

==== WARNING ====

THE MODULES BETWEEN THESE COMMENT LINES MAY GET OVERLAYED
WHEN USING THE MAPPER REGISTERS 2 & 3 (>2000->3FFF)

```
INCLUDE ADHOC.OBJ.RUN
INCLUDE ADHOC.OBJ.CVBD
INCLUDE ADHOC.OBJ.CVDB
INCLUDE ADHOC.OBJ.CVGC
INCLUDE ADHOC.OBJ.EVAL
INCLUDE ADHOC.OBJ.EXPF
INCLUDE ADHOC.OBJ.FIX
INCLUDE ADHOC.OBJ.FP
INCLUDE ADHOC.OBJ.BITF
INCLUDE ADHOC.OBJ.GETLINE
INCLUDE ADHOC.OBJ.GETP2
INCLUDE ADHOC.OBJ.GETPARM
INCLUDE ADHOC.OBJ.LOGF
INCLUDE ADHOC.OBJ.PLOT
INCLUDE ADHOC.OBJ.POLY
INCLUDE ADHOC.OBJ.POWF
INCLUDE ADHOC.OBJ.RNDF
INCLUDE ADHOC.OBJ.SINF
INCLUDE ADHOC.OBJ.SGRF
INCLUDE ADHOC.OBJ.EXTEND
INCLUDE ADHOC.OBJ.TEXT
INCLUDE ADHOC.OBJ.LAST
END
```


SDSLNK
LINK MAP

3.4.0 51.117

12/23/82 15:39:11

PAGE 3

CONTROL FILE = ADHOC.LINK.CNTRL

LINKED OUTPUT FILE = ADHOC.LINK.OBJ

LIST FILE = ADHOC.LINK.LINKMAP

OUTPUT FORMAT = COMPRESSED

PHASE 0, ADHOC11 ORIGIN = 0000 LENGTH = 60FO

MODULE	NO	ORIGIN	LENGTH	TYPE	DATE	TIME	CREATOR
START	1	0000	0278	INCLUDE	12/23/82	15:30:53	SDSMAC
MID	2	0278	038A	INCLUDE	12/23/82	11:11:21	SDSMAC
DSR02	3	0602	0152	INCLUDE	12/22/82	15:57:03	SDSMAC
CENTRON	4	0754	001A	INCLUDE	12/22/82	15:52:03	SDSMAC
VDPDSR	5	076E	0298	INCLUDE	12/22/82	16:42:01	SDSMAC
MONITOR	6	0A06	0BEC	INCLUDE	12/22/82	16:20:07	SDSMAC
JMP	7	15F2	001E	INCLUDE	12/22/82	16:14:39	SDSMAC
CASSETTE	8	1610	02A4	INCLUDE	12/22/82	15:50:14	SDSMAC
ATNF	9	18B4	00DE	INCLUDE	12/22/82	15:46:59	SDSMAC
VDP	10	1992	03B8	INCLUDE	12/22/82	16:40:33	SDSMAC
ERROR	11	1D4A	01D4	INCLUDE	12/22/82	16:01:57	SDSMAC
CALL	12	1F1E	002C	INCLUDE	12/22/82	15:49:23	SDSMAC
CHRF	13	1F4A	003C	INCLUDE	12/22/82	15:52:47	SDSMAC
CRUF	14	1FB6	0088	INCLUDE	12/22/82	15:53:13	SDSMAC
DEF	15	200E	0024	INCLUDE	12/22/82	15:56:02	SDSMAC
DIM	16	2032	00C4	INCLUDE	12/22/82	15:56:26	SDSMAC
ENTER	17	20F6	0042	INCLUDE	12/22/82	16:01:38	SDSMAC
ESCAPE	18	2138	000E	INCLUDE	12/22/82	16:03:08	SDSMAC
FOR	19	2146	019A	INCLUDE	12/22/82	16:07:34	SDSMAC
FORMAT	20	22E0	013C	INCLUDE	12/22/82	16:08:13	SDSMAC
FUNC	21	241C	00C6	INCLUDE	12/23/82	15:31:44	SDSMAC
GOSUB	22	24E2	00CC	INCLUDE	12/22/82	16:11:38	SDSMAC
IF	23	25AE	0080	INCLUDE	12/22/82	16:12:31	SDSMAC
INPUT	24	262E	0108	INCLUDE	12/22/82	16:13:22	SDSMAC
IPCOM	25	2736	003C	INCLUDE	12/22/82	16:13:47	SDSMAC
LET	26	2772	0146	INCLUDE	12/22/82	16:16:18	SDSMAC
LOGOPS	27	28B8	0050	INCLUDE	12/22/82	16:17:11	SDSMAC
MEMF	28	2908	0032	INCLUDE	12/22/82	16:17:42	SDSMAC
MOVE	29	293A	0064	INCLUDE	12/22/82	16:24:24	SDSMAC
MWDF	30	299E	002E	INCLUDE	12/22/82	16:24:51	SDSMAC
NUMBER	31	29CC	002E	INCLUDE	12/22/82	16:25:12	SDSMAC
ON	32	29FA	0032	INCLUDE	12/22/82	16:25:33	SDSMAC
PRINT	33	2A2C	01D0	INCLUDE	12/22/82	16:27:36	SDSMAC
PURGE	34	2BFC	0054	INCLUDE	12/22/82	16:28:07	SDSMAC
PUTB	35	2C50	004E	INCLUDE	12/22/82	16:28:37	SDSMAC
RANDOM	36	2C9E	0008	INCLUDE	12/22/82	16:28:58	SDSMAC
READ	37	2CA6	00B2	INCLUDE	12/22/82	16:29:15	SDSMAC
RENUM	38	2D58	02E0	INCLUDE	12/22/82	16:29:56	SDSMAC
SGNF	39	3038	0018	INCLUDE	12/22/82	16:31:48	SDSMAC
SIZE	40	3050	00BC	INCLUDE	12/22/82	16:32:24	SDSMAC
TICF	41	310C	0030	INCLUDE	12/22/82	16:38:41	SDSMAC
TIME	42	313C	009A	INCLUDE	12/22/82	16:39:05	SDSMAC
TRACE	43	31D6	000E	INCLUDE	12/22/82	16:39:37	SDSMAC
UNIT	44	31E4	0036	INCLUDE	12/22/82	16:40:05	SDSMAC
BOOT	45	321A	0119	INCLUDE	12/22/82	15:48:21	SDSMAC
SFSN	46	3334	0014	INCLUDE	12/22/82	16:31:26	SDSMAC
SWAP	47	3348	0064	INCLUDE	12/22/82	16:35:57	SDSMAC
RELOPS	48	33AC	003C	INCLUDE	12/22/82	16:29:38	SDSMAC
EDIT	49	33E8	0AE6	INCLUDE	12/22/82	15:58:00	SDSMAC
RUN	50	3ECE	0196	INCLUDE	12/22/82	16:30:51	SDSMAC
CVBD	51	4064	020C	INCLUDE	12/22/82	15:53:46	SDSMAC
CVDB	52	4270	01E8	INCLUDE	12/22/82	15:54:45	SDSMAC
CVGC	53	4458	005E	INCLUDE	12/22/82	15:55:38	SDSMAC

SDSLNK MODULE	NO	3.4.0 ORIGIN	81.117 LENGTH	12/23/82 TYPE	15:39:11 DATE	TIME	PAGE 5 CREATOR
EVAL	54	44B6	056E	INCLUDE	12/22/82	16:03:41	SDSMAC
EXPF	55	4A24	00C8	INCLUDE	12/22/82	16:05:34	SDSMAC
FIX	56	4AEC	003C	INCLUDE	12/22/82	16:07:05	SDSMAC
FP	57	4B28	0342	INCLUDE	12/22/82	16:08:41	SDSMAC
BITF	58	4E6A	0054	INCLUDE	12/22/82	15:47:38	SDSMAC
GETLINE	59	4EBE	015E	INCLUDE	12/22/82	16:10:33	SDSMAC
GETP2	60	501C	001A	INCLUDE	12/22/82	16:10:57	SDSMAC
GETPARM	61	5036	001E	INCLUDE	12/22/82	16:11:12	SDSMAC
LOGF	62	5054	00EA	INCLUDE	12/22/82	16:16:41	SDSMAC
PLOT	63	513E	014E	INCLUDE	12/22/82	16:26:17	SDSMAC
POLY	64	528C	0068	INCLUDE	12/22/82	16:26:46	SDSMAC
POWF	65	52F4	0022	INCLUDE	12/22/82	16:27:16	SDSMAC
RNDF	66	5316	0030	INCLUDE	12/22/82	16:30:39	SDSMAC
SINF	67	5346	00A0	INCLUDE	12/22/82	16:32:05	SDSMAC
SQRF	68	53E6	0070	INCLUDE	12/22/82	16:32:49	SDSMAC
EXTEND	69	5456	00C4	INCLUDE	12/22/82	16:05:54	SDSMAC
TEXT	70	551A	05D4	INCLUDE	12/23/82	15:38:36	SDSMAC
LAST	71	5AEE	0602	INCLUDE	12/22/82	16:15:18	SDSMAC

DEFINITIONS

NAME	VALUE	NO	NAME	VALUE	NO	NAME	VALUE	NO	NAME	VALUE	NO
ABSF	2466	21	ADDF	4926	54	ADRF	29C8	30	AINC	EFC8*	1
ANDF	33AC	48	ASCF	2936	28	ASKBP	55F2	70	ASMOPC	EF5E*	1
ASMPTR	EF16*	1	ATNF	18B4	9	AUTORN	55A3	70	B00	5571	70
B01	48BF	54	B05	48C3	54	B08	4F8E	59	B0D	5548	70
B1B	5541	70	B20	4F8A	59	B2A	55B8	70	B2D	55A9	70
B2E	555A	70	B30	40E3	51	B31	5540	70	B3A	55BF	70
B3C	5543	70	B3F	559B	70	B40	5611	70	B43	01C8	1
B45	5528	70	B47	5534	70	B4A	5544	70	B4C	553F	70
B56	5545	70	B62	5546	70	B7F	5531	70	BADADR	563C	70
BADTER	562A	70	BASY	1F86	14	BAUD	0164	1	BCRU	001A	1
BEGN	024A	1	BEGN1	0252	1	BELCNT	ED42	1	BELL	55C2	70
*BF1	5547	70	BFA	5542	70	*BFF	5532	70	BITF	4E9E	58
BITMAP	EE3C*	1	BITY	4E6A	58	BLSTOR	EF14*	1	BMVE	293A	29
BOOTY	321A	45	BPMSG	55CF	70	BPTOV	EF1A*	1	BPTADD	EF1E*	1
BPTDTA	EF3E*	1	BPTSET	EF1C*	1	BRAM	60F0	71	BUFEND	EE32*	1
BUS	ED04*	1	C0004	5520	70	C000A	29F6	31	C1	48BE	54
C10	064C	3	C2000	4F8A	59	C4	48C0	54	C4600	551E	70
C4A00	551C	70	C6	48C8	54	C7F	5530	70	C8000	4E8C	58
CALL12	ED5E*	1	CALLWP	ED46*	1	CASRDY	558A	70	CCNT	ED68*	1
CCSAVE	EE94*	1	CENTRO	0754	4	CFO	552E	70	CFF80	5532	70
CKEX	44F4	54	CLKADR	ED16*	1	CLKT01	ED1C*	1	CLKT02	ED1E*	1
CLKWS	ED14	1	CLLY	1F1E	12	CLRV	3ECE	50	CMOF*	0015*	2
CMON*	0014*	2	CNTDWN	ED38*	1	COLF	1B14	10	COLORY	1AD8	10
CONFIG	00B4	1	COSF	5346	67	CPYST	572C	70	CRASH*	2008	14
CRBF	1FEE	14	CRBY	1FA6	14	CRDELY	5524	70	CRFF	1FC4	14
CRFY	1F8C	14	CRLF	022A	1	*CRLF1	0244	1	CRLF1T	5586	70
CRLF2T	5586	70	CURFLG	ED6A*	1	CVBD	4070	51	CVBF	22E0	20
CVBFR	41D4	51	CVBI	4068	51	CVC10	4480	53	CVCH	EFEC*	1
CVDB20	43E4	52	CVDIFZ	4274	52	CVDIZ	4270	52	CVGCN	4458	53
CVGCN1	4460	53	CVHD	EFF2*	1	CVHDO1	EFF3*	1	CVHD12	EFFE*	1
CVHD15	F001*	1	D*LDPC	0A1E	6	D*LDWP	0A08	6	D*RTWP	0720	3
D10	29F6	31	D100	29EE	31	*D40	09C8	5	DATXB	4063	50
DCNT	EFCA*	1	DDM	EFD4*	1	DEBUG*	0AF8	6	DEFY	200E	15

SDSLNK	3. 4. 0	81. 117	2/23/82	15: 39: 11	PAGE	6	
NAME	VALUE NO	NAME	VALUE NO	NAME	VALUE NO	NAME	VALUE NO
DELAY*	0008* 2	DEVTBL	0084 1	DH*ARF	EFA6* 1	DH*END	EFBA* 1
DH*HCS	EFB8* 1	*DH*PGN	EFAB* 1	DH*PLL	EFB6* 1	DH*SI2	EFB4* 1
DH*SLT	EFB0* 1	DH*VNT	EFB2* 1	DIMY	2032 16	DLC	EFD2* 1
DLIM	EFD0* 1	DMYHDR	EFA6* 1	DS	ED20* 1	DS1	ED26* 1
DS2	EFDA 1	*DS3	EFE0* 1	DSRCNT	EDAA* 1	DTEND	00A4 1
E*TEMP	EFE6* 1	EBP	EFF2* 1	ECHOEN	OE30 6	EDER	4F9A 59
EDIT	3404 49	EDIT1	340A 49	EDTMP	ECF8* 1	EFIX*E	1AB8 10
EFLG	001C 1	ELNM	0018 1	ELSF	ED6C* 1	EMV	3800 49
ENTERY	20F6 17	ENUM	0016 1	*EORBUS	ED06* 1	EGUSGN	5623 70
EREGS	EF6C* 1	ERROOT	572C 70	ERR01T	573E 70	ERR02T	574A 70
ERR03T	575D 70	ERR04T	5770 70	ERR05T	5785 70	ERR06T	5797 70
ERR07T	57A8 70	ERR08T	57BA 70	ERR09T	57CF 70	ERR10T	57E8 70
ERR11T	57F5 70	ERR12T	5803 70	ERR13T	5812 70	ERR14T	5825 70
ERR15T	583E 70	ERR16T	5854 70	ERR17T	5872 70	ERR18T	5888 70
ERR19T	589A 70	ERR20T	58AD 70	ERR21T	58C6 70	ERR22T	58D9 70
ERR23T	58EB 70	ERR24T	58FB 70	ERR25T	5913 70	ERR26T	5931 70
ERR27T	594B 70	ERR28T	5961 70	ERR29T	5971 70	ERR3*M	0AEB 6
ERR30T	598B 70	ERR31T	5993 70	ERR32T	599F 70	ERR33T	59AA 70
ERR34T	59C2 70	ERR35T	59D2 70	ERR36T	59E1 70	ERR37T	59FC 70
ERR38T	5A0D 70	ERR39T	5A1F 70	ERR40T	5A35 70	ERR41T	5A47 70
ERR42T	5A60 70	ERR43T	5A76 70	ERR44T	5A87 70	ERR45T	5A95 70
ERR46T	5AA7 70	ERR47T	5ABC 70	ERR48T	5AD7 70	ERR49T	563C 70
ERRLS2	1D4A 11	*ERROR	2F80* 1	ERROR2	2FA0* 1	ERRS	1DAA 11
ERRY	1EA4 11	ESCFLG	0020 1	ESCY	2138 18	EUS	ECFC* 1
EVAL	469A 54	EVALS2	4692 54	EVARZ	44F0 54	*EVENRT	30E4 40
EVERZ	450A 54	EVFX	4522 54	EVOP3A	47DC 54	EVSDZ	44B6 54
EVSFR	466C 54	EVSFR*	467C 54	EVSKB	F000* 1	EVSKE	F094* 1
EXMSG	55FC 70	EXPF	4A28 55	EXT*ID	5526 70	EXTNDY	549E 69
EXTWP	EED4* 1	F*WHO	EFD8* 1	FAD	4B42 57	FADDI	4E1A 57
FBCOL	EE95* 1	FCLR	4D4A 57	FDCDON	0028 1	FDD	4D7E 57
FFLG	ED6E* 1	FGM*	0012* 2	FIX	4AEC 56	FLDD	4B30 57
FLOAT	4BFC 57	*FLUSH*	0013* 2	FMD	4C5E 57	FNEG	4BF2 57
FNRM	4C24 57	FNS	ECFE* 1	*FNSZ	000A* 1	FOR2	2186 19
FORV	2146 19	FP5E5	5528 70	FPAC	F09C* 1	FPAC2	F09E* 1
FPAC3	F09F* 1	FPAC4	F0A0* 1	FPWP	F09C* 1	FRAF	249A 21
FSCL	4E42 57	FSD	4B38 57	FSRD	4B28 57	FSUBI	4E2E 57
FTM*	0011* 2	FUNBK	528C 64	FUNFX	52B8 64	FUZZ	3B00* 1
GCLEAR	057E 2	GETC*	000B* 2	GETCR*	000C* 2	GETP2	501C 60
*GDS1	24F6 22	GDS2	24F0 22	GDSB1	24E2 22	GOSON	2518 22
GDSY	2500 22	GOTY	24FC 22	GPRM	5036 61	GPRM1	503C 61
GPRM2	503E 61	GRAPHY	1AB4 10	GSC	EFC4* 1	*GSIM	228C 19
GSS	ED00* 1	*GSSZ	0014* 1	*GTICZ	310C 41	GTLN	4EE4 59
HALTO*	025C 1	*HDSIZ	0014* 1	HFLG	0014 1	HOUT	2BB6 33
HOUT	2BB2 33	*HTXT	2BEC 33	IDRUN	A5A5* 70	IDTEG	5625 70
IFLG	ED70* 1	IFY	25AE 23	ILLMID	ED98* 1	INITFC	EF18* 1
INM1	55BC 70	INM2	55BB 70	INM3	55BF 70	INPEND	2746 25
INPTR	EDAB* 1	INPY	262E 24	INT4PC	0602 3	INTF	2440 21
IOB	ECFA* 1	IOSTOR	EE94* 1	*IREG12	EF90* 1	*IREG6	EF84* 1
IREGS	EF78* 1	JMPRO	15FA 7	JMPROA	15F2 7	LANDF	28E6 27
*LCN1	3BA9 49	*LCN2	3A2C 49	*LCN3	3B14 49	*LCN4	00AC* 49
LDCS	1C3E 10	LDCSY	1AB8 10	*LDEF	0084* 49	LDPY	173A 8
LENF	1F7C 13	LETY	2772 26	LINE	3F36 50	LINEO	3F4A 50
LINE2	3F5E 50	LINE5	3F3C 50	LNOTF	28F9 27	LNSZ	0084* 1
LNUM	EFD6* 1	*LNXT	0010* 49	LOADER	05E0 2	LOADPT	EDA4* 1
LOGF	5056 62	LOGON	55D5 70	LOOK*	000D* 2	LORF	28D4 27
LST	3C0A 49	LSTL	3C80 49	LXORF	28DC 27	MAGY	1BFA 10
MBEGN	5548 70	MC*LOD	171E 8	MC*SAV	1610 8	MCHF	1F6E 13
MCRLF	556F 70	MEMF	2924 28	MEMSIZ	0022 1	MEMY	2908 28
MENTRY	0E56 6	MIDO	ED96* 1	MIDPC	0278 2	MODE	EFCC* 1

SDSLNK	3.4.0	81.117	12/23/82	15:39:11	PAGE	7	
NAME	VALUE NO	NAME	VALUE NO	NAME	VALUE NO	NAME	VALUE NO
MODEOK	33F6 49	MODF	24CB 21	MONTOP	0A06 6	MOTORY	186E 8
MOVEB	30DA 40	MOVEBN	30EC 40	MOVEL	30DE 40	MOVELN	30F0 40
MPRDY	5567 70	*MREG1	EF64* 1	MREG13	EF7C* 1	*MREG14	EF7E* 1
MREG2	EF66 1	MREG3	EF68* 1	*MREG6	EF6E* 1	MREGS	EF62* 1
MWDF	29B8 30	MWDY	299E 30	NEWP	01F4 1	*NEWP1	01F8 1
NEWY	01CE 1	NIC	0064* 1	NINE	2E62 38	NKYF	2482 21
NLIN	3F30 50	NLINO	3F2C 50	NOERR	0024 1	NOEY	213E 18
NOTF	33DD 48	*NRV	0004* 1	NUMY	29CC 31	NVD	EFC0* 1
NVS	EFC2* 1	NXLOC	EE96* 1	NXTXB	4062 50	NXTY	2240 19
NYLOC	EE97* 1	ONY	29FA 32	ORF	33C8 48	OUTBUF	EDE4* 1
OUTDSR	ED9A* 1	OUTL1	2736 25	*OUTR1U	30B4 40	OVINT	ED94* 1
P#EUS	ECF4* 1	P#FNS	EC40* 1	P#QSS	EBF0* 1	P#IOB	EB04* 1
P#UFT	EB84* 1	PADIT	55C4 70	PC	EF7C* 1	PCHTB	5AEE 71
*PCHTE	60EE 71	PFIX	4B18 56	PGMFND	5535 70	PIXOFF	1CA8 10
PIXON	1CB2 10	*PIXTST	1C9E 10	PLC	EFCE* 1	PLF	ED72* 1
*PLOT	51BE 63	PLOTY	5142 63	PLYX	52DE 64	PLYXX	52D2 64
POPY	259C 22	POSF	1F4A 13	POWF	52F4 65	PRAM	ECF6* 1
PRDY	021C 1	PROMPT	5614 70	PRTEND	2742 25	PRTY	2AB8 33
PTRS	ECFA 1	PURGY	2BFC 34	PUTB	2C50 35	R8STOR	ED34* 1
RANDS	ECF6* 1	RANDZ	532E 66	RANY	2C9E 36	RBUF	EDBC* 1
RBUFE	EDE4* 1	RDDY	2CFA 37	RECPT	EDAC* 1	REGSTR	55F0 70
RENCMP	ED0A* 1	RENINC	ED12* 1	RENLE	ED0E* 1	RENRET	EDOC* 1
RENSTA	ED10* 1	RENTY	0E36 6	RENUMS	2D78 38	REST	3EE6 50
RHENTY	0E5C 6	RNELSE	0003* 49	RNERR	0009* 49	RNGDTG	0001* 49
RNGSUB	0002* 49	RNIF	0041* 49	RNINP	000D* 49	RNON	0040* 49
RNREM	0004* 49	RNRSTR	000F* 49	RNWY	2D36 37	RTNY	256C 22
RTSTOR	ED36* 1	RUNP	3EF2 50	*RUNP1	3EF4 50	S#SM	1C1A 10
SAVY	1642 8	SENDAD	05F2 2	SETGRA	0010* 2	SETTXT	000F* 2
SETVEC	012A 1	SFSN	3334 46	SGETY	1992 10	SGNF	3038 39
*SGRA	052C 2	SHAPEY	1B6A 10	SINF	535A 67	SIZE	3050 40
SLN	ED66* 1	SLT	EFBA* 1	SPACE2	55CC 70	SPACE5	55C9 70
*SPACE8	55C6 70	SPBP	5605 70	SPR	55EF 70	SPRITY	1B9E 10
SPUTY	19FC 10	SGRF	53E6 68	*SGRI	0004* 1	SRATE	01B8 1
SSP	F076* 1	STACNT	ED08* 1	START	5656 70	STPM5G	5609 70
STPY	1E3A 11	STPYA	1E20 11	STRTC	5522 70	STXT	04DA 2
SUBF	494E 54	SWAPY	3348 47	SWPTBL	00A4 1	SYMBLC	5611 70
SYSERR	5732 70	SYSF	241C 21	SYSFLG	0014 1	SYSLMT	0013* 1
TAPERR	5572 70	TBEND	5AEE 70	TEMP	F094* 1	TEMP2	F096* 1
TEMP4	F098* 1	TEMP6	F09A* 1	TEXTY	1A7E 10	TICF	3128 41
TIMC	551A 70	TIMY	313C 42	TMPBUF	5648 70	TOFY	31DE 43
TONY	31D6 43	TRAFLG	ED44* 1	TYO	EFC6* 1	TYPO*	0001* 2
TYP11*	0009* 2	TYPB*	0003* 2	TYPBE*	000A* 2	TYPC*	0002* 2
TYPE*	0004* 2	TYPEN*	0005* 2	TYPS*	0006* 2	TYPSN*	0007* 2
UCCNT	275C 25	UFT	ED02 1	UNIT	001E 1	UNPACK	1C20 10
*UNPLOT	51B4 63	UNTY	31E4 44	UPL0TY	513E 63	USERCS	5EE4 71
V1R3L	EE5B* 1	V1R4L	EE5D* 1	V1R5L	EE5F* 1	V1R6L	EE61* 1
VDPDSR	0808 5	VDPST	000B* 1	VDPWP0	EE34* 1	VDPWP1	EE54* 1
VDT	EFBE* 1	VMODE	0026 1	VNT	EFBC* 1	WAIT*	000E* 2
WAITY	1B58 10	WENTRY	0E40 6	WHENTY	0EC8 6	WHXETY	0E8E 6
WORK11	EFOA 1	*WORK5	EEFE* 1	WORKS	EEF4* 1	WP10L	F0DD* 1
WP9928	EE74* 1	WPR1	F0DC* 1	*WPR103	F0E2* 1	*WPR104	F0E4* 1
WPR108	F0EC 1	WPR2	F0BC* 1	WS	5619 70	*XFRPM	5456 69
XLOC	EE36* 1	XOPENT	0DBA 6	XREGS	EF86* 1	YLOC	EE37* 1
YORN	187E 8	ZZZZZ	60EE 71				

**** LINKING COMPLETED

SOURCE ACCESS NAME= ADHOC.SRC.START
OBJECT ACCESS NAME= ADHOC.OBJ.START
LISTING ACCESS NAME= ADHOC.LST.START
ERROR ACCESS NAME=
OPTIONS= BUNLST, TUNLST, DUNLST, MUNLST
MACRO LIBRARY PATHNAME=

LINE	KEY	NAME
------	-----	------

0006	A	ADHOC.SRC.IOBITS =>ADHOC.SRC.IOBITS
------	---	--

0002 IDT 'START'
0003 *
0004 * CORTEX BASIC REVISION 1.1
0005 *
0006 COPY ADHOC.SRC.IOBITS

```

A0002      *
A0003      * THIS MODULE IS INCLUDED IN SOURCE MODULES
A0004      * BY USING A 'COPY' DIRECTIVE.
A0005      *
A0006      *
A0007      *****
A0008      *
A0009      * CRU DEFINITIONS
A0010      *
A0011      *****
A0012      0000 PIO EQU >0000 PARALLEL I/O
A0013      * PARALLEL I/O - READ BITS
A0014      0000 KDS EQU PIO+0 KEYBOARD DATA STROBE
A0015      * EQU PIO+1 UNUSED
A0016      0002 D1$SIZ EQU PIO+2 DS01 SIZE (1=8",0=5.25")
A0017      0003 D1$DEN EQU PIO+3 DS01 DENSITY (0=SD,1=DD)
A0018      0004 FDCINT EQU PIO+4 FDC INTERRUPT-
A0019      0005 KBDINT EQU PIO+5 KBD INTERRUPT-
A0020      0006 VDPINT EQU PIO+6 VDP INTERRUPT-
A0021      0007 BUSINT EQU PIO+7 BUS INTERRUPT-
A0022      0008 KEYBRD EQU PIO+8 KEYBOARD DATA (8 BITS)
A0023      * PARALLEL I/O - WRITE BITS
A0024      0000 CLKLED EQU PIO+0 CLOCK LED
A0025      0001 KBDACK EQU PIO+1 KEYBOARD ACKNOWLEDGE-
A0026      0002 BUSACK EQU PIO+2 BUS INTERRUPT RESET-
A0027      0003 BTENBL EQU PIO+3 BUS TIMEOUT FLAG ENABLE
A0028      0004 DRVSIZ EQU PIO+4 DRIVE SIZE (1=8",0=5.25")
A0029      0005 ROMON EQU PIO+5 0=ROM ON,1=ROM OFF
A0030      0006 BELLON EQU PIO+6 BELL ENABLE BIT
A0031      * EQU PIO+7 UNUSED
A0032      *
A0033      * EQU >0020 UNUSED
A0034      0040 EIA02 EQU >0040 PRINTER HARDWARE BASE ADDRESS
A0035      * EQU >0060 UNUSED
A0036      * EQU >0080 UNUSED
A0037      * EQU >00A0 UNUSED
A0038      00C0 CASS02 EQU >00C0 CASSETTE HARDWARE BASE ADDRESS
A0039      00E0 DMAC EQU >00E0 DMAC HARDWARE BASE ADDRESS
A0040      *
A0041      * RESERVED OFF-BOARD CRU LOCATIONS
A0042      0200 PRMPOP EQU >200 FROM PROGRAMMER BOARD
A0043      * READ BITS
A0044      * EQU PRMPOP+0
A0045      * EQU PRMPOP+1
A0046      * EQU PRMPOP+2
A0047      * EQU PRMPOP+3
A0048      0204 PRORDY EQU PRMPOP+4
A0049      0205 PGM EQU PRMPOP+5
A0050      0206 PGMPUL EQU PRMPOP+6 * TEST
A0051      0207 V30 EQU PRMPOP+7
A0052      0208 EDATA EQU PRMPOP+8
A0053      * WRITE BITS
A0054      0200 PRORST EQU PRMPOP+0
A0055      0201 EPTYPE EQU PRMPOP+1
A0056      0204 VCCON EQU PRMPOP+4
A0057      *PGM EQU PRMPOP+5
A0058      0206 PROERR EQU PRMPOP+6
A0059      * EQU PRMPOP+7
A0060      *EDATA EQU PRMPOP+8
A0061      0210 EPADR EQU PRMPOP+16
  
```



```

A0062      *
A0063      *   CENTRONICS PARALLEL PRINTER PORT
A0064      *
A0065      *   BIT       0       1       2       3       4       5       6       7       8       9
A0066      *   READ
A0067      *   WRITE    D7      D6      D5      D4      D3      D2      D1      D0      D5      BUSY
A0068      *                               (LSB)                               (MSB)
A0069      *
A0070      0400 PPRINT EQU  >400
A0071      *
A0072      0408 PPDS   EQU  PPRINT+8          DATA STROBE  ____+____+____
A0073      *
A0074      *
A0075      0409 PPBUSY EQU  PPRINT+9          BUSY          ____+-----+____
A0076      *
A0077      *****
A0078      *
A0079      *   MEMORY MAPPED I/O DEFINITIONS
A0080      *
A0081      *****
A0082      F100 MAPPER EQU  >F100             MEMORY MAPPER LOCATION
A0083      F100 M$REG0 EQU  MAPPER+0          MAPPER REGISTERS 0..15
A0084      F102 M$REG1 EQU  MAPPER+2
A0085      F104 M$REG2 EQU  MAPPER+4
A0086      F106 M$REG3 EQU  MAPPER+6
A0087      F108 M$REG4 EQU  MAPPER+8
A0088      F10A M$REG5 EQU  MAPPER+10
A0089      F10C M$REG6 EQU  MAPPER+12
A0090      F10E M$REG7 EQU  MAPPER+14
A0091      F110 M$REG8 EQU  MAPPER+16
A0092      F112 M$REG9 EQU  MAPPER+18
A0093      F114 M$REGA EQU  MAPPER+20
A0094      F116 M$REGB EQU  MAPPER+22
A0095      F118 M$REGC EQU  MAPPER+24
A0096      F11A M$REGD EQU  MAPPER+26
A0097      F11C M$REGE EQU  MAPPER+28
A0098      F11E M$REGF EQU  MAPPER+30
A0099      *
A0100      F120 VDP     EQU  >F120
A0101      F120 VRAM    EQU  VDP+0            VDP VRAM ACCESS ADDRESS
A0102      F121 VDPREG  EQU  VDP+1            VDP REGISTER ACCESS ADDRESS
A0103      *
A0104      *   TEXT / GRAPHICS 1 MODE STORAGE
A0105      0400 NTBA    EQU  >400             NAME TABLE
A0106      0700 CTBA    EQU  >700             COLOUR TABLE
A0107      0800 PGBA    EQU  >800             PATTERN GENERATOR TABLE
A0108      0780 SNTBA   EQU  >780             SPRITE NAME TABLE
A0109      0000 SPGBA   EQU  >000             SPRITE PATTERN GENERATOR TBL.
A0110      *   GRAPHICS 2 MODE STORAGE
A0111      1800 NTBA1   EQU  >1800            NAME TABLE
A0112      2000 CTBA1   EQU  >2000            COLOUR TABLE
A0113      0000 PGTBA1  EQU  >0000            PATTERN GENERATOR TABLE
A0114      1800 SNTBA1  EQU  >1800            SPRITE NAME TABLE
A0115      3800 SPGBA1  EQU  >3800            SPRITE PATTERN GENERATOR TBL.
A0116      *
A0117      F140 FDC      EQU  >F140            TMS9909 FLOPPY DISC CONTROLLER
A0118      *

```

0008	DEF	BEGN1, NRV, LNSZ, NIC, BUFEND, RBUF, RBUFE, OUTBUF
0009	DEF	EXTWP, CONFIG, PRAM, MEMSIZ, SRATE, SETVEC
0010	DEF	B43, NEWP1, FUZZ, FNSZ, GSSZ, MIDO, OVINT, ILLMID
0011	DEF	WP10L, SQRI, RANDS, CALLWP, CALL12, SYSFLG
0012	DEF	DMYHDR, DH*ARF, DH*PGN, DH*SLT, DH*VNT, DH*SIZ
0013	DEF	NOERR, DTEND, DH*END, DH*PLL, DH*HCS, HDRSIZ
0014	DEF	E*TEMP, BELCNT, CLKWS, CLKADR, CLKTO1, CLKTO2
0015	DEF	NEWP, DS, DS1, DS2, DS3, TRAF LG, CNTDWN
0016	DEF	SWPTBL, WPR1, WPR103, WPR104, WPR108
0017	DEF	BEGN, TYO, IOB, EUS, FNS, GSS, EVSKE, EDTMP
0018	DEF	CCSAVE, UFT, BUS, EORBUS, SLT, EVSKB, EBP, SSP
0019	DEF	RENSTA, RENINC, VNT, VDT, NVD, NVS
0020	DEF	CURFLG, UNIT, DCNT, MODE, PLC, DLIM, DLC, DDM
0021	DEF	LNUM, SLN, CCNT, EFLG, ENUM, ELNM, ELSF
0022	DEF	FFLG, HFLG, IFLG, PLF, BCRU, BAUD, RBSTOR, SYSLMT
0023	DEF	GSC, AINC, RENCMP, FPAC3, FPAC4, FPWP
0024	DEF	TEMP, TEMP2, TEMP4, TEMP6, FPAC, FPAC2
0025	DEF	F*WHO, CVCH, CVHD, CVHDO1, CVHD12, CVHD15
0026	DEF	WPR2, WPR1, PTRS, NEWY, PRDY, DSRcnt, RTSTOR
0027	DEF	CRLF, CRLF1, OUTDSR, ERROR2, ERROR
0028	DEF	RECPT, INPTR, RENLNE, RENRET, STACNT, IOSTOR
0029	DEF	DEVTBL, VDPWP0, XLOC, YLOC, NXLOC, NYLOC, FBCOL, VMODE
0030	DEF	BITMAP, VDPWP1, V1R5L, V1R6L, ESCFLG, VDPST, FDCDON
0031	DEF	V1R3L, V1R4L, LOADPT, WP9928, ASMPTR
0032	DEF	WORKS, WORK5, WORK11
0033	DEF	INITFG, BPTOV, BPTSET, BPTADD
0034	DEF	BPTDTA, ASMOPC, MREGS, MREG1, MREG2, MREG3, MREG6
0035	DEF	MREG13, MREG14, EREGS, IREGS, PC, IREG6, IREG12
0036	DEF	P*EUS, P*FNS, P*GSS, P*UFT, P*IOB, XREGS
0037	DEF	HALTO*, BLSTOR
0038		*
0039	REF	CMOF*, VDPDSR, SETTXT
0040	REF	CENTRO
0041	REF	GETC*, TIMC, MIDPC, INT4PC, C1
0042	REF	FLDD, FSRD, FAD, FSD, FMD, FDD, FSCL, FNRM
0043	REF	FCLR, FNEG, FLOAT, EVFX, CVBD, CVBI, ERRS
0044	REF	CKEX, CLRV, EDIT, GTLN, NLIN
0045	REF	TYPE*, D*LDWP, D*LDPC, MONTOP
0046	REF	MBEGN, MPRDY, MCRLF, B62
0047	REF	LOOK*, B47, FP5E5, FIX, EVERZ, START

```

0049      *
0050      *
0051      * ALLOCATION ADDRESSES
0052      *
0053      F000  ERAM   EQU   >F000      END OF RAM
0054      0000  BROM   EQU   >0000      BEGINNING OF ROM
0055      ECF6  PRAM   EQU   >ECF6      BEGINING OF PERMANENT RAM
0056      *
0057      * SYSTEM EQUATES
0058      *
0059      0004  NRV     EQU   4          NUMBER OF RESERVED VARIABLES
0060      0084  LNSZ    EQU   132       I/O BUFFER SIZE
0061      0064  NIC     EQU   LNSZ-32   NUMBER ON INPUT CHARACTERS
0062      3B00  FUZZ    EQU   >3B00    APPROXIMATELY 1E-7
0063      000A  FNSZ    EQU   10       NEXTED FOR/NEXT STACK SIZE
0064      0014  GSSZ    EQU   20       NEXTED QOSUB STACK SIZE
0065      0004  SQRI    EQU   4        £ OF SQR NEWTON ITERATIONS
0066      *
0067      ECF4  P$EUS   EQU   PRAM-2
0068      EC40  P$FNS   EQU   P$EUS-(FNSZ*18)
0069      EBFO  P$GSS   EQU   P$FNS-(GSSZ*4)
0070      EB84  P$UFT   EQU   P$GSS-(26*4)-4
0071      EB04  P$IOB   EQU   P$UFT-LNSZ+4
0072      *
0073      * DEFINE FLOATING POINT XOPS
0074      *
0075      DXOP  LOADF,0      LOAD FPAC
0076      DXOP  STORE,1     STORE FPAC
0077      DXOP  FADD,2      ADD TO FPAC
0078      DXOP  FSUB,3      SUBTRACT FROM FPAC
0079      DXOP  FMUL,4      MULTIPLY FPAC
0080      DXOP  FDIV,5      DIVIDE FPAC
0081      DXOP  SCALE,6     SCALE FPAC
0082      DXOP  NORMAL,7    NORMALIZE FPAC
0083      DXOP  CLEAR,8     CLEAR FPAC
0084      DXOP  NEGATE,9     NEGATE FPAC
0085      DXOP  FLOATF,10   FLOAT FPAC
0086      DXOP  EVFIX,11    EVALUATE AND FIX
0087      DXOP  OUTFP,12    OUT FLOATING POINT £
0088      DXOP  OUTINT,13   OUT INTEGER
0089      2FB0  ERROR   EQU   >2FB0     XOP XX,14 (ERROR CALL)
0090      2FA0  ERROR2  EQU   ERROR+>20
  
```

0092		*			
0093		*	RAM		
0094		*			
0095	ECF6		DORG PRAM		
0096		*			
0097		*	RANDON NUMBER GENERATOR SEED		
0098		*			
0099	ECF6 0000		RANDS DATA 0		RANDOM SEED
0100			**** ABOVE RAM IS CLEARED ON RUN ****		
0101	ECF8 0000		EDTMP DATA 0		
0102		*			
0103		*	SYSTEM POINTERS		
0104		*			
0105	ECFA	PTRS	EGU	*	SYSTEM POINTERS
0106	ECFA 0000	IOB	DATA 0		I/O BUFFER
0107	ECFC 0000	EUS	DATA 0		END-USER-STORAGE
0108	ECFE 0000	FNS	DATA 0		FOR/NEXT STACK
0109	ED00 0000	GSS	DATA 0		GOSUB STACK
0110	ED02 0000	UFT	DATA 0		USER FUNCTION TABLE
0111	ED04 0000	BUS	DATA 0		BEGINNING OF USER STORAGE
0112	ED06 0000	EORBUS	DATA 0		END OF RAM - BEGINNING OF USER
0113	ED08 0000	STACNT	DATA 0		STATEMENT INDENT COUNTER
0114	ED0A 0000	RENCMP	DATA 0		COMPRESSION LIMIT
0115	ED0C 0000	RENRET	DATA 0		RENUMBER TMP. STORAGE
0116	ED0E 0000	RENLINE	DATA 0		RENUMBER CURRENT LINE &
0117	ED10 0000	RENSTA	DATA 0		RENUMBER START LINE &
0118	ED12 0000	RENINC	DATA 0		RENUMBER INCREMENT
0119		*			
0120		*	CLOCK INTERRUPT REGISTERS		
0121		*			
0122	ED14 0000	CLKWS	DATA 0		FINE COUNTER
0123	ED16 0000	CLKADR	DATA 0		HOURS
0124	ED18 0000		DATA 0		MINUTES
0125	ED1A 0000		DATA 0		SECONDS
0126	ED1C 0000	CLKT01	DATA 0		TIC COUNTER
0127	ED1E 0000	CLKT02	DATA 0		TIC COUNTER
0128	ED20 0000	DS	DATA 0,0,0		TEMPORARY DATA STORAGE
0129	ED26 0000	DS1	DATA 0,0,0		
0130	ED2C 0000		DATA 0,0,0,0		R12,R13,R14,R15
0131	ED34 0000	RBSTOR	DATA 0		STORE FOR R8 DURING ENTER
0132	ED36 0000	RTSTOR	DATA 0		TEMP. STORE FOR R11
0133	ED24	INT1WP	EGU *-20		
0134	ED38 0000	CNTDWN	DATA 0		R11 - DELAY COUNTER
0135	ED3A 0000	TRAP1	DATA 0		R12 - BLWP VECTOR POINTER
0136	ED3C 0000		DATA 0,0,0		R13-15 CONTEXT
0137	ED42 0000	BELCNT	DATA 0		BELL COUNTER
0138	ED44 0000	TRAF LG	DATA 0		TRACE FLAG
0139	ED46	CALLWP	BSS 16*2		CALL WORKSPACE
0140	ED5E	CALL12	EGU CALLWP+24		CALL WORKSPACE R12
0141	ED66 0000	SLN	DATA 0		SAVED LINE NUMBER
0142	ED68 0000	CCNT	DATA 0		COLUMN COUNTER
0143	ED6A 0000	CURFLG	DATA 0		CURSOR FLAG 0=ON, -1=OFF

0145		*		
0146	ED6C 0000	ELSF	DATA 0	ELSE FLAG
0147	ED6E 0000	FFLG	DATA 0	FORMATTING FLAG
0148	ED70 0000	IFLG	DATA 0	INPUT FLAG
0149	ED72 0000	PLF	DATA 0	PROGRAM LOAD FLAG
0150	ED74	MIDWP	BSS 16*2	MID HANDLER WORKSPACE
0151	ED94 0000	OVINT	DATA 0	A.O. INTERRUPT HANDLER
0152	ED96 0000	MIDO	DATA 0	MID OPCODE 0 HANDLER
0153	ED98 0000	ILLMID	DATA 0	ILLEGAL MID HANDLER
0154	ED9A 0000	OUTDSR	DATA 0	OUTPUT PRE-PROCESSOR
0155	ED9C	INT4WP	EGU *	LEVEL 4 INTERRUPT WORKSPACE
0156	ED9C 0000		DATA 0,0,0,0	R0-R3
0157	EDA4 0000	LOADPT	DATA 0	R4
0158	EDA6 0000		DATA 0	R5
0159	EDAB 0000	INPTR	DATA 0	R6
0160	EDAA 0000	DSRCNT	DATA 0	R7
0161	EDAC 0000	RECPT	DATA 0	R8
0162	EDAE 0000	TRAP4	DATA 0	R9
0163	EDB0 0000		DATA 0,0,0,0,0,0	R10-15
0164	EDBC	RBUF	BSS 40	RECEIVE BUFFER
0165	EDE4	RBUFE	EGU *	
0166	EDE4	OUTBUF	BSS 80	OUTPUT BUFFER
0167	EE32	BUFEND	EGU *-2	
0168		*		
0169		*	WORKSPACES FOR VDP ROUTINES	
0170		*		
0171	EE34		EVEN	
0172	EE34 0000	VDPWP0	DATA 0	R0
0173	EE36 00	XL0C	BYTE 0	R1 MSB
0174	EE37 00	YL0C	BYTE 0	R1 LSB
0175	EE38 0000		DATA 0,0	R2-R3
0176	EE3C 00	BITMAP	BYTE 0,0,0,0,0,0,0,0	R4-R7
0177	EE44 0000		DATA 0,0,0,0,0,0,0,0	R8-R15
0178		*****		
0179	EE54		EVEN	
0180	EE54	VDPWP1	BSS 16*2	R0-R15
0181	EE5B	V1R3L	EGU VDPWP1+7	LSB OF R3
0182	EE5D	V1R4L	EGU VDPWP1+9	LSB OF R4
0183	EE5F	V1R5L	EGU VDPWP1+11	LSB OF R5
0184	EE61	V1R6L	EGU VDPWP1+13	LSB OF R6
0185		*****		
0186	EE74		EVEN	
0187	EE74	WP9928	BSS 16*2	9928 DSR WORKSPACE

0189 * I/O LOCAL STORAGE AREA
0190 EE94 EVEN
0191 EE94 IDSTOR EQU \$
0192 EE94 00 CCSAVE BYTE 0
0193 EE95 00 FBCOL BYTE 0
0194 EE96 00 NXLOC BYTE 0
0195 EE97 00 NYLOC BYTE 0
0196 EE98 0000 DATA 0.0
0197 EE9C 0000 DATA 0.0
0198 EEA0 0000 DATA 0.0
0199 EEA4 0000 DATA 0.0
0200 EEAB 0000 DATA 0.0
0201 EEAC 0000 DATA 0.0
0202 EEB0 0000 DATA 0.0
0203 EEB4 0000 DATA 0.0
0204 EEB8 0000 DATA 0.0
0205 EEBC 0000 DATA 0.0
0206 EEC0 0000 DATA 0.0
0207 EEC4 0000 DATA 0.0
0208 EEC8 0000 DATA 0.0
0209 EECB 0000 DATA 0.0
0210 EED0 0000 DATA 0.0

DEVICE 1 CURSOR CHR. SAVE
DEVICE 1 FGND/BGND COLOUR
DEVICE 1 NEW X ORDINATE
DEVICE 1 NEW Y ORDINATE
DEVICE 2 UNLD PTR, CR COUNT
DEVICE 3 UNLD PTR, CR COUNT
DEVICE 4 CENTRONICS PRINTER
DEVICE 5
DEVICE 6
DEVICE 7
DEVICE 8
DEVICE 9
DEVICE 10
DEVICE 11
DEVICE 12
DEVICE 13
DEVICE 14
DEVICE 15
DEVICE 16

0212		*		
0213		*	EXTENDED COMMAND WORKSPACE	
0214		*		
0215	EED4	EXTWP	BSS 16*2	
0216		*		
0217		*	LOBUG WORKSPACE DEFINITION	
0218		*		
0219	EEF4		EVEN	
0220	EEF4	WORKS	EGU \$	ASMBLR WORKSPACE
0221	EEFE	WORK5	EGU WORKS+10	
0222	EFOA	WORK11	EGU WORKS+22	
0223	EF14	BLSTOR	EGU WORKS+32	GETLINE RETURN STORAGE
0224	EF16	ASMPTR	EGU BLSTOR+2	ASSEMBLER INPUT BUFFER PTR
0225	EF18	INITFG	EGU ASMPTR+2	INITIALIZED FLAG
0226	EF1A	BPTOV	EGU INITFG+2	USER VECTORS FOR BPT ZERO
0227	EF1C	BPTSET	EGU BPTOV+2	BREAKPOINTS ARE SET FLAG
0228	EF1E	BPTADD	EGU BPTSET+2	BREAKPOINT ADDRESS STORE
0229	EF3E	BPTDTA	EGU BPTADD+32	BREAKPOINT DATA STORE
0230	EF5E	ASMDPC	EGU BPTDTA+32	4 BYTES
0231	EF62	MREGS	EGU ASMDPC+4	MAIN REGISTERS
0232	EF64	MREG1	EGU MREGS+2	
0233	EF66	MREG2	EGU MREGS+4	
0234	EF68	MREG3	EGU MREGS+6	
0235	EF6E	MREG6	EGU MREGS+12	
0236	EF7C	MREG13	EGU MREGS+26	
0237	EF7E	MREG14	EGU MREGS+28	
0238	EF6C	EREGS	EGU MREGS+10	ECHO REGISTERS (R11-R15 ONLY)
0239	EF78	IREGS	EGU EREGS+12	R/W REGISTERS (R10-R15 ONLY)
0240	EF7C	PC	EGU EREGS+16	ASMBLR PC STORAGE
0241	EF84	IREG6	EGU IREGS+12	
0242	EF90	IREG12	EGU IREGS+24	
0243	EF86	XREGS	EGU IREGS+14	HEX I/O, MESSAGE
0244	EFA6		DORG XREGS+32	

```

0246 *****
0247 *
0248 *           TAPE/EPROM HEADER BLOCKS
0249 *
0250 *           BASIC           MACHINE CODE
0251 * ***** RUN =>A5A5
0252 * * AUTO RUN FLAG * * AUTO RUN FLAG * NORUN=>5A5A
0253 * *****
0254 * * 8 BYTE NAME * * 8 BYTE NAME * NULL FILLED
0255 * *****
0256 * * DISP TO SLT * * >0000 *
0257 * *****
0258 * * DISP TO VNT * * LOAD ADDRESS *
0259 * *****
0260 * * NVD - VDT * * ENTRY POINT * (OFFSET)
0261 * *****
0262 * * LOAD LENGTH * * LOAD LENGTH * (BYTES)
0263 * *****
0264 * * CHECKSUM * * CHECKSUM * HEADER ONLY
0265 * *****
0266 *
0267 EFA6 DMYHDR EQU $
0268 EFA6 DH$ARF BSS 2 RUN/NORUN
0269 EFA8 DH$PGN BSS 8 PROGRAM NAME
0270 EFB0 DH$SLT BSS 2 DISP TO SLT
0271 EFB2 DH$VNT BSS 2 DISP TO VNT
0272 EFB4 DH$SIZ BSS 2 NVD-VDT
0273 EFB6 DH$PLL BSS 2 LENGTH
0274 EFB8 DH$HCS BSS 2 HEADER CHECKSUM
0275 0014 HDRSIZ EQU $-DMYHDR HEADER BLOCK SIZE
0276 EFBA DH$END EQU $ END OF DUMMY HEADER
0277 *****
0278 EFBA 0000 SLT DATA 0 STATEMENT LOCATION TABLE
0279 EFBC 0000 VNT DATA 0 VARIABLE DEFINITION TABLE
0280 EFBE 0000 VDT DATA 0 NEXT VARIABLE DEFINITION
0281 EFC0 0000 NVD DATA 0 NEXT VARIABLE POINTER
0282 EFC2 0000 NVS DATA 0 NEXT VARIABLE STORAGE
0283 EFC4 0000 GSC DATA 0 GOSUB STACK COUNTER
0284 EFC6 0000 TYO DATA 0 OUTPUT CHARACTER BUFFER
0285 EFC8 0000 AINC DATA 0 AUTO-INCREMENT
0286 EFCA 0000 DCNT DATA 0 INDENT COUNTER
0287 EFCC 0000 MODE DATA 0 MODE SWITCH
0288 EFCE 0000 PLC DATA 0 PROGRAM LINE COUNTER
0289 EFD0 0000 DLIM DATA 0 DELIMITER
0290 EFD2 0000 DLC DATA 0 DATA LINE COUNTER
0291 EFD4 0000 DDM DATA 0 DATA DELIMITER PTR
0292 EFD6 0000 LNUM DATA 0 LAST ENTERED LINE £
0293 EFD8 0000 F$WHO DATA 0 0=BASIC, -1= MONITOR
0294 *
0295 EFDA 0000 DS2 DATA 0,0,0 TEMP FP STORAGE
0296 EFEE 0000 DS3 DATA 0,0,0 TEMP FP STORAGE
0297 EFE6 0000 E$TEMP DATA 0,0,0 EVALUATOR TEMP STORAGE
0298 *
0299 * STACKS FOR EDIT,LIST, AND EVAL
0300 *
0301 EFEC 0000 CVCH DATA 0,0,0
0302 EFF2 CVHD EQU $
0303 EFF3 CVHD01 EQU CVHD+1
0304 EFFE CVHD12 EQU CVHD+12
0305 F001 CVHD15 EQU CVHD+15

```


0306	F000	EVSKE	EGU	#+14	BEGINNING OF STACK
0307	EFF2	EBP	BSS	132	EDIT BUFFER
0308	F076	SSP	BSS	30	SUBSCRIPT STACK
0309	F094	EVSKE	EGU	#	END STACK
0310		*			
0311		*	* FLOATING POINT WORKSPACE AND TEMPORARY STORAGE		
0312		*			
0313	F094	0000	TEMP	DATA	0
0314	F096	0000	TEMP2	DATA	0
0315	F098	0000	TEMP4	DATA	0
0316	F09A	0000	TEMP6	DATA	0
0317	F09C	0000	FPAC	DATA	0
					FP ACCUMULATOR
0318	F09F	FPAC3	EGU	#+1	
0319	F09E	0000	FPAC2	DATA	0
0320	F0A0	0000	FPAC4	DATA	0
0321	F0A2		BSS	13*2	
0322	F09C	FPWP	EGU	FPAC	FLOATING POINT WORKSPACE
0323		*			
0324		*	* 2ND LEVEL REGISTERS		
0325		*			
0326	F0BC	WPR2	BSS	16*2	EVAL, TRIG, CONV REGISTERS
0327		*			
0328		*	* 1ST LEVEL REGISTERS - RESET WORKSPACE ALSO		
0329		*			
0330	F0DD	WP10L	EGU	#+1	R0 LSB REFERENCE
0331	F0DC	0000	WPR1	DATA	0,0,0
					R0 - R2
0332	F0E2	0000	WPR103	DATA	0
					R3
0333	F0E4	0000	WPR104	DATA	0
					R4
0334	F0E6	0000		DATA	0,0,0
					R5 - R7
0335	F0EC	0000	WPR108	DATA	0
					R8
0336	F0EE	0000	WPR109	DATA	0
					R9 (POINTER REGISTER)
0337	F0F0	0000		DATA	0,0,0,0,0,0
					R10 - R15
0338	F0FC	ENDRAM	EGU	#	
0339		*			
0340		*	* CHECK THAT ENDRAM IS OK		
0341		*			
0342			ASMIF (>F0FC-ENDRAM)&>8000		
0343			SPLAT !!! - RUN OUT OF INTERNAL RAM		
0344			ASMEND		

```

0346 0000          RORG BROM
0347          *
0348          * INTERRUPT VECTOR TABLE
0349          *
0350 0000 F0DC IVO    DATA WPR1, START
0351 0004 ED24 IV1    DATA INT1WP, INT1PC
0352 0008 ED74 IV2    DATA MIDWP, MIDPC
0353 000C ED14 IV3    DATA CLKWS, CLKI
0354 0010 ED9C IV4    DATA INT4WP, INT4PC
0355          *
0356          * USER READABLE SYSTEM FLAGS
0357          *
0358          0014' SYSFLG EQU *
0359 0014 0000 HFLG DATA 0
0360 0016 0000 ENUM DATA 0
0361 0018 0000 ELNM DATA 0
0362 001A 0000 BCRU DATA 0
0363 001C 0000 EFLG DATA 0
0364 001E 0000 UNIT DATA 0
0365 0020 0000 ESCFLG DATA 0
0366 0022 0000 MEMSIZ DATA 0
0367 0024 0000 NDERR DATA 0
0368 0026 0000 VMODE DATA 0
0369 0028 0000 FDCDON DATA 0
0370 002A FFFF DATA -1
0371          000B VDPST EQU 11
0372 002C 024A' DATA BEGN
0373 002E ED3A DATA TRAP1
0374 0030 EDAE DATA TRAP4
0375 0032 ED96 DATA MID0
0376 0034 ED94 DATA OVINT
0377 0036 ED98 DATA ILLMID
0378 0038 ED9A DATA OUTDSR
0379 003A EF1A DATA BPTOV
0380          0013 SYSLMT EQU 19
0381          *
0382 003C FFFF DATA -1, -1

```

```

*** SYSTEM FLAG AREA ***
HELP FLAG          SYS(0)
LAST ERROR NUMBER  SYS(1)
LAST ERROR LINE #  SYS(2)
CRU BASE ADDRESS   SYS(3)
ERROR FLAG         SYS(4)
UNIT FLAG          SYS(5)
ESCAPE DISABLE FLAG SYS(6)
AUTO-SIZE LOW LIMIT SYS(7)
EDIT ERR ENABLE FLG SYS(8)
MODE 0=TEXT, -1=GRAPH SYS(9)
FDC INTERRUPT FLAG SYS(10)
DUMMY
VDP STATUS FLAG    SYS(11)
BASIC ENTRY POINT  SYS(12)
INT1 BLWP VECTOR   SYS(13)
INT4 BLWP VECTOR   SYS(14)
MID 0 BLWP VECTOR   SYS(15)
AO BLWP VECTOR     SYS(16)
ILLMID BLWP VECTOR  SYS(17)
OUTPUT BLWP VECTOR  SYS(18)
BPO BL VECTOR      SYS(19)
MAX SYSTEM LIMIT

```

```

0384      *
0385      * XOP VECTOR TABLE
0386      *
0387 0040 F09C      DATA FPWP,FLDD      XOP XX,0  LOAD FPAC
0388 0044 F09C      DATA FPWP,FSRD      XOP XX,1  STORE FPAC
0389 0048 F09C      DATA FPWP,FAD      XOP XX,2  ADD TO FPAC
0390 004C F09C      DATA FPWP,FSD      XOP XX,3  SUBTRACT FROM FPAC
0391 0050 F09C      DATA FPWP,FMD      XOP XX,4  MULTIPLY FPAC
0392 0054 F09C      DATA FPWP,FDD      XOP XX,5  DIVIDE FPAC
0393 0058 F09C      DATA FPWP,FSCL      XOP XX,6  SCALE FPAC
0394 005C F09C      DATA FPWP,FNRM      XOP XX,7  NORMALIZE
0395 0060 F09C      DATA FPWP,FCLR      XOP XX,8  CLEAR
0396 0064 F09C      DATA FPWP,FNEG      XOP XX,9  NEGATE
0397 0068 F09C      DATA FPWP,FLOAT      XOP XX,10 FLOAT FPAC
0398 006C F0BC      DATA WPR2,EVFX      XOP XX,11 EVALUATE AND FIX
0399 0070 F0BC      DATA WPR2,CVBD      XOP XX,11 OUT FP £
0400 0074 F0BC      DATA WPR2,CVBI      XOP XX,13 OUT INTEGER £
0401 0078 F0DC      D$WPR1 DATA WPR1,ERRS      USER ERROR
0402 007C FFFF      DATA -1,-1
0403 0080 0460      MONITR B      @MONTOP      ENTRY TO MONITOR
      0082 0000
  
```

```

0405      *
0406      *      DEVICE TABLE: -
0407      *
0408      *      1,  IF THE ENTRY IS 0 THEN THE DEVICE IS NOT PRESENT.
0409      *      2,  IF THE LS BIT OF THE ENTRY IS A '0' THEN
0410      *              THE ENTRY IS THE ADDRESS OF THE TRANSFER
0411      *              VECTOR FOR THE SERVICE ROUTINE.
0412      *      3,  IF THE LS BIT OF THE ENTRY IS A '1' THEN
0413      *              THE DEVICE IS A 9902, BITS 1-14 FORM THE
0414      *              R12 CONTENTS FOR THE SERVICE ROUTINE AND
0415      *              A CR DELAY WILL BE ADDED IF THE MSB IS A '1'
0416      *
  
```

```

0417 0084 0000  DEVTBL DATA VDPDSR          1-VDP
0418 0086 0081      DATA (2*EIA02)+>0001    2-EIA 9902, NO CR DELAY
0419 0088 0181      DATA (2*CASS02)+>0001    3-CASS. 9902, NO CR DELAY
0420 008A 0000      DATA CENTRO              4-CENTRONICS PRINTER
0421 008C 0000      DATA 0                    5-UNUSED DEVICE
0422 008E 0000      DATA 0                    6-UNUSED DEVICE
0423 0090 0000      DATA 0                    7-UNUSED DEVICE
0424 0092 0000      DATA 0                    8-UNUSED DEVICE
0425 0094 0000      DATA 0                    9-UNUSED DEVICE
0426 0096 0000      DATA 0                   10-UNUSED DEVICE
0427 0098 0000      DATA 0                   11-UNUSED DEVICE
0428 009A 0000      DATA 0                   12-UNUSED DEVICE
0429 009C 0000      DATA 0                   13-UNUSED DEVICE
0430 009E 0000      DATA 0                   14-UNUSED DEVICE
0431 00A0 0000      DATA 0                   15-UNUSED DEVICE
0432 00A2 0000      DATA 0                   16-UNUSED DEVICE
  
```

```

0433      00A4' DTEND EQU *
  
```

```

0434      *
0435      *      COLOUR SWAP TABLE
0436      *
  
```

```

0437 00A4 0001  SWPTBL DATA >0001
0438 00A6 0203      DATA >0203
0439 00AB 0405      DATA >0405
0440 00AA 0607      DATA >0607
0441 00AC 0809      DATA >0809
0442 00AE 0A0B      DATA >0A0B
0443 00B0 0C0D      DATA >0C0D
0444 00B2 0E0F      DATA >0E0F
  
```

```

0445      *
0446      *      SYSTEM CONFIGURATION FLAGS
0447      *      =====
  
```

```

0448      *
0449      *      X X X X   X X X X   X X X X   X X X X
0450      *
0451      *
0452      *      ! ! !- MAPPER FITTED
0453      *      ! !--- PROG.  FITTED
0454      *      !----- FLOPPY FITTED
  
```

```

0455 00B4 0000  CONFIG DATA >0000          DEFAULT TO MINIMUM SYSTEM
  
```

```

0457      * CLOCK INTERRUPT ROUTINE
0458      *
0459      *      R0=FINE COUNTER
0460      *      R1=HOURS
0461      *      R2=MINUTES
0462      *      R3=SECONDS
0463      *      R4-R5=TOTAL TIC COUNTS
0464      *
0465      00B6' CLKI  EGU  $
0466      00B6 020C  LI   R12,>1EE0      REF ONCHIP DECREMENTER
0467      00B8 1EE0
0468      00BA 1D01  SBO   1      RE-ENABLE CLOCK INTERRUPT
0469      00BC 0620  DEC   @CNTDWN    COUNTDOWN
0470      00BE ED38
0471      00C0 C320  MOV   @BELCNT,R12   GET BELL COUNTER
0472      00C2 ED42
0473      00C4 091C  SRL   R12,1      COUNT
0474      00C6 C80C  MOV   R12,@BELCNT  RESTORE IT
0475      00CB ED42
0476      ASMIF CLKLED
0477      LI   R12,2*CLKLED    POINT TO CLOCK LED
0478      ASMELS
0479      CLR  R12
0480      ASMEMD
0481      * TEST SHIFT FROM A COUPLE OF LINES AGO
0482      00CC 1702  JNC   NOBELL      NOT IN USE, LEAVE BELL ALONE
0483      00CE 1D06  SBO   BELLON-CLKLED  IN USE, TURN BELL ON
0484      00D0 1001  JMP   DOTIC
0485      00D2 1E06  NOBELL SBZ   BELLON-CLKLED  BELL OFF
0486      00D4 1E00  DOTIC  SBZ   CLKLED-CLKLED  LED ON
0487      00D6 0280  CI     R0,100/2      TIME FOR LED OFF?
0488      00D8 0032
0489      00DA 1A01  JL     DOTIC1      N, LEAVE IT ON
0490      00DC 1D00  SBO   CLKLED-CLKLED  Y, TURN IT OFF
0491      00DE 0585  DOTIC1 INC   R5      COUNT TICS
0492      *
0493      * IF R5 = 0 THEN LEAST SIG WORD OF TIC COUNTER HAS
0494      * 'OVERFLOWED' - UPDATE R4
0495      *
0496      00E0 1601  JNE   $+4
0497      00E2 0584  INC   R4
0498      00E4 0280  CI     R0,99      100 TICS?
0499      00E6 0063
0500      00E8 1402  JHE   CLKI1      Y, 1 SECOND
0501      00EA 0580  INC   R0      N, COUNT
0502      00EC 1010  JMP   CLKI2      EXIT
0503      *
0504      * 100 TICKS HAVE OCCURED - UPDATE REAL TIME CLOCK
0505      *
0506      00EE 04C0  CLKI1  CLR   R0      CLEAR FINE COUNTER
0507      00F0 0583  INC   R3      COUNT SECONDS
0508      00F2 0283  CI     R3,60    <60?
0509      00F4 003C
0510      00F6 110B  JLT   CLKI2      Y, RETURN
0511      00F8 04C3  CLR   R3      N
0512      00FA 0582  INC   R2      INCREMENT MINUTE
0513      00FC 0282  CI     R2,60    <60?
0514      00FE 003C
0515      0100 1106  JLT   CLKI2      Y, RETURN
0516      0102 04C2  CLR   R2      N

```

0509	0104	0581	INC	R1	INCREMENT HOUR
0510	0106	0281	CI	R1,24	<24 HOURS?
	0108	0018			
0511	010A	1101	JLT	CLKI2	Y, RETURN
0512	010C	04C1	CLR	R1	N
0513		010E' CLKI2	EGU	*	
0514	010E	0380	RTWP		
0515	0110	1000	NOP		
0516	0112	0380	RTWP		

```

0518 *****
0519 *
0520 *          LEVEL 1 INTERRUPT SERVICE ROUTINE
0521 *
0522 *****
0523 0114 C30C INT1PC MOV R12,R12          TRAP SET UP?
0524 0116 1302 JEQ BUSTIM              N, ERROR
0525 0118 041C BLWP *R12                Y, TAKE TRAP
0526 011A 0380 RTWP
0527 *
0528 011C 1F07 BUSTIM TB BUSINT          WAS IT A BUS TIMEOUT ?
0529 011E 1303 JEQ ERR42              N, ERROR THEN
0530 0120 1E03 SBZ BTENBL              Y, RESET IT
0531 0122 1D03 SBO BTENBL
0532 0124 0380 RTWP
0533 0126 2FA0 ERR42 DATA ERROR2,42    AND EXIT
0534 *****                                INTERRUPT W/O TRAP
0535 *
0536 * SET LOAD VECTORS AND CRITICAL SYSTEM POINTERS
0537 *
0538 *****
0539 *
0540 * THIS ROUTINE MUST :-
0541 * 1. RESET THE RECEIVE BUFFER POINTERS
0542 * 2. SET THE UNIT FLAG TO >0001
0543 * 3. SETUP AND START THE DECREMETER
0544 * 4. SETUP THE NMI VECTORS FOR WARMSTART
0545 * 5. RESET THE ON-CHIP FLAG REGISTER
0546 * 6. SET THE VDP UP FOR TEXT MODE
0547 * 7. ENABLE ALL INTERRUPTS
0548 *
0549 012A' SETVEC EQU $
0550 012A 0000 DATA CMDF$              TURN CASSETTE OFF & FLUSH BUF.
0551 012C 0201 LI R1,>FFFA            ; POINT TO DECREMETER
0552 0130 C820 MOV @C1,@UNIT          ; SET UNIT 1
0553 0136 020C LI R12,>1EE0          ; POINT TO FLAGS
0554 013A CC60 MOV @TIMC,*R1+         ; LOAD DECREMETER
0555 013E CC60 MOV @D$LDWP,*R1+      ; SET UP LOAD WP
0556 0142 CC60 MOV @D$LDPC,*R1+     ; SET UP LOAD PC
0557 0146 3001 LDCR R1,0             ; CONFIG DECREMETER & CLR FLAGS
0558 0148 1D01 SBO 1                 ; ENABLE DECREMETER
0559 014A DB20 MOV @B47,@FBCOL       ; FCOL=BLUE,BCOL=CYAN
0560 0150 0000 DATA SETTXT          ; SET TEXT MODE
0561 0152 04E0 CLR @CURFLG           ; CURSOR ON
0562 0154 ED6A
0562 0156 045B RT                    ; EXIT
  
```

```

0564 *****
0565 *
0566 *
0567 *
0568 *
0569 0158 2FA0 ERR43 DATA ERROR2,43 INVALID BAUD RATE
0570 015C 2FA0 ERR37 DATA ERROR2,37 INVALID DELIMITER
0571 0160 2FA0 ERR46 DATA ERROR2,46 INVALID DEVICE £
0572 *
0573 0164 2EC1 BAUD EVFIX R1 GET DEVICE £
0574 0166 0601 DEC R1 DEVICE £ 1..16
0575 0168 0281 CI R1,15 VALID?
0576 016A 000F
0576 016C 1BF9 JH ERR46 N, ERROR IT
0577 016E A041 A R1,R1 Y, MAKE TABLE INDEX
0578 0170 C321 MOV @DEVTBL(1),R12 GET DEVICE ENTRY
0578 0172 0084
0579 0174 2320 CDC @C1,R12 IS IT A 9902?
0579 0176 0132
0580 0178 16F3 JNE ERR46 N, ERROR IT
0581 017A 024C ANDI R12,>7FFE ISOLATE CRUBASE
0581 017C 7FFE
0582 017E 0280 CI R0,>3F00 ' ' FOLLOWING ?
0582 0180 3F00
0583 0182 16EC JNE ERR37 N, ERROR IT
0584 0184 0420 BLWP @EVERZ Y, GET BAUD RATE
0584 0186 0000
0585 0188 2C12 LOADF *R2 EVAL STACK ==> FPAC
0586 018A 2E80 FLOATF 0 FORCE FLOATING POINT IF REQ.
0587 018C 2C60 STORE @DS1 SAVE IT
0587 018E ED26
0588 0190 2C20 LOADF @FP5E5 FPAC = 5E+5
0588 0192 0000
0589 0194 2D60 FDIV @DS1 CALC (5E+5/RATE)
0589 0196 ED26
0590 0198 0202 LI R2,FPAC POINT TO FPAC
0590 019A F09C
0591 019C 06A0 BL @FIX TRY TO FIX IT
0591 019E 0000
0592 01A0 04C2 CLR R2 SET FOR NO PRE-DIVIDE
0593 01A2 0203 LI R3,1024
0593 01A4 0400
0594 01A6 80C1 C R1,R3 IN RANGE ?
0595 01A8 1A04 JL BAUD1 Y, LEAVE IT
0596 01AA C0B3 MOV R3,R2 N, SET PRE-DIVIDE
0597 01AC 0931 SRL R1,3 ADJUST REGISTER VALUE
0598 01AE 80C1 C R1,R3 NOW VALID?
0599 01B0 14D3 JHE ERR43 N, ERROR IT
0600 01B2 A0B1 BAUD1 A R1,R2 01B2 ADD IN PRE-DIVIDE
0601 01B4 020B LI R11,NLIN SET RETURN ADDRESS TO NLIN
0601 01B6 0000
0602 *
0603 01B8 1D1F SRATE SBD 31 RESET 9902
0604 01BA C6DB MOV *R11,*R11 DELAY
0605 01BC 3220 LDCR @B62,8 2 STOP, EV PAR, 7 BITS, /3 CLOCK
0605 01BE 0000 5546 Data
0606 01C0 1E0D SBZ 13 NO TIMER
0607 01C2 3302 LDCR R2,12 LOAD XMT/REC BAUD RATE
0608 01C4 1D12 SBD 18 ENABLE REC. INTERRUPTS
0609 01C6 045B RT EXIT
  
