLHLLHLLLC3 10
CLXXXXLLLLLLLLHC7
CXXXXXXLLLLLLLLC0
CXLXXXLLHLLLLLLC1
CHXXXXLLHLHLLLLC2
CHXXXXLLHLLHLLLC3
CHXXXXLLLLLLHLLC4
CLXXXXLLLLLLLLHC7
CXXXXXXLLLLLLLLC0
CXLXXXLLHLLLLLLC1
CHXXXXLLHLHLLLLC2   20
CHXXXXLLHLLHLLLC3
```

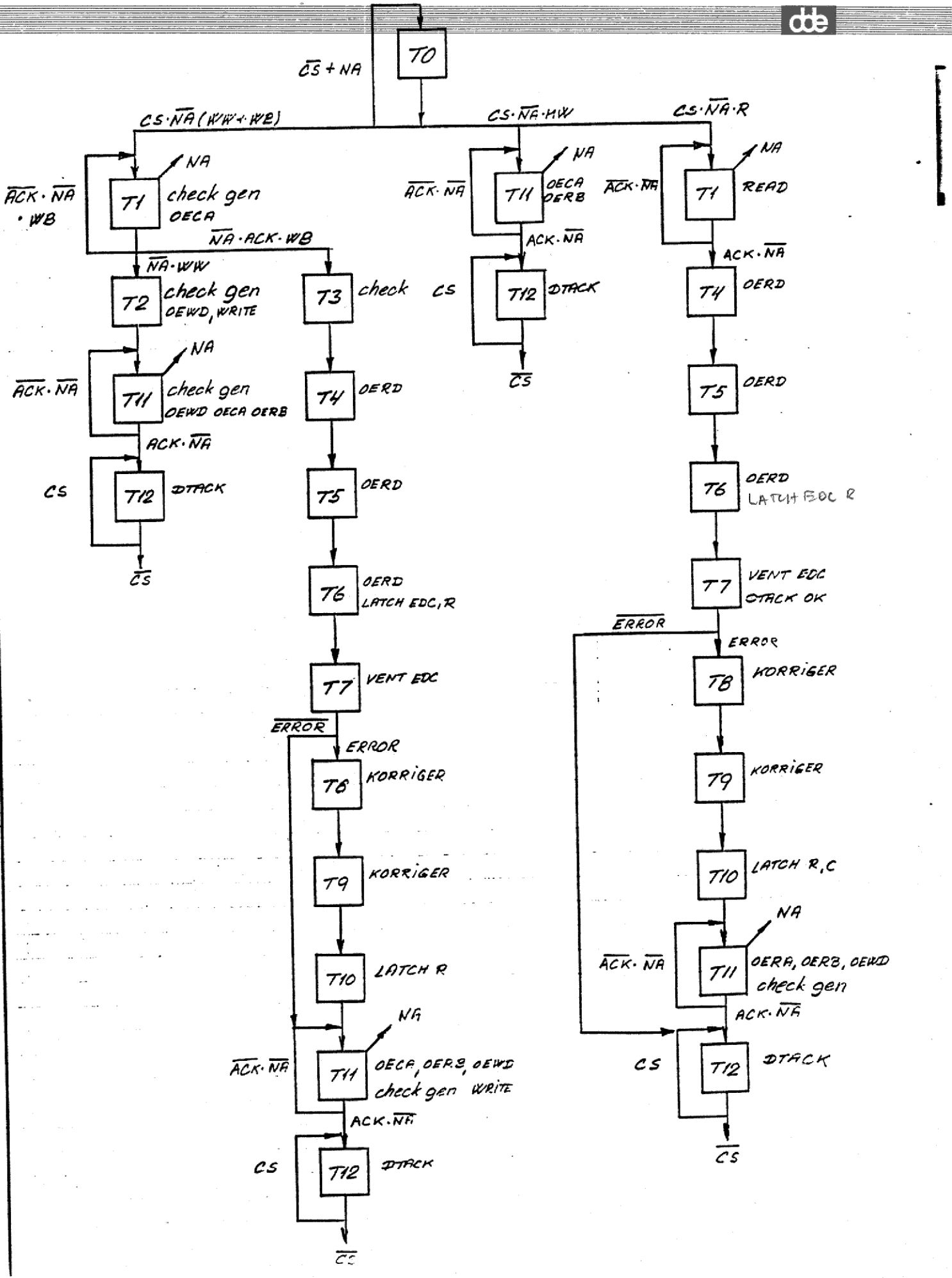
CHXXXXLLLLLLLHLLC4
CHXXXXLLLLLLLLHLC5
CXXLLXLLLLLLLLHLC5
CXLXLXLLLLLLLLHLC5
CLXLHXXLLLLLLLLLC0
CXLXXXLLHLLLLLLC1
CHXXXXLLHLHLLLLC2
CHXXXXLLHLLHLLLC3
CHXXXXLLLLLLLHLLC4
CHXXXXLLLLLLLLHLC5
CXHHXXLLLLLLLLHHC8
CXXXXXXLLLLLLLLLC0
CXLXXXLLHLLLLLLC1
CHXXXXLLHLHLLLLC2
CHXXXXLLHLLHLLLC3
CHXXXXLLLLLLLHLLC4
CHXXXXLLLLLLLLHLC5
CHXLHXLLLLLLLHHLC6
CHXXXLLLLLLLLHHLC6
CHXXXHLLLLLLLLHLC5
CHLXHXLLLLLLLHHLC6
CLXXXXXLLLLLLLLLC0
CXHXXXLHLHLLLLLLB1
CXLXXXLHLLLLLLLHB7
CXXXXXLHLLLLLLLLB0
CXXXXXLLHLLLLLLC1
CLHXXXLHLHLLLLLB1
CXHXXXLHLHHLLLLB2
CXLXXXLHLLLLLLLHB7
CXXXXXLHLLLLLLLLB0
CXXXXXLLHLLLLLLC1
CLHXXXLHLHLLLLLB1
CXHXXXLHLHHLLLLB2

```
CXHXXXLHLHLHLLLB3
CXLXXXLHLLLLLLHB7
CXXXXXLHLLLLLLLB0
CXXXXXLLHLLLLLLC1
CLHXXXLHLHLLLLLB1
CXHXXXLHLHHLLLLB2
CXHXXXLHLHLHLLLB3
CXHXXXLHLLLLHLLB4
CXLXXXLHLLLLLLHB7
CXXXXXLHLLLLLLLB0
CXXXXXLLHLLLLLLC1
CLHXXXLHLHLLLLLB1
CXHXXXLHLHHLLLLB2
CXHXXXLHLHLHLLLB3
CXHXXXLHLLLLHLLB4
CXHXXXLHLLLLLHLB5
CXXXLXLHLLLLLHLB5
CXLXHXLHLLLLLLLB0
CXXXXXLLHLLLLLLC1
CLHXXXLHLHLLLLLB1
CXHXXXLHLHHLLLLB2
CXHXXXLHLHLHLLLB3
CXHXXXLHLLLLHLLB4
CXHXXXLHLLLLLHLB5
CXHXHXLHLLLLHHLB6
CXHXXLLHLLLLHHLB6
CXHXXHLHLLLLLHLB5
CXHXHXLHLLLLHHLB6
CXLXXXLHLLLLLLLB0
CXXXXXLLHLLLLLLC1
CHHXXXHLLLLLLLLC0
------------------
DESCRIPTION

;BLA BLA
```

70

80

90


T0
CS + NA
CS·NA(WW+WB)
CS·NA·HW
CS·NA·R
T1
NA
check gen
OECA
ACK·NA·WB
T2
check gen
OEWD, WRITE
NA·WW
T11
check gen
OEWD OECA OERB
ACK·NA
T12
DTACK
CS
T3
check
T4
OERD
T5
OERD
T6
OERD
LATCH EDC, R
T7
VENT EDC
ERROR
T8
KORRIGER
T9
KORRIGER
T10
LATCH R
T11
OECA, OERB, OEWD
check gen
WRITE
T11
OECA
OERB
T12
DTACK
T1
READ
T4
OERD
T5
OERD
T6
OERD
LATCH EDC R
T7
VENT EDC
CTACK OK
ERROR
T8
KORRIGER
T9
KORRIGER
T10
LATCH R, C
T11
OERA, OERB, OEWD
check gen
T12
DTACK

palt

WW: Write Word; WB: Write Byte; MW: Modify Write; R: Read

```
t0 = T0*/CS + T0*NA + T1*NA + T11*NA + T7*R*/ERROR + T12*/CS
t1 = T0*/NA*CS*(WW+WB) + T0*/NA*CS*R + T1*/NA*(WB+R)*/ACK
t2 = T1*/NA*WW
t3 = T1*/NA*ACK*WB
t4 = T3 + T1*R*/NA*ACK
t5 = T4
t6 = T5
t7 = T6
t8 = T7*ERROR
t9 = T8
t10= T9
t11= T10 + T11*/NA*/ACK + T7*/ERROR*(WB+WW) + T0*CS*/NA*MW + T2
t12= T11*/NA*ACK + T12*CS + T7*R*/ERROR
```

	CODE			
	Y3	Y2	Y1	Y0
T0:	0	0	0	0
T1:	0	1	0	0
T2:	1	0	0	0
T3:	0	1	0	1
T4:	0	0	1	0
T5:	0	1	1	1
T6:	0	1	1	0
T7:	1	1	0	0
T8:	1	0	1	0
T9:	1	0	1	1
T10:	1	0	0	1
T11:	0	0	0	1
T12:	0	0	1	1

$$Y3 = t2 + t7 + t8 + t9 + t10$$

$$Y3 = T1*/NA*WW + T6 + T7*ERROR + (T8+T9)$$

$$Y2 = t1 + t3 + t5 + t6 + t7$$

$$Y2 = T0*/NA*CS*(WW+WB) + T0*/NA*CS*R + T1*/NA*(WB+R)*/ACK + T1*/NA*ACK*WB + T4 + (T4+T5)$$

$$Y1 = t4 + t5 + t6 + t8 + t9 + t12$$

$$Y1 = T1*R*/NA*ACK + (T3+T5) + (T4+T8) + T11*/NA*ACK + T7*ERROR + T12*CS + T7*R*/ERROR$$

$$Y0 = t3 + t5 + t9 + t10 + t11 + t12$$

$$Y0 = T0*CS*/NA*MW + T1*/NA*ACK*WB + (T2+T8+T9+T10) + T4 + T11*/NA + T7*/ERROR*(WB+WW) + T12*CS + T7*R*/ERROR$$

	START	WRITE	SS1	SS0	OE RD	OE WD	WRAB	OE RA	OE RB	WCAB	OE CA	OE CB	WSB	DTACK	DTDT1	TST1	
T0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
T1	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	W
T2	1	1	0	1	0	1	0	0	0	0	1	0	0	0	0	0	WW
T11	1	1	0	1	0	1	0	0	1	0	1	0	0	0	0	0	WW
T12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	WW
T3	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	WB
T4	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	WB
T5	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	WB
T6	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	WB
T7	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	WB
T8	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	WB
T9	0	0	1	1	0	0	1	0	0	1	0	0	1	0	0	0	WB
T10	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	WB
T11	1	1	0	1	0	1	0	0	1	0	1	0	0	0	0	0	WB
T12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	WB
T11	1	1	0	1	0	1	0	0	1	0	1	0	0	0	0	0	MW
T12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	MW
T1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	R
T4	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	R
T5	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	R
T6	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	R
T7	0	0	1	0	1	0	0	0	0	0	0	0	0	1	0	1	R
T8	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	R
T9	0	0	1	1	0	0	1	0	0	1	0	0	1	0	0	0	R
T10	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	R
T11	1	1	0	1	0	1	0	1	1	0	0	0	0	0	0	0	R
T12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	R

```
START = t1 + t2 + t3 + t11

START = T0*/NA*CS + T1*/NA*(WB+R)*/ACK + T1*/NA*WW
      + T1*/NA*ACK*WB + T11*/NA*/ACK + T7*/ERROR*(WB+WW)
      + (T2+T10)

WRITE = t2 + t11

WRITE = T1*/NA*WW + T11*/NA*/ACK + T7*/ERROR*(WB+WW)
      + T0*CS*/NA*MW + (T2+T10)

S1 = t6 + t7 + t8 + t9

S1 = T5 + T6 + T7*ERROR + T8

S0 = t1*(WB+WW) + t2 + t8 + t9 + t10 + t11

S0 = T0*/NA*CS*(WW+WB) + T0*CS*/NA*MW + T1*/NA*WB*/ACK
   + T1*/NA*WW + T7*ERROR + T7*/ERROR*(WB+WW)
   + T11*/NA*/ACK + (T2+T8+T9+T10)

OERD = t3 + t4 + t5 + t6 + t7

OERD = T1*/NA*ACK*(WB+R) + T3 + T4 + (T5+T6)

OEWD = t2 + t11

OEWD = T0*CS*/NA*MW + T1*/NA*WW + T7*/ERROR*(WB+WW)
     + T11*/NA*/ACK + (T2+T10)

WR = t5 + t9

WR = T4 + T8

OERA = t11*R

OERA = T10*R + T11*/NA*/ACK*R

OERB = t11
```

```
OERB = T10 + T11*/NA*/ACK + T7*/ERROR*(WB+WW)
     + T0*CS*/NA*MW + T2

WC = t5 + t9

WC = T4 + T8

OECA = t1*(WB+WW) + t2 + t11*(WB+WW+MW)

OECA = T0*/NA*CS*(WW+MW) + T1*/NA*WW + T1*/NA*/ACK*WB
     + (T2+T10)*(WB+WW) + (T2+T10)*MW + T7*/ERROR*(WB+WW)
     + T11*/NA*/ACK*(WB+WW) + T11*/ACK*/NA*MW

WSB = t9

WSB = T8

DTACKOK = t7*R

DTACKOK = T6*R

DT1 = t12

DT1 = T11*/NA*ACK + T12*CS + T7*R*/ERROR

TST1 = t7

TST1 = T6
```

```
PAL16R8
SEKVENSMASKINE SEKB TIL STYRING AF LAGER
CPAL30: CPU PAL NR. 3 VERSION 0. POSITION F7
KAN 821101
CPL   F1   F0   NA   ER  /ACK  NC      NC  /CS  GND
GND  /Y3  /Y2  /Y1  /Y0  /W   /START  /S0  /S1  VCC

Y3 :=  /Y3 *  Y2 * /Y1 * /Y0 *  F1 *  F0 * /NA
     + /Y3 *  Y2 *  Y1 * /Y0
     +  Y3 *  Y2 * /Y1 * /Y0 *                  ER
     +  Y3 * /Y2 *  Y1 * /Y0
     +  Y3 * /Y2 *  Y1 *  Y0

Y2:=   /Y3 * /Y2 * /Y1 * /Y0 *        F0 * /NA *                CS
     + /Y3 * /Y2 * /Y1 * /Y0 * /F1 * /F0 * /NA *                CS
     + /Y3 *  Y2 * /Y1 * /Y0 * /F1 *       /NA *        /ACK
     + /Y3 *  Y2 * /Y1 * /Y0 * /F1 *  F0 * /NA *         ACK
     + /Y3 * /Y2 *  Y1 * /Y0
     + /Y3 *  Y2 *  Y1 *  Y0
     + /Y3 *  Y2 *  Y1 * /Y0

Y1:=   /Y3 *  Y2 * /Y1 * /Y0 * /F1 * /F0 * /NA *         ACK
     + /Y3 *  Y2 *        Y0
     +       /Y2 *  Y1 * /Y0
     + /Y3 * /Y2 * /Y1 *  Y0 *             /NA *         ACK
     +  Y3 *  Y2 * /Y1 * /Y0 *                    ER
     + /Y3 * /Y2 *  Y1 *  Y0 *                                CS
     +  Y3 *  Y2 * /Y1 * /Y0 * /F1 * /F0 *        /ER


Y0:=   /Y3 * /Y2 * /Y1 * /Y0 *  F1 * /F0 * /NA *            CS
     + /Y3 *  Y2 * /Y1 * /Y0 * /F1 *  F0 * /NA *         ACK
     +  Y3 * /Y2
     + /Y3 * /Y2 *  Y1 * /Y0
     + /Y3 * /Y2 * /Y1 *  Y0 *             /NA
     +  Y3 *  Y2 * /Y1 * /Y0 *        F0 *        /ER
     +  Y3 *  Y2 * /Y1 * /Y0 * /F1 * /F0 *        /ER
     + /Y3 * /Y2 *  Y1 *  Y0 *                                CS
```

```
START:=/Y3 * /Y2 * /Y1 * /Y0 *                    /NA *                  CS
     + /Y3 *  Y2 * /Y1 * /Y0 * /F1 *              /NA *          /ACK
     + /Y3 *  Y2 * /Y1 * /Y0 *  F1 *  F0 * /NA
     + /Y3 *  Y2 * /Y1 * /Y0 * /F1 *  F0 * /NA *           ACK
     + /Y3 * /Y2 * /Y1 *  Y0 *                    /NA *          /ACK
     +  Y3 *  Y2 * /Y1 * /Y0 *           F0 *          /ER
     +  Y3 * /Y2 * /Y1

S1:=   /Y3 *  Y2 *  Y1 *  Y0
     + /Y3 *  Y2 *  Y1 * /Y0
     +  Y3 *  Y2 * /Y1 * /Y0 *                                  ER
     +  Y3 * /Y2 *  Y1 * /Y0

S0:=   /Y3 * /Y2 * /Y1 * /Y0 *           F0 * /NA *                     CS
     + /Y3 * /Y2 * /Y1 * /Y0 *  F1 * /F0 * /NA *                     CS
     + /Y3 *  Y2 * /Y1 * /Y0 * /F1 *  F0 * /NA *          /ACK
     + /Y3 *  Y2 * /Y1 * /Y0 *  F1 *  F0 * /NA
     +  Y3 *  Y2 * /Y1 * /Y0 *                              ER
     +  Y3 *  Y2 * /Y1 * /Y0 *           F0 *          /ER
     + /Y3 * /Y2 * /Y1 *  Y0 *                    /NA *          /ACK
     +  Y3 * /Y2

W :=   /Y3 *  Y2 * /Y1 * /Y0 *  F1 *  F0 * /NA
     +  Y3 * /Y2 * /Y1 *  Y0
     + /Y3 * /Y2 * /Y1 *  Y0 *                    /NA *          /ACK
     +  Y3 *  Y2 * /Y1 * /Y0 *           F0 *          /ER
     + /Y3 * /Y2 * /Y1 * /Y0 *  F1 * /F0 * /NA *                     CS
     +  Y3 * /Y2 * /Y1 * /Y0
```

FUNCTION TABLE

