Dansk Data Elektronik ApS

Kortfattet beskrivelse af
ID-7000 Debugger/monitor
Anden udgave. juni 1976.

Notat vedr. DEBUG Program til SPC/1 systemet.

Der er udformet en modificeret udgave af ID-7000 DEBUG-program for kørsel på SPC/1 systemet.

SPC/1-DEBUG fylder fra x.EC00 - x.FFFF, og benytter terminalporten på SPC/1-modulet. (Adresserne: x.E2 og x.E3).

På grund af forskelle i port-typer, samt processortype (8080A i ID-7000 og 8085A i SPC/1) har det ikke været praktisk muligt at lave en fuldstændig identisk version af DEBUG-programmet.

I følgende punkter beskrives forskelle mellem ID-7000 og SPC/1 DEBUG:

1. DEBUG CALL-funktion.

I SPC/1 systemet kan DEBUG CALL funktion (DC) realiseres uden brug af ID-7005 Manuel Control modul som i ID-7000 systemet. (ID-7005 modulet kan ikke benyttes i SPC/1). DEBUG CALL realiseres ved at generere et TRAP-interrupt til systemets 8085A-processor ved hjælp af en trykknap forbundet til CPU-modulet. Korrekt funktion af DEBUG CALL kræver at der i TRAP-indhops positionen (x.24) er placeret en JMP F000 instruktion. TRAP-interruptet er det højest prioriterede og kan ikke deaktiveres.

2. Interrupt faciliteter.

CPU-modulet i SPC/1 systemet er forsynet med interruptsystem for prioritering af 3 interne interrupt niveauer for terminalport, printerport samt realtime clock. Læsning/skrivning af maske m.v. for disse interrupt kilder udføres ved hjælp af maskininstruktionerne RIM og SIM.

Der er ikke i DEBUG-programmet indført nye kommandoer for behandling af disse interrupts. I og J kommandoerne behandler interrupt masken på et evt. tilstedeværende ID-7003 interrupt-prioritering/RTC-modul som i ID-7000 systemet.

2. Interrupt faciliteter (fortsat)

Med hensyn til den gamle status for interruptsystemets tilstand (disable/enable) er der, grundet forskelle i materialet, følgende ændring:

Kun ved indhop via DEBUG CALL (TRAP-interrupt) bliver status for interruptsystemet gemt rigtigt. Ved programmeret indhop (CALL F008 og CALL F010) samt ved breakpoints, bliver status altid gemt som "disable". Brugeren kan ændre den stakkede status til "enable" ved at ętstille bit 2 i stakket PSW ved hjælp af en MP kommando. Afstakning af interruptstatus ved 1XG kommandoen sker præcis som i ID-7000-systemet.

Information om evt. "PENDING" interrupt (P) udskrives ikke.

3. Terminalstatus ved udhop.

Ved udhop er terminalportens status altid: 2 stopbit lige paritet, 7 databit, d.v.s. samme status som anvendes ved kørsel med DEBUG-programmet.

NB! Ved kørsel med DEBUG-programmet genereres lige paritet ved output. Programmet tester ikke for paritet ved input.

Indholdsfortegnelse.

0.0 Indledning.

Denne rapport er en kortfattet beskrivelse af den ved Institutter for Datateknik, Danmarks Tekniske Højskole udviklede Debugger/monitor til ID-7000 mikrodatamatsystemet. Rapportens sigte er at introducere debuggeren for den utrænede bruger samt at fungere som opslagsværk for den mere trænede bruger. I de følgende afsnit beskrives debuggerens funktioner og i appendix kan se eksempler på anvendelse af debuggerens faciliteter.

Manualen forudsætter kendskab til registerstrukturen i INTEL's 8080 mikroprocessor og til dennes ordresæt.

Om debuggeren skal det indledningsvis oplyses, at den arbejder i 7 bit ASCII tegnkode, der sendes som 8 bit med lige paritet og modtages som 8 bit, hvor pariteten ignoreres.

1.0 Listning og modifikation af debuggerens registre.

Debuggeren opererer med to logiske 16 bit registre kaldet ADR og DATA. De to registre er elementer i en stak med ADR som bundelement og DATA som topelement. I denne toniveau stak gemmes parametrene til et funktionskald indet kaldet foretages. For at lette beregningen af parametre er debuggeren forsynet med en række bordkalkulatorfunktioner, der udføres på indholdet af ADR og DATA.