```


START SDSMAC 3.4.0 B1.117 15:30:53 THURSDAY, DEC 23, 1982.
START MODULE - CORTEX BASIC REV 1.1

PAGE 0020

0610 0108 4300 B43 DATA >4300

```

0612 *****
0613 *
0614 * NEW STATEMENT *
0615 *
0616 *****
0617 01CA 2FA0 ERR49 DATA ERROR2,49 ; ILLEGAL MEMORY ADDRESS
0618 01CE C060 NEWY MOV @EORBUS,R1 ; GET DEFAULT MEMORY SIZE
      01D0 ED06
0619 01D2 06A0 BL @CKEX ; EXPRESSION ?
      01D4 0000
0620 01D6 100A JMP NEWY1 ; N, TAKE DEFAULT
0621 01D8 2EC2 EVFIX R2 ; Y, GET NEW LIMIT
0622 01DA 0222 AI R2,HDRSIZ ; ALLOW ROOM FOR HEADER BLOCK
      01DC 0014
0623 01DE 8802 C R2,@IOB ; IN SYSTEM RAM ?
      01E0 ECFA
0624 01E2 14F3 JHE ERR49 ; Y, ILLEGAL MEMORY ADDRESS
0625 01E4 8802 C R2,@MEMSIZ ; BELOW MEMSIZ ???
      01E6 0022
0626 01E8 1AF0 JL ERR49 ; Y, ILLEGAL MEMORY ADDRESS
0627 01EA C042 MOV R2,R1 ; N, SET R1
0628 01EC C801 NEWY1 MOV R1,@EORBUS ; SET NEW EORBUS
      01EE ED06
0629 01F0 C801 MOV R1,@BUS ; SAVE NEW BUS
      01F2 ED04
0630 *
0631 * NEW
0632 *
0633 01F4 020B NEWP LI R11,BEGN1 ; LOAD CONTINUATION ADDRESS
      01F6 0252
0634 * ( FOR BL @CLRV )
0635 01F8 NEWP1 EQU $ ; AUTO-RUN ENTRY POINT
0636 01FB C060 MOV @BUS,R1 ; GET BEGINNING USER STORAGE
      01FA ED04
0637 01FC C801 MOV R1,@SLT ; SET STATEMENT LOCATION TABLE
      01FE EFBA
0638 0200 04F1 CLR *R1+ ; LEAVE NULL
0639 0202 C801 MOV R1,@VNT ; SET VARIABLE DEFINITION TABLE
      0204 EFBC
0640 0206 C0C1 MOV R1,R3
0641 0208 0223 AI R3,NRV*2 ; SKIP RESERVED WORDS
      020A 000B
0642 020C C803 MOV R3,@VDT ; SET NEXT VARIABLE DEFINITION
      020E EFBE
0643 0210 0223 AI R3,NRV*2
      0212 000B
0644 0214 C803 MOV R3,@NVD ; SET NEXT VARIABLE POINTER
      0216 EFC0
0645 0218 0460 B @CLRV ; CLEAR *R1 TO EUS
      021A 0000

```

```

0647 *****
0648 *
0649 *          KEYBOARD MODE ROUTINE
0650 *
0651 *****
0652 021C 0201 PRDY  LI  R1,MPRDY          ;GET MESSAGE
      021E 0000
0653 *
0654 0220 04E0 PRDY1 CLR  @AINC          ;CLEAR AUTO-INC FLAG
      0222 EFC8
0655 0224 04E0      CLR  @TRAFLO        ;TRACE OFF
      0226 ED44
0656 0228 1002      JMP  CRLF0
0657 *
0658 022A 0201 CRLF  LI  R1,MCRLF        ;OUT CRLF
      022C 0000
0659 *
0660 022E 04E0 CRLF0 CLR  @MODE          ;SET MODE TO IDLE
      0230 EFCC
0661 0232 04E0      CLR  @ESCFLG        ;ENABLE ESCAPE
      0234 0020
0662 0236 04E0      CLR  @F$WHO         ;FLAG IN BASIC
      0238 EFD8
0663 023A 0000      DATA TYPE$        ;OUT STRING R1
0664 023C 04E0      CLR  @PLF           ;CLEAR PROGRAM LOAD
      023E ED72
0665 0240 06A0      BL   @GTLN          ;GET INPUT LINE
      0242 0000
0666 0244 06A0 CRLF1 BL   @EDIT         ;EDIT STATEMENT
      0246 0000
0667 0248 10F0      JMP  CRLF
0668 *****
0669 *
0670 *          SYSTEM WARM START
0671 *          =====
0672 *
0673 *          SET CRITICAL SYSTEM POINTERS AND RE-ENTER THE
0674 *          INTERPRETER.
0675 *
0676 *****
0677 024A 02E0 BEGN  LWPI WPR1          ;GET RIGHT WORKSPACE
      024C F0DC
0678 024E 06A0      BL   @SETVEC        ;RE-INITIALIZE
      0250 012A
0679 0252 02E0 BEGN1 LWPI WPR1          ;MAKE SURE WORKSPACE OK
      0254 F0DC
0680 0256 0201      LI   R1,MBEGN       ;GET MESSAGE
      0258 0000
0681 025A 10E2      JMP  PRDY1

```

```

0683 *****
0684 *
0685 *          HALT OUTPUT CHECK ROUTINE
0686 *          =====
0687 *
0688 *          USE THE SPACE BAR TO HALT AND THEN SINGLE STEP
0689 *          THE OUTPUT.
0690 *
0691 *****
0692 025C F0BC HALT0$ DATA WPR2, HALT1
0693 *
0694 0260 0000 HALT1 DATA GETC$          CHARACTER?
0695 0262 0380          RTWP              N, EXIT
0696 0264 1000          NOP              Y, NORMAL CHARACTER
0697 0266 0280          CI   R0, >2000    SPACE ?
        0268 2000
0698 026A 1605          JNE  EXIT          N, EXIT
0699 026C 0000 WAITK DATA LOOK$          Y, ANOTHER KEY ?
0700 026E 10FE          JMP  WAITK        LOOP
0701 0270 0280          CI   R0, >2000    Y, SPACE?
        0272 2000
0702 0274 16F5          JNE  HALT1        N, READ IT THEN
0703 0276 0380 EXIT RTWP Y, LEAVE IT FOR NEXT TIME
0704          END
NO ERRORS,      NO WARNINGS

```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.MID
OBJECT ACCESS NAME= ADHOC.OBJ.MID
LISTING ACCESS NAME= ADHOC.LST.MID
ERROR ACCESS NAME=
OPTIONS= BUNLST, TUNLST, DUNLST, MUNLST
MACRO LIBRARY PATHNAME=

LINE	KEY	NAME
------	-----	------

0043	A	ADHOC.SRC.IOBITS =>ADHOC.SRC.IOBITS
------	---	--

```

0002          IDT 'MID'
0003          *****
0004          * THIS MODULE HANDLES THE LEVEL 2 INTERRUPT
0005          * DECIDING WETHER IT IS AN ARITHMETIC OVERFLOW
0006          * OR A MID. FOR A MID THE OPCODE IS CHECKED
0007          * TO SEE IF IT IS A SYSTEM MID, IF SO THE
0008          * REQUIRED HANDLER IS EXECUTED.
0009          *
0010          * MIDO      ADDRESS FOR A MID OPCODE >0000 HANDLER
0011          * OVINT     ADDRESS FOR AN ARITHMETIC OVERFLOW HANDLER
0012          * ILLMID    ADDRESS FOR AN ILLEGAL MID HANDLER
0013          *
0014          * IF NO HANDLER IS AVAILIABLE FOR AN ARITHMETIC OVERFLOW
0015          * OR NON SYSTEM MID AN 'INTERRUPT WITHOUT TRAP' ERROR OCCURS
0016          *
0017          * HANDLERS ARE ENTERED WITH ALL INTERRUPTS DISABLED AND EXIT
0018          * IS VIA AN 'RTWP'
0019          *
0020          * REGISTER USAGE ( ~ = RESERVED REGISTER )
0021          *   R0-3   NOT USED
0022          *   R4     GENERAL FLAG, CLEARED ON ENTRY
0023          *   R5-11  NOT USED
0024          *   R12    TEMP. REGISTER
0025          *   ~ R13-15 RETURN VECTORS
0026          *
0027          DEF  MIDPC, TYP0$, TYPC$, TYPB$, TYPE$, LOOK$
0028          DEF  TYP11$, TYPBE$, GETC$, GETCR$, TYP$
0029          DEF  GCLEAR, TYP$N$, TYP$EN$, DELAY$, WAIT$
0030          DEF  SETTXT, SETGRA, LOADER, SENDAD, STXT, SGRA
0031          DEF  CMON$, CMOF$, FLUSH$, FTM$, FGM$
0032          *
0033          REF  FBCOL, LDCS, XLOC, VMODE, CCSAVE
0034          REF  OVINT, ERROR2, MIDO, ILLMID
0035          REF  S$SM, B00, B20, OUTDSR, C1
0036          REF  MCRLF, TYO, UNIT, OUTBUF, BUFEND
0037          REF  DSRcnt, LOADPT
0038          REF  RECPtr, INPTr, RBUF
0039          REF  CNTDWN, B1B, IOB, IOSTOR, DEVTBL
0040          REF  B62, XREGS, WHXETY, RHENTY, WHENTY
0041          REF  EREGS, ECHOEN, IREGS, WENTRY, RENTRY
0042          REF  MREGS, MENTRY, XOPENT
0043          COPY ADHOC. SRC. IOBITS

```

```

A0002      *
A0003      * THIS MODULE IS INCLUDED IN SOURCE MODULES
A0004      * BY USING A 'COPY' DIRECTIVE.
A0005      *
A0006      *
A0007      *****
A0008      *
A0009      * CRU DEFINITIONS
A0010      *
A0011      *****
A0012      0000 PIO      EQU  >0000      PARALLEL I/O
A0013      * PARALLEL I/O - READ BITS
A0014      0000 KDS      EQU  PIO+0      KEYBOARD DATA STROBE
A0015      *          EQU  PIO+1      UNUSED
A0016      0002 D1$SIZ EQU  PIO+2      DS01 SIZE (1=8",0=5.25")
A0017      0003 D1$DEN EQU  PIO+3      DS01 DENSITY (0=SD,1=DD)
A0018      0004 FDCINT EQU  PIO+4      FDC INTERRUPT-
A0019      0005 KBDINT EQU  PIO+5      KBD INTERRUPT-
A0020      0006 VDPINT EQU  PIO+6      VDP INTERRUPT-
A0021      0007 BUSINT EQU  PIO+7      BUS INTERRUPT-
A0022      0008 KEYBRD EQU  PIO+8      KEYBOARD DATA (8 BITS)
A0023      * PARALLEL I/O - WRITE BITS
A0024      0000 CLKLED EQU  PIO+0      CLOCK LED
A0025      0001 KBDACK EQU  PIO+1      KEYBOARD ACKNOWLEDGE-
A0026      0002 BUSACK EQU  PIO+2      BUS INTERRUPT RESET-
A0027      0003 BTENBL EQU  PIO+3      BUS TIMEOUT FLAG ENABLE
A0028      0004 DRVSIZ EQU  PIO+4      DRIVE SIZE (1=8",0=5.25")
A0029      0005 ROMON  EQU  PIO+5      0=ROM ON,1=ROM OFF
A0030      0006 BELLON EQU  PIO+6      BELL ENABLE BIT
A0031      *          EQU  PIO+7      UNUSED
A0032      *
A0033      *          EQU  >0020      UNUSED
A0034      0040 EIA02  EQU  >0040      PRINTER HARDWARE BASE ADDRESS
A0035      *          EQU  >0060      UNUSED
A0036      *          EQU  >0080      UNUSED
A0037      *          EQU  >00A0      UNUSED
A0038      00C0 CASS02 EQU  >00C0      CASSETTE HARDWARE BASE ADDRESS
A0039      00E0 DMAC   EQU  >00E0      DMAC HARDWARE BASE ADDRESS
A0040      *
A0041      * RESERVED OFF-BOARD CRU LOCATIONS
A0042      0200 PRMPOP EQU  >200      PROM PROGRAMMER BOARD
A0043      * READ BITS
A0044      *          EQU  PRMPOP+0
A0045      *          EQU  PRMPOP+1
A0046      *          EQU  PRMPOP+2
A0047      *          EQU  PRMPOP+3
A0048      0204 PRORDY EQU  PRMPOP+4
A0049      0205 PGM    EQU  PRMPOP+5
A0050      0206 PGMPUL EQU  PRMPOP+6
A0051      0207 V30    EQU  PRMPOP+7
A0052      0208 EDATA  EQU  PRMPOP+8
A0053      * WRITE BITS
A0054      0200 PRORST EQU  PRMPOP+0
A0055      0201 EPTYPE EQU  PRMPOP+1
A0056      0204 VCCON  EQU  PRMPOP+4
A0057      *PGM    EQU  PRMPOP+5
A0058      0206 PROERR EQU  PRMPOP+6
A0059      *          EQU  PRMPOP+7
A0060      *EDATA  EQU  PRMPOP+8
A0061      0210 EPADR  EQU  PRMPOP+16

```

* TEST

```

A0062      *
A0063      *   CENTRONICS PARALLEL PRINTER PORT
A0064      *
A0065      *   BIT       0       1       2       3       4       5       6       7       8       9
A0066      *   READ
A0067      *   WRITE    D7      D6      D5      D4      D3      D2      D1      D0      D5
A0068      *                      (LSB)                                (MSB)
A0069      *
A0070      0400 PPRINT EQU  >400
A0071      *
A0072      0408 PPDS  EQU  PPRINT+8          DATA STROBE  ____+____+____
A0073      *
A0074      *
A0075      0409 PPBUSY EQU  PPRINT+9          BUSY          ____+____+____
A0076      *
A0077      *****
A0078      *
A0079      *   MEMORY MAPPED I/O DEFINITIONS
A0080      *
A0081      *****
A0082      F100 MAPPER EQU  >F100          MEMORY MAPPER LOCATION
A0083      F100 M$REG0 EQU  MAPPER+0      MAPPER REGISTERS 0..15
A0084      F102 M$REG1 EQU  MAPPER+2
A0085      F104 M$REG2 EQU  MAPPER+4
A0086      F106 M$REG3 EQU  MAPPER+6
A0087      F108 M$REG4 EQU  MAPPER+8
A0088      F10A M$REG5 EQU  MAPPER+10
A0089      F10C M$REG6 EQU  MAPPER+12
A0090      F10E M$REG7 EQU  MAPPER+14
A0091      F110 M$REG8 EQU  MAPPER+16
A0092      F112 M$REG9 EQU  MAPPER+18
A0093      F114 M$REGA EQU  MAPPER+20
A0094      F116 M$REGB EQU  MAPPER+22
A0095      F118 M$REGC EQU  MAPPER+24
A0096      F11A M$REGD EQU  MAPPER+26
A0097      F11C M$REGE EQU  MAPPER+28
A0098      F11E M$REGF EQU  MAPPER+30
A0099      *
A0100      F120 VDP      EQU  >F120
A0101      F120 VRAM     EQU  VDP+0          VDP VRAM ACCESS ADDRESS
A0102      F121 VDPREG   EQU  VDP+1          VDP REGISTER ACCESS ADDRESS
A0103      *
A0104      *   TEXT / GRAPHICS 1 MODE STORAGE
A0105      0400 NTBA     EQU  >400          NAME TABLE
A0106      0700 CTBA     EQU  >700          COLOUR TABLE
A0107      0800 PGBA     EQU  >800          PATTERN GENERATOR TABLE
A0108      0780 SNTBA    EQU  >780          SPRITE NAME TABLE
A0109      0000 SPGBA    EQU  >000          SPRITE PATTERN GENERATOR TBL.
A0110      *   GRAPHICS 2 MODE STORAGE
A0111      1800 NTBA1    EQU  >1800         NAME TABLE
A0112      2000 CTBA1    EQU  >2000         COLOUR TABLE
A0113      0000 PGTBA1   EQU  >0000         PATTERN GENERATOR TABLE
A0114      1800 SNTBA1   EQU  >1800         SPRITE NAME TABLE
A0115      3800 SPGBA1   EQU  >3800         SPRITE PATTERN GENERATOR TBL.
A0116      *
A0117      F140 FDC       EQU  >F140         TMS9909 FLOPPY DISC CONTROLLER
A0118      *
0044      0000 MBIAS     EQU  >0000         NORMAL MID
0045      *
0046      0000 0300 MIDPC  LIM1 3          NO I/O INTERRUPTS

```


0002 0003			
0047 0004 04C4	CLR R4	RESET FLAG	
0048 0006 026F	ORI R15,>000F	MAKE SURE IMASK=15 ON EXIT	
0008 000F			
0049 000A 020C	LI R12,>1FDA	POINT TO MID FLAG	
000C 1FDA			
0050 000E 1F00	TB 0	MID?	
0051 0010 1305	JEG TSTMID	Y, CHECK FOR SYSTEM MID	
0052 0012 024F	ANDI R15,>F7FF	N, KILL OVERFLOW BIT	
0014 F7FF			
0053 0016 C320	MOV @OVINT,R12	GET A.O. HANDLER	
0018 0000			
0054 001A 1006	JMP VERIFY	VALIDATE	
0055	*		
0056 001C 1E00	TSTMID SBZ 0	KILL MID FLAG	
0057 001E C32E	MOV @-2(14),R12	GET INSTRUCTION	
0020 FFFE			
0058 0022 160C	JNE SYSMID	NOT 0, CHECK IF SYSTEM MID	
0059 0024 C320	MOV @MIDO,R12	0, GET HANDLER ADDRESS	
0026 0000			
0060	*		
0061 0028 1608	VERIFY JNE SERV	HANDLER, EXECUTE	
0062 002A 0000	DATA ERROR2, 44	NO HANDLER -ILLEGAL OPCODE	
0063	*		
0064 002E C320	BADMID MOV @ILLMID,R12	GET ILLEGAL MID HANDLER	
0030 0000			
0065 0032 10FA	JMP VERIFY	CHECK IT	
0066 0034 A30C	DECODE A R12,R12	GET WORD INDEX INTO TABLE	
0067 0036 C32C	MOV @MIDTAB(12),R12	GET HANDLER	
0038 00C8			
0068 003A 045C	SERV B *R12	EXECUTE HANDLER	
0069	*		
0070 003C 028C	SYSMID CI R12,MAXMID	SYSTEM MID ?	
003E 0015			
0071 0040 12F9	JLE DECODE	Y, SERVICE IT	
0072 0042 0300	LIMI 15	N, RE-ENABLE INTERRUPTS	
0044 000F			
0073 0046 022C	AI R12,->0E00	SUBTRACT MONITOR MID BASE	
0048 F200			
0074 004A 11F1	JLT BADMID	ILLEGAL, TRY ERROR TRAP	
0075 004C 028C	CI R12,>01FF	TOO BIG ?	
004E 01FF			
0076 0050 1BEE	JH BADMID	Y, TRY ERROR TRAP	
0077	*		
0078	*	EVALUATE OPERAND	
0079	*		
0080 0052 C2CC	MOV R12,R11	COPY OPCODE	
0081 0054 024B	ANDI R11,>000F	ISOLATE REGISTER FIELD	
0056 000F			
0082 0058 630B	S R11,R12	REMOVE REG. FIELD FROM OPCODE	
0083 005A 093C	SRL R12,3	ALIGN Ts IN BITS 13&14	
0084 005C C20C	MOV R12,R8	SAVE IT	
0085 005E 024B	ANDI R8,>0006	ISOLATE (2 * Ts)	
0060 0006			
0086 0062 630B	S R8,R12	SUBTRACT IT FROM OPCODE	
0087 0064 091C	SRL R12,1	R12 =(4 * FUNCTION £)	
0088 0066 A2CB	A R11,R11	R11 = 2 * REGISTER £	
0089 0068 A2CD	A R13,R11	R11 = ADDRESS OF REGISTER	
0090 006A C22B	MOV @FMTTY(8),R8	GET ADDRESS OF Ts HANDLER	
006C 0070			

```

0091 006E 045B      B      *R8      GOTO ROUTINE
0092                *
0093 0070 00BE' FMTTYP DATA $REG, $IND, $SYMB, $AINC
0094                *
0095 007B C2DB      $IND      MOV      *R11, R11      FETCH REGISTER CONTENTS
0096 007A 1009      JMP      $REG      EXIT
0097                *
0098 007C C21B      $SYMB      MOV      *R11, R8      GET REGISTER CONTENTS
0099 007E 834B      C          R11, R13      USING R0 ?
0100 0080 1601      JNE      NOTRO      N, OK TO INDEX WITH IT
0101 0082 04C8      CLR      R8          Y, DON'T INDEX
0102 0084 A23E      NOTRO      A          *R14+, R8      ADD IN IMEDIATE OPERAND
0103 0086 1002      JMP      $REG8      GO PUT IT IN R11
0104                *
0105 0088 C21B      $AINC      MOV      *R11, R8      GET REGISTER CONTENTS
0106 008A 05DB      INCT      *R11      INC REGISTER
0107 008C C2C8      $REG8      MOV      R8, R11      PUT RESULT IN R11
0108                *
0109      00BE' $REG      EQU      $
0110                *
0111                * EFFECTIVE ADDRESS OF OPERAND IS IN R11
0112                *
0113                *
0114 008E C22C      MOV      @MONTBL(12), R8      PICK UP TARGET WP
0115 0090 00AA'      MOV      R11, @22(8)      PASS OVER EFFECTIVE ADDRESS
0116 0094 0016      MOV      R13, @26(8)      SAVE CALLERS WP
0117 0098 001A      MOV      R14, @28(8)      SAVE CALLERS PC
0118 009C 001C      MOV      R15, @30(8)      SAVE CALLERS ST
0119 00A0 001E      *
0120 00A2 C34B      MOV      R8, R13      SET UP WP FOR TRANSFER
0121 00A4 C3AC      MOV      @MONTBL+2(12), R14 SET UP PC FOR TRANSFER
0122 00A6 00AC'
0123 00AB 0380      RTWP      EXECUTE UTILITY
0124                *
0125                * VECTORS FOR MONITOR UTILITIES
0126 00AA 0000      MONTBL DATA XREGS, WHXETY      1 NIBBLE HEX OUTPUT
0127 00AE 00AA'      DATA XREGS, RHENTY      READ HEX INPUT
0128 00B2 00AE'      DATA XREGS, WHENTY      4 NIBBLE HEX OUTPUT
0129 00B6 0000      DATA EREGS, ECHOEN      CHARACTER ECHO
0130 00BA 0000      DATA IREGS, WENTRY      CHARACTER WRITE
0131 00BE 00BA'      DATA IREGS, RENTRY      CHARACTER READ
0132 00C2 00B2'      DATA XREGS, MENTRY      MESSAGE OUTPUT
0133 00C6 0000      DATA MREGS, XOPENT      BREAKPOINT ENTRY

```

0135	00C8'	MIDTAB EQU	*-2	0 - RESERVED MID
0136		*		
0137	0001	TYPO\$	EQU (\$-MIDTAB/2)+MBIAS	1 - OUT R0
0138	00CA 00F8'	DATA	ETYP0	
0139	0002	TYP\$	EQU (\$-MIDTAB/2)+MBIAS	2 - OUT 'CRLF'
0140	00CC 0106'	DATA	ETYP0	
0141	0003	TYPB\$	EQU (\$-MIDTAB/2)+MBIAS	3 - OUT IOB(0)
0142	00CE 0112'	DATA	ETYPB	
0143	0004	TYPE\$	EQU (\$-MIDTAB/2)+MBIAS	4 - OUT *R1 STRING(0)
0144	00D0 011A'	DATA	ETYPB	
0145	0005	TYPEN\$	EQU (\$-MIDTAB/2)+MBIAS	5 - OUT *R1 STRING(-)
0146	00D2 0118'	DATA	ETYPEN	
0147	0006	TYPS\$	EQU (\$-MIDTAB/2)+MBIAS	6 - OUT <INLINE ADR> S
0148	00D4 0122'	DATA	ETYPS	
0149	0007	TYPSN\$	EQU (\$-MIDTAB/2)+MBIAS	7 - OUT <INLINE ADR> S
0150	00D6 0120'	DATA	ETYPSN	
0151	0008	DELAY\$	EQU (\$-MIDTAB/2)+MBIAS	8 - DELAY LOOP
0152	00DB 0178'	DATA	EDELAY	
0153	0009	TYP11\$	EQU (\$-MIDTAB/2)+MBIAS	9 - OUT INLINE CHARACTER
0154	00DA 00F4'	DATA	ETYP11	
0155	000A	TYPBE\$	EQU (\$-MIDTAB/2)+MBIAS	A - TERM & OUT IOB
0156	00DC 010C'	DATA	ETYPBE	
0157	000B	GETC\$	EQU (\$-MIDTAB/2)+MBIAS	B - TEST FOR CHARACTER
0158	00DE 01F6'	DATA	EGETC	
0159	000C	GETCR\$	EQU (\$-MIDTAB/2)+MBIAS	C - GET CHARACTER
0160	00E0 01F4'	DATA	EGETCR	
0161	000D	LOOK\$	EQU (\$-MIDTAB/2)+MBIAS	D - TEST INPUT BUFFER
0162	00E2 0248'	DATA	E\$LOOK	
0163	000E	WAIT\$	EQU (\$-MIDTAB/2)+MBIAS	E - WAIT FOR OUTPUT TO
0164	00E4 0172'	DATA	EWAIT	
0165	000F	SETTXT	EQU (\$-MIDTAB/2)+MBIAS	F - SET TEXT MODE
0166	00E6 0262'	DATA	STXT	
0167	0010	SETGRA	EQU (\$-MIDTAB/2)+MBIAS	10 - SET GRAPHICS MODE
0168	00E8 02B4'	DATA	SGRA	
0169	0011	FTM\$	EQU (\$-MIDTAB/2)+MBIAS	11 - FORCE TEXT MODE
0170	00EA 025C'	DATA	EFTM\$	
0171	0012	FGM\$	EQU (\$-MIDTAB/2)+MBIAS	12 - FORCE GRAPHIC MODE
0172	00EC 02AE'	DATA	EFGM\$	
0173	0013	FLUSH\$	EQU (\$-MIDTAB/2)+MBIAS	13 - FLUSH INPUT BUFFER
0174	00EE 0204'	DATA	EFLUSH	
0175	0014	CMON\$	EQU (\$-MIDTAB/2)+MBIAS	14 - CASSETTE ON
0176	00F0 022E'	DATA	ECMON	
0177	0015	CMOF\$	EQU (\$-MIDTAB/2)+MBIAS	15 - CASSETTE OFF
0178	00F2 0238'	DATA	ECMOF	
0179	0015	MAXMID EQU	*-MIDTAB-2/2	HIGHEST SYSTEM MID

Change
Use for spare

14 = CMSG OUT.

```

0181      *
0182      * THIS SECTION PROVIDES THE OUTPUT ROUTINES
0183      * FOR BASIC I/O.
0184      *
0185      * TYP11$ - OUT INLINE CHARACTER
0186      * TYPO$  - OUT R0
0187      * TYPB$  - OUT 'CRLF'
0188      * TYPBE$ - TERMINATE & OUTPUT I/O BUFFER
0189      * TYPB$  - OUT I/O BUFFER(0)
0190      * TYPE$  - OUT *R1 STRING(0)
0191      * TYPEN$ - OUT *R1 STRING(-)
0192      * TYP$   - OUT <INLINE ADR> STRING(0)
0193      * TYP$   - OUT <INLINE ADR> STRING(-)
0194      * WAIT$  - WAIT FOR OUTPUT TO COMPLETE
0195      * DELAY$ - DELAY FOR CNTDWN * 40ms
0196      *
0197      *      REGISTER USAGE
0198      *      R0      UNUSED
0199      *      R1      UNUSED
0200      *      R2      SCRATCH
0201      *      R3      SCRATCH
0202      *      R4      TERMINATOR TYPE FLAG
0203      *      R5      UNIT FLAGS
0204      *      R6      SAVED RETURN ADDRESS
0205      *      R7      BUFFER START POINTER
0206      *      R8      BUFFER LOAD POINTER
0207      *      R9      SYSTEM POINTER
0208      *      R10     TEXT POINTER
0209      *      R11     RETURN ADDRESS
0210      *      R12     TEMP. REGISTER
0211      *      R13     )
0212      *      R14     ) RTWP VECTORS
0213      *      R15     )
0214      *
0215 00F4 C33E ETYP11 MOV  *R14+,R12      GET INLINE CHARACTER
0216 00F6 1001      JMP  FORTY0          SKIP R0 FETCH
0217 00F8 C31D ETYP0  MOV  *R13,R12      GET R0
0218 00FA 020A FORTY0 LI   R10,TY0      GET CHARACTER BUFFER
0219      00FC 0000
0219 00FE 024C      ANDI R12,>FF00      ZERO LSB
0219      0100 FF00
0220 0102 C68C      MOV  R12,*R10      SAVE CHARACTER IN BUFFER
0221 0104 100F      JMP  TYPE0          & OUTPUT
0222 0106 020A ETYPB  LI   R10,MCRLF   POINT TO 'CRLF'
0222      0108 0000
0223 010A 100C      JMP  TYPE0          & OUTPUT
0224 010C C32D ETYPBE MOV  @14(13),R12  GET IOB POINTER
0224      010E 000E
0225 0110 771C      SB   *R12,*R12     TERMINATE IOB
0226 0112 C2A0 ETYPB  MOV  @IOB,R10    GET IOB START
0226      0114 0000
0227 0116 1006      JMP  TYPE0          & OUTPUT
0228 0118 0704 ETYPEN SETO R4          SET -VE TERMINATOR FLAG
0229 011A C2AD ETYPE  MOV  @2(13),R10   GET CALLERS R1
0229      011C 0002
0230 011E 1002      JMP  TYPE0          & OUTPUT
0231 0120 0704 ETYPSN SETO R4          SET -VE TERMINATOR FLAG
0232 0122 C2BE ETYPS  MOV  *R14+,R10     GET INLINE ADDRESS & BUMP PC
0233      *      FALL THROUGH TO ETYP0
0234      *

```

```

0235          * NOW OUTPUT STRING *R10 (TERMINATED BY A NULL)
0236          *
0237 0124 06A0 TYPE0 BL @WAIT          GET BUFFER
      0126 0160'
0238 0128 0208 SETPTR LI R8,OUTBUF      GET BUFFER START
      012A 0000
0239 012C C1C8          MOV R8,R7        SAVE FOR DSRGO
0240 012E DE3A FILBUF MOVB *R10+,*R8+    COPY OVER BYTE
0241 0130 1309          JEQ ENDX1        NULL, EXIT
0242 0132 150F          JGT FILB1        +VE, CONTINUE COPY
0243 0134 C104          MOV R4,R4        -VE, LOOKING FOR -VE TERM. ?
0244 0136 130D          JEQ FILB1        N, CONTINUE COPY
0245 0138 04CB          CLR R11          Y, READY HOLDING REGISTER
0246 013A 0608          DEC R8           BACKUP TO -VE CHARACTER
0247 013C D2D8          MOVB *R8,R11     GET BYTE BACK
0248 013E 050B          NEG R11          CORRECT IT
0249 0140 D60B          MOVB R11,*R8     PUT IT BACK
0250 0142 1001          JMP ENDXFR       AND EXIT
0251 0144 0648 ENDX1 DECT R8             BACKUP TO LAST CHARACTER
0252 0146 81C8 ENDXFR C R8,R7           WAS IT A SINGLE NULL?
0253 0148 1401          JHE ENDX2        N, OUT AS NORMAL
0254 014A 058B          INC R8           Y, BUMP END POINTER
0255 014C 06A0 ENDX2 BL @DSRGO          START DSR'S
      014E 0182'
0256 0150 0380          RTWP
0257          *
0258 0152 0288 FILB1 CI R8,BUFEND        BUFFER FULL?
      0154 0000
0259 0156 1AEB          JL FILBUF        N, CONTINUE
0260 0158 060B          DEC R8
0261 015A 06A0          BL @DSRGO        Y, OUT BUFFER SO FAR
      015C 0182'
0262 015E 10E2          JMP TYPE0        & CONTINUE

```

```

0264      *
0265      *      SYSTEM WAIT ROUTINES
0266      *
0267      *
0268      * WAIT FOR BUFFER
0269      *
0270 0160 0208 WAIT LI R8,DSRCNT
      0162 0000
0271 0164 1001      JMP WAIT1
0272      *
0273 0166 0340 WAIT0 IDLE
0274 0168 0300 WAIT1 LIM1 15      ENABLE INTERRUPTS
      016A 000F
0275 016C C618      MOV *R8,*R8      GET ACTIVE DSR COUNT
0276 016E 15FB      JGT WAIT0      ACTIVE, IDLE
0277 0170 045B      B *R11      AVAILIABLE, RETURN
0278      *
0279      * WAIT FOR OUTPUT TO COMPLETE
0280      *
0281 0172 06A0 EWAIT BL @WAIT      GET BUFFER
      0174 0160'
0282 0176 0380      RTWP      EXIT WHEN AVAILIABLE
0283      *
0284      * WAIT TILL CNTDWN HAS GONE TO 0 (OR -VE)
0285      *
0286 0178 0208 EDELAY LI R8,CNTDWN      POINT TO CNTDWN
      017A 0000
0287 017C 06A0      BL @WAIT1      DELAY
      017E 0168'
0288 0180 0380      RTWP      & EXIT

```

```

0290      *
0291      * THIS ROUTINE HANDS THE OUTPUT BUFFER TO A PRE-PROCESSOR
0292      * IF REQUIRED AND THEN STARTS THE OUTPUT TRANSFER
0293      * ON 9902'S SELECTED BY 'UNIT' AND ANY NON-STANDARD DEVICE
0294      * HANDLERS ARE BID VIA THE BLWP VECTOR IN THE DEVICE TABLE
0295      * R7 POINTS TO THE 1ST BYTE OF THE MESSAGE
0296      * R8 POINTS TO THE LAST BYTE OF THE MESSAGE
0297      *
0298 0182 C18B DSRGD MOV R11,R6      SAVE RETURN
0299 0184 0300      LIM1 15      RE-ENABLE INTERRUPTS
      0186 000F
0300 0188 C808      MOV R8,@LOADPT      SAVE MSG END FOR DSR'S
      018A 0000
0301 018C C160      MOV @UNIT,R5      GET UNIT FLAGS
      018E 0000
0302 0190 C2E0      MOV @OUTDSR,R11    PRE-PROCESSOR ?
      0192 0000
0303 0194 1301      JEQ OUTIT          N, SKIP
0304 0196 041B      BLWP *R11          Y, EXECUTE IT
0305 0198 04E0 OUTIT CLR @DSRCNT      RESET DSR COUNT
      019A 0162
0306 019C 0208      LI R11,DEVTBL     POINT TO DEVICE TABLE
      019E 0000
0307 01A0 0202      LI R2,IOSTOR      REF I/O STORAGE AREA
      01A2 0000
0308 01A4 0203      LI R3,16         DO 16 DEVICES
      01A6 0010
0309      *
0310 01AB C33B DSRGD00 MOV *R11+,R12    GET DEVICE ENTRY
0311 01AA 130E      JEQ DSRGD01        O, NO DEVICE INSTALLED
0312 01AC 2320      CDC @C1,R12        IS IT A 9902?
      01AE 0000
0313 01B0 1611      JNE DSRGD02        N, CALL IT'S HANDLER
0314 01B2 024C      ANDI R12,>7FFE     Y, ISOLATE CRUBASE
      01B4 7FFE
0315 01B6 C487      MOV R7,*R2        SET UNLOAD POINTER
0316 01B8 0722      SETD @2(2)        RESET CR DELAY
      01BA 0002
0317 01BC 2160      CDC @C1,R5        DEVICE SELECTED?
      01BE 01AE
0318 01C0 1603      JNE DSRGD01        N, SKIP IT
0319 01C2 05A0      INC @DSRCNT        Y, COUNT IT
      01C4 019A
0320 01C6 1D13      SBD 19            ENABLE XMIT INTERRUPTS
0321 01C8 0B15 DSRGD01 SRC R5,1        SHIFT UNIT BIT
0322 01CA 1303      JEQ DGOXIT        O, NO MORE UNITS
0323 01CC 8CB2      C *R2+,*R2+      STEP TO NEXT STORAGE AREA
0324 01CE 0603      DEC R3            DONE ALL DEVICES?
0325 01D0 16EB      JNE DSRGD00        N, LOOP
0326 01D2 0456 DGOXIT B *R6            Y, RETURN
0327      *
0328 01D4 2160 DSRGD02 CDC @C1,R5        IS THE DEVICE SELECTED?
      01D6 01BE
0329 01D8 16F7      JNE DSRGD01        N, NEXT ENTRY
0330      *
0331      *      DEVICE HANDLER INSTALLED, PASS OVER POINTERS
0332      *      HANDLER WP:
0333      *      R9 = OUTPUT BUFFER START
0334      *      R10 = OUTPUT BUFFER END
0335      *      R11 = POINTER TO LOCAL STORAGE

```

	*	R13-15 RETURN CONTEXT	
0336	*		
0337	*		
0338 01DA C00C		MOV R12,R0	SAVE HANDLER ADDRESS
0339 01DC C31C		MOV *R12,R12	GET ITS WP
0340 01DE CB07		MOV R7,@2*R9(12)	SET BUFFER START
01E0 0012			
0341 01E2 CB08		MOV R8,@2*R10(12)	SET BUFFER END
01E4 0014			
0342 01E6 CB02		MOV R2,@2*R11(12)	SET LOCAL STORAGE PTR
01E8 0016			
0343 01EA 0410		BLWP *R0	CALL HANDLER
0344 01EC 10ED		JMP DSRG01	NEXT ENTRY


```

0346      *
0347      *   THIS MODULE HANDLES THE CHARACTER INPUT FOR BASIC.
0348      *   THE INPUT BUFFER IS FILLED BY THE 9902 DSR WHEN A
0349      *   RECEIVE INTERRUPT IS ENCOUNTERED. THESE ROUTINES UNLOAD
0350      *   THIS BUFFER AND RETURN THE CHARACTERS TO BASIC.
0351      *
0352      *
0353      *   WAIT FOR CHARACTER ROUTINE
0354      *
0355      *   CALLING SEQUENCE :
0356      *       DATA GETCR$           CALL GET CHARACTER MID
0357      *
0358      *   CHARACTER RETURNED IN R0 EXCEPT WHEN AN ESCAPE IS
0359      *   ENCOUNTERED WHILE ESCAPE IS ENABLED. IN THIS CASE
0360      *   PROGRAM EXECUTION IS RETURNED TO PRDY IN KEYBOARD MODE
0361      *   OR STPY IN RUN MODE.
0362      *   IF AN ESCAPE IS ENCOUNTERED AN HAS BEEN DISABLED VIA
0363      *   A 'NOESC' STATEMENT THE ESCAPE CHARACTER IS RETURNED
0364      *   IN THE MSB OF R0 AS WITH ANY OTHER CHARACTER.
0365      *
0366      *****
0367      *
0368      *       TEST FOR CHARACTER INPUT
0369      *
0370      *   CALLING SEQUENCE :
0371      *       DATA GETC$           CALL TEST INPUT MID
0372      *       <NO CHARACTER>       )
0373      *       < CHARACTER >       )RETURN POINTS
0374      *       <  ESCAPE  >       )
0375      *
0376      *   THIS ROUTINE TESTS THE INPUT BUFFER FOR A CHARACTER AND
0377      *   RETURNS TO ONE OF THE FOLLOWING THREE WORDS DEPENDING ON
0378      *   THE RESULTS OF THE TEST.
0379      *
0380      *   NO CHARACTER      EXECUTION RESUMES AT THE WORD FOLLOWING
0381      *                       THE CALL.
0382      *   CHARACTER        EXECUTION RETURNS TWO WORDS AFTER THE
0383      *                       CALL WITH THE CHARACTER IN THE MSB OF R7.
0384      *   ESCAPE           IF THE ESCAPE HAS BEEN DISABLED VIA THE
0385      *                       'NOESC' COMMAND THE RETURN IS AS FOR A
0386      *                       NORMAL CHARACTER. OTHERWISE EXECUTION
0387      *                       RESUMES THREE WORDS AFTER THE CALL.
0388      *
0389      01EE 0300  GETC1  LIM1 15           ALLOW INTERRUPTS
0390      01F0 000F
0391      0390 01F2 0340      IDLE           WAIT FOR ONE
0392      01F4 0704  EGETCR SET0 R4        SET WAIT FLAG
0393      *
0394      01F6 0300  EGETC  LIM1 0         DISABLE INTERRUPTS
0395      01F8 0000
0396      01FA C220      MOV  @RECPTR,R8    GET UNLOAD PTR
0397      01FC 0000
0398      01FE 8808      C    R8,@INPTR     BUFFER EMPTY?
0399      0200 0000
0400      0202 1A09      JL   GETC2        N, UNLOAD
0401      *
0402      *       FLUSH THE INPUT BUFFER ( R4 IS CLEARED )
0403      *
0404      *
0405      0204 0208  EFLUSH LI  R8,RBUF     Y, GET START OF BUFFER

```

0206 0000			
0402 0208 C808		MOV R8,@INPTR	RESET LOAD PTR
020A 0200			
0403 020C C808		MOV R8,@RECPTR	RESET UNLOAD PTR
020E 01FC			
0404 0210 C104		MOV R4,R4	WAIT ?
0405 0212 16ED		JNE GETC1	Y, LOOP
0406 0214 0380		RTWP	N, EXIT
0407	*		
0408 0216 04DD	GETC2	CLR *R13	READY CALLERS RO
0409 0218 D778		MOVB *R8+,*R13	COPY OVER BYTE
0410 021A C808		MOV R8,@RECPTR	UPDATE UNLOAD POINTER
021C 020E			
0411 021E C104		MOV R4,R4	WAS IT A WAIT CALL?
0412 0220 1605		JNE GETC5	Y, EXIT WITH CHARACTER
0413 0222 981D		CB *R13,@B1B	N, WAS IT AN ESCAPE CHARACTER?
0224 0000			
0414 0226 1601		JNE GETC4	N, EXIT 2(14) - NORMAL CHAR.
0415 0228 05CE		INCT R14	4(14) - ESCAPE
0416 022A 05CE	GETC4	INCT R14	2(14) - CHARACTER IN RO
0417 022C 0380	GETC5	RTWP	0(14) - NO CHARACTER
0418	*		
0419	*	CASSETTE SUPPORT ROUTINES	
0420	*		
0421 022E 020C	ECMON	LI R12,2*CASS02	SET CRUBASE
0230 0180			
0422 0232 1D10		SBO 16	TURN MOTOR ON
0423 0234 1D12		SBO 18	ENABLE IT'S RECEIVE INTERRUPTS
0424 0236 10E6		JMP EFLUSH	FLUSH BUFFER & EXIT
0425	*		
0426 0238 020C	ECMOF	LI R12,2*CASS02	SET CRUBASE
023A 0180			
0427 023C 1E12		SBZ 18	KILL IT'S RECEIVE INTERRUPTS
0428 023E 1E10		SBZ 16	TURN MOTOR OFF
0429 0240 1D0E		SBO 14	SET TO LOAD CONTROL REG.
0430 0242 3220		LDCR @B62,8	2 STOP,EV PAR,7 BITS,/3 CLOCK
0244 0000			
0431 0246 10DE		JMP EFLUSH	FLUSH BUFFER & EXIT

```

0433 *****
0434 *
0435 *          TEST THE INPUT BUFFER ROUTINE
0436 *
0437 *          THIS ROUTINE TESTS TO SEE IF THERE IS ANYTHING
0438 *          IN THE INPUT BUFFER, IF NOT THE BUFFER POINTERS
0439 *          ARE RESET AND THEN AN IMMEDIATE RETURN IS MADE.
0440 *          IF THERE IS A CHARACTER IN THE BUFFER
0441 *          THEN IT IS RETURNED IN THE CALLERS RO(MSB) WITH
0442 *          THE LSB CLEARED.
0443 *          NB. THE CHARACTER IS NOT REMOVED FROM THE BUFFER.
0444 *
0445 *          CALLING SEQUENCE :
0446 *          DATA LOOK$
0447 *          <EMPTY RETURN>
0448 *          <CHARACTER RETURN>
0449 *
0450 *****
0451 024B 0300 E$LOOK LIM1 0          MASK INTERRUPTS
      024A 0000
0452 024C C220      MOV @RECPTR,R8      FETCH RECEIVE POINTER
      024E 021C'
0453 0250 8808      C      RB,@INPTR      UPTO THE LOADING POINTER ?
      0252 020A'
0454 0254 14D7      JHE EFLUSH          Y, NOTHING THERE - EXIT
0455 0256 04DD      CLR *R13          N, CLEAR CALLERS RO
0456 0258 D758      MOVB *RB,*R13      COPY THE CHARACTER OVER
0457 025A 10E7      JMP GETC4          AND EXIT @2(R14)

```

```

0459 *****
0460 *
0461 *          VDP UTILITY ROUTINES
0462 *
0463 *****
0464 025C C020 EFTM$ MOV @VMODE,R0          TEXT MODE ALREADY?
      025E 0000
0465 0260 1325      JEQ STMXIT          Y, EXIT
0466 *
0467 *          SET VDP FOR TEXT MODE
0468 *
0469      0262' STXT EQU $
0470 0262 0300      LIM1 15          ALLOW INTERRUPTS
      0264 000F
0471 0266 D820      MOV8 @FBCOL,@SFBC      SET FGND/BGND COLOUR
      0268 0000
      026A 0278'
0472 026C 06A0      BL @LOADER          LOAD VDP REGISTERS
      026E 0368'
0473 0270 00      BYTE >00,>80+R0      TEXT MODE, EXT. VIDE0 OFF
0474 0272 90      BYTE >90,>80+R1      16K,NO DISPLAY,NO INT,TXT,SIZ&MAG=0
0475 0274 01      BYTE >01,>80+R2      PNTBA=>400
0476 0276 01      BYTE >01,>80+R4      PGTBA=>800
0477 0278 00 SFBC  BYTE >00,>80+R7      FGND/BGND COLOURS
0478 027A 0000      DATA 0
0479 027C 0420      BLWP @LDCS          LOAD DEFAULT CHARACTER SET
      027E 0000
0480 0280 0208      LI RB,NTBA->4000      REF PNT
      0282 4400
0481 0284 06A0      BL @SENDAD          SEND ADDRESS TO VDP
      0286 037A'
0482 0288 0208      LI R11,40*24          SET SCREEN SIZE
      028A 03C0
0483 *
0484 028C 04E0 CLS CLR @XLOC          XLOC=0, YLOC=0
      028E 0000
0485 0290 D820      MOV8 @B20,@VRAM      WRITE SPACE CHARACTER
      0292 0000
      0294 F120
0486 0296 0608      DEC R11          COUNT IT
0487 0298 16F9      JNE CLS          LOOP TILL ALL DONE
0488 029A D820      MOV8 @B20,@CCSAVE      SET SAVED CHARACTER TO A ' '
      029C 0292'
      029E 0000
0489 02A0 04E0      CLR @VMODE          FLAG VDP IN TEXT MODE
      02A2 025E'
0490 *
0491 02A4 06A0      BL @LOADER          RE-ENABLE DISPLAY
      02A6 0368'
0492 02A8 D0      BYTE >D0,>80+R1      16K,DISPLAY ON,NO INT,TXT,SIZ&MAG=0
0493 02AA 0000      DATA 0
0494 *
0495 02AC 0380      STMXIT RTWP          EXIT
0496 *
0497 02AE C020 EFGM$ MOV @VMODE,R0          IN TEXT MODE?
      02B0 02A2'
0498 02B2 16FC      JNE STMXIT          N, EXIT
0499 *
0500 *          SET VDP FOR GRAPHIC 2 MODE
0501 *

```

```

0502      02B4' SGRA      EQU  $
0503 02B4 0300          LIM1 15          ALLOW INTERRUPTS
      02B4 000F
0504 02B8 D820          MOV8 @FBCOL, @BDCOL
      02BA 0268'
      02BC 02D0'
0505 02BE 06A0          BL  @LOADER          LOAD VDP REGISTERS
      02C0 0368'
0506 02C2 02          BYTE >02, >80+R0      GRAPHICS 2, NO EXT. VIDE0
0507 02C4 80          BYTE >80, >80+R1      DISABLE DISPLAY
0508 02C6 06          BYTE >06, >80+R2      PNT=>1800
0509 02C8 FF          BYTE >FF, >80+R3      CTBA=>2000
0510 02CA 03          BYTE >03, >80+R4      PGT=>0000
0511 02CC 36          BYTE >36, >80+R5      SAT=>1800
0512 02CE 07          BYTE >07, >80+R6      SPG=>3800
0513 02D0 00 BDCOL     BYTE >00, >80+R7      BACKDROP=BGND COLOUR
0514 02D2 0000          DATA 0
0515      *
0516      * CLEAR DOWN THE SNT & SPG TABLES
0517      *
0518 02D4 04C6          CLR  R6          WRITE NULLS
0519 02D6 0207          LI   R7, VRAM      TO VRAM
      02D8 F120
0520 02DA 0208          LI   R8, SPGBA1+>4000 REF SPG
      02DC 7800
0521 02DE 06A0          BL  @SENDAD        POINT VDP AT IT
      02E0 037A'
0522 02E2 0208          LI   R8, 256*8      DO 256 8 BIT PATTERNS
      02E4 0800
0523      *
0524 02E6 D5C6 KILSPG   MOV8 R6, *R7      NULL OUT THE PATTERN
0525 02E8 C208          MOV  R8, R8      WASTE SOME TIME
0526 02EA 0608          DEC  R8      DONE ?
0527 02EC 16FC          JNE  KILSPG
0528      *
0529 02EE 0208          LI   R8, SNTBA1+>4000 REF SNT
      02F0 5800
0530 02F2 06A0          BL  @SENDAD        POINT VDP AT IT
      02F4 037A'
0531 02F6 0208          LI   R8, 32*8      KILL OFF ALL 32 PLANES
      02F8 0100
0532 02FA 0206          LI   R6, >D000      FILL IT FULL OF SPRITE TERM.
      02FC D000
0533      *
0534 02FE D5C6 KILSNT   MOV8 R6, *R7      CLEAR DOWN SNT
0535 0300 C208          MOV  R8, R8      WASTE SOME TIME
0536 0302 0608          DEC  R8      DONE?
0537 0304 16FC          JNE  KILSNT        N, LOOP
0538      *
0539      * SET UP PATTERN NAME TABLE
0540 0306 0208 GCLEAR   LI   R8, NTBA1+>4000 REF PNT
      0308 5800
0541 030A 06A0          BL  @SENDAD        SEN ADDRESS TO VDP
      030C 037A'
0542 030E 070B          SET0 R11          RESET COUNT
0543      *
0544 0310 058B SGRA1    INC  R11          NEXT PATTERN
0545 0312 06CB          SWPB R11          POSITION LS BYTE
0546 0314 D80B          MOV8 R11, @VRAM     WRITE IT
      0316 F120

```

0547	0318	060B		SWPB R11	RESTORE R11
0548	031A	028B		CI R11,3*256	DONE ALL ENTRIES ?
	031C	0300			
0549	031E	1AF8		JL SGRA1	N, LOOP
0550			*	SET UP PATTERN GENERATOR TABLE	
0551	0320	0208		LI RB,PGTBA1+>4000	REF PGT
	0322	4000			
0552	0324	06A0		BL @SENDAD	SEN ADDRESS TO VDP
	0326	037A			
0553	0328	020B		LI R11,3*256*8	RESET COUNT
	032A	1800			
0554			*		
0555	032C	04E0	SGRA2	CLR @XLOC	RESET XLOC & YLOC
	032E	028E			
0556	0330	D820		MOVB @B00,@VRAM	RESET ENTRY
	0332	0000			
	0334	F120			
0557	0336	060B		DEC R11	DONE ALL ENTRIES ?
0558	0338	16F9		JNE SGRA2	N, LOOP
0559			*	SET UP PATTERN COLOUR TABLE	
0560	033A	020B		LI RB,CTBA1+>4000	REF PCT
	033C	6000			
0561	033E	06A0		BL @SENDAD	SEN ADDRESS TO VDP
	0340	037A			
0562	0342	020B		LI R11,3*256*8	RESET COUNT
	0344	1800			
0563			*		
0564	0346	D820	SGRA3	MOVB @FBCOL,@VRAM	SET ENTRY
	0348	02BA			
	034A	F120			
0565	034C	C20B		MOV RB,RB	WASTE SOME TIME
0566	034E	060B		DEC R11	DONE ALL ENTRIES ?
0567	0350	16FA		JNE SGRA3	N, LOOP
0568	0352	0720		SET0 @VMODE	FLAG IN GRAPHICS MODE
	0354	02B0			
0569			*		
0570	0356	D820		MOVB @S\$SM,@S\$SM	RELOAD SPRITE SIZE & MAG
	0358	0000			
	035A	0360			
0571	035C	06A0		BL @LOADER	RE-ENABLE DISPLAY
	035E	0368			
0572	0360	C0	S\$SM	BYTE >C0,>B0+R1	
0573	0362	0000		DATA 0	
0574	0364	0380		RTWP	EXIT
0575			*		
0576	0366	D0	BDO	BYTE >D0	
0577	0368			EVEN	

```

0579      *
0580      *   THIS ROUTINE LOADS THE VDP REGISTERS
0581      *   FROM AN INLINE DATA TABLE.
0582      *
0583 0368 D83B  LOADER MOVB *R11+,@VDPREG      WRITE REGISTER DATA
          036A F121
0584 036C 86DB      C      *R11,*R11      DUMY DELAY FOR VDP
0585 036E D83B      MOVB *R11+,@VDPREG      WRITE REGISTER NUMBER
          0370 F121
0586 0372 C6DB      MOV  *R11,*R11      END OF TABEL?
0587 0374 16F9      JNE  LOADER          N, LOOP
0588 0376 05CB      INCT R11             Y, SKIP TERMINATOR
0589 0378 045B      RT
0590      *
0591      *   THIS ROUTINE SENDS THE ADDRESS IN R8
0592      *   TO THE VDP, EXIT IS VIA A NORMAL 'RT'
0593      *
0594 037A 06C8  SENDAD SWPB R8              POSITION LSB
0595 037C D808      MOVB R8,@VDPREG        SEND LSB
          037E F121
0596 0380 06C8      SWPB R8              POSITION MSB
0597 0382 D808      MOVB R8,@VDPREG        SEND MSB
          0384 F121
0598 0386 86DB      C      *R11,*R11      DUMY DELAY FOR VDP
0599 0388 045B      RT                  RETURN TO CALLER
0600      END
NO ERRORS,      NO WARNINGS

```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.DSR02
OBJECT ACCESS NAME= ADHOC.OBJ.DSR02
LISTING ACCESS NAME= ADHOC.LST.DSR02
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

LINE	KEY	NAME
------	-----	------

0004	A	ADHOC.SRC.IOBITS =>ADHOC.SRC.IOBITS
------	---	--

0002
0003 0000
0004

IDT 'DSR02'
RORG
COPY ADHOC.SRC.IOBITS

```

A0002      *
A0003      * THIS MODULE IS INCLUDED IN SOURCE MODULES
A0004      * BY USING A 'COPY' DIRECTIVE.
A0005      *
A0006      *
A0007      *****
A0008      *
A0009      * CRU DEFINITIONS
A0010      *
A0011      *****
A0012      0000 PIO      EQU  >0000      PARALLEL I/O
A0013      * PARALLEL I/O - READ BITS
A0014      0000 KDS      EQU  PIO+0      KEYBOARD DATA STROBE
A0015      *          EQU  PIO+1      UNUSED
A0016      0002 D1$SIZ EQU  PIO+2      DS01 SIZE (1=8",0=5.25")
A0017      0003 D1$DEN EQU  PIO+3      DS01 DENSITY (0=SD,1=DD)
A0018      0004 FDCINT EQU  PIO+4      FDC INTERRUPT-
A0019      0005 KBDINT EQU  PIO+5      KBD INTERRUPT-
A0020      0006 VDPINT EQU  PIO+6      VDP INTERRUPT-
A0021      0007 BUSINT EQU  PIO+7      BUS INTERRUPT-
A0022      0008 KEYBRD EQU  PIO+8      KEYBOARD DATA (8 BITS)
A0023      * PARALLEL I/O - WRITE BITS
A0024      0000 CLKLED EQU  PIO+0      CLOCK LED
A0025      0001 KBDACK EQU  PIO+1      KEYBOARD ACKNOWLEDGE-
A0026      0002 BUSACK EQU  PIO+2      BUS INTERRUPT RESET-
A0027      0003 BTENBL EQU  PIO+3      BUS TIMEOUT FLAG ENABLE
A0028      0004 DRVSIZ EQU  PIO+4      DRIVE SIZE (1=8",0=5.25")
A0029      0005 ROMON  EQU  PIO+5      0=ROM ON,1=ROM OFF
A0030      0006 BELLON EQU  PIO+6      BELL ENABLE BIT
A0031      *          EQU  PIO+7      UNUSED
A0032      *
A0033      *          EQU  >0020      UNUSED
A0034      0040 EIA02  EQU  >0040      PRINTER HARDWARE BASE ADDRESS
A0035      *          EQU  >0060      UNUSED
A0036      *          EQU  >0080      UNUSED
A0037      *          EQU  >00A0      UNUSED
A0038      00C0 CASS02 EQU  >00C0      CASSETTE HARDWARE BASE ADDRESS
A0039      00E0 DMAC   EQU  >00E0      DMAC HARDWARE BASE ADDRESS
A0040      *
A0041      * RESERVED OFF-BOARD CRU LOCATIONS
A0042      0200 PRMPOP EQU  >200      PROM PROGRAMMER BOARD
A0043      * READ BITS
A0044      *          EQU  PRMPOP+0
A0045      *          EQU  PRMPOP+1
A0046      *          EQU  PRMPOP+2
A0047      *          EQU  PRMPOP+3
A0048      0204 PRORDY EQU  PRMPOP+4
A0049      0205 PGM    EQU  PRMPOP+5
A0050      0206 PGMPUL EQU  PRMPOP+6      * TEST
A0051      0207 V30    EQU  PRMPOP+7
A0052      0208 EDATA  EQU  PRMPOP+8
A0053      * WRITE BITS
A0054      0200 PRORST EQU  PRMPOP+0
A0055      0201 EPTYPE EQU  PRMPOP+1
A0056      0204 VCCON  EQU  PRMPOP+4
A0057      *PGM    EQU  PRMPOP+5
A0058      0206 PROERR EQU  PRMPOP+6
A0059      *          EQU  PRMPOP+7
A0060      *EDATA  EQU  PRMPOP+8
A0061      0210 EPADR  EQU  PRMPOP+16

```

```

A0062      *
A0063      *   CENTRONICS PARALLEL PRINTER PORT
A0064      *
A0065      *   BIT       0       1       2       3       4       5       6       7       8       9
A0066      *   READ
A0067      *   WRITE    D7      D6      D5      D4      D3      D2      D1      D0      D5
A0068      *                   (LSB)                                (MSB)
A0069      *
A0070      0400 PPRINT EQU  >400
A0071      *
A0072      0408 PPDS   EQU  PPRINT+8          DATA STROBE  ____|____|____
A0073      *
A0074      *
A0075      0409 PPBUSY EQU  PPRINT+9          BUSY          ____|____|____
A0076      *
A0077      *****
A0078      *
A0079      *   MEMORY MAPPED I/O DEFINITIONS
A0080      *
A0081      *****
A0082      F100 MAPPER EQU  >F100             MEMORY MAPPER LOCATION
A0083      F100 M$REG0 EQU  MAPPER+0          MAPPER REGISTERS 0..15
A0084      F102 M$REG1 EQU  MAPPER+2
A0085      F104 M$REG2 EQU  MAPPER+4
A0086      F106 M$REG3 EQU  MAPPER+6
A0087      F108 M$REG4 EQU  MAPPER+8
A0088      F10A M$REG5 EQU  MAPPER+10
A0089      F10C M$REG6 EQU  MAPPER+12
A0090      F10E M$REG7 EQU  MAPPER+14
A0091      F110 M$REG8 EQU  MAPPER+16
A0092      F112 M$REG9 EQU  MAPPER+18
A0093      F114 M$REGA EQU  MAPPER+20
A0094      F116 M$REGB EQU  MAPPER+22
A0095      F118 M$REGC EQU  MAPPER+24
A0096      F11A M$REGD EQU  MAPPER+26
A0097      F11C M$REGE EQU  MAPPER+28
A0098      F11E M$REGF EQU  MAPPER+30
A0099      *
A0100      F120 VDP     EQU  >F120
A0101      F120 VRAM    EQU  VDP+0            VDP VRAM ACCESS ADDRESS
A0102      F121 VDPREG  EQU  VDP+1            VDP REGISTER ACCESS ADDRESS
A0103      *
A0104      *   TEXT / GRAPHICS 1 MODE STORAGE
A0105      0400 NTBA    EQU  >400             NAME TABLE
A0106      0700 CTBA    EQU  >700             COLOUR TABLE
A0107      0800 PGBA    EQU  >800             PATTERN GENERATOR TABLE
A0108      0780 SNTBA   EQU  >780             SPRITE NAME TABLE
A0109      0000 SPGBA   EQU  >000             SPRITE PATTERN GENERATOR TBL.
A0110      *   GRAPHICS 2 MODE STORAGE
A0111      1800 NTBA1   EQU  >1800            NAME TABLE
A0112      2000 CTBA1   EQU  >2000            COLOUR TABLE
A0113      0000 PGTBA1  EQU  >0000            PATTERN GENERATOR TABLE
A0114      1800 SNTBA1  EQU  >1800            SPRITE NAME TABLE
A0115      3800 SPGBA1  EQU  >3800            SPRITE PATTERN GENERATOR TBL.
A0116      *
A0117      F140 FDC      EQU  >F140            TMS9909 FLOPPY DISC CONTROLLER
A0118      *
0005      *
0006      DEF  INT4PC, D$RTWP, C10
0007      REF  MREG13

```

0008		REF	BELCNT, RBUFE, CRDELY
0009		REF	F\$WHO, BOO, BOD, BFA, CCNT, FFLG
0010		REF	UNIT, B1B, MODE, STPY, PRDY, ESCFLG
0011		REF	DEVTBL, IDSTOR, FDCDON, MONTOP
0012		REF	CMOF\$, NEWP, PLF, C1, WPR1
0013	*		
0014	*	R0	
0015	*	R1	
0016	*	R2	
0017	*	R3	SCRATCH
0018	*	R4	'LOADPT' OUTPUT BUFFER LOAD PTR
0019	*	R5	
0020	*	R6	'INPTR' RECEIVE BUFFER LOAD PTR
0021	*	R7	'DSRCNT' ACTIVE DSR COUNTER
0022	*	R8	'RECPTR' RECEIVE UNLOAD POINTER
0023	*	R9	'TRAP4' BAD LEVEL 4 TRAP
0024	*	R10	POINTER TO UNLOAD POINTER STORAGE
0025	*	R12	CRUBASE
0026	*	R13-R15	RTWP VECTORS
0027	*		

```

0029 *****
0030 *
0031 *           LEVEL 4 INTERRUPT SERVICE ROUTINE
0032 *
0033 *****
0034 *
0035 *   THIS ROUTINE CHECKS FOR :-
0036 *       1.   KEYBOARD INTERRUPT.
0037 *       2.   FDC INTERRUPT
0038 *       3.   ALL 9902 INTERRUPTS
0039 *       4.   USER LEVEL 4 INTERRUPT TRAP
0040 *
0041 0000 020C   INT4PC LI   R12,2*KEYBRD   SET CRUBASE TO KEYBOARD DATA
0042 0002 0010
0043 *****
0044 *
0045 *           CHECK FOR KEYBOARD INTERRUPT
0046 *
0047 *****
0047 0004 1DF9   SBO   KBDACK-KEYBRD   MAKE SURE IT IS ENABLED !!!
0048 0006 1FFD   TB    KBDINT-KEYBRD   KEYBOARD?
0049 0008 1313   JEQ   FDC1             N, CHECK THE FDC
0050 000A 04CB   CLR   R11              Y, READY HOLDING REGISTER
0051 000C 360B   STCR  R11,8            GET THE KEYBOARD DATA
0052 000E 1EF9   SBZ   KBDACK-KEYBRD   ACKNOWLEDGE IT
0053 0010 1DF9   SBO   KBDACK-KEYBRD
0054 0012 028B   CI    R11,>FF00       GRAPHIC RUBOUT ?
0055 0014 FF00
0055 0016 1605   JNE   KBDIN           N, NORMAL KEYBOARD INPUT
0056 0018 0707   SETO  R7              Y, KILL DSRcnt FLAG
0057 001A C820   MOV   @C1,@UNIT       AND RESET UNIT FLAG TO 1
0058 001C 0000
0058 001E 0000
0058 0020 0380   RTWP                  AND EXIT
0059 *
0060 0022 C320   KBDIN  MOV   @PLF,R12   LOAD/SAVE?
0061 0024 0000
0061 0026 132A   JEQ   CHKESC          N, DECODE AS NORMAL
0062 0028 980B   CB    R11,@B1B       Y, ESCAPE?
0063 002A 0000
0063 002C 1339   JEQ   ABRTIO         Y, ABORT
0064 002E 0380   RTWP                  N, THROW IT AWAY
0065 *****
0066 *
0067 *           CHECK FOR FDC INTERRUPT
0068 *
0069 *****
0070 0030 1FFC   FDC1   TB    FDCINT-KEYBRD   WAS IT THE FDC?
0071 0032 1306   JEQ   POLLO2          N, POLL 9902'S
0072 0034 D820   MOV   @B00,@FDC       Y, RESET FDC
0073 0036 0000
0073 0038 F140
0073 003A 0720   SETO  @FDCDON         SET FDC DONE FLAG
0074 003C 0000
0074 003E 0380   RTWP
0075 *****
0076 *
0077 *           ALL 9902'S FOR INTERRUPT
0078 *
0079 *****

```

0080	0040	020A	POLL02	LI	R10, IOSTOR	POINT TO I/O STORAGE AREA
	0042	0000				
0081	0044	020B		LI	R11, DEVTBL	POINT TO DEVICE TABLE
	0046	0000				
0082	004B	0203		LI	R3, 16	DO 16 DEVICES
	004A	0010				
0083		004A	C10	EGU	\$-2	
0084	004C	C33B	NXTENT	MOV	*R11+, R12	GET DEVICE ENTRY
0085	004E	2320		COC	@C1, R12	IS IT A 9902?
	0050	001C				
0086	0052	1604		JNE	NXTDEV	N, DO NEXT DEVICE
0087	0054	024C		ANDI	R12, >7FFE	Y, ISOLATE CRUBASE
	0056	7FFE				
0088	005B	1F1F		TB	31	THIS ONE?
0089	005A	1307		JEQ	SERV	Y, SERVICE IT
0090	005C	8EBA	NXTDEV	C	*R10+, *R10+	SKIP STORAGE AREA
0091	005E	0603		DEC	R3	COUNT DEVICE
0092	0060	16F5		JNE	NXTENT	LOOP TILL ALL DEVICES DONE
0093			*			
0094			* NON-STANDARD DEVICE INTERRUPT			
0095			*			
0096	0062	C249		MOV	R9, R9	TRAP SET?
0097	0064	1301		JEQ	EXIT4	N, IGNORE IT
0098	0066	0419		BLWP	*R9	Y, GOTO ROUTINE
0099	0068	0380	EXIT4	RTWP		EXIT
0100			*			
0101			* SERVICE 9902 INTERRUPT			
0102			*			
0103	006A	1F10	SERV	TB	16	RECEIVE INTERRUPT
0104	006C	1639		JNE	TRYTIM	N, TRY NEXT INTERRUPT SOURCE
0105	006E	360B		STCR	R11, 8	Y, GET CHARACTER
0106	0070	1D12		SBO	18	RESET RBRL
0107	0072	1F09		TB	9	WAS THERE AN ERROR ?
0108	0074	1340		JEQ	QUIT	Y, IGNORE IT !
0109	0076	C2A0		MOV	@PLF, R10	N, TEST LOAD/SAVE ?
	007B	0024				
0110	007A	1626		JNE	LOADIN	Y, DO SPECIAL CHECKS
0111			*			
0112	007C	C320	CHKESC	MOV	@ESCFLG, R12	RUNING WITH NOESC ?
	007E	0000				
0113	0080	1627		JNE	SERV1	Y, DONT CHECK FOR ESCAPE
0114	0082	980B		CB	R11, @B1B	N, WAS IT AN ESCAPE?
	0084	002A				
0115	0086	1624		JNE	SERV1	N, BUFFER IT
0116	008B	04E0		CLR	@FFLG	Y, RESET FORMATTING FLAG
	008A	0000				
0117			*			
0118	008C	C020		MOV	@F\$WHO, R0	WHO IS IN CONTROL ?
	008E	0000				
0119	0090	130C		JEQ	ESCO	BASIC, TAKE BASIC EXIT
0120	0092	0200		LI	R0, MREG13	LOAD POINTER TO R13
	0094	0000				
0121	0096	CC0D		MOV	R13, *R0+	COPY CONTEXT
0122	009B	CC0E		MOV	R14, *R0+	COPY CONTEXT
0123	009A	CC0F		MOV	R15, *R0+	COPY CONTEXT
0124	009C	0460		B	@MONTOP	MONITOR, EXIT TO MONTOP
	009E	0000				
0125			*			
0126			*	ABORT !	IF PLF > +1 THEN DO A NEW	
0127			*			

0128	00A0 C820	ABRTID MOV	@C1,@UNIT	<>0 SET UNIT TO 1
	00A2 0050			
	00A4 001E			
0129	00A6 060C	DEC	R12	ABORT ?
0130	00A8 150C	JGT	DONEW	+VE, DO A 'NEW'
0131		*		
0132	00AA 020E	ESCO	LI R14,STPY	DEFAULT TO 'STOP'
	00AC 0000			
0133	00AE C320	MOV	@MODE,R12	KEYBOARD MODE?
	00B0 0000			
0134	00B2 1602	JNE	ESC2	N, EXIT SET UP
0135	00B4 020E	ESC1	LI R14,PRDY	Y, EXIT TO PRDY
	00B6 0000			
0136	00B8 020D	ESC2	LI R13,WPR1	SET MAIN WORKSPACE
	00BA 0000			
0137	00BC 03C0	CKOF		MAKE SURE MAPPER IS OFF
0138	00BE 0000	DATA	CMDF#	TURN OFF THE TAPE
0139	00C0 0380	RTWP		
0140	00C2 020E	DONEW	LI R14,NEWP	DO A NEW
	00C4 0000			
0141	00C6 10F8	JMP	ESC2	
0142		*		
0143		*	9902	CHARACTER RECEIVED DURING LOAD/SAVE
0144		*		
0145	00CB 1116	LOADIN	JLT QUIT	SAVE - THROW IT AWAY
0146	00CA 028C	CI	R12,2*CASS02	LOAD - FROM CASSETTE ?
	00CC 0180			
0147	00CE 1613	JNE	QUIT	N, THROW IT THEN !
0148		*		Y, STORE IT
0149		*		
0150	00D0 0286	SERV1	CI R6,RBUFE	OFF BUFFER?
	00D2 0000			
0151	00D4 1402	JHE	BEEP	Y, THROW CHARACTER AWAY
0152	00D6 DD8B	MOVB	R11,*R6+	N, SAVE IT
0153	00D8 0380	RTWP		TERMINATE
0154	00DA 0720	BEEP	SET0 @BELCNT	BEEP !!
	00DC 0000			
0155	00DE 0380	RTWP		
0156		*		
0157	00E0 1F13	TRYTIM	TB 19	TIMER?
0158	00E2 160A	JNE	TRYDSC	N, TRY NEXT INTERRUPT SOURCE
0159	00E4 1E14	SBZ	20	Y, DISABLE TIMER INTERRUPTS
0160	00E6 1F17	TB	23	XSRE ?
0161	00E8 1605	JNE	ENBTIM	N, DON'T START COUNT YET
0162	00EA 1F16	TB	22	Y, XBRE ?
0163	00EC 1603	JNE	ENBTIM	N, DON'T COUNT YET
0164	00EE 062A	DEC	@2(10)	DECREMENT CR DELAY
	00F0 0002			
0165	00F2 1111	JLT	DONE	TIME UP, CONTINUE OUTPUT
0166	00F4 1D14	ENBTIM	SBO 20	RE-ENABLE TIMER
0167	00F6 0380	QUIT	RTWP	EXIT
0168		*		
0169	00F8 1F14	TRYDSC	TB 20	DSCH INTERRUPT
0170	00FA 1606	JNE	TRYXMT	N, TRY TRANSMITTER INTERRUPT
0171	00FC 1E15	SBZ	21	Y, DISABLE DSCH INTERRUPTS
0172	00FE 1007	JMP	SETREG	SET UP PTRS AND CHECK DSR
0173		*		
0174	0100 1F18	DSR	TB 27	ONLINE?
0175	0102 1313	JEQ	OUTPUT	Y, OUT CHARACTER
0176	0104 1D15	SBO	21	N, ENABLE DSCH INTERRUPTS