```
CPL CS F1 F0 NA ACK ER Y3 Y2 Y1 Y0 START W S1 S0
-----------------------------------------------
CLXXHHL XXXX XXXX
CLXXHHL XXXX XXXX
CLXXHHL XXXX XXXX
CLXXHHL XXXX XXXX
CLXXHHL XXXX XXXX
CLXXHHL XXXX XXXX
```

```
CLXXHHL XXXX XXXX
CLXXHHL XXXX XXXX
CLXXHHL XXXX XXXX
CLXXHHL XXXX XXXX       10
CLXXHHL XXXX XXXX
CLXXHHL XXXX XXXX
CLXXHHL LLLL LLLL T0
CLXXXXX LLLL LLLL T0
CXXXHXX LLLL LLLL T0
CHXHLXX LHLL HLLH T1W
CXLHLLX LHLL HLLH T1W
CXXXHXX LLLL LLLL T0
CHXHLXX LHLL HLLH T1W
CXHHLXX HLLL HHLH T2W   20
CXXXXXX LLLH HHLH T11W
CXXXLLX LLLH HHLH T11W
CXXXHXX LLLL LLLL T0
CHXHLXX LHLL HLLH T1W
CXHHLXX HLLL HHLH T2W
CXXXXXX LLLH HHLH T11W
CXXXLHX LLHH LLLL T12W
CHXXXXX LLHH LLLL T12W
CLXXXXX LLLL LLLL T0
CHXHLXX LHLL HLLH T1W   30
CXLHLHX LHLH HLLL T3B
CXXXXXX LLHL LLLL T4B
CXXXXXX LHHH LLLL T5B
CXXXXXX LHHL LLHL T6B
CXXXXXX HHLL LLHL T7B
CXXXXXH HLHL LLHH T8B
CXXXXXX HLHH LLHH T9B
CXXXXXX HLLH LLLH T10B
CXXXXXX LLLH HHLH T11B
CXXXLLX LLLH HHLH T11B  40
CXXXHXX LLLL LLLL T0
CHXHLXX LHLL HLLH T1W
CXLHLHX LHLH HLLL T3B
CXXXXXX LLHL LLLL T4B
CXXXXXX LHHH LLLL T5B
```

```
CXXXXXX LHHL LLHL T6B
CXXXXXX HHLL LLHL T7B
CXXHXXL LLLH HHLH T11B
CXXXLHX LLHH LLLL T12B
CHXXXXX LLHH LLLL T12B  50
CLXXXXX LLLL LLLL T0
CHLLLXX LHLL HLLL T1R
CXLLLLX LHLL HLLL T1R
CXXXHXX LLLL LLLL T0
CHLLLXX LHLL HLLL T1R
CXLLLHX LLHL LLLL T4R
CXXXXXX LHHH LLLL T5R
CXXXXXX LHHL LLHL T6R
CXXXXXX HHLL LLHL T7R
CXXXXXH HLHL LLHH T8R   60
CXXXXXX HLHH LXHH T9R
CXXXXXX HLLH LLLH T10R
CXXXXXX LLLH HHLH T11R
CXXXLLX LLLH HHLH T11R
CXXXHXX LLLL LLLL T0
CHLLLXX LHLL HLLL T1R
CXLLLHX LLHL LLLL T4R
CXXXXXX LHHH LLLL T5R
CXXXXXX LHHL LLHL T6R
CXXXXXX HHLL LLHL T7R                70
CXXXXXH HLHL LLHH T8R
CXXXXXX HLHH LLHH T9R
CXXXXXX HLLH LLLH T10R
CXXXXXX LLLH HHLH T11R
CXXXLHX LLHH LLLL T12R
CHXXXXX LLHH LLLL T12R
CLXXXXX LLLL LLLL T0
CHLLLXX LHLL HLLL T1R
CXLLLHX LLHL LLLL T4R
CXXXXXX LHHH LLLL T5R                  80
CXXXXXX LHHL LLHL T6R
CXXXXXX HHLL LLHL T7R
CXLLXXL LLHH LLLL T12R
CLXXXXX LLLL LLLL T0
```

```
CHHLLXX LLLH HHLH T11M
CXXXLLX LLLH HHLH T11M
CXXXHXX LLLL LLLL T0
CHHLLXX LLLH HHLH T11M
CXXXLHX LLHH LLLL T12M
CHXXXXX LLHH LLLL T12M                    90
CLXXXXX LLLL LLLL T0
-----------------------------
DESCRIPTION
;                F1  F0
;
;READ            0   0
;WRITE BYTE      0   1
;WRITE WORD      1   1
;MODIFY WRITE    1   0
```

```
PAL16R6
SEKVENSMASKINE TIL STYRING AF LAGER
CPAL41: CPU PAL NR. 4 VERSION 1. POSITION F6
KAN 830503
CPL   F1  F0   NA    ER  /ACK  /Y3   /Y2  /Y1  GND
GND  /Y0 /DT1 /DTOK /WSB /OECB /OECA /WC  /CS  VCC

WC :=      /Y3 * /Y2 *  Y1 * /Y0
         +  Y3 * /Y2 *  Y1 * /Y0

OECA:=     /Y3 *  Y2 * /Y1 * /Y0 *  F1 *  F0 * /NA              ;T1*WW
         + /Y3 *  Y2 * /Y1 * /Y0 * /F1 *  F0 * /NA */ACK       ;T1*WB
         +  Y3 * /Y2 * /Y1 *               F0  ;(T2+T10)*(WB+WW)
         +  Y3 * /Y2 * /Y1 *         F1 * /F0  ;(T2+T10)*MW
         + /Y3 * /Y2 * /Y1 *  Y0 *         F0 * /NA * /ACK ;T11*(WB+WW)
         + /Y3 * /Y2 * /Y1 *  Y0 *  F1 * /F0 * /NA *       /ACK ;T11*MW
         +  Y3 *  Y2 * /Y1 * /Y0 *         F0 *       /ER       ;T7*(WB+WW
         + /Y3 * /Y2 * /Y1 * /Y0 *  F1 *       /NA *       CS   ;T0*(WW+MW

;OECB:= GND  UDGANG SKAL VÆRE KONSTANT HØJ

WSB:=       Y3 * /Y2 *  Y1 * /Y0

DTOK:=     /Y3 *  Y2 *  Y1 * /Y0 * /F1 * /F0


DT1:=       Y3 *  Y2 * /Y1 * /Y0 * /F1 * /F0 *       /ER
         + /Y3 * /Y2 * /Y1 *  Y0 *             /NA *       ACK
         + /Y3 * /Y2 *  Y1 *  Y0 *                                CS

FUNCTION TABLE
CPL CS F1 F0 NA ACK ER Y3 Y2 Y1 Y0 DT1 DTOK WC OECB OECA WSB
-----------------------------------------------------------
CLXXXXX LLLL LLLHLL T0   -T0
CXXXHXX LLLL LLLHLL T0   -T0
CHLHLXX LLLL LLLHLL T0   -T1WB....
CHHHLXX LLLL LLLHHL T0   -T1WW....
CHHLLXX LLLL LLLHHL T0   -T11M
CHLLLXX LLLL LLLHLL T0   -T1R
```

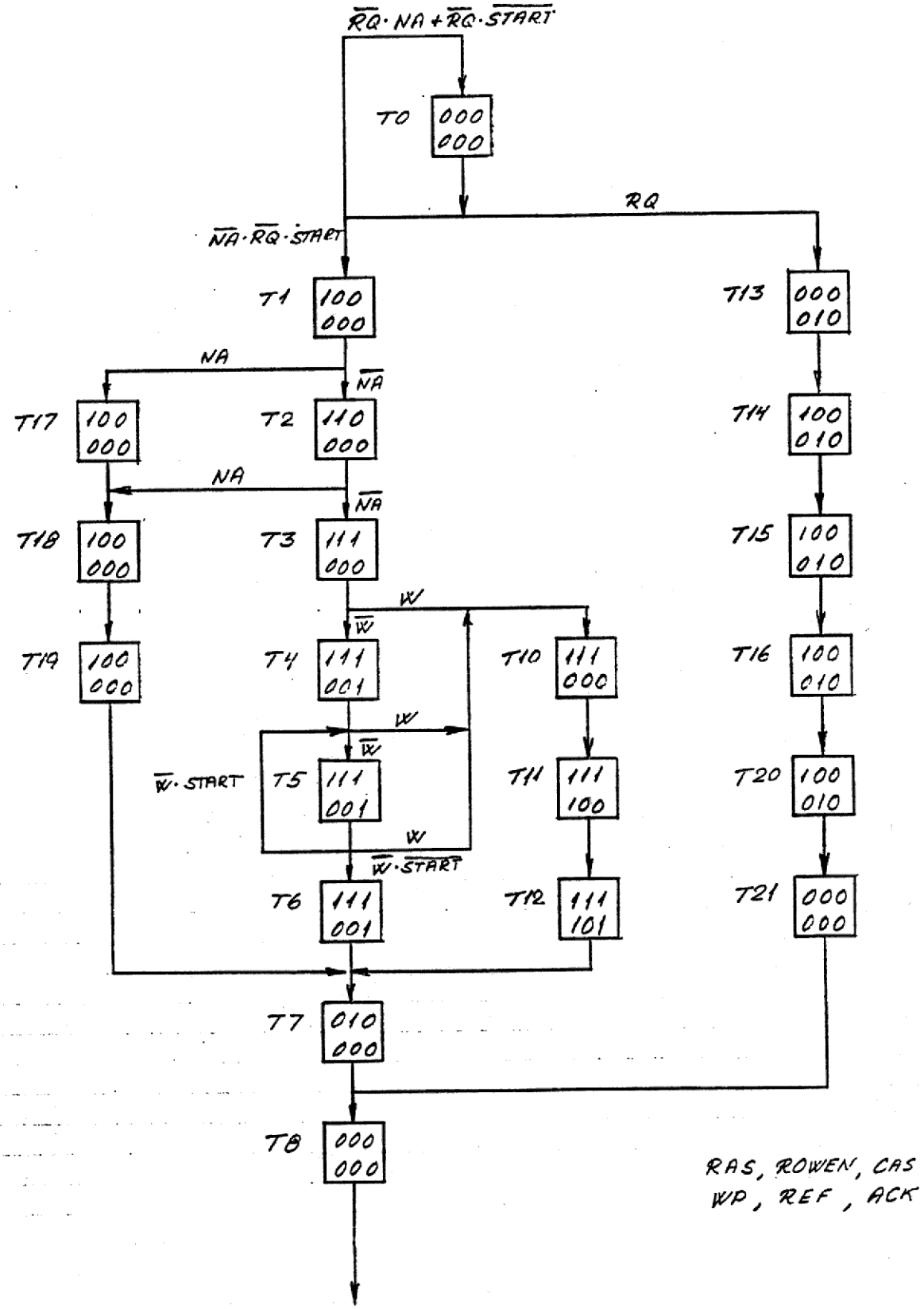
```
CXXXHXX LHLL LLLHLL T1W  -T0
CXLHLLX LHLL LLLHHL T1W  -T1W....
CXHHLXX LHLL LLLHHL T1W  -T2W
CXLHLHX LHLL LLLHLL T1W  -T3B
CXHHXXX HLLL LLLHHL T2W  -T11W
CXHHHXX LLLH LLLHLL T11W -T0
CXHHLLX LLLH LLLHHL T11W -T11W
CXHHLHX LLLH HLLHLL T11W -T12W
CLHHXXX LLHH LLLHLL T12W -T0
CHHHXXX LLHH HLLHLL T12W -T12W
CXXHXXX LHLH LLLHLL T3B  -T4B
CXXHXXX LLHL LLHHLL T4B  -T5B
CXXHXXX LHHH LLLHLL T5B  -T6B
CXXHXXX LHHL LLLHLL T6B  -T7B
CXXHXXL HHLL LLLHHL T7B  -T11B
CXXHXXH HHLL LLLHLL T7B  -T8B
CXXHXXX HLHL LLHHLH T8B  -T9B
CXXHXXX HLHH LLLHLL T9B  -T10B
CXXHXXX HLLH LLLHHL T10B -T11B
CXXHHXX LLLH LLLHLL T11B -T0
CXXHLLX LLLH LLLHHL T11B -T11B
CXXHLHX LLLH HLLHLL T11B -T12B
CLXHXXX LLHH LLLHLL T12B -T0
CHXHXXX LLHH HLLHLL T12B -T12B
CXHLHXX LLLH LLLHLL T11M -T0
CXHLLLX LLLH LLLHHL T11M -T11M
CXHLLHX LLLH HLLHLL T11M -T12M
CLHLXXX LLHH LLLHLL T12M -T0
CHHLXXX LLHH HLLHLL T12M -T12M
CXLLHXX LHLL LLLHLL T1R  -T0
CXLLLLX LHLL LLLHLL T1R  -T1R
CXLLLHX LHLL LLLHLL T1R  -T4R
CXLLXXX LLHL LLHHLL T4R  -T5R
CXLLXXX LHHH LLLHLL T5R  -T6R
CXLLXXX LHHL LHLHLL T6R  -T7R
CXLLXXH HHLL LLLHLL T7R  -T8R
CXLLXXL HHLL HLLHLL T7R  -T12R
CXLLXXX HLHL LLHHLH T8R  -T9R
CXLLXXX HLHH LLLHLL T9R  -T10R
```

```
CXLLXXX HLLH LLLHLL T10R -T11R
CXLLHXX LLLH LLLHLL T11R -T0
CXLLLLX LLLH LLLHLL T11R -T11R
CXLLLHX LLLH HLLHLL T11R -T12R
CLLLXXX LLHH LLLHLL T12R -T0
CHLLXXX LLHH HLLHLL T12R -T12R
--------------------------------

DESCRIPTION
; BLA  BLA
```

```
PAL16R6
SEKVENSMASKINE TIL STYRING AF LAGER
CPAL50: CPU PAL NR. 5 VERSION 0. POSITION F5
KAN 821101
CPL   F1  F0    NA    ER   /ACK  /Y3   /Y2    /Y1  GND
GND  /Y0 /TST1 /OERB /OERA /WR   /OEWD /OERD  /CS  VCC

OERD:=     /Y3 *  Y2 * /Y1 * /Y0 * /F1 *       /NA *         ACK
          +/Y3 *  Y2 * /Y1 *  Y0
          +/Y3 * /Y2 *  Y1 * /Y0
          +/Y3 *  Y2 *  Y1 *  Y0
          +/Y3 *  Y2 *  Y1 * /Y0

OEWD:=     /Y3 * /Y2 * /Y1 * /Y0 *  F1 * /F0 * /NA *                CS
          +/Y3 *  Y2 * /Y1 * /Y0 *  F1 *  F0 * /NA
          + Y3 *  Y2 * /Y1 * /Y0 *        F0 *       /ER
          +/Y3 * /Y2 * /Y1 *  Y0 *             /NA *        /ACK
          + Y3 * /Y2 * /Y1 * /Y0
          + Y3 * /Y2 * /Y1 *  Y0

WR:=       /Y3 * /Y2 *  Y1 * /Y0
          + Y3 * /Y2 *  Y1 * /Y0

OERA:=       Y3 * /Y2 * /Y1 *  Y0 * /F1 * /F0
          +/Y3 * /Y2 * /Y1 *  Y0 * /F1 * /F0 * /NA *        /ACK

OERB:=       Y3 * /Y2 * /Y1 *  Y0
          +/Y3 * /Y2 * /Y1 *  Y0 *             /NA        * /ACK
          + Y3 *  Y2 * /Y1 * /Y0 *        F0 *       /ER
          +/Y3 * /Y2 * /Y1 * /Y0 *  F1 * /F0 * /NA *                CS
          + Y3 * /Y2 * /Y1 * /Y0

TST1:=     /Y3 *  Y2 *  Y1 * /Y0

FUNCTION TABLE
CPL CS F1 F0 NA ACK ER Y3 Y2 Y1 Y0 TST1 OERB OERA WR OEWD OERD
-------------------------------------------------------------
CLXXXXX LLLL LLLLLL T0   -T0
CXXXHXX LLLL LLLLLL T0   -T0
```

```
CHXHLXX LLLL LLLLLL T0   -T1W
CHHLLXX LLLL LHLLHL T0   -T11M
CHLLLXX LLLL LLLLLL T0   -T1R
CXXXHXX LHLL LLLLLL T1W  -T0
CXLHLLX LHLL LLLLLL T1W  -T1W
CXHHLXX LHLL LLLLHL T1W  -T2W
CXLHLHX LHLL LLLLLH T1W  -T3B
CXHHXXX HLLL LHLLHL T2W  -T11W
CXHHHXX LLLH LLLLLL T11W -T0
CXHHLLX LLLH LHLLHL T11W -T11W
CXHHLHX LLLH LLLLLL T11W -T12W
CLHHXXX LLHH LLLLLL T12W -T0
CHHHXXX LLHH LLLLLL T12W -T12W
CXXHXXX LHLH LLLLLH T3B  -T4B
CXXHXXX LLHL LLLHLH T4B  -T5B
CXXHXXX LHHH LLLLLH T5B  -T6B
CXXHXXX LHHL HLLLLH T6B  -T7B
CXXHXXL HHLL LHLLHL T7B  -T11B
CXXHXXH HHLL LLLLLL T7B  -T8B
CXXHXXX HLHL LLLHLL T8B  -T9B
CXXHXXX HLHH LLLLLL T9B  -T10B
CXXHXXX HLLH LHLLHL T10B -T11B
CXXHHXX LLLH LLLLLL T11B -T0
CXXHLLX LLLH LHLLHL T11B -T11B
CXXHLHX LLLH LLLLLL T11B -T12B
CLXHXXX LLHH LLLLLL T12B -T0
CHXHXXX LLHH LLLLLL T12B -T12B
CXHLHXX LLLH LLLLLL T11M -T0
CXHLLLX LLLH LHLLHL T11M -T11M
CXHLLHX LLLH LLLLLL T11M -T12M
CLHLXXX LLHH LLLLLL T12M -T0
CHHLXXX LLHH LLLLLL T12M -T12M
CXLLHXX LHLL LLLLLL T1R  -T0
CXLLLLX LHLL LLLLLL T1R  -T1R
CXLLLHX LHLL LLLLLH T1R  -T4R
CXLLXXX LLHL LLLHLH T4R  -T5R
CXLLXXX LHHH LLLLLH T5R  -T6R
CXLLXXX LHHL HLLLLH T6R  -T7R
CXLLXXH HHLL LLLLLL T7R  -T8R
```