1.1 Listning af debuggerens registre.

Ordrenavn	Funktion
;	Listning af ADR på hexadecimal form.
#	Listning af DATA på hexadecimal form.
V	Listning af ADR og DATA på hexadecimal form
Y	Listning af DATA på decimal form. Nulstilling af DATA
Z	Listning af DATA på hexadecimal form og tegn form.
U	Listning af DATA opfattet som et PSW dvs som indholdet af AC og flagene.

1.2 Modifikation af debuggerens registre.

Når en konstant indtastes fra konsollen, indsættes den af debuggeren i øverste niveau i stakken dvs i DATA.

Ved en konstant skal her forstås:

- et hexadecimalt tal bestående af op til fire hexadecimale cifre. Indtastning af et hexadecimalt tal skal altid indledes med indtastning af et decimalt ciffer; hvis det før-

ste ciffer, der ønskes intastet, er A, B, C, D, E eller F indtastes forinden et nul.
- et decimalt tal bestående af I efterfulgt af op til fem decimale cifre. Det decimale talområde rækker fra -32.768 til 32.767.
- en ' efterfulgt af op til to ASCII tegn efterfulgt af '.

Ordrenavn	Funktion
<	ADR tildeles værdien af DATA.
>	DATA tildeles værdien af ADR.
=	DATA og ADR ombyttes.
+	ADR adderes til DATA.
-	ADR subtraheres fra DATA.
,	ADR tælles én op.
.	ADR tælles én ned.
N	DATA sættes til minus DATA.
^	DATA sættes til minus DATA minus én.

2.0 Listning og modifikation af maskinens tilstand.

Ved de normale indhop til debuggeren dvs ved aktivering af Debug-Call knappen på testforpladen eller ved kald fra et program (CALL Foo8) gemmer debuggeren indholdet af CPU'ens registre og flag samt afbrydesystemets status i stakken i arbejdslageret. De i dette afsnit beskrevne funktioner kan anvendes til til at undersøge og ændre de i stakken lagrede registerindhold, således at det er muligt at genstarte det midlertidigt stoppede program med et nyt register indhold.

2.1 Listning af status.

Ordrenavn	Funktion
S	Debuggeren udskriver indholdet af stakpointeren SP, samt det stakkede indhold af HL, DE, BC dobbeltregistrene og returadressen dvs det punkt i det kørende program, hvorfra udførelsen skal fortsættes efter udhop fra debuggeren. Endelig udskrives PSW dvs indholdet af AC og flagene. Flagene udskrives på følgende symbolske form:

{M/P} {Z/NZ} {A/NA} {E/O} {R/D} {C/NC} {Q}

Fem af positionerne angiver tilstanden af de normale INTEL statusflag med de af INTEL benyttede symboler. Herudover angiver R/D om afbrydesystemet er aktiveret (R) eller deaktiveret (D) ved udførelsen af det midlertidigt stoppede program. Q angiver, at den stakkede status ikke er en korrekt statusbyte.

I	Afbrydemasken listes som to hexadecimale cifre, og det angives om afbrydesystemet var aktiveret eller deaktiveret ved indhoppet til debuggeren (D/E).

Ordrenavn	Funktion
P	Indholdet af stakpointeren SP adderes til indholdet af DATA, og resultatet udskrives som et hexadecimalt tal.

2.2 Modifikation af status.

Ordrenavn	Funktion
MH	Dobbeltordet i stakken, der svarer til HL, tildeles værdien af DATA.
MD	Dobbeltordet i stakken, der svarer til DE, tildeles værdien af DATA.
MB	Dobbeltordet i stakken, der svarer til BC, tildeles værdien af DATA.
MP	Dobbeltordet i stakken, der svarer til AC og statusflagene, tildeles værdien af DATA.
MR	Dobbeltordet i stakken, der svarer til returadressen, tildeles værdien af DATA.
Q	SP registeret tildeles værdien af DATA.
J	Indholdet af DATA indlæses i afbrydemaskeregisteret.

Det er kun rimeligt at benytte M-ordrerne, hvis indhoppet til debuggeren er foretaget gennem en indgang, hvor maskinens status gemmes i stakken, eller hvis udhoppet fra debuggeren sker gennem en udgang, hvor registrene retableres udfra stakken inden kontrollen overgives til det midlertidigt standsede program.

3.o Listning og modifikation af lagerindholdet.

I dette afsnit beskrives mulighederne for listning af RAM og PROM lager samt mulighederne for modifikation af RAM lagerindhold. For skrivning i PROM lager henvises til et senere afsnit om dette emne.