0177 0106 0380	RTWP	EXIT
0178	*	
0179	*	
0180 0108 1F16	TRYXMT TB 22	XBRE INTERRUPT ?
0181 010A 16F5	JNE QUIT	N, GLITCH
0182 010C 1E13	SBZ 19	DISABLE XBRE INTERRUPTS
0183 010E C0CB	SETREG MOV R11,R3	Y, SAVE DEVTBL INDEX REG.
0184 0110 C2EA	MOV @2(10),R11	CR DELAY SET?
0112 0002		
0185 0114 1505	JGT SETTIM	Y, SET TIMER
0186	*	N, FALL THROUGH TO DONE
0187 0116 C2DA	DONE MOV *R10,R11	GET UNLOAD POINTER
0188 0118 810B	C R11,R4	DONE
0189 011A 12F2	JLE DSR	N, CHECK FOR ONLINE
0190 011C 0607	DEC R7	Y, FLAG DSR TERMINATED
0191 011E 0380	D\$RTWP RTWP	EXIT
0192	*	
0193 0120 1D0D	SETTIM SBO 13	SET LDIR FLAG
0194 0122 3220	LDCR @BFA,8	LOAD TIMER
0124 0000		
0195 0126 1D14	SBO 20	Y, START TIMER
0196 0128 0380	RTWP	EXIT
0197	*	
0198 012A 1D10	OUTPUT SBO 16	RTS ON
0199 012C 321B	LDCR *R11,R8	SEND CHARACTER
0200 012E 02BC	CI R12,2*CASS02	IS IT THE CASSETTE PORT ?
0130 0180		
0201 0132 1301	JEQ OPT1	Y, LEAVE RTS ALONE
0202 0134 1E10	SBZ 16	N, TURN RTS OFF
0203 0136 983B	OPT1 CB *R11+,@B0D	CR ?
0138 0000		
0204 013A 1608	JNE XMTEN	N, UPDATE POINTER
0205 013C 04E0	CLR @CCNT	Y, RESET COLUMN COUNTER
013E 0000		
0206 0140 C0E3	MOV @-2(R3),R3	FETCH DEVICE ENTRY
0142 FFFE		
0207 0144 1503	JGT XMTEN	+VE, NO CR DELAY
0208 0146 CAA0	MOV @CRDELY,@2(10)	Y, LOAD CR DELAY COUNT
0148 0000		
014A 0002		
0209 014C C68B	XMTEN MOV R11,*R10	UPDATE UNLOAD POINTER
0210 014E 1D13	SBO 19	ENABLE XBRE INTERRUPTS
0211 0150 0380	RTWP	EXIT
0212	END	

NO ERRORS, NO WARNINGS

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.CENTRON
OBJECT ACCESS NAME= ADHOC.OBJ.CENTRON
LISTING ACCESS NAME= ADHOC.LST.CENTRON
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

LINE	KEY	NAME
------	-----	------

0003	A	ADHOC.SRC.IOBITS =>ADHOC.SRC.IOBITS
------	---	--

0002
0003

IDT 'CENTRON'
COPY ADHOC.SRC. IOBITS

```

A0002      *
A0003      * THIS MODULE IS INCLUDED IN SOURCE MODULES
A0004      * BY USING A 'COPY' DIRECTIVE.
A0005      *
A0006      *
A0007      *****
A0008      *
A0009      * CRU DEFINITIONS
A0010      *
A0011      *****
A0012      0000  PIO      EQU  >0000      PARALLEL I/O
A0013      * PARALLEL I/O - READ BITS
A0014      0000  KDS      EQU  PIO+0      KEYBOARD DATA STROBE
A0015      *          EQU  PIO+1      UNUSED
A0016      0002  D1$SIZ  EQU  PIO+2      DS01 SIZE (1=8",0=5.25")
A0017      0003  D1$DEN  EQU  PIO+3      DS01 DENSITY (0=SD,1=DD)
A0018      0004  FDCINT  EQU  PIO+4      FDC INTERRUPT-
A0019      0005  KBDINT  EQU  PIO+5      KBD INTERRUPT-
A0020      0006  VDPINT  EQU  PIO+6      VDP INTERRUPT-
A0021      0007  BUSINT  EQU  PIO+7      BUS INTERRUPT-
A0022      0008  KEYBRD  EQU  PIO+8      KEYBOARD DATA (8 BITS)
A0023      * PARALLEL I/O - WRITE BITS
A0024      0000  CLKLED  EQU  PIO+0      CLOCK LED
A0025      0001  KBDACK  EQU  PIO+1      KEYBOARD ACKNOWLEDGE-
A0026      0002  BUSACK  EQU  PIO+2      BUS INTERRUPT RESET-
A0027      0003  BTENBL  EQU  PIO+3      BUS TIMEOUT FLAG ENABLE
A0028      0004  DRVSIZ  EQU  PIO+4      DRIVE SIZE (1=8",0=5.25")
A0029      0005  ROMON   EQU  PIO+5      0=ROM ON,1=ROM OFF
A0030      0006  BELLON  EQU  PIO+6      BELL ENABLE BIT
A0031      *          EQU  PIO+7      UNUSED
A0032      *
A0033      *          EQU  >0020      UNUSED
A0034      0040  EIA02   EQU  >0040      PRINTER HARDWARE BASE ADDRESS
A0035      *          EQU  >0060      UNUSED
A0036      *          EQU  >0080      UNUSED
A0037      *          EQU  >00A0      UNUSED
A0038      00C0  CASS02  EQU  >00C0      CASSETTE HARDWARE BASE ADDRESS
A0039      00E0  DMAC    EQU  >00E0      DMAC HARDWARE BASE ADDRESS
A0040      *
A0041      * RESERVED OFF-BOARD CRU LOCATIONS
A0042      0200  PRMPOP  EQU  >200      PROM PROGRAMMER BOARD
A0043      * READ BITS
A0044      *          EQU  PRMPOP+0
A0045      *          EQU  PRMPOP+1
A0046      *          EQU  PRMPOP+2
A0047      *          EQU  PRMPOP+3
A0048      0204  PRORDY  EQU  PRMPOP+4
A0049      0205  PGM     EQU  PRMPOP+5
A0050      0206  PGMFUL  EQU  PRMPOP+6      * TEST
A0051      0207  V30     EQU  PRMPOP+7
A0052      0208  EDATA   EQU  PRMPOP+8
A0053      * WRITE BITS
A0054      0200  PRORST  EQU  PRMPOP+0
A0055      0201  EPTYPE  EQU  PRMPOP+1
A0056      0204  VCCON   EQU  PRMPOP+4
A0057      *PGM        EQU  PRMPOP+5
A0058      0206  PROERR  EQU  PRMPOP+6
A0059      *          EQU  PRMPOP+7
A0060      *EDATA      EQU  PRMPOP+8
A0061      0210  EPADR   EQU  PRMPOP+16

```

```

A0062      *
A0063      *   CENTRONICS PARALLEL PRINTER PORT
A0064      *
A0065      *   BIT           0       1       2       3       4       5       6       7       8       9
A0066      *   READ
A0067      *   WRITE      D7      D6      D5      D4      D3      D2      D1      D0      D5
A0068      *                               (LSB)                               (MSB)
A0069      *
A0070      0400 PPRINT EQU  >400
A0071      *
A0072      0408 PPDS  EQU  PPRINT+8          DATA STROBE  ____|____|____
A0073      *
A0074      *
A0075      0409 PPBUSY EQU  PPRINT+9          BUSY          ____|____|____
A0076      *
A0077      *****
A0078      *
A0079      *   MEMORY MAPPED I/O DEFINITIONS
A0080      *
A0081      *****
A0082      F100 MAPPER EQU  >F100          MEMORY MAPPER LOCATION
A0083      F100 M$REG0 EQU  MAPPER+0          MAPPER REGISTERS 0..15
A0084      F102 M$REG1 EQU  MAPPER+2
A0085      F104 M$REG2 EQU  MAPPER+4
A0086      F106 M$REG3 EQU  MAPPER+6
A0087      F108 M$REG4 EQU  MAPPER+8
A0088      F10A M$REG5 EQU  MAPPER+10
A0089      F10C M$REG6 EQU  MAPPER+12
A0090      F10E M$REG7 EQU  MAPPER+14
A0091      F110 M$REG8 EQU  MAPPER+16
A0092      F112 M$REG9 EQU  MAPPER+18
A0093      F114 M$REGA EQU  MAPPER+20
A0094      F116 M$REGB EQU  MAPPER+22
A0095      F118 M$REGC EQU  MAPPER+24
A0096      F11A M$REGD EQU  MAPPER+26
A0097      F11C M$REGE EQU  MAPPER+28
A0098      F11E M$REGF EQU  MAPPER+30
A0099      *
A0100      F120 VDP      EQU  >F120
A0101      F120 VRAM     EQU  VDP+0          VDP VRAM ACCESS ADDRESS
A0102      F121 VDPREG   EQU  VDP+1          VDP REGISTER ACCESS ADDRESS
A0103      *
A0104      *   TEXT / GRAPHICS 1 MODE STORAGE
A0105      0400 NTBA     EQU  >400          NAME TABLE
A0106      0700 CTBA     EQU  >700          COLOUR TABLE
A0107      0800 PGBA     EQU  >800          PATTERN GENERATOR TABLE
A0108      0780 SNTBA    EQU  >780          SPRITE NAME TABLE
A0109      0000 SPGBA    EQU  >000          SPRITE PATTERN GENERATOR TBL.
A0110      *   GRAPHICS 2 MODE STORAGE
A0111      1800 NTBA1    EQU  >1800          NAME TABLE
A0112      2000 CTBA1    EQU  >2000          COLOUR TABLE
A0113      0000 PGTBA1   EQU  >0000          PATTERN GENERATOR TABLE
A0114      1800 SNTBA1   EQU  >1800          SPRITE NAME TABLE
A0115      3800 SPGBA1   EQU  >3800          SPRITE PATTERN GENERATOR TBL.
A0116      *
A0117      F140 FDC       EQU  >F140          TMS9909 FLOPPY DISC CONTROLLER
A0118      *
0004      *
0005      DEF  CENTRO
0006      *

```

```
0007                                REF WP9928
0008                                *
0009                                *      R9  = OUTPUT BUFFER START
0010                                *      R10 = OUTPUT BUFFER END
0011                                *      R11 = POINTER TO LOCAL STORAGE
0012                                *      R13-15 RETURN CONTEXT
0013                                *
0014 0000 0000  CENTRO DATA WP9928,$+2
0015 0004 020C      LI  R12,2*PPRINT      POINT TO THE PARALLEL PRINTER
      0006 0800
0016                                *
0017 0008 8289  DONE  C      R9,R10      DONE?
0018 000A 1B06      JH  EXIT
0019                                *
0020 000C 1F09  OUTCHR TB  PPBUSY-PPRINT  BUSY?
0021 000E 13FE      JEG  OUTCHR          Y, LOOP
0022                                *
0023 0010 3239      LDCR *R9+,8          OUTPUT DATA
0024 0012 1D08      SBO  PPDS-PPRINT     DATA STROBE =1
0025 0014 1E08      SBZ  PPDS-PPRINT     DATA STROBE =0
0026 0016 10F8      JMP  DONE
0027                                *
0028 0018 0380  EXIT  RTWP      RETURN TO CALLER
NO ERRORS,      NO WARNINGS
```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.VDPDSR
OBJECT ACCESS NAME= ADHOC.OBJ.VDPDSR
LISTING ACCESS NAME= ADHOC.LST.VDPDSR
ERROR ACCESS NAME=
OPTIONS= BUNLST, DUNLST, MUNLST
MACRO LIBRARY PATHNAME=

LINE	KEY	NAME
------	-----	------

0027	A	ADHOC.SRC.IOBITS =>ADHOC.SRC.IOBITS
------	---	--

```

0002          IDT  'VDPDSR'
0003          DEF  VDPDSR,D40
0004          REF  UNPACK,SENDAD,JMPRO,B20
0005          REF  FBCOL,STXT,B7F,XLOC,YLOC,VMODE
0006          REF  BITMAP,WP9928,CCSAVE,TMPBUF,BELCNT
0007          REF  CURFLG,CCNT,GCLEAR
0008          *
0009          *  SCREEN EQUATES
0010          *
0011          0027  TRHC   EQU  39          TOP RIGHT HAND CORNER
0012          0000  TLHC   EQU  0          TOP LEFT HAND CORNER
0013          0398  BLHC   EQU  920        BOTTOM LEFT HAND CORNER
0014          03BF  BRHC   EQU  959        BOTTOM RIGHT HAND CORNER
0015          03C0  SCRSIZ EQU  BRHC-TLHC+1  SCREEN SIZE
0016          *
0017          2F80  ERROR  EQU  >2F80
0018          2FA0  ERROR2 EQU  ERROR+>20
0019          *
0020          *          INTERNAL 9995 FLAGS USED
0021          *
0022          *          WHEN BIT IS SET
0023          *
0024          0005  F$SHOW EQU  5          SHOW ALL CONTROL CHARACTERS
0025          0006  F$ROLL EQU  6          INHIBIT SCROLL
0026          *
0027          COPY ADHOC.SRC.IOBITS
  
```

```

A0002      *
A0003      * THIS MODULE IS INCLUDED IN SOURCE MODULES
A0004      * BY USING A 'COPY' DIRECTIVE.
A0005      *
A0006      *
A0007      *****
A0008      *
A0009      * CRU DEFINITIONS
A0010      *
A0011      *****
A0012      0000  PID      EQU  >0000      PARALLEL I/O
A0013      * PARALLEL I/O - READ BITS
A0014      0000  KDS      EQU  PID+0      KEYBOARD DATA STROBE
A0015      *          EQU  PID+1      UNUSED
A0016      0002  D1$SIZ  EQU  PID+2      DS01 SIZE (1=8",0=5.25")
A0017      0003  D1$DEN  EQU  PID+3      DS01 DENSITY (0=SD,1=DD)
A0018      0004  FDCINT  EQU  PID+4      FDC INTERRUPT-
A0019      0005  KBDINT  EQU  PID+5      KBD INTERRUPT-
A0020      0006  VDPINT  EQU  PID+6      VDP INTERRUPT-
A0021      0007  BUSINT  EQU  PID+7      BUS INTERRUPT-
A0022      0008  KEYBRD  EQU  PID+8      KEYBOARD DATA (8 BITS)
A0023      * PARALLEL I/O - WRITE BITS
A0024      0000  CLKLED  EQU  PID+0      CLOCK LED
A0025      0001  KBDACK  EQU  PID+1      KEYBOARD ACKNOWLEDGE-
A0026      0002  BUSACK  EQU  PID+2      BUS INTERRUPT RESET-
A0027      0003  BTENBL  EQU  PID+3      BUS TIMEOUT FLAG ENABLE
A0028      0004  DRVSIZ  EQU  PID+4      DRIVE SIZE (1=8",0=5.25")
A0029      0005  ROMDN   EQU  PID+5      0=ROM ON,1=ROM OFF
A0030      0006  BELLON  EQU  PID+6      BELL ENABLE BIT
A0031      *          EQU  PID+7      UNUSED
A0032      *
A0033      *          EQU  >0020      UNUSED
A0034      0040  EIA02   EQU  >0040      PRINTER HARDWARE BASE ADDRESS
A0035      *          EQU  >0060      UNUSED
A0036      *          EQU  >0080      UNUSED
A0037      *          EQU  >00A0      UNUSED
A0038      00C0  CASS02  EQU  >00C0      CASSETTE HARDWARE BASE ADDRESS
A0039      00E0  DMAC    EQU  >00E0      DMAC HARDWARE BASE ADDRESS
A0040      *
A0041      * RESERVED OFF-BOARD CRU LOCATIONS
A0042      0200  PRMPOP  EQU  >200      PROM PROGRAMMER BOARD
A0043      * READ BITS
A0044      *          EQU  PRMPOP+0
A0045      *          EQU  PRMPOP+1
A0046      *          EQU  PRMPOP+2
A0047      *          EQU  PRMPOP+3
A0048      0204  PRORDY  EQU  PRMPOP+4
A0049      0205  PGM     EQU  PRMPOP+5
A0050      0206  PGMFUL  EQU  PRMPOP+6      * TEST
A0051      0207  V30     EQU  PRMPOP+7
A0052      0208  EDATA   EQU  PRMPOP+8
A0053      * WRITE BITS
A0054      0200  PRORST  EQU  PRMPOP+0
A0055      0201  EPTYPE  EQU  PRMPOP+1
A0056      0204  VCCON   EQU  PRMPOP+4
A0057      *PGM        EQU  PRMPOP+5
A0058      0206  PRDERR  EQU  PRMPOP+6
A0059      *          EQU  PRMPOP+7
A0060      *EDATA      EQU  PRMPOP+8
A0061      0210  EPADR   EQU  PRMPOP+16


```

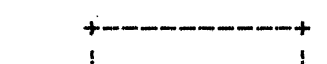


```

A0062      *
A0063      *   CENTRONICS PARALLEL PRINTER PORT
A0064      *
A0065      *   BIT       0       1       2       3       4       5       6       7       8       9
A0066      *   READ
A0067      *   WRITE    D7      D6      D5      D4      D3      D2      D1      D0      D5
A0068      *                   (LSB)                                (MSB)
A0069      *
A0070      0400 PPRINT EQU  >400
A0071      *
A0072      0408 PPDS   EQU  PPRINT+8
A0073      *
A0074      *
A0075      0409 PPBUSY EQU  PPRINT+9
A0076      *
A0077      *****
A0078      *
A0079      *   MEMORY MAPPED I/O DEFINITIONS
A0080      *
A0081      *****
A0082      F100 MAPPER EQU  >F100
A0083      F100 M$REG0 EQU  MAPPER+0
A0084      F102 M$REG1 EQU  MAPPER+2
A0085      F104 M$REG2 EQU  MAPPER+4
A0086      F106 M$REG3 EQU  MAPPER+6
A0087      F108 M$REG4 EQU  MAPPER+8
A0088      F10A M$REG5 EQU  MAPPER+10
A0089      F10C M$REG6 EQU  MAPPER+12
A0090      F10E M$REG7 EQU  MAPPER+14
A0091      F110 M$REG8 EQU  MAPPER+16
A0092      F112 M$REG9 EQU  MAPPER+18
A0093      F114 M$REGA EQU  MAPPER+20
A0094      F116 M$REGB EQU  MAPPER+22
A0095      F118 M$REGC EQU  MAPPER+24
A0096      F11A M$REGD EQU  MAPPER+26
A0097      F11C M$REGE EQU  MAPPER+28
A0098      F11E M$REGF EQU  MAPPER+30
A0099      *
A0100      F120 VDP     EQU  >F120
A0101      F120 VRAM    EQU  VDP+0
A0102      F121 VDPREG EQU  VDP+1
A0103      *
A0104      *   TEXT / GRAPHICS 1 MODE STORAGE
A0105      0400 NTBA    EQU  >400
A0106      0700 CTBA    EQU  >700
A0107      0800 PGBA    EQU  >800
A0108      0780 SNTBA   EQU  >780
A0109      0000 SPGBA   EQU  >000
A0110      *   GRAPHICS 2 MODE STORAGE
A0111      1800 NTBA1   EQU  >1800
A0112      2000 CTBA1   EQU  >2000
A0113      0000 PGTBA1  EQU  >0000
A0114      1800 SNTBA1  EQU  >1800
A0115      3800 SPGBA1  EQU  >3800
A0116      *
A0117      F140 FDC      EQU  >F140
A0118      *

```

DATA STROBE 

BUSY 

MEMORY MAPPER LOCATION
MAPPER REGISTERS 0..15

VDP VRAM ACCESS ADDRESS
VDP REGISTER ACCESS ADDRESS

NAME TABLE
COLOUR TABLE
PATTERN GENERATOR TABLE
SPRITE NAME TABLE
SPRITE PATTERN GENERATOR TBL.

NAME TABLE
COLOUR TABLE
PATTERN GENERATOR TABLE
SPRITE NAME TABLE
SPRITE PATTERN GENERATOR TBL.

TMS9909 FLOPPY DISC CONTROLLER

```

0029 *****
0030 *
0031 *          GRAPHICS MODE
0032 *
0033 *****
0034 0000 D220  GMODE  MOVB @YLOC,R8      PICK UP Y LOCATION
      0002 0000
0035 0004 0988      SRL  R8,8          PUT IN LS BYTE
0036 0006 0228      AI   R8,>0007      ALIGN WITH CHARACTER CELL
      0008 0007
0037 000A 0938      SRL  R8,3          FORM Y CELL£
0038 *
0039 000C D1E0      MOVB @XLOC,R7      PICK UP X LOCATION
      000E 0000
0040 0010 0987      SRL  R7,8          PUT IN LS BYTE
0041 0012 0227      AI   R7,>0007      ALIGN WITH CHARACTER CELL
      0014 0007
0042 0016 0937      SRL  R7,3          FORM X CELL£
0043 0018 0287      CI   R7,31         OFF RIGHT OF SCREEN ?
      001A 001F
0044 001C 1202      JLE  GM1          N, LEAVE IT
0045 001E 04C7      CLR  R7          Y, RESET X CELL£
0046 0020 0588      INC  R8          NEXT Y CELL
0047 *
0048 0022 0288  GM1  CI   R8,23        OFF BOTTOM OF SCREEN ?
      0024 0017
0049 0026 1201      JLE  GM2          N, LEAVE IT
0050 0028 04C8      CLR  R8          Y, RESET Y CELL£
0051 *
0052 002A 0A58  GM2  SLA  R8,5          R8=32*Y CELL£
0053 002C A207      A    R7,R8        R8=X CELL£ + (32*Y CELL£)
0054 002E 0A38      SLA  R8,3          R8=8*[X CELL£ + (32*Y CELL£)]
0055 0030 0228  GM3  AI   R8,PGTBA1+>4000  ADD IN PGT BASE ADDRESS
      0032 4000

```

```

0057 *****
0058 *
0059 *          CHECK FOR END OF TEXT
0060 *
0061 *****
0062 *
0063 *          RB CONTENTS :-
0064 *
0065 * BIT ==>      0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
0066 *
0067 * USE ==>      0  W  0  R  R  R  R  R  C  C  C  C  C  0  0  0
0068 *
0069 *
0070 *      W : VRAM WRITE BIT
0071 *      R : ROW ADDRESS      0..23
0072 *      C : COLUMN ADDRESS   0..31
0073 *
0074 0034 0694 GDONE1 BL *R4          SEND ADDRESS TO VDP
0075 *
0076 0036 8289 GDONE C R9,R10        REACHED THE END OF MSG ?
0077 0038 1B27 JH GREXIT            Y, EXIT
0078 003A D039 MOVB *R9+,R0        N, GET NEXT CHARACTER
0079 003C 9800 CB R0,@B20          CONTROL ?
0080 003E 0000
0080 0040 1A36 JL GCNTRL          Y, HANDLE IT.
0081 *
0082 0042 0420 BLWP @UNPACK        UNPACK THE BIT PATTERN
0083 0044 0000
0083 0046 0201 LI R1,BITMAP       POINT TO 'BITMAP'
0084 0048 0000
0084 004A 0200 LI R0,B           DO 8 BYTES
0085 004C 0008
0085 004E C080 MOV R0,R2         SAVE IT
0086 0050 A200 A R0,R8          READY FOR NEXT CHARACTER
0087 *
0088 0052 D831 CWRITE MOVB *R1+,@VRAM COPY OVER BYTE
0089 0054 F120
0089 0056 0600 DEC R0           DONE ?
0090 0058 16FC JNE CWRITE        N, LOOP
0091 * NOW UPDATE THE COLOUR TABLE
0092 005A 0228 AI R8,(CTBA1-PGTBA1)-8 REF COLOUR TABLE
0093 005C 1FF8
0093 005E 0694 BL *R4          SEND ADDRESS TO VDP
0094 0060 D820 UPCTBL MOVB @FBCOL,@VRAM WRITE THE COLOUR INFORMATION
0095 0062 0000
0095 0064 F120
0095 0066 0602 DEC R2           DONE?
0096 0068 16FB JNE UPCTBL        N, LOOP
0097 006A 0228 AI R8,-((CTBA1-PGTBA1)-8) Y, RESTORE CURSOR ADDRESS
0098 006C E008
0098 006E 0694 BL *R4          RELOAD VDP ADDRESS
0099 *
0100 0070 0288 CHKBOT CI R8,PGTBA1+(32*24*8)+>4000 OFF END OF TABLE ?
0101 0072 5800
0101 0074 1AE0 JL GDONE          N, CONTINUE
0102 0076 0228 AI R8,-(32*24*8)   Y, BACKUP A SCREEN
0103 0078 E800
0103 007A 10DC JMP GDONE1          AND RESEND ADDRESS
0104 *
0105 *          CURSOR ON/OFF CONTROL

```

```
0106      *
0107 007C 04E0 GFS CLR @CURFLG      ENABLE CURSOR
      007E 0000
0108 0080 10DA      JMP GDONE      AND CONTINUE
0109      *
0110 0082 0720 GGS SETD @CURFLG      DISABLE CURSOR
      0084 007E'
0111 0086 10D7      JMP GDONE      AND CONTINUE
0112 008B
0113      *****
0114      *
0115      *      EXIT FROM GRAPHICS MODE      *
0116      *
0117      *****
0118 008B 024B GREXIT ANDI RB,>1FFB      KILL PIXEL & WRITE BITS
      008A 1FFB
0119 008C 06CB      SWPB RB      POSITION X
0120 008E D80B      MOVB RB,@XLOC      UPDATE Y
      0090 000E'
0121 0092 0ABB      SLA RB,11      POSITION Y
0122 0094 D80B      MOVB RB,@YLOC      UPDATE Y
      0096 0002'
0123 009B 0380      RTWP
```

```

0125 *****
0126 *
0127 *
0128 *
0129 *****
0130 009A 0000 VDPDSR DATA WP9928, $+2 9928DSR ENTRY VECTOR
0131 009E 020C 080C LI R12, >1EE0 REF INTERNAL FLAGS
0132 00A2 0204 00A0 1EE0 LI R4, SENDAD05F2 REF 'SENDAD'
0133 00A6 C020 00A8 0000 MOV @VMODE, R0 WHAT MODE ?
0134 00AA 16AA JNE GMODE GRAPH MODE
0135 00AC 1030 JMP TMODE TEXT MODE
0136 *****
0137 *
0138 *
0139 *
0140 *****
0141 00AE 06A0 GCNTRL BL @JMPRO ; DO JUMP ON R0 (USES R1, R2)
0142 00B0 0000
0143 00B2 0D GJUMP BYTE GHT-GJUMP/2, >09 HT - CURSOR RIGHT
0144 00B4 11 BYTE GBS-GJUMP/2, >0B BS - CURSOR LEFT
0145 00B6 1A BYTE GLF-GJUMP/2, >0A LF - CURSOR DOWN
0146 00B8 1D BYTE GVT-GJUMP/2, >0B VT - CURSOR UP
0147 00BA 22 BYTE GFF-GJUMP/2, >0C FF - CLEAR SCREEN & HOME
0148 00BC 27 BYTE GCR-GJUMP/2, >0D CR - CURSOR BEGINING OF LINE
0149 00BE E5 BYTE GFS-GJUMP/2, >1C FS - CURSOR ON
0150 00C0 E8 BYTE GGS-GJUMP/2, >1D GS - CURSOR OFF
0151 00C2 24 BYTE GRS-GJUMP/2, >1E RS - CURSOR HOME
0152 00C4 0000 DATA 0
0153 00C6 0720 SETO @BELCNT ILLEGAL, SET BELL COUNTER
0154 00C8 0000
0155 00CA 10B5 JMP GDONE AND LOOP
0156 *****
0157 *
0158 *
0159 *
0160 *****
0161 00CC 022B GHT AI RB, B NEXT CHARACTER
0162 00CE 000B
0163 00D0 0694 GHT1 BL *R4 POINT VDP TO IT
0164 00D2 10CE JMP CHKBOT AND CHECK IT
0165 *****
0166 *
0167 *
0168 *
0169 *****
0170 00D4 022B GBS AI RB, -B BACKUP TO PRIOR CHARACTER
0171 00D6 FFFB
0172 00D8 0694 GBS1 BL *R4 SEND IT TO VDP
0173 00DA 028B CHKTOP CI RB, PGTBA1+>4000 OFF TOP ?
0174 00DC 4000
0175 00DE 14AB JHE GDONE N, CONTINUE
0176 00E0 022B AI RB, (32*24*B) Y, ADD IN A SCREEN
0177 00E2 1800
0178 00E4 10A7 JMP GDONE1 & RESEND IT
0179 *****
0180 *
0181 *
0182 *
0183 *****
0184 *
0185 *
0186 *
0187 *****
0188 *
0189 *
0190 *
0191 *****
0192 *
0193 *
0194 *
0195 *****
0196 *
0197 *
0198 *
0199 *****
0200 *
0201 *
0202 *
0203 *****
0204 *
0205 *
0206 *
0207 *****
0208 *
0209 *
0210 *
0211 *****
0212 *
0213 *
0214 *
0215 *****
0216 *
0217 *
0218 *
0219 *****
0220 *
0221 *
0222 *
0223 *****
0224 *
0225 *
0226 *
0227 *****
0228 *
0229 *
0230 *
0231 *****
0232 *
0233 *
0234 *
0235 *****
0236 *
0237 *
0238 *
0239 *****
0240 *
0241 *
0242 *
0243 *****
0244 *
0245 *
0246 *
0247 *****
0248 *
0249 *
0250 *
0251 *****
0252 *
0253 *
0254 *
0255 *****
0256 *
0257 *
0258 *
0259 *****
0260 *
0261 *
0262 *
0263 *****
0264 *
0265 *
0266 *
0267 *****
0268 *
0269 *
0270 *
0271 *****
0272 *
0273 *
0274 *
0275 *****
0276 *
0277 *
0278 *
0279 *****
0280 *
0281 *
0282 *
0283 *****
0284 *
0285 *
0286 *
0287 *****
0288 *
0289 *
0290 *
0291 *****
0292 *
0293 *
0294 *
0295 *****
0296 *
0297 *
0298 *
0299 *****
0300 *
0301 *
0302 *
0303 *****
0304 *
0305 *
0306 *
0307 *****
0308 *
0309 *
0310 *
0311 *****
0312 *
0313 *
0314 *
0315 *****
0316 *
0317 *
0318 *
0319 *****
0320 *
0321 *
0322 *
0323 *****
0324 *
0325 *
0326 *
0327 *****
0328 *
0329 *
0330 *
0331 *****
0332 *
0333 *
0334 *
0335 *****
0336 *
0337 *
0338 *
0339 *****
0340 *
0341 *
0342 *
0343 *****
0344 *
0345 *
0346 *
0347 *****
0348 *
0349 *
0350 *
0351 *****
0352 *
0353 *
0354 *
0355 *****
0356 *
0357 *
0358 *
0359 *****
0360 *
0361 *
0362 *
0363 *****
0364 *
0365 *
0366 *
0367 *****
0368 *
0369 *
0370 *
0371 *****
0372 *
0373 *
0374 *
0375 *****
0376 *
0377 *
0378 *
0379 *****
0380 *
0381 *
0382 *
0383 *****
0384 *
0385 *
0386 *
0387 *****
0388 *
0389 *
0390 *
0391 *****
0392 *
0393 *
0394 *
0395 *****
0396 *
0397 *
0398 *
0399 *****
0400 *
0401 *
0402 *
0403 *****
0404 *
0405 *
0406 *
0407 *****
0408 *
0409 *
0410 *
0411 *****
0412 *
0413 *
0414 *
0415 *****
0416 *
0417 *
0418 *
0419 *****
0420 *
0421 *
0422 *
0423 *****
0424 *
0425 *
0426 *
0427 *****
0428 *
0429 *
0430 *
0431 *****
0432 *
0433 *
0434 *
0435 *****
0436 *
0437 *
0438 *
0439 *****
0440 *
0441 *
0442 *
0443 *****
0444 *
0445 *
0446 *
0447 *****
0448 *
0449 *
0450 *
0451 *****
0452 *
0453 *
0454 *
0455 *****
0456 *
0457 *
0458 *
0459 *****
0460 *
0461 *
0462 *
0463 *****
0464 *
0465 *
0466 *
0467 *****
0468 *
0469 *
0470 *
0471 *****
0472 *
0473 *
0474 *
0475 *****
0476 *
0477 *
0478 *
0479 *****
0480 *
0481 *
0482 *
0483 *****
0484 *
0485 *
0486 *
0487 *****
0488 *
0489 *
0490 *
0491 *****
0492 *
0493 *
0494 *
0495 *****
0496 *
0497 *
0498 *
0499 *****
0500 *
0501 *
0502 *
0503 *****
0504 *
0505 *
0506 *
0507 *****
0508 *
0509 *
0510 *
0511 *****
0512 *
0513 *
0514 *
0515 *****
0516 *
0517 *
0518 *
0519 *****
0520 *
0521 *
0522 *
0523 *****
0524 *
0525 *
0526 *
0527 *****
0528 *
0529 *
0530 *
0531 *****
0532 *
0533 *
0534 *
0535 *****
0536 *
0537 *
0538 *
0539 *****
0540 *
0541 *
0542 *
0543 *****
0544 *
0545 *
0546 *
0547 *****
0548 *
0549 *
0550 *
0551 *****
0552 *
0553 *
0554 *
0555 *****
0556 *
0557 *
0558 *
0559 *****
0560 *
0561 *
0562 *
0563 *****
0564 *
0565 *
0566 *
0567 *****
0568 *
0569 *
0570 *
0571 *****
0572 *
0573 *
0574 *
0575 *****
0576 *
0577 *
0578 *
0579 *****
0580 *
0581 *
0582 *
0583 *****
0584 *
0585 *
0586 *
0587 *****
0588 *
0589 *
0590 *
0591 *****
0592 *
0593 *
0594 *
0595 *****
0596 *
0597 *
0598 *
0599 *****
0600 *
0601 *
0602 *
0603 *****
0604 *
0605 *
0606 *
0607 *****
0608 *
0609 *
0610 *
0611 *****
0612 *
0613 *
0614 *
0615 *****
0616 *
0617 *
0618 *
0619 *****
0620 *
0621 *
0622 *
0623 *****
0624 *
0625 *
0626 *
0627 *****
0628 *
0629 *
0630 *
0631 *****
0632 *
0633 *
0634 *
0635 *****
0636 *
0637 *
0638 *
0639 *****
0640 *
0641 *
0642 *
0643 *****
0644 *
0645 *
0646 *
0647 *****
0648 *
0649 *
0650 *
0651 *****
0652 *
0653 *
0654 *
0655 *****
0656 *
0657 *
0658 *
0659 *****
0660 *
0661 *
0662 *
0663 *****
0664 *
0665 *
0666 *
0667 *****
0668 *
0669 *
0670 *
0671 *****
0672 *
0673 *
0674 *
0675 *****
0676 *
0677 *
0678 *
0679 *****
0680 *
0681 *
0682 *
0683 *****
0684 *
0685 *
0686 *
0687 *****
0688 *
0689 *

```

```

0212 *****
0213 *
0214 * TEXT MODE *
0215 *
0216 *****
0217 010A 00F2' FTMODE DATA TMPBUF,STXT VECTOR INTO MID
0218 010E' TMODE EQU $
0219 010E D1E0 MOV B @YLOC,R7 GET Y POSITION
0220 0110 0096'
0220 0112 0987 SRL R7,B POSITION IT
0221 0114 39E0 MPY @D40,R7 R7=0,R8=40*YLOC
0222 0116 025A'
0222 0118 D1E0 MOV B @XLOC,R7 GET X POSITION
0223 011A 0090'
0223 011C 0987 SRL R7,B POSITION IT
0224 011E A207 A R7,R8 R8=LINEAR CURSOR ADDRESS
0225 0120 0288 CI R8,40*24 VALID ?
0226 0122 03C0
0226 0124 14F2 JHE FTMODE N, GO FORCE TEXT MODE
0227 *****
0228 *
0229 * REMOVE CURSOR *
0230 *
0231 *****
0232 0126 0228 REMCUR AI R8,NTBA->4000 ADD IN TABLE START & W. BIT
0233 0128 4400
0233 012A C020 MOV @CURFLG,R0 CURSOR ENABLED?
0234 012C 0084'
0234 012E 1604 JNE RVCA N, LEAVE SCREEN ALONE
0235 0130 0694 BL *R4 SEND ADDRESS TO VDP
0236 0132 D820 MOV B @CCSAVE,@VRAM WRITE BACK CHARACTER THAT
0237 0134 0000
0238 0136 F120
0237 * WAS UNDER THE CURSOR
0238 0138 0694 RVCA BL *R4 RESTORE VDP CURSOR ADDRESS
0239 *****
0240 *
0241 * CHECK FOR END OF MSG *
0242 *
0243 *****
0244 013A 8289 DONYET C R9,R10 REACHED THE END OF MSG ?
0245 013C 1B3B JH PUTCUR Y, EXIT
0246 013E D039 MOV B *R9+,R0 N, GET NEXT CHARACTER
0247 0140 1F05 TB F$SHOW SHOW ALL CONTROL CHARACTERS ?
0248 0142 1303 JEQ CSHOW Y, TREAT AS NORMAL
0249 0144 9800 CB R0,@B20 N, CONTROL ?
0250 0146 003E'
0250 0148 1A71 JL CNTRL Y, HANDLE IT.
0251 *
0252 014A D800 CSHOW MOV B R0,@VRAM N, WRITE CHARACTER
0253 014C F120
0253 014E 0588 INC R8 UPDATE CURSOR
0254 * CHECK FOR OFF BOTTOM OF SCREEN
0255 *
0256 0150 0288 OFFBOT CI R8,NTBA+BRHC->4000 OFF BOTTOM ?
0257 0152 47BF
0257 0154 12F2 JLE DONYET N, CONTINUE
0258 0156 1F06 TB F$ROLL Y, SCROLL DISABLED ?
0259 0158 1603 JNE DOROLL N, GO DO IT
0260 015A 0228 AI R8,-SCRSIZ Y, BACKUP R8

```

015C FC40
0261 015E 10EC

JMP RVCA

AND CONTINUE

```

0263      *
0264      *   SCROLL ROUTINE
0265      *
0266 0160 C1C8 DOROLL MOV R8,R7      SAVE CURSOR
0267 0162 0227      AI R7,-40      RESTORE CURSOR POSITION
      0164 FFD8
0268      *
0269 0166 0208      LI R8,NTBA+40    REF. START OF LINE 1
      0168 0428
0270 016A 0694      BL *R4          SEND IT TO VDP
0271 016C 04C1      CLR R1          GET START OF BUFFER
0272      *
0273 016E D860 RDSCR MOV @VRAM,@TMPBUF(R1) READ CHARACTER FROM VDP
      0170 F120
      0172 010A'
0274 0174 0581      INC R1          COUNT IT
0275 0176 0281      CI R1,920      DONE ?
      0178 0398
0276 017A 1AF9      JL RDSCR        N, CONTINUE READING SCREEN
0277      *
0278 017C 0208      LI R8,NTBA+>4000 SET FOR WRITE TO TOP OF SCREEN
      017E 4400
0279 0180 0694      BL *R4          SEND IT TO VDP
0280 0182 04C1      CLR R1          GET START OF BUFFER
0281      *
0282 0184 D821 WRSCR MOV @TMPBUF(R1),@VRAM WRITE CHARACTER TO VDP
      0186 0172'
      0188 F120
0283 018A 0581      INC R1          COUNT IT
0284 018C 0281      CI R1,920      DONE ?
      018E 0398
0285 0190 1AF9      JL WRSCR        N, LOOP
0286      *
0287 0192 0201      LI R1,40        NOW FILL BOTTOM LINE WITH ' '
      0194 0028
0288 0196 D820 WRSP  MOV @B20,@VRAM  WRITE SPACE
      0198 0146'
      019A F120
0289 019C 0601      DEC R1          COUNT IT
0290 019E 16FB      JNE WRSP        N, LOOP
0291      *
0292      *   NOW RESTORE THE CURSOR
0293      *
0294 01A0 C207      MOV R7,R8
0295 01A2 0694      BL *R4          GIVE IT TO VDP
0296 01A4 10CA      JMP DONYET      & CONTINUE OUTPUT
0297 01A6 0288 OFFTOP CI R8,NTBA+>4000 OFF TOP OF SCREEN
      01A8 4400
0298 01AA 14C7      JHE DONYET      N, CONTINUE
0299      *
0300 01AC 0228      AI R8,SCRSIZ    Y, ADJUST CURSOR
      01AE 03C0
0301 01B0 0694      BL *R4          SEND IT TO VDP
0302 01B2 10C3 FD$JMP JMP DONYET    & CONTINUE

```



```

0304      *
0305      *      PUT CURSOR AND EXIT ROUTINE
0306      *
0307 01B4 0228  PUTCUR AI   R8,->4000      PUT VDP IN READ MODE
      01B6 C000
0308 01B8 0694      BL   *R4      SEND TO VDP
0309 01BA 04C3      CLR   R3
0310 01BC D0E0      MOVB @VRAM,R3      GET CHARACTER UNDER CURSOR
      01BE F120
0311 01C0 0228      AI   R8,>4000      PUT BACK IN WRITE MODE
      01C2 4000
0312 01C4 0694      BL   *R4      SEND IT
0313 01C6 C060      MOV  @CURFLG,R1      CURSOR ENABLED
      01C8 012C
0314 01CA 1603      JNE  PUT#1      N, LEAVE SCREEN
0315 01CC D820      MOVB @B7F,@VRAM      WRITE CURSOR CHARACTER
      01CE 0000
      01D0 F120
0316      * NOW WORK OUT THE X,Y CURSOR
0317 01D2 0228  PUT#1 AI   R8,-(NTBA->4000) REMOVE TABLE BASE & WR. BIT
      01D4 BC00
0318 01D6 04C7      CLR   R7      READY FOR DIVIDE
0319 01D8 3DE0      DIV  @D40,R7      DO DIVIDE
      01DA 025A
0320      * R7 = Y , R8 = X
0321 01DC 02A1      STWP R1      GET WP
0322 01DE D821      MOVB @1+(2*R7)(R1),@YLOC      SET YLOC
      01E0 000F
      01E2 0110
0323 01E4 D821      MOVB @1+(2*R8)(R1),@XLOC      SET XLOC
      01E6 0011
      01E8 011A
0324 01EA D803      MOVB R3,@CCSAVE      SAVE CHARACTER
      01EC 0134
0325 01EE 0953      SRL  R3,5      R3= 8 * CHAR
0326 01F0 C203      MOV  R3,R8      SET FOR ADDRESS
0327      * READ THE PATTERN FOR SAVED CHARACTER
0328 01F2 0228  AI   R8,PGBA      R8 = ADDRESS OF CHAR PAT.
      01F4 0800
0329 01F6 0694      BL   *R4      SEND ADDRESS TO VDP
0330 01F8 0208  LI   R8,BITMAP      POINT TO 'BITMAP'
      01FA 0048
0331 01FC C048      MOV  R8,R1      SAVE IT
0332 01FE 0207  LI   R7,8      DO 8 BYTES
      0200 0008
0333 0202 C087      MOV  R7,R2      SAVE IT
0334 0204 DE20  RDPAT MOVB @VRAM,*R8+      READ BYTE FROM VDP
      0206 F120
0335 0208 0607      DEC  R7      COUNT IT
0336 020A 16FC      JNE  RDPAT      LOOP TILL DONE
0337      * WRITE INVERSE AS CHARACTER >7F BIT PATTERN
0338 020C 0208  LI   R8,PGBA->4000+(8*>7F) REF >7F PAT. ENTRY
      020E 4BFB
0339 0210 0694      BL   *R4      SEND IT TO VDP
0340      *
0341 0212 D031  WRPAT MOVB *R1+,R0      GET PATTERN
0342 0214 0540      INV  R0      INVERT IT
0343 0216 D800      MOVB R0,@VRAM      SEND IT TO VDP
      0218 F120
0344 021A 0602      DEC  R2      DONE ?

```

0345	021C	16FA		JNE	WRPAT	N, LOOP
0346			*			
0347	021E	0380	DEXIT	RTWP		Y, EXIT
0348			*			
0349			*		CURSOR ON/OFF CONTROL	
0350			*			
0351	0220	04E0	FS	CLR	@CURFLG	ENABLE CURSOR
	0222	01C8				
0352	0224	10C6		JMP	FD\$JMP	AND CONTINUE
0353			*			
0354	0226	0720	GS	SETD	@CURFLG	DISABLE CURSOR
	0228	0222				
0355	022A	10C3		JMP	FD\$JMP	AND CONTINUE

```

0357      *
0358      *      HANDLE TEXT CONTROL CHARACTERS
0359      *
0360 022C 06A0 CNTRL BL @JMPRO ; DO JUMP ON R0 (USES R1,R2)
      022E 00B0'
0361 0230 0D CJUMP BYTE HT-CJUMP/2,>09 HT - CURSOR RIGHT
0362 0232 11 CJUMP BYTE BS-CJUMP/2,>08 BS - CURSOR LEFT
0363 0234 14 CJUMP BYTE LF-CJUMP/2,>0A LF - CURSOR DOWN
0364 0236 17 CJUMP BYTE VT-CJUMP/2,>0B VT - CURSOR UP
0365 0238 1A CJUMP BYTE FF-CJUMP/2,>0C FF - CLEAR SCREEN & HOME
0366 023A 28 CJUMP BYTE CR-CJUMP/2,>0D CR - CURSOR BEGINING OF LINE
0367 023C FB CJUMP BYTE FS-CJUMP/2,>1C FS - CURSOR ON
0368 023E FB CJUMP BYTE GS-CJUMP/2,>1D GS - CURSOR OFF
0369 0240 24 CJUMP BYTE RS-CJUMP/2,>1E RS - CURSOR HOME
0370 0242 0000 DATA 0
0371 0244 0720 SETD @BELCNT ILLEGAL, SET BELL COUNTER
      0246 00C8'
0372 0248 10B4 FDONE JMP FD$JMP & TEST FOR COMPLETION
0373      *
0374      *      CURSOR RIGHT (HT)
0375      *
0376 024A 0588 HT INC R8 ADJUST CURSOR
0377 024C 0694 UDATED BL *R4 SEND IT TO VDP
0378 024E 0460 B @OFFBOT CHECK CURSOR ON SCREEN
      0250 0150'
0379      *
0380      *      CURSOR LEFT (BS)
0381      *
0382 0252 0608 BS DEC R8 BACKUP CURSOR
0383 0254 0694 UDATEU BL *R4 SEND IT TO VDP
0384 0256 10A7 JMP OFFTOP CHECK CURSOR ON SCREEN
0385      *
0386      *      CURSOR DOWN (LF)
0387      *
0388 0258 0228 LF AI R8,40 DOWN A LINE
      025A 0028
0389 025A' D40 EQU #-2
0390 025C 10F7 JMP UDATED UPDATE & CHECK
0391      *
0392      *      CURSOR UP (VT)
0393      *
0394 025E 0228 VT AI R8,-40 UP A LINE
      0260 FFD8
0395 0262 10F8 JMP UDATEU UPDATE & CHECK
0396      *
0397      *      CLEAR SCREEN (FF)
0398      *
0399 0264 0208 FF LI R8,NTBA+>4000 REF TOP OF SCREEN
      0266 4400
0400 0268 0694 BL *R4 SEND IT TO VDP
0401 026A 0201 LI R1,960 DO 960 CHARACTERS
      026C 03C0
0402 026E D820 CLRSCN MOVB @B20,@VRAM WRITE ' ' TO VDP
      0270 0198'
      0272 F120
0403 0274 0601 DEC R1 COUNT IT
0404 0276 16FB JNE CLRSCN LOOP TILL DONE
0405      * FALL THROUGH TO CURSOR HOME
0406      *
0407      *      CURSOR HOME (RS)

```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.MONITOR
OBJECT ACCESS NAME= ADHOC.OBJ.MONITOR
LISTING ACCESS NAME= ADHOC.LST.MONITOR
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

LINE	KEY	NAME
------	-----	------

0024	A	ADHOC.MACRO =>ADHOC.MACRO
------	---	------------------------------

```

0002          IDT  'MONITOR'
0003          *
0004          DEF  DEBUG$, MONTOP
0005          DEF  WHXETY, RHENTY, WHENTY
0006          DEF  ECHOEN, WENTRY, RENTRY
0007          DEF  MENTRY, XOPENT
0008          DEF  ERR3$M, D$LDWP, D$LDPC
0009          *
0010          REF  HALTO$, BEGN1, MC$LOD, MC$SAV
0011          REF  WORKS, ZZZZ$
0012          REF  INITFG, BPTOV, BPTSET, BPTADD
0013          REF  BPTDTA, ASMOPC, MREGS, MREG3
0014          REF  MREG13, PC
0015          REF  BOD, GETCR$, TYPO$
0016          REF  UNIT, MCRLF, TYPEN$, TYPC$
0017          REF  BRAM, BADTER, BADADR
0018          REF  BELL, SPACE5, SPACE2, BPMSG
0019          REF  LOGON, SPR, REGSTR, ASKBP, EXMSG, SPBP
0020          REF  SETVEC, DSRcnt, STPMSG, SYMBLC, MODE
0021          REF  PROMPT, WS, EQU$GN, GTLN, BLSTOR, ASMPTR, IOB
0022          REF  ESCFLG, F$WHO, PADIT, TYP$S, WPR1
0023          *
0024          COPY ADHOC.MACRO
A0001          *
A0002          *  DEFINE A MACRO TO DO THE 'READ Gs' ETC. INSTRUCTIONS
A0003          *  WHERE THE RESULTING OPCODE WILL BE OF THE FORM
A0004          *
A0005          *      0 0 0 0   1 1 1 X   X X X X   X X X X
A0006          *                      | N       | Ts |   Rs   |
A0007          *
A0008          *      WNBL      N=0
A0009          *      RHXW      N=1
A0010          *      WHXW      N=2
A0011          *      EKO       N=3
A0012          *      WRIT      N=4
A0013          *      READ      N=5
A0014          *      MSG       N=6
A0015          *      BKPT      N=7
A0016          *
A0017          *
A0018          *      NOTE : IF THE OPERAND IS INDEXED THEN THE CORRECT
A0019          *      CODE IS GENERATED BUT SDSMAC WILL ALSO ERROR
A0020          *      IT.
A0021          *
A0022          Z$$Z$$ $MACRO P1,0
A0023          *      DEFINE THE REQUIRED DATA PASSED THE BASE OPCODE
A0024          *      AND THE OPERAND.
A0025          $VAR T
A0026          $ASG P1 TO T
A0027          *      IS IT INDEXED
A0028          $IF P1.A&$PNDX
A0029          $ASG 0.V+>20 TO 0.V
A0030          $ASG :0.V: '+R':P1.V: ', ':P1.S: TO T.S
A0031          DATA :T.S:      INDEX
A0032          $EXIT
A0033          $ENDIF
A0034          *      NO , IS IT SYMBOLIC ?
A0035          $IF P1.A&$PSYM
A0036          $ASG 0.V+>20 TO 0.V
A0037          $ASG :0.V: ', ':P1.S: TO T.S

```

```

A0038          DATA : T. S:          SYMBOLIC
A0039          $EXIT
A0040          $ENDIF
A0041          *   NO , IS IT INDIRECT
A0042          $IF P1. A&$PIND
A0043          $ASG O. V+>10 TO O. V
A0044          $ASG : O. V: '+R': P1. V: TO T. S
A0045          DATA : T. S:          INDIRECT
A0046          $EXIT
A0047          $ENDIF
A0048          *   NO , IS IT INDIRECT AUTOINCREMENT
A0049          $IF P1. A&$PATO
A0050          $ASG O. V+>30 TO O. V
A0051          $ASG : O. V: '+R': P1. V: TO T. S
A0052          DATA : T. S:          AUTOINC
A0053          $EXIT
A0054          $ENDIF
A0055          *   NO IT MUST BE REGISTER THEN
A0056          $ASG : O. V: '+R': P1. V: TO T. S
A0057          DATA : T. S:          REGISTER
A0058          $END          END OF INTERNAL MACRO 'Z$$Z$$'
A0059          *
A0060          *   DEFINE UTILITY MACRO'S
A0061          *
A0062          WNLB          $MACRO P2
A0063          Z$$Z$$ : P2. S: , >0E00          WRITE HEX NIBBLE
A0064          $END
A0065          RHXW          $MACRO P2
A0066          Z$$Z$$ : P2. S: , >0E40          READ HEX WORD
A0067          $END
A0068          WHXW          $MACRO P2
A0069          Z$$Z$$ : P2. S: , >0E80          WRITE HEX WORD
A0070          $END
A0071          EKO          $MACRO P2
A0072          Z$$Z$$ : P2. S: , >0EC0          ECHO CHARACTER
A0073          $END
A0074          WRIT          $MACRO P2
A0075          Z$$Z$$ : P2. S: , >0F00          WRITE CHARACTER
A0076          $END
A0077          READ          $MACRO P2
A0078          Z$$Z$$ : P2. S: , >0F40          READ CHARACTER
A0079          $END
A0080          MSG          $MACRO P2
A0081          Z$$Z$$ : P2. S: , >0F80          OUT MESSAGE
A0082          $END
A0083          BKPT          $MACRO P2
A0084          Z$$Z$$ : P2. S: , >0FC0          BREAKPOINT
A0085          $END
A0086          *
A0087          *   END OF MACRO DEFINITIONS
A0088          *

```

```

0026 *****
0027 *
0028 * MENU OF CORTEX COMMANDS (ALPHABETICAL)
0029 *
0030 * ? WHERE AM I
0031 * A LINE BY LINE ASSEMBLER
0032 * PC ADDRESS
0033 * B BREAKPOINT SETTING
0034 * BREAKPOINT NUMBER (CR=DISPLAY ALL BK. PTS.)
0035 * '-' = CLEAR BREAKPOINT LIST
0036 * C CRU INSPECT CHANGE
0037 * BASE ADDRESS
0038 * NO. OF BITS
0039 * D TAG DUMP
0040 * START ADDRESS
0041 * STOP ADDRESS
0042 * ENTRY ADDRESS
0043 * PROMPT FOR IDT
0044 * E EXECUTE - MESSAGES FOLLOW -
0045 * PC
0046 * F FIND ('CR' =WORD, '-' =BYTE)
0047 * START ADDRESS
0048 * STOP ADDRESS
0049 * PATTERN
0050 * G GOTO BASIC (WARM START)
0051 * H I INITIALIZE MEMORY
0052 * J START ADDRESS
0053 * K STOP ADDRESS
0054 * VALUE
0055 * L LOADER LINK LOADER
0056 * PROMPT FOR IDT
0057 * M MEMORY INSPECT CHANGE
0058 * START ADDRESS (CR-MODIFY)
0059 * STOP ADDRESS
0060 * N NEGATIVE FIND (FIND ANYTHING BUT PATTERN)
0061 * O AS 'FIND'
0062 * P OUTPUT PORT TOGGLE
0063 * R INSPECT/CHANGE WP, PC, AND ST REGISTERS
0064 * (- GIVES PREVIOUS REGISTER)
0065 * S SINGLE/MULTIPLE STEPPING
0066 * VALUE LESS THAN >80=
0067 * NO. OF STEPS (NULL OR 0 =1 )
0068 * VALUE GREATER THAN >80=
0069 * SINGLE STEP UNTIL MEM ADDR REACHED
0070 * T TRACE-SINGLE STEP WITH PRINTOUT
0071 * NO. OF STEPS (NULL OR 0 =1 )
0072 * U UN-ASSEMBLER
0073 * V START ADDRESS (CR-SINGLE LINE)
0074 * STOP ADDRESS
0075 * W WORKSPACE REGISTER INSPECT/CHANGE
0076 * REGISTER NUMBER (CR-DUMP ALL REGS.)
0077 * X TRANSFER (XFER) MEMORY
0078 * Y START BYTE ADDRESS
0079 * Z STOP BYTE ADDRESS
0080 * DESTINATION START BYTE ADDR
0081 *
0082 *****
  
```

J MAPPER ON
 K MAPPER OF
 H EDITOR ASSEMBLER
 V VERIFY MEMORY

```
0084 *****
0085 * MODIFICATIONS DONE SINCE FIRST OPERATIONAL *
0086 * *
0087 * BREAKPOINT HANDLER CHANGED SO AS NOT TO DELETE B.P.'S *
0088 * WHEN THEY ARE 'HIT', ALSO CHANGED SO AS NOT TO CLEAR *
0089 * BREAKPOINTS ON SYSTEM RESET. *
0090 * OUTPUT ROUTINE: ESCAPE DETECT WRITTEN IN, SO AS TO GIVE *
0091 * RETURN TO COMMAND SCANNER WHEN ESCAPE IS PRESSED *
0092 * ASSEMBLER AND DISASSEMBLER RE-WRITTEN SO AS TO USE *
0093 * COMBINED TEXT/DATA TABLES. *
0094 * NEGATIVE FIND ROUTINE ADDED *
0095 * SINGLE STEP TO VALUE ADDED *
0096 * MID, AD, & UNDEFINED XOP HANDLERS ADDED. *
0097 * PRINT FORMAT IN 'R' & 'M' COMMANDS ALTERED SO AS TO USE *
0098 * 40 CHARACTER VDU'S *
0099 * BREAKPOINT HANDLING CHANGED SO AS TO REMOVE BPTS FROM *
0100 * USER RAM IN THE MONTOP. ALSO CHANGED SO THAT A SINGLE *
0101 * STEP IS EXECUTED BEFORE THE BPTS ARE SET. *
0102 * SINGLE STEP ROUTINES RE-WRITTEN SO AS TO USE LESS CODE *
0103 * USER VECTOR FOR BREAKPOINT ZERO INTRODUCED. *
0104 * PRINT FORMAT FOR 'F' & 'N' COMMANDS CHANGED *
0105 * SINGLE STEP/TRACE CHANGED SO AS NOT TO OUTPUT IN XOP'S *
0106 *****
```



```

0108 *
0109 0406 * MONITOR RE-ENTRY POINT
0110 0406 *
0111 0000 02E0 MONTOP LWPI MREGS >EF62
      0002 0000
0112 0002' LOADWP EQU $-2
0113 0002' D$LDWP EQU LOADWP
0114 0004 0720 SETO @F$WHO >EFD8
      0006 0000
0115 0008 06A0 BL @RMVBPT >0E12 REMOVE BREAKPOINTS IF NEC.
      000A 040C'
0116 000C' BPTSDK EQU $
0117 *
0118 * INITIALIZE LOAD ETC
0119 *
0120 000C 04C0 CLR R0
0121 000E 0201 LI R1, >FFFC LOAD POINTER
      0010 FFFC
0122 0012 CC60 MOV @LOADWP, *R1+ COPY WORKSPACE POINTER
      0014 0002' >0A08
0123 0018' D$LDPC EQU $+2
0124 0016 0202 LI R2, LOAD >0D10 LOAD 2ND POINTER
      0018 030A'
0125 001A CC42 MOV R2, *R1+ COPY LOAD ENTRY POINT
0126 001C 0A22 MSG @PROMPT >5614
0127 *
0128 * WAIT FOR A COMMAND ENTRY
0129 *
0130 0020 CMDIN READ R5
0131 0022 04C2 CLR R2 CLEAR KEY
0132 0024 04C3 CLR R3 CLEAR COUNT
0133 0026 0208 LI R8, 1 WORD BOUNDARY REG (LSB= 1)
      0028 0001
0134 002A 0209 LI R9, SPACE2 >55CC
      002C 0000
0135 002E 020A LI R10, MCRLF >556F
      0030 0000
0136 0032 06A0 BL @SRCH >0A96
      0034 0090'
  
```

0138		*		
0139		*	COMMAND SEARCH TABLE	
0140		*		
0141	0036	4D	TEXT 'M'	MEMORY INSPECT/CHANGE
0142	0037	03	BYTE 3	(START, STOP)
0143	0038	0194'	DATA M	
0144	003A	44	TEXT 'D'	DUMP MEMORY TO CASSETTE
0145	003B	07	BYTE 7	(START, STOP, ENTRY)
0146	003C	0000	DATA MC\$SAV	(NOW IN BASIC)
0147	003E	49	TEXT 'I'	INITIALIZE MEMORY COMMAND
0148	003F	07	BYTE 7	(START, STOP, VALUE)
0149	0040	0140'	DATA I	
0150	0042	41	TEXT 'A'	ZERO LABEL ASSEMBLER
0151	0043	01	BYTE 1	(START PC)
0152	0044	05B4'	DATA ZLABGN	
0153	0046	55	TEXT 'U'	UNASSEMBLER
0154	0047	03	BYTE 3	(START, STOP)
0155	0048	080A'	DATA U	
0156	004A	57	ADREVN TEXT 'W'	USER WORKSPACE INSPECT/CHANGE
0157	004B	01	BYTE 1	(REG NO.)
0158	004C	01EB'	DATA W	
0159	004E	45	TEXT 'E'	EXECUTE
0160	004F	00	BYTE 0	(PC ADDR-OPTIONAL)
0161	0050	034C'	DATA E	
0162	0052	42	TEXT 'B'	BREAKPOINT
0163	0053	01	BYTE 1	(BREAKPOINT NO.)
0164	0054	01DE'	BPCMD DATA B	
0165	0056	53	TEXT 'S'	EXECUTE SINGLE STEPS
0166	0057	03	BYTE 3	(VARIOUS)
0167	0058	02D2'	DATA S	
0168	005A	4C	TEXT 'L'	LOAD MEMORY FROM CASSETTE
0169	005B	00	BYTE 0	
0170	005C	0000	DATA MC\$LOD	(NOW IN BASIC)
0171	005E	43	TEXT 'C'	CRU INSPECT/CHANGE
0172	005F	03	BYTE 3	(CRU BASE, & OF BITS)
0173	0060	0250'	DATA C	
0174	0062	52	TEXT 'R'	WP, PC, ST INSPECT/CHANGE
0175	0063	00	BYTE 0	
0176	0064	0294'	DATA R	
0177	0066	46	TEXT 'F'	FIND BYTE/WORD
0178	0067	07	BYTE 7	(START, STOP, VALUE)
0179	0068	014E'	DATA F	
0180	006A	4E	TEXT 'N'	NEGATIVE FIND BYTE/WORD
0181	006B	07	BYTE 7	(START, STOP, VALUE)
0182	006C	018E'	DATA N	
0183	006E	54	TEXT 'T'	TRACE STEP AND PRINT WP, PC, ST
0184	006F	01	BYTE 1	(& OF STEPS)
0185	0070	033C'	DATA T	
0186	0072	58	TEXT 'X'	TRANSFER (XFER) MEMORY
0187	0073	07	BYTE 7	(START, STOP, DEST.)
0188	0074	011A'	DATA X	
0189	0076	50	TEXT 'P'	TOGGLE OUTPUT PORT
0190	0077	01	BYTE 1	(UNIT &)
0191	0078	0404'	DATA P	
0192	007A	47	TEXT 'G'	GOTO BASIC (WARM START)
0193	007B	00	BYTE 0	
0194	007C	0112'	DATA G	
0195	007E	3F	TEXT '?'	WHERE AM I?
0196	007F	00	BYTE 0	
0197	0080	03E2'	DATA QUERY	

0198 0082 00	BYTE 0,0,0,0	SPARE ENTRY ³ 4A 6000
0199 0086 00	BYTE 0,0,0,0	SPARE ENTRY ² 4B 6006
0200 008A 0000	DATA 0	END OF TABLE
0202	*	
0203	* COMMAND SEARCH ROUTINE	
0204	*	
0205 008C 022B	SRCHLP AI R11,3	UPDATE POINTER
008E 0003		
0206 0090 C1DB	SRCH MOV *R11,R7	SEARCH FAIL?
0207 0092 132C	JEQ ERR4 >0AF2	YES, ERROR
0208 0094 917B	CB *R11+,R5	DOES INPUT MATCH A TABLE ENTRY
0209 0096 16FA	JNE SRCHLP >0A92	NO, TRY NEXT TABLE ENTRY
0210 0098	WRIT R5	
0211 009A	WRIT *R9	
0212 009C D1BB	MOVB *R11+,R6	NUMBER OF HEX INPUTS TO ICOUNT
0213	*	
0214	* ICOUNT SPECIFIES NUMBER OF HEX INPUT FIELDS	
0215	*	
0216 009E 0986	SRL R6,8	ALIGN ICOUNT
0217 00A0 02A7	STWP R7	POINT TO R0
0218 00A2 0916	INLOOP SRL R6,1	DONE?
0219 00A4 1707	JNC CEXIT 0ABA	YES, TO COMMAND PROCESSOR
0220 00A6	RHXW R4	ACCEPT HEX ENTRY
0221 00AB 00C4	DATA NULL,ERR2 0AC4	0AE2
0222 00B2 CDC4	MOV R4,*R7+	SAVE HEX INPUT VIA POINTER
0223 00AE 0583	CNT INC R3	COUNT THE NUMBER OF ENTRIES
0224 00B0 9685	CB R5,*R10	END OF INPUT?
0225 00B2 16F7	JNE INLOOP >0A48	YES, TO COMMAND PROCESSOR
0226 00B4 028B	CEXIT CI R11,ADREVN >0A50	EVEN BOUNDARY REQUIRED
00B6 004A		
0227 00BB 1B02	JH BRCMD 0AC4	NO - JUMP
0228 00BA 4008	SZC R8,R0	
0229 00BC 4048	SZC R8,R1	
0230 00BE C2DB	BRCMD MOV *R11,R11	GET ENTRY ADDRESS
0231 00C0 069B	BL *R11	YES, TO COMMAND PROCESSOR
0232 00C2 109E	JMP MONTOP >0A06	RETURN FROM COMMANDS
0233 00C4 05C7	NULL INCT R7	UPDATE POINTER
0234 00C6 9685	CB R5,*R10	NO INPUT?
0235 00C8 13F5	JEQ CEXIT >0ABA	Y, GOT COMMAND
0236 00CA 028B	CI R11,BPCMD >0ASA	TEST FOR BREAKPOINT
00CC 0054		
0237 00CE 16EF	JNE CNT >0AB4	NO - DEFAULT PARAMETER
0238 00D0 0285	CI R5,'-'*256 "2000"	TEST FOR '-'
00D2 2D00		
0239 00D4 16EC	JNE CNT 0AB4	NO - DEFAULT
0240 00D6 0208	LI R8,BPTADD EFIE	
00DB 0000		
0241 00DA 1016	JMP CLRBPT 0B0E	YES, CLEAR BREAKPOINTS
0242	*	
0243	* ERROR HANDLER	
0244	*	
0245 00DC 0201	ERR2 LI R1,BADTER >562A	TERM. CHARACTER ERROR
00DE 0000		
0246 00E0 1002	JMP OUTERR 0AEC	
0247 00E2 0201	ERR3#M EQU #	
0248 00E2 0201	ERR3 LI R1,BADADR >563C	DUMP ADDRESS ERROR
00E4 0000		
0249	*	
0250 00E6 0000	OUTERR DATA TYPC# 0002	OUT 'CRLF'
0251 00E8 0000	DATA TYPEN# 0005	OUT ERROR TEXT

```

0252 00EA 108A JMMONT JMP MONTOP >0A06 RETURN TO MONITOR
0253 00EC ERR4 MSG @BELL @>55C2
0254 00FO 1097 JMP CMDIN 0A26
0255 *
0256 * INITIAL ENTRY POINT
0257 *
0258 00F2 02E0 DEBUG$ LWPI MREGS >EF62
      00F4 0002'
0259 00F6 BANNER MSG @LOGON 55D5 OUTPUT BANNER
0260 00FA 0208 LI RB, INITFG EF8 POINT TO INIT FLAG
      00FC 0000
0261 00FE 8818 C *RB, @INITIZ 0D2C TEST INIT. FLAG
      0100 0326'
0262 0102 13F3 JEQ JMMONT 0AF0 IF SET, TO MONTOP
0263 0104 CE20 MOV @INITIZ, *RB+ SET FLAG
      0106 0326'
0264 0108 04F8 CLRBPT CLR *RB+ CLEAR BOTH TABLES
0265 010A 0288 CI RB, BPTADD+64 EF5E
      010C 0040
0266 010E 1AFC JL CLRBPT 0B0E
0267 0110 10EC JMP JMMONT 0AF0 TO TOP OF MONITOR
0268 *****
0269 *
0270 * EXIT TO BASIC
0271 *
0272 *****
0273 0112 0420 G BLWP @BVEC1 0B1C WARMSTART VECTORS
      0114 0116'
0274 0116 0000 BVEC1 DATA WPR1, BEGN1 F0DC 0252

```

```

0276 *****
0277 *
0278 * TRANSFER (XFER) COMMAND -- 'X'
0279 *
0280 * CALLING SEQUENCE:          BL    X
0281 *                          RO=START BYTE ADDRESS
0282 *                          R1=STOP BYTE ADDRESS
0283 *                          R2=DESTINATION START ADDR
0284 *
0285 * RETURN                      RT
0286 *
0287 * IF THE DESTINATION ADDRESS IS BETWEEN THE START AND STOP
0288 * ADDRESS, THE TRANSFER IS BOTTOM UP TO AVOID COPYING
0289 * OVER THE ORIGINAL BEFORE IT IS TRANSFERED.
0290 *
0291 *****
0292 *
0293 0820 X      C      R0,R1      START < STOP
0294 011A 8040 XERR JH      ERR3 0AE8      N, EXIT
0295 011C 1BE2      C      R2,R0      DEST. LESS THAN START?
0296 011E 8002      JLE     TOPDN 0B3E      Y, TOP DOWN
0297 0120 120B      C      R2,R1      DEST. GREATER THEN STOP?
0298 0122 8042      JH      TOPDN 0B3E      Y, TOP DOWN
0299 0124 1B09      MOV     R1,R3      N, BOTTOM UP
0300 0126 C0C1      S      R0,R3
0301 0128 60C0      A      R3,R2      DEST BOT= DEST+(START-STOP)
0302 012A A083      BOTUP  MOVB *R1,*R2
0303 012C D491      DEC     R1      BUMP POINTERS
0304 012E 0601      DEC     R2
0305 0130 0602      C      R1,R0      PAST START?
0306 0132 8001      JHE     BOTUP 0B32      N, CONTINUE
0307 0134 14FB      RT              Y, EXIT TO MONITOR
0308 0136 045B      TOPDN  MOVB *R0+,*R2+
0309 0138 DCB0      C      R0,R1      PAST STOP?
0310 013A 8040      JLE     TOPDN 0B3E      N, CONTINUE
0311 013C 12FD      RT              Y, EXIT TO MONITOR
0312 013E 045B      *