```
CXLLXXL HHLL LLLLLL T7R  -T12R
CXLLXXX HLHL LLLHLL T8R  -T9R
CXLLXXX HLHH LLLLLL T9R  -T10R
CXLLXXX HLLH LHHLHL T10R -T11R
CXLLHXX LLLH LLLLLL T11R -T0
CXLLLLX LLLH LHHLHL T11R -T11R
CXLLLHX LLLH LLLLLL T11R -T12R
CLLLXXX LLHH LLLLLL T12R -T0
CHLLXXX LLHH LLLLLL T12R -T12R
--------------------------------

DESCRIPTION
; BLA BLA
```


RQ·NA + RQ·START
T0
000
000
RQ
NA·RQ·START
T1
100
000
T13
000
010
NA
NA
T17
100
000
T2
110
000
T14
100
010
NA
NA
T18
100
000
T3
111
000
T15
100
010
W
W
T19
100
000
T4
111
001
T10
111
000
T16
100
010
W
W
W·START
T5
111
001
T11
111
100
T20
100
010
W
W·START
T6
111
001
T12
111
101
T21
000
000
T7
010
000
T8
000
000
RAS, ROWEN, CAS
WP, REF, ACK

```
t0 = T0*/RQ*NA + T0*/RQ*/START + T8                                    T60

t1 = T0*/NA*/RQ*START

t2 = T1*/NA

t3 = T2*/NA

t4 = T3*/W

t5 = T4*/W + T5*/W*START

t6 = T5*/W*/START

t7 = T6 + T19 + T12

t8 = T7 + T21

t10 = T3*W + T4*W + T5*W

t11 = T10

t12 = T11

t13 = T0*RQ

t14 = T13

t15 = T14

t16 = T15

t17 = T1*NA

t18 = T17 + T2*NA

t19 = T18

t20 = T16          ; t21 = T20
```

	R O R A S	W E N	C A S	R W P	A E F	C K	N 1	N 2
T0:	0	0	0	0	0	0	0	0
T1:	1	0	0	0	0	0	0	0
T2:	1	1	0	0	0	0	0	0
T3:	1	1	1	0	0	0	0	0
T4:	1	1	1	0	0	1	0	0
T5:	1	1	1	0	0	1	0	1
T6:	1	1	1	0	0	1	1	1
T7:	0	1	0	0	0	0	0	0
T8:	0	0	0	0	0	0	1	1
T10:	1	1	1	0	0	0	0	1
T11:	1	1	1	1	0	0	0	1
T12:	1	1	1	1	0	1	0	0
T13:	0	0	0	0	1	0	0	0
T14:	1	0	0	0	1	0	0	0
T15:	1	0	0	0	1	0	0	1
T16:	1	0	0	0	1	0	1	1
T17:	1	0	0	0	0	0	0	1
T18:	1	0	0	0	0	0	1	1
T19:	1	0	0	0	0	0	1	0
T20:	1	0	0	0	1	0	1	0
T21:	0	0	0	0	0	0	1	0

```
RAS = t1 + t2 + t3 + t4 + t5 + t6 + t10 + t11 + t12
    + t14 + t15 + t16 + t17 + t18 + t19 + t20

RAS = T0*/NA*/RQ*START + (T1+T2) + (T3+T4) + T5
    + (T15+T16+T17+T18) + (T10+T11) + (T13+T14)

ROWEN = t2 + t3 + t4 + t5 + t6 + t7 + t10 + t11 + t12

ROWEN = (T1+T2)*/NA + (T3+T4) + (T5+T6) +(T10+T11) + T12 + T19

CAS = t3 + t4 + t5 + t6 + t10 + t11 + t12

CAS = T2*/NA + (T3+T4+T5+T10) + T11

WP = t11 + t12

WP = T10 + T11

REF = t13 + t14 + t15 + t16 + t20

REF = T0*RQ + (T13+T14) + (T15+T16)

ACK = t4 + t5 + t6 + t12

ACK = T11 + (T3+T4)*/W + T5*/W

N1 = t6 + t8 + t16 + t18 + t19 + t20 + t21

N1 = T2*NA + T5*/W*/START + T7 + (T15+T16+T17+T18) + T20 + T21

N2 = t5 + t6 + t8 + t10 + t11 + t15 + t16 + t17 + t18

N2 = (T1+T2)*NA + T3*W + (T4+T5) + T7 + T10 + (T14+T15)
   + T17 + T21
```

```
PAL16R8
SEKVENSMASKINE TIL GENERERING AF RAS CAS MV
CPAL60: CPU PAL NR. 6 VERSION 0. POSITION F8
KAN 821101
CPL  /ST  /WR   NA    RQ    NC   NC    NC    NC   GND
GND  /N1  /N2  /ACK  /REF  /WP  /CAS  /ROW  /RAS  VCC

RAS:=  /RAS */ROW */CAS */WP */REF */ACK */N1 */N2 * ST *     /NA */RQ
     + RAS *      /CAS */WP */REF */ACK */N1 */N2
     + RAS * ROW * CAS */WP */REF *      /N1 */N2
     + RAS */ROW */CAS */WP *      /ACK *      N2
     + RAS * ROW * CAS *     /REF */ACK */N1 * N2
     +      /ROW */CAS */WP * REF */ACK */N1 */N2
     + RAS * ROW * CAS */WP */REF * ACK */N1 * N2

ROW:=   RAS *      /CAS */WP */REF */ACK */N1 */N2 *          /NA
     + RAS * ROW * CAS */WP */REF *      /N1 */N2
     + RAS * ROW * CAS */WP */REF * ACK *      N2
     + RAS * ROW * CAS *     /REF */ACK */N1 * N2
     + RAS * ROW * CAS * WP */REF * ACK */N1 */N2
     + RAS */ROW */CAS */WP */REF */ACK * N1 */N2

CAS:=   RAS * ROW */CAS */WP */REF */ACK */N1 */N2 *          /NA
     + RAS * ROW * CAS */WP */REF *      /N1
     + RAS * ROW * CAS * WP */REF */ACK */N1 * N2

WP:=    RAS * ROW * CAS */WP */REF */ACK */N1 * N2
     + RAS * ROW * CAS * WP */REF */ACK */N1 * N2

REF:=  /RAS */ROW */CAS */WP */REF */ACK */N1 */N2 *                RQ
     +/RAS */ROW */CAS */WP * REF */ACK */N1 */N2
     + RAS */ROW */CAS */WP * REF */ACK */N1 */N2
     + RAS */ROW */CAS */WP * REF */ACK */N1 * N2
     + RAS */ROW */CAS */WP * REF */ACK * N1 * N2

ACK:=   RAS * ROW * CAS */WP */REF */ACK */N1 */N2 *     /WR
     + RAS * ROW * CAS */WP */REF * ACK */N1 */N2 *     /WR
     + RAS * ROW * CAS */WP */REF * ACK */N1 * N2 *     /WR
     + RAS * ROW * CAS * WP */REF */ACK */N1 * N2
```

```
N1:=    RAS * ROW */CAS */WP */REF */ACK */N1 */N2 *            NA
      + RAS * ROW * CAS */WP */REF * ACK */N1 * N2 */ST */WR
      +/RAS * ROW */CAS */WP */REF */ACK */N1 */N2
      + RAS */ROW */CAS */WP *       /ACK *      N2
      + RAS */ROW */CAS */WP * REF */ACK * N1 */N2
      +/RAS */ROW */CAS */WP */REF */ACK * N1 */N2

N2:=    RAS *       /CAS */WP */REF */ACK */N1 */N2 *            NA
      + RAS * ROW * CAS */WP */REF */ACK */N1 */N2 *       WR
      + RAS * ROW * CAS */WP */REF * ACK */N1
      +/RAS * ROW */CAS */WP */REF */ACK */N1 */N2
      + RAS * ROW * CAS */WP */REF */ACK */N1 * N2
      + RAS */ROW */CAS */WP * REF */ACK */N1
      + RAS */ROW */CAS */WP */REF */ACK */N1 * N2
      +/RAS */ROW */CAS */WP */REF */ACK * N1 */N2
```

FUNCTION TABLE:

```
CPL ST WR NA RQ RAS ROW CAS WP REF ACK N1 N2
-------------------------------------------
CLXHL XXXXXXXX ??
CLXHL XXXXXXXX ??
CLXHL XXXXXXXX ??
CLXHL XXXXXXXX ??
CLXHL XXXXXXXX ??
CLXHL XXXXXXXX ??
CLXHL XXXXXXXX ??
CLXHL XXXXXXXX ??
CLXHL XXXXXXXX ??
CLXHL XXXXXXXX ??
CLXHL XXXXXXXX ??
CLXHL XXXXXXXX ??
CLXHL XXXXXXXX ??
CLXHL XXXXXXXX ??
CLXHL XXXXXXXX ??
CLXXL LLLLLLLL TO
CXXHL LLLLLLLL TO
CHXLL HLLLLLLL T1
```

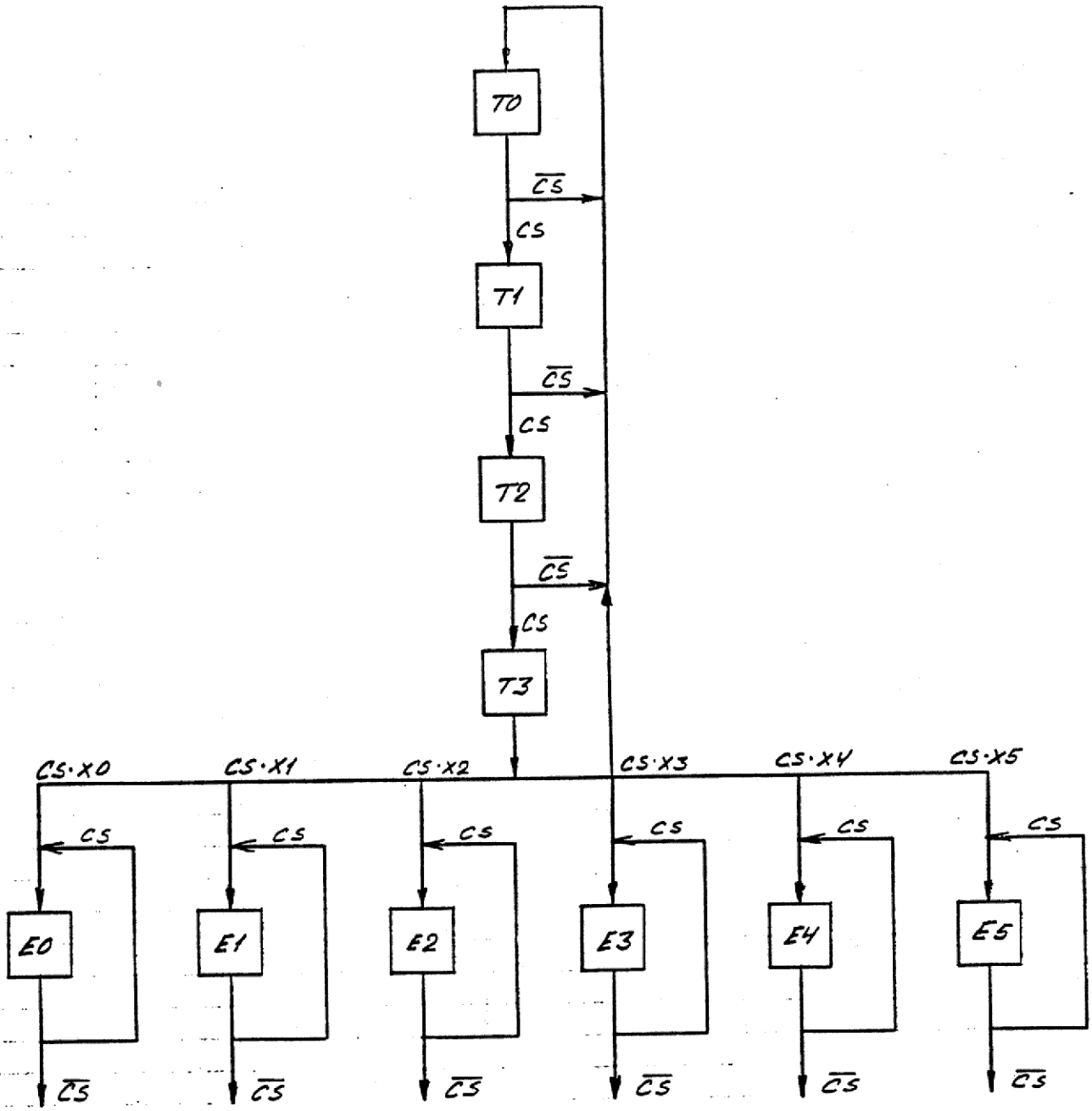
```
CXXHX HLLLLLLH T17
CXXXX HLLLLLHH T18
CXXXX HLLLLLHL T19
CXXXX LHLLLLLL T7
CXXXX LLLLLLHH T8
CXXXX LLLLLLLL T0
CHXLL HLLLLLLL T1
CXXLX HHLLLLLL T2
CXXHX HLLLLLHH T18
CXXXX HLLLLLHL T19
CXXXX LHLLLLLL T7
CXXXX LLLLLLHH T8
CXXXX LLLLLLLL T0
CHXLL HLLLLLLL T1
CXXLX HHLLLLLL T2
CXXLX HHHLLLLL T3
CXHXX HHHLLLLH T10
CXXXX HHHHLLLH T11
CXXXX HHHHLHLL T12
CXXXX LHLLLLLL T7
CXXXX LLLLLLHH T8
CXXXX LLLLLLLL T0
CHXLL HLLLLLLL T1
CXXLX HHLLLLLL T2
CXXLX HHHLLLLL T3
CXLXX HHHLLHLL T4
CXHXX HHHLLLLH T10
CXXXX HHHHLLLH T11
CXXXX HHHHLHLL T12
CXXXX LHLLLLLL T7
CXXXX LLLLLLHH T8
CXXXX LLLLLLLL T0
CHXLL HLLLLLLL T1
CXXLX HHLLLLLL T2
CXXLX HHHLLLLL T3
CXLXX HHHLLHLL T4
CXLXX HHHLLHLH T5
CHLXX HHHLLHLH T5
CXHXX HHHLLLLH T10
```

```
CXXXX HHHHLLLH T11
CXXXX HHHHLHLL T12
CXXXX LHLLLLLL T7
CXXXX LLLLLLHH T8
CXXXX LLLLLLLL T0
CHXLL HLLLLLLL T1
CXXLX HHLLLLLL T2
CXXLX HHHLLLLL T3
CXLXX HHHLLHLL T4
CXLXX HHHLLHLH T5
CLLXX HHHLLHHH T6
CXXXX LHLLLLLL T7
CXXXX LLLLLLHH T8
CXXXX LLLLLLLL T0
CXXXH LLLLHLLL T13
CXXXX HLLLHLLL T14
CXXXX HLLLHLLH T15
CXXXX HLLLHLHH T16
CXXXX HLLLHLHL T20
CXXXX LLLLLLHL T21
CXXXX LLLLLLHH T8
CXXXX LLLLLLLL T0
------------------
DESCRIPTION

;BLA BLA
```

```
PAL16L8
DEKODNING AF ADRESSE TIL YDRE ENHEDER
CPAL70: CPU PAL NR 7 VERSION 0. POSITION M0
KAN 821122
FC2  FC1  FC0  A23  A22  A21  A20  A19  A18  GND
CS   /I1  CON  COM7 COM5 A15  A16  A17  /BS  VCC

IF (VCC) I1 =
    /CON * CS */A23 */A22 */A21 * A20 */A19 */A18 */A17 */A16 */A15
    *  FC2 * /FC1 * FC0                    ;SUPD*SEG1
  + /CON * CS */A23 */A22 */A21 * A20 */A19 */A18 */A17 */A16 */A15
    * /FC2 * /FC1 * FC0 * /COM5            ;USD*SEG1
  +  CON * CS */A23 */A22 */A21 * A20 */A19 */A18 */A17 */A16 */A15
    * COM7                                 ;BUS*SEG1

IF (VCC) BS =
    /CON * CS */A23 */A22 */A21 * A20 * A19 * A18 * FC2 */FC1 * FC0

  + /CON * CS */A23 */A22 */A21 * A20 * A19 * A18 */FC2 */FC1 * FC0
    * /COM5

FUNCTION TABLE
CON CS A23 A22 A21 A20 A19 A18 A17 A16 A15 FC2 FC1 FC0 COM7 COM5 I1 BS
-------------------------------------------------------------------
LHLLLHLLLLLHLHLHHX
HHLLLHLLLLLHLHLHLX
LLLLLHLLLLLHLHLHLX
LHHLLHLLLLLHLHLHLX
LHLHLHLLLLLHLHLHLX
LHLLHHLLLLLHLHLHLX
LHLLLLLLLLLHLHLHLX
LHLLLHHLLLLHLHLHLX
LHLLLHLHLLLHLHLHLX
LHLLLHLLHLLHLHLHLX 10
LHLLLHLLLHLHLHLHLX
LHLLLHLLLLHHLHLHLX
LHLLLHLLLLLLLHLHLX
LHLLLHLLLLLHHHLHLX
LHLLLHLLLLLHLLLHLX
```

LHLLLHLLLLLHLHHHHX
LHLLLHLLLLLHLHLLHX
LHLLLHLLLLLLLHLLHX
HHLLLHLLLLLLLHLLLX
LLLLLHLLLLLLLHLLLX 20
LHHLLHLLLLLLLHLLLX
LHLHLHLLLLLLLHLLLX
LHLLHHLLLLLLLHLLLX
LHLLLLLLLLLLLHLLLX
LHLLLHHLLLLLLHLLLX
LHLLLHLHLLLLLHLLLX
LHLLLHLLHLLLLHLLLX
LHLLLHLLLHLLLHLLLX
LHLLLHLLLLHLLHLLLX
LHLLLHLLLLLHLHLLHX 30
LHLLLHLLLLLLHHLLLX
LHLLLHLLLLLLLLLLLX
LHLLLHLLLLLLLHHLHX
LHLLLHLLLLLLLHLHLX
HHLLLHLLLLLXXLHHHX
LHLLLHLLLLLXXLHHLX
HLLLLHLLLLLXXLHHLX
HHHLLHLLLLLXXLHHLX
HHLHLHLLLLLXXLHHLX
HHLLHHLLLLLXXLHHLX 40
HHLLLLLLLLLXXLHHLX
HHLLLHHLLLLXXLHHLX
HHLLLHLHLLLXXLHHLX
HHLLLHLLHLLXXLHHLX
HHLLLHLLLLHXXLHHLX
HHLLLHLLLLLXXLLHLX
LHLLLHHHXXXHLHXHXH
HHLLLHHHXXXHLHXHXL
LLLLLHHHXXXHLHXHXL 50
LHHLLHHHXXXHLHXHXL
LHLHLHHHXXXHLHXHXL
LHLLHHHHXXXHLHXHXL
LHLLLLHHXXXHLHXHXL