Listning af lagerindhold.

Ordrenavn	Funktion
K	Listning af lagerindhold, idet dette opfattes som maskininstruktioner, der listes på symbolsk form med operanderne skrevet som hexadecimale tal. ADR skal udpege den første lagerposition, der ønskes listet. DATA styrer, hvor mange bytes der skal listes efter følgende regler: DATA < ADR medfører listning af én ordre. DATA < ADR+16 medfører listning af 16 bytes. DATA ≥ ADR+16 medfører listning af 128 bytes.
DEL	Listning af én maskininstruktion på symbolsk form. ADR skal udpege første byte i den instruktion, der ønskes listet. Efter udførelsen af ordren er ADR opdateret, således at listning af den følgende maskininstruktion kan ske blot ved aktivering af DEL-tasten.

Ordrenavn	Funktion
BREAK	Ved aktivering af denne taste i minimum o.5 sek. afbrydes K-og L-listninger.
L	Listning af lagerindhold, idet dobbeltord listes som firecifrede hexadecimale tal og som to ASCII tegn. ADR udpeger den første byte, der listes og DATA udpeger den sidste byte, der listes; begge som absolutte lageradresser.
ESC	Lister ét dobbeltord i lageret som et firecifret hexadecimalt tal og som to ASCII tegn. ADR opdateres ved udførelsen af ordren, således at det følgende dobbeltord i lageret kan listes blot ved aktivering af ESC-tasten

Ved listning på tegnform listes kontroltegnene som blanke.

3.2 Modifikation af RAM lager indhold.

Når debuggeren udfører en funktion, der indebærer skrivning i lageret, testes det om lageret har fået det ønskede indhold. Hvis dette ikke er tilfældet udskrives en fejlmeddelelse. Årsagen til fejlen kan være, at der er forsøgt skrevet i PROM lager eller i skrivebeskyttet RAM lager og endelig, at der ikke er lager tilstede på den position i lageret, hvor der er forsøgt skrevet. Debuggeren bringes ud af fejltilstanden ved aktivering af RETURN-tasten.

Debuggeren kan anvendes i to tilstande til modifikation af lagerindhold.

I normaltilstanden kan de i afsnit 3.1 nævnte listefunktioner udføres, og herudover haves følgende funktioner til rådighed:

Ordrenavn	Funktion
::	Indholdet af DATA lagres i det dobbeltord, der udpeges af ADR dvs i lagerpositionerne med adresserne ADR og ADR+1.
H:	Indholdet af den laveste byte i DATA lagres i det lagerord, der udpeges af ADR.
Ø	DATA tildeles indholdet af det dobbeltord i lageret, der udpeges af ADR.
HØ	Den laveste byte i DATA tildeles indholdet af det ord i lageret, der udpeges af ADR. Den mest betydende byte i DATA nulstilles.

I autotilstanden sørger debuggeren for at opdatere og udskrive adressen på den lagerposition, der skal skrives i, efterhånden som skrivningen i lageret skrider frem. Det er muligt i flæng at indtaste maskininstruktioner på symbolsk form, tekststrenge af fri længde og talkonstanter. Endvidere haves til rådighed en række funktioner til ændring af adressen på den celle, der skal skrives i. Disse funktioner kan benyttes, hvis der ønskes skrevet i lagercellerne i en anden rækkefølge end den strengt sekventielle; f. eks. hvis en indtastningsfejl ønskes rettet.

På følgende side beskrives de funktioner, der er til rådighed i autotilstanden.

Ordrenavn	Funktion
"	Debuggeren bringes i autotilstanden. Indholdet af DATA opfattes ved indhoppet til autotilstanden, som adressen på det ord i lageret, hvor indskrivning ønskes påbegyndt. I autotilstanden kan indtastes maskinordrer på symbolsk form, tegnstrenge af fri længde omgivet af ', og talkonstanter som defineret i afsnit 1.2. Indtastningen af en maskininstruktion og af en talkonstant afsluttes med vognretur (CR), hvorimod indtastningen af en tegnstreng afsluttes med ' (ping). Indtastes en ikke genkendelig maskininstruktion, eller hvis brugeren begår en anden syntaksfejl, udskriver debuggeren en fejlmeddelelse. Debuggeren bringes ud af fejltilstanden ved aktivering af vognretur (CR).
.	ADR tælles én ned.
,	ADR tælles én op.
<	ADR tildeles den umiddelbart forinden indtastede adresse, således at indskrivningen i lageret fortsætter fra denne adresse.
@	ADR tildeles indholdet af det dobbeltord, der udpeges af ADR-2. Denne ordre benyttes til at forfølge et hop i et program, således at listningen/indtastningen kan fortsættes fra hopadressen, uden at denne skal indtastes.
DEL	Lister én maskininstruktion som tidligere beskrevet.
?	Annulerer den indtastede konstant eller instruktion, således at en ny indtastning kan foretages.
-	Bringer debuggeren ud af autotilstanden.