```

```

0315 *****
0316 * INITILIZE MEMORY COMMAND -- 'I' *
0317 * * *
0318 * FILL MEMORY WITH THE KEY FROM 'START' TO 'STOP' BY WORDS *
0319 * * *
0320 * CALLING SEQUENCE:          BL      I *
0321 *          RO=START          *
0322 *          R1=STOP          *
0323 *          R2=KEY           *
0324 * * *
0325 * RETURN                      B      *R11 *
0326 * * *
0327 *****
0328 *
0329 0140 8040 I      C      RO,R1          TEST FOR VALID START/STOP
0330 0142 1BEC      JH      XERR
0331 0144 CC02 I11    MOV    R2,*RO+        DATA TO MEMORY
0332 0146 8040      C      RO,R1          DONE?
0333 0148 16FD      JNE    I11            NO, LOOP
0334 014A CC02      MOV    R2,*RO+
0335 014C 045B      RT              EXIT
0336 *

```

```

0339 *****
0340 *
0341 * FIND COMMAND -- 'F'
0342 *
0343 * LOOK FROM START ADDRESS TO STOP ADDRESS FOR
0344 * THE SPECIFIED DATA PATTERN.
0345 * RO-START ADDRESS
0346 * R1-STOP ADDRESS
0347 * R2-PATTERN
0348 * THE TERMINATION CHARACTER DETERMINES THE SEARCH
0349 * INCREMENT.
0350 * CARRAIGE RETURN - WORD
0351 * MINUS SIGN - BYTE
0352 *
0353 *****
0354 *
0355 014E 0206 F LI R6,SKIP2-SKIP11/2-1+>1600 JUMP NOT EQUAL
0356 0150 1603
0356 0152 0203 FIND LI R3,>8402 LOAD COMPARE- C R2,*R0
0356 0154 8402
0357 0156 0204 LI R4,>5C0 LOAD - INCT RO
0357 0158 05C0
0358 015A 9685 CB R5,*R10 TCHAR='CR'?
0359 015C 1306 JEQ SKIP0 YES, LEAVE AS C, INCT
0360 015E 0223 AI R3,>1000 CHANGE TO- CB
0360 0160 1000
0361 0162 0224 AI R4,->40 CHANGE TO- INC
0361 0164 FFC0
0362 0166 0A82 SLA R2,8 ALIGN PATTERN
0363 0168 1005 JMP SKIP1
0364 016A 4008 SKIP0 SZC R8,R0
0365 016C 4048 SZC R8,R1
0366 016E 1002 JMP SKIP1
0367 0170 C1C7 NXTBLK MOV R7,R7 DONE 4 ADDRESSES?
0368 0172 1603 JNE NOCRTN NO - JUMP
0369 0174 0207 SKIP1 LI R7,4
0369 0176 0004
0370 0178 MSG *R10
0371 017A 0483 NOCRTN X R3 EXECUTE THE COMPARE
0372 017C 0486 SKIP11 X R6 JNE (FIND) JEQ (-VE FIND)
0373 017E WHXW R0 OUTPUT ADDRESS
0374 0180 MSG *R9 TWO SPACES
0375 0182 0607 DEC R7
0376 0184 0484 SKIP2 X R4 EXECUTE THE INCREMENT
0377 0186 1802 JOC FEXIT AVOID >FFFF TRAP!
0378 0188 8040 C R0,R1 DONE?
0379 018A 12F2 JLE NXTBLK NO, LOOP
0380 018C 045B FEXIT RT
0381 *****
0382 *
0383 * NEGATIVE FIND COMMAND.
0384 * THE OPERATION OF THIS COMMAND IS THE SAME AS THE 'FIND'
0385 * COMMAND, BUT WILL OUTPUT ADDRESS OF DATA WHICH IS NOT
0386 * THE SAME AS THE INPUT PATTERN
0387 *
0388 *****
0389 018E 0206 N LI R6,SKIP2-SKIP11/2-1+>1300 =JUMP EQUAL
0389 0190 1303
0390 0192 10DF JMP FIND

```

```

0393 *****
0394 *
0395 * INSPECT/CHANGE MEMORY - 'M' COMMAND
0396 *
0397 * OPTIONS:
0398 *      1) START ADDRESS, CARRIAGE RETURN --
0399 *      DISPLAY ADDRESS, CONTENTS, AND
0400 *      OPEN THE MEMORY LOCATION FOR A CHANGE.
0401 *
0402 *      2) CARRIAGE RETURN -- SAME AS 1) BUT THE
0403 *      DEFAULT START ADDRESS IS 0000.
0404 *
0405 *      3) START ADDRESS, BLANK (OR COMMA), STOP
0406 *      STOP ADDRESS, CARRIAGE RETURN -- OUTPUT
0407 *      MEMORY CONTENTS FROM START ADDRESS TO
0408 *      STOP ADDRESS. DEFAULT VALUES FOR BOTH
0409 *      ADDRESSES ARE 0000.
0410 *
0411 *****
0412 *
0413 0194 0603 M      DEC R3      1 INPUT?
0414 0196 1315      JEG MIC      YES, TO MEMORY INSPECT/CHANGE
0415 *
0416 * MEMORY DUMP ROUTINE
0417 *
0418 0198 0203 MLOOP1 LI R3,4
      019A 0004
0419 019C      MSG *R10      NEXT LINE
0420 019E      WHXW R0      PRINT FIRST ADDRESS
0421 01A0      MSG @EQUUSGN  DELIMITER
0422 01A4 MLOOP2 WHXW *R0+  PRINT MEMORY CONTENTS
0423 01A6 C000      MOV R0,R0  AVOID THE >FFFF TRAP!
0424 01AB 1319      JEG MEXIT
0425 01AA 8040      C R0,R1      DONE?
0426 01AC 1B17      JH MEXIT     YES, BACK TO THE MONITOR
0427 01AE 0603      DEC R3      DONE WITH LINE?
0428 01B0 13F3      JEG MLOOP1  YES, NEW LINE
0429 01B2      WRIT *R9      NO, OUTPUT SPACE
0430 01B4 10F7      JMP MLOOP2
  
```



```
0432      *
0433      * MEMORY INSPECT/CHANGE ROUTINE
0434      *
0435 01B6 0640 MIC1 DECT R0          LAST ADDRESS?
0436 01B8 0285      CI  R5, '--'*256
      01BA 2D00
0437 01BC 1302      JEQ  MIC          YES
0438 01BE 0220      AI  R0,4          POINT TO NEXT ADDRESS
      01C0 0004
0439 01C2 00E6' MIC  DATA TYPC$      NEXT LINE
0440 01C4          WHXW R0          PRINT MEMORY ADDRESS
0441 01C6          MSG  @EQUSGN      PRINT '='
0442 01CA C110      MOV  *R0,R4      GET DATA
0443 01CC          WHXW R4          PRINT MEMORY CONTENTS
0444 01CE          MSG  *R9          DELIMITER
0445 01D0          RHXW R4          ACCEPT NEW INPUT
0446 01D2 01DB'      DATA MNULL,ERR2
0447 01D6 C404      MOV  R4,*R0      UPDATE CONTENTS
0448 01DB 9685 MNULL CB  R5,*R10     RETURN TO COMMAND SCANNER?
0449 01DA 16ED      JNE  MIC1
0450 01DC 045B MEXIT RT              EXIT TO MONITOR
0451      *
```

```

0454 *****
0455 *
0456 * INSPECT/CHANGE USER WORKSPACE REGISTER -- 'W' COMMAND *
0457 *
0458 * OPTIONS: 1) 'W' FOLLOWED BY CARRIAGE RETURN -- *
0459 * DISPLAY THE CONTENTS OF ALL CURRENT USER *
0460 * WORKSPACE REGISTERS AND RETURN TO THE *
0461 * COMMAND SCANNER. *
0462 *
0463 * 2) 'W', REGISTER NUMBER IN HEX, CARRIAGE *
0464 * RETURN -- DISPLAY THE CONTENTS OF THE *
0465 * DESIGNATED REGISTER. USER MAY ALTER *
0466 * THE CONTENTS FOLLOWED BY A TERMINATION *
0467 * CHARACTER OR MERELY ENTER A TERMINATION *
0468 * CHARACTER. THE TERMINATION CHARACTER *
0469 * SIGNIFIES WHAT IS TO BE DONE NEXT: *
0470 *
0471 * SPACE -- DISPLAY THE CONTENTS OF THE NEXT REG. *
0472 * MINUS -- DISPLAY THE CONTENTS OF THE PREVIOUS REG. *
0473 * CARRIAGE RETURN -- TO THE COMMAND SCANNER. *
0474 *
0475 * NOTE THAT THIS ROUTINE IS ALSO USED BY THE INSPECT/CHANGE *
0476 * BREAKPOINTS. *
0477 *****
0478 01DE 0207 B LI R7,BPTADD SET UP POINTER
0479 01E0 00D8'
0479 01E2 0208 LI R8,SPBP SET UP BREAKPOINT INITIALS
0480 01E4 0000
0480 01E6 1003 JMP BPOINT USE THE INS/CH WS REGR ROUTINE
0481 *
0482 01E8 C1CD W MOV R13,R7 GET WORKSPACE POINTER
0483 01EA 0208 LI R8,SPR SET TO PRINT ' R'
0484 01EC 0000
0484 01EE C0C3 BPOINT MOV R3,R3 NULL INPUT?
0485 01F0 1321 JEQ WNULL1 YES, TO FORMATTED DUMP
0486 *
0487 * INSPECT/CHANGE A WORKSPACE REGISTER OR BREAKPOINT *
0488 *
0489 01F2 0240 ANDI R0,>F ISOLATE NUMBER
0490 01F4 000F
0490 01F6 C180 MOV R0,R6 SAVE R/BPT NUMBER
0491 01F8 0A10 SLA R0,1
0492 01FA A1C0 A R0,R7 FORM R/BPT ADDRESS
0493 01FC ICLOOP MSG *R10 NEXT LINE
0494 01FE MSG *R8
0495 0200 WLBL R6 OUTPUT R/BPT NUMBER
0496 0202 MSG @EQUUSGN PRINT '='
0497 0206 C117 MOV *R7,R4
0498 0208 WHXW R4 PRINT RR/BPT CONTENTS
0499 020A WRIT *R9 DELIMITER
0500 020C RHXW R4 NEW CONTENTS?
0501 020E 0214' DATA WNULL2,ERR2
0502 0212 C5C4 MOV R4,*R7 UPDATE R/BPT
0503 0214 9685 WNULL2 CB R5,*R10 RETURN TO COMMAND SCANNER?
0504 0216 1601 JNE SKIP NO, CHECK FOR ' '
0505 0218 045B WEXIT RT TO SCANNER
0506 021A 9645 SKIP CB R5,*R9 NEXT R/BPT?
0507 021C 1305 JEQ NREG YES
0508 *
0509 * CHECK FOR REGISTER OR BREAKPOINT ZERO

```

```

0510      *
0511 021E C186      MOV R6,R6      AT REGISTER 0?
0512 0220 13FB      JEQ WEXIT      YES, TO SCANNER
0513 0222 0606      DEC R6          UPDATE R/BPT NUMBER
0514 0224 0647      DECT R7         UPDATE ADDRESS
0515 0226 10EA      JMP ICLOOP
0516      *
0517      * CHECK FOR REGISTER OR BREAKPOINT >F
0518      *
0519 0228 0286      NREG CI R6,>F
      022A 000F
0520 022C 13F5      JEQ WEXIT      R/BPT >F, TO SCANNER
0521 022E 0586      INC R6          UPDATE R/BPT NUMBER
0522 0230 05C7      INCT R7         UPDATE ADDRESS
0523 0232 10E4      JMP ICLOOP
0524      *
0525      * FORMATTED REGISTER OR BREAKPOINT DISPLAY
0526      *
0527 0234 0206      WNULL1 LI R6,->10
      0236 FFF0
0528 0238      NLINE MSG *R10      NEXT LINE
0529 023A      WLOOP MSG *R8
0530 023C      WNBL R6             R/BPT NUMBER
0531 023E      MSG @EGUSGN        PRINT '='
0532 0242      WHXW *R7+          PRINT CONTENTS
0533 0244 0586      INC R6         TO NEXT R/BPT
0534 0246 13E8      JEQ WEXIT      DONE?
0535 0248 25A0      CZC @C3,R6    4 ENTRIES ON THE LINE?
      024A 0BEA'
0536 024C 13F5      JEQ NLINE      YES, NEXT LINE
0537 024E 10F5      JMP WLOOP      NO, CONTINUE ON THIS LINE
0538      *

```

```

0541 *****
0542 *
0543 * CRU INSPECT/CHANGE -- 'C' COMMAND
0544 *
0545 * INPUT THE CRU BASE ADDRESS FOLLOWED BY THE BIT COUNT.
0546 * ALL INPUT AND OUTPUT TO THE CRU IS RIGHT JUSTIFIED IN
0547 * THE 16 BIT INPUT/OUTPUT DATA FIELDS.
0548 *
0549 * INPUT OF A CARR. RET. AS A TERMINATION CHARACTER RETURNS
0550 * CONTROL TO THE COMMAND SCANNER. A ' ' AS TERMINATION
0551 * CHARACTER CAUSES THE CRU INPUT BITS TO BE OUTPUT AGAIN
0552 * WELL AS THE CRU OUTPUT BITS TO BE CHANGED.
0553 *
0554 *****
0555 *
0556 0250 C300 C      MOV  R0,R12      UPDATE CRU BASE REGISTER
0557 0252 04C7      CLR  R7        RESET WORD FLAG
0558 0254 0AC1      SLA  R1,12     ISOLATE BIT COUNT
0559 0256 1303      JEQ  SETFG     SET FLAG FOR 16-BIT TRANS.?
0560 0258 0281      CI   R1,>9000  BYTE JUSTIFIED I/O?
0561      025A 9000
0562 025C 1A01      JL   CSKIP1    YES, SKIP
0563 025E 0587      SETFG INC  R7    WORD JUSTIFIED I/O FLAG
0564 *
0565 * FORM 'STCR' COMMAND AND READ CRU
0566 *
0566 0260 0961      CSKIP1 SRL  R1,6      POSITION BIT COUNT
0567 0262 0208      CLOOP LI   R8,>3404  'STCR R4' OP CODE
0568      0264 3404
0568 0266 E201      SOC  R1,R8      COMBINE WITH BIT COUNT
0569 0268 0488      X    R8        EXECUTE 'STCR'
0570 *
0571 * OUTPUT STATE OF CRU
0572 *
0573 026A          MSG  *R10        NEXT LINE
0574 026C          WHXW R12        PRINT BASE ADDRESS
0575 026E          MSG  @EQU$GN    PRINT '='
0576 0272 C1C7      MOV  R7,R7     WORD I/O?
0577 0274 1601      JNE  CSKIP2    YES, SKIP ALIGN
0578 0276 0984      SRL  R4,8      ALIGN BYTE INPUT TO WORD
0579 0278          CSKIP2 WHXW R4   PRINT CRU DATA
0580 027A          MSG  *R9        DOUBLE SPACE
0581 *
0582 * ACCEPT INPUT FOR ALTERATION OF CRU
0583 *
0584 027C          RHXW R4          CRU OUTPUT?
0585 027E 028E'      DATA CNULL,ERR2
0586 0282 C1C7      MOV  R7,R7     WORD I/O?
0587 0284 1601      JNE  CSKIP3    YES, SKIP ALIGN
0588 0286 0A84      SLA  R4,8      ALIGN BYTE FOR OUTPUT
0589 0288 0248      CSKIP3 ANDI R8,>F3FF CHANGE 'STCR' TO 'LDCR'
0590      028A F3FF
0590 028C 0488      X    R8        EXECUTE 'LDCR'
0591 028E 9685      CNULL CB  R5,*R10 EXIT?
0592 0290 16E8      JNE  CLOOP    NO LOOP
0593 0292 045B      RT           EXIT TO MONITOR

```

```

0596 *****
0597 *
0598 * DISPLAY WP, PC, ST REGISTERS
0599 *
0600 * TERMINATION CHARACTERS:
0601 * SPACE -- TO NEXT REGISTER
0602 * CARRIAGE RETURN -- TO MONITOR SCANNER
0603 * MINUS -- INSPECT PREVIOUS REGISTER
0604 *
0605 * ORDER OF DISPLAY: WP, PC, ST.
0606 *
0607 *****
0608 *
0609 0294 0206 R LI R6, WS+4 INIT. MESSAGE POINTER
      0296 0004
0610 0298 0588 INC R8 SET LOOP COUNT
0611 029A 0207 LI R7, MREG13+2 POINT TO WP+2
      029C 0002
0612 029E 0226 BACKWP AI R6, -4
      02A0 FFFC
0613 02A2 0647 DECT R7
0614 02A4 0588 INC R8
0615 02A6 0288 CI R8, 4
      02A8 0004
0616 02AA 1412 JHE REXIT
0617 02AC 01C2' RLOOP DATA TYPC$
0618 02AE MSG *R6 OUTPUT REGISTER SLOGAN
0619 02B0 C117 MOV *R7, R4
0620 02B2 WHXW R4 PRINT CONTENTS
0621 02B4 MSG *R9 DELIMITER
0622 02B6 RHXW R4 NEW DATA?
0623 02B8 02BE' DATA RNULL, ERR2
0624 02BC C5C4 MOV R4, *R7
0625 02BE 0285 RNULL CI R5, '-'*256 PREVIOUS REGISTER?
      02C0 2D00
0626 02C2 13ED JEQ BACKWP YES, THEN GO BACK
0627 02C4 9685 CB R5, *R10 TO SCANNER?
0628 02C6 1304 JEQ REXIT YES, EXIT
0629 02C8 8DB6 C *R6+, *R6+ UPDATE MESSAGE POINTER BY 4
0630 02CA 05C7 INCT R7 POINT TO NEXT OUTPUT
0631 02CC 0608 DEC R8 DONE?
0632 02CE 16EE JNE RLOOP NO
0633 02D0 045B REXIT RT EXIT TO MONITOR
0634 *
  
```

```

0637 *****
0638 *
0639 * SINGLE STEP COMMAND -- 'S'
0640 *
0641 * THE SINGLE STEP ENTRY HAS BEEN MODIFIED TO ALLOW FOR
0642 * MULTIPLE STEPPING. IT REQUIRES AN ENTRY WHILE IN THE
0643 * SCANNER.
0644 *
0645 *****
0646 *
0647 02D2 0207 S LI R7,>9999 SET CODE
      02D4 9999
0648 02D6 0603 DEC R3 STEP TO VALUE?
0649 02D8 1504 JGT VALADR YES, JUMP
0650 *
0651 02DA 0647 DECT R7 NEXT CODE (>9997)
0652 02DC 0280 CI R0,>80 STEPS OR MEM ADR?
      02DE 0080
0653 02E0 1A02 JL SINGLE STEPS - JUMP
0654 *
0655 * STEP UNTIL MEMORY ADDRESS CONTAINS VALUE
0656 *
0657 02E2 4008 VALADR SZC R8,R0 WORD ALIGN ADDRESS
0658 02E4 1003 JMP STEP GO TO STEPPER
0659 *
0660 * SINGLE/MULTIPLE STEPS
0661 *
0662 02E6 0647 SINGLE DECT R7 NEXT CODE (>9995)
0663 02E8 LDCNT MSG *R10 GO TO NEW LINE
0664 02EA 0600 DECSTP DEC R0 COUNTING REGISTER
0665 02EC 03E0 STEP LREX INITIATE THE LOAD VECTOR
0666 02EE 0380 GO RTWP GOTO USER PROGRAM
0667 *
0668 02F0 C000 STPCHK MOV R0,R0 DONE STEPPING?
0669 02F2 15FB JGT DECSTP NO, DECREMENT AND LREX AGAIN
0670 02F4 1073 JMP WPS10 YES, OUTPUT ONCE
0671 *****
0672 * USER SUBROUTINE ON BREAKPOINT ZERO *
0673 *****
0674 02F6 0690 CONTIN BL *R0 EXECUTE USER ROUTINE
0675 02FB C7A0 MOV @BPTDTA,*R14 INSERT USER INSTRUCTION
      02FA 0000
0676 02FC C00E MOV R14,R0 SAVE LOCATION
0677 02FE 0202 LI R2,BPTLOD GET LOAD ADDRESS
      0300 0304
0678 0302 1045 JMP SETLOD DO A SINGLE STEP
0679 0304 C420 BPTLOD MOV @COD15,*R0 BREAKPOINT TO RAM
      0306 03AA
0680 0308 0380 RTWP

```

```

0683      *
0684      * LOAD INTERRUPT ENTRY POINT
0685      *
0686      *      POWER UP:      R7=RANDOM VALUE
0687      *      SINGLE STEP:   R7=>9995
0688      *      SS TO ADR:     R7=>9997
0689      *      SS TO VALUE:   R7=>9999
0690      *      TRACE:        R7=>9900
0691      *
0692 030A 0287 LOAD CI R7,>9997      MEM ADDRESS STEP?
030C 9997
0693 030E 1605      JNE CHKVAL      NO, JUMP
0694 0310 8380      C R0,R14        REACHED REQUIRED ADDRESS?
0695 0312 16EC      CHKDUN JNE STEP  DONE?
0696 0314      PRTSTP MSG @STPMSG   GO PRINT DATA
0697 0318 1061      JMP WPS10
0698 031A 0287      CHKVAL CI R7,>9999 TEST FOR VALUE STEP
031C 9999
0699 031E 1602      JNE CHKSGL      IF NOT, JUMP
0700 0320 8050      C *R0,R1        CHECK IF ADR=VAL
0701 0322 10F7      JMP CHKDUN      GO FIX IT
0702 0324 0287      CHKSGL CI R7,>9995 SINGLE STEP?
0326 9995
0703      0326' INITIZ EGU $-2
0704 0328 13E3      JEQ STPCHK
0705 032A 0287      CI R7,>9900      TRACE STEP?
032C 9900
0706 032E 1356      JEQ WPS10        YES, OUTPUT WP, PC, ST, INSTR
0707      *
0708      * IF NOT SINGLE STEP OR TRACE, IT IS A SYSTEM INITIALIZATION
0709      *
0710 0330 04E0      CLR @DSRCNT      CLEAR DSR COUNTER
0332 0000
0711 0334 06A0      BL @SETVEC       RESET ALL IMPORTANT THINGS
0336 0000
0712 0338 0460      B @DEBUG$       GOTO MONITOR
033A 00F2'
0713      *****
0714      *
0715      * TRACE COMMAND -- 'T'
0716      *
0717      * MULTIPLE STEPPING WITH WP, PC, ST, AND INSTRUCTION
0718      * PRINTED AFTER EACH STEP
0719      *
0720      *****
0721      *
0722 033C 0207 T LI R7,>9900      SET TRACE FLAG
033E 9900
0723 0340 C14B      MOV R11,R5      SAVE RETURN ADDR
0724 0342 06A0      BL @LDCNT      GO TO THE STEPPER
0344 02EB'
0725      *
0726      * TRACE COMES HERE AFTER PRINTING WP, PC, ST, AND INSTR
0727      *
0728      * CHECK TRACE COUNT
0729      *
0730 0346 C000      MOV R0,R0      IF COUNT=0 DO ONLY ONE
0731 0348 15CF      JGT LDCNT      IF >0 DECREMENT AND STEP
0732 034A 0455      B *R5          RETURN TO MONITOR

```

```

0735 *****
0736 *
0737 * EXECUTE COMMAND -- 'E'
0738 *
0739 * ASK USER IF BREAKPOINTS ARE TO BE SET. IF SO SET ALL
0740 * BREAKPOINTS WHICH DO NOT HAVE A ZERO ADDRESS.
0741 * OUTPUT THE WP,PC,ST DATA. WAIT FOR USER INPUT.
0742 * IF A PARAMETER IS ENTERED IT REPLACES THE PC BEFORE
0743 * THE RTWP. OTHERWISE ONLY THE RTWP IS EXECUTED.
0744 *
0745 *****
0746 *
0747 034C E MSG @ASKBP ASK IF BP'S TO BE SET
0748 0350 EKD R4 ANSWER IN R4
0749 0352 0284 CI R4,>1B00 ESCAPE?
0750 0354 1B00
0751 0354' ESC EQU $-2
0752 0356 1358 JEQ TOPRTN YES, EXIT
0753 0358 02AC' DATA TYPC$ NO, OUT CRLF
0754 035A MSG @EXMSG 'EXECUTE'
0755 035E 0207 LI R7,MREG13 GET WP POINTER
0756 0360 0000
0757 0362 0206 LI R6,WS POINT TO MESSAGE
0758 0364 0000
0759 0366 WSPLP MSG *R9 OUTPUT 2 SPACES
0760 0368 0287 CI R7,MREG13+6 DONE?
0761 036A 0006
0762 036C 1304 JEQ INPC YES, JUMP
0763 036E MSG *R6+ OUTPUT MESSAGE & INC POINTER
0764 0370 WHXW *R7+ OUTPUT DATA & INC. POINTER
0765 0372 05C6 INCT R6 POINT TO NEXT MESSAGE
0766 0374 10F8 JMP WSPLP LOOP BACK
0767 0376 INPC RHXW R1 GET NEW PC (IF ANY)
0768 0378 037E' DATA PCOK,ERR2
0769 037C C381 MOV R1,R14 TRANSFER NEW PC
0770 037E 0284 PCOK CI R4,'Y'*256 SET BK PTS?
0771 0380 5900
0772 0382 16B5 JNE GO IF NOT, BYPASS
0773 0384 C820 MOV @INITIZ,@BPTSET SET BREAKPOINT FLAG
0774 0386 0326'
0775 0388 0000
0776 038A 0202 LI R2,EBPTLP GET LOAD ADDRESS
0777 038C 0396'
0778 038E 0201 SETLOD LI R1,>FFFE SET POINTER ( & COUNTER)
0779 0390 FFFE
0780 0392 C442 MOV R2,*R1 MOVE TO LOAD VECTORS
0781 0394 10AB JMP STEP GO TO STEPPER
0782 *
0783 0396 05C1 EBPTLP INCT R1 UPDATE BPT NUMBER
0784 0398 0281 CI R1,32 DONE?
0785 039A 0020
0786 039C 1BA8 JH GO YES - EXECUTE
0787 039E C081 MOV R1,R2 GET NUMBER
0788 03A0 0912 SRL R2,1 DIVIDE BY TWO
0789 03A2 C0E1 MOV @BPTADD(R1),R3 GET BP ADDRESS
0790 03A4 01E0'
0791 03A6 13F7 JEQ EBPTLP ZERO - BYPASS
0792 03A8 0222 AI R2,>OFC0 ADD BKPT MID OPCODE
0793 03AA OFC0
0794 03AA' COD15 EQU $-2

```


EXCEUTE

0783 03AC C853
03AE 02FA'
0784 03B0 C4C2
0785 03B2 10F1
0786 *

MOV *R3,@BPTDTA(R1) SAVE DATA
MOV R2,*R3 INSTALL BP
JMP EBPTLP AND LOOP FOR NEXT

```

0789      *
0790      * MID >OF80 ENTRY POINT (BREAKPOINT)
0791      *
0792 03B4 0300  XOPENT LIM1 0          GIVE LOBUG COMPLETE CONTROL
      03B6 0000
0793 03B8 064E      DECT R14          UPDATE USER PC
0794 03BA C05E      MOV  *R14,R1      GET BREAKPOINT OPCODE
0795 03BC 0241      ANDI R1,>F        ISOLATE THE NUMBER
      03BE 000F
0796 03C0 1603      JNE  OPBPMS        NOT ZERO, JUMP
0797 03C2 C020      MOV  @BPTOV,R0     GET USER VECTORS (IF ANY)
      03C4 0000
0798 03C6 1697      JNE  CONTIN        ZERO, NO VECTORS
0799 03CB      OPBPMS MSG @BPMSG      'BP:'
0800 03CC      WNB L R1
0801 03CE 0A11      SLA  R1,1          WORD ALIGN
0802 03D0 C7A1      MOV  @BPTDTA(R1),*R14  REPLACE INSTRUCTION
      03D2 03AE'
0803 03D4 06A0      BL   @RMVBPT       REMOVE BREAKPOINTS FROM RAM
      03D6 040C'
0804 03D8 020B      LI   R11,MONTOP    RESTORE MONITOR RETURN
      03DA 0000'
0805      *
0806      * OUTPUT USER WP, PC, AND ST AT BREAKPOINT
0807      *
0808 03DC 028E  WPS10 CI  R14,ZZZZZ#    INTERNAL ROUTINE ?
      03DE 0000
0809 03E0 1A85      JL   STEP          YES, IGNORE AND STEP AGAIN
0810      *
0811      * '?' (WHERE AM I) COMMAND ENTRY POINT
0812      *
0813      03E2' QUERY EQU $
0814 03E2 020B  WPS20 LI  R8,MREG13      USER WP, PC, ST
      03E4 0360'
0815 03E6 0206      LI   R6,WS          MSG POINTER
      03E8 0364'
0816 03EA      BLOOP MSG *R9            PRINT 2 SPACES
0817 03EC 0288      CI   R8,MREG13+6    DONE?
      03EE 0006
0818 03F0 1304      JEQ  BINST          YES, PRINT INSTRUCTION NEXT
0819 03F2      MSG *R6                  PRINT 'WP=,PC=,ST=' MSG
0820 03F4      WHXW *R8+                GET WP, PC, ST
0821 03F6 8DB6      C    *R6+,*R6+      ADD 4 TO MSG POINTER
0822 03F8 10F8      JMP  BLOOP
0823      *
0824      * OUTPUT DATA AND INSTRUCTION AT PC ADDRESS
0825      *
0826 03FA 0000  BINST DATA TYP$*,PADIT  OUT CRLF AND 8 SPACES
0827 03FE 0420      BLWP @INSTR         CALL UNASSEMBLER
      0400 0814'
0828 0402 045B      RT                  EXIT TO MONITOR (OR TRACE)

```

```

0831 *****
0832 *
0833 * OUTPUT PORT TOGGLE ----- 'P'
0834 *
0835 *****
0836 0404 C800 P MOV RO,@UNIT
      0406 0000
0837 0408 0460 TOPRTN B @MONTOP
      040A 0000
0838 *
0839 * SUBROUTINE TO REMOVE BREAKPOINTS FROM USER PROGRAM
0840 *
0841 040C 0201 RMVBPT LI R1,BPTSET SET POINTER
      040E 0388
0842 0410 8811 C *R1,@INITIZ INITIALIZED?
      0412 0326
0843 0414 1609 JNE RMVRTN NO - EXIT
0844 0416 04F1 CLR *R1+ CLEAR FLAG
0845 0418 C0D1 REPLP MOV *R1,R3 GET ADDRESS (IF ANY)
0846 041A 1302 JEQ NEXTBP IF ZERO, IGNORE
0847 041C C4E1 MOV @32(R1),*R3 REPLACE INTRUCTION
      041E 0020
0848 0420 05C1 NEXTBP INCT R1
0849 0422 0281 CI R1,BPTADD+32 FINISHED?
      0424 0020
0850 0426 1AF8 JL REPLP
0851 0428 045B RMVRTN RT
0852 *

```

CHARACTER ECHO

```
0855 *****
0856 *
0857 * ECHO A CHARACTER TO THE TERMINAL (EKO) **USES EREGS** *
0858 *
0859 * --WARNING-- SYSTEM FLAGS ARE KEPT IN R0-R7 OF EREGS *
0860 *
0861 * CALLING SEQUENCE: EKO Register *
0862 *
0863 * RETURN RTWP *
0864 *
0865 * THE CHARACTER IS RETURNED IN THE MOST SIGNIFICANT BYTE *
0866 * OF THE REGISTER WITH THE LOWER BYTE ZEROED. *
0867 *****
0868 *
0869 042A ECHOEN READ *R11 READ CHARACTER
0870 042C WRIT *R11 ECHO THE CHARACTER
0871 042E 0380 RTWP
0872 *
```

```

0875 *****
0876 * READ CHARACTER (READ) ** USES IREGS ** *
0877 * * *
0878 * CALLING SEQUENCE: READ Register *
0879 * * *
0880 * RETURN RTWP *
0881 * * *
0882 * READ WAITS FOR A CHARACTER TO BE ASSEMBLED IN *
0883 * THE UART. THE CHARACTER IS PLACED IN THE LEFT *
0884 * BYTE OF USER REGISTER. THE RIGHT BYTE IS ZEROED *
0885 * * *
0886 *****
0887 *
0888 0430 C300 RENTRY MOV R0,R12 SAVE R0
0889 0432 0000 DATA GETCR$ GET A CHARACTER
0890 0434 C6C0 MOV R0,*R11 RETURN CHARACTER TO CALLER
0891 0436 C00C MOV R12,R0 RESTORE R0
0892 0438 0380 RTWP EXIT
0893 *

```

```
0896 *****
0897 * WRITE CHARACTER (WRIT) ** USES IREGS ** *
0898 * * *
0899 * CALLING SEQUENCE: WRIT Register *
0900 * * *
0901 * RETURN: RTWP *
0902 * * *
0903 * TRANSMIT THE CHARACTER IN THE LEFT BYTE OF THE USER *
0904 * REGISTER. IF THE CHARACTER IS A CARRIAGE RETURN, AND *
0905 * THE BAUD RATE IS 1200 OR LESS, THE ROUTINE DELAYS 200MS *
0906 * TO ALLOW THE CARRIAGE TO RETURN. *
0907 * IF THE TERMINAL IS A 733 ASR, THEN EACH CHARACTER *
0908 * IS PADDED WITH 25 MSEC TO REDUCE THE TRANSFER RATE TO *
0909 * 300 BAUD. *
0910 * NOTE THAT EACH TIME A CARRIAGE RETURN IS OUTPUT, A *
0911 * TEST IS MADE TO SEE IF AN 'ESCAPE' HAS BEEN RECEIVED. *
0912 * IF IT HAS, THEN CONTROL IS RETURNED TO THE COMMAND *
0913 * SCANNER. USER PROGRAMMES REQUIRING AN 'ESCAPE' INPUT *
0914 * INTO THEM SHOULD THEREFORE AVOID USING THIS XOP IF *
0915 * INPUT IS REQUIRED WHILST OUTPUTTING. *
0916 *****
0917 *
0918 043A C300 WENTRY MOV R0,R12 SAVE R0
0919 043C D01B MOV B *R11,R0 FETCH BYTE TO BE SENT
0920 043E 0000 DATA TYP0$ SEND IT
0921 *
0922 0440 981B CB *R11,@BOD 'CR' ?
0923 0442 0000
0923 0444 1602 JNE W40 N, EXIT
0924 0446 0420 BLWP @HALT0$ Y, CHECK FOR HALT OUTPUT
0925 0448 0000
0925 044A C00C W40 MOV R12,R0 RESTORE R0
0926 044C 0380 RTWP N, EXIT
```

```
0929 *****
0930 *
0931 * MESSAGE OUTPUT (MSG) ** USES XREGS **
0932 *
0933 * CALLING SEQUENCE: MSG @Message Address
0934 *
0935 * RETURN RTWP
0936 *
0937 * THE ASCII STRING POINTED TO BY MESSAGE ADDRESS IS
0938 * OUTPUT UNTIL A ZERO BYTE IS ENCOUNTERED.
0939 *
0940 *****
0941 *
0942 044E MLOOP WRIT R12 O/P THE CHARACTER
0943 0450 D33B MENTRY MOVB *R11+,R12 GET CHARACTER
0944 0452 16FD JNE MLOOP IF NOT ZERO, LOOP BACK
0945 0454 0380 RTWP RETURN
0946 *
```

```

0949 *****
0950 * HEX INPUT ROUTINE (RHXW) *
0951 * * *
0952 * CALLING SEQUENCE: RHXW @Register no. *
0953 * DATA NULL *
0954 * DATA ERROR *
0955 * RETURN RTWP *
0956 * * *
0957 * RETURNS A 16-BIT NUMBER INPUT FROM TERMINAL. DIGITS *
0958 * ARE ACCEPTED UNTIL A TERMINATION CHARACTER IS FOUND. *
0959 * * *
0960 * TERMINATION CHARACTERS: SPACE, COMMA, CAR. RETURN, MINUS *
0961 * THE TERMINATION CHARACTER IS RETURNED IN THE LEFT *
0962 * BYTE OF THE REGISTER FOLLOWING 'R'. *
0963 * * *
0964 * RETURN IS TO THE NULL RETURN ADDRESS IF INPUT IS *
0965 * A TERMINATION CHARACTER ONLY. (Rn IS UNCHANGED) *
0966 * * *
0967 * IF A FAULTY TERMINATION CHARACTER IS FOUND, *
0968 * RETURN IS TO THE ERROR ENTRY. (Rn,Rn+1 ARE UNCHANGED) *
0969 *****
0970 *
0971 0456 04C9 RHENTY CLR R9 RESET NUMBER INPUT FLAG
0972 0458 04CC CLR R12 CLEAR ACCUMULATOR
0973 045A LOOP EKO R10 GET A CHARACTER INPUT
0974 *
0975 * CHECK FOR VALID HEX INPUT
0976 *
0977 045C 028A CI R10, '0'*256 MIN NUMERIC
0978 045E 3000 JL NOTHEX
0979 0462 028A CI R10, '9'*256 MAX NUMERIC
0980 0464 3900
0981 0466 1208 JLE GOTONE
0982 0468 028A CI R10, 'A'*256 MIN ALPHA
0983 046A 4100
0984 046C 1A0B JL NOTHEX
0985 046E 028A CI R10, 'F'*256 MAX ALPHA
0986 0470 4600
0987 0472 1B0B JH NOTHEX
0988 0474 022A AI R10, >900 ALPHA ADJUST
0989 0476 0900
0990 0478 0A4A GOTONE SLA R10, 4 ISOLATE DIGIT
0991 047A 09CA SRL R10, 12 WORD ALIGN DIGIT
0992 *
0993 * DIGIT TO ACCUMULATOR
0994 *
0995 SLA R12, 4
0996 A R10, R12
0997 INC R9 SET INPUT FLAG
0998 0482 10EB JMP LOOP
0999 *
1000 * CHECK FOR TERMINATION CHARACTER
1001 *
1002 0484 028A NOTHEX CI R10, ' '*256 ' '?
1003 0486 2000
1004 0488 130B JEQ SPCK
1005 048A 028A CI R10, '- '*256 '- '?
1006 048C 2D00
1007 048E 130B JEQ SPCK

```


1002 0490 028A		CI	R10,>D00	CARRIAGE RETURN?
0492 0D00				
1003 0494 1305		JEQ	SPCK	
1004 0496 028A		CI	R10,' '*256	COMMA?
0498 2C00				
1005 049A 160C		JNE	ERR	NO, TERMINATION CHAR ERROR
1006 049C 020A		LI	R10,' '*256	CHANGE TO SPACE
049E 2000				
1007 04A0 C249	SPCK	MOV	R9,R9	NULL INPUT?
1008 04A2 1304		JEQ	NEXIT	YES, SKIP
1009 04A4 CECC		MOV	R12,*R11+	RETURN VALUE
1010 04A6 C6CA		MOV	R10,*R11	RETURN TERMINATOR
1011 04A8 8FBE		C	*R14+,*R14+	BUMP PAST NULL POINTER
1012 04AA 0380		RTWP		
1013 04AC 05CB	NEXIT	INCT	R11	
1014 04AE C6CA		MOV	R10,*R11	RETURN TERMINATOR
1015 04B0 C39E	EXIT1	MOV	*R14,R14	GET POINTER
1016 04B2 0380		RTWP		
1017 04B4 05CE	ERR	INCT	R14	POINT TO ERROR POINTER
1018 04B6 10FC		JMP	EXIT1	
1019	*			

```

1022 *****
1023 *
1024 * HEX OUTPUT ROUTINES (WNBL AND WHXW)
1025 *
1026 *
1027 * ROUTINE 1 (WHXW)
1028 * CALLING SEQUENCE: WHXW Register
1029 *
1030 * RETURN RTWP
1031 *
1032 * OUTPUT THE BINARY CONTENTS OF 'R' AS
1033 * 4 HEXADECIMAL DIGITS.
1034 *
1035 *
1036 * ROUTINE 2 (WNBL)
1037 * CALLING SEQUENCE: WNBL Register
1038 *
1039 * RETURN: RTWP
1040 *
1041 * OUTPUT RIGHT MOST HEX DIGIT IN R.
1042 *
1043 *****
1044 *
1045 * WHX1 ENTRY POINT
1046 *
1047 04B8 C31B WHXETY MOV *R11,R12 GET VALUE TO PRINT
1048 04BA 0ACC SLA R12,12
1049 04BC 0209 LI R9,1 SET COUNT FOR 1 DIGIT
1050 04BE 0001
1050 04C0 1003 JMP LOOP1
1051 *
1052 * WHEX ENTRY POINT
1053 *
1054 04C2 C31B WHENTY MOV *R11,R12 GET THE VALUE
1055 04C4 0209 LI R9,4 SET COUNT FOR 4 DIGITS OUT
1056 04C6 0004
1056 04C8 C28C LOOP1 MOV R12,R10
1057 04CA 09CA SRL R10,12 ISOLATE HEX DIGIT
1058 04CC 0ABA SLA R10,8 BYTE ALIGN
1059 04CE 028A CI R10,>900 NUMERIC?
1060 04D0 0900
1060 04D2 1202 JLE NUM YES, SKIP
1061 04D4 022A AI R10,>700 ALPHA ADJUST
1062 04D6 0700
1062 04D8 022A NUM AI R10,'0'*256 NUMERIC TO ASCII
1063 04DA 3000
1063 04DC WRIT R10 WRITE CHARACTER
1064 04DE 0BCC SRC R12,12 ALIGN NEXT DIGIT
1065 04E0 0609 DEC R9 DONE?
1066 04E2 16F2 JNE LOOP1 NO, LOOP
1067 04E4 0380 RTWP

```

```
1070 *****
1071 * TITLE: ZERO LABEL ASSEMBLER *
1072 * *
1073 * REVISION: 9/04/79 REPRINT ADDR FOR CRT'S *
1074 * 21/2/78 MODIFIED FOR HIBUG *
1075 * 22/6/81 MODIFIED FOR LOBUG *
1076 * 8/7/81 MODIFIED FOR TMS 9995 *
1077 * 10/12/81 MODIFIED TO USE DIS-ASM TABLE *
1078 * ABSTRACT: PROVIDES LIMITED ASSEMBLER *
1079 * CAPABILITY. MOST FEATURES OF *
1080 * THE 990/4 ASSEMBLER ARE INCLUDED *
1081 * EXCEPT LABEL DEFINITION/REFERENCE. *
1082 * CALLING SEQUENCE: ENTER 'A' COMMAND TO *
1083 * LOBUG *
1084 * *
1085 * THE ENTRY ADDRESS IS AT ZLABGN *
1086 *****
1087 *
1088 * GET ONE CHARACTER FROM IO BUFFER
1089 * CHARACTER RETURNED RIGHT JUSTIFIED IN R4
1090 *
1091 04E6 C80B INPT MOV R11,@BLSTOR SAVE RETURN ADDRESS
1092 04EB 0000
1092 04EA C2E0 MOV @ASMPTR,R11 FETCH IO POINTER
1093 04EC 0000
1093 04EE D11B MOVB #R11,R4 FETCH BYTE
1094 04F0 1302 JEQ INPTX SKIP IF NULL
1095 04F2 05A0 INC @ASMPTR INCREMENT IO POINTER
1096 04F4 04EC
1096 04F6 0984 INPTX SRL R4,8 RIGHT JUSTIFY
1097 04F8 C2E0 MOV @BLSTOR,R11 FETCH RETURN ADDRESS
1098 04FA 04EB
1098 04FC 045B RT
1099 *
1100 * BRANCH TABLE FOR OPERANDS
1101 *
1102 * 0 - N/A
1103 * 1 - S OR D
1104 * 2 - W OR C
1105 * 3 - IOP
1106 * 4 - N (SHIFT COUNT)
1107 * 5 - DIS
1108 * 6 - BIT
1109 *
1110 04FE 0000 OPER DATA 0,OPA,OPF,OPE
1111 0506 079C DATA OPD,OPG,OPH
1112 *
1113 *
1114 * FORMAT CODE CONVERSION TABLE
1115 *
1116 050C 00 FMCONV BYTE 0,9,5,>A,>A,>14,8,0,>13,>A,6,3,>10
1117 *
1118 051A EVEN
1119 *
1120 * HEX, BINARY, OR DECIMAL INPUT
1121 *
1122 051A C04B HEX MOV R11,R1 SAVE RETURN
1123 051C 020B LI R8,16 PRESET BASE
1124 051E 0010
1124 0520 1007 JMP DEC5
```

1125	0522	0208	BIN	LI	R8,2	PRESET BASE
	0524	0002				
1126	0526	069F		BL	*R15	
1127	0528	1003		JMP	DEC5	
1128	052A	C04B	DEC	MOV	R11,R1	SAVE RETURN
1129	052C	0208	DEC1	LI	R8,10	PRESET BASE
	052E	000A				
1130	0530	04C7	DEC5	CLR	R7	PRESET VALUE
1131	0532	C184	DEC10	MOV	R4,R6	PUT CHAR IN R6
1132	0534	0226		AI	R6,->30	REMOVE ASCII BIAS
	0536	FFD0				
1133	0538	110A		JLT	DEC30	NOT VALID
1134	053A	0286		CI	R6,10	
	053C	000A				
1135	053E	1105		JLT	DEC20	O. K.
1136	0540	0226		AI	R6,-7	
	0542	FFF9				
1137	0544	0286		CI	R6,10	
	0546	000A				
1138	0548	1102		JLT	DEC30	NOT VALID
1139	054A	B206	DEC20	C	R6,R8	IF NOT LT BASE - NOT GOOD
1140	054C	1103		JLT	DEC40	
1141	054E	C2C1	DEC30	MOV	R1,R11	RESTORE EXIT
1142	0550	C047		MOV	R7,R1	R1=ANS.
1143	0552	045B		B	*R11	EXIT
1144	0554	C006	DEC40	MOV	R6,R0	
1145	0556	C187		MOV	R7,R6	
1146	0558	398B		MPY	R8,R6	
1147	055A	A1C0		A	R0,R7	
1148	055C	069F		BL	*R15	
1149	055E	10E9		JMP	DEC10	
1150			*			
1151			* GET REGISTER NAME			
1152			*			
1153	0560	C04B	GETR	MOV	R11,R1	SAVE RET
1154	0562	069F		BL	*R15	
1155	0564	C2C1		MOV	R1,R11	TEMP. RESET OF R11
1156	0566	C34B	GETRA	MOV	R11,R13	SAVE RET
1157	0568	0284		CI	R4,'R'	
	056A	0052				
1158	056C	1601		JNE	GETR20	
1159	056E	069F		BL	*R15	
1160	0570	06A0	GETR20	BL	@DEC	GET X
	0572	052A				
1161	0574	0281		CI	R1,15	TEST RANGE
	0576	000F				
1162	0578	1B01		JH	GETR30	
1163	057A	045D		B	*R13	EXIT
1164	057C	0204	GETR30	LI	R4,'*R'	ISSUE RANGE ERROR
	057E	2A52				
1165	0580	106C		JMP	TYPE1	
1166			*			
1167			* GET ADDRESS			
1168			*			
1169	0582	C04B	GETL	MOV	R11,R1	SAVE RET
1170	0584	069F		BL	*R15	
1171	0586	1001		JMP	GETL10	
1172	0588	C04B	GETLA	MOV	R11,R1	SAVE RETURN
1173	058A	0284	GETL10	CI	R4,'%'	CHECK FOR BINARY
	058C	0025				

1174	058E	13C9	JEG	BIN	
1175	0590	0284	CI	R4,>27	CHECK FOR STRING (')
	0592	0027			
1176	0594	1305	JEG	GETL20	
1177	0596	0284	CI	R4,'>'	CHECK FOR HEX
	0598	003E			
1178	059A	16C8	JNE	DEC1	MUST BE DEFAULT
1179	059C	069F	BL	*R15	MUST BE HEX
1180	059E	10BE	JMP	HEX+2	
1181	05A0	04C7	GETL20	CLR R7	PRESET STRING
1182	05A2	069F	GETL30	BL *R15	GET A CHAR
1183	05A4	0284	CI	R4,>27	IF ', DONE
	05A6	0027			
1184	05A8	1303	JEG	GETL40	
1185	05AA	0A87	SLA	R7,8	
1186	05AC	E1C4	SOC	R4,R7	
1187	05AE	10F9	JMP	GETL30	
1188	05B0	069F	GETL40	BL *R15	GET TERM.
1189	05B2	10CD	JMP	DEC30	EXIT
1190					
1191			*		
1192			*	CONTROL LOOP - REQUEST ADDRESS,	
1193			*	PRINT TRANSLATED OPCODES	
1194			*		
1194	05B4'		ZLABGN	EGU \$	*** ENTRY ***
1195	05B4	02E0	LWPI	WORKS	SET WORKSPACE
	05B6	0000			
1196	05BB	0201	LI	R1,BRAM	DEFAULT START PC
	05BA	0000			
1197	05BC	00A0	MOV	@MREG3,R2	WAS A PC ENTERED?
	05BE	0000			
1198	05C0	1302	JEG	PT100	NO, USE DEFAULT
1199	05C2	0060	MOV	@MREGS,R1	YES, USE ENTRY
	05C4	00F4'			
1200	05C6	020F	PT100	LI R15,INPT	SET R15 FOR INPT CALL
	05C8	04E6'			
1201	05CA	0209	LI	R9,GETL	
	05CC	0582'			
1202	05CE	0801	PT110	MOV R1,@PC	SAVE PC
	05D0	0000			
1203	05D2	00A0	PT120	MOV @PC,R2	R2=PC
	05D4	05D0'			
1204	05D6	04C3	CLR	R3	R3=WORD COUNT
1205	05DB	0358'	PT130	DATA TYPC\$	PRINT CR LF
1206	05DA		PT135	WHXW R2	PRINT (R2) IN HEX
1207	05DC	0205	LI	R5,' '	SPACE OVER ONE
	05DE	2020			
1208	05E0		WRIT	R5	
1209	05E2	00C3	MOV	R3,R3	IF WORD COUNT NONZERO
1210	05E4	1305	JEG	PT150	DISPLAY INST. WORDS
1211	05E6		WHXW	*R2+	
1212	05EB	0802	MOV	R2,@PC	UPDATE PC
	05EA	05D4'			
1213	05EC	0643	DECT	R3	REDUCE WORD COUNT
1214	05EE	10F4	JMP	PT130	CONT. TILL ALL DONE
1215	05F0		PT150	MSG @SPACE5	TAB OVER 6 PLACES
1216			*		
1217			*	ACCEPT THE OP-CODE MNEMONIC	
1218			*		
1219	05F4	0720	SETO	@MODE	STOP USER USING ^E (EDIT)
	05F6	0000			

1220	05F8 04E0	CLR	@ESCFLG	MAKE SURE USER CAN ESCAPE
	05FA 0000			
1221	05FC 02E0	LWPI	WPR1	USE BASIC'S WP
	05FF 0116'			
1222	0600 06A0	BL	@GTLN	CALL GET LINE
	0602 0000			
1223	0604 02E0	LWPI	WORKS	RESTORE WP
	0606 05B6'			
1224	0608 C820	MOV	@IOB, @ASMPTR	SET UP IO POINTER
	060A 0000			
	060C 04F4'			
1225	060E 020A	LI	R10, ASMOPC	POINT TO BUFFER
	0610 0000			
1226	0612 C685	MOV	R5, *R10	SET 2 SPACES
1227	0614 C805	MOV	R5, @ASMOPC+2	SET 2 MORE SPACES
	0616 0002			
1228	0618 069F	GETASC BL	*R15	GET A CHARACTER
1229	061A 0284	CI	R4, 'A'	TEST IF VALID ALPHA
	061C 0041			
1230	061E 1109	JLT	NOTOPC	NOT OPCODE
1231	0620 0284	CI	R4, 'Z'	TEST UPPER ALPHA
	0622 005A			
1232	0624 1506	JGT	NOTOPC	NOT OPCODE
1233	0626 06C4	GOTASC SWPB	R4	
1234	0628 DE84	MOVB	R4, *R10+	STORE THE CHARACTER
1235	062A 028A	CI	R10, ASMOPC+4	CHECK NOT TOO MANY CHARS.
	062C 0004			
1236	062E 1B13	JH	PAT90	TOO MANY - JUMP
1237	0630 10F3	JMP	GETASC	GET ANOTHER CHAR
1238		*		
1239	0632 0284	NOTOPC CI	R4, ' '	IF SPACE, FINISHED OPCODE
	0634 0020			
1240	0636 1333	JEQ	GOTOPC	
1241	0638 C104	MOV	R4, R4	IF NULL, FINISHED OPCODE
1242	063A 1331	JEQ	GOTOPC	
1243	063C 028A	CI	R10, ASMOPC	TEST IF FIRST CHARACTER
	063E 0610'			
1244	0640 160A	JNE	PAT90	IF NOT, ERROR IT
1245	0642 0284	CI	R4, '\$'	CHECK FOR \$(STRING)
	0644 0024			
1246	0646 130D	JEQ	PT220	
1247	0648 0284	CI	R4, '/'	CHECK FOR ADDR RESET
	064A 002F			
1248	064C 1604	JNE	PAT90	
1249	064E 069F	BL	*R15	GET ANOTHER CHARACTER
1250	0650 06A0	BL	@HEX	GET NEW ADDRESS
	0652 051A'			
1251	0654 10BC	JMP	PT110	
1252	0656 0204	PAT90 LI	R4, '*S'	ERROR - SNATCH CONTROL
	0658 2A53			
1253	065A 0205	TYPE1 LI	R5, >0700	SET UP BELL AND ZERO BYTE
	065C 0700			
1254	065E	MSG	R4	OUTPUT R4
1255	0660 10B8	JMP	PT120	DON'T CHANGE PC
1256		*		
1257		* HANDLE STRING ENTRIES.	COLLECT CHARACTERS	
1258		* UNTIL A NULL.	THEN FORCE ADDRESS EVEN AND	
1259		* EXIT		
1260		*		
1261	0662 069F	PT220 BL	*R15	GET A CHAR.

```
1262 0664 C104      MOV R4,R4      IF NULL - EXIT
1263 0666 1304      JEQ PT230
1264 0668 0A84      SLA R4,8      SAVE THE CHAR.
1265 066A DC84      MOV B R4,*R2+
1266 066C 0583      INC R3
1267 066E 10F9      JMP PT220
1268 0670 C003 PT230 MOV R3,R0      IF ODD-INST. SPACE
1269 0672 0810      SRA R0,1
1270 0674 1702      JNC PT240
1271 0676 D485      MOV B R5,*R2      PAD WITH SPACE
1272 0678 0583      INC R3
1273 067A C0A0 PT240 MOV @PC,R2      RESET PC
      067C 05EA
1274 067E 1039      JMP PT300      GO PRINT RESULTS
1275      *
1276      * GET SPECIAL OPCODES FROM TABLE
1277      *
1278 0680 0205 SPECDC LI R5,SOPSTT-6      GET TABLE START
      0682 0BCA
1279 0684 05CA      INCT R10
1280 0686 0225 SPEC1 AI R5,6
      0688 0006
1281 068A 064A      DECT R10
1282 068C C035      MOV *R5+,R0
1283 068E 13E3      JEQ PAT90      NOT FOUND, ERROR
1284 0690 8E80      C R0,*R10+
1285 0692 16F9      JNE SPEC1
1286 0694 8695      C *R5,*R10
1287 0696 16F7      JNE SPEC1
1288 0698 C065      MOV @4(R5),R1
      069A 0004
1289 069C 107B      JMP PT270
1290      *
1291      * THE INPUT OPCODE IS NOW IN ASMOPC. SEARCH THE TABLE
1292      * UNTIL IT IS LOCATED.
1293      *
1294 069E 0205 GOTOPC LI R5,CODSTT-4      GET TABLE ADDRESS
      06A0 09CC
1295 06A2 020A      LI R10,ASMOPC+2      INPUT OPCODE PTR
      06A4 0002
1296 06A6 0225 OCSH1 AI R5,4      POINT TO NEXT OPCODE
      06AB 0004
1297 06AA 064A      DECT R10      BACK TO OPCODE START
1298 06AC C035      MOV *R5+,R0      GET DATA
1299 06AE 13E8      JEQ SPECDC      ZERO, END OF TABLE
1300 06B0 8E80      C R0,*R10+      CHECK FOR EQUALITY
1301 06B2 16F9      JNE OCSH1      NO, -LOOP BACK
1302 06B4 8695      C *R5,*R10      CHECK LAST TWO LETTERS
1303 06B6 16F7      JNE OCSH1
1304      *
1305      * THE OPCODE HAS BEEN LOCATED . NOW
1306      * COLLECT THE OPERANDS.
1307      *
1308 06B8 C2A5 PT280 MOV @2(R5),R10      R10=INST&PARSING INST.
      06BA 0002
1309 06BC C00A      MOV R10,R0      PRESET THE INST.
1310 06BE 0240      ANDI R0,>FFFO
      06C0 FFF0
1311 06C2 C480      MOV R0,*R2
1312 06C4 05C3      INCT R3      COUNT=2
```

```
1313 06C6 C04A      MOV  R10,R1      SAVE
1314 06C8 0241      ANDI  R1,>F      REMOVE OP CODE, LEAVE FORMAT
      06CA 000F
1315 06CC 1353      JEQ   GETDTA
1316 06CE D061      MOVB  @FMCONV(R1),R1  GET TRANSLATION
      06D0 050C'
1317 06D2 0981      SRL   R1,8        TO LSB
1318 06D4 C301      MOV   R1,R12      STORE FOR FUTURE USE
1319 06D6 0921      SRL   R1,2
1320 06D8 0241      ANDI  R1,>6
      06DA 0006
1321 06DC C061      MOV   @OPER(R1),R1  R1=OPERAND INDEX
      06DE 04FE'
1322 06E0 1301      JEQ   PT290        SKIP IF NO FIRST ONE
1323 06E2 0691      BL    *R1          COLLECT FIRST ONE
1324 06E4 0ADC      PT290  SLA   R12,13  COLLECT SECOND OPERAND.
1325 06E6 09CC      SRL   R12,12
1326 06E8 C1AC      MOV   @OPER(R12),R6
      06EA 04FE'
1327 06EC 1302      JEQ   PT300        JUMP IF NONE
1328 06EE 04CA      CLR   R10        SET FLAG
1329 06F0 0696      BL    *R6
1330
1331      *
1332      * THE ENTIRE STATEMENT HAS BEEN
1333      * ACCEPTED - PRINT TRANSLATION
1334      * AND UPDATE P.C.
1335      *
1335 06F2 C104      PT300  MOV   R4,R4      EOL ?
1336 06F4 1302      JEQ   PT310        YES, JUMP
1337 06F6 069F      BL    *R15        NO, FETCH NEXT CHAR
1338 06F8 10FC      JMP   PT300        LOOP
1339 06FA          PT310  WRIT  @MCRLF    PRINT CR
1340 06FE 0460      B      @PT135      GO DISPLAY ADDR AND OBJECT
      0700 05DA'
1341
1342      *
1343      * RANGE ERROR
1344      *
1344 0702 0204      RNGERR LI   R4, '*D'
      0704 2A44
1345 0706 10A9      JMP   TYPE1
1346
1347      *
1347      * HANDLE S OR D
1348      *
1348      * N
1349      *
1349      * *N
1350      *
1350      * *N+
1351      *
1351      * @X(N)
1352      *
1352      * @X
1353      *
1354 0708 C38B      OPA   MOV   R11,R14    SAVE RETURN ADDRESS
1355 070A 069F      BL    *R15        GET CHAR
1356 070C 0284      CI     R4, '*'      CHECK FOR *N OR *N+
      070E 002A
1357 0710 1322      JEQ   OPB        JUMP IF YES
1358 0712 0284      CI     R4, '@'      CHECK FOR @X OR @X(N)
      0714 0040
1359 0716 162B      JNE   OPC        JUMP IF NOT
1360 0718 0699      BL    *R9
1361 071A C183      MOV   R3,R6        ADD TO MEMORY
1362 071C A182      A      R2,R6
1363 071E C581      MOV   R1,*R6        SAVE X
```


1364	0720	05C3		INCT R3	UPDATE COUNT
1365	0722	0201		LI R1,>20	ADDRESS MODE 2
	0724	0020			
1366	0726	C104		MOV R4,R4	IF NULL OR ',' DONE
1367	0728	1311		JEG OPA10	
1368	072A	0284		CI R4,','	
	072C	002C			
1369	072E	130E		JEG OPA10	
1370	0730	0284		CI R4,','	IF SPACE - DONE
	0732	0020			
1371	0734	130B		JEG OPA10	
1372	0736	0284		CI R4,('(IF NOT (- ERROR
	0738	0028			
1373	073A	168D		JNE PAT90	
1374	073C	06A0		BL @GETR	GET REG. N
	073E	0560'			
1375	0740	0261		ORI R1,>20	SET MODE 2
	0742	0020			
1376	0744	0284		CI R4,')'	IF NOT) - ERROR
	0746	0029			
1377	0748	1686		JNE PAT90	
1378	074A	069F		BL *R15	
1379	074C	C00A	OPA10	MOV R10,R0	REPOS. IT
1380	074E	1601		JNE OPA15	
1381	0750	0A61		SLA R1,6	
1382	0752	E481	OPA15	SOC R1,*R2	INSERT IT
1383	0754	045E	OPA20	B *R14	EXIT
1384	0756	06A0	OPB	BL @GETR	GET N(FOR *N)
	0758	0560'			
1385	075A	0200		LI R0,>10	SET MODE = 1
	075C	0010			
1386	075E	0284		CI R4,'+'	IF TERM. BY +
	0760	002B			
1387	0762	1603		JNE OPB10	CHANGE MODE
1388	0764	069F		BL *R15	
1389	0766	0200		LI R0,>30	SET MODE = 3
	0768	0030			
1390	076A	E040	OPB10	SOC R0,R1	R1=REG&MODE
1391	076C	10EF		JMP OPA10	
1392	076E	06A0	OPC	BL @GETRA	GET N(FOR N)
	0770	0566'			
1393	0772	10EC		JMP OPA10	MODE=0 - GO INSERT
1394			*		
1395			* HANDLE DATA ENTRIES.		
1396			*		
1397	0774	069F	GETDTA	BL *R15	GET NEXT CHARACTER
1398	0776	04CA		CLR R10	DEFAULT TO +
1399	0778	0284		CI R4,'+'	
	077A	002B			
1400	077C	1304		JEG DTASUB	
1401	077E	0284		CI R4,'-'	
	0780	002D			
1402	0782	1603		JNE NOSIGN	
1403	0784	070A		SETO R10	SET FLAG
1404	0786	0699	DTASUB	BL *R9	
1405	0788	1002		JMP TSTSGN	
1406	078A	06A0	NOSIGN	BL @GETLA	
	078C	058B'			
1407	078E	C28A	TSTSGN	MOV R10,R10	
1408	0790	1301		JEG PT270	

```

1409 0792 0501          NEG R1
1410 0794 C481 PT270 MOV R1,*R2          SAVE IT
1411 0796 0203          LI R3,2          SET R3
          0798 0002
1412 079A 10AB          JMP PT300          GO PRINT
1413          *
1414          * HANDLE SHIFT COUNT
1415          *
1416 079C C38B OPD MOV R11,R14          SAVE RETURN
1417 079E 06A0          BL @GETR          GET COUNT
          07A0 0560'
1418 07A2 0A41          SLA R1,4          REPOSITION
1419 07A4 10D6          JMP OPA15          INSERT
1420          *
1421          * HANDLE IMMEDIATE OPERANDS
1422          *
1423 07A6 C38B OPE MOV R11,R14          SAVE RETURN
1424 07A8 0699          BL *R9          GET IOP
1425 07AA C183          MOV R3,R6          ADD TO MEMORY
1426 07AC A182          A R2,R6
1427 07AE C581          MOV R1,*R6
1428 07B0 05C3          INCT R3          ADJUST COUNT
1429 07B2 045E          B *R14          CONTINUE
1430          *
1431          * HANDLE W
1432          *
1433 07B4 C38B OPF MOV R11,R14
1434 07B6 06A0          BL @GETR
          07B8 0560'
1435 07BA 10CB          JMP OPA10
1436          *
1437          * HANDLE DISPLACEMENTS
1438          * + DIS
1439          * - DIS
1440          * ADDRESS (CALCULATE DISPLACEMENT)
1441          *
1442 07BC C38B OPG MOV R11,R14          SAVE RETURN
1443 07BE 069F          BL *R15          GET LEADER (*)
1444 07C0 0284          CI R4,'$'
          07C2 0024
1445 07C4 1607          JNE OPG5
1446 07C6 069F          BL *R15          GET FIRST CHAR
1447 07C8 0284          CI R4,'+'          CHECK FOR +DIS
          07CA 002B
1448 07CC 1315          JEQ OPG30
1449 07CE 0284          CI R4,'-'          CHECK FOR -DIS
          07D0 002D
1450 07D2 1315          JEQ OPG40
1451 07D4 06A0 OPG5 BL @GETLA
          07D6 0588'
1452 07D8 C002          MOV R2,R0          MUST BE ADDRESS
1453 07DA 05C0          INCT R0          DIS*2=ADDRESS-(PC+2)
1454 07DC 6040          S R0,R1
1455 07DE 0811 OPG10 SRA R1,1          DISP=BYTE STUFF/2
1456 07E0 0281          CI R1,>7F          CHECK RANGE
          07E2 007F
1457 07E4 158E          JGT RNGERR
1458 07E6 0281          CI R1,>FF80
          07E8 FF80
1459 07EA 118B          JLT RNGERR

```

1460	07EC	0241	OPG15	ANDI	R1,>FF	RANGE O.K. SO
	07EE	00FF				
1461	07F0	E481		SOC	R1,*R2	INSERT IT
1462	07F2	0201		LI	R1,2	RESET R3
	07F4	0002				
1463	07F6	045E		B	*R14	EXIT
1464	07F8	0699	OPG30	BL	*R9	+DIS
1465	07FA	0641	OPG35	DECT	R1	ADJUST DIS FOR CUR. INST
1466	07FC	10F0		JMP	OPG10	
1467	07FE	0699	OPG40	BL	*R9	
1468	0800	0501		NEG	R1	-DIS
1469	0802	10FB		JMP	OPG35	
1470			*			
1471			* HANDLE BIT			
1472			*			
1473	0804	C38B	OPH	MOV	R11,R14	SAVE RETURN
1474	0806	0699		BL	*R9	
1475	0808	10F1		JMP	OPG15	GO PROCESS IT

```

1478 *****
1479 **
1480 **          TMS 9900  DISASSEMBLER          **
1481 **
1482 **          PROGRAMMED BY:  M. J. STEWART      **
1483 **          MODIFIED BY:    C.R. HINSON  19TH JUNE 1981  **
1484 **          COPYRIGHT 1978                      **
1485 **
1486 *****
1487 *
1488 *  INTERNAL REGISTER USAGE:
1489 *      R0  DISASSEMBLY ADDRESS
1490 *      R1  OP CODE DATA / MNEMONIC ADDRESS
1491 *      R2  PRGM CALCULATIONS
1492 *      R3  PRGM CALCULATIONS
1493 *      R4  PRGM CALCULATIONS
1494 *      R5  DUMP MODE END ADDRESS / FLAG
1495 *      R6  ASCII DATA
1496 *      R7  PGM CALCULATIONS
1497 *      R8  ADDR. OPERAND ADDR. MODES SUBR.
1498 *      R9  ADDR. PRINT REG. NO. SUBR.
1499 *      R10 PGM CALCULATIONS
1500 *      R11 LINK ADDRESS
1501 *      R12 SAVED LINK / UART CRU BASE ADDR.
1502 *      R13 CALLERS WP
1503 *      R14 CALLERS PC
1504 *      R15 CALLERS ST
1505 *
1506 *  MONITOR LINKAGE
1507 *
1508 080A' U      EQU $
1509 080A 0420    BLWP @UNVEC          CALL UNASSEMBLER
1510 080C 0810'
1511 080E 045B    RT                  RETURN TO MONITOR
1512 *
1512 0810 0606' UNVEC  DATA WORKS      VECTORS FOR UNASSEMBLER
1513 0812 0822'    DATA UENTRY
1514 *
1515 0814 0810' INSTR DATA WORKS, TRCENT
1516 *
1517 0818 C02D    TRCENT MOV  @28(R13), R0  GET START ADDRESS
1518 081A 001C
1518 081C C140    MOV  R0, R5          COPY FOR STOP ADDRESS
1519 081E 04CC    CLR  R12
1520 0820 100E    JMP  U2              DO DISASSEMBLY

```

1522		*		
1523		*	* TRANSFER START AND STOP ADDRESSES FROM CALLER	
1524		*		
1525	0822'	UENTRY	EQU	\$ *** ENTRY ***
1526	0822 C16D		MOV	@2(R13),R5 LOAD STOP ADDR
	0824 0002			
1527	0826 C01D		MOV	*R13,R0 LOAD START ADDR
1528	0828 1605		JNE	BEGIN IF NON-ZERO, GO ON
1529	082A C1AD		MOV	@6(R13),R6 ANY PARMS ?
	082C 0006			
1530	082E 1602		JNE	BEGIN YES , SKIP DEFAULT
1531	0830 0200		LI	R0,BRAM OTHERWISE LOAD DEFAULT
	0832 05BA'			
1532		*		
1533		*	* BEGIN DISASSEMBLY	
1534		*		
1535	0834	BEGIN	MSG	@MCRLF SEND CR AND LF
1536	0838 0240		ANDI	R0,>FFFE WORD ALIGN ADDRESS
	083A FFFE			
1537	083C		WHXW	R0 PRINT MEM ADDRESS
1538	083E' U2		EQU	\$ *** ALTERNATE ENTRY FOR TRACE
1539	083E 0206		LI	R6,' , ' ASCII SPACE AND COMMA
	0840 202C			
1540	0842		WRIT	R6 PRINT SPACE
1541	0844 C050		MOV	*R0,R1 GET INSTRUCTION OP CODE
1542	0846		WHXW	R1 PRINT OP CODE DATA
1543	0848		WRIT	R6 PRINT SPACE
1544	084A 028C		CI	R12,>0400
	084C 0400			
1545	084E 1343		JEQ	DATAOP
1546	0850 0208	LDREG	LI	R8,AMODE ADDR FOR ADDRESSING MODES SUBR
	0852 0988'			
1547	0854 0209		LI	R9,REGNO ADDR FOR PRINT REG NUMBER SUBR
	0856 096A'			
1548	0858 020A		LI	R10,SOPSTT-2
	085A 0BCE'			
1549	085C 022A	SOPLP	AI	R10,8
	085E 0008			
1550	0860 028A		CI	R10,SOPEND
	0862 0BE8'			
1551	0864 1B08		JH	NOTSOP
1552	0866 8681		C	R1,*R10
1553	0868 16F9		JNE	SOPLP
1554	086A C04A		MOV	R10,R1
1555	086C 0221		AI	R1,-6
	086E FFFA			
1556	0870 06A0		BL	@PRTOP
	0872 0906'			
1557	0874 106D		JMP	CMDSCN
1558	0876 020A	NOTSOP	LI	R10,CODEND GET TABLE POINTER
	0878 0BCC'			
1559	087A C09A	GETOPC	MOV	*R10,R2 GET OPCODE VALUE
1560	087C 132C		JEQ	DATAOP
1561	087E 0242		ANDI	R2,>FFFO REMOVE FORMAT
	0880 FFF0			
1562	0882 1302		JEQ	NODIR
1563	0884 8081		C	R1,R2 CORRECT OPCODE?
1564	0886 1403		JHE	OPCFND YES - EXIT
1565	0888 022A	NODIR	AI	R10,-6 NEXT OPCODE
	088A FFFA			