```
LHLLLHLHXXXHLHXHXL
LHLLLHHLXXXHLHXHXL
LHLLLHHHXXXLLHXHXL
LHLLLHHHXXXHHHXHXL
LHLLLHHHXXXHLLXHXL
LHLLLHHHXXXHLHXLXH          60
LHLLLHHHXXXLLHXLXH
HHLLLHHHXXXLLHXLXL
LLLLLHHHXXXLLHXLXL
LHHLLHHHXXXLLHXLXL
LHLHLHHHXXXLLHXLXL
LHLLHHHHXXXLLHXLXL
LHLLLLHHXXXLLHXLXL
LHLLLHLHXXXLLHXLXL
LHLLLHHLXXXLLHXLXL
LHLLLHHHXXXHLHXLXH
LHLLLHHHXXXLHHXLXL
LHLLLHHHXXXLLLXLXL
-------------------------
DESCRIPTION
;BLA BLA
```


T0
T1
T2
T3
CS
CS·X0
CS·X1
CS·X2
CS·X3
CS·X4
CS·X5
E0
E1
E2
E3
E4
E5

t80

```
t1 = T0*CS

t2 = T1*CS

t3 = T2*CS

e0 = T3*CS*X0 + E0*CS    ;X0 = /I1*/A12

e1 = T3*CS*X1 + E1*CS    ;X1 = /I1*/I2*A12*/A5*/A4*/A3

e2 = T3*CS*X2 + E2*CS    ;X2 = /I1*/I2*A12*/A5*/A4*A3

e3 = T3*CS*X3 + E3*CS    ;X3 = /I1*/I2*A12*/A5*A4*/A3

e4 = T3*CS*X4 + E4*CS    ;X4 = /I1*/I2*A12*/A5*A4*A3

e5 = T3*CS*X5 + E5*CS    ;X5 = /I1*/I2*A12*A5*/A4
```

CODE

	Y1	Y0	E5	E4	E3	E2	E1	E0
T0:	0	0	0	0	0	0	0	0
T1:	0	1	0	0	0	0	0	0
T2:	1	1	0	0	0	0	0	0
T3:	1	0	0	0	0	0	0	0
E0:	0	0	0	0	0	0	0	1
E1:	0	0	0	0	0	0	1	0
E2:	0	0	0	0	0	1	0	0
E3:	0	0	0	0	1	0	0	0
E4:	0	0	0	1	0	0	0	0
E5:	0	0	1	0	0	0	0	0

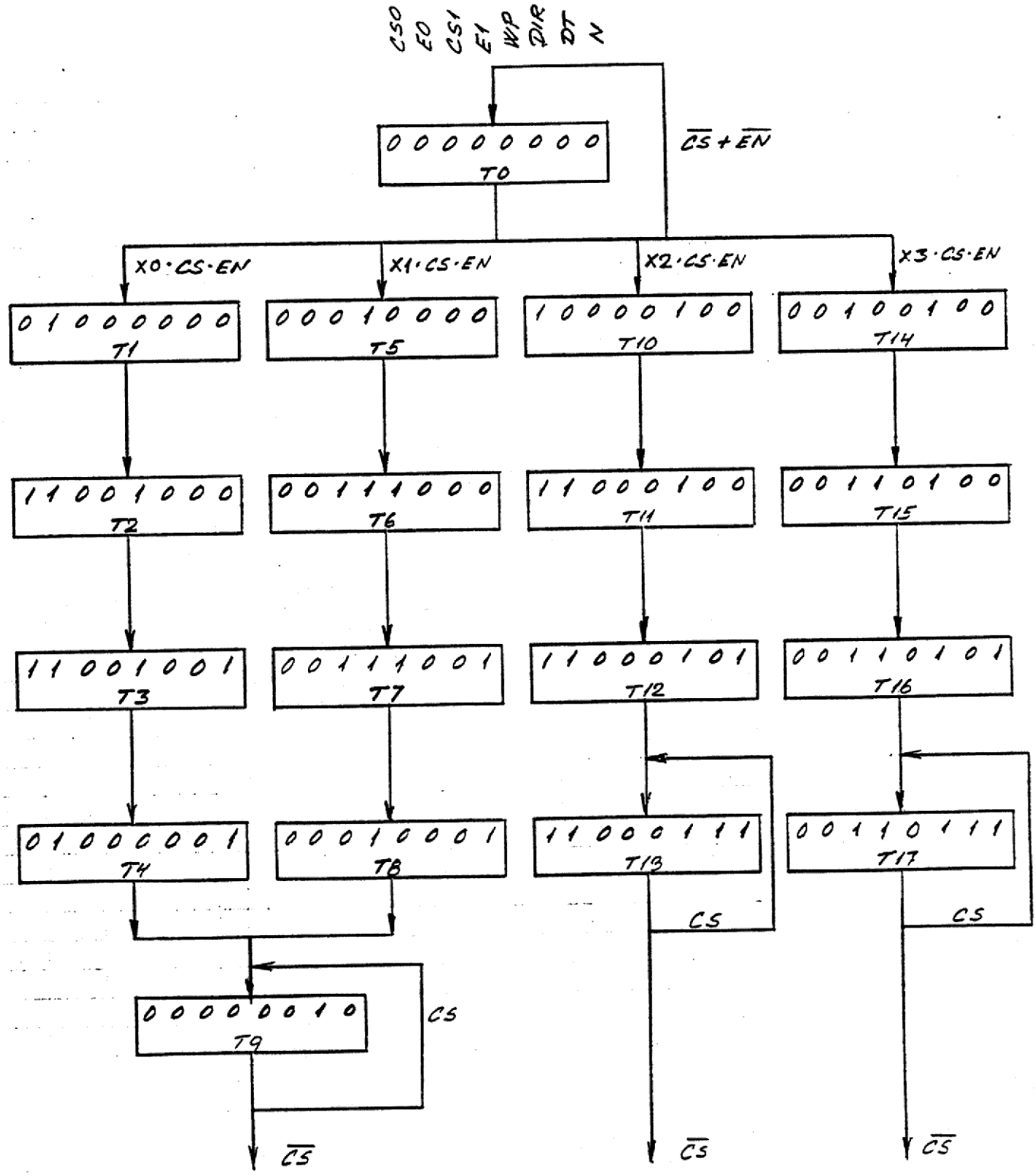
```
y1 = t2 + t3

y1 = T1*CS + T2*CS

y0 = t1 + t2

y0 = T0*CS + T1*CS

e0 = E0*CS + T3*CS*/I1*/A12

e1 = E1*CS + T3*CS*/I1*/I2*A12*/A5*/A4*/A3

e2 = E2*CS + T3*CS*/I1*/I2*A12*/A5*/A4*A3

e3 = E3*CS + T3*CS*/I1*/I2*A12*/A5*A4*/A3

e4 = E4*CS + T3*CS*/I1*/I2*A12*/A5*A4*A3

e5 = E5*CS + T3*CS*/I1*/I2*A12*A5*/A4
```

```
PAL16R8
GENERERING AF ENABLE SIGNALER TIL IOSTYRING
CPAL80: CPU PAL NR 8 VERSION 0 POSITION M3
KAN 821122
CPL  /I1  /I2   CS   GND  A12  A5   A4   A3  GND
GND  /Y1  /Y0  /E5  /E4  /E3  /E2  /E1  /E0  VCC

Y1:= /Y1 *  Y0 * /E5 * /E4 * /E3 * /E2 * /E1 * /E0 *  CS        ;T1
   +  Y1 *  Y0 * /E5 * /E4 * /E3 * /E2 * /E1 * /E0 *  CS        ;T2

Y0:= /Y1 * /Y0 * /E5 * /E4 * /E3 * /E2 * /E1 * /E0 *  CS        ;T0
   + /Y1 *  Y0 * /E5 * /E4 * /E3 * /E2 * /E1 * /E0 *  CS        ;T1


E5:=  Y1 * /Y0 * /E5 * /E4 * /E3 * /E2 * /E1 * /E0 *  CS        ;T3
   *  I1 *  I2 *  A12*  A5 * /A4
   + /Y1 * /Y0 *  E5 * /E4 * /E3 * /E2 * /E1 * /E0 *  CS        ;E5

E4:=  Y1 * /Y0 * /E5 * /E4 * /E3 * /E2 * /E1 * /E0 *  CS        ;T3
   *  I1 *  I2 *  A12* /A5 *  A4 *  A3
   + /Y1 * /Y0 * /E5 *  E4 * /E3 * /E2 * /E1 * /E0 *  CS        ;E4

E3:=  Y1 * /Y0 * /E5 * /E4 * /E3 * /E2 * /E1 * /E0 *  CS        ;T3
   *  I1 *  I2 *  A12* /A5 *  A4 * /A3
   + /Y1 * /Y0 * /E5 * /E4 *  E3 * /E2 * /E1 * /E0 *  CS        ;E3

E2:=  Y1 * /Y0 * /E5 * /E4 * /E3 * /E2 * /E1 * /E0 *  CS        ;T3
   *  I1 *  I2 *  A12* /A5 * /A4 *  A3
   + /Y1 * /Y0 * /E5 * /E4 * /E3 *  E2 * /E1 * /E0 *  CS        ;E2

E1:=  Y1 * /Y0 * /E5 * /E4 * /E3 * /E2 * /E1 * /E0 *  CS        ;T3
   *  I1 *  I2 *  A12* /A5 * /A4 * /A3
   + /Y1 * /Y0 * /E5 * /E4 * /E3 * /E2 *  E1 * /E0 *  CS        ;E1

E0:=  Y1 * /Y0 * /E5 * /E4 * /E3 * /E2 * /E1 * /E0 *  CS        ;T3
   *  I1 * /A12
   + /Y1 * /Y0 * /E5 * /E4 * /E3 * /E2 * /E1 *  E0 *  CS        ;E0

FUNCTION TABLE
```

```
CPL CS I1 I2 A12 A5 A4 A3 Y1 Y0 E5 E4 E3 E2 E1 E0
-------------------------------------------------
CLXXXXXX XXXXXXXX ??
CLXXXXXX LLLLLLLL T0
CHXXXXXX LHLLLLLL T1
CLXXXXXX LLLLLLLL T0
CHXXXXXX LHLLLLLL T1
CHXXXXXX HHLLLLLL T2
CLXXXXXX LLLLLLLL T0
CHXXXXXX LHLLLLLL T1
CHXXXXXX HHLLLLLL T2
CHXXXXXX HLLLLLLL T3
CLXXXXXX LLLLLLLL T0 10
CHXXXXXX LHLLLLLL T1
CHXXXXXX HHLLLLLL T2
CHXXXXXX HLLLLLLL T3
CHHXLXXX LLLLLLLH E0
CHXXXXXX LLLLLLLH E0
CLXXXXXX LLLLLLLL T0
CHXXXXXX LHLLLLLL T1
CHXXXXXX HHLLLLLL T2
CHXXXXXX HLLLLLLL T3
CHHHHLLL LLLLLLHL E1    20
CHXXXXXX LLLLLLHL E1
CLXXXXXX LLLLLLLL T0
CHXXXXXX LHLLLLLL T1
CHXXXXXX HHLLLLLL T2
CHXXXXXX HLLLLLLL T3
CHHHHLLH LLLLLHLL E2
CHXXXXXX LLLLLHLL E2
CLXXXXXX LLLLLLLL T0
CHXXXXXX LHLLLLLL T1
CHXXXXXX HHLLLLLL T2      30
CHXXXXXX HLLLLLLL T3
CHHHHLHL LLLLHLLL E3
CHXXXXXX LLLLHLLL E3
CLXXXXXX LLLLLLLL T0
CHXXXXXX LHLLLLLL T1
CHXXXXXX HHLLLLLL T2
```

```
CHXXXXXX HLLLLLLL T3
CHHHHLHH LLLHLLLL E4
CHXXXXXX LLLHLLLL E4
CLXXXXXX LLLLLLLL T0
CHXXXXXX LHLLLLLL T1
CHXXXXXX HHLLLLLL T2
CHXXXXXX HLLLLLLL T3
CHHHHHLX LLHLLLLL E5
CHXXXXXX LLHLLLLL E5
CLXXXXXX LLLLLLLL T0
--------------------
DESCRIPTION
;BLA BLA
```

40


CSO EO CS1 E1 WP DIR DT N
0 0 0 0 0 0 0 0 0
T0
CS + EN
X0·CS·EN
X1·CS·EN
X2·CS·EN
X3·CS·EN
0 1 0 0 0 0 0 0
T1
0 0 0 1 0 0 0 0
T5
1 0 0 0 0 1 0 0
T10
0 0 1 0 0 1 0 0
T14
1 1 0 0 1 0 0 0
T2
0 0 1 1 1 0 0 0
T6
1 1 0 0 0 1 0 0
T11
0 0 1 1 0 1 0 0
T15
1 1 0 0 1 0 0 1
T3
0 0 1 1 1 0 0 1
T7
1 1 0 0 0 1 0 1
T12
0 0 1 1 0 1 0 1
T16
0 1 0 0 0 0 0 1
T4
0 0 0 1 0 0 0 1
T8
1 1 0 0 0 1 1 1
T13
0 0 1 1 0 1 1 1
T17
CS
CS
CS
0 0 0 0 0 0 1 0
T9
CS
CS
CS
CS

t1 = T0*CS*EN*X0 X0 = /R*/A1*UDS*LDS t90

t2 = T1

t3 = T2

t4 = T3

t5 = T0*CS*EN*X1 X1 = /R*A1*UDS*LDS

t6 = T5

t7 = T6

t8 = T7

t9 = T4 + T8 + T9*CS

t10 = T0*CS*EN*X2 X2 = R*/A1*UDS*LDS

t11 = T10

t12 = T11

t13 = T12 + T13*CS

t14 = T0*CS*EN*X3 X3 = R*A1*UDS*LDS

t15 = T14

t16 = T15

t17 = T16 + T17*CS

	CS0	CE0	CS1	CE1	WP	IRD	TD	NX
T0	0	0	0	0	0	0	0	0
T1	0	1	0	0	0	0	0	0
T2	1	1	0	0	1	0	0	0
T3	1	1	0	0	1	0	0	1
T4	0	1	0	0	0	0	0	1
T5	0	0	0	1	0	0	0	0
T6	0	0	1	1	1	0	0	0
T7	0	0	1	1	1	0	0	1
T8	0	0	0	1	0	0	0	1
T9	0	0	0	0	0	0	1	0
T10	1	0	0	0	0	1	0	0
T11	1	1	0	0	0	1	0	0
T12	1	1	0	0	0	1	0	1
T13	1	1	0	0	0	1	1	1
T14	0	0	1	0	0	1	0	0
T15	0	0	1	1	0	1	0	0
T16	0	0	1	1	0	1	0	1
T17	0	0	1	1	0	1	1	1

CS0 = t2 + t3 + t10 + t11 + t12 + t13

CS0 = T1 + T2 + T0*CS*EN*X2 + T10 + T11 + T12 + T13*CS

CS1 = t6 + t7 + t14 + t15 + t16 + t17

CS1 = T5 + T6 + T0*CS*EN*X3 + T14 + T15 + T16 + T17*CS

E0 = t1 + t2 + t3 + t4 + t11 + t12 + t13

E0 = T0*CS*EN*X0 + T1 + T2 + T3 + T10 + T11 + T12 + T13*CS

E1 = t5 + t6 + t7 + t8 + t15 + t16 + t17

E1 = T0*CS*EN*X1 + T5 + T6 + T7 + T14 + T15 + T16 + T17*CS

WP = t2 + t3 + t6 + t7

$$WP = T1 + T2 + T5 + T6$$

$$DIR = t10 + t11 + t12 + t13 + t14 + t15 + t16 + t17$$

$$DIR = T0*CS*EN*(X2+X3) + (T10+T11) + T12 + T13*CS + (T14+T15) + T16 + T17*CS$$

$$DT = t9 + t13 + t17$$

$$DT = T4 + T8 + T9*CS + T12 + T13*CS + T16 + T17*CS$$

$$N = t3 + t4 + t7 + t8 + t12 + t13 + t16 + t17$$

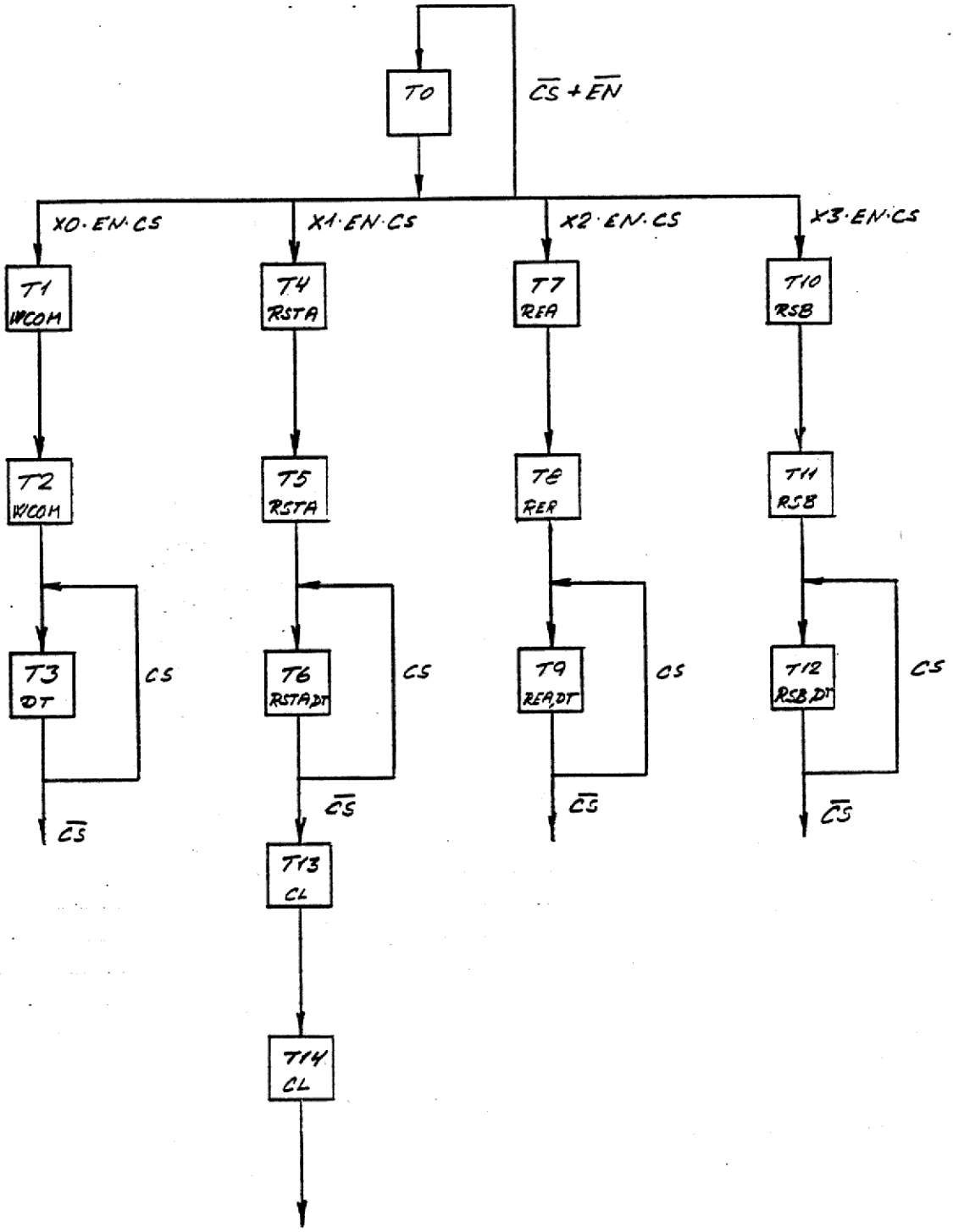
$$N = (T2+T3) + (T6+T7) + T11 + T12 + T17*CS + T15 + T16 + T17*CS$$