4.o Indlæsning til og udlæsning fra lager.

I dette afsnit beskrives det, hvorledes der kan foretages indlæsning fra strimmellæser og udlæsning til strimmelhuller. Debuggeren kan betjene strimmeludstyret på en Teletype ASR-33 eller DDE's hurtigstrimmellæser og hurtigstrimmelhuller. Det strimmelformat,der benyttes, er komprimeret i forhold til INTEL's, hvilket medfører at indlæsningstiden bliver omtrent halveret. ID-7ooo Utility programmerne indeholder dog en INTEL loader, således at også strimler i INTEL's format kan indlæses.

Ordrenavn	Funktion
R	Debuggeren udfører en strimmelindlæsning eller en checklæsning mellem strimmel og lager afhængig af DATAś værdi. Hvilken strimmellæser, der skal benyttes ved indlæsningen styres ligeledes af DATA's værdi. Bit o, 1 og 2 i DATA styrer indlæsningen; de resterende bit skal være nulstillet. Strimlerne er opdelt i blokke på op til 256 bytes. Blokkene er forsynet med information om, fra hvilken adresse indlæsningen skal påbegyndes. Afhængig af DATA's værdi indlæses én blok, eller de resterende blokke på den strimmel, der er anbragt i læseren.

Ordrenavn Funktion

Af nedenstående tabel fremgår det, hvorledes de tre mindst betydende bit i DATA styrer indlæsningen.

Værdi	bit 2	bit 1	bit o
1	alle blokke	check-læsning	hurtig-læser
o	én blok	ind-læsning	TTY læser

I det følgende beskrives det, hvilke typer af fejl der kan opstå ved indlæsning af strimler, og hvorledes debuggeren reagerer på disse.

Udskrives et 'E' på konsollen betyder det, at debuggeren har følt en time-out fejl. Dette kan skyldes, at det interface den hurtige strimmellæser tilsluttes ikke er tilstede i systemet og at en indlæsning søges foretaget via hurtiglæseren, eller at der ikke er lagt strimmel i den læser, der ønskes benyttet,eller at strimmelen sidder fast.

Udskrives et ':' på konsollen betyder det, at debuggeren har konstateret en skrivefejl dvs at den ved kontrollæsningen af lageret, der efterfølger enhver af debuggerens skrivninger i lageret, har konstateret, at dette ikke har fået det ønskede indhold. Denne fejl kan skyldes, at indlæsningen søges foretaget til en ikke skrivbar del af lageret, eller at der ikke er lager tilstede i det område, hvortil indlæsningen søges foretaget.

Ved konstatering af andre fejl udskriver debuggeren tre firecifrede hexadecimale tal. Det første, der står på en linie for sig, angiver ved at indeholde FF i den mest betydende byte, at debuggeren har fanget en fejl. De to hexadecimale tal,der udskrives på den følgende linie er indholdet af ADR efterfulgt af indholdet af DATA. ADR indeholder lageradressen, hvor fejlen er blevet konstateret; dette er dog kun tilfældet, hvis blokheaderen er blevet læst. Yderligere information om fejlen fås af indholdet af DATA. Følgende fejlmeldinger er mulige:

Mestbetydende byte af DATA	Mindstbetydende byte af DATA
o	Angiver at debuggeren har konstateret en difference mellem den checksum strimmelen er forsynet med, og den checksum debuggeren har udregnet. Mindstbetydende byte af DATA indeholder den aktuelle sum.
1	Angiver at det første tegn efter leader ikke er startmærket (E7). Mindstbetydende byte indeholder det læste starttegn.
2	Angiver sammenligningsfejl dvs, at debuggeren ikke formår at indskrive det læste tegn i lageret. Mindstbe-