1566	08BC	10F6	JMP	GETOPC	LOOP BACK
1567	08BE	C1DA	OPCFND	MOV *R10,R7	STORE VALUE
1568	0890	C04A	MOV	R10,R1	STORE VALUE
1569	0892	0221	AI	R1,-4	GET TEXT PNTR
	0894	FFFC			
1570	0896	06A0	BL	@PRTOP	
	0898	0906			
1571			*		
1572	089A	0247	ANDI	R7,>F	GET THE FORMAT
	089C	000F			
1573	089E	D1E7	MOVB	@BTABL(R7),R7	GET THE ADDRESS
	08A0	08A8			
1574	08A2	0877	SRA	R7,7	
1575	08A4	0467	B	@BTABL(R7)	GO EXECUTE AS REQUIRED
	08A6	08A8			
1576			*		
1577			*	FORMAT JUMP TABLE	
1578			*		
1579	08A8	1B	BTABL	BYTE PRTDAT-BTABL/2	
1580	08A9	07		BYTE FRMAT1-BTABL/2	
1581	08AA	13		BYTE FRMAT2-BTABL/2	
1582	08AB	0D		BYTE FRMAT3-BTABL/2	
1583	08AC	0A		BYTE FRMAT4-BTABL/2	
1584	08AD	1F		BYTE FRMAT5-BTABL/2	
1585	08AE	11		BYTE FRMAT6-BTABL/2	
1586	08AF	54		BYTE CMDSCN-BTABL/2	
1587	08B0	26		BYTE FRMAT8-BTABL/2	
1588	08B1	0D		BYTE FRMAT3-BTABL/2	
1589	08B2	39		BYTE BITFMT-BTABL/2	
1590	08B3	2A		BYTE FRMATB-BTABL/2	
1591	08B4	2C		BYTE FRMATC-BTABL/2	
1592	08B6			EVEN	
1593			*		
1594	08B6	0698	FRMAT1	BL *R8	PRINT SOURCE OPERAND
1595	08B8	0961		SRL R1,6	
1596	08BA	1006		JMP TST42	
1597	08BC	0698	FRMAT4	BL *R8	PRINT OPERAND
1598	08BE	0961		SRL R1,6	
1599	08C0	1016		JMP TST81	
1600	08C2	0698	FRMAT3	BL *R8	PRINT SOURCE OPERAND
1601	08C4	0A61		SLA R1,6	
1602	08C6	09C1		SRL R1,12	
1603	08C8		TST42	WRIT R6	PRINT COMMA
1604	08CA	0698	FRMAT6	BL *R8	PRINT DESTN OPERAND
1605	08CC	1041		JMP CMDSCN	
1606	08CE	06C1	FRMAT2	SWPB R1	
1607	08D0	0871		SRA R1,7	SIGNED DISPLACEMENT
1608	08D2	A040		A R0,R1	NEW PC ADDRESS
1609	08D4	1004		JMP PRTDAT	
1610	08D6	0201	DATAOP	LI R1,DTATXT	
	08D8	09DC			
1611	08DA	06A0		BL @PRTOP	PRINT MNEMONIC
	08DC	0906			
1612	08DE		PRTDAT	WRIT @SYMBLC+1	PRINT ">"
1613	08E2			WHXW R1	PRINT DATA WORD
1614	08E4	1035		JMP CMDSCN	
1615	08E6		FRMAT5	MSG @REGSTR	
1616	08EA	0699		BL *R9	PRINT REG NUMBER
1617	08EC	0941		SRL R1,4	
1618	08EE		TST81	WRIT R6	PRINT COMMA

1619	08F0	0699	TSTB2	BL	*R9	PRINT NUMBER
1620	08F2	102E	JCMD	JMP	CMDSCN	
1621	08F4	0241	FRMATB	ANDI	R1,>F	
	08F6	000F				
1622	08F8	0698		BL	*R8	PRINT DESTINATION REGISTER
1623	08FA			WRIT	R6	PRINT COMMA
1624	08FC	C070	FRMATB	MOV	*R0+,R1	GET IMMEDIATE OPERAND
1625	08FE	10EF		JMP	PRTDAT	
1626	0900	0241	FRMATC	ANDI	R1,>F	
	0902	000F				
1627	0904	10E2		JMP	FRMAT6	
1628			*			
1629			*	SUBROUTINE WHICH PRINTS THE MNEMONIC BASED ON		
1630			*	THE TABLE OFFSET COMPUTED IN REG 1		
1631			*			
1632	0906	0203	PRTOP	LI	R3,4	LOAD CHAR COUNT
	0908	0004				
1633	090A	D0B1	PRT1	MOVB	*R1+,R2	GET CHARACTER FROM TABLE
1634	090C			WRIT	R2	PRINT CHARACTER
1635	090E	0603		DEC	R3	DECREMENT COUNT
1636	0910	16FC		JNE	PRT1	REPEAT FOR FOUR CHAR
1637	0912			WRIT	R6	PRINT SPACE
1638	0914	06C6		SWPB	R6	ASCII ',' IN UPPER BYTE
1639	0916	C070		MOV	*R0+,R1	RELOAD OP CODE DATA, INC ADDR
1640	0918	045B		RT		

1642		*		
1643		*	* ROUTINE FOR PRINTING THE CRU BIT DISPLACEMENT AS	
1644		*	* A SIGNED DECIMAL NUMBER (127 TO -128)	
1645		*		
1646	091A 0203	BITFMT LI	R3, '-1'	ASCII DATA FOR MINUS AND ONE
	091C 2D31			
1647	091E 06C1	SWPB R1		ISOLATE BIT DISPLACEMENT
1648	0920 08B1	SRA R1,8		AND SIGN EXTEND
1649	0922 1315	JEQ BIT5		JUMP IF ZERO
1650	0924 1502	JGT BIT1		JUMP IF POSITIVE
1651	0926	WRIT R3		OTHERWISE PRINT MINUS SIGN
1652	0928 0501	NEG R1		AND MAKE POSITIVE
1653	092A 0281	BIT1 CI	R1,100	ONE HUNDRED OR GREATER?
	092C 0064			
1654	092E 1104	JLT BIT2		NO, JUMP
1655	0930 0A83	SLA R3,8		MOVE OVER ASCII ONE
1656	0932	WRIT R3		PRINT A ONE
1657	0934 0221	AI	R1,-100	SUBTRACT 100 FROM DISPLACEMENT
	0936 FF9C			
1658	0938 0204	BIT2 LI	R4,10	LOAD DIVISOR
	093A 000A			
1659	093C C081	MOV	R1,R2	RIGHT JUSTIFY DISPLACEMENT
1660	093E 04C1	CLR	R1	IN REG PAIR 1 AND 2
1661	0940 3C44	DIV	R4,R1	DIVIDE BY TEN
1662	0942 0A83	SLA	R3,8	WAS ONE HUNDRED PRINTED?
1663	0944 1302	JEQ BIT3		YES, PRINT BOTH TENS AND ONES
1664	0946 C041	MOV	R1,R1	NO, THEN TEST TENS VALUE
1665	0948 1301	JEQ BIT4		IF ZERO PRINT ONLY ONES
1666	094A 0699	BIT3 BL	*R9	PRINT TENS DIGIT
1667	094C C042	BIT4 MOV	R2,R1	LOAD ONES DIGIT
1668	094E 0699	BIT5 BL	*R9	PRINT ONES DIGIT


```

1670      *
1671      * COMMAND SCANNER: TWO MODES
1672      * 1) DUMP MODE:  DISASSEMBLES FROM START TO END
1673      *      ADDRESS.
1674      * 2) SINGLE STEP MODE:  AT THE END OF EACH LINE,
1675      *      A) ESCAPE CAUSES A RETURN TO LOBUG CMD SCANNER
1676      *      B) ANY OTHER CHARACTER CONTINUES DIS-ASSEMBLY AT THE
1677      *      NEXT INSTRUCTION
1678      *
1679 0950 C145  CMDSCN MOV  R5,R5      TEST FOR DUMP MODE
1680 0952 1608      JNE  CMD5      Y, GO TO DUMP HANDLER
1681      *
1682 0954      MSG  @SPACE2      OUTPUT TWO SPACES
1683 0958  CMD2  READ  R12
1684 095A 028C      CI   R12,>1B00  ESCAPE?
1685      095C 1B00
1685 095E 1304      JEQ  CMD6      Y, EXIT DISASSEMBLER
1686 0960 0460  CMD4  B      @BEGIN  CONTINUE DISASSEMBLY
1687      0962 0834
1687      *
1688 0964 8140  CMD5  C      R0,R5    TEST DUMP END ADDRESS
1689 0966 12FC      JLE  CMD4      N, CONTINUE DISASSEMBLY
1690 0968 0380  CMD6  RTWP      Y, EXIT DISASSEMBLER

```

```

1692      *
1693      * SUBROUTINE WHICH PRINTS THE DECIMAL VALUE OF
1694      * THE LOWER FOUR BITS OF REGISTER 1
1695      *
1696 096A C0C1 REGNO MOV R1,R3      MOVE DATA FROM R1 TO R3 AND
1697 096C 0AC3      SLA R3,12      ISOLATE LOWER 4 BITS RIGHT
1698 096E 0943      SRL R3,4       JUSTIFIED IN UPPER BYTE
1699 0970 0283      CI R3,>900     COMPARE TO NINE
      0972 0900
1700 0974 1203      JLE REG1      EQUAL OR LESS, SINGLE DIGIT
1701 0976 06C3      SWPB R3       OTHERWISE TWO DIGITS
1702 0978 0223      AI R3,>0126    BCD CORRECT ONES, MAKE ASCII
      097A 0126
1703 097C 0223 REG1 AI R3,>3000    MAKE UPPER BYTE ASCII
      097E 3000
1704 0980      REG2 WRIT R3      PRINT ASCII NUMBER
1705 0982 0A83      SLA R3,8      CHECK FOR SECOND DIGIT
1706 0984 16FD      JNE REG2     IF THERE, PRINT IT
1707 0986 045B      REGRTN RT     RETURN
1708      *
1709      * SUBROUTINE TO HANDLE THE MULTIPLE ADDRESSING MODES:
1710      * WORKSPACE REGISTER
1711      * WORKSPACE REGISTER INDIRECT
1712      * SYMBOLIC AND INDEXED
1713      * WORKSPACE REG. INDIRECT AUTO INCREMENT
1714      *
1715 0988 0204 AMODE LI R4,'*R'     ACSII DATA FOR REG FORMATS
      098A 2A52
1716 098C C081      MOV R1,R2     GET OP CODE AND
1717 098E 0AA2      SLA R2,10      ISOLATE ADDR MODE (T) FIELD
1718 0990 09E2      SRL R2,14      RIGHT JUSTIFIED IN R2
1719      *
1720      * WORKSPACE REGISTER
1721      *
1722 0992 1603      JNE AMD2      JUMP IF T FIELD NOT ZERO
1723 0994 06C4 AMD1 SWPB R4      PUT ASCII "R" IN UPPER BYTE
1724 0996      WRIT R4          PRINT "R"
1725 0998 10E8      JMP REGNO     PRINT REG. NO. AND RETURN
1726      *
1727      * WORKSPACE REGISTER INDIRECT
1728      *
1729 099A 0602 AMD2 DEC R2        WAS T FIELD = 1?
1730 099C 1602      JNE AMD3      NO, JUMP
1731 099E      WRIT R4          PRINT "*"
1732 09A0 10F9      JMP AMD1      GO HANDLE REG FORMAT
1733      *
1734      * SYMBOLIC AND INDEXED
1735      *
1736 09A2 0602 AMD3 DEC R2        WAS T FIELD = 2?
1737 09A4 1610      JNE AMD6      NO, JUMP
1738 09A6      MSG @SYMBLC      ASCII DATA FOR SYMBOLIC
1739 09AA      WHXW *RO+        PRINT SYMBOLIC WORD
1740 09AC C081      MOV R1,R2     GET OP CODE AGAIN AND
1741 09AE 0AC2      SLA R2,12      CHECK IF REG. NO. IS ZERO
1742 09B0 13EA      JEQ REGRTN    YES, RETURN
1743 09B2 0202      LI R2,'()'    NO, INDEXED MODE
      09B4 2829
1744 09B6      WRIT R2          PRINT "("
1745 09B8 06C4 AMD4 SWPB R4      PUT ASCII R IN UPPER BYTE
1746 09BA      WRIT R4          PRINT "R"

```

```

1747 09BC C30B      MOV R11,R12      SAVE LINK ADDR
1748 09BE 0699      BL *R9        PRINT REG NO.
1749 09C0 06C2      SWPB R2       NEXT ASCII BYTE
1750 09C2           WRIT R2       PRINT ")" OR PRINT "+"
1751 09C4 045C      B *R12        RETURN
1752                *
1753                *      WORKSPACE REG. INDIRECT AUTO INCREMENT
1754                *
1755 09C6           AMD6      WRIT R4      PRINT "*"
1756 09C8 0202      LI R2,'+'      LOAD ASCII "+"
      09CA 002B
1757 09CC 10F5      JMP AMD4      GO HANDLE REST OF FORMAT

```

```

1760      *
1761      *****
1762      *          OP-CODE TABLE          *
1763      *
1764      * EACH ENTRY HAS THE OP CODE, OPERAND *
1765      * ONE AND OPERAND TWO DESCRIPTION.   *
1766      *
1767      *          BRANCH TABLE FOR OPERANDS
1768      *
1769      *          0 - N/A
1770      *          1 - S OR D
1771      *          2 - IOP
1772      *          3 - W
1773      *          4 - C
1774      *          5 - DIS
1775      *          6 - BIT
1776      *          7 - N (SHIFT CNT)
1777      *****
1778      0001 FM1 EQU >1          FORMAT 1 - S, D
1779      0002 FM2 EQU >2          FORMAT 2 - DIS
1780      0003 FM3 EQU >3          FORMAT 3 - S, W
1781      0004 FM4 EQU >4          FORMAT 4 - S, C
1782      0005 FM5 EQU >5          FORMAT 5 - W, N
1783      0006 FM6 EQU >6          FORMAT 6 - S
1784      0007 FM7 EQU >7          FORMAT 7 - N/A
1785      0008 FM8 EQU >8          FORMAT 8 - W, IOP
1786      0003 FM9 EQU >3          FORMAT 9 - S, W
1787      000A FMA EQU >A          FORMAT A - BIT
1788      000B FMB EQU >B          FORMAT B - IOP
1789      000C FMC EQU >C          FORMAT C - W
1790      000D FMD EQU >D          FORMAT D - LMF
1791      *
1792      *          COMBINED TEXT AND OPCODE TABLE
1793      *
1794      *          TEXT 'AB '          ;TEXT FOR MNEMONICS
1795      *          DATA >B000+FM1    ;OPCODE AND FORMAT INFORMATION
1796      *
1797      09CE 0000 CMBTAB DATA 0          *** TABLE TERMINATOR ***
1798      *
1799      09D0 4C CODSTT TEXT 'LST '
1800      09D1 53
1801      09D2 54
1802      09D3 20
1803      09D4 008C DATA >0080+FMC      LST
1804      09D5 4C TEXT 'LWP '
1805      09D6 57
1806      09D7 50
1807      09D8 20
1808      09D9 009C DATA >0090+FMC      LWP
1809      09DA 44 DTATXT TEXT 'DATA'
1810      09DB 41
1811      09DC 54
1812      09DD 41
1813      09DE 41
1814      09DF 41
1815      09E0 00A0 DATA >00A0
1816      09E1 44 TEXT 'DIVS'
1817      09E2 49
1818      09E3 56
1819      09E4 53
1820      09E5 0186 DATA >0180+FM6      DIVS
1821      09E6 4D TEXT 'MPYS'

```

	09E9	50			
	09EA	59			
	09EB	53			
1808	09EC	01C6	DATA >01C0+FM6	MPYS	
1809	09EE	4C	TEXT 'LI '		
	09EF	49			
	09F0	20			
	09F1	20			
1810	09F2	0208	DATA >0200+FM8	LI	
1811	09F4	41	TEXT 'AI '		
	09F5	49			
	09F6	20			
	09F7	20			
1812	09F8	0228	DATA >0220+FM8	AI	
1813	09FA	41	TEXT 'ANDI '		
	09FB	4E			
	09FC	44			
	09FD	49			
1814	09FE	0248	DATA >0240+FM8	ANDI	
1815	0A00	4F	TEXT 'ORI '		
	0A01	52			
	0A02	49			
	0A03	20			
1816	0A04	0268	DATA >0260+FM8	ORI	
1817	0A06	43	TEXT 'CI '		
	0A07	49			
	0A08	20			
	0A09	20			
1818	0A0A	0288	DATA >0280+FM8	CI	
1819	0A0C	53	TEXT 'STWP '		
	0A0D	54			
	0A0E	57			
	0A0F	50			
1820	0A10	02AC	DATA >02A0+FMC	STWP	
1821	0A12	53	TEXT 'STST '		
	0A13	54			
	0A14	53			
	0A15	54			
1822	0A16	02CC	DATA >02C0+FMC	STST	
1823	0A18	4C	TEXT 'LWPI '		
	0A19	57			
	0A1A	50			
	0A1B	49			
1824	0A1C	02EB	DATA >02E0+FMB	LWPI	
1825	0A1E	4C	TEXT 'LIMI '		
	0A1F	49			
	0A20	4D			
	0A21	49			
1826	0A22	030B	DATA >0300+FMB	LIMI	
1827	0A24	44	TEXT 'DATA '		
	0A25	41			
	0A26	54			
	0A27	41			
1828	0A28	0320	DATA >0320		
1829	0A2A	49	TEXT 'IDLE '		
	0A2B	44			
	0A2C	4C			
	0A2D	45			
1830	0A2E	0347	DATA >0340+FM7	IDLE	
1831	0A30	52	TEXT 'RSET '		

	0A31	53		
	0A32	45		
	0A33	54		
1832	0A34	0367	DATA >0360+FM7	RSET
1833	0A36	52	TEXT 'RTWP'	
	0A37	54		
	0A38	57		
	0A39	50		
1834	0A3A	0387	DATA >0380+FM7	RTWP
1835	0A3C	43	TEXT 'CKON'	
	0A3D	4B		
	0A3E	4F		
	0A3F	4E		
1836	0A40	03A7	DATA >03A0+FM7	CKON
1837	0A42	43	TEXT 'CKOF'	
	0A43	4B		
	0A44	4F		
	0A45	46		
1838	0A46	03C7	DATA >03C0+FM7	CKOF
1839	0A48	4C	TEXT 'LREX'	
	0A49	52		
	0A4A	45		
	0A4B	5B		
1840	0A4C	03E7	DATA >03E0+FM7	LREX
1841	0A4E	42	TEXT 'BLWP'	
	0A4F	4C		
	0A50	57		
	0A51	50		
1842	0A52	0406	DATA >0400+FM6	BLWP
1843	0A54	42	TEXT 'B'	
	0A55	20		
	0A56	20		
	0A57	20		
1844	0A58	0446	DATA >0440+FM6	B
1845	0A5A	5B	TEXT 'X'	
	0A5B	20		
	0A5C	20		
	0A5D	20		
1846	0A5E	0486	DATA >0480+FM6	X
1847	0A60	43	TEXT 'CLR'	
	0A61	4C		
	0A62	52		
	0A63	20		
1848	0A64	04C6	DATA >04C0+FM6	CLR
1849	0A66	4E	TEXT 'NEG'	
	0A67	45		
	0A68	47		
	0A69	20		
1850	0A6A	0506	DATA >0500+FM6	NEG
1851	0A6C	49	TEXT 'INV'	
	0A6D	4E		
	0A6E	56		
	0A6F	20		
1852	0A70	0546	DATA >0540+FM6	INV
1853	0A72	49	TEXT 'INC'	
	0A73	4E		
	0A74	43		
	0A75	20		
1854	0A76	0586	DATA >0580+FM6	INC
1855	0A78	49	TEXT 'INCT'	

	0A79	4E		
	0A7A	43		
	0A7B	54		
1856	0A7C	05C6	DATA >05C0+FM6	INCT
1857	0A7E	44	TEXT 'DEC '	
	0A7F	45		
	0A80	43		
	0A81	20		
1858	0A82	0606	DATA >0600+FM6	DEC
1859	0A84	44	TEXT 'DECT'	
	0A85	45		
	0A86	43		
	0A87	54		
1860	0A88	0646	DATA >0640+FM6	DECT
1861	0A8A	42	TEXT 'BL '	
	0A8B	4C		
	0A8C	20		
	0A8D	20		
1862	0A8E	0686	DATA >0680+FM6	BL
1863	0A90	53	TEXT 'SWPB'	
	0A91	57		
	0A92	50		
	0A93	42		
1864	0A94	06C6	DATA >06C0+FM6	SWPB
1865	0A96	53	TEXT 'SET0'	
	0A97	45		
	0A98	54		
	0A99	4F		
1866	0A9A	0706	DATA >0700+FM6	SET0
1867	0A9C	44	TEXT 'DATA'	
	0A9D	41		
	0A9E	54		
	0A9F	41		
1868	0AA0	0780	DATA >0780	
1869	0AA2	41	TEXT 'ABS '	
	0AA3	42		
	0AA4	53		
	0AA5	20		
1870	0AA6	0746	DATA >0740+FM6	ABS
1871	0AA8	53	TEXT 'SRA '	
	0AA9	52		
	0AAA	41		
	0AAB	20		
1872	0AAC	0805	DATA >0800+FM5	SRA
1873	0AAE	53	TEXT 'SRL '	
	0AAF	52		
	0AB0	4C		
	0AB1	20		
1874	0AB2	0905	DATA >0900+FM5	SRL
1875	0AB4	53	TEXT 'SLA '	
	0AB5	4C		
	0AB6	41		
	0AB7	20		
1876	0AB8	0A05	DATA >0A00+FM5	SLA
1877	0ABA	53	TEXT 'SRC '	
	0ABB	52		
	0ABC	43		
	0ABD	20		
1878	0ABE	0B05	DATA >0B00+FM5	SRC
1879	0AC0	44	TEXT 'DATA'	

OPB

	OAC1	41			
	OAC2	54			
	OAC3	41			
1880	OAC4	0C00	DATA >0C00	MID OPCODES >0C00 - >0DFF	
1881	OAC6	57	TEXT 'WNBL'		
	OAC7	4E			
	OAC8	42			
	OAC9	4C			
1882	OACA	0E06	DATA >0E00+FM6	WRITE HEX NIBBLE	
1883	OACC	52	TEXT 'RHXW'		
	OACD	48			
	OACE	58			
	OACF	57			
1884	OAD0	0E46	DATA >0E40+FM6	READ HEX WORD	
1885	OAD2	57	TEXT 'WHXW'		
	OAD3	48			
	OAD4	58			
	OAD5	57			
1886	OAD6	0E86	DATA >0E80+FM6	WRITE HEX WORD	
1887	OADB	45	TEXT 'EKO '		
	OAD9	4B			
	OADA	4F			
	OADB	20			
1888	OADC	0EC6	DATA >0EC0+FM6	ECHO CHARACTER	
1889	OADE	57	TEXT 'WRIT'		
	OADF	52			
	OAE0	49			
	OAE1	54			
1890	OAE2	0F06	DATA >0F00+FM6	WRITE BYTE XOP	
1891	OAE4	52	TEXT 'READ'		
	OAE5	45			
	OAE6	41			
	OAE7	44			
1892	OAE8	0F46	DATA >0F40+FM6	READ XOP	
1893	OAEA	4D	TEXT 'MSG '		
	OAE8	53			
	OAEC	47			
	OAED	20			
1894	OAEE	0F86	DATA >0F80+FM6	MESSAGE	
1895	OAF0	42	TEXT 'BKPT'		
	OAF1	4B			
	OAF2	50			
	OAF3	54			
1896	OAF4	0FC6	DATA >0FC0+FM6	BKPT	
1897					
1898	OAF6	4A	TEXT 'JMP '		
	OAF7	4D			
	OAF8	50			
	OAF9	20			
1899	OAFA	1002	DATA >1000+FM2	JMP	
1900	OAFB	4A	TEXT 'JLT '		
	OAFD	4C			
	OAFE	54			
	OAFF	20			
1901	OB00	1102	DATA >1100+FM2	JLT	
1902	OB02	4A	TEXT 'JLE '		
	OB03	4C			
	OB04	45			
	OB05	20			
1903	OB06	1202	DATA >1200+FM2	JLE	

1904	OB08	4A	TEXT 'JEG '	
	OB09	45		
	OB0A	51		
	OB0B	20		
1905	OB0C	1302	DATA >1300+FM2	JEG
1906	OB0E	4A	TEXT 'JHE '	
	OB0F	48		
	OB10	45		
	OB11	20		
1907	OB12	1402	DATA >1400+FM2	JHE
1908	OB14	4A	TEXT 'JGT '	
	OB15	47		
	OB16	54		
	OB17	20		
1909	OB18	1502	DATA >1500+FM2	JGT
1910	OB1A	4A	TEXT 'JNE '	
	OB1B	4E		
	OB1C	45		
	OB1D	20		
1911	OB1E	1602	DATA >1600+FM2	JNE
1912	OB20	4A	TEXT 'JNC '	
	OB21	4E		
	OB22	43		
	OB23	20		
1913	OB24	1702	DATA >1700+FM2	JNC
1914	OB26	4A	TEXT 'JOC '	
	OB27	4F		
	OB28	43		
	OB29	20		
1915	OB2A	1802	DATA >1800+FM2	JOC
1916	OB2C	4A	TEXT 'JND '	
	OB2D	4E		
	OB2E	4F		
	OB2F	20		
1917	OB30	1902	DATA >1900+FM2	JND
1918	OB32	4A	TEXT 'JL '	
	OB33	4C		
	OB34	20		
	OB35	20		
1919	OB36	1A02	DATA >1A00+FM2	JL
1920	OB38	4A	TEXT 'JH '	
	OB39	48		
	OB3A	20		
	OB3B	20		
1921	OB3C	1B02	DATA >1B00+FM2	JH
1922	OB3E	4A	TEXT 'JOP '	
	OB3F	4F		
	OB40	50		
	OB41	20		
1923	OB42	1C02	DATA >1C00+FM2	JOP
1924	OB44	53	TEXT 'SBO '	
	OB45	42		
	OB46	4F		
	OB47	20		
1925	OB48	1D0A	DATA >1D00+FMA	SBO
1926	OB4A	53	TEXT 'SBZ '	
	OB4B	42		
	OB4C	5A		
	OB4D	20		
1927	OB4E	1E0A	DATA >1E00+FMA	SBZ

1928	OB50	54	TEXT 'TB '	
	OB51	42		
	OB52	20		
	OB53	20		
1929	OB54	1F0A	DATA >1F00+FMA	TB
1930	OB56	43	TEXT 'CDC '	
	OB57	4F		
	OB58	43		
	OB59	20		
1931	OB5A	2003	DATA >2000+FM3	CDC
1932	OB5C	43	TEXT 'CZC '	
	OB5D	5A		
	OB5E	43		
	OB5F	20		
1933	OB60	2403	DATA >2400+FM3	CZC
1934	OB62	58	TEXT 'XOR '	
	OB63	4F		
	OB64	52		
	OB65	20		
1935	OB66	2803	DATA >2800+FM3	XOR
1936	OB68	58	TEXT 'XOP '	
	OB69	4F		
	OB6A	50		
	OB6B	20		
1937	OB6C	2C04	DATA >2C00+FM4	XOP
1938	OB6E	4C	TEXT 'LDCR '	
	OB6F	44		
	OB70	43		
	OB71	52		
1939	OB72	3004	DATA >3000+FM4	LDCR
1940	OB74	53	TEXT 'STCR '	
	OB75	54		
	OB76	43		
	OB77	52		
1941	OB78	3404	DATA >3400+FM4	STCR
1942	OB7A	4D	TEXT 'MPY '	
	OB7B	50		
	OB7C	59		
	OB7D	20		
1943	OB7E	3803	DATA >3800+FM9	MPY
1944	OB80	44	TEXT 'DIV '	
	OB81	49		
	OB82	56		
	OB83	20		
1945	OB84	3C03	DATA >3C00+FM9	DIV
1946	OB86	53	TEXT 'SZC '	
	OB87	5A		
	OB88	43		
	OB89	20		
1947	OB8A	4001	DATA >4000+FM1	SZC
1948	OB8C	53	TEXT 'SZCB '	
	OB8D	5A		
	OB8E	43		
	OB8F	42		
1949	OB90	5001	DATA >5000+FM1	SZCB
1950	OB92	53	TEXT 'S '	
	OB93	20		
	OB94	20		
	OB95	20		
1951	OB96	6001	DATA >6000+FM1	S

1952	OB98	53	TEXT 'SB '	
	OB99	42		
	OB9A	20		
	OB9B	20		
1953	OB9C	7001	DATA >7000+FM1	SB
1954	OB9E	43	TEXT 'C '	
	OB9F	20		
	OBA0	20		
	OBA1	20		
1955	OBA2	8001	DATA >8000+FM1	C
1956	OBA4	43	TEXT 'CB '	
	OBA5	42		
	OBA6	20		
	OBA7	20		
1957	OBA8	9001	DATA >9000+FM1	CB
1958	OBA A	41	TEXT 'A '	
	OBA B	20		
	OBA C	20		
	OBA D	20		
1959	OBA E	A001	DATA >A000+FM1	A
1960	OB B0	41	TEXT 'AB '	
	OB B1	42		
	OB B2	20		
	OB B3	20		
1961	OB B4	B001	DATA >B000+FM1	AB
1962	OB B6	4D	TEXT 'MOV '	
	OB B7	4F		
	OB B8	56		
	OB B9	20		
1963	OB B A	C001	DATA >C000+FM1	MOV
1964	OB B C	4D	TEXT 'MOVB '	
	OB B D	4F		
	OB B E	56		
	OB B F	42		
1965	OB C0	D001	DATA >D000+FM1	MOVB
1966	OB C2	53	TEXT 'SOC '	
	OB C3	4F		
	OB C4	43		
	OB C5	20		
1967	OB C6	E001	DATA >E000+FM1	SOC
1968	OB C8	53	TEXT 'SOCB '	
	OB C9	4F		
	OB C A	43		
	OB C B	42		
1969	OB C C	F001	CODEND DATA >F000+FM1	SOCB
1970	OB C E	0000	DATA 0	
1971		*		
1972		*	SPECIAL OPCODE TABLE	
1973		*		
1974		*	TEXT 'xxxx'	MNEMONIC
1975		*	DATA OP#x	SPECIAL FORMAT TYPE
1976		*	DATA >xxxx	UNIQUE OP-CODE
1977		*		
1978	OB D0	53	SOPSTT TEXT 'SPIN'	
	OB D1	50		
	OB D2	49		
	OB D3	4E		
1979	OB D4	0000	DATA 0, >10FF	
1980	OB D8	4E	CONOP TEXT 'NOP '	
	OB D9	4F		

	OBDA	50		
	OBDB	20		
1981	OBDC	0000		DATA 0,>1000
1982	OBEO	52	CORT	TEXT 'RT '
	OBE1	54		
	OBE2	20		
	OBE3	20		
1983	OBE4	0000		DATA 0,>045B
1984	OBEB	0000	SOPEND	DATA 0
1985			*	
1986			*	
1987			*	
1988	OBEA	0003	C3	DATA >0003
1989				END

NO ERRORS, NO WARNINGS

OUTPUT FORMAT CONSTANT

ACCESS NAMES TABLE

SOURCE ACCESS NAME= ADHOC.SRC.JMP
OBJECT ACCESS NAME= ADHOC.OBJ.JMP
LISTING ACCESS NAME= ADHOC.LST.JMP
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

```
0002          IDT 'JMP'
0003          *
0004          * ROUTINE LIST:
0005          *      JMPROA      PICK UP NEXT BYTE FROM PROGRAM, STORE
0006          *                  IT IN THE DLC AND JUMP ON ITS CONTENTS
0007          *      JMPRO      JUMP ON CONTENTS OF R0
0008          *
0009          * MODULE DEFINITIONS:
0010          *
0011          *      DEF JMPROA, JMPRO
0012          *
0013          * EXTERNAL ROUTINES:
0014          *
0015          *
0016          * EXTERNAL DATA:
0017          *
0018          *      REF DLIM
0019          *
0020          * ABSTRACT:
0021          *
0022          * JMPROA IS SIMPLY A ROUTINE TO SAVE MEMORY WHEN
0023          * DECIDING UPON AN ACTION FROM A TABLE OF ALTERNATIVES
0024          * SELECTED BY A BYTE CODE.
0025          *
0026          * CALLING SEQUENCE:
0027          *
0028          *      BL @JMPRO OR BL @JMPROA
0029          * TABLE  BYTE DISP, BYTE
0030          *          BYTE DISP, BYTE
0031          *
0032          *      DATA 0
0033          *
0034          *      IN R8 = PBC
0035          *      OUT DLIM UPDATED (IF JMPROA)
0036          *
0037          * EXIT TO TABLE+DISP ON BYTE FOUND
0038          *      OTHERWISE INSTRUCTION IMMEDIATELY AFTER TABLE
0039          *
```

```

0041      *
0042      * ENTRY POINT FOR JMPROA
0043      *
0044 0000 04C0 JMPROA CLR R0
0045 0002 D038      MOVB *R8+,R0      GET NEXT BYTE FROM PROGRAM
0046 0004 C800      MOV R0,@DLIM      SAVE AND DROP THRU INTO JMPRO
      0006 0000
0047      *
0048      * ENTRY POINT FOR JMPRO
0049      *
0050 0008 C08B JMPRO MOV R11,R2      GET TABLE ADR
0051      *
0052      * NEXT INSTRUCTION INSERTED TO OVERCOME PROBLEM WITH
0053      * ODD ADDRESS GENERATION (WITH TMS9980 AND 81 PROCESSORS)
0054      *
0055 000A 04C1      CLR R1
0056      *
0057 000C D07B JMPRO1 MOVB *R11+,R1      GET DISPLACEMENT, DONE?
0058 000E 1305      JEQ JMPRO2      Y - INSTRUCTION FOLLOWING TABLE
0059 0010 9EC0      CB R0,*R11+      N - CODE FOUND?
0060 0012 16FC      JNE JMPRO1      N - KEEP LOOKING
0061 0014 0871      SRA R1,7      Y - POSITION
0062 0016 A081      A R1,R2      ADD DISPLACEMENT
0063 0018 0452      B *R2      GOTO ROUTINE
0064      *
0065      * BYTE OPCODE NOT FOUND - RETURN TO INSTRUCTION
0066      * IMMEDIATELY FOLLWOING THE TABLE
0067      *
0068 001A 058B JMPRO2 INC R11      MOVE TO NEXT LINE
0069 001C 045B      RT
0070      END

```

NO ERRORS, NO WARNINGS

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.CASSETTE
OBJECT ACCESS NAME= ADHOC.OBJ.CASSETTE
LISTING ACCESS NAME= ADHOC.LST.CASSETTE
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

LINE	KEY	NAME
------	-----	------

0052	A	ADHOC.SRC.IOBITS =>ADHOC.SRC.IOBITS
------	---	--


```

0002      IDT  'CASSETTE'
0003      DEF  MC$LOD, MOTORY, MC$SAV, SAVY, LDPY, YORN
0004      REF  DH$VNT, DH$SLT, PRDY, DMYHDR
0005      REF  B43, CMON$, CMOF$, DH$HCS, DH$END, DH$PLL, ERR3$M
0006      REF  UNIT, C0004, D100, CNTDWN, TYP11$, TYPO$, DELAY$
0007      REF  MODEOK, IDRUN, AUTORN, TYP$N$, CASRDY, PLF, GETC$
0008      REF  MODE, GTLN, IOB, BUS, GETCR$
0009      REF  NEWP, NVD, RUNP, SLT, STRTC, TAPERR, VDT, VNT
0010      REF  TYP$N$, DH$SIZ, DH$ARF, PGMFND, CALLWP
0011      REF  ESCFLG, EFIX$E, IDTEQ, WPR1
0012      *
0013      2FB0  ERROR  EQU  >2FB0
0014      2FA0  ERROR2 EQU  ERROR+>20
0015      *
0016      DXOP  EVFIX, 11
0017      1600  SYN    EQU  >1600      TAPE SYNC CHARACTER
0018      0200  STX    EQU  >0200      TAPE START OF TEXT CHARACTER
0019      0300  ETX    EQU  >0300      TAPE END OF TEXT CHARACTER
0020      *****
0021      *
0022      *      TAPE/EPROM HEADER BLOCKS
0023      *
0024      *      BASIC      MACHINE CODE
0025      *      *****      ***** RUN =>A5A5
0026      *      * AUTO RUN FLAG *      * AUTO RUN FLAG * NORUN=>5A5A
0027      *      *****      *****
0028      *      * 8 BYTE NAME *      * 8 BYTE NAME * NULL FILLED
0029      *      *****      *****
0030      *      * DISP TO SLT *      * >0000 *
0031      *      *****      *****
0032      *      * DISP TO VNT *      * LOAD ADDRESS *
0033      *      *****      *****
0034      *      * NVD - VDT *      * ENTRY POINT * (OFFSET)
0035      *      *****      *****
0036      *      * LOAD LENGTH *      * LOAD LENGTH * (BYTES)
0037      *      *****      *****
0038      *      * CHECKSUM *      * CHECKSUM * HEADER ONLY
0039      *      *****      *****
0040      *
0041      *DMYHDR EQU $
0042      *DH$ARF BSS 2      RUN/NORUN
0043      *DH$PON BSS 8      PROGRAM NAME
0044      *DH$SLT BSS 2      DISP TO SLT
0045      *DH$VNT BSS 2      DISP TO VNT
0046      *DH$SIZ BSS 2      NVD-VDT
0047      *DH$PLL BSS 2      LENGTH
0048      *DH$HCS BSS 2      HEADER CHECKSUM
0049      *HDRSIZ EQU $-DMYHDR  HEADER BLOCK SIZE
0050      *DH$END EQU $      END OF DUMMY HEADER
0051      *****
0052      COPY ADHOC.SRC.IOBITS

```

```

A0002      *
A0003      * THIS MODULE IS INCLUDED IN SOURCE MODULES
A0004      * BY USING A 'COPY' DIRECTIVE.
A0005      *
A0006      *
A0007      *****
A0008      *
A0009      * CRU DEFINITIONS
A0010      *
A0011      *****
A0012      0000 P10 EQU >0000 PARALLEL I/O
A0013      * PARALLEL I/O - READ BITS
A0014      0000 KDS EQU P10+0 KEYBOARD DATA STROBE
A0015      * EQU P10+1 UNUSED
A0016      0002 D1$SIZ EQU P10+2 DS01 SIZE (1=8",0=5.25")
A0017      0003 D1$DEN EQU P10+3 DS01 DENSITY (0=SD,1=DD)
A0018      0004 FDCINT EQU P10+4 FDC INTERRUPT-
A0019      0005 KBDINT EQU P10+5 KBD INTERRUPT-
A0020      0006 VDPINT EQU P10+6 VDP INTERRUPT-
A0021      0007 BUSINT EQU P10+7 BUS INTERRUPT-
A0022      0008 KEYBRD EQU P10+8 KEYBOARD DATA (8 BITS)
A0023      * PARALLEL I/O - WRITE BITS
A0024      0000 CLKLED EQU P10+0 CLOCK LED
A0025      0001 KBDACK EQU P10+1 KEYBOARD ACKNOWLEDGE-
A0026      0002 BUSACK EQU P10+2 BUS INTERRUPT RESET-
A0027      0003 BTENBL EQU P10+3 BUS TIMEOUT FLAG ENABLE
A0028      0004 DRVSIZ EQU P10+4 DRIVE SIZE (1=8",0=5.25")
A0029      0005 ROMON EQU P10+5 0=ROM ON,1=ROM OFF
A0030      0006 BELLON EQU P10+6 BELL ENABLE BIT
A0031      * EQU P10+7 UNUSED
A0032      *
A0033      * EQU >0020 UNUSED
A0034      0040 EIA02 EQU >0040 PRINTER HARDWARE BASE ADDRESS
A0035      * EQU >0060 UNUSED
A0036      * EQU >0080 UNUSED
A0037      * EQU >00A0 UNUSED
A0038      00C0 CASS02 EQU >00C0 CASSETTE HARDWARE BASE ADDRESS
A0039      00E0 DMAC EQU >00E0 DMAC HARDWARE BASE ADDRESS
A0040      *
A0041      * RESERVED OFF-BOARD CRU LOCATIONS
A0042      0200 PRMPOP EQU >200 FROM PROGRAMMER BOARD
A0043      * READ BITS
A0044      * EQU PRMPOP+0
A0045      * EQU PRMPOP+1
A0046      * EQU PRMPOP+2
A0047      * EQU PRMPOP+3
A0048      0204 PRORDY EQU PRMPOP+4
A0049      0205 PGM EQU PRMPOP+5
A0050      0206 PGMFUL EQU PRMPOP+6
A0051      0207 V30 EQU PRMPOP+7
A0052      0208 EDATA EQU PRMPOP+8
A0053      * WRITE BITS
A0054      0200 PRORST EQU PRMPOP+0
A0055      0201 EPTYPE EQU PRMPOP+1
A0056      0204 VCCON EQU PRMPOP+4
A0057      *PGM EQU PRMPOP+5
A0058      0206 PROERR EQU PRMPOP+6
A0059      * EQU PRMPOP+7
A0060      *EDATA EQU PRMPOP+8
A0061      0210 EPADR EQU PRMPOP+16

```

* TEST

```

A0062      *
A0063      *   CENTRONICS PARALLEL PRINTER PORT
A0064      *
A0065      *   BIT           0       1       2       3       4       5       6       7       8       9
A0066      *   READ
A0067      *   WRITE      D7      D6      D5      D4      D3      D2      D1      D0      D5
A0068      *                               (LSB)                               (MSB)
A0069      *
A0070      0400 PPRINT EQU  >400
A0071      *
A0072      0408 PPDS   EQU  PPRINT+8          DATA STROBE  ___+___+___
A0073      *
A0074      *
A0075      0409 PPBUSY EQU  PPRINT+9          BUSY          ___+___+___
A0076      *
A0077      *****
A0078      *
A0079      *   MEMORY MAPPED I/O DEFINITIONS
A0080      *
A0081      *****
A0082      F100 MAPPER EQU  >F100          MEMORY MAPPER LOCATION
A0083      F100 M$REG0 EQU  MAPPER+0      MAPPER REGISTERS 0..15
A0084      F102 M$REG1 EQU  MAPPER+2
A0085      F104 M$REG2 EQU  MAPPER+4
A0086      F106 M$REG3 EQU  MAPPER+6
A0087      F108 M$REG4 EQU  MAPPER+8
A0088      F10A M$REG5 EQU  MAPPER+10
A0089      F10C M$REG6 EQU  MAPPER+12
A0090      F10E M$REG7 EQU  MAPPER+14
A0091      F110 M$REG8 EQU  MAPPER+16
A0092      F112 M$REG9 EQU  MAPPER+18
A0093      F114 M$REGA EQU  MAPPER+20
A0094      F116 M$REGB EQU  MAPPER+22
A0095      F118 M$REGC EQU  MAPPER+24
A0096      F11A M$REGD EQU  MAPPER+26
A0097      F11C M$REGE EQU  MAPPER+28
A0098      F11E M$REGF EQU  MAPPER+30
A0099      *
A0100      F120 VDP   EQU  >F120
A0101      F120 VRAM  EQU  VDP+0          VDP VRAM ACCESS ADDRESS
A0102      F121 VDPREG EQU  VDP+1        VDP REGISTER ACCESS ADDRESS
A0103      *
A0104      *   TEXT / GRAPHICS 1 MODE STORAGE
A0105      0400 NTBA   EQU  >400          NAME TABLE
A0106      0700 CTBA   EQU  >700          COLOUR TABLE
A0107      0800 PGBA   EQU  >800          PATTERN GENERATOR TABLE
A0108      0780 SNTBA  EQU  >780          SPRITE NAME TABLE
A0109      0000 SPGBA  EQU  >000          SPRITE PATTERN GENERATOR TBL.
A0110      *   GRAPHICS 2 MODE STORAGE
A0111      1800 NTBA1  EQU  >1800          NAME TABLE
A0112      2000 CTBA1  EQU  >2000          COLOUR TABLE
A0113      0000 PGTBA1 EQU  >0000          PATTERN GENERATOR TABLE
A0114      1800 SNTBA1 EQU  >1800          SPRITE NAME TABLE
A0115      3800 SPGBA1 EQU  >3800          SPRITE PATTERN GENERATOR TBL.
A0116      *
A0117      F140 FDC    EQU  >F140          TMS9909 FLOPPY DISC CONTROLLER
A0118      *
0053      *
0054      *   WORKSPACE
0055      *   ROUTINE(S)

```

```

0056      *   REGISTER USAGE :
0057      *
0058      *                                     GLOBAL DATA
0059      *                                     ----- LABELS
0060      *   R0      :   SCRATCH
0061      *   R1      :   SCRATCH
0062      *   R2      :   SCRATCH
0063      *   R3      :   UNIT SAVE
0064      *   R4      :   SCRATCH
0065      *   R5      :   PTR TO DMYHDR
0066      *   R6      :
0067      *   R7      :   CHECKSUM
0068      *   R8      :   POINTER TO PROG NAME
0069      *   R9      :   POINTER TO PROG START
0070      *   R10     :   SAVED RETURN
0071      *   R11     :   BL RETURN ADDRESS
0072      *   R12     :   CASSETTE CRUBASE
0073      *   R13     :   RETURN WP
0074      *   R14     :   RETURN PC
0075      *   R15     :   RETURN ST
0076      *
0077      * *****
0078      *
0079      * MONITOR DUMP ROUTINE -- 'D' COMMAND
0080      *
0081      * DUMP RAM IMAGE TO CASSETTE TAPE IN IMAGE FORMAT
0082      *   R0 = START   R1=STOP   R2=ENTRY POINT
0083      *
0084      * *****
0085 0000 408B MC$SAV SZC  R8,R2      WORD ALIGN ENTRY POINT
0086      *
0087      * START ADDRESS > STOP ADDRESS--ERROR
0088      *
0089 0002 8040      C   R0,R1      START <= STOP ?
0090 0004 1202      JLE  ADDR0K    YES
0091 0006 0460      B    @ERR3$M   ERROR EXIT TO MONITOR
0092      *
0093 000A 0204 ADDR0K LI  R4,DH$SLT REF HDR SLT ENTRY
0094 000C 0000
0095 000E 04F4      CLR  *R4+     CLEAR IT (FLAGS AS M/C)
0096 0010 CD00      MOV  R0,*R4+  SAVE LOAD ADDRESS
0097 0012 C240      MOV  R0,R9    SAVE IT FOR SAVO
0098 0014 CD02      MOV  R2,*R4+  SAVE ENTRY POINT
0099 0016 6040      S    R0,R1    GET LENGTH
0100 0018 C501      MOV  R1,*R4   SAVE LENGTH
0101 001A C28B      MOV  R11,R10  SAVE RETURN ADDRESS
0102 001C 0000      DATA TYP$N$,IDTEG  PROMPT FOR IDT
0103 0020 0720      SETO @MODE    STOP GETLIN USING ^E
0104      *
0105 0022 0000
0106 0024 04E0      CLR  @ESCFLG   MAKE SURE HE CAN BOM OUT
0107 0026 0000
0108 0028 06A0      BL    @GTLN    GET A LINE
0109 002A 0000
0110 002C C220      MOV  @IOB,R8   GET ITS START
0111 002E 0000
0112 0030 1016      JMP  SAVO     DO SAVE

```

0107	*			
0108	*			
0109	*		ENTRY POINT FOR BASIC 'SAVE' COMMAND	
0110	*			
0111	0032 06A0	SAVY	BL @MODEOK	CHECK NOT RUNNING
	0034 0000			
0112	0036 0201		LI R1,DH\$SLT	OK, REF THE SLT ENTRY
	0038 000C			
0113	003A 020A		LI R10,PRDY	RETURN TO 'READY*'
	003C 0000			
0114	003E C260		MOV @BUS,R9	GET BUS
	0040 0000			
0115	0042 C460		MOV @SLT,*R1	COPY IN THE SLT
	0044 0000			
0116	0046 6C49		S R9,*R1+	SUBTRACT BUS
0117	0048 C460		MOV @VNT,*R1	COPY IN THE VNT
	004A 0000			
0118	004C 6C49		S R9,*R1+	SUBTRACT BUS
0119	004E C460		MOV @NVD,*R1	COPY IN THE NVD
	0050 0000			
0120	0052 6C60		S @VDT,*R1+	CALCULATE NVD-VDT (VAR NAMES)
	0054 0000			
0121	0056 C460		MOV @NVD,*R1	CALCULATE LOAD LENGTH FROM
	0058 0050			
0122	005A 6449		S R9,*R1	NVD-BUS
0123	005C 0588		INC R8	SKIP THE QUOTE ON "NAME"
0124	*			
0125	*		SET UP DUMMY HEADER	
0126	*			
0127	005E 0201	SAVO	LI R1,DMYHDR	POINT TO DUMMY HEADER
	0060 0000			
0128	0062 C101		MOV R1,R4	SAVE FOR LATER
0129	0064 C141		MOV R1,R5	SAVE FOR LATER
0130	0066 0202		LI R2,IDRUN	SET FOR AUTO-RUN
	0068 0000			
0131	006A 001C	SAV1	DATA TYP\$N\$,AUTORN	AUTO-RUN PROMPT
0132	006E 06A0		BL @YORN	GET REPLY
	0070 026E			
0133	0072 10FB		JMP SAV1	LOOP IF NOT A 'Y' OR 'N'
0134	0074 1001		JMP SAV2	Y, R2 SET UP OK
0135	0076 0542		INV R2	N, SET FOR NO AUTORUN
0136	0078 CC42	SAV2	MOV R2,*R1+	STORE AUTO-RUN FLAG
0137	007A 0202		LI R2,8	MAX OF 8 CHARACTERS
	007C 0008			
0138	007E D018	SAV3	MOVB *R8,R0	GET CHARACTER
0139	0080 1301		JEQ SAV4	0, END OF NAME
0140	0082 0588		INC R8	MOVE TO NEXT CHARACTER
0141	0084 DC40	SAV4	MOVB R0,*R1+	COPY NAME
0142	0086 0602		DEC R2	LIMIT?
0143	0088 16FA		JNE SAV3	N, LOOP
0144	*			
0145	*		NOW WORK OUT THE CHECKSUM	
0146	*			
0147	008A 0201		LI R1,DH\$HCS	REF HEADER CHECKSUM WORD
	008C 0000			
0148	008E 04D1		CLR *R1	CLEAR CHECKSUM
0149	0090 A474	SAV5	A *R4+,*R1	ADD IN HEADER WORD
0150	0092 8044		C R4,R1	DONE
0151	0094 1AFD		JL SAV5	N, LOOP
0152	*			

0153	*	NOW DUMP TO TAPE	
0154	*		
0155	0096 006A'	SAV6 DATA TYP5N\$,CASRDY	CASSETTE READY PROMPT
0156	009A 06A0	BL @YORN	GET RESPONSE
	009C 026E'		
0157	009E 10FB	JMP SAV6	NEITHER
0158	00A0 1001	JMP SAV7	Y
0159	00A2 10F9	JMP SAV6	N, LOOP
0160	*		
0161	00A4 0720	SAV7 SET0 @PLF	FLAG AS SAVING (-VE)
	00A6 0000		
0162	00A8 06A0	BL @SET02	GO SET UP 9902
	00AA 028E'		
0163	00AC 0000	DATA CMON\$	TURN TAPE MOTOR ON
0164	00AE C0E0	MOV @UNIT,R3	SAVE UNIT FLAGS
	00B0 0000		
0165	00B2 C820	MOV @C0004,@UNIT	SET UNIT FOR CASSETTE ONLY
	00B4 0000		
	00B6 00B0'		
0166	00B8 C820	MOV @D200,@CNTDWN	SET FOR 2 SEC. START UP
	00BA 010C'		
	00BC 0000		
0167	00BE 0000	SNDSYN DATA TYP11\$,SYN	SEND SYNC CHARACTER
0168	00C2 C020	MOV @CNTDWN,R0	DONE STARTUP ?
	00C4 00BC'		
0169	00C6 15FB	JGT SNDSYN	LOOP TILL ENOUGH SENT
0170	00C8 00BE'	DATA TYP11\$,STX	SEND STX
0171	00CC 0201	LI R1,DMYHDR	POINT TO HEADER BLOCK
	00CE 0060'		
0172	00D0 D035	DUMP MOVB *R5+,R0	GET BYTE
0173	00D2 0000	DATA TYP0\$	SEND IT
0174	00D4 0285	CI R5,DH\$END	DONE HEADER?
	00D6 0000		
0175	00D8 16FB	JNE DUMP	N, LOOP
0176	*		
0177	* NOW DUMP THE PROGRAM		
0178	*		
0179	00DA C060	MOV @DH\$PLL,R1	GET LENGTH
	00DC 0000		
0180	00DE 04C7	CLR R7	RESET CHECK COUNTER
0181	00E0 D039	DUMP1 MOVB *R9+,R0	GET BYTE
0182	00E2 00D2'	DATA TYP0\$	SEND IT
0183	00E4 0980	SRL R0,8	PUT IN LSB
0184	00E6 A1C0	A R0,R7	ADD IN TO CHECKSUM
0185	00E8 0601	DEC R1	DONE?
0186	00EA 16FA	JNE DUMP1	N, LOOP
0187	*		
0188	00EC 00C8'	DATA TYP11\$,ETX	SEND 'ETX' CHARACTER
0189	00F0 C007	MOV R7,R0	GET CHECK WORD
0190	00F2 00E2'	DATA TYP0\$	SEND MSB
0191	00F4 06C0	SWPB R0	POSITION LSB
0192	00F6 00F2'	DATA TYP0\$	SEND LSB
0193	*		
0194	00F8 C820	MOV @D100,@CNTDWN	SET FOR 1 SECOND DELAY
	00FA 0000		
	00FC 00C4'		
0195	00FE 0000	DATA DELAY\$	AND WAIT
0196	0100 0000	DATA CMOF\$	TURN TAPE OFF
0197	0102 C803	MOV R3,@UNIT	RESTORE UNIT FLAGS
	0104 00B6'		

0198 0106 04E0 CLR @PLF
0108 00A6'
0199 010A 045A B *R10
0200 *
0201 010C 00CB D200 DATA 200

RESET PROGRAM LOAD FLAG

AND EXIT

TWO SECOND WAIT FOR TAPE M/C

```

0203      *
0204      *   WORKSPACE      :
0205      *   ROUTINE(S)    :
0206      *   REGISTER USAGE :
0207      *
0208      *
0209      *   R0      :   SCRATCH
0210      *   R1      :
0211      *   R2      :
0212      *   R3      :
0213      *   R4      :
0214      *   R5      :   POINTER TO DMYHDR
0215      *   R6      :   CHECKSUM
0216      *   R7      :   AUTO-RUN FLAG
0217      *   R8      :   POINTER TO NAME
0218      *   R9      :
0219      *   R10     :
0220      *   R11     :
0221      *   R12     :   CASSETTE CRUBASE
0222      *   R13     :   RETURN WP
0223      *   R14     :   RETURN PC
0224      *   R15     :   RETURN ST
0225      *
0226      *
0227      *
0228      *
0229      *   MONITOR 'L' COMMAND
0230      *
0231      *
0232      *
0233      *
0234      *
0235      *
0236      *
0237      *
0238      *
0239      *
0240      *
0241      *
0242      *
0243      *
0244      *
0245      *
0246      *
0247      *
0248      *
0249      *
0250      *
0251      *
0252      *
0253      *

```

GLOBAL DATA LABELS

change to error XOP

all 000c changed to my routine, which loads character from memory expansion

```

0233 010E C28B MC$LOD MOV R11,R10 SAVE RETURN ADDRESS
0234 0110 0096' DATA TYP SN$, IDTEG 5625 PROMPT FOR IDT
0235 0114 0720 SETO @MODE EFCC STOP GTLN USING ^E
0236 0118 04E0 CLR @ESCFLG 0020 MAKE SURE HE CAN BOM OUT
0237 011C 06A0 BL @GTLN 4EE4 GET THE NAME
0238 0120 04E0 CLR @MODE EFCC RESET MODE
0239 0124 C220 MOV @IOB,R8 POINT TO NAME
0240 0128 1005 JMP LDPYO AND DO LOAD
0241      *
0242      *   ENTRY POINT FOR BASIC 'LOAD' COMMAND
0243      *
0244 012A 020A LDPY LI R10,PRDY 021C EXIT TO '*READY'
0245 012E 06A0 BL @MODEOK >33F6 ; CHECK MODE
0246 0132 0588 INC R8 0007 558A SKIP LEADING QUOTE
0247 0134 0110 LDPYO DATA TYP SN$, CASRDY 'CASSETTE READY' PROMPT
0248 0138 06A0 BL @YORN 187E GET A RESPONSE
0249 013C 10FB JMP LDPYO NEITHER
0250 013E 1001 JMP LDPYOA Y
0251 0140 10F9 JMP LDPYO N
0252 0142 05A0 LDPYOA INC @PLF @ED72 PLF=1 NON DESTRUCTIVE ESC.
0253 0146 06A0 BL @SETO2 189E GO SET UP 9902

```


MID MOTOR ON

```

0148 028E'
0254 014A 00AC' DATA CMON$ 0014 TURN MOTOR ON
0255 *
0256 * LOOK FOR SOME 'SYN' CHARACTERS
0257 *
0258 014C 04C7 NOSYN CLR R7 RESET 'SYN' COUNT
0259 014E 0000 FND SYN DATA GETCR$ 000C GET A CHARACTER
0260 0150 0280 CI R0,STX 0200 'STX' CHARACTER
0152 0200
0261 0154 1305 JEQ SYN$1 1770 Y, CHECK IF THIS IS OK
0262 0156 0280 CI R0,SYN 1600 N, IS IT A 'SYN'
0158 1600
0263 015A 16F8 JNE NOSYN 175C N, RESET COUNTER
0264 015C 0587 INC R7 Y, COUNT IT
0265 015E 10F7 JMP FND SYN LOOK FOR SOME MORE
0266 *
0267 * 'STX' FOUND - IS THIS OK ?
0268 *
0269 0160 0287 SYN$1 CI R7,6 HAVE WE FOUND ENOUGH ?
0162 0006
0270 0164 1AF3 JL NOSYN N, KEEP LOOKING
0271 * Y, IS THE NEXT WORD THE HEADER
0272 0166 014E' DATA GETCR$ 000C GET BYTE
0273 0168 D1C0 MOV B R0,R7 SAVE IT
0274 016A 0987 SRL R7,8 POSITION
0275 016C 0166' DATA GETCR$ 000C GET 2ND BYTE
0276 016E D1C0 MOV B R0,R7 ADD IT IN
0277 0170 0287 CI R7,IDRUN ASAS IS IT A RUNID?
0172 0068'
0278 0174 1305 JEQ SYN$2 1790 Y, FOUND HEADER
0279 0176 0547 INV R7 N, MAYBE NOT AUTO-RUN
0280 0178 0287 CI R7,IDRUN ASAS IS IT OK?
017A 0172
0281 017C 16E7 JNE NOSYN 175C N, NOT THE HEADER BLOCK
0282 017E 0547 INV R7 Y, RESTORE ID WORD
0283 *
0284 * HEADER BLOCK FOUND, R7 CONTAINS THE AUTO-RUN FLAG
0285 *
0286 0180 0205 SYN$2 LI R5,DMYHDR EFA6 POINT TO THE DMYHDR
0182 00CE'
0287 0184 C045 MOV R5,R1 SAVE IT
0288 0186 CD47 MOV R7,*R5+ SAVE THE AUTO RUN FLAG
0289 0188 C085 MOV R5,R2 SAVE FOR LATER
0290 018A 0000 DATA TYP$*,PGMFND 0006 OUTPUT 'FOUND " "
0291 018E C108 MOV R8,R4 SAVE SEARCH NAME START
0292 0190 016C' LOAD1 DATA GETCR$ 000C GET A CHARACTER
0293 0192 DD40 MOV B R0,*R5+ SAVE IT
0294 0194 1304 JEQ LOAD1A 17AE NULL, DON'T PRINT IT THEN
0295 0196 0285 CI R5,DH$SLT EF80 GOT NAME?
0198 0038'
0296 019A 1B01 JH LOAD1A 17AE Y, SKIP
0297 019C 00F6 17AC DATA TYP0$ 0001 OUTPUT CHARACTER
0298 019E 0285 LOAD1A CI R5,DH$END EF8A DONE HEADER ?
01A0 00D6'
0299 01A2 1AF6 JL LOAD1 17A0 N, LOOP
0300 01A4 00EC' DATA TYP11$, ""*256 0009 Y, NOW OUT CLOSING "
0301 *
0302 * IS IT THE ONE WE ARE LOOKING FOR ?
0303 *
0304 01AB 0200 LI R0,8 MAX 8 BYTES

```

```

01AA 0008
0305 01AC 9C94 LOAD1B CB *R4,*R2+ MATCH ? match NAME
0306 01AE 16CE JNE NOSYN 176C N, LOOK FOR NEW HEADER
0307 01B0 DD14 MOVB *R4,*R4+ WAS IT A NULL
0308 01B2 1302 JEQ LOAD1C 175C Y, MATCH FOUND
0309 01B4 0600 DEC R0 DONE ?
0310 01B6 16FA JNE LOAD1B 17BC N, LOOP
0311 *
0312 * NOW CALCULATE THE HEADER CHECKSUM
0313 *
0314 01B8 04C6 LOAD1C CLR R6 RESET CHECKSUM COUNTER
0315 01BA A1B1 LOAD2 A *R1+,R6 ADD IN WORD OF HEADER
0316 01BC 02B1 CI R1,DH#HCS EFB8 DONE?
0317 01C0 1AFC JL LOAD2 17CA N, LOOP
0318 01C2 8446 C R6,*R1 Y, DO THEY MATCH ?
0319 01C4 1621 JNE LDERR 1818 N, ERROR
0320 *
0321 * FIND OUT WHERE TO LOAD THE PROGRAM
0322 *
0323 01C6 C120 LOAD3 MOV @DH#VNT,R4 GET M/C LOAD ADDRESS
0324 01C8 0000 EFB2
0325 01CA C0E0 MOV @DH#PLL,R3 GET LOAD LENGTH
0326 01CC 00DC EFB6
0327 01CE 04C6 CLR R6 RESET CHECKSUM
0328 01D0 C3E0 MOV @DH#SLT,R15 M/C LOAD?
0329 01D2 0198 EFB6
0330 *
0331 *
0332 *
0333 01DE 0190 LOAD4 DATA GETCR$ 000C GET CHARACTER loads m/c program
0334 01E0 DD00 MOVB R0,*R4+ STORE IT
0335 01E2 0980 SRL R0,8 PUT IN LSB
0336 01E4 A1B0 A R0,R6 ADD INTO CHECKSUM
0337 01E6 0603 DEC R3 DONE?
0338 01E8 16FA JNE LOAD4 N, LOOP
0339 *
0340 01EA 01DE DATA GETCR$ 000C Y, GET CHARACTER
0341 01EC 0280 CI R0,ETX 0300 'ETX' CHARACTER ?
0342 01F0 1607 JNE ABORT 1810 N, ABORT
0343 *
0344 * NOW READ IN CHECKSUM
0345 *
0346 01F2 01EA DATA GETCR$ 000C GET MSB CHECKSUM
0347 01F4 C0C0 MOV R0,R3 COPY IT
0348 01F6 01F2 DATA GETCR$ 000C GET LSB CHECKSUM
0349 01F8 0980 SRL R0,8 POSITION IT
0350 01FA A0C0 A R0,R3 ADD IT
0351 01FC 8183 C R3,R6 CHECKSUM OK?
0352 01FE 130A JEQ LOAD9 1824 Y, SET POINTERS ETC
0353 *
0354 * TAPE LOAD ERROR - ERROR & ABORT
0355 *
0356 0200 C3CF ABORT MOV R15,R15 M/C LOAD?