```
PAL16R8
PAL TIL IO STYRING AF MMU
CPAL90: CPU PAL NR. 9 VERSION 0. POSITION M4
KAN 821130
CPL  GND  /EN   CS   /UDS  /LDS  R     A1   A2   GND
GND  /DT  /N   /DIR  /WP   /E1  /CS1  /E0  /CS0  VCC

CS0 := /CS0 * /E0 * /CS1 * /E1 * /WP * /DIR * /DT * /N * CS     ;T0
     *  EN  *  R  * /A1  *  UDS*  LDS
     + /CS0 *  E0 * /CS1 * /E1 * /WP * /DIR * /DT * /N          ;T1
     +  CS0 *  E0 * /CS1 * /E1 *  WP * /DIR * /DT * /N          ;T2
     +  CS0 * /E0 * /CS1 * /E1 * /WP *  DIR * /DT * /N          ;T10
     +  CS0 *  E0 * /CS1 * /E1 * /WP *  DIR * /DT * /N          ;T11
     +  CS0 *  E0 * /CS1 * /E1 * /WP *  DIR * /DT *  N          ;T12
     +  CS0 *  E0 * /CS1 * /E1 * /WP *  DIR *  DT *  N * CS     ;T13

CS1 := /CS0 * /E0 * /CS1 * /E1 * /WP * /DIR * /DT * /N * CS     ;T0
     *  EN  *  R  *  A1  *  UDS*  LDS
     + /CS0 * /E0 * /CS1 *  E1 * /WP * /DIR * /DT * /N          ;T5
     + /CS0 * /E0 *  CS1 *  E1 *  WP * /DIR * /DT * /N          ;T6
     + /CS0 * /E0 *  CS1 * /E1 * /WP *  DIR * /DT * /N          ;T14
     + /CS0 * /E0 *  CS1 *  E1 * /WP *  DIR * /DT * /N          ;T15
     + /CS0 * /E0 *  CS1 *  E1 * /WP *  DIR * /DT *  N          ;T16
     + /CS0 * /E0 *  CS1 *  E1 * /WP *  DIR *  DT *  N * CS     ;T17

E0  := /CS0 * /E0 * /CS1 * /E1 * /WP * /DIR * /DT * /N * CS     ;T0
     *  EN  * /R  * /A1  *  UDS*  LDS
     + /CS0 *  E0 * /CS1 * /E1 * /WP * /DIR * /DT * /N          ;T1
     +  CS0 *  E0 * /CS1 * /E1 *  WP * /DIR * /DT * /N          ;T2
     +  CS0 *  E0 * /CS1 * /E1 *  WP * /DIR * /DT *  N          ;T3
     +  CS0 * /E0 * /CS1 * /E1 * /WP *  DIR * /DT * /N          ;T10
     +  CS0 *  E0 * /CS1 * /E1 * /WP *  DIR * /DT * /N          ;T11
     +  CS0 *  E0 * /CS1 * /E1 * /WP *  DIR * /DT *  N          ;T12
     +  CS0 *  E0 * /CS1 * /E1 * /WP *  DIR *  DT *  N * CS     ;T13

E1  := /CS0 * /E0 * /CS1 * /E1 * /WP * /DIR * /DT * /N * CS     ;T0
     *  EN  * /R  *  A1  *  UDS*  LDS
     + /CS0 * /E0 * /CS1 *  E1 * /WP * /DIR * /DT * /N          ;T5
     + /CS0 * /E0 *  CS1 *  E1 *  WP * /DIR * /DT * /N          ;T6
```

```
      + /CS0 * /E0 *  CS1 *  E1 *  WP * /DIR * /DT *  N          ;T7
      + /CS0 * /E0 *  CS1 * /E1 * /WP *  DIR * /DT * /N          ;T14
      + /CS0 * /E0 *  CS1 *  E1 * /WP *  DIR * /DT * /N          ;T15
      + /CS0 * /E0 *  CS1 *  E1 * /WP *  DIR * /DT *  N          ;T16
      + /CS0 * /E0 *  CS1 *  E1 * /WP *  DIR *  DT *  N * CS     ;T17

WP  := /CS0 *  E0 * /CS1 * /E1 * /WP * /DIR * /DT * /N           ;T1
     +  CS0 *  E0 * /CS1 * /E1 *  WP * /DIR * /DT * /N           ;T2
     + /CS0 * /E0 * /CS1 *  E1 * /WP * /DIR * /DT * /N           ;T5
     + /CS0 * /E0 *  CS1 *  E1 *  WP * /DIR * /DT * /N           ;T6


DIR := /CS0 * /E0 * /CS1 * /E1 * /WP * /DIR * /DT * /N * CS      ;T0
     *  EN  *  R  *          UDS*  LDS
     +  CS0 *       /CS1 * /E1 * /WP *  DIR * /DT * /N           ;T10 + T11
     +  CS0 *  E0 * /CS1 * /E1 * /WP *  DIR * /DT *  N           ;T12
     +  CS0 *  E0 * /CS1 * /E1 * /WP *  DIR *  DT *  N * CS      ;T13
     + /CS0 * /E0 *  CS1 *       /WP *  DIR * /DT * /N           ;T14 + T15
     + /CS0 * /E0 *  CS1 *  E1 * /WP *  DIR * /DT *  N           ;T16
     + /CS0 * /E0 *  CS1 *  E1 * /WP *  DIR *  DT *  N * CS      ;T17

DT  := /CS0 *  E0 * /CS1 * /E1 * /WP * /DIR * /DT *  N           ;T4
     + /CS0 * /E0 * /CS1 *  E1 * /WP * /DIR * /DT *  N           ;T8
     + /CS0 * /E0 * /CS1 * /E1 * /WP * /DIR *  DT * /N * CS      ;T9
     +  CS0 *  E0 * /CS1 * /E1 * /WP *  DIR * /DT *  N           ;T12
     +  CS0 *  E0 * /CS1 * /E1 * /WP *  DIR *  DT *  N * CS      ;T13
     + /CS0 * /E0 *  CS1 *  E1 * /WP *  DIR * /DT *  N           ;T16
     + /CS0 * /E0 *  CS1 *  E1 * /WP *  DIR *  DT *  N * CS      ;T17

N   :=  CS0 *  E0 * /CS1 * /E1 *  WP * /DIR * /DT                ;T2 + T3
     + /CS0 * /E0 *  CS1 *  E1 *  WP * /DIR * /DT                ;T6 + T7
     +  CS0 *  E0 * /CS1 * /E1 * /WP *  DIR * /DT * /N           ;T11
     +  CS0 *  E0 * /CS1 * /E1 * /WP *  DIR * /DT *  N           ;T12
     +  CS0 *  E0 * /CS1 * /E1 * /WP *  DIR *  DT *  N * CS      ;T17
     + /CS0 * /E0 *  CS1 *  E1 * /WP *  DIR * /DT * /N           ;T15
     + /CS0 * /E0 *  CS1 *  E1 * /WP *  DIR * /DT *  N           ;T16
     + /CS0 * /E0 *  CS1 *  E1 * /WP *  DIR *  DT *  N * CS      ;T17
```

FUNCTION TABLE

```
CPL CS EN UDS LDS R A1 CS0 E0 CS1 E1 WP DIR DT N
-------------------------------------------------
CLXXXXX XXXXXXXX ??
CLXXXXX XXXXXXXX ??
CLXXXXX XXXXXXXX ??
CLXXXXX XXXXXXXX ??
CLXXXXX XXXXXXXX ??
CLXXXXX XXXXXXXX ??
CLXXXXX XXXXXXXX ??
CLXXXXX XXXXXXXX ??
CLXXXXX LLLLLLLL T0
CXLXXXX LLLLLLLL T0
CXXLXXX LLLLLLLL T0
CXXXLXX LLLLLLLL T0
CHHHHLL LHLLLLLL T1
CXXXXXX HHLLHLLL T2
CXXXXXX HHLLHLLH T3
CXXXXXX LHLLLLLH T4
CXXXXXX LLLLLLHL T9
CHXXXXX LLLLLLHL T9
CLXXXXX LLLLLLLL T0
CHHHHLH LLLHLLLL T5
CXXXXXX LLHHHLLL T6
CXXXXXX LLHHHLLH T7
CXXXXXX LLLHLLLH T8
CXXXXXX LLLLLLHL T9
CLXXXXX LLLLLLLL T0
CHHHHHL HLLLLHLL T10
CXXXXXX HHLLLHLL T11
CXXXXXX HHLLLHLH T12
CXXXXXX HHLLLHHH T13
CHXXXXX HHLLLHHH T13
CLXXXXX LLLLLLLL T0
CHHHHHH LLHLLHLL T14
CXXXXXX LLHHLHLL T15
CXXXXXX LLHHLHLH T16
CXXXXXX LLHHLHHH T17
CHXXXXX LLHHLHHH T17
CLXXXXX LLLLLLLL T0
```


T0
CS + EN
X0·EN·CS
X1·EN·CS
X2·EN·CS
X3·EN·CS
T1 WCOM
T2 WCOM
T3 DT
CS
CS
T4 RSTA
T5 RSTA
T6 RSTA DT
CS
CS
T13 CL
T14 CL
T7 REA
T8 REA
T9 REA DT
CS
CS
T10 RSB
T11 RSB
T12 RSB DT
CS
CS

t100

```
t1 = T0*CS*EN*X0
t2 = T1
t3 = T2 + T3*CS
t4 = T0*CS*EN*X1
t5 = T4
t6 = T5 + T6*CS
t7 = T0*CS*EN*X2
t8 = T7
t9 = T8 + T9*CS
t10 = T0*CS*EN*X3
t11 = T10
t12 = T11 + T12*CS
t13 = T6*/CS
t14 = T13
```

```
X0 = /R*/A2*/A1*UDS*LDS

X1 = R*/A2*A1*UDS*LDS

X2 = R*A2*/A1*/UDS*LDS

X3 = R*A2*A1*UDS*LDS
```

	WCOM	RST	RERA	RSBT	RDLC	RN
T0	0	0	0	0	0	0	0
T1	1	0	0	0	0	0	0
T2	1	0	0	0	0	0	1
T3	0	0	0	0	1	0	0
T4	0	1	0	0	0	0	0
T5	0	1	0	0	0	0	1
T6	0	1	0	0	1	0	0
T7	0	0	1	0	0	0	0
T8	0	0	1	0	0	0	1
T9	0	0	1	0	1	0	0
T10	0	0	0	1	0	0	0
T11	0	0	0	1	0	0	1
T12	0	0	0	1	1	0	0
T13	0	0	0	0	0	1	0
T14	0	0	0	0	0	1	1

WCOM = t1+t2

WCOM = T0*CS*EN*/R*/A2*/A1*UDS*LDS + T1

RSTA = t4 + t5 + t6

RSTA = T0*CS*EN*R*/A2*A1*UDS*LDS + T4 + T5 + T6*CS

REA = t7 + t8 + t9

REA = T0*CS*EN*R*A2*/A1*/UDS*LDS + T7 + T8 + T9*CS

RSB = t10 + t11 + t12

RSB = T0*CS*EN*R*A2*A1*UDS*LDS + T10 + T11 + T12*CS

DT = t3 + t6 + t9 + t12

DT = T2 + T3*CS + T5 + T6*CS + T8 + T9*CS + T11 + T12*CS

CL = t13 + t14

CL = T6*/CS + T13

N = t2 + t5 + t8 + t11 + t14

N = T1 + T4 + T7 + T10 + T13

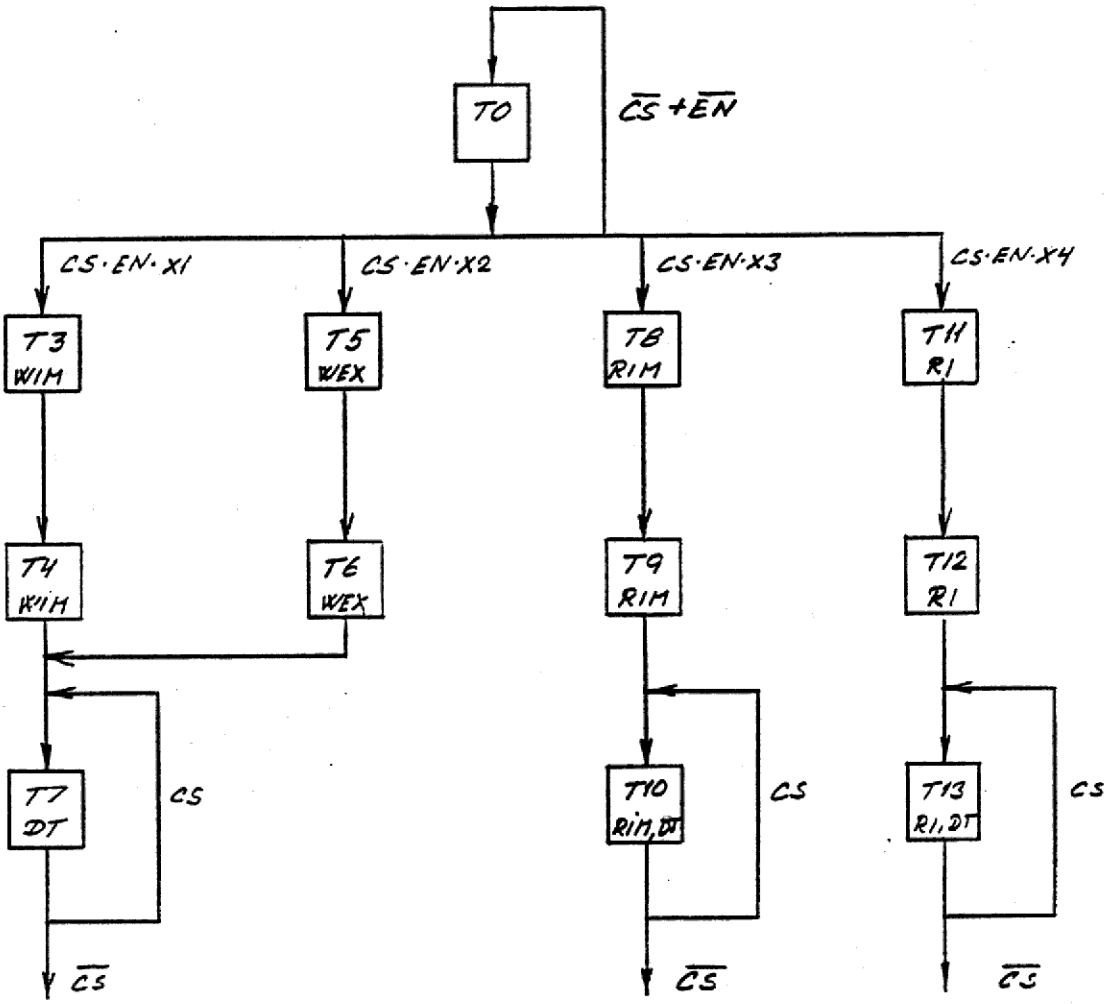
```
PAL16R8
STYRING AF KOMMANDO   STATUS   SYNDROM REGISTER
CPAL100  CPU PAL NR  10 VERSION 0  POSITION M5
KAN 821110
CPL  GND  /EN  CS   /UDS  /LDS   R     A1     A2    GND
GND  /DT   NC  /N   /CL   /RSB  /REA  /RSTA  /WCOM  VCC

WCOM  = /WCOM * /RSTA * /REA * /RSB * /CL * /DT * /N * CS        T0
      *  EN   * /R    * /A2  * /A1  *  UDS*  LDS
      +  WCOM * /RSTA * /REA * /RSB * /CL * /DT * /N             T1

RSTA  = /WCOM * /RSTA * /REA * /RSB * /CL * /DT * /N * CS        T0
      *  EN   *  R    * /A2  *  A1  *  UDS*  LDS
      + /WCOM *  RSTA * /REA * /RSB * /CL * /DT * /N             T4
      + /WCOM *  RSTA * /REA * /RSB * /CL * /DT *  N             T5
      + /WCOM *  RSTA * /REA * /RSB * /CL *  DT * /N * CS        T6

REA   = /WCOM * /RSTA * /REA * /RSB * /CL * /DT * /N * CS        T0
      *  EN   *  R    *  A2  * /A1  * /UDS*  LDS
      + /WCOM * /RSTA *  REA * /RSB * /CL * /DT * /N             T7
      + /WCOM * /RSTA *  REA * /RSB * /CL * /DT *  N             T8
      + /WCOM * /RSTA *  REA * /RSB * /CL *  DT * /N * CS        T9