Ordrenavn	Funktion	
		tydende byte af DATA indeholder strimmeltegnet. Denne fejl kan kun forekomme ved checklæsning.
	3	Angiver statusfejl på læser. Mindstbetydende byte af DATA indeholder fejlbits.
	4	Angiver time-out. Fejlen opstår f. eks hvis en strimmel er knækket midt i en blok.
W		Debuggeren huller ved brug af hulleren på TTY ASR-33 en del af maskinens lager i blokke på 256 bytes. Hvilken del af lageret, der hulles ud afhænger af indholdet af ADR og DATA. ADR angiver adressen på den første byte i lageret, der ønskes hullet ud, og DATA angiver adressen på den sidste byte, der ønskes hullet ud. Det er muligt i én ordre enten at få udhullet et afsnit af en blok eller et helt antal blokke. Ved en blok forstås en del af lageret dækkende det hexadecimale adresserum xxoo til xxFF. Efter en W-kommando bliver debuggeren først klar efter indtastning af 'BREAK'. Dette giver brugeren mulighed for at slå TTY terminalen i local, og her ved brug af tastaturet at hulle den afsluttende trailer. Efter indtastning af 'BREAK' er debuggeren klar til at udføre en ny ordre.
OW		Debuggeren huller ved brug af hurtighulleren en del af maskinens lager i blokke på 256 bytes. Hvilken del af lageret, der skal udhulles, styres som ved W-kommandoen. Efter udhulningen af den sidste blok laver brugeren en trailer ved aktivering af 'FEED'-tasten på hurtighulleren.

5.o PROM-skrivning.

Det er nødvendigt,for at et ID-7ooo system kan benyttes til PROM-programmering, at det er forsynet med et ID-7oll PROM-skriver modul.

Inden en PROM-programmering kan sættes igang, skal det, der ønskes programmeret ind i PROM lager, anbringes i maskinens arbejdslager. For at PROM-programmering kan ske over hele adresseområdet, er debuggeren udstyret med en ordre til relokeret indlæsning af strimler. Relokeret indlæsning iværksættes med et 'R', og indholdet af DATA styrer indlæsningen, men udover de i afsnit 4.o beskrevne styrekoder skal DATA også indeholde oplysning om, at der ønskes foretaget en relokeret indlæsning, og med hvilken relokeringskonstant denne indlæsning ønskes foretaget.

Relokeringskonstanten er det antal gange, der skal adderes hex 1oo (decimalt 256) til strimmeladressen for at den ønskede indlæseadresse fremkommer. Additionen skal foretages modulo hex 1oooo (decimalt 64k). Endvidere skal det gælde, at relokeringen skal være et multiplum af hex 8oo (decimalt 2k), dvs at relokeringskonstanten skrevet hexadecimalt skal have et mindst betydende ciffer, der er enten o eller 8.

Eksempel på beregning af relokeringskonstant:
En strimmel assembleret til indlæsning i adressen f000 ønskes indlæst i 1000. Da f000+2000 taget modulo 10000 er lig 1000, er relokeringskonstanten i dette eksempel 20, da der netop skal adderes 20 gange 100 til strimmeladressen, for at den ønskede indlæseadresse fremkommer. Det ses at relokeringskonstantens mindst betydende ciffer er nul, hvorfor kravet om at relokeringen skal være et multiplum af 800 er opfyldt. Alle talkonstanter i eksemplet er hexadecimale.

Indholdet af DATA, der styrer indlæsningen, kan udregnes, når relokeringskonstanten er beregnet. Den mest betydende byte af DATA skal indeholde FF til markering af, at der ønskes foretaget en relokering. Den mindst betydende byte af DATA skal indeholde relokeringskonstanten plus styrekoden, der som beskrevet i forrige afsnit afhænger af, hvilken læser der ønskes benyttet, og hvorledes strimmelindlæsningen skal ske. Når DATA har fået det rette indhold iværksættes indlæsningen som tidligere beskrevet ved aktivering af 'R'.

Programmeringen af en PROM iværksættes med ordren 'OS'. ADR angiver startadressen for det areal i maskinens arbejdslager, der ønskes indlæst i PROM'men, og arealet har samme længde som PROM arealet (max 1k byte), der specificeres i DATA. I nedenstående tabel kan det ses, hvilken betydning de enkelte hexadecimale cifre i DATA har ved en PROM programmering. En nærmere forklaring følger efter tabellen.