```

0357	0202	1302	JEG	LDERR 1818	Y, JUST ERROR IT
0358	0204	020A	LI	R10, NEWP 01F4	N, DO A NEW
	0206	0000			
0359			*		
0360			*	TAPE LOAD ERROR - ERROR & EXIT	
0361			*		
0362	0208	0100	LDERR	DATA CMDF\$ 0015	TURN MOTOR OFF
0363	020A	0134		DATA TYP\$N\$, TAPERR 0007 5572	OUT TAPE ERROR
0364	020E	04E0	CLR	@PLF E072	RESET PROGRAM LOAD FLAG
	0210	01DC			
0365	0212	045A	B	*R10	EXIT
0366			*		
0367			*	SET UP VECTORS AND EXECUTE IF NEEDED	
0368			*		
0369	0214	0208	LOAD9	DATA CMDF\$ 0015	TURN TAPE OFF
0370	0216	C3CF	MOV	R15, R15	M/C LOAD ?
0371	0218	160C	JNE	LOAD6 1842	N, BASIC — JNE 7842
0372	021A	C3A0	MOV	@DH\$SIZ, R14	Y, SET PC
	021C	0000		EF84	
0373	021E	020D	LI	R13, CALLWP	SET UP WP
	0220	0000		ED46	
0374	0222	04E0	LOAD8	CLR @PLF E072	RESET LOADING FLAG
	0224	0210			
0375	0226	8820	C	@DH\$ARF, @STRTC	AUTO-RUN?
	0228	0000		EF86 5522	
	022A	0000			
0376	022C	1301	JEG	LOAD7 1840	Y, DO AUTO-RUN
0377	022E	045A	B	*R10	N, EXIT
0378	0230	0380	LOAD7	RTWP	DO AUTO-RUN
0379			*		
0380			*	BASIC LOAD	
0381			*	R4 POINTS AFTER END OF PGM	
0382			*		
0383	0232	0202	LOAD4	LI R2, DH\$SLT	REF SLT DISP ENTRY
	0234	01D2			
0384	0236	C060	MOV	@BUS, R1	GET BUS
	0238	01D8		EF04	
0385	023A	A481	A	R1, *R2	MAKE SLT ADDRESS
0386	023C	C832	MOV	*R2+, @SLT E00	GET SLT
	023E	0044			
0387	0240	A481	A	R1, *R2	MAKE VNT ADDRESS
0388	0242	C832	MOV	*R2+, @VNT	GET VNT
	0244	004A		EF8C	
0389	0246	C804	MOV	R4, @VDT	SET VDT TO FOLLOW PGM.
	0248	0054		EF8E	
0390	024A	6832	S	*R2+, @VDT	ADJUST FOR VARIABLE SPACE
	024C	0248		EF8E	
0391	024E	C804	MOV	R4, @NVD	SET NVD
	0250	0058		EF8C	
0392	0252	020E	LI	R14, RUNP	SET RUN ADDRESS
	0254	0000		3EF2	
0393	0256	020D	LI	R13, WPR1	SET WORKSPACE
	0258	0000		F0DC	
0394	025A	02CF	STST	R15	SAVE STATUS
0395	025C	10E2	JMP	LOAD8 1832	

```

0397      *
0398      *
0399      *      BASIC 'MOTOR' STATEMENT
0400      *      FORMAT
0401      *      MOTOR <ARG>
0402      *
0403      *      IF ARG = 0 THEN TURN CASSETTE MOTOR OFF
0404      *      IF ARG <> 0 THEN TURN CASSETTE MOTOR ON
0405      *
0406 025E 2EC1  *870 MOTORY EVFIX R1      GET ARG
0407 0260 C041      MOV R1,R1      OFF ?
0408 0262 1603 1872 JNE MTRON >1874      N, GO TURN IT ON
0409 0264 0214'      DATA CMDF$ 0015      Y, TURN IT OFF
0410 0266 0460 MOTXIT B @EFIX$E      AND BACK TO BASIC
           0268 0000      1AB8
0411 026A 014A' MTRON DATA CMON$ 0014      TURN CASSETTE MOTOR ON
0412 026C 10FC      JMP MOTXIT 71876      AND EXIT
0413      *
0414      *      GET CHAR FROM KEYBOARD AND CHECK FOR Y OR N
0415      *
0416      *      BL @YORN
0417      *      NEITHER Y NOR N RETURN
0418      *      Y RETURN
0419      *      N RETURN
0420      *
0421      026E' YORN EQU $
0422 026E 0000      DATA GETC$      GET CHAR FROM KEYBOARD
0423 0270 10FE      JMP YORN      NO CHAR
0424 0272 1001      JMP YORN2      VALID CHAR
0425 0274 100A      JMP ESCAPE      ESCAPE CHAR
0426      *
0427 0276 019C' YORN2 DATA TYP0$      ECHO THE CHARACTER
0428 0278 0280      CI RO,>5900      Y?
           027A 5900
0429 027C 1304      JEQ YORN3
0430 027E 0280      CI RO,>4E00      N?
           0280 4E00
0431 0282 1602      JNE YORN4
0432 0284 05CB      INCT R11      'N' DOUBLE SKIP
0433 0286 05CB YORN3 INCT R11      'Y' SINGLE SKIP
0434 0288 045B YORN4 B *R11      RETURN
0435 028A 0460 ESCAPE B @PRDY      EXIT
           028C 012C'

0436      *
0437      *      SET UP CASSETTE 9902 FOR TAPE I/O
0438      *
0439 028E 020C 180E SET02 LI R12,2*CASS02      SET UP CRUBASE
           0290 0180
0440 0292 1D1F      SBO 31      RESET 9902
0441 0294 C6DB      MOV *R11,*R11      DELAY
0442 0296 3220      LDCR @B43,B      2 STOP, NO PAR, 8 BITS, /3 CLOCK
           0298 0000      01C8
0443 029A 1E0D      SBZ 13      NO TIMER
0444 029C 3320 8AC LDCR @CBRATE,12      LOAD XMT/REC BAUD RATE
           029E 02A2'      18B2
0445 02A0 045B      RT      EXIT
0446      *
0447 02A2 04D0 CBRATE DATA >04D0      300 BAUD CASSETTE
0448      END
NO ERRORS,      NO WARNINGS

```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.ATNF
OBJECT ACCESS NAME= ADHOC.OBJ.ATNF
LISTING ACCESS NAME= ADHOC.LST.ATNF
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

```
0002          IDT 'ATNF'
0003          *
0004          *          ATNF          ; ARC-TANGENT FUNCTION
0005          *
0006          REF FUNFX          ; FIX ARGUMENT
0007          REF PLYX,PLYXX     ; EVALUATE POLYMONIAL
0008          REF EVSFR$        ; EXIT TO EVALUATOR
0009          REF C8000
0010          REF FPAC          ; FLOATING POINT ACCUMULATOR
0011          REF TEMP          ; TEMP REGISTER
0012          REF DS,DS1        ; DATA STORAGE
0013          DEF ATNF          ; ARCTANGENT FUNCTION
0014          *
0015          DXOP LOADF,0       ; LOAD FPAC
0016          DXOP STORE,1      ; STORE FPAC
0017          DXOP FADD,2        ; ADD TO FPAC
0018          DXOP FSUB,3        ; SUBTRACT FROM FPAC
0019          DXOP FMUL,4        ; MULTIPLY FPAC
0020          DXOP FDIV,5        ; DIVIDE FPAC
0021          DXOP SCALE,6       ; SCALE FPAC
0022          DXOP NORMAL,7      ; NORMALIZE FPAC
0023          DXOP CLEAR,8       ; CLEAR FPAC
0024          DXOP NEGATE,9      ; NEGATE FPAC
0025          DXOP FLOATF,10     ; FLOAT FPAC
```

```

0027      *
0028      *      COMPUTE ARC-TANGENT OF ARGUMENT (R2).
0029      *      RESULT IS IN RADIANS.
0030      *
0031      * CALLING SEQUENCE:
0032      *
0033      *      B @ATNF
0034      *
0035      *      EXIT TO EVSFR$
0036      *
0037      *
0038      * EXCEPTIONS AND CONDITIONS:
0039      *
0040      *      FLOATING POINT ERRORS
0041      *
0042 0000 06A0 ATNF BL @FUNFX FIX
0002 0000
0043 0004 1041 JMP ATNF5 0
0044      *
0045 0006 04C1 ATNFO CLR R1
0046 0008 D052 MOVB *R2,R1
0047 000A 04C4 CLR R4
0048 000C C0A0 MOV @FPAC,R2
000E 0000
0049 0010 60A0 S @ATNC6,R2 <1?
0012 009A'
0050 0014 1107 JLT ATNF1 Y
0051 0016 2C60 STORE @TEMP MOVE TO TEMP
0018 0000
0052 001A 2C20 LOADF @ATNC6 LOAD 1
001C 009A'
0053 001E 2D60 FDIV @TEMP GET INVERSE
0020 0018'
0054 0022 0584 INC R4 SET FLAG
0055      *
0056 0024 2C60 ATNF1 STORE @TEMP MOVE TO TEMP
0026 0020'
0057 0028 2CE0 FSUB @ATNC0 F-CO
002A 008C'
0058 002C C0A0 MOV @FPAC,R2 GET SIGN
002E 000E'
0059 0030 2C20 LOADF @TEMP MOVE TEMP TO FPAC
0032 0026'
0060 0034 C082 MOV R2,R2 >.268...?
0061 0036 110C JLT ATNF2 N
0062 0038 2CA0 FADD @ATNC1 Y
003A 0092'
0063 003C 2C60 STORE @DS SAVE IN DS
003E 0000
0064 0040 2C20 LOADF @TEMP MOVE TEMP TO FPAC
0042 0032'
0065 0044 2D20 FMUL @ATNC1
0046 0092'
0066 0048 2CE0 FSUB @ATNC6 SUBTRACT 1
004A 009A'
0067 004C 2D60 FDIV @DS F=F/DS
004E 003E'
0068      *
0069 0050 2C60 ATNF2 STORE @DS STORE IN DS
0052 004E'

```

```

0070 0054 06A0      BL   @PLYXX      EVALUATE
      0056 0000
0071 0058 0098'     DATA ATNC2
0072 005A 2C60      STORE @DS1      SAVE IN DS1
      005C 0000
0073 005E 06A0      BL   @PLYX      EVALUATE
      0060 0000
0074 0062 00B8'     DATA ATNC3
0075 0064 2D60      FDIV @DS1      DIVIDE BY DS1
      0066 005C'
0076 0068 2D20      FMUL @DS      MULTIPLY BY DS
      006A 0052'
0077 006C C082      MOV  R2,R2      WAS IT >.268...?
0078 006E 1102      JLT  ATNF3      N
0079 0070 2CA0      FADD' @ATNC4    Y, ADD CONSTANT
      0072 00D2'
0080
0081 0074 0604      * ATNF3 DEC  R4
0082 0076 1602      JNE  ATNF4
0083 0078 2CA0      FADD @ATNC5
      007A 00D8'
0084
0085 007C 0B13      * ATNF4 SRC  R3,1      GET SIGN BIT
0086 007E 4820      SZC  @CB000,@FPAC
      0080 0000
      0082 002E'
0087 0084 AB03      A    R3,@FPAC      ADD SIGN
      0086 0082'
0088
0089 0088 0460      * EXIT TO EVALUATOR & RELOAD R2 WITH FPAC
      008A 0000      ATNF5 B    @EVSFR$
0090
0091 008C 4044      * ATNC0 DATA >4044,>9851,>7A7B
0092 0092 411B      ATNC1 DATA >411B,>B67A,>E858
0093 0098 0004      ATNC2 DATA 4
0094 009A 4110      ATNC6 DATA >4110,>0000,>0000
0095 00A0 4225      DATA >4225,>10EB,>4200
0096 00A6 42CF      DATA >42CF,>B153,>9710
0097 00AC 4316      DATA >4316,>CA99,>3433
0098 00B2 42C5      DATA >42C5,>33FE,>142D
0099 00B8 0003      ATNC3 DATA 3
0100 00BA 41C9      DATA >41C9,>8867,>F42A
0101 00C0 427D      DATA >427D,>9444,>406E
0102 00C6 4312      DATA >4312,>AED9,>3E72
0103 00CC 42C5      DATA >42C5,>33FE,>142D
0104 00D2 4086      ATNC4 DATA >4086,>0A91,>C16C
0105 00DB C119      ATNC5 DATA >C119,>21FB,>5444
0106
      END
NO ERRORS,      NO WARNINGS
    
```


ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.VDP
OBJECT ACCESS NAME= ADHOC.OBJ.VDP
LISTING ACCESS NAME= ADHOC.LST.VDP
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

LINE	KEY	NAME
------	-----	------

0023	A	ADHOC.SRC.IOBITS =>ADHOC.SRC.IOBITS
------	---	--

```
0002          IDT 'VDP'
0003          *
0004          * NOTE :
0005          *   ACCESSES TO THE VDP NOT REQUIRING VRAM ACCESS
0006          *   MUST BE AT LEAST 3uS APART, ACCESSES REQUIRING
0007          *   VRAM ACCESS MUST BE AT LEAST 8uS APART.
0008          *
0009          DEF UNPACK, LDOS, PIXON, PIXOFF, COLF
0010          DEF PIXTST, TEXTY, GRAPHY, WAITY, LDOSY
0011          DEF S$SM, MAGY, COLORY, SHAPEY, SPRITY
0012          DEF EFIX$E, SPUTY, SGETY
0013          REF XLDC, VDPWP0, VDPWP1, BITMAP, C2000
0014          REF WAIT$, DELAY$, CNTDWN, CKEX, FPAC3
0015          REF FPAC2, NLINO, FBCOL
0016          REF EVSFR$, SETTXT, SETGRA, SENDAD, LOADER
0017          REF FGM$, VMODE, USERCS, PCHTB
0018          REF FIX, EVARZ
0019          *
0020          DXOP EVFIX, 11
0021          2F80 ERROR EQU >2F80
0022          2FA0 ERROR2 EQU ERROR+>20
0023          COPY ADHOC. SRC. IOBITS
```

```

A0002      *
A0003      * THIS MODULE IS INCLUDED IN SOURCE MODULES
A0004      * BY USING A 'COPY' DIRECTIVE.
A0005      *
A0006      *
A0007      *****
A0008      *
A0009      * CRU DEFINITIONS
A0010      *
A0011      *****
A0012      0000 P10      EQU  >0000      PARALLEL I/O
A0013      * PARALLEL I/O - READ BITS
A0014      0000 KDS      EQU  P10+0      KEYBOARD DATA STROBE
A0015      *          EQU  P10+1      UNUSED
A0016      0002 D1$SIZ  EQU  P10+2      DS01 SIZE (1=8",0=5.25")
A0017      0003 D1$DEN  EQU  P10+3      DS01 DENSITY (0=SD,1=DD)
A0018      0004 FDCINT  EQU  P10+4      FDC INTERRUPT-
A0019      0005 KBDINT  EQU  P10+5      KBD INTERRUPT-
A0020      0006 VDPINT  EQU  P10+6      VDP INTERRUPT-
A0021      0007 BUSINT  EQU  P10+7      BUS INTERRUPT-
A0022      0008 KEYBRD  EQU  P10+8      KEYBOARD DATA (8 BITS)
A0023      * PARALLEL I/O - WRITE BITS
A0024      0000 CLKLED  EQU  P10+0      CLOCK LED
A0025      0001 KBDACK  EQU  P10+1      KEYBOARD ACKNOWLEDGE-
A0026      0002 BUSACK  EQU  P10+2      BUS INTERRUPT RESET-
A0027      0003 BTENBL  EQU  P10+3      BUS TIMEOUT FLAG ENABLE
A0028      0004 DRVSIZ  EQU  P10+4      DRIVE SIZE (1=8",0=5.25")
A0029      0005 ROMON   EQU  P10+5      0=ROM ON,1=ROM OFF
A0030      0006 BELLON  EQU  P10+6      BELL ENABLE BIT
A0031      *          EQU  P10+7      UNUSED
A0032      *
A0033      *          EQU  >0020      UNUSED
A0034      0040 EIA02   EQU  >0040      PRINTER HARDWARE BASE ADDRESS
A0035      *          EQU  >0060      UNUSED
A0036      *          EQU  >0080      UNUSED
A0037      *          EQU  >00A0      UNUSED
A0038      00C0 CASS02  EQU  >00C0      CASSETTE HARDWARE BASE ADDRESS
A0039      00E0 DMAC    EQU  >00E0      DMAC HARDWARE BASE ADDRESS
A0040      *
A0041      * RESERVED OFF-BOARD CRU LOCATIONS
A0042      0200 PRMPOP  EQU  >200      PROM PROGRAMMER BOARD
A0043      * READ BITS
A0044      *          EQU  PRMPOP+0
A0045      *          EQU  PRMPOP+1
A0046      *          EQU  PRMPOP+2
A0047      *          EQU  PRMPOP+3
A0048      0204 PRORDY  EQU  PRMPOP+4
A0049      0205 PGM     EQU  PRMPOP+5
A0050      0206 PGMFUL  EQU  PRMPOP+6      * TEST
A0051      0207 V30     EQU  PRMPOP+7
A0052      0208 EDATA   EQU  PRMPOP+8
A0053      * WRITE BITS
A0054      0200 PRORST  EQU  PRMPOP+0
A0055      0201 EPTYPE  EQU  PRMPOP+1
A0056      0204 VCCON   EQU  PRMPOP+4
A0057      *PGM        EQU  PRMPOP+5
A0058      0206 PRDERR  EQU  PRMPOP+6
A0059      *          EQU  PRMPOP+7
A0060      *EDATA      EQU  PRMPOP+8
A0061      0210 EPADR   EQU  PRMPOP+16

```

```

A0062      *
A0063      *   CENTRONICS PARALLEL PRINTER PORT
A0064      *
A0065      *   BIT      0      1      2      3      4      5      6      7      8      9
A0066      *   READ
A0067      *   WRITE    D7     D6     D5     D4     D3     D2     D1     D0     DS
A0068      *                   (LSB)                                (MSB)
A0069      *
A0070      0400 PPRINT EQU  >400
A0071      *
A0072      0408 PPDS  EQU  PPRINT+8          DATA STROBE  ___|___|___
A0073      *
A0074      *
A0075      0409 PPBUSY EQU  PPRINT+9          BUSY          ___|___|___
A0076      *
A0077      *****
A0078      *
A0079      *   MEMORY MAPPED I/O DEFINITIONS
A0080      *
A0081      *****
A0082      F100 MAPPER EQU  >F100          MEMORY MAPPER LOCATION
A0083      F100 M$REG0 EQU  MAPPER+0       MAPPER REGISTERS 0..15
A0084      F102 M$REG1 EQU  MAPPER+2
A0085      F104 M$REG2 EQU  MAPPER+4
A0086      F106 M$REG3 EQU  MAPPER+6
A0087      F108 M$REG4 EQU  MAPPER+8
A0088      F10A M$REG5 EQU  MAPPER+10
A0089      F10C M$REG6 EQU  MAPPER+12
A0090      F10E M$REG7 EQU  MAPPER+14
A0091      F110 M$REG8 EQU  MAPPER+16
A0092      F112 M$REG9 EQU  MAPPER+18
A0093      F114 M$REGA EQU  MAPPER+20
A0094      F116 M$REGB EQU  MAPPER+22
A0095      F118 M$REGC EQU  MAPPER+24
A0096      F11A M$REGD EQU  MAPPER+26
A0097      F11C M$REGE EQU  MAPPER+28
A0098      F11E M$REGF EQU  MAPPER+30
A0099      *
A0100      F120 VDP    EQU  >F120
A0101      F120 VRAM   EQU  VDP+0          VDP VRAM ACCESS ADDRESS
A0102      F121 VDPREG EQU  VDP+1          VDP REGISTER ACCESS ADDRESS
A0103      *
A0104      *   TEXT / GRAPHICS 1 MODE STORAGE
A0105      0400 NTBA   EQU  >400          NAME TABLE
A0106      0700 CTBA   EQU  >700          COLOUR TABLE
A0107      0800 PGBA   EQU  >800          PATTERN GENERATOR TABLE
A0108      0780 SNTBA  EQU  >780          SPRITE NAME TABLE
A0109      0000 SPGBA  EQU  >000          SPRITE PATTERN GENERATOR TBL.
A0110      *   GRAPHICS 2 MODE STORAGE
A0111      1800 NTBA1  EQU  >1800        NAME TABLE
A0112      2000 CTBA1  EQU  >2000        COLOUR TABLE
A0113      0000 PGTBA1 EQU  >0000        PATTERN GENERATOR TABLE
A0114      1B00 SNTBA1 EQU  >1B00        SPRITE NAME TABLE
A0115      3B00 SPGBA1 EQU  >3B00        SPRITE PATTERN GENERATOR TBL.
A0116      *
A0117      F140 FDC     EQU  >F140        TMS9909 FLOPPY DISC CONTROLLER
A0118      *

```

```

0025      *
0026      *                SGET STATEMENT
0027      *                =====
0028      *
0029      *      FORMAT :-
0030      *      SGET  ARG1 , ARG 2
0031      *
0032      *      TEXT MODE :-
0033      *      THE VARIABLE SPECIFIED BY ARG 2 IS GIVEN THE ASCII
0034      *      VALUE OF THE CHARACTER AT THE LINEAR CURSOR ADDRESS
0035      *      SPECIFIED BY ARG1.
0036      *
0037      *      GRAPHIC MODE :-
0038      *      THE SHAPE SPECIFIED BY ARG 2 IS SET TO THE PATTERN
0039      *      AT THE LINEAR CURSOR ADDRESS SPECIFIED BY ARG1.
0040      *
0041 0000 2EC1  SGETY  EVFIX R1          GET ARG 1
0042 0002 C020      MOV  @VMODE,R0      TEXT MODE?
0043 0004 0000
0043 0006 1612      JNE  GGET          N, DO GRAPHIC 'SGET'
0044      *
0045 0008 02B1      CI   R1,40*24      VALID CURSOR ADDRESS ?
0046 000A 03C0
0046 000C 146E      JHE  ERR15A        N, ERROR
0047 000E 0420      BLWP @EVARZ        GET VARIABLE ADDRESS IN R2
0048 0010 0000
0048 0012 C3CB      MOV  R8,R15        SAVE PBC
0049 0014 C201      MOV  R1,R8         GET CURSOR ADDRESS IN R8
0050 0016 022B      AI   R8,NTBA       ADD IN NAME TABLE START
0051 0018 0400
0051 001A 06A0      BL   @SENDAD       POINT VDP AT IT
0052 001C 0000
0052 001E 04F2      CLR  *R2+         CLEAR 1ST WORD OF VAR
0053 0020 04F2      CLR  *R2+         CLEAR 2ND WORD OF VAR
0054 0022 04D2      CLR  *R2          CLEAR 3RD WORD OF VAR
0055 0024 DBA0      MOVB @VRAM,@-1(R2) READ CHARACTER CODE INTO VARIABLE
0056 0026 F120
0056 0028 FFFF
0056 002A 105D      JMP  PUTXIT        EXIT
0057      *
0058 002C 02B1  GGET  CI   R1,32*24    VALID CURSOR ADDRESS ?
0059 002E 0300
0059 0030 145C      JHE  ERR15A        N, ERROR
0060 0032 2EC2      EVFIX R2          Y, GET ARG 2
0061 0034 C3CB      MOV  R8,R15        SAVE PBC
0062 0036 C201      MOV  R1,R8         SET UP R8
0063 0038 0A3B      SLA  R8,3          FORM TABLE INDEX
0064      ASMIF PGTBA1
0065      AI   R8,PGTBA1              ADD IN PGT BASE IF NEEDED
0066      ASMEND
0067 003A 06A0      BL   @SENDAD
0068 003C 001C
0068 003E 0203      LI   R3,BITMAP     REF BITMAP
0069 0040 0000
0069 0042 C103      MOV  R3,R4         SAVE FOR LATER
0070 0044 0700      SETO R0           DO 8 BYTES (SLOWLY)
0071 0046 DCE0  GREAD MOVB @VRAM,*R3+   READ BYTE
0072 0048 F120
0072 004A C618      MOV  *R8,*R8      <<< SLOW THE BLODDY THING DOWN >>>
0073 004C 0A20      SLA  R0,2          DONE?

```

0074	004E	16FB		JNE	GREAD	N, LOOP
0075			*			
0076	0050	C202		MOV	R2,R8	GET SHAPE £
0077	0052	0A38		SLA	R8,3	GET TABLE INDEX
0078	0054	0228		AI	R8,SPGBA1+>4000	ADD IN TABLE BASE
	0056	7800				
0079	0058	06A0		BL	@SENDAD	POINT VDP AT IT
	005A	003C				
0080	005C	0700		SETD	R0	DO 8 BYTES (SLOWLY)
0081	005E	D834	GWRIT	MOVB	*R4+,@VRAM	WRITE BYTE
	0060	F120				
0082	0062	C618		MOV	*R8,*R8	<<< SLOW THE BLODDY THING DOWN >>>
0083	0064	0A20		SLA	R0,2	DONE?
0084	0066	16FB		JNE	GWRIT	N, LOOP
0085	0068	103E		JMP	PUTXIT	Y, EXIT

```

0087      *
0088      *           SPUT STATEMENT
0089      *           =====
0090      *
0091      *           FORMAT :-
0092      *           SPUT  ARG1 , ARG 2
0093      *
0094      *           TEXT MODE :-
0095      *           THE CHARACTER AT THE LINEAR CURSOR ADDRESS
0096      *           SPECIFIED BY ARG1 IS SET TO THE ASCII CHARACTER
0097      *           GIVEN BY ARG2.
0098      *
0099      *           GRAPHIC MODE :-
0100      *           THE CHARACTER AT THE LINEAR CURSOR ADDRESS
0101      *           SPECIFIED BY ARG1 IS SET TO THE SHAPE SPECIFIED
0102      *           BY ARG2.
0103      *
0104 006A 2EC1 SPUTY EVFIX R1          GET ARG 1
0105 006C 2EC2      EVFIX R2          GET ARG 2
0106 006E D082      MOV B R2,R2       ARG 2 VALID ? 0..255
0107 0070 163C      JNE  ERR15A       N, ERROR IT
0108 0072 C3C8      MOV  R8,R15       SAVE PBC
0109 0074 C020      MOV  @VMODE,R0    TEXT MODE ?
0110      0076 0004'
0110 0078 160C      JNE  GPUT          N, DO GRAPHICS 'SPUT'
0111      *
0112 007A 0281      CI   R1,24*40     VALID CURSOR ADDRESS ?
0113      007C 03C0
0113 007E 1435      JHE  ERR15A       N, ERROR
0114 0080 C201      MOV  R1,R8        Y, SET UP R8
0115 0082 0228      AI   R8,NTBA+>4000 ADD TABLE START
0116      0084 4400
0116 0086 06A0      BL   @SENDAD      GIVE IT TO VDP
0117      0088 005A'
0117 008A 06C2      SWPB R2           POSITION CHARACTER
0118 008C D802      MOV B R2,@VRAM    WRITE IT TO VRAM
0119      008E F120
0119 0090 102A      JMP  PUTXIT       EXIT
0120      *
0121 0092 0281 SPUT CI   R1,24*32     VALID CURSOR ADDRESS ?
0122      0094 0300
0122 0096 1429      JHE  ERR15A       N, ERROR
0123 0098 C202      MOV  R2,R8        GET SHAPE £
0124 009A 0A38      SLA  R8,3         FORM INDEX INTO SPRITE PAT. GEN TBL
0125 009C 0228      AI   R8,SPGBA1    ADD IN TABLE START
0126      009E 3800
0126 00A0 06A0      BL   @SENDAD      POINT VDP AT IT
0127      00A2 0088'
0127 00A4 0208      LI   R8,8         READ 8 BYTES
0128      00A6 0008
0128 00AB C0C8      MOV  R8,R3        SAVE FOR LATER
0129 00AA 0200      LI   R0,BITMAP    INTO 'BITMAP'
0130      00AC 0040'
0130 00AE C100      MOV  R0,R4        SAVE FOR LATER
0131 00B0 DC20 RSHAPE MOV B @VRAM,*R0+ READ SHAPE BYTE
0132      00B2 F120
0132 00B4 C618      MOV  *R8,*R8     <<< SLOW THE BLODDY THING DOWN >>>
0133 00B6 0608      DEC  R8          DONE?
0134 00B8 16FB      JNE  RSHAPE      N, LOOP
0135      *

```

```

0136 00BA C201      MOV R1,R8      GET CURSOR ADDRESS IN R8
0137 00BC 0A38      SLA R8,3      FORM INDEX INTO PGT
0138 00BE 0228      AI R8,PGTBA1+>4000 ADD IN TABLE START
00C0 4000
0139 00C2 06A0      BL @SENDAD     GIVE IT TO VDP
00C4 00A2'
0140 00C6 C618      WSHAPE MOV *R8,*R8    <<< SLOW THE BLODDY THING DOWN >>>
0141 00C8 DB34      MOV *R4+,@VRAM  WRITE SHAPE TO VRAM
00CA F120
0142 00CC 0603      DEC R3      DONE ?
0143 00CE 16FB      JNE WSHAPE     N, LOOP
0144 *
0145 00D0 0228      AI R8,(CTBA1-PGTBA1) POINT TO COLOUR TABLE ENTRY
00D2 2000
0146 00D4 06A0      BL @SENDAD     POINT VDP AT IT
00D6 00C4'
0147 00DB 0705      SETO R5      8 LOOPS THE HARD WAY
0148 00DA DB20      WCOLOR MOV *R4+,@VRAM WRITE FCOL/BCOL TO IT
00DC 0000
00DE F120
0149 00E0 C618      MOV *R8,*R8    <<< SLOW THE BLODDY THING DOWN >>>
0150 00E2 0A25      SLA R5,2      DONE ? (SLOWLY !!)
0151 00E4 16FA      JNE WCOLOR     N, LOOP
0152 00E6 C20F      PUTXIT MOV R15,R8    RESTORE PBC
0153 00E8 101E      JMP FNLINO     AND EXIT
0154 *
0155 00EA 1065      ERR15A JMP ERR15    "DISP TOO BIG"

```



```

0199      *
0200      *          CHAR STATEMENT
0201      *          =====
0202      *
0203      *          FORMAT :
0204      *          CHAR EXP1, EXP2, EXP3, EXP4
0205      *          OR
0206      *          CHAR
0207      *
0208      *          THIS STATEMENT RE-DEFINES A CHARACTER.
0209      *          EXP1 - CHARACTER NUMBER    0..255
0210      *          EXP2 - FIRST 16 BITS OF DEFINITION
0211      *          EXP3 - SECOND 16 BITS OF DEFINITION
0212      *          EXP4 - THIRD 16 BITS OF DEFINITION
0213      *
0214      *          IF THE STATEMENT IS ENTERED WITHOUT PARAMETERS
0215      *          THEN THE CHARACTERS 0..127 ARE RE-LOADED WITH
0216      *          THEIR INITIAL PATTERNS.
0217      *
0218      00F6 06A0  LDCSY  BL  @CKEX          EXPRESSIONS ?
          00FB 0000
0219      00FA 1017      JMP  RELOAD          N, RELOAD CHARACTERS 0..127
0220      00FC 2EC5      EVFIX R5          Y, GET CHARACTER &
0221      00FE 0285      CI   R5,255        VALID CHARACTER ?
          0100 00FF
0222      0102 1B5B      JH   ERR35        N, ERROR IT
0223      0104 0204      LI   R4,3         DO 3 PARAMETERS
          0106 0003
0224      0108 02A6      STWP R6
0225      010A 05C6      INCT R6          AND SAVE IN R1-R3
0226      *
0227      010C 0280      GETBIT CI  R0,>3F00  ', ' ?
          010E 3F00
0228      0110 1656      JNE  ERR37        N, ERROR IT
0229      0112 2EF6      EVFIX *R6+      GET BIT PATTERN
0230      0114 0604      DEC  R4          COUNT IT
0231      0116 16FA      JNE  GETBIT
0232      *
0233      0118 3960      MPY  @C0006,R5    FORM INDEX INTO TABLE
          011A 02EE'
0234      011C 0226      AI   R6,PCHTB    ADD IN TABLE START
          011E 0000
0235      0120 CD81      MOV  R1,*R6+      UPDATE TABLE ENTRY
0236      0122 CD82      MOV  R2,*R6+
0237      0124 CD83      MOV  R3,*R6+
0238      0126' EFIX$E EQU  $          <<< EXIT FOR AN EVFIX ROUTINE >>>
0239      0126 0608      FNLI NO DEC  R8    BACKUP TO DELIM
0240      0128 10E2      JMP  TGEXIT      EXIT
0241      *
0242      *          RELOAD CHARACTERS 0..USRC5
0243      *
0244      012A' RELOAD EQU  $
0245      ASMIF PIO
0246      LI   R12,2*PIO
0247      ASMELS
0248      CLR  R12
0249      ASMEND
0250      012C 0201      LI   R1,PCHTB    POINT TO START OF C. SET
          012E 011E'
0251      *

```

0252	0130	0300	COPY	LIMI 0	CLOSE INTERRUPT WINDOW
	0132	0000			
0253	0134	1E05	SBZ	ROMON-PIO	TURN ON ROM
0254	0136	CC51	MOV	*R1,*R1+	COPY IT
0255	0138	1D05	SBO	ROMON-PIO	TURN ROM OFF
0256	013A	0300	LIMI	15	OPEN WINDOW FOR INTERRUPTS
	013C	000F			
0257	013E	0281	CI	R1,USERCS	UPTO USER CHARACTER SET ?
	0140	0000			
0258	0142	1AF6	JL	COPY	N, CONTINUE COPY
0259	0144	10D4	JMP	TGEXIT	EXIT

```

0261      *
0262      *          COLOUR STATEMENT
0263      *          =====
0264      *
0265      *          FORMAT :-
0266      *          COLOUR ARG1<, ARG 2 >
0267      *
0268      *          THE FOREGROUND COLOUR IS SET TO ARG1,
0269      *          THE BACKGROUND COLOUR IS SET TO ARG2 IF IT IS
0270      *          PRESENT, IF NOT THE CURRENT BACKGROUND COLOUR
0271      *          IS USED. IF THE STATEMENT IS ISSUED WITHOUT
0272      *          PARAMETERS A DEFAULT OF 4,7 IS USED
0273      *
0274 0146 06A0  COLORY BL  @CKEX          EXPRESSION?
      0148 00F8'
0275 014A 1017          JMP  SET47          N, DEFAULT TO 4,7
0276 014C D060          MOVB @FBCOL,R1      GET CURRENT COLOURS
      014E 00DC'
0277 0150 06C1          SWPB R1             PUT IT IN LSB
0278 0152 2EC2          EVFIX R2           GET FOREGROUND COLOUR
0279 0154 0280          CI  R0,>3F00       ',' FOLLOWING ?
      0156 3F00
0280 0158 1601          JNE  COLR1          N, USE DEFAULT BGND. COLOUR
0281 015A 2EC1          EVFIX R1           Y, GET BACKGROUND
0282 015C 0B41  COLR1  SRC  R1,4          MS NIBBLE = BACKGROUND
0283 015E D081          MOVB R1,R2         PUT IT IN R2
0284 0160 0B42          SRC  R2,4         POSITION FOREGROUND
0285 0162 D802  COLR2  MOVB R2,@FBCOL      R2 MSB = FCOL/BCOL
      0164 014E'
0286 0166 C060          MOV  @VMODE,R1      WHAT MODE AM I IN ?
      0168 0076'
0287 016A 16DD          JNE  FNLINO        GRAPHIC, ALL DONE THEN
0288 016C D802          MOVB R2,@SFBCLR    TEXT, PUT IN CALL BLOCK
      016E 0174'
0289 0170 06A0          BL  @LOADER        CALL VDP LOADER
      0172 0000
0290 0174 00          SFBCLR BYTE >00,>B0+R7  LOAD R7 WITH FCOL/BCOL
0291 0176 0000          DATA 0
0292 0178 10D6          JMP  FNLINO        EXIT
0293      *
0294 017A 0202  SET47  LI  R2,>4700        SET COLOUR 4,7
      017C 4700
0295 017E 0588          INC  R8            ADJUST R8
0296 0180 10F0          JMP  COLR2        & EXIT

```

```

0298      *
0299      *          COL FUNCTION
0300      *          =====
0301      *
0302      *          FORMAT :-
0303      *          A = COL [ ARG 1 , ARG 2 ]
0304      *
0305      *          COL RETURNS THE COLOUR OF THE SPECIFIED PIXEL.
0306      *          TESTING AN OFF SCREEN PIXEL RETURNS -1.
0307      *          ARG 1 FORMS THE HORIZONTAL ORDINATE
0308      *          ARG 2 FORMS THE VERTICAL ORDINATE
0309      *
0310      *          R1 CONTAINS ARG 2, *R2 CONTAINS ARG 1
0311      *          R3 IS SPARE
0312      *          EXIT IS OT EVSFR$
0313      *
0314 0182 0000  COLF  DATA FGM$          FORCE TO GRAPHIC
0315 0184 C0E0      MOV  @XLOC,R3          SAVE CURRENT X,Y
0316      *          0186 0000
0317 0188 C281      MOV  R1,R10          COPY
0318 018A 06A0      BL   @FIX            FIX
0319      *          018C 0000
0320 018E 028A      CI   R10,192        'Y' VALID ?
0321      *          0190 00C0
0322 0192 140E      JHE  BADXY          N, RETURN INVALID COLOUR
0323 0194 0281      CI   R1,256        'X' VALID
0324      *          0196 0100
0325 0198 140B      JHE  BADXY          N, RETURN INVALID COLOUR
0326 019A 06C1      SWPB R1            PUT 'X' IN MSB
0327 019C D281      MOVB R1,R10        R1 MSB = X , LSB = Y
0328 019E C80A      MOV  R10,@XLOC     SET CURSOR LOCATION
0329      *          01A0 0186'
0330 01A2 0420      BLWP @PIXTST        GET PIXEL COLOUR IN FPAC3
0331      *          01A4 030C'
0332 01A6 0000      DATA FPAC3
0333 01A8 C803      MOV  R3,@XLOC        RESTORE CURSOR
0334      *          01AA 01A0'
0335 01AC 0460  COLXIT B  @EVSFR$        EXIT
0336 01AE 0000
0337 01B0 0720  BADXY  SETO @FPAC2      RETURN COLOUR = -1
0338 01B2 0000
0339 01B4 10FB      JMP  COLXIT
0340
0341      *
0342 01B6 2F8F  ERR15  DATA ERROR+15    ILLEGAL SCREEN COMMAND
0343 01B8 2F9E  ERR30  DATA ERROR+30    RANGE ERROR
0344 01BA 2FA0  ERR35  DATA ERROR2,35   PARAMETER ERROR
0345 01BE 2FA0  ERR37  DATA ERROR2,37   ILLEGAL DELIMETER
0346 01C2 2FA0  ERR48  DATA ERROR2,48   WRONG MODE !

```

```

0338 *****
0339 *
0340 *          WAIT STATEMENT
0341 *
0342 *    FORMAT :-
0343 *        WAIT <ARG1>
0344 *
0345 *
0346 *    IF ARG 1 IS PRESENT THEN IT IS USED AS A DELAY
0347 *        COUNT AND THE ROUTINE WAITS FOR ARG 1 *10 MS
0348 *        BEFORE RETURNING TO THE USER.
0349 *    IF ARG 1 IS NOT PRESENT THEN THE ROUTINE WAITS FOR
0350 *        OUTPUT COMPLETION.
0351 *
0352 *****
0353 01C6 06A0 WAITY BL @CKEX          EXPRESSION ?
      01C8 014B'
0354 01CA 1004      JMP WAITO          N, WAIT OUTPUT COMPLETION
0355 01CC 2EE0      EVFIX @CNTDWN      Y, SET DELAY COUNT
      01CE 0000
0356 01D0 0000      DATA DELAY$      WAIT FOR IT
0357 01D2 10A9      JMP FNLINO        THEN EXIT
0358 *
0359 01D4 0000 WAITO DATA WAIT$      WAIT OUTPUT COMPLETION
0360 01D6 10BB      JMP TGEXIT        EXIT

```

```

0362      *
0363      *           SHAPE STATEMENT
0364      *           =====
0365      *
0366      *           FORMAT :-
0367      *           SHAPE SHAPE No. , WORD1, WORD2, WORD3, WORD4
0368      *
0369      *           THE SPRITE PATTERN REFERENCED BY THE SHAPE £
0370      *           IS SET TO THE BIT PATTERN AS DEFINED BY THE
0371      *           ARGUMENTS WORD1 TO WORD4.
0372      *
0373 01D8 0182' SHAPEY DATA FGM$           FORCE GRAPHIC MODE
0374 01DA 2EC1          EVFIX R1           N, GET SHAPE £
0375 01DC D041          MOV B R1,R1        VALID?
0376 01DE 16EB          JNE ERR15          N, ERROR IT
0377 01E0 0A31          SLA R1,3           R1=8* SHAPE£
0378 01E2 0221          AI R1,SPGBA1+>4000 POINT TO ITS STORAGE IN VRAM
0379      *           01E4 7800
0379 01E6 C088          MOV R8,R2          SAVE PBC
0380 01EB C201          MOV R1,R8          SET R8 FOR SENDAD
0381 01EA 06A0          BL @SENDAD         SEND ADDRESS TO VDP
0382      *           01EC 00D6'
0382 01EE C202          MOV R2,R8          RESTORE PBC
0383 01F0 0203          LI R3,4            DO 4 WORDS
0384      *           01F2 0004
0384      *
0385 01F4 0280          GETPAT CI R0,>3F00  ', ' ?
0386      *           01F6 3F00
0386 01F8 16E2          JNE ERR37          N, ERROR
0387 01FA 2EC1          EVFIX R1          Y, GET NEXT WORD
0388 01FC D801          MOV B R1,@VRAM     WRITE MSB
0389      *           01FE F120
0389 0200 06C1          SWPB R1           POSITION LSB
0390 0202 D801          MOV B R1,@VRAM     SEND IT
0391      *           0204 F120
0391 0206 0603          DEC R3            DONE?
0392 0208 16F5          JNE GETPAT         N, LOOP
0393 020A 108D          FNLIN JMP FNLINO   Y, EXIT TO NLINO

```

```

0395      *
0396      *          SPRITE STATEMENT
0397      *          =====
0398      *
0399      *          FORMAT :-
0400      *          SPRITE PLANE, X, Y<, SHAPE£, COLOUR>
0401      *
0402      *          THE SPRITE ON THE SPECIFIED PLANE IS POSITIONED
0403      *          AT X,Y. IF NO OTHER ARGUMENTS FOLLOW IT IS ASSUMED
0404      *          THAT THE SPRITE CURRENTLY ON THAT PLANE IS TO BE
0405      *          MOVED. IF OTHER ARGUMENTS ARE PRESENT THESE SPECIFY
0406      *          THE NEW SHAPE £ AND ITS COLOUR.
0407      *
0408 020C 01DB' SPRITY DATA FGM$          FORCE GRAPHIC MODE
0409 020E 2ECF          EVFIX R15          GET PLANE £
0410 0210 C388          MOV  R8,R14        SAVE PBC
0411 0212 028F          CI   R15,31        VALID ?
0412      0214 001F
0413 0216 1BCF          JH   ERR15         N, ERROR IT
0414 0218 0A2F          SLA  R15,2         R15=4* PLANE£
0415 021A 022F          AI   R15,SNTBA1+2  REF ENTRY (LAST 2 BYTES ONLY)
0416      021C 1B02
0417 021E C20F          MOV  R15,R8        SET R8 FOR SENDAD
0418 0220 06A0          BL   @SENDAD       SEND ADDRESS TO VDP
0419      0222 01EC'
0420 0224 C618          MOV  *R8,*R8      <<< SLOW THE BLODDY THING DOWN >>>
0421 0226 D0A0          MOVVB @VRAM,R2    GET SPRITE £
0422      0228 F120
0423 022A 06C2          SWPB R2
0424 022C 0228          AI   R8,>4000-2   REF START OF ENTRY (WRITE)
0425      022E 3FFE
0426 0230 D0A0          MOVVB @VRAM,R2    GET COLOUR
0427      0232 F120
0428      * R2 = COLOUR,SPRITE£
0429 0234 C618          MOV  *R8,*R8      <<< SLOW THE BLODDY THING DOWN >>>
0430 0236 06A0          BL   @SENDAD       POINT VDP AT IT
0431      0238 0222'
0432 023A C20E          MOV  R14,R8        RESTORE PBC
0433 023C 2EC1          EVFIX R1          GET X
0434 023E 2EC3          EVFIX R3          GET Y
0435 0240 06C3          SWPB R3          POSITION Y
0436 0242 D043          MOVVB R3,R1       FORM R1=Y,X
0437 0244 0280          CI   R0,>3F00     ', ' FOLLOWING ?
0438      0246 3F00
0439 0248 1604          JNE  PUTSAT        N, UPDATE ENTRY
0440      *
0441 024A 2EC2          EVFIX R2          Y, GET SPRITE £
0442 024C 2EC3          EVFIX R3          GET COLOUR
0443 024E 06C3          SWPB R3          N, POSITION IT
0444 0250 D083          MOVVB R3,R2       FORM R2=COLOUR,SPRITE£
0445 0252 06C2          PUTSAT SWPB R2    POSITION COLOUR & SPRITE £
0446      *
0447      *          +-----+
0448      *          |          Y          |      MSB
0449      *          +-----+
0450      *          |          X          |      LSB
0451      *          +-----+
0452      *          |  SPRITE £  |      MSB
0453      *          +-----+
0454      *          |  COLOUR   |      LSB
0455      *          +-----+

```



```
0447      *      +-----+
0448      *
0449 0254 02A3      STWP R3      GET WP
0450 0256 05C3      INCT R3      POINT TO R1
0451 0258 0204      LI R4,4      DO 4 BYTES
      025A 0004
0452      *
0453 025C D833 WRTIT MOVW *R3+,@VRAM      WRITE IT BACK TO VRAM
      025E F120
0454 0260 C618      MOV *R8,*R8      <<< SLOW THE BLODDY THING DOWN >>>
0455 0262 0604      DEC R4      DONE ?
0456 0264 16FB      JNE WRTIT      N, LOOP
0457 0266 10D1      JMP FNLIN      Y, EXIT
```

```

0459      *
0460      *           MAG STATEMENT
0461      *           =====
0462      *
0463      *           FORMAT :-
0464      *           MAG    SPRITE MAG, SPRITE SIZE
0465      *
0466 0268 020C' MAGY    DATA FGM$           FORCE GRAPHIC MODE
0467 026A 2EC2          EVFIX R2           GET SPRITE MAG
0468 026C 2EC1          EVFIX R1           GET SPRITE SIZE
0469 026E 0203          LI    R3,>00C0      DEFAULT TO SIZE=0, MAG=0
           0270 00C0
0470 0272 C041          MOV  R1,R1           BIG SPRITES ?
0471 0274 1301          JEQ  MAGY1          N, LEAVE R3
0472 0276 05C3          INCT R3             Y, SET BIT
0473 0278 C082 MAGY1    MOV  R2,R2           MAG THEM ?
0474 027A 1301          JEQ  MAGY2          N, LEAVE R3
0475 027C 0583          INC  R3             Y, SET BIT
0476 027E 06C3 MAGY2    SWPB R3             POSITION BYTE
0477 0280 D803          MOVB R3,@S$SM       SET UP LOADER TABLE
           0282 0288'
0478 0284 06A0          BL   @LOADER         RE-ENABLE DISPLAY
           0286 0172'
0479 0288 C0    S$SM    BYTE >C0,>80+R1     RELOAD VDP R1
0480 028A 0000          DATA 0
0481 028C 10BE          JMP  FNLIN          EXIT

```

0483 * THESE ROUTINES UNPACK ENTRIES FROM THE
0484 * CHARACTER GENERATOR TABLE AND EITHER LOAD THEM
0485 * INTO THE VDP OR SAVE THEM IN A 8 BYTE STORAGE
0486 * AREA CALLED 'BITMAP'. LDOS ALSO SETS UP THE PNT
0487 *

0488 * CALLING SEQUENCE:
0489 * BLWP @UNPACK
0490 *

0491 * IN : R0 MSB=CHARACTER CODE TO BE UNPACKED
0492 * OUT: BIT PATTERN IN 'BITMAP'
0493 *

0494 *
0495 * CALLING SEQUENCE:
0496 * BLWP @LDOS
0497 * OUT: CHARACTERS WRITTEN TO PGT
0498 * PNT INITIALIZED
0499 *

0500 * WORKSPACE : VDPWP1
0501 * ROUTINE(S) : UNPACK, LDOS
0502 * REGISTER USAGE :
0503 *

GLOBAL DATA
LABELS

0504 *			
0505 *	R0	:	:
0506 *	R1	:	:
0507 *	R2	:	:
0508 *	R3	:	:
0509 *	R4	:	:
0510 *	R5	:	2 UNPACKED DATA
0511 *	R6	:	PACKED DATA
0512 *	R7	:	COMPARE BIT MASK
0513 *	R8	:	VRAM/STORAGE ADDRESS
0514 *	R9	:	STORAGE INCREMENT
0515 *	R10	:	BYTE COUNT
0516 *	R11	:	BL RETURN ADDRESS
0517 *	R12	:	TABLE POINTER
0518 *	R13	:	RETURN WP
0519 *	R14	:	RETURN PC
0520 *	R15	:	RETURN ST
0521 *			

0523	028E	0000	UNPACK DATA VDPWP1,\$+2	
0524	0292	0208	LI RB,BITMAP	STORE IN 'BITMAP'
	0294	00AC		
0525	0296	0209	LI R9,1	STORAGE INCREMENT = 1
	0298	0001		
0526	029A	C2DD	MOV *R13,R11	CHARACTER & FROM CALLERS RO
0527	029C	098B	SRL R11,8	PUT IN LSB
0528	029E	3AE0	MPY @C0006,R11	CALCULATE INDEX INTO TABLE
	02A0	02EE		
0529	02A2	022C	AI R12,PCHTB	ADD IN TABLE START
	02A4	012E		
0530	02A6	020A	LI R10,8	UNPACK 8 BYTES WORTH
	02AB	0008		
0531	02AA	101A	JMP MODO	UNPACK
0532			*	
0533	02AC	028E	LDCS DATA VDPWP1,\$+2	
0534			* WRITE PATTERN NAMES TO PNT	
0535	02B0	0208	LI RB,NTBA+>4000	REFERENCE PNT
	02B2	4400		
0536	02B4	06A0	BL @SENDAD	SEND ADDRESS TO VDP
	02B6	0238		
0537	02B8	0709	SETO R9	RESET COUNT
0538	02BA	0589	LDCS1 INC R9	NEXT NAME
0539	02BC	06C9	SWPB R9	POSITION LSB
0540	02BE	D809	MOV8 R9,@VRAM	WRITE IT
	02C0	F120		
0541	02C2	06C9	SWPB R9	RESTORE R9
0542	02C4	0289	CI R9,40*24	PNT FULL?
	02C6	03C0		
0543	02CB	1AF8	JL LDCS1	N, LOOP
0544			* WRITE CHARACTER SET TO PGT	
0545	02CA	0208	LI RB,PGBA+>4000	REFERENCE PGT
	02CC	4800		
0546	02CE	06A0	BL @SENDAD	SEND ADDRESS TO VDP
	02D0	02B6		
0547	02D2	0208	LI RB,VRAM	STORE IN VRAM
	02D4	F120		
0548	02D6	04C9	CLR R9	STORAGE INCREMENT = 0
0549	02D8	020C	LI R12,PCHTB	START AT BEGINNING OF TABLE
	02DA	02A4		
0550	02DC	020A	LI R10,256*8	CALCULATE UNPACKED SIZE
	02DE	0800		
0551			*	
0552			* FALL THROUGH TO 'MODO'	
0553			*	
0554			*	
0555			* UNPACK CHARACTER GENERATOR TABLE	
0556			*	
0557	02E0	0207	MODO LI R7,>0101	SET SHIFT MASK
	02E2	0101		
0558	02E4	0706	SETO R6	INIT. HOLDING REGISTER
0559	02E6	04C5	MOD1 CLR R5	INIT. DATA REGISTER
0560	02E8	0204	LI R4,>0001	INIT. ADDER
	02EA	0001		
0561	02EC	0203	LI R3,6	DO 6 BITS
	02EE	0006		
0562		02EE	C0006 EQU #-2	
0563	02F0	0B14	MOD2 SRC R4,1	POSITION ADDER
0564	02F2	0B17	SRC R7,1	POSITION MASK, NEW BYTE?
0565	02F4	1701	JNC MOD3	N, SKIP

0566 02F6 D1BC		MOVB *R12+,R6	Y, GET NEW BYTE
0567 02F8 2187	MOD3	CDC R7,R6	HAVE WE A '1' HERE ?
0568 02FA 1601		JNE MOD4	N, VALUE OK
0569 02FC B144		AB R4,R5	Y, ADD IN ADDER
0570 02FE 0603	MOD4	DEC R3	COUNT BIT
0571 0300 15F7		JGT MOD2	LOOP TILL 6 BITS DONE
0572 0302 D605		MOVB R5,*R8	STORE UNPACKED BYTE
0573 0304 A209		A R9,R8	UPDATE STORAGE POINTER
0574 0306 060A		DEC R10	COUNT BYTE
0575 0308 15EE		JGT MOD1	LOOP TILL ALL BYTES DONE
0576 030A 0380		RTWP	DONE, RETURN TO CALLER

```

0578      *      THIS ROUTINE SETS, RESETS OR TESTS A SPECIFIED
0579      *      PIXEL, DEPENDING ON ENTRY POINT. IF REFERENCE IS MADE
0580      *      TO A PIXEL OFF SCREEN THEN AN IMMEDIATE RETURN IS
0581      *      MADE.
0582      *
0583      * CALLING SEQUENCE:
0584      *      BLWP @PIXOFF          RESET PIXEL
0585      *      OR
0586      *      BLWP @PIXON          SET PIXEL
0587      *      OR
0588      *      BLWP @PIXTST        TEST PIXEL
0589      *      DATA <BYTE REPLY ADDRESS>
0590      *
0591      * IN : XLOC = X CO-ORDINATE
0592      *      YLOC = Y CO-ORDINATE
0593      *      FBCOL= FGND./BGND. COLOURS
0594      *
0595      * OUT ST2 = BIT STATUS (PIXTST ONLY)
0596      *
0597      *      WORKSPACE      :      VDPWPO
0598      *      ROUTINE(S)     :      PIXON, PIXOFF, SENDAD
0599      *      REGISTER USAGE :
0600      *
0601      *
0602      *      R0      TMP. STORAGE ! SHIFT COUNT
0603      *      R1      X-LOC      ! Y-LOC      XLOC, YLOC
0604      *      R2
0605      *      R3
0606      *      R4      ***** RESERVED *****
0607      *      R5      ***** RESERVED *****
0608      *      R6      ***** RESERVED *****
0609      *      R7      ***** RESERVED *****
0610      *      R8      Y ORDINATE / VRAM ADDRESS
0611      *      R9      X ORDINATE / BIT MASK
0612      *      R10     CELL DOT ROW
0613      *      R11     BL RETURN ADDRESS
0614      *      R12     SOCB/SZCB/COC INSTRUCTION
0615      *      R13     RETURN WP
0616      *      R14     RETURN PC
0617      *      R15     RETURN ST
0618      *
0619      *
0620 030C 0000  PIXTST DATA VDPWPO, $+2
0621 0310 020C      LI R12, >2009          R12='COC R9, R0'
0622      0312 2009
0622      0312' TMASK EQU $-2
0623 0314 1009      JMP PIX1
0624      *
0625 0316 030C' PIXOFF DATA VDPWPO, $+2
0626 031A 020C      LI R12, >5009          R12='SZCB R9, R0'
0627      031C 5009
0627      031C' RMASK EQU $-2
0628 031E 1004      JMP PIX1
0629      *
0630 0320 0316' PIXON DATA VDPWPO, $+2
0631 0324 020C      LI R12, >F009          R12='SOCB R9, R0'
0632      0326 F009
0632      *
0633 0328 D241  PIX1  MOVB R1, R9          GET X ORDINATE
0634 032A 0989      SRL R9, 8            POSITION IT

```

0635	032C	C201	MOV R1,R8	GET CO-ORDINATES
0636			* CALCULATE 32 x CELL ROW	
0637	032E	0A28	SLA R8,2	
0638	0330	0248	ANDI R8,>03E0	
	0332	03E0		
0639			* CHECK TO SEE IF Y-ORDINATE IS LEGAL (X MUST BE!!)	
0640	0334	0288	CI R8,192*4	VALID?
	0336	0300		
0641	0338	142C	JHE NBUMP	N, EXIT
0642			*	
0643	033A	C281	MOV R1,R10	GET CO-ORDINATES
0644	033C	024A	- ANDI R10,>0007	CALCULATE CELL DOT ROW
	033E	0007		
0645			* CALCULATE CELL DOT COLUMN, KEEP AS SHIFT COUNT IN R0	
0646	0340	C009	MOV R9,R0	
0647	0342	0240	ANDI R0,>0007	
	0344	0007		
0648			*	
0649	0346	0939	SRL R9,3	CALCULATE CELL COLUMN
0650			* CALCULATE CELL £ (=CELL COLUMN+[32xCELL ROW])	
0651	0348	A209	A R9,R8	
0652	034A	0A38	SLA R8,3	FORM PGT INDEX
0653	034C	A20A	A R10,R8	REFERENCE REQUIRED ROW
0654			*	
0655			* IF THE PGT IS NOT LOCATED AT 0 THEN THIS NEEDS TO BE	
0656			* ADDED TO R8 !!!	
0657			ASMIF PGTBA1	
0658			AI R8,PGTBA1	ADD IN PGT BASE ADDRESS
0659			ASMEND	
0660			*	
0661	034E	06A0	BL @SENDAD	SEND ADDRESS TO VDP
	0350	02D0		
0662	0352	04C9	CLR R9	READY MASK REGISTER
0663	0354	0220	AI R0,MASKTB	ADD IN MASK TABLE BASE
	0356	03B0		
0664	0358	D250	MOVB *R0,R9	FETCH MASK
0665	035A	D020	MOVB @VRAM,R0	GET CURRENT VRAM DATA
	035C	F120		
0666	035E	0268	ORI R8,>4000	SET WRITE BIT IN ADDRESS
	0360	4000		
0667	0362	06A0	BL @SENDAD	SEND ADDRESS TO VDP
	0364	0350		
0668	0366	048C	X R12	SET/RESET/TEST BIT
0669	0368	02C9	STST R9	SAVE STATUS
0670	036A	880C	C R12,@TMASK	TEST ?
	036C	0312		
0671	036E	1612	JNE WVRAM	N, UPDATE VRAM
0672			*	
0673			* TEST BIT EXIT ROUTINE	
0674			*	
0675	0370	0248	ANDI R8,>BFFF	SET READ BIT IN ADDRESS
	0372	BFFF		
0676	0374	0228	AI R8,CTBA1-PGTBA1	GET ADDRESS OF PCT ENTRY
	0376	2000		
0677	0378	06A0	BL @SENDAD	SEND ADDRESS TO VDP
	037A	0364		
0678	037C	8F3C	C *R12+,*R12+	DUMMY DELAY FOR VDP
0679	037E	D220	MOVB @FBCOL,R8	GET COLOUR TABLE ENTRY
	0380	0164		
0680	0382	2260	CDC @C2000,R9	Y, 'EQ' SET?

```

0384 0000
0681 0386 1601          JNE  BITOFF          N, BIT IS BACKGROUND
0682 0388 0948          SRL  R8,4            Y, POSITION FGND COLOUR
0683 038A 0248  BITOFF ANDI R8,>0F00        ISOLATE COLOUR
038C 0F00
0684 038E C27E          MOV  *R14+,R9        PICK UP STORAGE ADDRESS
0685 0390 D648          MOVB R8,*R9          SAVE COLOUR
0686 0392 0380  NBUMP  RTWP                  RETURN
0687          *
0688          *  SET/RESET EXIT ROUTINE
0689          *
0690 0394 D800  WVRAM  MOVB R0,@VRAM          WRITE UPDATED VRAM DATA
0396 F120
0691 0398 880C          C    R12,@RMASK        RESET PIXEL ?
039A 031C'
0692 039C 13FA          JEQ  NBUMP            Y, LEAVE COLOUR TABLE
0693          *  NOW UPDATE COLOUR TABLE ENTRY
0694 039E 0228          AI   R8,CTBA1-PGTBA1  GET ADDRESS OF PCT ENTRY
03A0 2000
0695 03A2 06A0          BL   @SENDAD          SEND ADDRESS TO VDP
03A4 037A'
0696 03A6 8F3C          C    *R12+,*R12+      DUMMY DELAY FOR VDP
0697 03A8 D820          MOVB @FBCOL,@VRAM      UPDATE COLOUR ENTRY
03AA 0380'
03AC F120
0698 03AE 0380          RTWP                  EXIT
0699          *
0700          *  BIT MASK TABLE
0701          *
0702          *  PIXEL -    0 1    2 3    4 5    6 7
0703 03B0 8040  MASKTB DATA >8040,>2010,>0804,>0201
0704          END
NO ERRORS,      NO WARNINGS

```


ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.ERROR
OBJECT ACCESS NAME= ADHOC.OBJ.ERROR
LISTING ACCESS NAME= ADHOC.LST.ERROR
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

LINE	KEY	NAME
------	-----	------

0028	A	ADHOC.SRC.IOBITS =>ADHOC.SRC.IOBITS
------	---	--

```
0002          IDT  'ERROR'
0003          *
0004          *
0005          *
0006          DEF  ERRS, STPY
0007          DEF  STPYA, ERRY
0008          *
0009          REF  CRLF, EFLG, ENUM, RTSTOR, TBEND
0010          REF  EVSKB, TYP$, ESCFLG, PLF, GOSB1, MODE, B2E, NLIN
0011          REF  TYP11$, MOVEB, MOVEL, PLC, SLN, CPYST
0012          REF  TYP$N$, TYPBE$, TYPC$, TYPEN$, IOB
0013          REF  NOERR, CRLF1T, CRLF2T, SYSERR, D10
0014          REF  ERR00T, ERR01T, ERR02T, ERR03T, ERR04T, ERR05T
0015          REF  ERR06T, ERR07T, ERR08T, ERR09T, ERR10T, ERR11T
0016          REF  ERR12T, ERR13T, ERR14T, ERR15T, ERR16T, ERR17T
0017          REF  ERR18T, ERR19T, ERR20T, ERR21T, ERR22T, ERR23T
0018          REF  ERR24T, ERR25T, ERR26T, ERR27T, ERR28T, ERR29T
0019          REF  ERR30T, ERR31T, ERR32T, ERR33T, ERR34T, ERR35T
0020          REF  ERR36T, ERR37T, ERR38T, ERR39T, ERR40T, ERR41T
0021          REF  ERR42T, ERR43T, ERR44T, ERR45T, ERR46T, ERR47T
0022          REF  ERR48T, ERR49T
0023          DEF  ERRLS2
0024          REF  SLT, FFLG, STACNT
0025          REF  D$RTWP, CRASH$, WAIT$
0026          DXOP OUTINT, 13
0027          DXOP EVFIX, 11
0028          COPY ADHOC. SRC. IOBITS
```


```

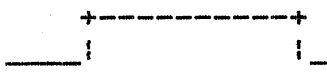
A0002      *
A0003      * THIS MODULE IS INCLUDED IN SOURCE MODULES
A0004      * BY USING A 'COPY' DIRECTIVE.
A0005      *
A0006      *
A0007      *****
A0008      *
A0009      * CRU DEFINITIONS
A0010      *
A0011      *****
A0012      0000 P10      EQU  >0000      PARALLEL I/O
A0013      * PARALLEL I/O - READ BITS
A0014      0000 KDS      EQU  P10+0      KEYBOARD DATA STROBE
A0015      *          EQU  P10+1      UNUSED
A0016      0002 D1$SIZ EQU  P10+2      DS01 SIZE (1=8",0=5.25")
A0017      0003 D1$DEN EQU  P10+3      DS01 DENSITY (0=SD,1=DD)
A0018      0004 FDCINT EQU  P10+4      FDC INTERRUPT-
A0019      0005 KBDINT EQU  P10+5      KBD INTERRUPT-
A0020      0006 VDPINT EQU  P10+6      VDP INTERRUPT-
A0021      0007 BUSINT EQU  P10+7      BUS INTERRUPT-
A0022      0008 KEYBRD EQU  P10+8      KEYBOARD DATA (8 BITS)
A0023      * PARALLEL I/O - WRITE BITS
A0024      0000 CLKLED EQU  P10+0      CLOCK LED
A0025      0001 KBDACK EQU  P10+1      KEYBOARD ACKNOWLEDGE-
A0026      0002 BUSACK EQU  P10+2      BUS INTERRUPT RESET-
A0027      0003 BTENBL EQU  P10+3      BUS TIMEOUT FLAG ENABLE
A0028      0004 DRVSIZ EQU  P10+4      DRIVE SIZE (1=8",0=5.25")
A0029      0005 ROMON  EQU  P10+5      0=ROM ON,1=ROM OFF
A0030      0006 BELLON EQU  P10+6      BELL ENABLE BIT
A0031      *          EQU  P10+7      UNUSED
A0032      *
A0033      *          EQU  >0020      UNUSED
A0034      0040 EIA02  EQU  >0040      PRINTER HARDWARE BASE ADDRESS
A0035      *          EQU  >0060      UNUSED
A0036      *          EQU  >0080      UNUSED
A0037      *          EQU  >00A0      UNUSED
A0038      00C0 CASS02 EQU  >00C0      CASSETTE HARDWARE BASE ADDRESS
A0039      00E0 DMAC   EQU  >00E0      DMAC HARDWARE BASE ADDRESS
A0040      *
A0041      * RESERVED OFF-BOARD CRU LOCATIONS
A0042      0200 PRMPOP EQU  >200      PROM PROGRAMMER BOARD
A0043      * READ BITS
A0044      *          EQU  PRMPOP+0
A0045      *          EQU  PRMPOP+1
A0046      *          EQU  PRMPOP+2
A0047      *          EQU  PRMPOP+3
A0048      0204 PRORDY EQU  PRMPOP+4
A0049      0205 PGM     EQU  PRMPOP+5
A0050      0206 PGMPUL EQU  PRMPOP+6      * TEST
A0051      0207 V30     EQU  PRMPOP+7
A0052      0208 EDATA  EQU  PRMPOP+8
A0053      * WRITE BITS
A0054      0200 PRORST EQU  PRMPOP+0
A0055      0201 EPTYPE EQU  PRMPOP+1
A0056      0204 VCCON  EQU  PRMPOP+4
A0057      *PGM      EQU  PRMPOP+5
A0058      0206 PROERR EQU  PRMPOP+6
A0059      *          EQU  PRMPOP+7
A0060      *EDATA    EQU  PRMPOP+8
A0061      0210 EPADR  EQU  PRMPOP+16
  
```

```

A0062      *
A0063      *   CENTRONICS PARALLEL PRINTER PORT
A0064      *
A0065      *   BIT           0       1       2       3       4       5       6       7       8       9
A0066      *   READ
A0067      *   WRITE      D7      D6      D5      D4      D3      D2      D1      D0      D5
A0068      *                               (LSB)                               (MSB)
A0069      *
A0070      0400 PPRINT EQU  >400
A0071      *
A0072      0408 PPDS  EQU  PPRINT+8
A0073      *
A0074      *
A0075      0409 PPBUSY EQU  PPRINT+9
A0076      *
A0077      *****
A0078      *
A0079      *   MEMORY MAPPED I/O DEFINITIONS
A0080      *
A0081      *****
A0082      F100 MAPPER EQU  >F100
A0083      F100 M$REG0 EQU  MAPPER+0
A0084      F102 M$REG1 EQU  MAPPER+2
A0085      F104 M$REG2 EQU  MAPPER+4
A0086      F106 M$REG3 EQU  MAPPER+6
A0087      F108 M$REG4 EQU  MAPPER+8
A0088      F10A M$REG5 EQU  MAPPER+10
A0089      F10C M$REG6 EQU  MAPPER+12
A0090      F10E M$REG7 EQU  MAPPER+14
A0091      F110 M$REG8 EQU  MAPPER+16
A0092      F112 M$REG9 EQU  MAPPER+18
A0093      F114 M$REGA EQU  MAPPER+20
A0094      F116 M$REGB EQU  MAPPER+22
A0095      F118 M$REGC EQU  MAPPER+24
A0096      F11A M$REGD EQU  MAPPER+26
A0097      F11C M$REGE EQU  MAPPER+28
A0098      F11E M$REGF EQU  MAPPER+30
A0099      *
A0100      F120 VDP  EQU  >F120
A0101      F120 VRAM EQU  VDP+0
A0102      F121 VDPREG EQU  VDP+1
A0103      *
A0104      *   TEXT / GRAPHICS 1 MODE STORAGE
A0105      0400 NTBA  EQU  >400
A0106      0700 CTBA  EQU  >700
A0107      0800 PGBA  EQU  >800
A0108      0780 SNTBA EQU  >780
A0109      0000 SPGBA EQU  >000
A0110      *   GRAPHICS 2 MODE STORAGE
A0111      1800 NTBA1 EQU  >1800
A0112      2000 CTBA1 EQU  >2000
A0113      0000 PGTBA1 EQU  >0000
A0114      1800 SNTBA1 EQU  >1800
A0115      3800 SPGBA1 EQU  >3800
A0116      *
A0117      F140 FDC  EQU  >F140
A0118      *

```

DATA STROBE 

BUSY 

MEMORY MAPPER LOCATION
 MAPPER REGISTERS 0..15

VDP VRAM ACCESS ADDRESS
 VDP REGISTER ACCESS ADDRESS

NAME TABLE
 COLOUR TABLE
 PATTERN GENERATOR TABLE
 SPRITE NAME TABLE
 SPRITE PATTERN GENERATOR TBL.

NAME TABLE
 COLOUR TABLE
 PATTERN GENERATOR TABLE
 SPRITE NAME TABLE
 SPRITE PATTERN GENERATOR TBL.