RSB   = /WCOM * /RSTA * /REA * /RSB * /CL * /DT * /N * CS        T0
      *  EN   *  R    *  A2  *  A1  *  UDS*  LDS
      + /WCOM * /RSTA * /REA *  RSB * /CL * /DT * /N             T10
      + /WCOM * /RSTA * /REA *  RSB * /CL * /DT *  N             T11
      + /WCOM * /RSTA * /REA *  RSB * /CL *  DT * /N * CS        T12

CL    = /WCOM *  RSTA * /REA * /RSB * /CL *  DT * /N */CS        T6
      + /WCOM * /RSTA * /REA * /RSB *  CL * /DT * /N             T13

DT    =  WCOM * /RSTA * /REA * /RSB * /CL * /DT *  N             T2
      + /WCOM * /RSTA * /REA * /RSB * /CL *  DT * /N * CS        T3
      + /WCOM *  RSTA * /REA * /RSB * /CL * /DT *  N             T5
      + /WCOM *  RSTA * /REA * /RSB * /CL *  DT * /N * CS        T6
      + /WCOM * /RSTA *  REA * /RSB * /CL * /DT *  N             T8
      + /WCOM * /RSTA *  REA * /RSB * /CL *  DT * /N * CS        T9
      + /WCOM * /RSTA * /REA *  RSB * /CL * /DT *  N             T11
      + /WCOM * /RSTA * /REA *  RSB * /CL *  DT * /N * CS        T12
```

```
N     =  WCOM * /RSTA * /REA * /RSB * /CL * /DT * /N      T1
       + /WCOM *  RSTA * /REA * /RSB * /CL * /DT * /N     T4
       + /WCOM * /RSTA *  REA * /RSB * /CL * /DT * /N     T7
       + /WCOM * /RSTA * /REA *  RSB * /CL * /DT * /N     T10
       + /WCOM * /RSTA * /REA * /RSB *  CL * /DT * /N     T13
```

FUNCTION TABLE

```
CPL CS EN UDS LDS R A2 A1 WCOM RSTA REA RSB CL DT N

CLXXXXXX XXXXXXX ??
CLXXXXXX XXXXXXX ??
CLXXXXXX XXXXXXX ??
CLXXXXXX XXXXXXX ??
CLXXXXXX XXXXXXX ??
CLXXXXXX XXXXXXX ??
CLXXXXXX XXXXXXX ??
CLXXXXXX LLLLLLL T0
CXLXXXXX LLLLLLL T0
CXXXLXXX LLLLLLL T0
CHHLXXLL LLLLLLL T0
CHHXXHLL LLLLLLL T0
CHHLXXLH LLLLLLL T0
CHHXXLLH LLLLLLL T0
CHHHXXHL LLLLLLL T0
CHHXXLHL LLLLLLL T0
CHHLXXHH LLLLLLL T0
CHHXXLHH LLLLLLL T0
CHHHHLLL HLLLLLL T1
CXXXXXXX HLLLLLH T2
CXXXXXXX LLLLLHL T3
CHXXXXXX LLLLLHL T3
CLXXXXXX LLLLLLL T0
CHHHHHLH LHLLLLL T4
CXXXXXXX LHLLLLH T5
CXXXXXXX LHLLLHL T6
CHXXXXXX LHLLLHL T6
CLXXXXXX LLLLHLL T13
CXXXXXXX LLLLHLH T14
CXXXXXXX LLLLLLL T0
```

```
CHHLHHHL LLHLLLL T7
CXXXXXXX LLHLLLH T8
CXXXXXXX LLHLLHL T9
CHXXXXXX LLHLLHL T9
CLXXXXXX LLLLLLL T0
CHHHHHHH LLLHLLL T10
CXXXXXXX LLLHLLH T11
CXXXXXXX LLLHLHL T12
CHXXXXXX LLLHLHL T12
CLXXXXXX LLLLLLL T0


DESCRIPCTION
 BLA BLA
```


T0
CS + EN
CS·EN·X1
CS·EN·X2
CS·EN·X3
CS·EN·X4
T3 WIM
T5 WEX
T8 RIM
T11 RI
T4 WIM
T6 WEX
T9 RIM
T12 RI
T7 DT
CS
T10 RIM, DT
CS
T13 RI, DT
CS
CS
CS
CS

T110

t3 = T0*CS*EN*X1

X1 = /R*/A2*/A1*UDS*LDS

t4 = T3

t5 = T0*CS*EN*X2

X2 = /R*A2*/A1*/UDS*LDS

t6 = T5

t7 = T4 + T6 + T7*CS

t8 = T0*CS*EN*X3

X3 = R*/A2*/A1*UDS*LDS

t9 = T8

t10 = T9 + T10*CS

t11 = T0*CS*EN*X4

X4 = R*/A2*A1*UDS*LDS

t12 = T11

t13 = T12 + T13*CS

	W I M	W E X	R I M	R I	D T	N
T0	0	0	0	0	0	0
T3	1	0	0	0	0	0
T4	1	0	0	0	0	1
T5	0	1	0	0	0	0
T6	0	1	0	0	0	1
T7	0	0	0	0	1	0
T8	0	0	1	0	0	0
T9	0	0	1	0	0	1
T10	0	0	1	0	1	0
T11	0	0	0	1	0	0
T12	0	0	0	1	0	1
T13	0	0	0	1	1	0

WIM = t3 + t4

WIM = T0*CS*EN*/R*/A2*/A1*UDS*LDS + T3

WEX = t5 + t6

WEX = T0*CS*EN*/R*A2*/A1*/UDS*LDS +T5

RIM = t8 + t9 + t10

RIM = T0*CS*EN*R*/A2*/A1*UDS*LDS + T8 + T9 + T10*CS

RI = t11 + t12 + t13

RI = T0*CS*EN*R*/A2*A1*UDS*LDS + T11 + T12 + T13*CS

DT = t7 + t10 + t13

DT = T4 + T6 + T7*CS + T9 + T10*CS + T12 + T13*CS

N = t4 + t6 + t9 + t12

N = T3 + T5 + T8 + T11

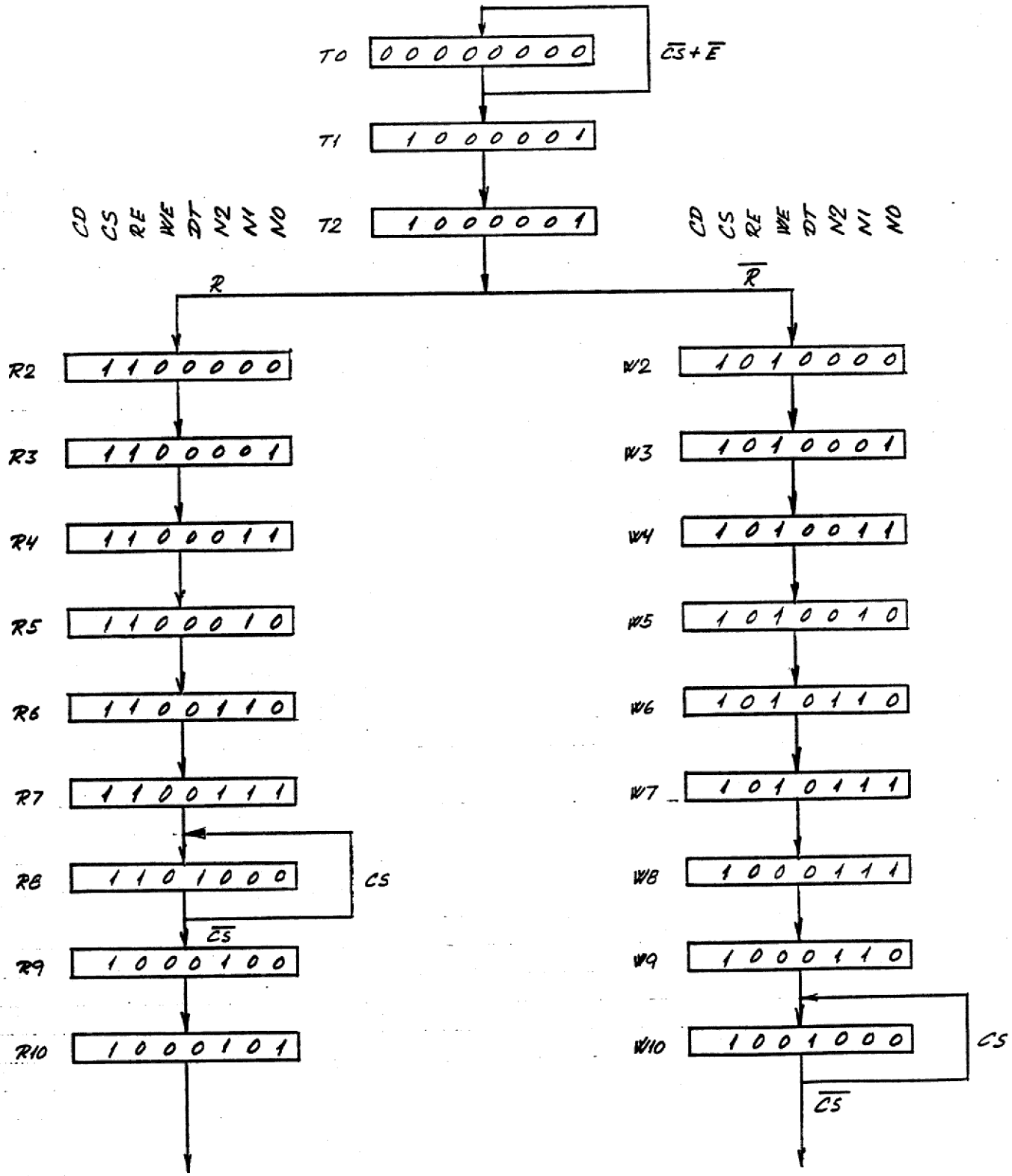
```
PAL16R8
PAL TIL STYRING AF INTERRUPT ENHED
CPAL110  CPU PAL NR. 11 VERSION 0 POSITION M6
KAN 821109
CPL   GND  /EN  CS  /UDS  /LDS   R    A1    A2   GND
GND  /DT    NC  NC  /N    /WEX  /RI  /RIM  /WIM  VCC

WIM  = /WIM * /WEX * /RIM * /RI * /DT * /N * CS                    T0
     *  EN  * /R   * /A2  * /A1 *  UDS*  LDS
     +  WIM * /WEX * /RIM * /RI * /DT * /N                         T3

WEX  = /WIM * /WEX * /RIM * /RI * /DT * /N * CS                    T0
     *  EN  * /R   *  A2  * /A1 * /UDS*  LDS
     + /WIM *  WEX * /RIM * /RI * /DT * /N                         T5

RIM  = /WIM * /WEX * /RIM * /RI * /DT * /N * CS                    T0
     *  EN  *  R   * /A2  * /A1 *  UDS*  LDS
     + /WIM * /WEX *  RIM * /RI * /DT * /N                         T8
     + /WIM * /WEX *  RIM * /RI * /DT *  N                         T9
     + /WIM * /WEX *  RIM * /RI *  DT * /N * CS                    T10

RI   = /WIM * /WEX * /RIM * /RI * /DT * /N * CS                    T0
     *  EN  *  R   * /A2  *  A1 *  UDS*  LDS
     + /WIM * /WEX * /RIM *  RI * /DT * /N                         T11
     + /WIM * /WEX * /RIM *  RI * /DT *  N                         T12
     + /WIM * /WEX * /RIM *  RI *  DT * /N * CS                    T13

DT   =  WIM * /WEX * /RIM * /RI * /DT *  N                         T4
     + /WIM *  WEX * /RIM * /RI * /DT *  N                         T6
     + /WIM * /WEX * /RIM * /RI *  DT * /N * CS                    T7
     + /WIM * /WEX *  RIM * /RI * /DT *  N                         T9
     + /WIM * /WEX *  RIM * /RI *  DT * /N * CS                    T10
     + /WIM * /WEX * /RIM *  RI * /DT *  N                         T12
     + /WIM * /WEX * /RIM *  RI *  DT * /N * CS                    T13

N    =  WIM * /WEX * /RIM * /RI * /DT * /N                         T3
     + /WIM *  WEX * /RIM * /RI * /DT * /N                         T5
     + /WIM * /WEX *  RIM * /RI * /DT * /N                         T8
     + /WIM * /WEX * /RIM *  RI * /DT * /N                         T11
```

FUNCTION TABLE

CPL CS EN UDS LDS R A2 A1 WIM WEX RIM RI DT N

CLXXXXXX XXXXXX ??
CLXXXXXX XXXXXX ??
CLXXXXXX XXXXXX ??
CLXXXXXX XXXXXX ??
CLXXXXXX XXXXXX ??
CLXXXXXX LLLLLL T0
CXLXXXXX LLLLLL T0
CXXXLXXX LLLLLL T0
CHHLHXLL LLLLLL T0
CHHLHXLH LLLLLL T0
CHHXHLLH LLLLLL T0
CHHHHXHL LLLLLL T0
CHHXHHHL LLLLLL T0
CXXXXXHH LLLLLL T0
CHHHHLLL HLLLLL T3
CXXXXXXX HLLLLH T4
CXXXXXXX LLLLHL T7
CHXXXXXX LLLLHL T7
CLXXXXXX LLLLLL T0
CHHLHLHL LHLLLL T5
CXXXXXXX LHLLLH T6
CXXXXXXX LLLLHL T7
CLXXXXXX LLLLLL T0
CHHHHHLL LLHLLL T8
CXXXXXXX LLHLLH T9
CXXXXXXX LLHLHL T10
CHXXXXXX LLHLHL T10
CLXXXXXX LLLLLL T0
CHHHHHLH LLLHLL T11
CXXXXXXX LLLHLH T12
CXXXXXXX LLLHHL T13
CHXXXXXX LLLHHL T13
CLXXXXXX LLLLLL T0

DESCRIPCTION
:BLA BLA


T0
0 0 0 0 0 0 0 0
CS + E
T1
1 0 0 0 0 0 1
T2
1 0 0 0 0 0 1
CD CS RE WE DT N2 N1 N0
R
R
R2 1 1 0 0 0 0 0
R3 1 1 0 0 0 0 1
R4 1 1 0 0 0 1 1
R5 1 1 0 0 0 1 0
R6 1 1 0 0 1 1 0
R7 1 1 0 0 1 1 1
R8 1 1 0 1 0 0 0
CS
CS
R9 1 0 0 0 1 0 0
R10 1 0 0 0 1 0 1
W2 1 0 1 0 0 0 0
W3 1 0 1 0 0 0 1
W4 1 0 1 0 0 1 1
W5 1 0 1 0 0 1 0
W6 1 0 1 0 1 1 0
W7 1 0 1 0 1 1 1
W8 1 0 0 0 1 1 1
W9 1 0 0 0 1 1 0
W10 1 0 0 1 0 0 0
CS
CS

t120

```
t1 = T0*CS*EN*X0
t2 = T1
r2 = T2*R
r3 = R2
r4 = R3
r5 = R4
r6 = R5
r7 = R6
r8 = R7 + R8*CS
r9 = R8*/CS
r10= R9
```

```
·X0 = /A2*/UDS*LDS

w2 = T2*/R
w3 = W2
w4 = W3
w5 = W4
w6 = W5
w7 = W6
w8 = W7
w9 = W8
w10= W9 + W10*CS
```

	CE	RE	WE	DT	N2	N1	N0
T0:	0	0	0	0	0	0	0
T1:	1	0	0	0	0	0	1
T2:	1	0	0	0	0	0	0
R2:	1	1	0	0	0	0	0
R3:	1	1	0	0	0	0	1
R4:	1	1	0	0	0	1	1
R5:	1	1	0	0	0	1	0
R6:	1	1	0	0	1	1	0
R7:	1	1	0	0	1	1	1
R8:	1	1	0	1	0	0	0
R9:	1	0	0	0	1	0	0
R10:	1	0	0	0	1	0	1
W2:	1	0	1	0	0	0	0
W3:	1	0	1	0	0	0	1
W4:	1	0	1	0	0	1	1
W5:	1	0	1	0	0	1	0
W6:	1	0	1	0	1	1	0
W7:	1	0	1	0	1	1	1
W8:	1	0	0	0	1	1	1
W9:	1	0	0	0	1	1	0
W10:	1	0	0	1	0	0	0

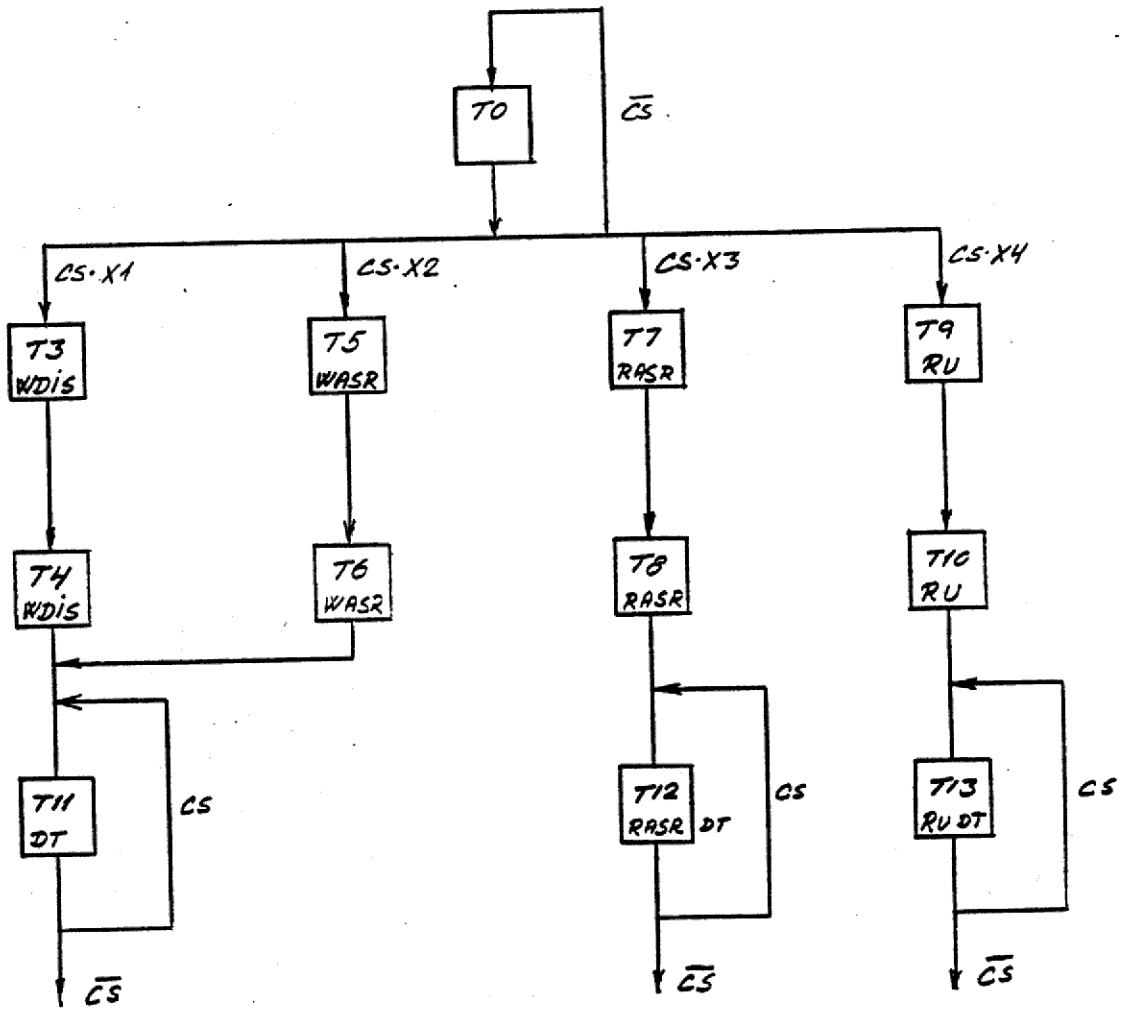
```
CE = t1 + t2 + r2 + r3 + r4 + r5 + r6 + r7 + r8 + r9 + r10
                + w2 + w3 + w4 + w5 + w6 + w7 + w8 + w9 + w10