DATA opfattet som fire hexadecimale cifre

Mest betydende — Mindst betydende

Angiver øverste blok af PROM (0-3), der skal programmeres.	Angiver nederste blok af PROM (0-3), der skal programmeres.	0	0 Sammenligning og tilbagelæsn. 1 Checklæsning. 2 Hvis PROM tom, så PROM'ning. 3 Hvis programmerbar, så PROM'ning

Sammenligning. Indholdet af PROM'men sammenlignes med lagerindholdet. Udførelsen af sammenligningen afsluttes med udskrivningen af tre firecifrede hexadecimale talkonstanter. Den første, der står på en linie for sig er 0, hvis PROM og lager har samme indhold. Hvis en uoverensstemmelse er fundet, indeholder mest betydende byte af den første hexadecimale konstant en fejlkode, disse beskrives i det følgende, og den anden hexadecimale konstant indeholder fejladressen i lageret og den tredje hexadecimale konstant angiver fejladressen i PROM'men.

Tilbagelæsning. Adderes fire til forreste ciffer i DATA foretager debuggeren i stedet for en sammenligning en tilbagelæsning af PROM indholdet til lageret. Dette benyttes, når der skal laves tilføjelser til en delvist programmeret PROM.

Checklæsning. Indholdet af PROM'men checklæses mod lageret, idet det undersøges om PROM'men er tom, blot har et galt indhold eller er umulig at programmere med det ønskede indhold uden en forudgående sletning.

Hvis PROM tom, så PROM'ning. Det undersøges om PROM'men er tom, dvs om den indeholder FF i alle lagerpositioner, og hvis dette er tilfældet udføres den ønskede programmering. Under PROM'ningen udskrives for hvert ottende gennemløb af skrive-og scanne-sekvensen et '-', hvis PROM'men ikke havde det rette indhold ved sidste scanning, og et '+', hvis PROM'men havde det rette indhold. Der foretages mindst 112 skrivninger inden programmeringen standes, men det kræves af debuggeren, at der har været mindst 9o scan, der har vist at PROM'men har det rigtige indhold. PROM'ningen stoppes under alle omstændigheder efter maksimalt 224 skrivninger. Programmeringen kan afbrydes af brugeren ved aktivering af 'BREAK'-tasten.

Hvis PROM programmerbar, så PROM'ning. PROM'men checklæses mod lageret, og hvis det konstateres, at der ingen steder i PROM'men er 'o', hvor der ønskes PROM'met et '1', erklæres PROM'men for programmerbar og programmeringen iværksættes. PROM-ningen forløber som hvis PROM'men var tom.

Ved udhop fra checklæsning og de to typer PROM'ning udskriver debuggeren tre firecifrede hexadecimale talkonstanter. Bittene i den mest betydende byte af den første hexadecimale talkonstant har følgende betydning, hvis de er '1':

bit nummer	betydning
4	PROM'men er ikke godt programmeret; der har været for få gode gennemløb.
3	PROM'men kan ikke programmeres til det ønskede indhold uden en forudgående sletning.
2	PROM'men er ikke tom.
1	PROM'men har ikke det rigtige indhold.
o	Programmeringen er afbrudt ved aktivering af 'BREAK'-tasten.

De sidste to hexadecimale talkonstanter, der udskrives ved udhop, er ADR henholdsvis DATA. Efter en vellykket PROM'ning (ses af at første byte i første hexadecimale talkonstant er o) indeholder mindstbetydende byte af første hexadecimale talkonstant antallet afskrivninger, der er udført, og mindstbetydende byte af DATA indeholder antallet af korrekte scan.

6.o Indhop til og udhop fra debuggeren.

Indhop til debuggeren kan ske manuelt (ved aktivering af Debug-call 'DC' knappen på testforpladen, programstyret eller fra et breakpoint.

Ved aktivering af Debug-call knappen tvinges maskinen til at udføre debuggeren, såsnart den maskinordre, der er under udførelse, er afsluttet. Denne indgang i debuggeren sørger som det første for at gemme maskinens status i stakken. Når dette er sket udskrives et '>' på konsollen.

Ved programstyret aktivering af debuggeren forårsaget af en CALL Foo8 maskinordre gemmes maskinens status ligeledes, og debuggeren svarer ved at udskrive et '>'.

Når debuggeren aktiveres gennem et breakpoint gemmes maskinens status, og der udskrives et '>' på konsollen.

Efter udskriften af '>' har brugeren to muligheder. Ved aktivering af 'CR' udskriver debuggeren en række statusmeddelelser; aktiveres istedet 'ESC' overspringes denne udskrift. Nedenfor beskrives udskriften.

Udskriften indeholder information om status for konsollen på indhopstidspunktet skrevet som to hexadecimale cifre. Herefter udskrives fire hexadecimale cifre, der alle er nul, undtagen når debuggeren under udførelsen af indhopsekvensen har konstateret, at konsolinterfacekortet har modtaget et tegn fra konsollen. I dette tilfælde angiver de to første hexadecimale cifre den ny status for konsollen, de to sidste hexadecimale cifre er det modtagne tegn.