TMS9909 FLOPPY DISC CONTROLLER

```

0030      *
0031      * ERROR HANDLER CALLED FROM EDER IN GETLINE MODULE
0032      *
0033 0000 C80B  ERRLS2 MOV  R11,@RTSTOR      SAVE RETURN
      0002 0000
0034 0004 03C0      CKOF
0035 0006 0000      DATA TYP$N$,CRLF1T      OUT 'CRLF** '
0036 000A 04E0      CLR  @ESCFLG          ENABLE ESCAPE
      000C 0000
0037 000E C003      MOV  R3,R0            GET ASCII ERROR NUMBER
0038 0010 06C0      SWPB R0              RJ TENS DIGIT
0039 0012 0240      ANDI R0,>000F        MASK
      0014 000F
0040 0016 3820      MPY  @D10,R0          CONVERT
      0018 0000
0041 001A C283      MOV  R3,R10           COPY ERROR NUMBER
0042 001C 024A      ANDI R10,>000F        MASK
      001E 000F
0043 0020 A04A      A    R10,R1           ADD IN
0044 0022 C2A0      MOV  @MODE,R10        IDLE ?
      0024 0000
0045 0026 1304      JEQ  GETRR            Y, OK
0046 0028 C081      MOV  R1,R2            N, PUT ERROR # IN R2
0047 002A 0000      DATA TYP11$,>0D00    OUT A 'CR' TO COVER '** '
0048 002E 102A      JMP  ERRS0            AND DO PROPPER ERROR
0049 0030 0281  GETRR CI  R1,MAXERR        VALID?
      0032 0031
0050 0034 1203      JLE  GETER4            Y, LOOKUP
0051 0036 0202      LI   R2,SYSEERR        N, GET 'SYTEM ERROR'
      0038 0000
0052 003A 1003      JMP  GETER3
0053 003C A041  GETER4 A    R1,R1          GET WORD INDEX INTO TABLE
0054 003E C0A1      MOV  @ERRTBL(1),R2     GET ERROR TEXT ADDRESS
      0040 0170'
0055 0042 0207  GETER3 LI   R7,EVSKB      PUT THE MSG IN THE EVAL STACK
      0044 0000
0056 0046 06A0      BL   @MOVE#R          GO COPY IT
      0048 013A'
0057 004A 1D05      SBO  ROMON-PIO        TURN ROM OFF
0058 004C 75D7      SB   *R7,*R7          PUT NULL ON END OF ERR MSG
0059 004E 0000      DATA TYP$*,EVSKB     OUTPUT IT
0060 0052 0201      LI   R1,CRLF2T        POINT TO ' ** '
      0054 0000
0061 0056 0000      DATA TYPEN$          OUT STRING
0062 0058 0000      DATA TYPC$          OUT CRLF
0063 005A C2E0      MOV  @RTSTOR,R11      GET RETURN ADDRESS
      005C 0002'
0064 005E 045B      B    *R11            RETURN

```

```

0066      *
0067      * ERROR RECOVERY ROUTINE
0068      *
0069 0060 COAE ERRS   MOV  @-2(14),R2      GET INSTRUCTION
      0062 FFFE
0070 0064 0242      ANDI  R2,>3F          ISOLATE ERROR #
      0066 003F
0071 0068 03C0      CKOF
0072      *
0073      * COPY BACK THE ERROR MESSAGES
0074      *
0075 006A 0201      LI    R1,CPYST        REF START
      006C 0000
0076      ASMIF PIO
0077      LI    R12,PIO
0078      ASMELS
0079 006E 04CC      CLR  R12
0080      ASMEND
0081      *
0082 0070 0300  CPY1  LIM1 0              NO INTERRUPTS
      0072 0000
0083 0074 1E05      SBZ  ROMON-PIO        ROM ON
0084 0076 CC51      MOV  *R1,*R1+        COPY WORD
0085 0078 1D05      SBD  ROMON-PIO        ROM OFF
0086 007A 0300      LIM1 15              ENABLE INTERRUPTS
      007C 000F
0087 007E 02B1      CI   R1,TBEND        DONE COPY ?
      0080 0000
0088 0082 1AF6      JL   CPY1            N, LOOP
0089      *
0090 0084 C802  ERRS0 MOV  R2,@ENUM        SAVE IT
      0086 0000
0091 0088 04E0      CLR  @FFLG          KILL FORMATTING FLAG
      008A 0000
0092 008C 04E0      CLR  @PLF           KIL LOAD/SAVE FLAG
      008E 0000
0093 0090 C060      MOV  @EFLG,R1        ERROR COMMAND EXECUTED?
      0092 0000
0094 0094 1304      JEQ  ERRS01
0095 0096 04E0      CLR  @EFLG          Y - RESET ERROR HANDLER
      0098 0092'
0096 009A 0460      B    @GOSB1         DO SYSTEM GOSUB
      009C 0000
0097 009E C042  ERRS01 MOV  R2,R1        GET ERROR NUMBER
0098 00A0 C1E0      MOV  @IOB,R7        GET I/O BUFFER
      00A2 0000
0099 00A4 0202      LI   R2,CRLF1T      POINT TO 'CRLF** '
      00A6 0008'
0100 00AB 06A0      BL   @MOVE          COPY INTO BUFFER
      00AA 0142'
0101 00AC 02B1      CI   R1,MAXERR      VALID ERROR ?
      00AE 0031
0102 00B0 1203      JLE  ERR11          Y, LEAVE ALONE
0103 00B2 0202      LI   R2,SYSERR      N, GET SYSTEM ERROR
      00B4 0038'
0104 00B6 1003      JMP  ERR12          & OUTPUT
0105 00B8 A041  ERR11 A    R1,R1        WORD INDEX
0106 00BA C0A1      MOV  @ERRTBL(1),R2  GET ERROR TEXT
      00BC 0170'
0107 00BE 06A0  ERR12 BL   @MOVE        OUT TEXT

```

	00C0 0142'			
0108	00C2 0202	LI	R2, CRLF2T	OUT ' ** '
	00C4 0054'			
0109	00C6 06A0	BL	@MOVE	OUTPUT IT
	00C8 0142'			
0110	00CA C020	ERRS02	MOV @MODE, R0	CHECK MODE
	00CC 0024'			
0111	00CE 1325	JEG	ERRS2	IDLE
0112	00D0 C120	MOV	@SLN, R4	RUN, PICK UP LINE £
	00D2 0000			
0113	00D4 1015	JMP	ERRS1	

```

0115      *
0116      *      ENTRY POINT FOR STOP STATEMENT
0117      *
0118 00D6 COA0 STPYA MOV @PLC,R2
      00DB 0000
0119 00DA 0222      AI R2,-4      CHECK TO SEE IF THERE IS A
      00DC FFFC
0120 00DE 8B02      C R2,@SLT      STATEMENT AFTER THE STOP
      00E0 0000
0121 00E2 1A06      JL STPY
0122 00E4 C120      MOV @SLN,R4
      00E6 00D2
0123 00EB CB22      MOV @-2(R2),@SLN      SAVE LINE & FOR CONT
      00EA FFFE
      00EC 00E6
0124 00EE 1002      JMP STPY1
0125      *
0126      *      ENTRY POINT FOR ESCAPE KEY OR END
0127      *
0128 00F0 C120 STPY MOV @SLN,R4
      00F2 00EC
0129 00F4 06A0 STPY1 BL @MOVEB
      00F6 0000
0130 00FB 0A0D      DATA >0A0D
0131 00FA 53      TEXT 'Stop '
      00FB 74
      00FC 6F
      00FD 70
      00FE 20
0132 00FF 00      BYTE 0
0133      *
0134 0100 0720 ERRS1 SETD @ENUM      NON-FATAL ERROR
      0102 00B6
0135 0104 06A0      BL @MOVEB
      0106 0000
0136 0108 61      TEXT 'at '
      0109 74
      010A 20
0137 010B 00      BYTE 0
0138 010C 2F44 ERRS1A OUTINT R4      OUT CURRENT LINE &
0139 010E C060      MOV @STACNT,R1      GET STATEMENT INDENT COUNT
      0110 0000
0140 0112 1303      JEQ ERRS2      0, LEAVE LINE No.
0141 0114 DDE0      MOVB @B2E,*R7+      OUT ' '
      0116 0000
0142 011B 2F41      OUTINT R1      OUT INDENT COUNT
0143      *
0144 011A C060 ERRS2 MOV @ENUM,R1      GET ERROR NUMBER
      011C 0102
0145 011E 1604      JNE ERRS2A      NON-FATAL, SKIP
0146 0120 0201      LI R1,D$RTWP      FATAL, POINT TO A RTWP
      0122 0000
0147 0124 CB01      MOV R1,@>FFFE      KILL OFF NMI
      0126 FFFE
0148 012B 0000 ERRS2A DATA TYPBE$      OUTPUT BUFFER
0149 012A C060      MOV @ENUM,R1      GET ERROR NUMBER
      012C 011C
0150 012E 1302      JEQ ERRS2B      0, CRASH
0151 0130 0460      B @CRLF      <>0, RETURN TO CRLF
      0132 0000

```


0152	0134	0000	ERRS2B	DATA	WAIT\$	WAIT FOR I/O TO COMPLETE
0153	0136	0460		B	@CRASH\$	CALL SYSTEM CRASH HANDLER
	0138	0000				
0154			*			
0155	013A	0300	MOVE\$R	LIMI	0	NO INTERRUPTS
	013C	0000				
0156				ASMIF	PIO	
0157				LI	R12,2*PIO	LOAD CRUBASE
0158				ASMELS		
0159	013E	04CC		CLR	R12	
0160				ASMEND		
0161	0140	1E05		SBZ	ROMON-PIO	TURN ROM ON
0162		0142	MOVE	EQU	\$	
0163	0142	DDF2		MOVB	*R2+,*R7+	MOVE CHARACTER
0164	0144	15FE		JGT	MOVE	+VE, COPY
0165	0146	1102		JLT	NTERM	-VE, SPECIAL EXIT PATH
0166			*			0, EXIT
0167	0148	0607		DEC	R7	BACKUP BUFFER POINTER
0168	014A	045B	DONIT	B	*R11	RETURN
0169			*			
0170	014C	04C9	NTERM	CLR	R9	-VE, READY HOLDING REG
0171	014E	D267		MOVB	@-1(R7),R9	GET BACK BYTE
	0150	FFFF				
0172	0152	0509		NEG	R9	RESTORE IT
0173	0154	D9C9		MOVB	R9,@-1(R7)	PUT IT BACK
	0156	FFFF				
0174	0158	10F8	JMP	DONIT		EXIT

0176		*			
0177		*	ERROR COMMAND		
0178		*			
0179	015A C060	ERRY	MOV @MODE,R1		IDLE?
	015C 00CC'				
0180	015E 1304		JEG ERRY1		Y, TOGGLE ERROR ENABLE FLAG
0181	0160 2EE0		EVFIX @EFLG		SAVE ERROR HANDLER LINE NUMBER
	0162 0098'				
0182	0164 0460	B	@NLIN		NEW LINE
	0166 0000				
0183	0168 0560	ERRY1	INV @NDERR		TOGGLE FLAG
	016A 0000				
0184	016C 0460	B	@CRLF		EXIT
	016E 0132'				

0186	0170	0000	ERRTBL	DATA	ERR00T
0187	0172	0000		DATA	ERR01T
0188	0174	0000		DATA	ERR02T
0189	0176	0000		DATA	ERR03T
0190	0178	0000		DATA	ERR04T
0191	017A	0000		DATA	ERR05T
0192	017C	0000		DATA	ERR06T
0193	017E	0000		DATA	ERR07T
0194	0180	0000		DATA	ERR08T
0195	0182	0000		DATA	ERR09T
0196	0184	0000		DATA	ERR10T
0197	0186	0000		DATA	ERR11T
0198	0188	0000		DATA	ERR12T
0199	018A	0000		DATA	ERR13T
0200	018C	0000		DATA	ERR14T
0201	018E	0000		DATA	ERR15T
0202	0190	0000		DATA	ERR16T
0203	0192	0000		DATA	ERR17T
0204	0194	0000		DATA	ERR18T
0205	0196	0000		DATA	ERR19T
0206	0198	0000		DATA	ERR20T
0207	019A	0000		DATA	ERR21T
0208	019C	0000		DATA	ERR22T
0209	019E	0000		DATA	ERR23T
0210	01A0	0000		DATA	ERR24T
0211	01A2	0000		DATA	ERR25T
0212	01A4	0000		DATA	ERR26T
0213	01A6	0000		DATA	ERR27T
0214	01A8	0000		DATA	ERR28T
0215	01AA	0000		DATA	ERR29T
0216	01AC	0000		DATA	ERR30T
0217	01AE	0000		DATA	ERR31T
0218	01B0	0000		DATA	ERR32T
0219	01B2	0000		DATA	ERR33T
0220	01B4	0000		DATA	ERR34T
0221	01B6	0000		DATA	ERR35T
0222	01B8	0000		DATA	ERR36T
0223	01BA	0000		DATA	ERR37T
0224	01BC	0000		DATA	ERR38T
0225	01BE	0000		DATA	ERR39T
0226	01C0	0000		DATA	ERR40T
0227	01C2	0000		DATA	ERR41T
0228	01C4	0000		DATA	ERR42T
0229	01C6	0000		DATA	ERR43T
0230	01C8	0000		DATA	ERR44T
0231	01CA	0000		DATA	ERR45T
0232	01CC	0000		DATA	ERR46T
0233	01CE	0000		DATA	ERR47T
0234	01D0	0000		DATA	ERR48T
0235	01D2	0000		DATA	ERR49T

0236 0031 MAXERR EQU (*-ERRTBL-2)/2
0237 END

HIGHEST USED BASIC ERROR

NO ERRORS, NO WARNINGS

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.CALL
OBJECT ACCESS NAME= ADHOC.OBJ.CALL
LISTING ACCESS NAME= ADHOC.LST.CALL
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

```
0002          IDT 'CALL'
0003          *
0004          * CALL HANDLER
0005          *
0006          DEF CLLY
0007          REF EVSDZ
0008          REF NLIN
0009          REF CALLWP,CALL12
0010 2F80 ERROR EQU >2F80
0011          DXOP EVFIX,11
0012          *
0013          * CALL COMMAND
0014          *
0015          * CALL "ROUTINE",ADDRESS,PARMS.....,PARMS
0016          *
0017          * THE ROUTINE NAME FIELD IS OPTIONAL
0018          * PARMS (UPTO 12) ARE SEPARATED BY COMMAS
0019          * THESE ARE STORED IN THE USER'S REGISTERS R0 - R11;
0020          * R12 CONTAINS THE NUMBER OF ARGUMENTS PASSED OVER.
0021          *
0022          * PARMS SHOULD BE EITHER:
0023          *     INTEGER VALUES/EXPRESSIONS OR
0024          *     THE ADDRESS OF THE VARIABLE/ARRAY ELEMENT
0025          *     CONTAINING THE DESIRED PARAMETER.
0026          *
0027          * THE ADDRESS OF X IS PASSED BY - ADR(X)
```

```

0029      *
0030      * SET UP THE BLWP VECTOR IN R4,R5 FOR THE USER'S PROG
0031      *
0032 0000 020C  CLLY  LI  R12,CALLWP      REF "USER'S" WORKSPACE
          0002 0000
0033 0004 020B      LI  R11,CALL12      REF LAST+1 USEABLE REGISTER
          0006 0000
0034 0008 0420      BLWP @EVSDZ          ALLOW STRING CONSTANT
          000A 0000
0035 000C 1001      JMP  GETPC          OK, STRING CONSTANT ("----")
0036 000E 2F81      DATA ERROR+1      WRONG, STRING VARIABLE (*VAR)
0037      *                                NEITHER, DROP THROUGH TO GETPC
0038 0010 04C2  GETPC CLR  R2          RESET COUNT
0039 0012 C10C      MOV  R12,R4        SET CALLVEC'S WP
0040 0014 2EC5      EVFIX R5          GET PC
0041 0016 0280  CLL2  CI  R0,>3F00     PARAMETERS?
          0018 3F00
0042 001A 1604      JNE  CLL4          N, EXIT TO USER ROUTINE
0043 001C 0582      INC  R2            Y, INCREMENT COUNT
0044 001E 2EFC  CLL2A EVFIX *R12+     STUFF NEXT ARGUMENT INTO REGIST
0045 0020 82CC      C    R12,R11      REGISTER'S FULL?
0046 0022 1AF9      JL   CLL2         N - BACK FOR NEXT ARGUMENT
0047      *
0048      * ALL ARGUMENTS HAVE BEEN STORED IN THE REGISTERS
0049      * (OR ALL THAT ARE GOING TO BE HANDLED)
0050      *
0051 0024 C6C2  CLL4  MOV  R2,*R11      SAVE COUNT IN USER'S R12
0052 0026 0404      BLWP R4          BLWP TO USER'S ROUTINE
0053 0028 0460      B    @NLIN        RETURNS TO HERE
          002A 0000
0054      END
NO ERRORS,      NO WARNINGS

```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.CHRF
OBJECT ACCESS NAME= ADHOC.OBJ.CHRF
LISTING ACCESS NAME= ADHOC.LST.CHRF
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

```
0002          IDT  'CHRF'
0003          *
0004          *      POSF          ;STRING SEARCH FUNCTION
0005          *      MCHF          ;STRING MATCH FUNCTION
0006          *      LENF          ;STRING LENGTH FUNCTION
0007          *
0008          REF  EVSFR$          EXIT ENTRY TO EVALUATOR
0009          REF  FPAC2          FLOATING POINT ACCUMULATOR
0010          DEF  POSF          ENTRY PT. FOR 'POS' FUNCTION
0011          DEF  MCHF          ENTRY PT. FOR 'MCH' FUNCTION
0012          DEF  LENF          ENTRY PT. FOR 'LEN' FUNCTION
0013          *
0014          *      POS:-  SEARCH FOR THE FIRST STRING IN THE SECOND
0015          *              STRING AND RETURN THE STARTING POSITION
0016          *              OF THE FIRST MATCH.
0017          *              IF NO MATCH IS FOUND A '0' IS RETURNED.
0018          *
0019          *      MCH:-  MATCH STRING 1 INTO STRING 2 AND RETURN
0020          *              THE NUMBER OF CHARACTERS TO WHICH THEY
0021          *              AGREE. IF NO MATCH IS FOUND A '0' IS RETURNED
0022          *
0023          *      LEN:-  RETURN THE LENGTH OF THE STRING UPTO THE 1ST
0024          *              NULL BYTE.
```



```

0026      * CALLING SEQUENCE:
0027      *
0028      *      B @POSF
0029      *
0030      *      EXIT TO EVSFR$
0031      *
0032 0000 05A0 POSF INC @FPAC2      ; COUNT
      0002 0000
0033 0004 D031      MOV B *R1+,R0      ; GET FIRST BYTE
0034 0006 130A      JEQ POSF2      ; NOT FOUND, RETURN 0
0035 0008 9012      CB *R2,R0      ; SAME?
0036 000A 16FA      JNE POSF      ; N, CONTINUE LOOKING
0037 000C C0C2      MOV R2,R3      ; Y, LOOK FURTHER
0038 000E 0583      INC R3
0039 0010 C101      MOV R1,R4
0040      *
0041 0012 D033 POSF1 MOV B *R3+,R0      ; MATCH?
0042 0014 1305      JEQ POSF3      ; Y, RETURN FPAC
0043 0016 9D00      CB R0,*R4+      ; SAME?
0044 0018 13FC      JEQ POSF1      ; Y, CONTINUE
0045 001A 10F2      JMP POSF      ; N, START AGAIN
0046      *
0047 001C 04E0 POSF2 CLR @FPAC2
      001E 0002'
0048      * EXIT TO EVSFR WITH R2 RELOADED WITH 'FPAC'
0049 0020 0460 POSF3 B @EVSFR$
      0022 0000
0050      *
0051      *
0052      * CALLING SEQUENCE:
0053      *
0054      *      B @MCHF
0055      *
0056      *      EXIT TO EVSFR$
0057      *
0058 0024 9C52 MCHF CB *R2,*R1+      SAME?
0059 0026 16FC      JNE POSF3      N
0060 0028 DC92      MOV B *R2,*R2+      Y - BUT WERE THEY NULL?
0061 002A 13FA      JEQ POSF3      Y - THEN DON'T COUNT
0062 002C 05A0      INC @FPAC2      N - NOT NULL; THEN COUNT
      002E 001E'
0063 0030 10F9      JMP MCHF
0064      *
0065      *
0066      * CALLING SEQUENCE:
0067      *
0068      *      B @LENF
0069      *
0070      *      EXIT TO EVSFR$
0071      *
0072 0032 D072 LENF MOV B *R2+,R1
0073 0034 13F5      JEQ POSF3      ; DONE
0074 0036 05A0      INC @FPAC2      ; COUNT
      0038 002E'
0075 003A 10FB      JMP LENF
0076      END

```

NO ERRORS, NO WARNINGS

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.CRUF
OBJECT ACCESS NAME= ADHOC.OBJ.CRUF
LISTING ACCESS NAME= ADHOC.LST.CRUF
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

```

0002          IDT 'CRUF'
0003          *
0004          *      BASY          ;BASE COMMAND
0005          *      CRBY          ;CRB COMMAND
0006          *      CRBF          ;CRB FUNCTION
0007          *      CRFY          ;CRF COMMAND
0008          *      CRFF          ;CRF FUNCTION
0009          *
0010          * THIS MODULE ALSO CONTAINS THE SYSTEM CRASH ROUTINE.
0011          *
0012          REF FIX          ; INTEGER FIX
0013          REF EVSFR$      ; EXIT TO EVALUATOR WITH R2 RELOAD.
0014          REF NLIN        ; ENTRY EXIT TO MULTIPLEXOR
0015          REF GPRM        ; GET PARAMETER
0016          REF FPAC2       ; FLOATING POINT ACCUMULATOR
0017          REF BCRU        ; BASE CRU INDEX
0018          *
0019          DEF CRFY         ; CRF STATEMENT ENTRY POINT
0020          DEF CRBY         ; CRB STATEMENT ENTRY POINT
0021          DEF CRFF        ; CRF FUNCTION ENTRY POINT
0022          DEF CRBF        ; CRB FUNCTION ENTRY POINT
0023          DEF BASY        ; BASE STATEMENT ENTRY POINT
0024          *
0025          DEF CRASH$      <<< SYSTEM CRASH ROUTINE >>>
0026          *
0027          DXOP OUTINT,13   ; OUT INTEGER
0028          DXOP EVFIX,11   ; EVALUATE AND FIX
0029          *
0030          *****
0031          *
0032          *      SET USER CRU BASE.
0033          *
0034          *****
0035 0000 2EE0  BASY  EVFIX @BCRU          ; SAVE IN BCRU
0036          0002 0000
0036 0004 101A      JMP  CRUXIT          ; EXIT TO NLIN
0037          2F80  ERROR EQU >2F80      ; XOP XX,14 (ERROR CALL)
  
```

```

0039 *****
0040 *
0041 *           MULTIPLE BIT CRU WRITE.
0042 *
0043 *****
0044 *
0045 * CALLING SEQUENCE:
0046 *
0047 *       B @CRFY
0048 *
0049 *       EXIT TO NLIN
0050 *
0051 0006 06A0 CRFY BL @GPRM GET PARAMETERS
      000B 0000
0052 000A 0241 ANDI R1,>F MASK COUNT
      000C 000F
0053 000E 1304 JEQ CRF1 O=16 BITS
0054 0010 0281 CI R1,B BYTE?
      0012 000B
0055 0014 1501 JGT CRF1 N
0056 0016 06C3 SWPB R3 Y
0057 *
0058 0018 0A61 CRF1 SLA R1,6 POSITION
0059 001A 0221 AI R1,>3003 MAKE 'LDCR R3,X'
      001C 3003
0060 001E 100A JMP CRBY1
0061 *****
0062 *
0063 *           SINGLE BIT CRU WRITE.
0064 *
0065 *****
0066 *
0067 * CALLING SEQUENCE:
0068 *
0069 *       B @CRBY
0070 *
0071 *       EXIT TO NLIN
0072 *
0073 *
0074 0020 06A0 CRBY BL @GPRM GET DISPLACEMENT
      0022 000B'
0075 0024 0241 ANDI R1,>FF MASK
      0026 00FF
0076 0028 0221 AI R1,>1D00 GET SBD
      002A 1D00
0077 002C C0C3 MOV R3,R3 SET TO 1?
0078 002E 1602 JNE CRBY1 N
0079 0030 0221 AI R1,>0100 Y, CHANGE TO SBZ
      0032 0100
0080 *
0081 0034 C320 CRBY1 MOV @BCRU,R12
      0036 0002'
0082 0038 0481 D$XR1 X R1 DO CRU INSTRUCTION
0083 003A 0460 CRUXIT B @NLIN RETURN
      003C 0000

```

```

0085 *****
0086 *
0087 *           MULTIPLE BIT CRU READ
0088 *
0089 *****
0090 *
0091 * CALLING SEQUENCE:
0092 *
0093 *       B @CRFF
0094 *
0095 *       EXIT TO EVSFR$
0096 *
0097 003E 06A0 CRFF BL @FIX GET PARAMETER
      0040 0000
0098 0042 0241 ANDI R1,>F MASK
      0044 000F
0099 0046 C001 MOV R1,R0 SAVE
0100 0048 0A61 SLA R1,6 POSITION
0101 004A 0221 AI R1,>3401 MAKE 'STCR R1,X'
      004C 3401
0102 004E C320 MOV @BCRU,R12 GET BASE
      0050 0036
0103 0052 0481 X R1 DO STCR
0104 0054 C000 MOV R0,R0 16?
0105 0056 1304 JEQ CRFF1 Y
0106 0058 0280 CI R0,8 N, BYTE?
      005A 0008
0107 005C 1501 JGT CRFF1 N
0108 005E 0981 SRL R1,8 Y, POSITION RESULTS
0109 *
0110 0060 C801 CRFF1 MOV R1,@FPAC2
      0062 0000
0111 * RELOAD R2 WITH FPAC & RETURN TO EVSFR
0112 0064 0460 CRFF2 B @EVSFR$ RETURN ADR
      0066 0000
0113 *****
0114 *
0115 *           SINGLE BIT CRU READ
0116 *
0117 *****
0118 *
0119 * CALLING SEQUENCE:
0120 *
0121 *       B @CRBF
0122 *
0123 *       EXIT TO EVSFR$
0124 *
0125 *
0126 0068 06A0 CRBF BL @FIX GET DISPLACEMENT
      006A 0040
0127 006C 0241 ANDI R1,>FF MASK
      006E 00FF
0128 0070 0221 AI R1,>1F00 MAKE 'TB XX'
      0072 1F00
0129 0074 C320 MOV @BCRU,R12 GET BASE
      0076 0050
0130 0078 0481 EXECR1 X R1 DO TEST
0131 007A 16F4 JNE CRFF2 0, RT
0132 007C 05A0 INC @FPAC2 1
      007E 0062

```

0133 0080 10F1

JMP CRFF2

```
0135 *****
0136 *
0137 *          SYSTEM CRASH ROUTINE
0138 *          THIS ROUTINE IS ENTERED WITH R1=0 AND IS MEANT TO
0139 *          LOCKUP THE SYSTEM SUCH THAT THE ONLY WAY OUT IS
0140 *          VIA RESET.
0141 *          ENTRY POINT IS 'CRASH$'
0142 *
0143 *****
0144 0082 2860 CRASH$ XOR @D$XR1,R1          BUILD 'X R1' IN R1
      0084 003B'
0145 0086 0441          B      R1          EXECUTE IT
0146          END
NO ERRORS,      NO WARNINGS
```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.DEF
OBJECT ACCESS NAME= ADHOC.OBJ.DEF
LISTING ACCESS NAME= ADHOC.LST.DEF
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=


```
0002          IDT 'DEF'
0003          *
0004          *      DEFY          ;DEF COMMAND
0005          *
0006          REF LINE          ;MULTIPLEXOR
0007          REF UFT          ;USER FUNCTION TABLE
0008          REF MODE          ;MODE FLAG
0009          REF B56
0010          DEF DEFY
0011          * ABSTRACT:
0012          *
0013          *      PLACE THE USER FUNCTION PROGRAM-
0014          *      BYTE-COUNTER AND NUMBER OF ARGUMENTS
0015          *      IN USER-FUNCTION-TABLE (UFT) THUS
0016          *      DEFINING THE FUNCTION.
0017          *
0018          * CALLING SEQUENCE:
0019          *
0020          *      B @DEFY
```

```

0022      *
0023      *DEF COMMAND
0024      *
0025      *USER FUNCTION TABLE:
0026      *
0027      *      PBC
0028      *      £ OF ARGUMENTS
0029      *      ...
0030      *
0031 0000 0020  DEFY  MOV  @MODE,R0      ; RUNNING?
      0002 0000
0032 0004 130D      JEQ  DEF2          ; N, DON'T DEFINE
0033 0006 04C0      CLR  R0
0034 0008 D038      MOVB *R8+,R0      ; GET TYPE
0035 000A 0701      SETD R1          ; START COUNT
0036      *
0037 000C 0581  DEF1  INC  R1          ; COUNT ARGUMENTS
0038 000E 9838      CB   *R8+,@B56    ; =?
      0010 0000
0039 0012 16FC      JNE  DEF1          ; N
0040 0014 C0E0      MOV  @UFT,R3      ; Y, DEFINE FUNCTION
      0016 0000
0041 0018 0960      SRL  R0,6          ; GET INDEX
0042 001A A0C0      A    R0,R3        ; INDEX
0043 001C CCC8      MOV  R8,*R3+      ; SAVE ADR
0044 001E C4C1      MOV  R1,*R3      ; SAVE COUNT
0045 0020 0460  DEF2  B    @LINE      ; GOTO NEXT LINE
      0022 0000
0046      END
NO ERRORS,      NO WARNINGS
  
```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.DIM
OBJECT ACCESS NAME= ADHOC.OBJ.DIM
LISTING ACCESS NAME= ADHOC.LST.DIM
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

0002			IDT	'DIM'	
0003		*			
0004		*	DIMY		; DIMENSION COMMAND
0005		*			
0006			REF	EVAL, EVALS2	; RECURSIVE EVALUATOR
0007			REF	EVARZ	; EVALUATE VARIABLE
0008			REF	FIX	; FIX ARGUMENT
0009			REF	NLIN	; EXIT TO MULTIPLEXOR
0010			REF	VDT	; VARIABLE DEFINITION TABLE INDE
0011			REF	NVS	; NEXT VARIABLE STORAGE
0012			REF	NVD	; NEXT VARIABLE DEFINITION
0013			REF	DLIM	; PROGRAM DELIMITER
0014			REF	C6	; CONSTANT 6
0015			DEF	DIMY	
0016		*			
0017	2F80	ERROR	EQU	>2F80	; XOP XX, 14 (ERROR CALL)
0018	2FA0	ERROR2	EQU	ERROR+>20	
0019		*			
0020	0004	NRV	EQU	4	; NUMBER OF RESERVED WORDS

```
0022      *
0023      *      DIMENSION A VARIABLE.  A DIMENSIONED
0024      *      VARIABLE IS INDICATED IN THE VDT BY A
0025      *      NEGATIVE VALUE.  THE VARIABLE
0026      *      DEFINITION POINT TO THE INFORMATION
0027      *      VECTOR.
0028      *
0029      *INFORMATION VECTOR FORMAT:
0030      *
0031      *      D1, X1, D2, X2, ... DN, -1, ... DATA...
0032      *
0033      *WHERE  D1, D2, ... DN = DIMENSIONS
0034      *      X1, X2, ...    = MULTIPLIERS
0035      *
0036      *      X1 = D2+1 X D3+1 X ... DN+1
0037      *      X2 = D3+1 X D4+1 X ... DN+1
0038      *      ...
0039      *      XN = 1
0040      *
0041      * CALLING SEQUENCE:
0042      *
0043      *      B @DIMY
0044      *
0045      * EXCEPTIONS AND CONDITIONS:
0046      *
0047      *      EVALUATION ERRORS, STORAGE OVERFLOW
0048      *      EXPECTING DIMENSIONED VARIABLE
0049      *      INVALID DELIMITER
0050      *
```

```

0052 0000 04C0 DIMY CLR R0 ; READY R0
0053 0002 D038 MOV B *RB+, R0 ; GET BYTE
0054 0004 0280 CI R0, >4300 ; '$' ?
      0006 4300
0055 0008 1601 JNE TSTVAR ; N, CHECK IF ITS A VARIABLE
0056 000A D038 MOV B *RB+, R0 ; Y, GET THE VARIABLE NAME BYTE
0057 000C 0280 TSTVAR CI R0, NRV*>100+>7000 ; VARIABLE?
      000E 7400
0058 0010 1A58 JL ERR16 ; N
0059 0012 C3A0 MOV @VDT, R14 ; Y, GET SYMBOL ADR
      0014 0000
0060 0016 0970 SRL R0, 7
0061 0018 A380 A R0, R14 ; INDEX
0062 001A 022E AI R14, ->70*2 ; ELIMINATE DISPLACEMENT
      001C FF20
0063 001E C05E MOV *R14, R1 ; DEFINED?
0064 0020 161E JNE DIM3 ; Y, SEE IF EXCEEDS PREVIOUS
0065 *
0066 0022 0705 SETD R5 ; DO FULL EVALUATIONS
0067 0024 06A0 BL @EVALS2 ; DO INITIAL EVAL
      0026 0000
0068 0028 04F6 CLR *R6+ ; ZERO COUNT
0069 002A 1002 JMP DIM2
0070 *
0071 002C 06A0 DIM1 BL @EVAL ; EVALUATE NEXT SUBSCRIPT
      002E 0000
0072 *
0073 0030 06A0 DIM2 BL @FIX ; FIX PARAMETER (R2)
      0032 0000
0074 0034 0646 DECT R6 ; POP COUNT
0075 0036 C0D6 MOV *R6, R3
0076 0038 0583 INC R3 ; COUNT DIMENSION
0077 003A CDB1 MOV R1, *R6+ ; PUSH DIMENSION
0078 003C 0736 SETD *R6+ ; LEAVE SPACE FOR DEL MULTIPLIER
0079 003E CDB3 MOV R3, *R6+ ; PUSH COUNT AGAIN
0080 0040 0280 CI R0, >3F00 ; ?
      0042 3F00
0081 0044 13F3 JEQ DIM1 ; Y
0082 0046 0280 CI R0, >4B00 ; ??
      0048 4B00
0083 004A 1607 JNE DIME37 ; N, ERROR
0084 004C 0646 DECT R6 ; RETRIEVE COUNT
0085 004E C016 MOV *R6, R0
0086 0050 C280 MOV R0, R10 ; GET INFORMATION VECTOR LENGTH
0087 0052 0A2A SLA R10, 2 ; COUNT X 4
0088 0054 0201 LI R1, 1 ; SET DEL(N) TO 1
      0056 0001
0089 0058 1011 JMP DIM5
0090 *
0091 005A 2FA0 DIME37 DATA ERROR2, 37
  
```

```

0093 005E 0608 DIM3 DEC R8 ; BACKUP TO VARIABLE
0094 0060 0420 BLWP @EVARZ ; EVALUATE
      0062 0000
0095 0064 1026 JMP DIM7
0096 *
0097 0066 0226 DIM4 AI R6, -4 ; DO ANOTHER DIMENSION
      0068 FFFC
0098 006A C096 MOV *R6, R2 ; GET DIMENSION
0099 006C 0582 INC R2 ; INCREMENT
0100 006E 3881 MPY R1, R2 ; GET PRODUCT
0101 0070 C082 MOV R2, R2 ; OVERFLOW?
0102 0072 1626 JNE DIME10 ; Y
0103 0074 C043 MOV R3, R1 ; SET R1
0104 0076 1124 JLT DIME10
0105 0078 C981 MOV R1, @-2(6) ; STORE DEL IN INFORMATION VECTO
      007A FFFE
0106 *
0107 007C 0600 DIM5 DEC R0 ; DONE?
0108 007E 16F3 JNE DIM4 ; N
0109 0080 0226 AI R6, -4 ; FINISHED, CALCULATE VECTOR
      0082 FFFC
0110 0084 C016 MOV *R6, R0
0111 0086 0580 INC R0 ; 1ST DIMENSION + 1
0112 0088 3840 MPY R0, R1 ; GET FINAL DIMENSION SIZE
0113 008A 38A0 MPY @C6, R2 ; X 6 BYTES
      008C 0000
0114 008E C082 MOV R2, R2 CHECK FOR OVERFLOW
0115 0090 1617 JNE DIME10 REPORT ERROR
0116 0092 A0CA A R10, R3 ; ADD INFORMATION VECTOR LENGTH
0117 0094 C0A0 MOV @NVS, R2 ; DEFINE
      0096 0000
0118 0098 6083 S R3, R2
0119 009A 8802 C R2, @NVD ; OK?
      009C 0000
0120 009E 1A10 JL DIME10 ; N, STORAGE OVERFLOW
0121 00A0 C782 MOV R2, *R14 ; Y, SET IN SYMBOL TABLE
0122 00A2 C802 MOV R2, @NVS ; UPDATE NVS
      00A4 0096
0123 *
0124 00A6 CCB6 DIM6 MOV *R6+, *R2+ ; MOVE INFORMATION VECTOR
0125 00A8 05A6 INC @-2(6) ; DONE?
      00AA FFFE
0126 00AC 16FC JNE DIM6 ; N
0127 *
0128 00AE 04C0 CLR R0
0129 00B0 D038 MOVB *R8+, R0 ; GET NEXT BYTE
0130 *
0131 00B2 0280 DIM7 CI R0, >3F00 ; ", ?
      00B4 3F00
0132 00B6 13A4 JEQ DIMY ; Y, DO NEXT VARIABLE
0133 00B8 C800 MOV R0, @DLIM ; N, SAVE IN DLIM
      00BA 0000
0134 00BC 0460 DIM8 B @NLIN ; RETURN
      00BE 0000
0135 *
0136 00C0 2FBA DIME10 DATA ERROR+10 ; STORAGE OVERFLOW
0137 *
0138 00C2 2F90 ERR16 DATA ERROR+16 ; EXPECTING DIMENSIONED VARIABLE
0139 END

```

NO ERRORS, NO WARNINGS

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC. ENTER
OBJECT ACCESS NAME= ADHOC.OBJ. ENTER
LISTING ACCESS NAME= ADHOC.LST. ENTER
ERROR ACCESS NAME=
OPTIONS= BUNLST, DUNLST, MUNLST
MACRO LIBRARY PATHNAME=


```

0002          IDT  'ENTER'
0003          *
0004          *
0005          *
0006          REF  EDIT, EVSDZ, LNSZ, IOB, LINE, C1, ESCFLG
0007          REF  PLC, SLN, SFSN
0008          DEF  ENTERY
0009          *
0010          2FB0  ERROR  EGU  >2FB0
0011          2FA0  ERROR2 EGU  ERROR+>20
0012          *
0013 0000 0420  ENTERY BLWP @EVSDZ          ; LOOK FOR STRING
          0002 0000
0014 0004 1002          JMP  ENT1          ; " OR $
0015 0006 1001          JMP  ENT1
0016 0008 2F8E          DATA ERROR+14     ; EXPECTING STRING VARIABLE
0017          *
0018 000A C1E0  ENT1   MOV  @IOB, R7        ; GET I/O BUFFER START
          000C 0000
0019 000E 0205          LI   R5, LNSZ      ; GET MAX. SIZE
          0010 0000
0020          *
0021 0012 DDF2  ENT2   MOVB *R2+, *R7+
0022 0014 1304          JEG  ENT3
0023 0016 0605          DEC  R5
                                ; COPY BYTE OVER
                                ; NULL, END OF STRING
                                ; CHARACTER, STILL ROOM?

```

```

002
0024 0018 16FC          JNE  ENT2          ; Y, CONTINUE COPY
0025 001A 2FA0          DATA ERROR2, 35   ; N, PARAMETER ERROR
0026          *
0027 001E C160  ENT3   MOV  @PLC, R5        ; GET CURRENT PLC
          0020 0000
0028 0022 C825          MOV  @-2(R5), @SLN  ; SAVE CURRENT LINE £
          0024 FFFE
          0026 0000
0029 0028 06A0          BL   @EDIT         ; EDIT I/O BUFFER
          002A 0000
0030 002C 4820          SZC  @C1, @ESCFLG   ; REMOVE NOESC
          002E 0000
          0030 0000
0031 0032 C060          MOV  @SLN, R1        ; GET BACK MY LINE £
          0034 0026'
0032 0036 06A0          BL   @SFSN         ; GO FIND IT
          0038 0000
0033 003A C808          MOV  R8, @PLC       ; SAVE MY NEW PLC
          003C 0020'
0034 003E 0460          B    @LINE         ; CONTINUE EXECUTION
          0040 0000
0035          END
NO ERRORS,      NO WARNINGS

```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.ESCAPE
OBJECT ACCESS NAME= ADHOC.OBJ.ESCAPE
LISTING ACCESS NAME= ADHOC.LST.ESCAPE
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

```
0002          IDT  'ESCAPE'
0003          *
0004          *
0005          *
0006          DEF  ESCY
0007          DEF  NOEY
0008          REF  ESCFLG
0009          REF  NLINO
0010          *
0011          *ESCAPE ENABLE COMMAND
0012          *
0013 0000 04E0  ESCY  CLR  @ESCFLG          ; RESET ESCAPE FLAG
          0002 0000
0014 0004 1002          JMP  ESCR          ; RETURN
0015          *
0016          *
0017          *ESCAPE DISABLE COMMAND
0018          *
0019 0006 0720  NOEY  SETD @ESCFLG          ; SET ESCAPE FLAG
          0008 0002'
0020 000A 0460  ESCR  B    @NLINO
          000C 0000
0021          END
NO ERRORS,      NO WARNINGS
```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.FOR
OBJECT ACCESS NAME= ADHOC.OBJ.FOR
LISTING ACCESS NAME= ADHOC.LST.FOR
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

```

0002      IDT 'FOR'
0003      *
0004      *      FORY      ;FOR COMMAND
0005      *      NXTY      ;NEXT COMMAND
0006      *      FOR2      ;DELETE FROM FOR/NEXT STACK
0007      *      GSIM      ;GET SIMPLE VARIABLE
0008      *
0009      2F80 ERROR EQU >2F80      ;XOP XX,14 (ERROR CALL)
0010      2FA0 ERROR2 EQU ERROR+>20
0011      *
0012      REF ADDF      ;ADD
0013      REF SUBF      ;SUBTRACT
0014      REF EVERZ      ;EVALUATE EXPRESSION
0015      REF NLIN,NLINO ;EXIT TO MULTIPLEXOR
0016      REF BUS      ;BEGINNING USER STORAGE
0017      REF EUS      ;END USER STORAGE
0018      REF FNS      ;FOR/NEXT STACK
0019      REF NVD      ;NEXT VARIABLE DEFINITION
0020      REF NVS      ;NEXT VARIABLE STORAGE
0021      REF PLC      ;PROGRAM LINE COUNTER
0022      REF SLT      ;STATEMENT LINE TABLE
0023      REF VDT      ;VARIABLE DEFINITION TABLE
0024      REF VNT      ;VARIABLE NAME TABLE
0025      REF B56
0026      REF NXTXB      ;NEXT COMMAND BYTE
0027      DEF FORY, FOR2
0028      DEF NXTY
0029      DEF GSIM
0030      *
0031      0004 NRV EQU 4      ;NUMBER OF RESERVED WORDS
  
```

```
0033      *      EXECUTE FOR COMMAND.  WHEN A FOR
0034      *      STATEMENT IS EXECUTED, THE FOR/NEXT
0035      *      STACK (FNS) IS SEARCHED FOR A ZERO
0036      *      ENTRY.  DURING THE SEARCH, IF
0037      *      AN IDENTICAL ENTRY IS FOUND, IT IS
0038      *      DELETED AND THE STACK IS ROLLED DOWN.
0039      *
0040      *      A PRE-TEST IS MADE TO SEE IF THE
0041      *      CONDITION IS MET IN WHICH CASE, THE
0042      *      FOR COMMAND WILL SEARCH FOR A MATCHING
0043      *      NEXT.
0044      *
0045      *      THE FOR/NEXT STACK FORMAT IS:
0046      *
0047      *      !      /      /      /      /      /      /      /      /      /      !
0048      *      VAR -----STEP----- -----TO----- PBC  PLC
0049      *
0050      *      VAR = 0 INDICATES EMPTY SLOT
0051      *
0052      *      CALLING SEQUENCE:
0053      *
0054      *      B @FORY
0055      *
0056      *      EXIT TO NLIN
0057      *      EXCEPTIONS AND CONDITIONS:
0058      *
0059      *      FOR W/O NEXT, STACK OVERFLOW
0060      *      EVALUATION ERRORS
0061      *      STORAGE OVERFLOW, ILLEGAL DELIMITER
0062      *      EXPECTING SIMPLE VARIABLE
```

0064	0000	06A0	FOR Y	BL	@GSIM	; GET SIMPLE VARIABLE
	0002	0176				
0065	0004	C0D4		MOV	*R4, R3	; DEFINED?
0066	0006	160A		JNE	FOR0	; Y
0067	0008	C0E0		MOV	@NVS, R3	; N, DEFINE
	000A	0000				
0068	000C	0223		AI	R3, -6	
	000E	FFFA				
0069	0010	8803		C	R3, @NVD	; OK?
	0012	0000				
0070	0014	1A6F		JL	FORE10	; N
0071	0016	C503		MOV	R3, *R4	; Y, DEFINE
0072	0018	C803		MOV	R3, @NVS	; UPDATE NVS
	001A	000A				
0073			*			
0074	001C	C120	FOR0	MOV	@FNS, R4	; GET F/N STACK ADR
	001E	0000				
0075	0020	9838		CB	*R8+, @B56	; =?
	0022	0000				
0076	0024	1663		JNE	FORE36	; N, PROBLEM
0077			*			
0078	0026	C014	FOR1	MOV	*R4, R0	; DONE?
0079	0028	1314		JEQ	FOR4	; Y
0080	002A	8001		C	R1, R0	; N, SAME VARIABLE?
0081	002C	1306		JEQ	FOR1A	; Y
0082	002E	0224		AI	R4, 18	; N, MOVE TO NEXT
	0030	0012				
0083	0032	8804		C	R4, @EUS	; ANYMORE?
	0034	0000				
0084	0036	1AF7		JL	FOR1	; Y
0085	0038	2F8B		DATA	ERROR+11	; N, STACK OVERFLOW
0086			*			
0087	003A	06A0	FOR1A	BL	@FOR2	; DELETE
	003C	0040				
0088	003E	10F3		JMP	FOR1	
0089			*			
0090	0040	C144	FOR2	MOV	R4, R5	; DELETE FROM STACK
0091	0042	C004		MOV	R4, R0	
0092	0044	0220		AI	R0, 18	
	0046	0012				
0093			*			
0094	0048	CD70	FOR3	MOV	*R0+, *R5+	; MOVE UP
0095	004A	8800		C	R0, @EUS	; DONE?
	004C	0034				
0096	004E	12FC		JLE	FOR3	; N
0097	0050	045B		B	*R11	; Y

0099	0052	CD01	FOR4	MOV	R1,*R4+	; INSERT VAR NAME
0100	0054	0420		BLWP	@EVERZ	; GET INITIAL VALUE
	0056	0000				
0101	0058	0280		CI	RO,>3800	; TO?
	005A	3800				
0102	005C	1649		JNE	FORE37	; N, ERROR
0103	005E	C183		MOV	R3,R6	; SAVE FOR PRE-TEST
0104	0060	CCF2		MOV	*R2+,*R3+	; MOVE IN INITIAL VALUE
0105	0062	CCF2		MOV	*R2+,*R3+	
0106	0064	C4D2		MOV	*R2,*R3	
0107			*			
0108	0066	04F4		CLR	*R4+	; SET DEFAULT STEP TO 1
0109	0068	04D4		CLR	*R4	
0110	006A	05B4		INC	*R4+	
0111	006C	04F4		CLR	*R4+	
0112			*			
0113	006E	0420		BLWP	@EVERZ	; GET 'TO' VALUE
	0070	0056				
0114	0072	C1C4		MOV	R4,R7	; SAVE FOR PRE-TEST
0115	0074	CD32		MOV	*R2+,*R4+	; MOVE IN TERMINATING VALUE
0116	0076	CD32		MOV	*R2+,*R4+	
0117	0078	CD12		MOV	*R2,*R4+	
0118	007A	04C5		CLR	R5	; SET DEFAULT SIGN
0119	007C	0280		CI	RO,>3A00	; 'STEP'?
	007E	3A00				
0120	0080	160C		JNE	FOR6	; N
0121	0082	0420		BLWP	@EVERZ	; Y, GET STEP
	0084	0070				
0122	0086	C044		MOV	R4,R1	
0123	0088	0221		AI	R1,-12	; MOVE BACK TO STEP
	008A	FFF4				
0124	008C	C152		MOV	*R2,R5	; GET DIRECTION
0125	008E	1602		JNE	*+6	; FP
0126	0090	C162		MOV	@2(2),R5	; INTEGER
	0092	0002				
0127			*			
0128	0094	CC72		MOV	*R2+,*R1+	; MOVE INTO STEP
0129	0096	CC72		MOV	*R2+,*R1+	
0130	0098	C452		MOV	*R2,*R1	
0131			*			
0132	009A	C508	FOR6	MOV	R8,*R4	; MOVE IN PBC,PLC
0133	009C	0634		DEC	*R4+	; BACKUP OVER DLIM
0134	009E	C520		MOV	@PLC,*R4	
	00A0	0000				
0135			*			
0136	00A2	C086		MOV	R6,R2	; DO PRE-TEST
0137	00A4	C047		MOV	R7,R1	
0138	00A6	06A0		BL	@SUBF	; R2=R2-R1 (VAR-TERM)
	00A8	0000				
0139	00AA	C072		MOV	*R2+,*R1	; LOOK AT RESULT, INTEGER?
0140	00AC	1602		JNE	*+6	; N
0141	00AE	C052		MOV	*R2,R1	; Y
0142	00B0	131B		JEQ	FOR9	; = IMPLIES NOT DONE
0143			*			
0144	00B2	2845		XOR	R5,R1	; 'EXCLUSIVE OR' SIGNS
0145	00B4	1119		JLT	FOR9	; - IMPLIES LOOP NOT COMPLETE


```

0147 00B6 C164 FOR7 MOV @-16(4),R5 ;LOOK FOR NEXT
      00B8 FFF0
0148 00BA 04E4 CLR @-16(4) ;CLEAR FROM STACK
      00BC FFF0
0149 00BE C1A0 MOV @PLC,R6 ;GET PLC
      00C0 00A0'
0150 *
0151 00C2 0226 FOR8 AI R6,-4
      00C4 FFFC
0152 00C6 8806 C R6,@SLT ;ANY MORE STATEMENTS?
      00C8 0000
0153 00CA 1A15 JL ERR31 ;N
0154 00CC C216 MOV *R6,R8 ;GET PBC
0155 00CE A220 A @BUS,R8 ;GET ABSOLUTE ADDRESS
      00D0 0000
0156 00D2 9838 CB *R8+,@NXTXB ;NEXT?
      00D4 0000
0157 00D6 16F5 JNE FOR8 ;N
0158 00D8 06A0 BL @GSIM ;Y, GET SIMPLE VARIABLE
      00DA 0176'
0159 00DC 8141 C R1,R5 ;SAME?
0160 00DE 16F1 JNE FOR8 ;N, CONTINUE
0161 00E0 C806 MOV R6,@PLC ;Y, UPDATE PLC
      00E2 00C0'
0162 00E4 0460 B @NLINO
      00E6 0000
0163 *
0164 *FOR9 EQU DIM8
0165 00EB 0460 FOR9 B @NLIN
      00EA 0000
0166 *
0167 00EC 2FA0 FORE36 DATA ERROR2,36 ;MISSING "="
0168 *
0169 00F0 2FA0 FORE37 DATA ERROR2,37 ;ILLEGAL DELIMITER
0170 *
0171 00F4 2F8A FORE10 DATA ERROR+10 ;STORAGE OVERFLOW
0172 *
0173 00F6 2F9F ERR31 DATA ERROR+31 ;FOR W/O NEXT
0174 *
0175 00F8 2F94 ERR20 DATA ERROR+20 ;EXPECTING SIMPLE VARIABLE

```

```
0177      *      PROCESS THE FOOT OF A FOR/NEXT LOOP.
0178      *
0179      *      NEXT SEARCHS THE FOR/NEXT STACK (FNS)
0180      *      FOR A MATCHING SIMPLE VARIABLE.  THE
0181      *      STEP IS ADDED TO THE VARIABLE AND
0182      *      A COMPLETION CHECK IS MADE.  IF LESS
0183      *      THAN OR EQUAL, IT LOOPS BACK.  OTHERWISE,
0184      *      THE STACK VARIABLE IS SET TO ZERO,
0185      *      AND PROGRAM EXECUTION CONTINUES AFTER
0186      *      THE NEXT STATEMENT.
0187      *
0188      *  CALLING SEQUENCE:
0189      *
0190      *      B @NXTY
0191      *
0192      *      EXIT TO NLIN
0193      *
0194      *  EXCEPTIONS AND CONDITIONS:
0195      *
0196      *      EVALUATION ERRORS
0197      *      NEXT W/O FOR
0198      *
```

```

0200 00FA 06A0 NXTY BL @GSIM ; GET SIMPLE VARIABLE
      00FC 0176'
0201 00FE C194 MOV *R4,R6 ; GET ADR
0202 0100 1309 JEQ ERR32 ; NOT DEFINED
0203 0102 C120 MOV @FNS,R4 ; GET F/N STACK ADR
      0104 001E'
0204 *
0205 0106 8D01 NXT1 C R1,*R4+ ; SAME?
0206 0108 1307 JEQ NXT2 ; Y
0207 010A 0224 AI R4,16 ; N, MOVE TO NEXT
      010C 0010
0208 010E 8B04 C R4,@EUS ; MORE?
      0110 004C'
0209 0112 1AF9 JL NXT1 ; Y
0210 *
0211 0114 2FA0 ERR32 DATA ERROR->20 ; NEXT W/O FOR
0212 0116 0020 DATA 32
0213 *
0214 * (R4) = STEP 6 BYTE STORAGE AREA
0215 * (R6) = VARIABLE STORAGE 6 BYTE
0216 * 6(R4) = 'TO' VALUE 6 BYTE STORAGE AREA
0217 *
0218 0118 C084 NXT2 MOV R4,R2 SAVE PTR TO STEP
0219 011A C172 MOV *R2+,R5 STEP INTEGER ?
0220 011C 160F JNE NXT2C N, DO FP
0221 011E C152 MOV *R2,R5 Y, GET SIGN
0222 0120 C2C6 MOV R6,R11 SAVE PTR TO VARIABLE
0223 0122 C07B MOV *R11+,R1 INTEGER VARIABLE?
0224 0124 160B JNE NXT2C N, DO FP
0225 0126 C064 MOV @6(R4),R1 Y, LIMIT INTEGER?
      0128 0006
0226 012A 160B JNE NXT2C N, DO FP
0227 *
0228 * EVERYTHING INTEGER
0229 *
0230 012C C05B MOV *R11,R1 GET VARIABLE
0231 012E A052 A *R2,R1 R1= VAR + STEP
0232 0130 1901 JNO SPLAT1 OK
0233 0132 1004 JMP NXT2C OVERFLOW, DO FP
0234 0134 C6C1 SPLAT1 MOV R1,*R11 UPDATE VARIABLE
0235 0136 6064 S @6+2(R4),R1 R1= VAR - TO
      0138 0008
0236 013A 1010 JMP NXT2A MAYBE
0237 *
0238 * FP FOR/NEXT LOOP
0239 *
0240 013C C046 NXT2C MOV R6,R1 ; READY R2,R1
0241 013E C084 MOV R4,R2
0242 0140 06A0 BL @ADDF ; ADD STEP
      0142 0000
0243 0144 C042 MOV R2,R1 ; MOVE IN NEW VARIABLE
0244 0146 CDB1 MOV *R1+,*R6+
0245 0148 CDB1 MOV *R1+,*R6+
0246 014A C591 MOV *R1,*R6
0247 014C C044 MOV R4,R1
0248 014E 0221 AI R1,6 ; MOVE TO 'TO'
      0150 0006
0249 0152 06A0 BL @SUBF ; R2=R2-R1 (VAR-TO)
      0154 00A8'
0250 0156 C072 MOV *R2+,R1 ; LOOK AT RESULT, INTEGER?

```

0251	0158	1602		JNE	\$+6		;N
0252	015A	C052		MOV	*R2,R1		;Y
0253	015C	1305	NXT2A	JEQ	NXT3		;O, NOT DONE
0254			*				
0255	015E	2845		XOR	R5,R1		;EXCLUSIVE OR WITH STEP
0256	0160	1103		JLT	NXT3		;NOT DONE
0257	0162	04E4		CLR	@-2(4)		;DONE, SET TOP OF STACK
	0164	FFFE					
0258	0166	1005		JMP	NXT3A		
0259			*				
0260	0168	C224	NXT3	MOV	@12(R4),R8		;LOOP NOT COMPLETE
	016A	000C					
0261	016C	C824		MOV	@14(R4),@PLC		;RESTORE PBC & PLC
	016E	000E					
	0170	00E2'					
0262	0172	0460	NXT3A	B	@NLINO		
	0174	00E6'					

```

0264      *      GSIM WILL GET NEXT ITEM AND CHECK TO
0265      *      SEE IF IT IS A SIMPLE VARIABLE.
0266      *
0267      * CALLING SEQUENCE:
0268      *
0269      *      BL @GSIM
0270      *
0271      *      OUT (R4) = VARIABLE DEFINITION
0272      *
0273      * EXCEPTIONS AND CONDITIONS:
0274      *
0275      *      EXPECTING SIMPLE VARIABLE
0276      *
0277 0176 04C0  GSIM  CLR  R0                      ; GET VARIABLE
0278 0178 D038      MOV  *RB+,R0
0279 017A 0280      CI   R0,NRV*>100+>7000
0280      017C 7400
0281 017E 1ABC      JL   ERR20                    ; N, EXPECTING SIMPLE VARIABLE
0282 0180 C120      MOV  @VNT,R4                  ; GET SYMBOL TABLE ADR
0283      0182 0000
0284 0184 0970      SRL  R0,7                      ; GET INDEX
0285 0186 A100      A    R0,R4
0286 0188 C064      MOV  @->70*2(4),R1
0287      018A FF20
0288 018C 11B5      JLT  ERR20                    ; DIMENSIONED, ERROR
0289 018E C120      MOV  @VDT,R4
0290      0190 0000
0291 0192 A100      A    R0,R4
0292 0194 0224      AI   R4,->70*2
0293      0196 FF20
0294 0198 045B      RT
0295      END
NO ERRORS,      NO WARNINGS
  
```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.FORMAT
OBJECT ACCESS NAME= ADHOC.OBJ.FORMAT
LISTING ACCESS NAME= ADHOC.LST.FORMAT
ERROR ACCESS NAME=
OPTIONS= BUNLST, DUNLST, MUNLST
MACRO LIBRARY PATHNAME=

0002		IDT	'FORMAT'	
0003	*			
0004	*			
0005	*			
0006		DEF	CVBF	
0007		REF	CVBFR	
0008		REF	B2A, B30	
0009	*			
0010		*PRINT	FORMATTING	
0011	*			
0012	0000 C209	CVBF	MOV R9, R8	MARK FORMAT
0013	0002 C187		MOV R7, R6	MARK CHARACTER BUFFER
0014	0004 04C3		CLR R3	CLEAR DECIMAL FLAG
0015	0006 04C4		CLR R4	CLEAR BEFORE DECIMAL COUNT
0016	0008 04C5		CLR R5	CLEAR TOTAL DIGIT HOLDER COUNT
0017	*			
0018	000A 06A0	CVBF1	BL @CVBFT	GET TYPE
	000C 00EE			
0019	000E 100A		JMP CVBF4	NULL, DONE
0020	0010 1002		JMP CVBF2	DIGIT HOLDER
0021	0012 0703		SETO R3	DECIMAL
0022	0014 1004		JMP CVBF3	CHARACTER
0023	*			
0024	0016 C0C3	CVBF2	MOV R3, R3	BEFORE DECIMAL?
0025	0018 1601		JNE #+4	N
0026	001A 0584		INC R4	Y, COUNT
0027	001C 0585		INC R5	
0028	*			
0029	001E DDA0	CVBF3	MOVB @B2A, *R6+	OUT "**
	0020 0000			
0030	0022 10F3		JMP CVBF1	
0031	*			
0032	0024 022A	CVBF4	AI R10, -13	GET £ OF LEADING DIGITS
	0026 FFF3			
0033	0028 050A		NEG R10	
0034	002A 810A		C R10, R4	GREATER THAN DIGIT HOLDERS?
0035	002C 151A		JGT CVBFR1	Y, ERROR RETURN
0036	002E 6144		S R4, R5	GET NUMBER OF DIGITS PAST DECI
0037	0030 A14A		A R10, R5	INDEX TO ROUNDING DIGIT
0038	0032 0285		CI R5, 12	TOO LARGE?
	0034 000C			
0039	0036 1102		JLT CVBF5	N
0040	0038 0205		LI R5, 12	Y, SET TO 12
	003A 000C			
0041	*			
0042	003C C005	CVBF5	MOV R5, R0	SET TO ROUND
0043	003E 0600		DEC R0	
0044	0040 06A0		BL @CVBFR	ROUND
	0042 0000			
0045	0044 058A		INC R10	ADD NEW DIGIT
0046	0046 C209		MOV R9, R8	RESET FOR CVBFT
0047	0048 04C5		CLR R5	CLR £ FLAG
0048	*			
0049	004A 06A0	CVBF6	BL @CVBFT	GET FORMAT TYPE
	004C 00EE			
0050	004E 100A	CVBD7A	JMP CVBFR2	DONE
0051	0050 100C		JMP CVBF8	DIGIT HOLDER
0052	0052 102C		JMP CVBF15	"
0053	0054 0280		CI R0, >2C00	", ?
	0056 2C00			

0054	0058	1602	JNE	CVBF7	N
0055	005A	C145	MOV	R5,R5	Y, DIGIT OUT?
0056	005C	1320	JEQ	CVBF11	N, OUT SPACE
0057	005E	DDC0	CVBF7	MOVB R0,*R7+	Y, OUT CHARACTER
0058	0060	10F4	JMP	CVBF6	

0060	0062	C1C6	CVBFR1	MOV	R6,R7	FORMATTING OVERFLOW
0061	0064	CB47	CVBFR2	MOV	R7,@14(13)	RETURN UPDATE PTR
	0066	000E				
0062	0068	0380		RTWP		DONE
0063			*			
0064	006A	0604	CVBFB	DEC	R4	DIGIT HOLDER
0065	006C	8284		C	R4,R10	TIME FOR DIGIT?
0066	006E	111B		JLT	CVBF13	Y
0067	0070	1302		JEQ	CVBF8A	N, CHECK FLOATER
0068	0072	C104		MOV	R4,R4	FLOATER?
0069	0074	1611		JNE	CVBF10	N
0070	0076	0280	CVBF8A	CI	RO,>2400	N, MAYBE FLOATER THOUGH, "\$?
	0078	2400				
0071	007A	13F1		JEQ	CVBF7	Y, INSERT
0072	007C	0280		CI	RO,>5300	"S?
	007E	5300				
0073	0080	1605		JNE	CVBF9	N
0074	0082	C30C		MOV	R12,R12	Y, POSITIVE?
0075	0084	130C		JEQ	CVBF11	Y, INSERT BLANK
0076	0086	0200		LI	RO,>2D00	N, INSERT "-
	0088	2D00				
0077	008A	10E9		JMP	CVBF7	
0078			*			
0079	008C	0280	CVBF9	CI	RO,>3C00	"<?
	008E	3C00				
0080	0090	1603		JNE	CVBF10	N
0081	0092	C30C		MOV	R12,R12	Y, POSITIVE?
0082	0094	16E4		JNE	CVBF7	N, INSERT "<
0083	0096	1003		JMP	CVBF11	Y, INSERT BLANK
0084			*			
0085	0098	0280	CVBF10	CI	RO,>3000	"0?
	009A	3000				
0086	009C	1302		JEQ	CVBF12	Y
0087	009E	0200	CVBF11	LI	RO,>2000	N, INSERT BLANK
	00A0	2000				
0088	00A2	DDC0	CVBF12	MOVB	RO,*R7+	INSERT
0089	00A4	10D2		JMP	CVBF6	
0090			*			
0091	00A6	06A0	CVBF13	BL	@CVBF30	
	00A8	00DC				
0092	00AA	10CF		JMP	CVBF6	
0093			*			
0094	00AC	DDC0	CVBF15	MOVB	RO,*R7+	INSERT "
0095			*			
0096	00AE	06A0	CVBF16	BL	@CVBFT	GET BYTE
	00B0	00EE				
0097	00B2	10CD		JMP	CVBD7A	DONE
0098	00B4	100A		JMP	CVBF20	DIGIT HOLDER
0099	00B6	1000		JMP	#+2	
0100	00B8	0280		CI	RO,>3E00	">?
	00BA	3E00				
0101	00BC	1604		JNE	CVBF18	N
0102	00BE	C30C		MOV	R12,R12	Y, POSITIVE?
0103	00C0	1602		JNE	CVBF18	N
0104	00C2	0200		LI	RO,>2000	Y, OUT SPACE
	00C4	2000				
0105	00C6	DDC0	CVBF18	MOVB	RO,*R7+	OUT
0106	00C8	10F2		JMP	CVBF16	
0107			*			
0108	00CA	0604	CVBF20	DEC	R4	COUNT

0109 00CC 8284	C	R4,R10	TIME FOR DIGIT?
0110 00CE 1103	JLT	CVBF21	Y
0111 00D0 0200	LI	RO,>3000	N, OUT "0
00D2 3000			
0112 00D4 10F8	JMP	CVBF18	
0113	*		
0114 00D6 06A0	CVBF21 BL	@CVBF30	OUT DIGIT
00D8 00DC			
0115 00DA 10E9	JMP	CVBF16	
0116	*		
0117 00DC 060A	CVBF30 DEC	R10	COUNT DOWN
0118 00DE 0705	SETD	R5	ALLOW COMMA'S
0119 00E0 DDF3	MOVB	*R3+,*R7+	INSERT DIGIT
0120 00E2 1604	JNE	CVBF31	
0121 00E4 0607	DEC	R7	NO DIGIT
0122 00E6 0603	DEC	R3	
0123 00EB DDE0	MOVB	@B30,*R7+	INSERT "0
00EA 0000			
0124 00EC 045B	CVBF31 RT		

```

0126      *CHECK FORMAT TYPE
0127      *      BL @CVBFT
0128      *      NULL
0129      *      DIGIT HOLDER < $ S 0 9
0130      *      DECIMAL
0131      *      CHAR ,
0132      *
0133 00EF 04C0  CVBFT CLR R0          GET BYTE
0134 00F0 D038  MOVB *RB+,R0
0135 00F2 1323  JEQ CVBFT3          NULL
0136 00F4 0280  CI R0,>3C00        "<?"
      00F6 3C00
0137 00F8 131F  JEQ CVBFT2          Y
0138 00FA 0280  CI R0,>2400        "$?"
      00FC 2400
0139 00FE 131C  JEQ CVBFT2          Y
0140 0100 0280  CI R0,>5300        "S?"
      0102 5300
0141 0104 1319  JEQ CVBFT2          Y
0142 0106 0280  CI R0,>3000        "0?"
      0108 3000
0143 010A 1316  JEQ CVBFT2          Y
0144 010C 0280  CI R0,>3900        "??"
      010E 3900
0145 0110 1313  JEQ CVBFT2          Y
0146      0114' B2E EQU $+2
0147 0112 0280  CI R0,>2E00        ".?"
      0114 2E00
0148 0116 130F  JEQ CVBFT1          Y
0149 0118 0280  CI R0,>4500        "E?"
      011A 4500
0150 011C 1606  JNE CVBFT4          N
0151 011E 0200  LI R0,>2000        Y, DEFAULT TO SPACE
      0120 2000
0152 0122 C30C  MOV R12,R12        NEGATIVE?
0153 0124 1302  JEQ $+6          N
0154 0126 0200  LI R0,>2D00        Y, OUT "-"
      0128 2D00
0155      *
0156 012A 0280  CVBFT4 CI R0,>5E00  "??"
      012C 5E00
0157 012E 1602  JNE CVBFT0          N
0158 0130 0200  LI R0,>2E00        Y, REPLACE WITH ".
      0132 2E00
0159      *
0160 0134 05CB  CVBFT0 INCT R11
0161 0136 05CB  CVBFT1 INCT R11
0162 0138 05CB  CVBFT2 INCT R11
0163 013A 045B  CVBFT3 RT
0164      END
  
```

NO ERRORS, NO WARNINGS

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.FUNC
OBJECT ACCESS NAME= ADHOC.OBJ.FUNC
LISTING ACCESS NAME= ADHOC.LST.FUNC
ERROR ACCESS NAME=
OPTIONS= BUNLST, TUNLST, DUNLST, MUNLST
MACRO LIBRARY PATHNAME=

LINE	KEY	NAME
------	-----	------

0003	A	ADHOC.SRC.IOBITS =>ADHOC.SRC.IOBITS
------	---	--

0002
0003

IDT 'FUNC'
COPY ADHOC.SRC.IOBITS

```

A0002      *
A0003      * THIS MODULE IS INCLUDED IN SOURCE MODULES
A0004      * BY USING A 'COPY' DIRECTIVE.
A0005      *
A0006      *
A0007      *****
A0008      *
A0009      * CRU DEFINITIONS
A0010      *
A0011      *****
A0012      0000 PIO EQU >0000 PARALLEL I/O
A0013      * PARALLEL I/O - READ BITS
A0014      0000 KDS EQU PIO+0 KEYBOARD DATA STROBE
A0015      * EQU PIO+1 UNUSED
A0016      0002 D1$SIZ EQU PIO+2 DS01 SIZE (1=8",0=5.25")
A0017      0003 D1$DEN EQU PIO+3 DS01 DENSITY (0=SD,1=DD)
A0018      0004 FDCINT EQU PIO+4 FDC INTERRUPT-
A0019      0005 KBDINT EQU PIO+5 KBD INTERRUPT-
A0020      0006 VDPINT EQU PIO+6 VDP INTERRUPT-
A0021      0007 BUSINT EQU PIO+7 BUS INTERRUPT-
A0022      0008 KEYBRD EQU PIO+8 KEYBOARD DATA (8 BITS)
A0023      * PARALLEL I/O - WRITE BITS
A0024      0000 CLKLED EQU PIO+0 CLOCK LED
A0025      0001 KBDACK EQU PIO+1 KEYBOARD ACKNOWLEDGE-
A0026      0002 BUSACK EQU PIO+2 BUS INTERRUPT RESET-
A0027      0003 BTENBL EQU PIO+3 BUS TIMEOUT FLAG ENABLE
A0028      0004 DRVSIZ EQU PIO+4 DRIVE SIZE (1=8",0=5.25")
A0029      0005 ROMON EQU PIO+5 0=ROM ON,1=ROM OFF
A0030      0006 BELLON EQU PIO+6 BELL ENABLE BIT
A0031      * EQU PIO+7. UNUSED
A0032      *
A0033      * EQU >0020 UNUSED
A0034      0040 EIA02 EQU >0040 PRINTER HARDWARE BASE ADDRESS
A0035      * EQU >0060 UNUSED
A0036      * EQU >0080 UNUSED
A0037      * EQU >00A0 UNUSED
A0038      00C0 CASS02 EQU >00C0 CASSETTE HARDWARE BASE ADDRESS
A0039      00E0 DMAC EQU >00E0 DMAC HARDWARE BASE ADDRESS
A0040      *
A0041      * RESERVED OFF-BOARD CRU LOCATIONS
A0042      0200 PRMPOP EQU >200 FROM PROGRAMMER BOARD
A0043      * READ BITS
A0044      * EQU PRMPOP+0
A0045      * EQU PRMPOP+1
A0046      * EQU PRMPOP+2
A0047      * EQU PRMPOP+3
A0048      0204 PRORDY EQU PRMPOP+4
A0049      0205 PGM EQU PRMPOP+5
A0050      0206 PGMFUL EQU PRMPOP+6
A0051      0207 V30 EQU PRMPOP+7
A0052      0208 EDATA EQU PRMPOP+8
A0053      * WRITE BITS
A0054      0200 PRORST EQU PRMPOP+0
A0055      0201 EPTYPE EQU PRMPOP+1
A0056      0204 VCCON EQU PRMPOP+4
A0057      *PGM EQU PRMPOP+5
A0058      0206 PROERR EQU PRMPOP+6
A0059      * EQU PRMPOP+7
A0060      *EDATA EQU PRMPOP+8
A0061      0210 EPADR EQU PRMPOP+16

```

* TEST

```

A0062      *
A0063      *   CENTRONICS PARALLEL PRINTER PORT
A0064      *
A0065      *   BIT           0       1       2       3       4       5       6       7       8       9
A0066      *   READ
A0067      *   WRITE      D7      D6      D5      D4      D3      D2      D1      D0      D5
A0068      *                   (LSB)                                (MSB)
A0069      *
A0070      0400 PPRINT EQU  >400
A0071      *
A0072      0408 PPDS  EQU  PPRINT+8          DATA STROBE  ____+____+____
A0073      *
A0074      *
A0075      0409 PPBUSY EQU  PPRINT+9          BUSY          ____+-----+____
A0076      *
A0077      *****
A0078      *
A0079      *   MEMORY MAPPED I/O DEFINITIONS
A0080      *
A0081      *****
A0082      F100 MAPPER EQU  >F100          MEMORY MAPPER LOCATION
A0083      F100 M$REG0 EQU  MAPPER+0        MAPPER REGISTERS 0..15
A0084      F102 M$REG1 EQU  MAPPER+2
A0085      F104 M$REG2 EQU  MAPPER+4
A0086      F106 M$REG3 EQU  MAPPER+6
A0087      F108 M$REG4 EQU  MAPPER+8
A0088      F10A M$REG5 EQU  MAPPER+10
A0089      F10C M$REG6 EQU  MAPPER+12
A0090      F10E M$REG7 EQU  MAPPER+14
A0091      F110 M$REG8 EQU  MAPPER+16
A0092      F112 M$REG9 EQU  MAPPER+18
A0093      F114 M$REGA EQU  MAPPER+20
A0094      F116 M$REGB EQU  MAPPER+22
A0095      F118 M$REGC EQU  MAPPER+24
A0096      F11A M$REGD EQU  MAPPER+26
A0097      F11C M$REGE EQU  MAPPER+28
A0098      F11E M$REGF EQU  MAPPER+30
A0099      *
A0100      F120 VDP  EQU  >F120
A0101      F120 VRAM EQU  VDP+0          VDP VRAM ACCESS ADDRESS
A0102      F121 VDPREG EQU  VDP+1        VDP REGISTER ACCESS ADDRESS
A0103      *
A0104      *   TEXT / GRAPHICS 1 MODE STORAGE
A0105      0400 NTBA  EQU  >400          NAME TABLE
A0106      0700 CTBA  EQU  >700          COLOUR TABLE
A0107      0800 PGBA  EQU  >800          PATTERN GENERATOR TABLE
A0108      0780 SNTBA EQU  >780          SPRITE NAME TABLE
A0109      0000 SPGBA EQU  >000          SPRITE PATTERN GENERATOR TBL.
A0110      *   GRAPHICS 2 MODE STORAGE
A0111      1800 NTBA1 EQU  >1800        NAME TABLE
A0112      2000 CTBA1 EQU  >2000        COLOUR TABLE
A0113      0000 PGTBA1 EQU  >0000       PATTERN GENERATOR TABLE
A0114      1B00 SNTBA1 EQU  >1B00       SPRITE NAME TABLE
A0115      3800 SPGBA1 EQU  >3800       SPRITE PATTERN GENERATOR TBL.
A0116      *
A0117      F140 FDC   EQU  >F140          TMS9909 FLOPPY DISC CONTROLLER
A0118      *
      0004      DEF  MODF, SYSF, ABSF, NKYF
      0005      DEF  INTF, FRAF
      0006      REF  FPAC, FPAC2, TEMP

```

```
0007          REF  FIX,EVSFR
0008          REF  SYSFLG,GETC#,SYSLMT
0009          REF  C4A00,VDPST
0010 2FB0  ERROR  EQU  >2FB0
0011 2FA0  ERROR2 EQU  ERROR+>20
0012      *
0013      *  DEFINE FLOATING POINT XOPS FOR THIS MODULE
0014      *
0015          DXOP  LOADF,0          ;LOAD FPAC
0016          DXOP  STORE,1         ;STORE FPAC
0017          DXOP  FADD,2          ;ADD TO FPAC
0018          DXOP  FSUB,3          ;SUBTRACT FROM FPAC
0019          DXOP  SCALE,6         ;SCALE FPAC
0020          DXOP  NORMAL,7        ;NORMALIZE FPAC
0021          DXOP  CLEAR,8         ;CLEAR FPAC
0022          DXOP  NEGATE,9        ;NEGATE FPAC
0023          DXOP  FLOATF,10       ;FLOAT FPAC
```


0025	*			
0026	*		SYS FUNCTION	
0027	*			
0028		SYSF	BL @FIX	GET PARAMETER
0029			CI R1,VDPST	VDP STATUS CHECK ?
0030			JNE SYSFA1	NO , NORMAL SYSTEM FUNCTION
0031			MOVB @VDPREG,R1	YES , FETCH VDP STATUS
0032			SRL R1,8	ALIGN
0033			JMP INPF1	RETURN VALUE TO CALLER
0034	*			
0035		SYSFA1	CI R1,SYSLMT	VALID ?
0036			JLE SYSFA2	YES , RETURN SYSTEM CONSTANT
0037			DATA ERROR2,35	NO , PARAMETER ERROR
0038	*			
0039		SYSFA2	A R1,R1	GET WORD INDEX
0040			MOV @SYSFLG(R1),R1	GET VALUE
0041			JMP INPF1	RETURN VALUE TO CALLER

```

0043      *
0044      * INTEGER PART FUNCTION
0045      *
0046 0024 C052 INTF  MOV  *R2,R1      INTEGER ALREADY?
0047 0026 130F      JEQ  INPF3      Y
0048 0028 2C12      LOADF *R2      N - LOAD FPAC
0049      *
0050      * IF NUMBER => SCALING FACTOR  GET FLOATING PT OVERFLOW
0051      *
0052 002A 0241      ANDI R1,>7F00    MASK OUT ALL BUT EXPONENT
0053      002C 7F00      C      R1,@C4A00    => SCALING FACTOR?
0054 0032 1407      JHE  INPF2
0055 0034 2DA0      SCALE @C4A00    N - SCALE OFF FRACTION
0056      0036 0030'
0056 003B 2DC0      NORMALIZE 0
0057 003A 1003      JMP  INPF2
0058      *
0059 003C 2E00      INPF1  CLEAR 0      SAVE £
0060 003E C801      MOV  R1,@FPAC2
0061      0040 0000
0061 0042 0202      INPF2  LI   R2,FPAC    RETURN ADDRESS
0062      0044 0000
0062 0046 0460      INPF3  B    @EVSFR     RETURN
0063      004B 0000
0063      *
0064      *ABSOLUTE FUNCTION
0065      *
0066 004A 2C12      ABSF   LOADF *R2      ; LOAD FPAC
0067 004C C052      MOV  *R2,R1      ; CHECK SIGN
0068 004E 1303      JEQ  ABSF1      ; INTEGER
0069 0050 15FA      JGT  INPF3      ; +, RT
0070 0052 2E40      NEGATE 0      ; -, NEGATE
0071 0054 10F6      JMP  INPF2      ; RETURN
0072      *
0073 0056 0760      ABSF1  ABS  @FPAC2    ; INTEGER, TAKE ABS
0074      0058 0040'
0074 005A 19F3      JND  INPF2      ; NO OVERFLOW
0075 005C 0200      LI   R0,>4480    ; OVERFLOW, LOAD 32768 (FP)
0076      005E 4480
0076 0060 04C2      CLR  R2
0077 0062 2C00      LOADF R0      ; LOAD R0,R1,R2
0078 0064 10EE      JMP  INPF2
0079      *
0080      *KEY FUNCTION
0081      *
0082 0066 0000      NKYF   DATA GETC$    ; CHARACTER?
0083 0068 10EC      JMP  INPF2      ; N - RETURN 0
0084 006A 1000      NOP                      ; Y - NORMAL CHARACTER
0085 006C 06C0      SWPB R0      ; Y - ESCAPE CHARACTER
0086 006E 06A0      BL    @FIX      ; FIX ARGUMENT
0087      0070 0002'
0087 0072 C041      MOV  R1,R1      ; ARG=0?
0088 0074 1302      JEQ  NKYF1      ; Y, RETURN LEC
0089 0076 8040      C      R0,R1      ; N, ARG=LEC?
0090 0078 16E4      JNE  INPF2      ; N, RETURN 0
0091      *
0092 007A C040      NKYF1  MOV  R0,R1
0093 007C 10DF      JMP  INPF1