CE = TO*CS*EN*A2*/UDS*LDS + (T1+T2+R2+R3) + (R4+R5+R6+R7) + R8
     + R9 + (W2+W3+W4+W5) + (W6+W7+W8+W9) +W10*CS

RE = r2 + r3 + r4 + r5 + r6 + r7 + r8

RE = T2*R + (R2+R3+R4+R5) + (R6+R7) + R8*CS

WE = w2 + w3 + w4 + w5 + w6 + w7

WE = T2*/R + (W2+W3+W4+W5) + W6

DT = r8 + w10

DT = R7 + R8*CS + W9 + W10*CS

N2 = r6 + r7 + r9 + r10 + w6 + w7 + w8 + w9

N2 = (R5+R6) + R8*/CS + R9 + (W5+W6) + (W7+W8)

N1 = r4 + r5 + r6 + r7 + w4 + w5 + w6 + w7 + w8 + w9

N1 = (R3+R4) + (R5+R6) + (W3+W4) + (W5+W6) + (W7+W8)

NO = t1 + r3 + r4 + r7 + r10 + w3 + w4 + w7 + w8

NO = TO*CS*EN*/A2*/UDS*LDS + (R2+R3) + R6 + R9 + (W2+W3)
   + (W6+W7)

CD = TO*CS*EN*/A2*/A1*/UDS*LDS
   + CD*((T1+T2+R2+R3) + (R4+R5+R6+R7) + R8 + R9)
   + CD*((W2+W3+W4+W5) + (W6+W7+W8+W9) + W10*CS)
```

```
PAL16R8
STYRING AF USART
CPAL120  CPU PAL NR 12 VERSION 0. POSITION M7
KAN 821119
CPL   GND  /EN   CS  /UDS  /LDS  R    A1   A2  GND
GND  /DT   /N0  /N1  /N2   /RE  /WE  /CD  /CE  VCC

CE  := /CE * /RE * /WE * /DT * /N2 * /N1 * /N0 * CS          ;T0
     *  EN * /A2 */UDS *  LDS
     +  CE *        /WE * /DT * /N2 * /N1                    ;T1+T2+R2+R3
     +  CE *  RE * /WE * /DT *        N1                     ;R4+R5+R6+R7
     +  CE *  RE * /WE *  DT * /N2 * /N1 * /N0               ;R8
     +  CE * /RE * /WE * /DT *  N2 * /N1 * /N0               ;R9
     +  CE * /RE *  WE * /DT * /N2                           ;W2+W3+W4+W5
     +  CE * /RE *       /DT *  N2 *  N1                    ;W6+W7+W8+W9
     +  CE * /RE * /WE *  DT * /N2 * /N1 * /N0 * CS          ;W10

RE  :=  CE * /RE * /WE * /DT * /N2 * /N1 * /N0 * R           ;T2
     +  CE *  RE * /WE * /DT * /N2                           ;R2+R3+R4+R5
     +  CE *  RE * /WE * /DT *  N2 *  N1                     ;R6+R7
     +  CE *  RE * /WE *  DT * /N2 * /N1 * /N0 * CS          ;R8

WE  :=  CE * /RE * /WE * /DT * /N2 * /N1 * /N0 * /R          ;T2
     +  CE * /RE *  WE * /DT * /N2                           ;W2+W3+W4+W5
     +  CE * /RE *  WE * /DT *  N2 *  N1 * /N0               ;W6

DT  :=  CE *  RE * /WE * /DT *  N2 *  N1 *  N0               ;R7
     +  CE *  RE * /WE *  DT * /N2 * /N1 * /N0 * CS          ;R8
     +  CE * /RE * /WE * /DT *  N2 *  N1 * /N0               ;W9
     +  CE * /RE * /WE *  DT * /N2 * /N1 * /N0 * CS          ;W10

N2  :=  CE *  RE * /WE * /DT *        N1 * /N0               ;R5+R6
     +  CE *  RE * /WE *  DT * /N2 * /N1 * /N0 */CS          ;R8
     +  CE * /RE * /WE * /DT *  N2 * /N1 * /N0               ;R9
     +  CE * /RE *  WE * /DT *        N1 * /N0               ;W5+W6
     +  CE * /RE *       /DT *  N2 *  N1 *  N0               ;W7+W8

N1  :=  CE *  RE * /WE * /DT * /N2 *        N0               ;R3+R4
     +  CE *  RE * /WE * /DT *        N1 * /N0               ;R5+R6
```

```
      +  CE * /RE *  WE * /DT * /N2 *        NO              ;W3+W4
      +  CE * /RE *  WE * /DT *       N1 * /NO              ;W5+W6
      +  CE * /RE *       /DT *  N2 *  N1 *  NO              ;W7+W8

NO  := /CE * /RE * /WE * /DT * /N2 * /N1 * /NO * CS          ;T0
       *  EN * /A2 */UDS *  LDS
      +  CE *  RE * /WE * /DT * /N2 * /N1                    ;R2+R3
      +  CE *  RE * /WE * /DT *  N2 *  N1 * /NO              ;R6
      +  CE * /RE * /WE * /DT *  N2 * /N1 * /NO              ;R9
      +  CE * /RE *  WE * /DT * /N2 * /N1                    ;W2+W3
      +  CE * /RE *  WE * /DT *  N2 *  N1                    ;W6+W7

CD  := /CE * /RE * /WE * /DT * /N2 * /N1 * /NO * CS          ;T0
       *  EN * /A2 * /A1 * /UDS*  LDS
      +  CE *       /WE * /DT * /N2 * /N1 *       CD        ;T2+T3+R2+R3
      +  CE *  RE * /WE * /DT *       N1 *       CD        ;R4+R5+R6+R7
      +  CE *  RE * /WE *  DT * /N2 * /N1 * /NO * CD        ;R8
      +  CE * /RE * /WE * /DT *  N2 * /N1 * /NO * CD        ;R9
      +  CE * /RE *  WE * /DT * /N2 *            CD        ;W2+W3+W4+W5
      +  CE * /RE *       /DT *  N2 *  N1 *       CD        ;W6+W7+W8+W9
      +  CE * /RE * /WE *  DT * /N2 * /N1 * /NO * CD        ;W10

FUNCTION TABLE
CPL CS EN UDS LDS R A2 A1 CD CE RE WE DT N2 N1 NO
-------------------------------------------------
CLXXXXXX XXXXXXXX ??
CLXXXXXX XXXXXXXX ??
CLXXXXXX XXXXXXXX ??
CLXXXXXX XXXXXXXX ??
CLXXXXXX XXXXXXXX ??
CLXXXXXX XXXXXXXX ??
CLXXXXXX XXXXXXXX ??
CLXXXXXX XXXXXXXX ??
CLXXXXXX XXXXXXXX ??
CLXXXXXX XXXXXXXX ??
CLXXXXXX XXXXXXXX ??
CLXXXXXX XXXXXXXX ??
CLXXXXXX XXXXXXXX ??
CLXXXXXX XXXXXXXX ??
```

```
CLXXXXXX LLLLLLLL T0
CXLXXXXX LLLLLLLL T0
CXXHXXXX LLLLLLLL T0
CXXXLXXX LLLLLLLL T0
CXXXXXHX LLLLLLLL T0
CHHLHXLH LHLLLLLH TA1
CXXXXXXX LHLLLLLL TA2
CXXXXHXX LHHLLLLL RA2
CXXXXXXX LHHLLLLH RA3
CXXXXXXX LHHLLLHH RA4 10
CXXXXXXX LHHLLLHL RA5
CXXXXXXX LHHLLHHL RA6
CXXXXXXX LHHLLHHH RA7
CXXXXXXX LHHLHLLL RA8
CHXXXXXX LHHLHLLL RA8
CLXXXXXX LHLLLHLL RA9
CXXXXXXX LHLLLHLH RA10
CXXXXXXX LLLLLLLL T0
CHHLHXLL HHLLLLLH TB1
CXXXXXXX HHLLLLLL TB2    20
CXXXXHXX HHHLLLLL RB2
CXXXXXXX HHHLLLLH RB3
CXXXXXXX HHHLLLHH RB4
CXXXXXXX HHHLLLHL RB5
CXXXXXXX HHHLLHHL RB6
CXXXXXXX HHHLLHHH RB7
CXXXXXXX HHHLHLLL RB8
CHXXXXXX HHHLHLLL RB8
CLXXXXXX HHLLLHLL RB9
CXXXXXXX HHLLLHLH RB10     30
CXXXXXXX LLLLLLLL T0
CHHLHXLH LHLLLLLH TA1
CXXXXXXX LHLLLLLL TA2
CXXXXLXX LHLHLLLL WA2
CXXXXXXX LHLHLLLH WA3
CXXXXXXX LHLHLLHH WA4
CXXXXXXX LHLHLLHL WA5
CXXXXXXX LHLHLHHL WA6
CXXXXXXX LHLHLHHH WA7
```

```
CXXXXXXX LHLLLHHH WA8
CXXXXXXX LHLLLHHL WA9
CXXXXXXX LHLLHLLL WA10
CHXXXXXX LHLLHLLL WA10
CLXXXXXX LLLLLLLL T0
CHHLHXLL HHLLLLLH TB1
CXXXXXXX HHLLLLLL TB2
CXXXXLXX HHLHLLLL WB2
CXXXXXXX HHLHLLLH WB3
CXXXXXXX HHLHLLHH WB4
CXXXXXXX HHLHLLHL WB5
CXXXXXXX HHLHLHHL WB6
CXXXXXXX HHLHLHHH WB7
CXXXXXXX HHLLLHHH WB8
CXXXXXXX HHLLLHHL WB9
CXXXXXXX HHLLHLLL WB10
CHXXXXXX HHLLHLLL WB10
CLXXXXXX HLLLLLLL TXX
CLXXXXXX LLLLLLLL T0
-------------------------
DESCRIPTION
;BLA BLA
```


T0
CS
CS·X1
CS·X2
CS·X3
CS·X4
T3 WDIS
T5 WASR
T7 RASR
T9 RU
T4 WDIS
T6 WASR
T8 RASR
T10 RU
T11 DT
CS
T12 RASR DT
CS
T13 RU DT
CS
CS
CS
CS

t130

```
t3 = T0*CS*X1
t4 = T3
t5 = T0*CS*X2
t6 = T5
t7 = T0*CS*X3
t8 = T7
t9 = T0*CS*X4
t10 = T9
t11 = T4 + T6 + T11*CS
t12 = T8 + T12*CS
t13 = T10 + T13*CS
```

```
·X1 = /E5*E3*A2*/A1*UDS*LDS*/R
·X2 = /E5*E3*A2*A1*/UDS*LDS*/R
·X3 = /E5*E3*A2*A1*/UDS*LDS*R
·X4 = E5*/E3*/UDS*LDS*R
```

	WDI	WASR	RASR	RUDS	TLDS	NR
T0:	0	0	0	0	0	0
T3:	1	0	0	0	0	0
T4:	1	0	0	0	0	1
T5:	0	1	0	0	0	0
T6:	0	1	0	0	0	1
T7·	0	0	1	0	0	0
T8:	0	0	1	0	0	1
T9·	0	0	0	1	0	0
T10:	0	0	0	1	0	1
T11:	0	0	0	0	1	0
T12:	0	0	1	0	1	0
T13:	0	0	0	1	1	0

WDIS = t3 + t4

WDIS = TO*CS*/E5*E3*A2*/A1*UDS*LDS*/R + T3

WASR = t5 + t6

WASR = TO*CS*/E5*E3*A2*A1*/UDS*LDS*/R + T5

RASR = t7 + t8 + t12

RASR = TO*CS*/E5*E3*A2*A1*/UDS*LDS*R + T7 + T8 + T12*CS

RU = t9 + t10 + t13

RU = TO*CS*E5*/E3*/UDS*LDS*R + T9 + T10 + T13*CS

DT = t11 + t12 + t13

DT = T4 + T6 + T11*CS + T8 + T12*CS + T10 + T13*CS

N = t4 + t6 + t8 + t10

N = T3 + T5 + T7 + T9

```
PAL16R8
PAL TIL STYRING AF DISP. ASR OG UNITNUMBER
CPAL130: CPU PAL NR. 13 VERSION 0. POSITION M8
KAN 821122
CPL   /E3   /E5   CS   /UDS    /LDS   R       A1      A2     GND
GND   /DT   /N    NC    NC     /RU   /RASR   /WASR   /WDIS   VCC

WDIS := /WDIS * /WASR * /RASR * /RU *  /DT *  /N * CS          ;T0
      *  E3   * A2    * /A1   * UDS *  LDS * /R * /E5
      +  WDIS * /WASR * /RASR * /RU *  /DT *  /N               ;T3

WASR := /WDIS * /WASR * /RASR * /RU *  /DT *  /N * CS          ;T0
      *  E3   * A2    *  A1   * /UDS *  LDS * /R * /E5
      + /WDIS *  WASR * /RASR * /RU *  /DT *  /N               ;T5

RASR := /WDIS * /WASR * /RASR * /RU *  /DT *  /N * CS          :T0
      *  E3   * A2    *  A1   * /UDS *  LDS *  R * /E5
      + /WDIS * /WASR *  RASR * /RU *  /DT *  /N               ;T7
      + /WDIS * /WASR *  RASR * /RU *  /DT *   N               ;T8
      + /WDIS * /WASR *  RASR * /RU *   DT *  /N * CS          ;T12

RU   := /WDIS * /WASR * /RASR * /RU *  /DT *  /N * CS          ;T0
      *  E5   *                 /UDS *  LDS *  R * /E3
      + /WDIS * /WASR * /RASR *  RU *  /DT *  /N               ;T9
      + /WDIS * /WASR * /RASR *  RU *  /DT *   N               :T10
      + /WDIS * /WASR * /RASR *  RU *   DT *  /N * CS          ;T13

N    :=  WDIS * /WASR * /RASR * /RU *  /DT *  /N               ;T3
      + /WDIS *  WASR * /RASR * /RU *  /DT *  /N               ;T5
      + /WDIS * /WASR *  RASR * /RU *  /DT *  /N               ;T7
      + /WDIS * /WASR * /RASR *  RU *  /DT *  /N               ;T9

DT   :=  WDIS * /WASR * /RASR * /RU *  /DT *   N               ;T4
      + /WDIS *  WASR * /RASR * /RU *  /DT *   N               ;T6
      + /WDIS * /WASR * /RASR * /RU *   DT *  /N * CS          :T11
      + /WDIS * /WASR *  RASR * /RU *  /DT *   N               ;T8
      + /WDIS * /WASR *  RASR * /RU *   DT *  /N * CS          ;T12
      + /WDIS * /WASR * /RASR *  RU *  /DT *   N               ;T10
      + /WDIS * /WASR * /RASR *  RU *   DT *  /N * CS          :T13
```

FUNCTION TABLE

CPL	CS	E3	E5	UDS	LDS	R	A2	A1	WDIS	WASR	RASR	RU	DT	N
CLX	XXX	XXX		XX	XX	X	X	X	XXX	XXX	XXX	?	?	?
CLX	XXX	XXX		XX	XX	X	X	X	XXX	XXX	XXX	?	?	?
CLX	XXX	XXX		XX	XX	X	X	X	XXX	XXX	XXX	?	?	?
CLX	XXX	XXX		XX	XX	X	X	X	XXX	XXX	XXX	?	?	?
CLX	XXX	XXX		XX	XX	X	X	X	XXX	XXX	XXX	?	?	?
CXX	XXL	XXX		LL	LL	L	L	L	T	O				
CLX	XXX	XXX		LL	LL	L	L	L	T	O				
CHH	LXX	XXX		LL	LL	L	L	L	T	O				
CHH	HXX	XXX		LL	LL	L	L	L	T	O				
CHH	LHH	HXX		LL	LL	L	L	L	T	O				
CHH	LHH	LLX		LL	LL	L	L	L	T	O				
CHH	LHH	LHH		LL	LL	L	L	L	T	O				
CHH	LLH	XLX		LL	LL	L	L	L	T	O				
CHH	LLH	XXL		LL	LL	L	L	L	T	O				
CHL	HXX	XXX		LL	LL	L	L	L	T	O				
CHL	HLH	LXX		LL	LL	L	L	L	T	O				
CHH	LHH	LHL		HL	LL	L	L	L	T	3				
CXX	XXX	XXX		HL	LL	L	L	H	T	4				
CXX	XXX	XXX		LL	LL	H	L	L	T	11				
CHX	XXX	XXX		LL	LL	H	L	L	T	11				
CLX	XXX	XXX		LL	LL	L	L	L	T	O				
CHH	LLH	LHH		LH	LL	L	L	L	T	5				
CXX	XXX	XXX		LH	LL	L	L	H	T	5				
CXX	XXX	XXX		LL	LL	H	L	L	T	11				
CLX	XXX	XXX		LL	LL	L	L	L	T	O				
CHH	LLH	HHH		LL	HL	L	L	L	T	7				
CXX	XXX	XXX		LL	HL	L	L	H	T	8				
CXX	XXX	XXX		LL	HL	H	L	L	T	12				
CHX	XXX	XXX		LL	HL	H	L	L	T	12				
CLX	XXX	XXX		LL	LL	L	L	L	T	O				
CHL	HLH	HXX		LL	LH	L	L	L	T	9				
CXX	XXX	XXX		LL	LH	L	L	H	T	10				
CXX	XXX	XXX		LL	LH	H	L	L	T	13				
CHX	XXX	XXX		LL	LH	H	L	L	T	13				
CLX	XXX	XXX		LL	LL	L	L	L	T	O				