Herefter udskrives et tegn til oplysning om på hvilken måde, indhoppet til debuggeren er sket. Udskrives 'M' er indhoppet til debuggeren sket ved aktivering af Debug-call knappen. Er debuggeren blevet aktiveret fra et program med kaldet 'CALL Foo8' udskrives et 'P'. Endelig udskrives, hvis indhoppet til debuggeren er sket gennem et breakpoint,et 'B' efterfulgt af den hexadecimale adresse på det til breakpointet knyttede lagerareal (adressen på dette er breakpointets identifikation - se afsnit 6.1), og til sidst udskrives som to hexadecimale cifre det antal gange programmet har passeret gennem det pågældende breakpoint siden sidste indhop til debuggeren.

Efter oplysning om indhopårsagen udskrives 'R' efterfulgt af den hexadecimale adresse på den maskinordre, der skulle udføres, da indhoppet til debuggeren blev foretaget.

Endelig udskrives information om afbrydesystemets status på indhoptidspunktet. Der udskrives 'E' som tegn på,at afbrydesystemet var enabled og 'D' som tegn på, at afbrydesystemet var disabled. Hvis der på indhoptidspunktet var nogle ikke behandlede afbrydelser, udskrives et 'P'. Til sidst udskrives afbrydemasken som to hexadecimale cifre.

6.1 Oprettelse og nedlæggelse af breakpoints.

Til ethvert breakpoint skal knyttes et areal i lageret, hvori debuggeren kan lagre de til breakpointet knyttede data. Et breakpointareal består af 16 byte RAM lager, hvor det om adressen på den første byte i området skal gælde, at den er et multiplum af hex lo. Som en sikkerhed er det indført, at debuggeren kun godtager et areal som breakpointareal, hvis det første lagerord i arealet indeholder 8.

Ordrenavn	Funktion
(	Der indsættes et breakpoint på det sted i lageret, der udpeges af DATA, og til dette knyttes breakpointarealet, der udpeges af ADR. Både DATA og ADR skal udpege skrivbart lager. Når ordren er udført, udskrives det på konsollen, hvilke maskininstruktioner debuggeren har flyttet over i breakpointarealet.
)	Breakpointet svarende til det breakpointareal, der udpeges af ADR nedlægges.

Ordrenavn	Funktion
&	Breakpointet svarende til det breakpointareal, der udpeges af ADR, gøres multibelt, dvs at debuggeren først stopper programmet under udførelse, når dette har passeret breakpointet et specificeret antal gange. Antallet af gange breakpointet skal passeres,inden programmet stoppes, angives i DATA. Ønskes multipliciteten af et breakpoint ændret udføres en ny '&'-ordre.

6.2 Udhop fra debuggeren.

Ordrenavn	Funktion
oXJ	Debuggeren udfører et hop til et brugerprogram med afbrydesystemet disabled. ADR skal indeholde den adresse, der ønskes hoppet til.
1XJ	Debuggeren udfører et hop til et brugerprogram med afbrydesystemet enabled. ADR skal indeholde hopadressen.
oXG	Debuggeren afstakker den i stakken lagrede status, og det midlertidigt stoppede brugerprogram genstartes med den i stakken lagrede status, dog er afbrydesystemet altid disabled ved genstartningen.
1XG	Debuggeren afstakker den i stakken lagrede status, og det midlertidigt stoppede brugerprogram genstartes med den i stakken lagrede status herunder status for afbrydesystemet (enabled/disabled).

APPENDIX 1: EKSEMPLER PÅ ANVENDELSE AF ID-7000 DEBUGGER/MONITOR.

1. Modifikation og listning af DEBUGGERENS registre DATA og ADR.

12A5	:	indlæser det hexadecimale tal x.12A5 i DATA.
T12453	:	indlæser det decimale tal 12453 i DATA.
T12453N	:	indlæser det decimale tal ÷12453 i DATA idet 2's komplement repræsentation benyttes for negative tal.
'DQ'	:	indlæser ASCII-værdierne for tegnene D og Q i DATA. (D i loworder DATA og Q i highorder DATA).
12A5<	:	indlæser det hexadecimale tal x.12A5 i DATA og ADR.
ABCD<1234+	:	udregner summen af de to hexadecimale tal med resultatet i DATA.
ABCDY	:	Lister det hexadecimale tal x.ABCD på decimal form.
T1234#	:	Lister det decimale tal 1234 på hexadecimal form.