```
PAL16L8
PAL TIL DEKODNING AF CYCLES
CPAL140· CPU PAL NR. 14 VERSION 0. POSITION M9
KAN 821111
C2   C5   C7   CON  FC2  FC1  FC0  A23  A22  GND
A21  /U   /T   A20  /S   /INA /ILU /NA  /NAT  VCC

·C2: COMMAND REGISTER BIT 2: ENABLE ADDRESS MAPPING
:C5: COMMAND REGISTER BIT 5: ENABLE PROTECTION AGAINST USER IN
:                            SEGMENT 0 AND SEGMENT 1
:C7: COMMAND REGISTER BIT 7: ENABLE ACCESS TO REGISTERS FROM BUS
:
:S: SUPERVISOR MODE ACTIVE HIGH
:S= /CON*(SUPP+SUPD)

IF (VCC) S =
     CON + /FC2 +  FC1 *  FC0 + /FC1 * /FC0

:U: USER MODE ACTIVE HIGH
:U= /CON*(USP+USD)

IF (VCC) U =
     CON +  FC2 +  FC1 *  FC0 + /FC1 * /FC0

:INA: INTERRUPT ACKNOWLEDGE CYCLE. ACTIVE HIGH

IF (VCC) INA =
      CON  + /FC2 + /FC1 + /FC0

:NAT: NO ADDRESS TRANSLATION. ACTIVE HIGH
:NAT= /C2 + /CON * SEG0 + CON*C7*SEG0

IF (VCC) NAT =
      C2 * CON * /C7
    + C2 * A23
    + C2 * A22
    + C2 * A21
    + C2 * A20
```

```
:T: ENABLE ACCESS CODE TEST. ACTIVE HIGH
:T= SEG215*(SUPP+SUPD+USP+USD)*/CON
: + SEG215*CON
: + (SEGO+SEG1)*CON*/C7

IF (VCC) T =
    /CON * /A23 * /A22 * /A21
  +  C7  * /A23 * /A22 * /A21
  + /CON * /FC1 * /FCO
  + /CON *  FC1 *  FCO

:ILU: ILLEGAL USER. ACTIVE LOW
:ILU=/CON*(USP+USD)*(SEGO+SEG1)*C5

IF (VCC) ILU =
      /CON * /FC2 * /FC1 *  FCO * /A23 * /A22 * /A21 *  C5
    + /CON * /FC2 *  FC1 * /FCO * /A23 * /A22 * /A21 *  C5

:NA: NO ACCESS TO MEMORY. ACTIVE LOW
:NA=/CON*INA+/CON*SEG1+/CON*USP*SEGO*C5+/CON*USD*SEGO*C5+CON*SEG1*C7

IF (VCC) NA =
     /CON *  FC2 *  FC1 *  FCO
   + /CON * /A23 * /A22 * /A21 *  A20
   + /CON * /A23 * /A22 * /A21 * /A20 * /FC2 *  FC1 * /FCO *  C5
   + /CON * /A23 * /A22 * /A21 * /A20 * /FC2 * /FC1 *  FCO *  C5
   +  CON * /A23 * /A22 * /A21 *  A20 *  C7

FUNCTION TABLE
C7 C5 C2 CON FC2 FC1 FCO A23 A22 A21 A20  S U INA NAT T ILU NA
-------------------------------------------------------------
XXXHHLHXXXXHXXXXXX
XXXLHLHXXXXLXXXXXX
XXXLLLHXXXXHXXXXXX
XXXLHLHXXXXLXXXXXX
XXXLHHHXXXXHXXXXXX
XXXLHLLXXXXHXXXXXX
XXXLHHLXXXXLXXXXXX
XXXHLLHXXXXXHXXXXX
```

```
XXXLLLHXXXXXLXXXXX
XXXLHLHXXXXXHXXXXX
XXXLLLHXXXXXLXXXXX
XXXLLHLXXXXXLXXXXX
XXXLLHHXXXXXHXXXXX
XXXLLHHXXXXXHXXXXX
XXXHHHHXXXXXXHXXXX
XXXLHHHXXXXXXLXXXX
XXXLLHHXXXXXXHXXXX
XXXLHLHXXXXXXHXXXX
XXXLHHLXXXXXXHXXXX
LXHHXXXLLLLXXXHXXX
HXHHXXXLLLLXXXLXXX
LXLHXXXLLLLXXXLXXX
LXHLXXXLLLLXXXLXXX
XXHLXXXHLLLXXXHXXX
XXLXXXXHLLLXXXLXXX
XXHXXXXLHLLXXXHXXX
XXLXXXXLHLLXXXLXXX
XXHXXXXLLHLXXXHXXX
XXLXXXXLLHLXXXLXXX
XXHXXXXLLLHXXXHXXX
XXLXXXXLLLHXXXLXXX
LXXLXLHLLLXXXXXHXX
LXXHXLLLLLXXXXXLXX
LXXLXLHHLLXXXXXLXX
LXXLXLHLHLXXXXXLXX
LXXLXLHLLHXXXXXLXX
HXXHXXXLLLXXXXXHXX
LXXHXXXLLLXXXXXLXX
HXXHXXXHLLXXXXXLXX
HXXHXXXLHLXXXXXLXX
HXXHXXXLLHXXXXXLXX
XXXLXLLHHHXXXXXHXX
XXXHXLLHHHXXXXXLXX
XXXLXLHHHHXXXXXLXX
XXXLXHLHHHXXXXXLXX
XXXLXHHHHHXXXXXHXX
XXXHXHHHHHXXXXXLXX
```

```
XHXLLLHLLLXXXXXXHX
XLXHLLHLLLXXXXXXLX
XHXLHLHLLLXXXXXXLX
XHXLLHHLLLXXXXXXLX
XHXLLLLLLLXXXXXXLX
XHXLLLHHLLXXXXXXLX
XHXLLLHLHLXXXXXXLX
XHXLLLHLLHXXXXXXLX
XHXLLLHLLLXXXXXXHX
XHXLLHLLLLXXXXXXHX
XLXLLHLLLLXXXXXXLX
XHXHLHLLLLXXXXXXLX
XHXLHHLLLLXXXXXXLX
XHXLLLLLLLXXXXXXLX
XHXLLHHLLLXXXXXXLX
XHXLLHLHLLXXXXXXLX
XHXLLHLLHLXXXXXXLX
XHXLLHLLLLFXXXXXXLX
XXXLHHHHHHXXXXXXXH
XXXHHHHHHHXXXXXXXL
XXXLLHHHHHXXXXXXXL
XXXLHLHHHHXXXXXXXL
XXXLHHLHHHXXXXXXXL
LLXLLLLLLLHXXXXXXH
LLXHLLLLLLHXXXXXXL
LLXLLLLHLLHXXXXXXL
LLXLLLLLHLHXXXXXXL
LLXLLLLLLHHXXXXXXL
LLXLLLLLLLLXXXXXXL
LHXLLHLLLLLXXXXXXH
LLXLLHLLLLLXXXXXXL
LHXHLHLLLLLXXXXXXL
LHXLHHLLLLLXXXXXXL
LHXLLLLLLLLXXXXXXL
LHXLLHHLLLLXXXXXXL
LHXLLHLHLLLXXXXXXL
LHXLLHLLHLLXXXXXXL
LHXLLHLLLHLXXXXXXL
LHXLLHLLLLHXXXXXXH
```

50

60

70

80

```
LHXLLLHLLLLXXXXXXH
LLXLLLHLLLLXXXXXXL
LHXHLLHLLLLXXXXXXL
LHXLHLHLLLLXXXXXXL                    90
LHXLLHHLLLLXXXXXXL
LHXLLLLLLLLXXXXXXL
LHXLLLHHLLLXXXXXXL
LHXLLLHLHLLXXXXXXL
LHXLLLHLLHLXXXXXXL
LHXLLLHLLLHXXXXXXH
HLXHLLLLLLHXXXXXXH
HLXLLLLLLLHXXXXXXH
HLXHLLLHLLHXXXXXXL
HLXHLLLLHLHXXXXXXL                   100
HLXHLLLLLHHXXXXXXL
HLXHLLLLLLLXXXXXXL
```

DESCRIPTION
;BLABLA

```
PAL16L8
PAL TIL TEST AF ACCESS KODE FRA MMU
CPAL151· CPU PAL NR. 15 VERSION 1. POSITION NO
KAN 831010
S      U      T      CON     R      C3     AC3     AC2     AC1      GND
NC     /BL    ACO    TL     /SEGTL /RO    /ILB    /ILU    /NA       VCC

:SEGMENT TOO LONG: SEGTL. ACTIVE HIGH
;SEGTL=T*C3*TL

IF (VCC) SEGTL =
        /T + /C3 + /TL

:READ ONLY: RO. ACTIVE HIGH
:RO=T*C3*/R*/AC2

IF (VCC) RO =
     /T + /C3 + R + AC2

:ILLEGAL BUS ACCESS: ILB. ACTIVE HIGH
:ILB=T*C3*CON*/ACO

IF (VCC) ILB =
      /T + /C3 + /CON + ACO

:ILLEGAL USER ACCESS: ILU. ACTIVE HIGH
:ILU=T*C3*U*/AC1

IF (VCC) ILU =
      /T + /C3 + /U + AC1

:NO ACCESS TO MEMORY: NA. ACTIVE LOW

IF (VCC) NA =
     T * C3 *  TL
   + T * C3 *  CON * /ACO
   + T * C3 *  U   * /AC1
   + T * C3 * /R   * /AC2
   + T * C3 *        AC3
```

:LONG ACCESS TO BUS
:BL=T*AC3

IF (VCC) BL =
    /T + /AC3

FUNCTION TABLE

S	U	T	CON	R	C3	AC3	AC2	AC1	ACO	TL	SEGTL	RO	ILB	ILU	NA	BL
X	X	L	X	X	X	X	X	X	X	X	X	X	X	X	X	H
X	X	H	X	X	X	X	X	X	X	X	X	X	X	X	X	H
X	X	H	X	X	X	L	X	X	X	X	X	X	X	X	X	H
X	X	H	X	X	X	H	X	X	X	X	X	X	X	X	X	L
X	X	L	X	X	H	X	X	X	H	H	X	X	X	X	X	X
X	X	H	X	X	L	X	X	X	H	H	X	X	X	X	X	X
X	X	H	X	X	H	X	X	X	X	L	H	X	X	X	X	X
X	X	H	X	X	H	X	X	X	X	H	L	X	X	X	X	X
X	X	L	X	L	H	X	L	X	X	X	X	H	X	X	X	X
X	X	H	X	H	H	X	L	X	X	X	X	H	X	X	X	X
X	X	L	X	L	L	X	L	X	X	X	X	H	X	X	X	X
X	X	L	X	L	H	X	H	X	X	X	X	H	X	X	X	X
X	X	L	X	L	H	X	L	X	X	X	X	L	X	X	X	X
X	X	L	H	X	L	H	X	X	X	X	X	H	X	X	X	X
X	X	H	H	X	L	H	X	X	X	X	X	H	X	X	X	X
X	X	H	X	H	L	H	X	X	X	X	X	H	X	X	X	X
X	X	H	X	H	L	H	X	L	X	X	X	H	X	X	X	X
X	X	H	X	H	L	H	X	H	X	X	X	H	X	X	X	X
X	X	H	X	H	L	H	X	X	H	X	X	H	X	X	X	X
X	X	H	X	H	L	H	X	X	X	L	X	H	X	X	X	X
X	X	H	X	H	L	H	X	X	X	H	X	H	X	X	X	X
X	H	L	X	X	L	X	X	X	X	X	X	H	X	X	X	X
X	H	H	X	X	L	X	X	X	X	X	X	H	X	X	X	X
X	L	H	X	X	H	X	X	X	X	X	X	H	X	X	X	X
X	H	H	X	X	H	X	X	L	X	X	X	H	X	X	X	X
X	H	H	X	X	H	X	X	H	X	X	X	L	X	X	X	X
X	X	H	X	X	H	X	X	X	H	X	X	H	X	X	X	X
X	X	L	H	H	H	X	X	X	H	X	X	H	X	X	X	X
X	X	L	H	H	H	X	X	X	X	L	X	H	X	X	X	X
X	X	H	H	H	H	X	X	X	X	L	X	H	X	X	X	X
X	X	H	H	H	L	X	X	X	X	L	X	H	X	X	X	X
X	X	H	H	H	L	L	X	X	X	X	X	H	X	X	X	X
X	X	L	H	H	L	L	X	X	X	X	X	L	X	X	X	X
X	X	H	L	H	L	L	X	X	X	X	X	L	X	X	X	X

```
XXHHXLLHHLLXXXXLX
XXHHXHLHHHLXXXXLX
XHHLXHLHLHLXXXXHX
XHLLXHLHLHLXXXXLX
XHHLXLLHLHLXXXXLX
XLHLXHLHLHLXXXXLX
XHHLXHLHHHLXXXXLX
XLHLLHLLHHLXXXXHX
XLLLLHLLHHLXXXXLX
XLHLLLLLHHLXXXXLX
XLHLHHLLHHLXXXXLX
XLHLLHLHHHLXXXXLX
XLHLHHHXXXLXXXXHX
XLLLHHHXXXLXXXXLX
XLHLHLHXXXLXXXXLX
XLHLHHLXXXLXXXXLX
```

```
DESCRITION
;BLABLA
```

```
PAL16L8
RECOGNITION OF UNIT NUMBER ON IO BUS. SERIAL NUMBER.
CPAL173: CPU PAL NR.17 VERSION 0. POSITION R8
KAN 821112
/ASB  A33  A32  A31  A30 /RU   C1   A03   A02  GND
 A01 /D0  /D1  /D2  /D3  /D4  /D5  /D6   /UN   VCC

IF (VCC) UN =
     /A33 * /A32 *  A31 *  A30 * ASB * C1 ;UNIT NUMBER 3

IF (RU) D6 =
       /A03 * /A02 * /A01 *  RU
     + /A03 * /A02 *  A01 * /RU
     + /A03 *  A02 * /A01 * /RU
     + /A03 *  A02 *  A01 * /RU
     +  A03 * /A02 * /A01 * /RU
     +  A03 * /A02 *  A01 * /RU
     +  A03 *  A02 * /A01 * /RU

IF (RU) D5 =
       /A03 * /A02 * /A01 *  RU
     + /A03 * /A02 *  A01 * /RU
     + /A03 *  A02 * /A01 * /RU
     + /A03 *  A02 *  A01 * /RU
     +  A03 * /A02 * /A01 * /RU
     +  A03 * /A02 *  A01 * /RU
     +  A03 *  A02 * /A01 * /RU

IF (RU) D4 =
       /A03 * /A02 * /A01 *  RU
     + /A03 * /A02 *  A01 * /RU
     + /A03 *  A02 * /A01 * /RU
     + /A03 *  A02 *  A01 * /RU
     +  A03 * /A02 * /A01 * /RU
     +  A03 * /A02 *  A01 * /RU
     +  A03 *  A02 * /A01 * /RU

IF (RU) D3 =
       /A03 * /A02 * /A01 *  RU
```

```
     + /A03 * /A02 *  A01 * /RU
     + /A03 *  A02 * /A01 * /RU
     + /A03 *  A02 *  A01 * /RU
     +  A03 * /A02 * /A01 * /RU
     +  A03 * /A02 *  A01 * /RU
     +  A03 *  A02 * /A01 * /RU

IF (RU) D2 =
        /A03 * /A02 * /A01 *  RU
     + /A03 * /A02 *  A01 * /RU
     + /A03 *  A02 * /A01 * /RU
     + /A03 *  A02 *  A01 * /RU
     +  A03 * /A02 * /A01 * /RU
     +  A03 * /A02 *  A01 * /RU
     +  A03 *  A02 * /A01 * /RU

IF (RU) D1 =
        /A03 * /A02 * /A01 * /RU
     + /A03 * /A02 *  A01 * /RU
     + /A03 *  A02 * /A01 * /RU
     + /A03 *  A02 *  A01 * /RU
     +  A03 * /A02 * /A01 * /RU
     +  A03 * /A02 *  A01 * /RU
     +  A03 *  A02 * /A01 * /RU

IF (RU) D0 =
        /A03 * /A02 * /A01 * /RU
     + /A03 * /A02 *  A01 * /RU
     + /A03 *  A02 * /A01 * /RU
     + /A03 *  A02 *  A01 * /RU
     +  A03 * /A02 * /A01 * /RU
     +  A03 * /A02 *  A01 * /RU
     +  A03 *  A02 * /A01 * /RU

FUNCTION TABLE
ASB C1 A33 A32 A31 A30 UN RU A03 A02 A01 D6 D5 D4 D3 D2 D1 D0
-------------------------------------------------------------
HHLLHH H LXXX ZZZZZZZ
LHLLHH L LXXX ZZZZZZZ
```

```
HLLLHH L LXXX ZZZZZZZ
HHHLHH L LXXX ZZZZZZZ
HHLHHH L LXXX ZZZZZZZ
HHLLLH L LXXX ZZZZZZZ
HHLLHL L LXXX ZZZZZZZ
HHLLHH H HLLL HHHHHLL
HHLLHH H HLLH LLLLLLL
HHLLHH H HLHL LLLLLLL
HHLLHH H HLHH LLLLLLL
HHLLHH H HHLL LLLLLLL
HHLLHH H HHLH LLLLLLL
HHLLHH H HHHL LLLLLLL
HHLLHH H HHHH LLLLLLL
--------------------
```

```
DESCRIPTION
;BLA BLA
```