2. Modifikation og listning af maskinens tilstand.

AB12MH : indlæser det hexadecimale tal x.AB12 i maskinens stak, hvor maskinens H og L registre er gemt ved indhop. Efter udhop vil H-registeret indeholde x.AB og L-registeret indeholde x.12

100MR : indlæser det hexadecimale tal x.100 i maskinens stak, hvor returadressen fra indhoppet er gemt. Ved udhop vil maskinen udføre instruktioner fra adressen x.100.

800Q : Stackpointer registeret SP i mikroprocessoren tildeles indholdet x.800.

42J : Afbrydemaskeregisteret (på ID-7003 Interruptprioriteringsmodulet) tildeles værdien x.42.

OP : Lister indholdet i stackpointer registeret SP.

3. Modifikation og listning af lagerindhold:

100<1234::	:	indlæser det hexadecimale tal x.1234 i den dobbelte lagerposition x.100 og x.101. (x.100 tildeles værdien x.34 mens x.101 tildeles værdien x.12).
100<T1234::	:	indlæser det decimale tal 1234 i den dobbelte lagerposition x.100 og x.101.
100<12H:	:	indlæser det hexadecimale tal x.12 i lagerpositionen x.100.
100"	:	Bringer DEBUGGEREN i automode med start-adresse x.100.
@ 0100 LXI H,A12	:	instruktionen indsættes.
@ 0103 JMP 10	:	instruktionen indsættes.
@ 0106 T100	:	decimaltallet 100 indsættes som dobb. ord.
@ 0108 0ABCD	:	x.ABCD indsættes som dobb. ord.
@ 010A 'TEXT'	:	tekststrengen indsættes i lager fra x.010A.
@ 010E -	:	automode forlades.
R		
100<K	:	Lister 16 bytes på ordreform fra lager-adressen udpeget ved ADR.
100<110L	:	Lister dobbeltord på hex. form og tegnform fra x.100 til og med x.111 i lager.

4. Indlæsning til og udlæsning fra lager.

0R	:	indlæser 1 blok fra TTY læser.
1R	:	indlæser 1 blok fra hurtig læser.
2R	:	checklæser 1 blok fra TTY læser.
3R	:	checklæser 1 blok fra hurtig læser.
4R	:	indlæser alle blokke fra TTY læser.
5R	:	indlæser alle blokke fra hurtig læser.
6R	:	checklæser alle blokke fra TTY læser.
7R	:	checklæser alle blokke fra hurtig læser.
110<127W	:	huller fra x.110 til x.127 (incl.) på TTY huller.
110<127ØW	:	huller fra x.110 til x. 127 (incl.) på hurtig huller.
0<7FFW	:	huller fra 0 til x.7FF (incl.) på TTY huller i 8 blokke.

x) Bogstavet 0, ikke nul.

5. PROM-skrivning.

0<30020$S	:	Overfører indholdet i lager fra x.0 til x.3FF til blokkene 0, 1, 2 og 3 i PROM (hele PROM'en). Det checkes forud at PROM'en er tom.
0<11020$S	:	Overfører indholdet i lager fra x.0 til x.FF til blok 1 i PROM'en.
400<70000$S	:	Tilbagelæser indhold i hel PROM til RAM-lager fra adresse x.400 og fremefter.
400<30000$S	:	Checklæser indhold i hel PROM mod lager fra x.400.

x) Bogstavet O, ikke nul.

6. Indhop til og udhop fra DEBUGGER.

800<8H:

800<10(	:	Opretter breakpoint i adresse x.10 med breakpointareal i x.800 - x.80F.
800<T25&	:	Ændrer breakpoint, med breakpointareal i x.800 til multipelt breakpoint med 25 (decimalt) ganges gennemløb før udhop.
800<)	:	Nedlægger breakpoint med breakpointareal i x.800. Lokation 800 indeholder 8 efter nedlæggelsen.
100<0XJ	:	Starter programudførelse i x.100 med udefineret registerindhold. Afbrydesystemet er deaktiveret efter udhop.
100<1XJ	:	Starter programudførelse i x.100 med udefineret registerindhold. Afbrydesystem er aktiveret efter udhop.
0XG	:	Fortsætter programudførelse med stakket status og afbrydesystem deaktiveret.
1XG:	:	Fortsætter programudførelse med stakket status, herunder status for afbrydesystem.