```

```

0094      *
0095      * FRACTIONAL PART FUNCTION
0096      *
0097      * EXPONENT SIZE:
0098      * ALL FRA 40 < INT+FRA < 4A ALL INT
0099      *
0100 007E 0012 FRAF MOV *R2,R0 ; INTEGER?
0101 0080 1603 JNE FRAF2 ; N
0102 0082 0202 FRAF1 LI R2,FPAC ; Y, RETURN 0
      0084 0044
0103 0086 045B RT
0104      *
0105 0088 0240 FRAF2 ANDI R0,>7F00 ; GET EXPONENT
      008A 7F00
0106 008C 0280 CI R0,>4900 ; ALL INTEGER ?
      008E 4900
0107 0090 15F8 JGT FRAF1 ; Y
0108 0092 2C12 LOADF *R2 ; N, LOAD FPAC
0109 0094 0280 CI R0,>4100 ; ALL FRACTION ?
      0096 4100
0110 0098 11F4 JLT FRAF1 ; Y
0111 009A 2C60 STORE @TEMP ; N, MOVE TO TEMP
      009C 0000
0112 009E 2DA0 SCALE @C4A00 ; GET INTEGER
      00A0 0036
0113 00A2 2DC0 NORMAL 0 ; NORMALIZE
0114 00A4 2E40 NEGATE 0 ; NEGATE
0115 00A6 2CA0 FADD @TEMP ; ADD NEGATED INTEGER IN FPAC
      00AB 009C
0116 00AA 10EB JMP FRAF1

```

```

0118      *
0119      *          MOD FUNCTION
0120      *          =====
0121      *
0122      *          FORMAT :-
0123      *          A = MOD [ ARG 1 , ARG 2 ]
0124      *
0125      *          THIS FUNCTION PROVIDES AN INTEGER MOD OF ARG 1
0126      *          THE MOD £ IS GIVEN BY ARG 2.
0127      *          EG. A=MOD(9,7) RETURNS A VALUE FOR A OF 2
0128      *
0129      *          R1 CONTAINS ARG 2, *R2 CONTAINS ARG 1
0130      *          R3,R10 ARE SPARE
0131      *          EXIT IS VIA RT.
0132      *
0133 00AC C0C1 MODF  MOV  R1,R3          SAVE ARG 2
0134 00AE 1501      JGT  MODF1         +VE, OK
0135 00B0 2F9E ERR30 DATA ERROR+30    RANGE ERROR
0136      *
0137 00B2 06A0 MODF1 BL   @FIX          FIX PARAMETER
0138      00B4 0070'
0138 00B6 C001      MOV  R1,R0          COPY INTO R0
0139 00B8 08F0      SRA  R0,15         FILL WITH SIGN BIT
0140      * R0,R1 NOW CONTAINS ARG 1
0141 00BA 0183      DIVS R3            DO DIVIDE
0142 00BC C041      MOV  R1,R1         TEST SIGN BIT
0143 00BE 15BE      JGT  INPF1         +VE, OK
0144 00C0 13BD      JEG  INPF1         0, OK
0145 00C2 A043      A    R3,R1         -VE, ADJUST !
0146 00C4 10BB      JMP  INPF1
0147      END
NO ERRORS,      NO WARNINGS
  
```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.GOSUB
OBJECT ACCESS NAME= ADHOC.OBJ.GOSUB
LISTING ACCESS NAME= ADHOC.LST.GOSUB
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

```

0002          IDT 'GOSUB'
0003      *
0004      * ROUTINE LIST:
0005      *
0006      *      GOTY          ;GOTO COMMAND
0007      *      GOSY          ;GOSUB COMMAND
0008      *      RTNY          ;RETURN COMMAND
0009      *      POPY          ;POP COMMAND
0010      *
0011      * EXTERNAL ROUTINES:
0012      *
0013      *      REF SFSN,CRLF,SLN
0014      *      REF LINE0,LINE2,LINE5,NLINO
0015      *      DXOP EVFIX,11
0016      2F80 ERROR EQU >2F80          XOP XX,14 (ERROR CALL)
0017      2FA0 ERROR2 EQU ERROR+>20
0018      *
0019      * EXTERNAL DATA:
0020      *
0021      *      REF ELNM          ;ERROR LINE NUMBER
0022      *      REF FNS          ;FOR/NEXT STACK
0023      *      REF GSC          ;GOSUB STACK COUNTER
0024      *      REF GSS          ;GOSUB STACK
0025      *      REF PLC          ;PROGRAM LINE COUNTER
0026      *      REF SLT          ;STATEMENT LOCATION TABLE
0027      *      REF BUS          ;BEGINNING USER STORAGE
0028      *      REF C4          ;>0004
0029      *
0030      * ABSTRACT:
0031      *
0032      * EXECUTE GOTO, GOSUB AND SYSTEM GOSUB COMMANDS.
0033      * PROVIDE FOR ON COMMAND
0034      *
0035      *      STACK FORMAT:  PLC
0036      *                      PBC
0037      *
0038      *                      PLC
0039      *                      PBC
0040      *
0041      *                      ...
0042      *
0043      * IF PBC = 0, GOTO NEXT LINE, OTHERWISE CONTINUE EXECUTION ON
0044      * THE SAME LINE AS THE CALL WAS MADE (NEEDED FOR ON COMMAND T
0045      * ENSURE YOU DO NOT JUMP BACK INTO THE PARAMETER LIST.)
0046      *
0047      * CALLING SEQUENCE:
0048      *
0049      *      B @GOSB1
0050      *      B @GOSY
0051      *      B @GOTY
0052      *
0053      *      EXIT TO LINE0 IN DEMULTIPLEXOR (RUN)
0054      *
0055      * EXCEPTIONS AND CONDITIONS:
0056      *
0057      *      STACK OVERFLOW
0058      *      STACK UNDERFLOW
0059      *      NO SUCH LINE £
0060      *

```

```

0062      *
0063      * ENTRY POINT:
0064      *
0065      DEF GOSY      ; GOSUB COMMAND
0066      DEF GOS1,GOS2
0067      DEF GOSB1     ; SYSTEM ENTRY
0068      DEF GOSON     ; ON COMMAND ENTRY
0069      DEF GOTY      ; GOTO ENTRY
0070      *
0071      * ENTRY FOR INPUT ERROR HANDLER/ERRECOVY
0072      *
0073 0000 C160 GOSB1  MOV  @PLC,R5          GET ADDRESS OF CURRENT LINE
      0002 0000
0074 0004 C825      MOV  @-2(5),@ELNM      GET ERROR LINE £
      0006 FFFE
      0008 0000
0075 000A 04C6      CLR  R6                SET FOR FIRST OF NEXT LINE
0076 000C 102A      JMP  GOS1A
0077      *
0078      * ENTRY FOR CONT (EDIT)
0079      *
0080 000E C160 GOS2   MOV  @PLC,R5          GET ADDRESS OF CURRENT LINE
      0010 0002
0081 0012 1031      JMP  GOS2A
0082      *
0083      * INTERRUPT ENTRY POINT (RUN)
0084      *
0085 0014 C160 GOS1   MOV  @PLC,R5          GET ADDRESS OF CURRENT LINE
      0016 0010
0086 0018 1024      JMP  GOS1A
0087      *
0088      * GOTO ENTRY - CHECK SYNTAX (VALID TERMINATOR?)
0089      *
0090 001A 04C3 GOTY   CLR  R3                INDICATE A GOTO      JG 12/1/8
0091 001C 1001      JMP  GOSYA
0092      *
0093      * GOSUB ENTRY - CHECK SYNTAX (VALID TERMINATOR?)
0094      *
0095 001E 0703 GOSY   SETO R3                INDICATE A GOSUB      JG 12/1/8
0096 0020 C188 GOSYA MOV  R8,R6                ; SET TO SAVE PBC
0097 0022 05C6      INCT R6
0098 0024 04C0      CLR  R0
0099 0026 D036      MOVB *R6+,R0                ; GET DELIMITER
0100 0028 1306      JEQ  GOSY1                ; EOL
0101 002A 0280      CI  R0,>3C00                ; ::?
      002C 3C00
0102 002E 1313      JEQ  GOSON1                ; Y      JG 02/3/8
0103 0030 0280      CI  R0,>4700                ; !!
      0032 4700
0104 0034 1628      JNE  GOSYE                ; N - ERROR
0105      *
0106      * ON ENTRY POINT - IGNORE REST OF THE LINE
0107      *
0108      0036' GOSON  EQU  $                                JG 12/1/8
0109 0036 C160 GOSY1  MOV  @PLC,R5      SAVE ADD OF CURRENT STMT £ JG 12/1/ 8
      0038 0016'
0110 003A 130D      JEQ  GOSON1                ENTERED FROM KEYBOARD MODE? JG 15/1/
0111 003C C0C3      MOV  R3,R3                N - GOTO?      JG 12/1/8
0112 003E 130B      JEQ  GOSON1                Y      JG 12/1/8
0113      *

```

```

0114      * GOSUB - NOTHING FOLLOWING STATEMENT - GET PBC OF NEXT LINE
0115      *
0116 0040 C185      MOV R5,R6      GOTO NEXT STATMENT      JG 12/1/8
0117 0042 0226      AI R6,-4
           0044 FFFC
0118 0046 8806      C R6,@SLT      ; DONE?
           0048 0000
0119 004A 1A19      JL GOS6A      Y - SET PBC=0      JG 12/1/8
0120 004C C806      MOV R6,@PLC    ; N, UPDATE
           004E 0038
0121 0050 C196      MOV *R6,R6
0122 0052 A1A0      A @BUS,R6
           0054 0000
0123 0056 D078      GOSON1 MOVB *R8+,R1      GET LINE NUMBER
0124 0058 06C1      SWPB R1
0125 005A D078      MOVB *R8+,R1
0126 005C 06C1      SWPB R1
0127 005E 0922      SRL R2,2      GOTO?
0128 0060 130A      JEQ GOS2A
0129 0062 C0E0      GOS1A MOV @GSC,R3      N - GOSUB
           0064 0000
0130 0066 8803      C R3,@FNS      ; ROOM?
           0068 0000
0131 006A 140B      JHE ERR11      ; N, ERROR
0132 006C CCC6      MOV R6,*R3+    ; Y, SAVE PBC OR NULL AND PLC
0133 006E CCE0      MOV @PLC,*R3+
           0070 004E
0134 0072 C803      MOV R3,@GSC    ; UPDATE GSC
           0074 0064
0135 0076 06A0      GOS2A BL @SFSN      SEARCH FOR STATEMENT £
           0078 0000
0136 007A 0460      B @LINE0
           007C 0000
0137      *
0138 007E 04C6      GOS6A CLR R6      PBC=0      JG 12/1/8
0139 0080 10EA      JMP GOSON1      JG 12/1/8
0140      *
0141 0082 2F8B      ERR11 DATA ERROR+11 ; STACK OVERFLOW
0142 0084 2F8C      ERR12 DATA ERROR+12 ; STACK UNDERFLOW
0143 0086 2FA0      GOSYE DATA ERROR2,37 ILLEGAL DELIMITER
  
```



```

0145      *
0146      * ABSTRACT:
0147      *
0148      * POPS THE RETURN ADDRESS FROM THE GOSUB STACK (GSS) AND
0149      * EFFECTS A TRANSFER TO THAT POPPED ADDRESS. IF THE
0150      * UNSTACKED PLC = 0 THEN IT INDICATES THAT THE GOSUB WAS
0151      * ENTERED DIRECTLY FROM KEYBOARD MODE. IF THE PBC <> 0
0152      * THEN AN EXIT IS MADE TO THE LINE DEMULTIPLEXER; OTHERWISE
0153      * A NEW STATMENT LINE IS INTERPRETED.
0154      *
0155      * CALLING SEQUENCE:
0156      *
0157      *      B @RTNY
0158      *
0159      *      EXIT TO MULTIPLEXOR
0160      *
0161      * EXCEPTIONS AND CONDITIONS:
0162      *
0163      *      STACK UNDERFLOW
0164      *
0165      * ENTRY POINT:
0166      *
0167      *      DEF RTNY
0168      *
0169 008A C0E0 RTNY  MOV @GSC,R3 ;SEE IF STACK EMPTY
      008C 0074 '
0170 008E 8803      C R3,@GSS ;EMPTY?
      0090 0000
0171 0092 12F8      JLE ERR12      Y
0172 0094 0643      DECT R3      N, POP PLC
0173 0096 C153      MOV *R3,R5
0174 0098 0643      DECT R3      POP PBC
0175 009A C803      MOV R3,@GSC  UPDATE GSC
      009C 008C '
0176 009E C805      MOV R5,@PLC  UPDATE THE PLC
      00A0 0070 '
0177 00A2 1309      JEQ RTRN3      0 - KEYBOARD MODE WHEN STACKED
0178 00A4 C825      MOV @-2(R5),@SLN UPDATE SLN
      00A6 FFFE
      00A8 0000
0179 00AA C213      MOV *R3,R8 ;GET PBC
0180 00AC 1302      JEQ RTRN2 ;0, GOTO NEXT LINE (ON)
0181 00AE 0460      B @LINE2
      00B0 0000
0182      *
0183 00B2 0460 RTRN2 B @LINE5
      00B4 0000
0184      *
0185 00B6 0460 RTRN3 B @CRLF RETURN TO INTERPRETER LOOP
      00B8 0000

```

```
0187      *
0188      * ABSTRACT:
0189      *
0190      * POP THE PLC AND PBC FROM THE GOSUB STACK AND THROW
0191      * THE VALUES AWAY.
0192      *
0193      * CALLING SEQUENCE:
0194      *
0195      *      B @POPY
0196      *
0197      *      EXIT TO NLINO
0198      *
0199      * EXCEPTIONS AND CONDITIONS:
0200      *
0201      *      STACK UNDERFLOW
0202      *
0203      * ENTRY POINT:
0204      *
0205      *      DEF POPY
0206      *
0207 00BA 8820 POPY C @GSC,@GSS      SOMETHING ON THE STACK ?
      00BC 009C '
      00BE 0090 '
0208 00C0 12E1      JLE ERR12      N, ERROR
0209 00C2 6820      S @C4,@GSC      Y, BACKUP STACK POINTER
      00C4 0000
      00C6 00BC '
0210 00CB 0460      B @NLINO      CONTINUE
      00CA 0000
0211      *
0212      *      END
NO ERRORS,      NO WARNINGS
```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.IF
OBJECT ACCESS NAME= ADHOC.OBJ.IF
LISTING ACCESS NAME= ADHOC.LST.IF
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

```
0002          IDT 'IF'
0003          *
0004          *      IFY          ; IF COMMAND
0005          *
0006          REF EVERZ          ; EVALUATE EXPRESSION
0007          REF EVSDZ          ; EVALUATE STRING
0008          REF LINE,NLIN      ; MULTIPLEXOR ENTRY POINTS
0009          REF ELSF          ; ELSE FLAG
0010          DEF IFY
0011          *
0012          DXOP EVFIX,11      ; EVALUATE AND FIX
0013          2F80 ERROR EQU >2F80 ; XOP XX,14 (ERROR CALL)
0014          2FA0 ERROR2 EQU ERROR+>20
0015          *
0016          *      EXECUTE THE IF COMMAND. THE FORM IS:
0017          *
0018          *      IF <EXP> THEN <STATEMENT>
0019          *      IF <STRING> THEN <STATEMENT>
0020          *      IF <STRING> <RELATION> <STRING> THEN <STATEMENT>
0021          *      IF <STING> <RELATION> <STRING> , <EXP>
0022          *              THEN <STATEMENT>
0023          *
0024          *      CALLING SEQUENCE:
0025          *
0026          *      B @IFY
0027          *
0028          *      EXIT TO LINE OR NLIN
0029          *
0030          *      EXCEPTIONS AND CONDITIONS:
0031          *
0032          *      ILLEGAL DELIMITER, SYNTAX ERROR
0033          *
```

```

0035 0000 04E0 IFY CLR @ELSF ; CLEAR ELSE FLAG
      0002 0000
0036 0004 0420 BLWP @EVSDZ ; CHECK FOR " OR $
      0006 0000
0037 0008 1008 JMP IF2
0038 000A 1007 JMP IF2
0039 000C 0420 BLWP @EVERZ ; £
      000E 0000
0040 0010 C072 MOV *R2+,R1
0041 0012 161A JNE IFRT
0042 0014 C052 MOV *R2,R1
0043 0016 1618 JNE IFRT
0044 0018 102F JMP IF13
0045 *
0046 001A 0280 IF2 CI R0,>3B00 ; THEN?
      001C 3B00
0047 001E 1603 JNE IF4 ; N
0048 *
0049 0020 D052 IF3 MOV B *R2,R1 ; Y, STRING?
0050 0022 1612 JNE IFRT ; Y, CONTINUE
0051 0024 1029 JMP IF13 ; N, GOTO LINE
0052 *
0053 0026 06C0 IF4 SWPB R0
0054 0028 0220 AI R0,->55 ; LEGAL?
      002A FFAB
0055 002C 1502 JGT IF5 ; Y
0056 002E 2FA0 IFE37 DATA ERROR2,37 ; N
0057 *
0058 0032 0280 IF5 CI R0,>6
      0034 0006
0059 0036 15FB JGT IFE37 ; N
0060 0038 C180 MOV R0,R6
0061 003A C0C2 MOV R2,R3 ; STRINGS
0062 003C 0705 SETO R5
0063 003E 0420 BLWP @EVSDZ ; GET SECOND OPERAND
      0040 0006
0064 0042 100B JMP IF7 ; "
0065 0044 100A JMP IF7 ; $
0066 0046 2F8E DATA ERROR+14 ; EXPECTING STRING
0067 *
0068 0048 0460 IFRT B @NLIN
      004A 0000
0069 *
0070 004C D072 IF6 MOV B *R2+,R1 ; LOAD BYTE FROM SECOND STRING
0071 004E 1601 JNE $+4 ; OK
0072 0050 0701 SETO R1 ; NULL
0073 0052 7044 SB R4,R1 ; SAME BYTE?
0074 0054 160B JNE IF10 ; N
0075 0056 0605 DEC R5 ; Y, DONE?
0076 0058 130B JEQ IF11 ; Y
0077 *
0078 005A 0280 IF7 CI R0,>3F00 ; , ?
      005C 3F00
0079 005E 1601 JNE IF8 ; N
0080 0060 2EC5 EVFIX R5 ; Y, GET COUNT
0081 *
0082 0062 0200 IF8 LI R0,4 ; GET MATCH
      0064 0004
0083 0066 D133 MOV B *R3+,R4 ; SOURCE NULL?
0084 0068 16F1 JNE IF6 ; N

```

```
0085      *
0086 006A D072 IF9      MOVB *R2+,R1      ;LOAD BYTE FROM SECOND STRING
0087 006C 1503 IF10     JGT IF12          ;POSITIVE
0088 006E 1601          JNE $+4          ;NEGATIVE
0089 0070 0600 IF11     DEC R0            ;ZERO
0090 0072 0640          DECT R0
0091      *
0092 0074 2180 IF12     CDC R0,R6        ;CONDITION MET?
0093 0076 13E8          JEQ IFRT         ;Y
0094 0078 0720 IF13     SETO @ELSF       ;N
      007A 0002'
0095 007C 0460          B @LINE
      007E 0000
0096      END
NO ERRORS,      NO WARNINGS
```

ACCESS NAMES TABLE

SOURCE ACCESS NAME= ADHOC.SRC.INPUT
OBJECT ACCESS NAME= ADHOC.OBJ.INPUT
LISTING ACCESS NAME= ADHOC.LST.INPUT
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

```
0002          IDT 'INPUT'
0003          *
0004          *      INPY          ; INPUT ROUTINE
0005          *
0006          REF INPEND          ; QUIT INPUT
0007          REF OUTL1          ; OUT LINE
0008          REF CVDIFZ          ; CONVERT DECIMAL TO INTEGER/FP
0009          REF EVARZ          ; EVALUATE VARIABLE
0010          REF EVSDZ          ; EVALUATE STRING
0011          REF GOSB1          ; GOSUB ENTRY
0012          REF BMVE          ; BYTE MOVE
0013          REF JMPROA          ; BYTE MULTIPLEXOR JUMP
0014          REF PUTB          ; PUT BYTE IN BUFFER
0015          REF GETCR$          ; GET CHARACTER
0016          REF TYP0$          ; ECHO R0
0017          REF FFLG          ; FORMATTING FLAG
0018          REF HFLG          ; HELP FLAG
0019          REF IFLG          ; INPUT FLAG
0020          REF NIC,LNSZ          ; IO BUFFER SIZE
0021          REF IOB          ; I/O BUFFER
0022          REF INM1,INM2,INM3    PROMPTS
0023          DEF INPY
0024          *
0025          DXOP STORE,1          ; STORE FPAC
0026          DXOP EVFIX,11        ; EVALUATE AND FIX
```



```

0028      *
0029      *      EXECUTE THE INPUT COMMAND
0030      *
0031      * CALLING SEQUENCE:
0032      *
0033      *      B @INPY
0034      *
0035      *      EXIT TO INPEND, GOSB1
0036      *
0037      *      R7 = OBC
0038      *      R8 = PBC
0039      *      R15 = CR/LF FLAG
0040      *
0041 0000 04E0 INPY CLR @HFLG      ; CLEAR HELP FLAG
      0002 0000
0042 0004 04E0      CLR @IFLG      ; CLEAR INPUT FLAG
      0006 0000
0043 0008 1003      JMP INP1
0044      *
0045      *HELP QUERY SETUP
0046      *
0047 000A 2EE0 INPQM EVFIX @HFLG    ; GET LINE £
      000C 0002
0048      *
0049 000E 0608 INPO DEC R8          ; BACKUP TO DELIMITER
0050      *
0051 0010 070F INP1 SETD R15        ; SET FOR CR
0052      *
0053 0012 058F INPSC INC R15        ; SET FOR NO CRLF
0054 0014 06A0      BL @JMPROA      ; SWITCH BOARD
      0016 0000
0055 0018 4E INPTB BYTE INPE-INPTB/2,>00 ; NULL
0056 001A 4E      BYTE INPE-INPTB/2,>3C ; :
0057 001C 4E      BYTE INPE-INPTB/2,>47 ; !
0058 001E FC      BYTE INP1-INPTB/2,>3F ; ,
0059 0020 FD      BYTE INPSC-INPTB/2,>40 ; .
0060 0022 31      BYTE INPFM-INPTB/2,>3E ; £
0061 0024 F9      BYTE INPQM-INPTB/2,>41 ; ?
0062 0026 2F      BYTE INPFL-INPTB/2,>42 ; %
0063 0028 0000      DATA 0
0064 002A 0608      DEC R8          ; NONE OF ABOVE, TRY STRING
0065 002C 0420      BLWP @EVSDZ
      002E 0000
0066 0030 1027      JMP INPO        ; STRING
0067 0032 1033      JMP INPD        ; $VAR
0068 0034 0420      BLWP @EVARZ      ; NEITHER, TRY VARIABLE
      0036 0000
0069 0038 C182      MOV R2,R6        ; SAVE
0070 003A 0202      LI R2,INM1      ; '?'
      003C 0000
0071      *
0072 003E 06A0 INP3 BL @INPP        ; PROMPT AND GET STRING
      0040 00B8
0073 0042 0420      BLWP @CVDIFZ    ; CONVERT
      0044 0000
0074 0046 100C      JMP INP4        ; FP
0075 0048 04C1      CLR R1          ; NO NUMBER
0076 004A 1000      NOP
0077 004C C000      MOV R0,R0        ; CHECK DELIMITER
0078 004E 160C      JNE INP5        ; PROBLEM

```

0079	0050	04F6	CLR *R6+	; INTEGER
0080	0052	CD81	MOV R1, *R6+	
0081	0054	04F6	CLR *R6+	
0082	0056	04E0	CLR @IFLG	CLEAR % FLAG (EXACT # OF CHARS)
	0058	0006		
0083	005A	04E0	CLR @FFLG	CLEAR NUMBER OF CHARS FLAG
	005C	0000		
0084	005E	10D7	JMP INPO	; LOOK AT DELIMITER

```

0086      *GET NUMERIC INPUT
0087      *
0088 0060 C000  INP4      MOV R0,R0      ;EOL?
0089 0062 1602      JNE INP5      ;N, PROBLEM
0090 0064 2C56      STORE *R6      ;MOVE NUMBER
0091 0066 10F7      JMP INP3A
0092      *
0093      *ERROR IN NUMERIC INPUT
0094      *
0095 0068 0202  INP5      LI R2,INM2      ;OUT '?? '
      006A 0000
0096 006C C060      MOV @HFLG,R1 ;HELP WANTED?
      006E 000C'
0097 0070 13E6      JEQ INP3      ;N
0098 0072 0700      SETO R0      ;YES, RETURN -1
0099 0074 1041      JMP INPP6
0100      *
0101      *% FORMATTING
0102      *
0103 0076 0720  INPFL    SETO @IFLG      ;SET FLAG
      0078 005B'
0104      *
0105      *E FORMATTING
0106      *
0107 007A 2EE0  INPFM    EVFIX @FFLG      ;GET FLAG
      007C 005C'
0108 007E 10C7      JMP INPO
0109      *
0110      *OUT LITERAL
0111      *
0112 0080 020B  INPG     LI R11,INPO      ;GET RETURN ADR
      0082 000E'
0113      *
0114      *OUT STRING
0115      *      R2 = STRING ADR
0116      *
0117 0084 C34B  OUTL     MOV R11,R13      ;SAVE RETURN
0118 0086 C1E0      MOV @IOB,R7      ;SET R7 (WSBC)
      0088 0000
0119 008A 0205      LI R5,LNSZ      ;GET MAX
      008C 0000
0120 008E 06A0      BL @BMVE      ;MOVE TO OUTPUT BUFFER
      0090 0000
0121 0092 1000      NOP
0122 0094 C2CD      MOV R13,R11
0123 0096 0460      B @OUTL1
      0098 0000

```

```
0125      *GET STRING
0126      *
0127 009A C182 INPD      MOV R2,R6      ;SAVE ADR
0128 009C 0202      LI R2,INM3      ;GET ':'
      009E 0000
0129 00A0 06A0      BL @INPP      ;PROMPT AND GET CHARACTERS
      00A2 00B8'
0130 00A4 C087      MOV R7,R2      ;SET FOR MOVE
0131 00A6 C1C6      MOV R6,R7
0132 00AB 0205      LI R5,NIC      ;MAXIMUM OF EIGHTY CHARACTERS
      00AA 0000
0133 00AC 06A0      BL @BMVE      ;MOVE INTO VARIABLE
      00AE 0090'
0134 00B0 10D2      JMP INP3A      ;OK
0135 00B2 10D1      JMP INP3A      ;OK
0136      *
0137      *END INPUT LINE
0138      *
0139 00B4 0460 INPE      B @INPEND
      00B6 0000
```

```

0141      *PROMPT AND GET CHARACTERS
0142      *
0143 00B8 C38B INPP   MOV R11,R14      ;SAVE RETURN
0144 00BA C3CF      MOV R15,R15      ;SUPPRESS PROMPT?
0145 00BC 1602      * JNE INPP1      ;Y
0146 00BE 06A0      BL @OUTL        ;N, LIST
      00C0 00B4'
0147      *
0148 00C2 C1E0 INPP1  MOV @IOB,R7      ;GET IOB (WSBC)
      00C4 0088'
0149 00C6 C120      MOV @FFLG,R4 ;GET INPUT COUNTER
      00C8 007C'
0150 00CA 1602      JNE INPP2      ;OK
0151 00CC 0204      LI R4,>80      ;SET MAX      ??????????????????????
      00CE 0080
0152      *
0153 00D0 0000 INPP2  DATA GETCR$    ;GET CHARACTER
0154 00D2 06A0      BL @PUTB        ;STORE
      00D4 0000
0155 00D6 1005      JMP INPP3      ;CR
0156 00D8 100B      JMP INPP4      ;CONTROL CHARACTER
0157 00DA 0000      DATA TYPO$    ;ECHO
0158 00DC 0604      DEC R4          ;MORE CHARACTERS?
0159 00DE 16F8      JNE INPP2      ;Y
0160 00E0 1003      JMP INPP5      ;N
0161      *
0162 00E2 C020 INPP3  MOV @IFLG,R0 EXACT NUMBER REQUIRED?
      00E4 007B'
0163 00E6 16F4      JNE INPP2      Y - CONTINUE
0164      *
0165 00E8 75D7 INPP5  SB *R7,*R7      ;TERMINATE
0166 00EA C1E0      MOV @IOB,R7      ;RESET BYTE COUNTERS (WSBC)
      00EC 00C4'
0167 00EE 045E      B *R14          ;RETURN
0168      *
0169 00F0 C060 INPP4  MOV @HFLG,R1 ;HELP WANTED?
      00F2 006E'
0170 00F4 13ED      JEG INPP2      ;N, CONTINUE
0171 00F6 06C0      SWPB R0        ;Y
0172      *
0173 00F8 C800 INPP6  MOV R0,@HFLG ;SAVE CONTROL CHARACTER
      00FA 00F2'
0174 00FC 04E0      CLR @FFLG      ;CLEAR FORMATTING
      00FE 00C8'
0175 0100 04E0      CLR @IFLG      CLEAR % FLAG
      0102 00E4'
0176 0104 0460      B @GOSB1      ;DO SYSTEM GOSUB
      0106 0000
0177      *
0178      END
NO ERRORS,      NO WARNINGS

```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.IPCOM
OBJECT ACCESS NAME= ADHOC.OBJ.IPCOM
LISTING ACCESS NAME= ADHOC.LST.IPCOM
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

```
0002          IDT 'IPCOM'
0003      *
0004      * ROUTINE LIST:
0005      *
0006          OUTL1          ;LIST LINE
0007          INPEND        ;END INPUT
0008          PRTEND        ;END PRINT
0009          UCCNT          ;UPDATE CCNT
0010      *
0011      * EXTERNAL ROUTINES:
0012      *
0013          REF  TYPBE$
0014          REF  TYPC$          ;OUTPUT CRLF
0015          REF  NLIN          ;EXIT TO MULTIPLEXOR
0016          REF  IOB          ;I/O BUFFER
0017      *
0018      * EXTERNAL DATA:
0019      *
0020          REF  CCNT          ;COLUMN COUNTER
0021          REF  FFLG          ;FORMATTING FLAG
```

```
0023      * ABSTRACT:
0024      *
0025      *      OUTL1 - UPDATES CCNT AND PRINTS BUFFER.
0026      *
0027      *      PRTEND - UPDATES CCNT, PRINTS BUFFER, AND
0028      *                  AND ENDS WITH A CRLF IF R15=0
0029      *                  BEFORE EXITING TO NLIN.
0030      *
0031      *      INPEND - PRINTS A CRLF IF R15=0 AND
0032      *                  EXITS TO NLIN.
0033      *
0034      *      UCCNT - UPDATES CCNT SUCH THAT IF REFLECTS
0035      *                  WHAT HAS BEEN PRINTED PLUS WHAT
0036      *                  IS IN THE BUFFER AND TERMINATES
0037      *                  BUFFER WITH NULL.
0038      *
0039      * CALLING SEQUENCE:
0040      *
0041      *      BL @OUTL1
0042      *
0043      *      IN  R7 = BUFFER PTR
0044      *      OUT R7 = BUFFER ADR
0045      *
0046      *
0047      *      B @PRTEND
0048      *
0049      *      IN  R7 = BUFFER PTR
0050      *          R15 = CRLF FLAG
0051      *      EXITS TO NLIN
0052      *
0053      *
0054      *      B @INPEND
0055      *
0056      *      IN R15 = CRLF FLAG
0057      *      EXITS TO NLIN
0058      *
0059      *
0060      *      BL @UCCNT
0061      *
0062      *      IN R7 = BUFFER PTR
0063      *
0064      * EXCEPTIONS AND CONDITIONS: (NONE)
0065      *
0066      *      DEF  OUTL1, PRTEND, INPEND, UCCNT
```



```

0068      *
0069 0000 C34B OUTL1 MOV R11,R13      ;SAVE RETURN
0070 0002 06A0      BL @PBIN        ;UPDATE CCNT & OUT BUFFER
      0004 001E'
0071 0006 C1E0      MOV @IOB,R7
      0008 0000
0072 000A 045D      B *R13          ;RETURN
0073      *
0074      *END PRINT LINE
0075      *
0076 000C 06A0 PRTEND BL @PBIN      ;UPDATE CCNT & OUT BUFFER
      000E 001E'
0077      *
0078      *END INPUT LINE
0079      *
0080 0010 04E0 INPEND CLR @FFLG      ;CLEAR FORMATTING FLAG
      0012 0000
0081 0014 C3CF      MOV R15,R15      ;NEED CRLF?
0082 0016 1601      JNE INPE1        ;N
0083 0018 0000      DATA TYP C$     ;Y, OUT CRLF
0084 001A 0460 INPE1 B @NLIN        ;CONTINUE
      001C 0000
0085      *
0086      * PRINT BUFFER IF NEEDED
0087      *
0088 001E 8807 PBIN C R7,@IOB        BUFFER EMPTY
      0020 0008'
0089 0022 1301      JEQ PBIN1        Y, EXIT
0090 0024 0000      DATA TYP B$     N, PRINT BUFFER
0091      0026' PBIN1 EQU $
0092      *
0093      *UPDATE CCNT
0094      *
0095 0026 C020 UCCNT MOV @CCNT,R0      ;GET COUNT
      0028 0000
0096 002A A007      A R7,R0          ;ADD CURRENT PTR
0097 002C 6020      S @IOB,R0        ;SUBTRACT IOB
      002E 0020'
0098 0030 0240      ANDI R0,>7F      ;MOD 128
      0032 007F
0099 0034 C800      MOV R0,@CCNT      ;SAVE RESULT
      0036 0028'
0100 0038 75D7      SB *R7,*R7      ;TERMINATE STRING
0101 003A 045B      RT
0102      END
NO ERRORS,      NO WARNINGS

```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.LET
OBJECT ACCESS NAME= ADHOC.OBJ.LET
LISTING ACCESS NAME= ADHOC.LST.LET
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

```
0002          IDT 'LET'
0003          *
0004          * ROUTINE LIST:
0005          *
0006          *      LETY          ;LET COMMAND
0007          *
0008          REF EVARZ          ;EVALUATE VARIABLE
0009          REF EVERZ          ;EVALUATE EXPRESSION
0010          REF EVSDZ          ;EVALUATE STRING
0011          REF CKEX           ;CHECK FOR EXPRESSION
0012          REF CVDIFZ        ;CONVERT DECIMAL TO INTEGER/FP
0013          REF NLIN           ;EXIT ENTRY TO MULTIPLEXOR
0014          REF FFLG           ;FORMATTING FLAG
0015          REF IOB            ;I/O BUFFER PTR
0016          DXOP LOADF,0       ;LOAD FPAC
0017          DXOP STORE,1       ;STORE FPAC
0018          DXOP SCALE,6       ;SCALE FPAC
0019          DXOP NORMAL,7      ;NORMALIZE FPAC
0020          DXOP EVFIX,11      ;EVALUATE AND FIX
0021          DXOP OUTFP,12      ;OUT FLOATING POINT &
0022          DXOP OUTINT,13     ;OUT INTEGER
0023          2F80 ERROR EQU >2F80 ;XOP XX,14 (ERROR CALL)
```

```
0025      * ABSTRACT:
0026      *
0027      *      THE LET COMMAND IS THE ASSIGNMENT STATEMENT
0028      *      AND CAN HAVE ANY OF THE FOLLOWING FORMS:
0029      *
0030      *      A[-]=B[-]                NUMERIC ASSIGNMENT
0031      *      $A[-]=$B[-]              ASSIGNMENT
0032      *      $A[-]=$B[-],N            PICK
0033      *      $A[-]=$B[-],N            REPLACE
0034      *      $A[-]=$B[-]+$C[0]        CONCATENATE
0035      *      $A[-]=/N                  DELETE
0036      *      $A[-]=/$B[-]              INSERT
0037      *      $A[-]=N                    CONVERT £ TO ASCII
0038      *      $A[-]=£$B[-],N            CONVERT £ TO ASCII FORMATTED
0039      *      $A[-]=%N                  CONVERT BYTE
0040      *      N=$A[-],E                 CONVERT ASCII TO £
0041      *
0042      * CALLING SEQUENCE:
0043      *
0044      *      B @LETY
0045      *
0046      *      EXIT TO NLIN
0047      *
0048      * EXCEPTIONS AND CONDITIONS:
0049      *
0050      *      MISSING ASSIGNMENT OPERATOR
0051      *      EVALUATION ERRORS
0052      *
```

```

0054      * ENTRY POINT:
0055      *
0056      DEF LETY
0057      *
0058 0000 0420 LETY BLWP @EVSDZ
0002 0000
0059 0004 2F81      DATA ERROR+1 ; "
0060 0006 101A      JMP LET3 ; $
0061 0008 0420      BLWP @EVARZ ; EVALUATE VARIABLE
000A 0000
0062 000C 0280      CI R0,>5600 ;=?
000E 5600
0063 0010 1618      JNE ERR36 ;N, ERROR
0064 0012 C0C2      MOV R2,R3
0065 0014 06A0      BL @CKEX ; CHECK FOR EXPRESSION
0016 0000
0066 0018 1007      JMP LET1 ; PROBLEM
0067 001A 0420      BLWP @EVERZ ; EVALUATE
001C 0000
0068      *
0069      * [VAR]=[EXP]
0070      *
0071 001E CCF2      MOV *R2+,*R3+ ; STORE RESULTS
0072 0020 CCF2      MOV *R2+,*R3+
0073 0022 C4D2      MOV *R2,*R3
0074 0024 0460 LETR B @NLIN ; GOTO NEXT LINE
0026 0000
0075      *
0076      * [VAR]=_
0077      *
0078 0028 0420 LET1 BLWP @EVSDZ ; LOOK FOR STRING
002A 0002
0079 002C 1002      JMP LET2 ; "
0080 002E 1001      JMP LET2 ; $
0081 0030 2F81      DATA ERROR+1 ; SYNTAX ERROR
0082      *
0083      * [VAR]=[$VAR]_
0084      *
0085 0032 C102 LET2 MOV R2,R4 ; SAVE ADR
0086 0034 0280      CI R0,>3F00 ;,?
0036 3F00
0087 0038 1353      JEQ LET6 ;Y, CONVERT ASCII TO BINARY
0088      *
0089 003A 2F87 ERR7 DATA ERROR+7 ;N, EXPECTING OPERATER
0090      *
0091      * $ _ STRING ENTRY
0092      *
0093 003C 0280 LET3 CI R0,>5600 ;=?
003E 5600
0094 0040 1309      JEQ LET3A ;Y
0095 0042 2FA0 ERR36 DATA ERROR+>20 ; MISSING ASSIGNMENT OPERATER
0096 0044 0024      DATA 36

```

```

0098      *BYTE ASSIGNMENT
0099      *
0100      *      $A[-]=%N...
0101      *
0102 0046 2EC1 LET11  EVFIX R1      ;GET BYTE
0103 0048 06C1      SWPB R1
0104      *
0105 004A DDC1 LET11A MOV B R1,*R7+ ;SAVE £
0106 004C 0280      CI R0,>4200 ;%?
      004E 4200
0107 0050 13FA      JEQ LET11      ;Y
0108 0052 10E8      JMP LETR      ;RETURN
0109      *
0110      *MULTIPLEXOR FOR STRING ASSIGNMENTS
0111      * $[VAR]=_
0112      *
0113 0054 C1C2 LET3A  MOV R2,R7
0114 0056 0420 LET3E  BLWP @EVSDZ      ;GET STRING OR $VAR
      005B 002A'
      JMP LET4      ;"
      JMP LET4      ;$
0115 005A 101E      MOV B *R8+,R0 ;GET BYTE
0116 005C 101D      CI R0,>4200 ;%?
0117 005E D038
0118 0060 0280      JEQ LET11      ;Y
      0062 4200      CI R0,>5E00 ;/?
0119 0064 13F0      JEQ LET12      ;Y, INSERT
0120 0066 0280      CI R0,>3E00 ;£?
      006B 5E00
0121 006A 134B      JNE LET3C      ;N
0122 006C 0280
      006E 3E00
0123 0070 160B
0124      *
0125      *CONVERT NUMBER TO ASCII
0126      *
0127      *      $A[-]=N
0128      *      $A[-]=£$B[-],N
0129      *
0130 0072 0420      BLWP @EVSDZ      ;Y, GET FORMAT
      0074 005B'
      JMP LET3B
0131 0076 1002      JMP LET3B
0132 0078 1001 ERR14  DATA ERROR+14 ;ERROR, EXPECTING STRING
0133 007A 2F8E      *
0134      *
0135 007C 0280 LET3B  CI R0,>3F00 ;,?
      007E 3F00
      JNE ERR7      ;N, ERROR
0136 0080 16DC      MOV R2,@FFLG ;Y, SET TO FORMAT
0137 0082 C802
      0084 0000
      JMP LET3D
0138 0086 1001
0139      *
0140 0088 0608 LET3C  DEC R8      ;N, BACKUP
0141      *
0142 008A 0420 LET3D  BLWP @EVERZ      ;EVALUATE EXPRESSION
      008C 001C'
0143 008E 2F12      OUTFP *R2      ;CONVERT £
0144 0090 04E0      CLR @FFLG      ;CLEAR FORMATTING
      0092 0084'
0145 0094 75D7      SB *R7,*R7      ;END STRING
0146 0096 10C6      JMP LETR

```

```

0148      *ASSIGN, PICK, REPLACE, CONCATENATE
0149      *
0150      *      $A[-]=$B[-]
0151      *      $A[-]=$B[-],N
0152      *      $A[-]=$B[-];N
0153      *      $A[-]=$B[-]+$C[-]
0154      *
0155 0098 0705 LET4 SET0 R5      ;NO NULL
0156 009A 0280      CI R0,>4000      ;?
      009C 4000
0157 009E 1305      JEQ LET4A      ;Y, REPLACEMENT
0158 00A0 04C5      CLR R5      ;SET TO NULL
0159 00A2 0280      CI R0,>3F00      ;?
      00A4 3F00
0160 00A6 1605      JNE LET4B      ;N, DIRECT ASSIGNMENT
0161 00AB 0585      INC R5
0162      *
0163 00AA 2EC1 LET4A EVFIX R1      ;EVALUATE COUNT
0164 00AC C041      MOV R1,R1      ;CHECK CONDITION
0165 00AE 1505      JGT LET4C      ; >0 THEN DO PICK
0166 00B0 10B9      JMP LETR      ;ELSE IGNORE
0167      *
0168 00B2 C060 LET4B MOV @IOB,R1      ;GET MAX MOVE
      00B4 0000
0169 00B6 6047      S R7,R1
0170 00B8 0601      DEC R1
0171      *
0172 00BA C0C7 LET4C MOV R7,R3      ;MARK
0173      *
0174 00BC D132 LET4D MOV8 *R2+,R4      ;MOVE
0175 00BE 1602      JNE LET4I      ;CHARACTER
0176 00C0 C145      MOV R5,R5      ;NULL, ASSIGNMENT?
0177 00C2 1307      JEQ LET4F      ;Y
0178 00C4 DDC4 LET4I MOV8 R4,*R7+
0179 00C6 81C2      C R2,R7      ;IS S<D?
0180 00CB 1402      JHE LET4E      ;N
0181 00CA 80C2      C R2,R3      ;Y, IS S>=D0?
0182 00CC 1402      JHE LET4F      ;Y, ABORT
0183 00CE 0601 LET4E DEC R1      ;DONE?
0184 00D0 16F5      JNE LET4D      ;N
0185      *
0186 00D2 0915 LET4F SRL R5,1      ;DONE, NEED NULL?
0187 00D4 1601      JNE LET4G      ;N
0188 00D6 75D7      SB *R7,*R7      ;Y
0189      *
0190 00DB 0280 LET4G CI R0,>5D00      ;+?
      00DA 5D00
0191 00DC 1632      JNE LETRR      ;N, RETURN
0192 00DE 10BB      JMP LET3E      ;Y, GET STRING

```

```
0194          *ASCII TO DECIMAL
0195          *
0196          *      N=$A[-1,E
0197          *
0198 00E0 0420 LET6    BLWP @EVARZ      ;GET VARIABLE ADR
      00E2 000A
0199 00E4 04D2          CLR *R2
0200 00E6 C1C4          MOV R4,R7
0201 00E8 0420          BLWP @CVDIFZ    ; CONVERT TO £
      00EA 0000
0202 00EC 1007          JMP LET6A
0203 00EE 04C1          CLR R1          ; NOTHING
0204 00F0 1000          NOP
0205 00F2 D480          MOVB R0,*R2     ; SAVE DELIMITER
0206 00F4 04F3          CLR *R3+       ; MOVE IN RESULTS
0207 00F6 CCC1          MOV R1,*R3+
0208 00FB 04D3          CLR *R3
0209 00FA 1023          JMP LETRR
0210          *
0211 00FC D480 LET6A    MOVB R0,*R2     ; SAVE DELIMITER
0212 00FE 2C53          STORE *R3       ; STORE RESULTS
0213 0100 1020          JMP LETRR       ; RETURN
```



```

0215      *INSERT STRING
0216      *
0217      *      $A[-]=/$B[-]
0218      *      (R7)=DESTINATION
0219      *
0220 0102 0420 LET12  BLWP @EVSDZ      ;LOOK AT NEXT STRING
      0104 0074'
0221 0106 100A      JMP LET13A
0222 0108 1009      JMP LET13A
0223 010A 2EC1      EVFIX R1      ;GET £
0224      *
0225      *DELETE CHARACTERS
0226      *
0227      *      $A[-]=/N
0228      *      (R2)=(R7)=STRING
0229      *      R1=COUNT
0230      *
0231 010C 0601 LET12A DEC R1      ;MOVE BY HOLE
0232 010E 1103      JLT LET12B      ;DONE
0233 0110 D032      MOVB *R2+,R0      ;SKIP 1 CHARACTER
0234 0112 16FC      JNE LET12A      ;CONTINUE
0235 0114 0602      DEC R2      ;EOL
0236      *
0237 0116 DDF2 LET12B MOVB *R2+,*R7+ ;MOVE REST OF STRING
0238 0118 16FE      JNE LET12B
0239 011A 1013      JMP LETRR
0240      *
0241      *INSERT STRING
0242      *
0243 011C C0C2 LET13A MOV R2,R3      ;COUNT INSERT STRING
0244 011E 0701      SETD R1
0245      *
0246 0120 0581 LET13B INC R1      ;COUNT
0247 0122 D033      MOVB *R3+,R0
0248 0124 16FD      JNE LET13B
0249 0126 04C3      CLR R3      ;MOVE AND COUNT TO END OF DES STRING
0250      *
0251 0128 0583 LET13C INC R3      ;COUNT
0252 012A D037      MOVB *R7+,R0
0253 012C 16FD      JNE LET13C
0254      *
0255 012E C107      MOV R7,R4      ;MOVE DES STRING
0256 0130 A101      A R1,R4
0257      *
0258 0132 0607 LET13D DEC R7
0259 0134 0604      DEC R4
0260 0136 D517      MOVB *R7,*R4      ;MOVE CHARACTERS
0261 0138 0603      DEC R3      ;TO HOLE?
0262 013A 15FB      JGT LET13D      ;N
0263      *
0264 013C DDF2 LET13E MOVB *R2+,*R7+ ;Y, INSERT STRING
0265 013E 0601      DEC R1      ;DONE?
0266 0140 15FD      JGT LET13E      ;N
0267 0142 0460 LETRR B @NLIN      ;Y
      0144 0026'
0268      END

```

NO ERRORS,

NO WARNINGS

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.LOGOPS
OBJECT ACCESS NAME= ADHOC.OBJ.LOGOPS
LISTING ACCESS NAME= ADHOC.LST.LOGOPS
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

0002		IDT 'LOGOPS'	
0003	*		
0004	*	LNOTF	; LOGICAL NOT
0005	*	LORF	; LOGICAL OR
0006	*	LXORF	; LOGICAL EXCLUSIVE OR
0007	*	LANDF	; LOGICAL AND
0008	*		
0009		REF FIX	; FIX ARGUMENT
0010		REF EVOP3A	; EXIT ENTRY TO EVALUATOR
0011		REF FPAC,FPAC2	; FLOATING POINT ACCUMULATOR
0012		DXOP CLEAR,B	; CLEAR FPAC

```
0014      *
0015      *      PERFORM LOGICAL OPERATIONS OF
0016      *
0017      *      LOR      = LOGICAL 'OR'
0018      *      LNOT     = LOGICAL 'NOT'
0019      *      LXOR     = LOGICAL EXCLUSIVE 'OR'
0020      *      LAND     = LOGICAL 'AND'
0021      *
0022      * CALLING SEQUENCE:
0023      *
0024      *      B @LNOTF
0025      *      B @LORF
0026      *      B @LXORF
0027      *      B @LANDF
0028      *
0029      *      IN  (R1) = ARG1
0030      *           (R2) = ARG2
0031      *      OUT (R2) = RESULT (1 OR 0)
0032      *      EXIT TO EVOP3A
0033      *
```

```

0035      *
0036      *LOGICAL OPERATORS
0037      *
0038      *      R1 = XXI R4,...
0039      *      R2 = DATA
0040      *      R3 = B *R10
0041      *
0042 0000 C28B LFIX      MOV R11,R10
0043 0002 2E00      CLEAR 0
0044 0004 C101      MOV R1,R4
0045 0006 06A0      BL @FIX
0046      *
0047 000A C084      MOV R4,R2
0048 000C C101      MOV R1,R4      ; STORE OBJECT
0049 000E 06A0      BL @FIX
0050      *
0051 0010 0008      MOV R1,R2
0052 0012 C081      MOV *R10+,R1
0053 0014 C07A      LI R3,>045A      ; B *R10
0054 0016 0203
0055 0018 045A
0056 001A 0441      B R1
0057      *
0058      * ENTRY POINT:
0059      *
0060      *      DEF LORF
0061      *
0062 001C 06A0 LORF      BL @LFIX
0063 001E 0000      *
0064      *      DATA >0264      ; ORI R4,...
0065 0020 0264      JMP LANDF1
0066 0022 1008
0067      *
0068      * ENTRY POINT:
0069      *
0070      *      DEF LXORF
0071      *
0072 0024 06A0 LXORF      BL @LFIX
0073 0026 0000      *
0074      *      DATA >045A      ; B *R10
0075 0028 045A      XOR R2,R4
0076 002A 2902      JMP LANDF1
0077 002C 1003
0078      *
0079      * ENTRY POINT:
0080      *
0081      *      DEF LANDF
0082      *
0083 002E 06A0 LANDF      BL @LFIX
0084 0030 0000      *
0085      *      DATA >0244      ; ANDI R4,...
0086 0032 0244
0087      *
0088 0034 C804 LANDF1      MOV R4,@FPAC2
0089 0036 0000
0090      *
0091 0038 0202 LANDF2      LI R2,FPAC
0092 003A 0000
0093      *
0094 003C 0460      B @EVOP3A
0095 003E 0000
0096      *
0097      * ENTRY POINT:
0098      *
0099      *      DEF LNOTF

```

```
0086      *
0087      *LOGICAL NOT
0088      *
0089 0040 2E00 LNOT    CLEAR 0          ; CLEAR FPAC
0090 0042 C081      MOV R1,R2
0091 0044 06A0      BL @FIX
          0046 0010'
0092 0048 0541      INV R1
0093 004A C801      MOV R1,@FPAC2
          004C 0036'
0094 004E 10F4      JMP LANDF2
0095      *
0096      0041' LNOTF EQU LNOT+1
0097      END
NO ERRORS.      NO WARNINGS
```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.MEMF
OBJECT ACCESS NAME= ADHOC.OBJ.MEMF
LISTING ACCESS NAME= ADHOC.LST.MEMF
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

```

0002          IDT 'MEMF'
0003          *
0004          *      MEMY          ; MEM COMMAND
0005          *      MEMF          ; MEM FUNCTION
0006          *      ASCF          ; ASC FUNCTION
0007          *
0008          REF EVERZ          ; EVALUATE EXPRESSION
0009          REF PFIX          ; FIX 16 BIT NUMBER
0010          REF GPRM2          ; GET PARAMETERS
0011          REF FPAC2          ; FLOATING POINT ACCUMULATOR
0012          REF B4A          ; >4A
0013          REF NLIN          ; EXIT ENTRY TO MULTIPLEXOR
0014          REF EVSFR$          ; EXIT TO EVAL WITH R2 = 'FPAC'
0015          DXOP STORE,1          ; STORE FPAC
0016          DXOP CLEAR,8          ; CLEAR FPAC
0017          DXOP EVFIX,11          ; EVALUATE AND FIX
0018          DXOP OUTFP,12          ; OUT FLOATING POINT &
0019          2F80 ERROR EQU >2F80          ; XOP XX,14 (ERROR CALL)
    
```



```
0021      *
0022      *      THE MEM COMMAND ALLOWS PBASIC TO ALTER
0023      *      A BYTE OF MEMORY WHILE THE MEM FUNCTION
0024      *      ALLOWS PBASIC TO READ A BYTE OF
0025      *      MEMORY.
0026      *
0027      * CALLING SEQUENCE:
0028      *
0029      *      B @MEMY
0030      *
0031      *      EXIT TO NLIN
0032      *
0033      *
0034      *      B @MEMF
0035      *
0036      *      IN (R2) = ADDRESS
0037      *      OUT (R2) = RESULT
0038      *      EXIT TO EVSFR$
0039      *
0040      * EXCEPTIONS AND CONDITIONS:
0041      *
0042      *      EVALUATION ERRORS
0043      *      SYNTAX ERROR
0044      *
```

```

0046                                DEF MEMY
0047                                *
0048 0000 9838 MEMY CB *R8+, @B4A ; LEFT BRACKET?
                                0002 0000
0049 0004 160A JNE ERR1 ; N, ERROR
0050 0006 0420 BLWP @EVERZ ; GET ADDRESS
                                0008 0000
0051 000A 06A0 BL @PFX ; GET INTEGER
                                000C 0000
0052 000E 06A0 BL @GPRM2 ; CHECK PARAMETERS
                                0010 0000
0053 0012 06C3 SWPB R3 ; LEFT JUSTIFY
0054 0014 D443 MOVB R3, *R1 ; STORE BYTE IN MEMORY
0055 0016 0460 B @NLIN
                                0018 0000
0056                                *
0057 001A 2F81 ERR1 DATA ERROR+1
0058                                *
0059 *READ MEMORY
0060                                *
0061 * ENTRY POINT:
0062                                *
0063                                DEF MEMF
0064                                *
0065 001C 06A0 MEMF BL @PFX ; GET PARAMETER
                                001E 000C
0066 0020 D051 MOVB *R1, R1 ; GET BYTE
0067 0022 0981 MEMF1 SRL R1, 8 ; RIGHT JUSTIFY
0068 0024 2E00 CLEAR 0 ; CLEAR FPAC
0069 0026 C801 MOV R1, @FPAC2 ; SAVE
                                0028 0000
0070 002A 0460 B @EVSFR$
                                002C 0000
    
```

```
0072      * ABSTRACT:
0073      *
0074      *      ASC RETURNS THE DECIMAL INTEGER VALUE
0075      *      OF THE FIRST BYTE OF THE ARGUMENT.
0076      *
0077      * CALLING SEQUENCE:
0078      *
0079      *      B @ASCF
0080      *
0081      *      IN (R2) = ADDRESS
0082      *      OUT (R2) = RESULT
0083      *      EXIT TO EVSFR$
0084      *
0085      * EXCEPTIONS AND CONDITIONS: (NONE)
0086      *
0087      * EXTERNAL ROUTINES:
0088      *
0089      *      REF MEMF1      /USE MEMF EXIT
0090      *
0091      * LOCAL DATA: (NONE)
0092      *
0093      * ENTRY POINT:
0094      *
0095      *      DEF ASCF
0096      *
0097      *      002E' ASCF EQU $
0098      *      002E D052 MOVB *R2,R1
0099      *      0030 10F8 JMP MEMF1
0100      *      END
```

NO ERRORS,

NO WARNINGS

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.MOVE
OBJECT ACCESS NAME= ADHOC.OBJ.MOVE
LISTING ACCESS NAME= ADHOC.LST.MOVE
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

```

0002          IDT 'MOVE'
0003          *
0004          *          BMVE          ; BYTE MOVE
0005          *
0006          *          MOVE BYTES FROM (R2) TO (R7) WATCHING FOR
0007          *          BACKSPACES AND HEX CHARACTERS IN
0008          *          ANGLE BRACKETS.  ALSO WATCH FOR
0009          *          BUFFER OVERFLOW.
0010          *
0011          * CALLING SEQUENCE:
0012          *
0013          *          BL @BMVE
0014          *          MOVE OK
0015          *          OVERFLOW
0016          *
0017          *          IN  R2 = SOURCE
0018          *          R5 = MAX COUNT
0019          *          R7 = DESTINATION
0020          *          OUT R3 = -(£ OF <10>)*2
0021          *
0022          DEF BMVE
0023          REF IOB, BOB, B3C
0024          *
0025 0000 04C3  BMVE  CLR R3          ; RESET BS COUNT
0026          *
0027 0002 D5F2  BMVE0  MOVB *R2+, *R7  ; MOVE BYTE
0028 0004 130D          JEQ BMVE2      ; DONE
0029 0006 9817          CB *R7, @B3C   ; "<?"
0030 0008 0000          *
0031 000A 130B          JEQ BMVE3      ; Y
0032 000C 0587  BMVE1  INC R7          ; N
0033 000E C2A0          MOV @IOB, R10  ; GET IOB
0034 0010 0000          *
0035 0012 060A          DEC R10        ; BACKUP TO LAST BYTE
0036 0014 8287          C R7, R10     ; ABOUT TO STORE IN LAST BYTE?
0037 0016 1303          JEQ BMVE6     ; Y, QUIT
0038 0018 0605          DEC R5        ; MORE ROOM?
0039 001A 15F3          JGT BMVE0     ; Y
0040 001C 05CB          INCT R11      ; N, RETURN 2(11)
0041          *
0042 001E 75D7  BMVE6  SB *R7, *R7    ; MARK
0043          *
0044 0020 045B  BMVE2  B   *R11
0045          *
0046 0022 04C1  BMVE3  CLR R1          ; BUILD £ IN R1
0047 0024 C2B2          MOV R2, R10   ; SAVE R2
0048          *
0049 0026 04C0  BMVE4  CLR R0
0050 0028 D03A          MOVB *R10+, R0 ; GET NEXT CHARACTER
0051 002A 0220          AI R0, ->3000
0052 002C D000          *
0053 002E 110E          JLT GHEX2
0054 0030 0280          CI R0, >0900
0055 0032 0900          *
0056 0034 120B          JLE GHEX1
0057 0036 0220          AI R0, ->0700
0058 0038 F900          *
0059 003A 0280          CI R0, >0A00
0060 003C 0A00          *
0061 003E 1106          JLT GHEX2

```

```

0056 0040 0280      CI R0,>0F00
      0042 0F00
0057 0044 1B03      JH GHEX2
0058 0046 0A41  GHEX1  SLA R1,4      ;SHIFT R1
0059 0048 B040      AB R0,R1      ;ADD NEW DIGIT
0060 004A 10ED      JMP BMVE4
0061      *
0062      004C' GHEX2  EQU $
0063 004C 0280  BMVE5  CI R0,>3E00->3700      ;">?
      004E 0700
0064 0050 16DD      JNE BMVE1      ;N, DISREGUARD E
0065 0052 D041      MOV B R1,R1      ;ARE WE GOING TO MOVE IN A NULL?
0066 0054 13DB      JEQ BMVE1      ;Y, IGNORE
0067 0056 C08A      MOV R10,R2      ;Y, SET R2
0068 0058 D5C1      MOV B R1,*R7      ;MOVE INTO STRING
0069 005A 9801      CB R1,@B08      ;BS?
      005C 0000
0070 005E 16D6      JNE BMVE1      ;N
0071 0060 0643      DECT R3      ;Y, COUNT
0072 0062 10D4      JMP BMVE1
0073      END

```

NO ERRORS, NO WARNINGS

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.MWDF
OBJECT ACCESS NAME= ADHOC.OBJ.MWDF
LISTING ACCESS NAME= ADHOC.LST.MWDF
ERROR ACCESS NAME=
OPTIONS= BUNLST, DUNLST, MUNLST
MACRO LIBRARY PATHNAME=

```

0002          IDT 'MWDF'
0003          DEF  MWDY,MWDF,ADRF
0004          *
0005          *      MWDY          MWD COMMAND
0006          *      MWDF          MWD FUNCTION
0007          *      ADRF          ADR FUNCTION
0008          *
0009          REF EVERZ          ;EVALUATE EXPRESSION
0010          REF PFIX          ;FIX 16 BIT NUMBER
0011          REF GPRM2         ;GET PARAMETERS
0012          REF NLIN          ;EXIT ENTRY TO MULTIPLEXOR
0013          REF EVSFR$        ;EXIT ENTRY TO EVALUATOR, LOAD R2 WIT
0014          REF FPAC2         ;FLOATING POINT ACCUMULATOR
0015          REF B4A           ;>4A
0016          *
0017          DXOP STORE,1       ;STORE FPAC
0018          DXOP CLEAR,8      ;CLEAR FPAC
0019          DXOP EVFIX,11      ;EVALUATE AND FIX
0020          DXOP OUTFP,12      ;OUT FLOATING POINT &
0021          2F80 ERROR EQU >2F80 ;XOP XX,14 (ERROR CALL)

```



```

0023      *      THE MWD COMMAND ALLOWS PBASIC TO ALTER
0024      *      A WORD OF MEMORY WHILE THE MEM FUNCTION
0025      *      ALLOWS PBASIC TO READ A WORD OF
0026      *      MEMORY.
0027      *
0028      * CALLING SEQUENCE:
0029      *
0030      *      B @MWDY
0031      *      EXIT TO NLIN
0032      *
0033      *      B @MWDF
0034      *      IN (R2) = ADDRESS
0035      *      OUT (R2) = RESULT
0036      *      EXIT TO EVSFR$
0037      *
0038      * EXCEPTIONS AND CONDITIONS:
0039      *
0040      *      EVALUATION ERRORS
0041      *      SYNTAX ERROR
0042      *
0043 0000 9838 MWDF  CB *R8+,@B4A      ; LEFT BRACKET?
          0002 0000
0044 0004 1609      JNE ERR1      ; N, ERROR
0045 0006 0420      BLWP @EVERZ    ; GET ADDRESS
          0008 0000
0046 000A 06A0      BL @PFX      ; GET INTEGER
          000C 0000
0047 000E 06A0      BL @GPRM2     ; CHECK PARAMETERS
          0010 0000
0048 0012 C443      MOV R3,*R1    ; STORE BYTE IN MEMORY
0049 0014 0460      B @NLIN
          0016 0000
0050      *
0051 0018 2FB1 ERR1  DATA ERROR+1
0052      *
0053      * READ MEMORY
0054      *
0055 001A 06A0 MWDF  BL @PFX      ; GET PARAMETER
          001C 000C
0056 001E C051      MOV *R1,R1    ; GET BYTE
0057 0020 2E00      CLEAR 0        ; CLEAR FPAC
0058 0022 C801 MWDF1 MOV R1,@FPAC2 ; SAVE
          0024 0000
0059 0026 0460      B @EVSFR$     ; RELOAD R2 & EXIT TO EVAL
          0028 0000
0060      *
0061      * ADR - RETURN THE ADDRESS OF THE SPECIFIED VARIABLE/ARRAY
0062      *      ELEMENT.  APPEARS IN THIS MODULE TO SAVE MEMORY.
0063      *
0064 002A C042 ADRF  MOV R2,R1      ARGUMENT'S ADDRESS IN R2
0065 002C 10FA      JMP MWDF1
0066      END
NO ERRORS,      NO WARNINGS

```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC. SRC. NUMBER
OBJECT ACCESS NAME= ADHOC. OBJ. NUMBER
LISTING ACCESS NAME= ADHOC. LST. NUMBER
ERROR ACCESS NAME=
OPTIONS= BUNLST, DUNLST, MUNLST
MACRO LIBRARY PATHNAME=

```

0002          IDT  'NUMBER'
0003          *
0004          DEF  NUMY, D100, D10, C000A
0005          *
0006          REF  MODEOK, AINC, CRLF, LNUM
0007          *
0008          2F80 ERROR EQU  >2F80          USER ERROR
0009          2FA0 ERROR2 EQU  ERROR+>20
0010          DXOP EVFIX, 11
0011          *
0012          * SET AUTO-INCREMENT
0013          *
0014          *
0015 0000 06A0 NUMY BL @MODEOK          ; ABORT IF RUNNING !
          0002 0000
0016 0004 D018          MOVB *R8, R0          ; PARAMETERS
0017 0006 130C          JEQ  NUM2          ; N
0018 0008 2EE0          EVFIX @LNUM          ; Y, GET START
          000A 0000
0019 000C 0280          CI   R0, >3F00          ; , ?
          000E 3F00
0020 0010 160B          JNE  NUM3          ; N
0021 0012 2EC1          EVFIX R1          ; GET AUTO INCREMENT
0022 0014 C801 NUM1 MOV  R1, @AINC          ; SAVE AINC
          0016 0000
0023 0018 6801          S    R1, @LNUM          ; SUBTRACT INC FROM START VALUE
          001A 000A'
0024 001C 0460          B    @CRLF          ; RETURN
          001E 0000
0025 0020 0201 NUM2 LI   R1, 100
          0022 0064
0026 0022' D100 EQU  $-2
0027 0024 C801 MOV  R1, @LNUM          ; SET DEFAULT START &
          0026 001A'
0028 0028 0201 NUM3 LI   R1, 10
          002A 000A
0029 002A' D10 EQU  $-2
0030 002A' C000A EQU  D10
0031 002C 10F3          JMP  NUM1
0032          END
NO ERRORS,          NO WARNINGS

```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.ON
OBJECT ACCESS NAME= ADHOC.OBJ.ON
LISTING ACCESS NAME= ADHOC.LST.ON
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

```
0002          IDT 'ON'
0003      *
0004      * ROUTINE LIST:
0005      *
0006      *      ONY          ; ON COMMAND
0007      *
0008      * EXTERNAL ROUTINES:
0009      *
0010          REF GOSON          ; GOSUB ENTRY
0011          REF LINE          ; EXIT ENTRY TO MULTIPLEXOR
0012          DXOP EVFIX, 11    ; EVALUATE AND FIX
0013      2F80 ERROR EQU >2F80          ; XOP XX, 14 (ERROR CALL)
0014      *
0015      * EXTERNAL DATA:
0016      *
0017          REF B3F          ; >3F
0018      *
0019      * ABSTRACT:
0020      *
0021      *      ON COMMAND
0022      *
0023      * CALLING SEQUENCE:
0024      *
0025      *      B @ONY
0026      *
0027      *      EXIT TO LINE OR GOSON
0028      *
0029      * EXCEPTIONS AND CONDITIONS:
0030      *
0031      *      SYNTAX ERROR
0032      *      EVALUATION ERRORS
0033      *
```

```

0035      *
0036      * ENTRY POINT:
0037      *
0038      DEF ONY
0039 0000 2EC1  ONY  EVFIX R1      ;EVALUATE & FIX
0040 0002 0280      CI R0,>3B00  ;THEN?
0041      0004 3B00
0041 0006 1614      JNE ERR1      ;N, SYNTAX ERROR
0042 0008 04C2  ON1  CLR R2      ;CLEAR DESTINATION
0043 000A D0B8      MOVB *R8+,R2  ;GET TYPE CODE
0044 000C 0972      SRL R2,7      TYPE CODE TO LOW BYTE & *2
0045      *
0046      * TYPE CODE SHOULD BE EITHER 'GOTO' OR 'GOSUB' - IN THE
0047      * MODULE GOSUB, 'GOTO' IS IDENTIFIED BY R3=0 AND 'GOSUB'
0048      * IS IDENTIFIED BY R3=>FFFF
0049      *
0050 000E C0C2      MOV R2,R3      JG 12/1/82
0051 0010 0643      DECT R3      GOTO?      JG 12/1/82
0052 0012 1303      JEQ ON2      JG 12/1/82
0053 0014 0223      AI R3,-3      N - GOSUB?  JG 12/1/82
0054      0016 FFFD
0054 0018 150B      JGT ERR1      N - THEN ERROR  JG 12/1/82
0055      *
0056 001A 0601  ON2  DEC R1      ;COUNT
0057 001C 1105      JLT ON3      ;DONE
0058 001E 1306      JEQ ON4      ;DO GOTO
0059 0020 05C8      INCT R8      ;MOVE TO NEXT
0060 0022 9838      CB *R8+,@B3F ;, ?
0061      0024 0000
0061 0026 13F9      JEQ ON2      ;Y, CONTINUE
0062 0028 0460  ON3  B @LINE     ;DROP THRU
0063      002A 0000
0063      *
0064 002C 0460  ON4  B @GOSON
0065      002E 0000
0065      *
0066 0030 2F81  ERR1 DATA ERROR+1 ;SYNTAX ERROR
0067      END

```

NO ERRORS, NO WARNINGS

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.PRINT
OBJECT ACCESS NAME= ADHOC.OBJ.PRINT
LISTING ACCESS NAME= ADHOC.LST.PRINT
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

```

0002          IDT 'PRINT'
0003          DEF PRTY, HOUT, HOUT$, HTXT
0004          REF B20, B3F, XLOC, YLOC, B4C, B40
0005          REF BMVE, CCNT, FFLG, VMODE, TYP0$, TYP11$
0006          REF CVDIZ, EVERZ, EVSDZ
0007          REF CURFLG, LNSZ, MODE, IOB
0008          REF OUTL1, PRTEND, JMPROA
0009          DXOP EVFIX, 11
0010          DXOP OUTFP, 12
0011          *
0012          2F80 ERROR EQU >2F80
0013          2FA0 ERROR2 EQU ERROR+>20
0014          *
0015          * PRINT @
0016          *
0017 0000 0420 PRTAT BLWP @EVSDZ          ; LOOK FOR STRING
          0002 0000
0018 0004 1028 JMP SCRNG          ; "
0019 0006 1027 JMP SCRNG          ; $
0020 0008 9838 CB *R8+, @B4C          ; (?)
          000A 0000
0021 000C 163E JNE ERR15          ; N
0022 000E 2EC3 EVFIX R3          ; EVALUATE X
0023 0010 2EC1 EVFIX R1          ; EVALUATE Y
0024 0012 0280 CI R0, >4D00          ; ??
          0014 4D00
0025 0016 1639 JNE ERR15          ; N, ERROR
0026 0018 0281 CI R1, 23          ; VALID Y?
          001A 0017
0027 001C 1B36 JH ERR15          ; N
0028 001E 0283 CI R3, 39          ; VALID X?
          0020 0027
0029 0022 1B36 JH ERR15          ; N
0030 0024 C2A0 MOV @VMODE, R10          ; TEXT MODE?
          0026 0000
0031 0028 1305 JEQ UCLO          ; Y, EVERYTHING OK
0032 002A 0283 CI R3, 31          ; N, CHECK VALID X
          002C 001F
0033 002E 1B2D JH ERR15          ; N
0034 0030 0A31 SLA R1, 3          ; OK, FORM Y PIXEL CURSOR
0035 0032 0A33 SLA R3, 3          ; FORM X PIXEL CURSOR
0036 0034 06C1 UCLO SWPB R1          ; PUT IN MSB
0037 0036 06C3 UCLO SWPB R3          ; PUT IN MSB
0038          *
0039          * UPDATE CURSOR LOCATION
0040          *
0041 0038 C020 MOV @CURFLG, R0          ; GET CURSOR FLAG
          003A 0000
0042 003C 1602 JNE UCL1          ; OFF, LEAVE IT
0043 003E 0000 DATA TYP11$, >1D00          ; ON, TURN IT OFF THEN
0044 0042 DB03 UCL1 MOV B R3, @XLOC          ; UPDATE X
          0044 0000
0045 0046 DB01 MOV B R1, @YLOC          ; UPDATE Y
          0048 0000
0046 004A CB00 MOV R0, @CURFLG          ; RESTORE CURSOR FLAG
          004C 003A
0047 004E 1627 JNE PRT2          ; WAS OFF, LEAVE IT
0048 0050 003E DATA TYP11$, >1C00          ; WAS ON, PUT IT BACK ON
0049 0054 1024 JMP PRT2          ; CONTINUE PRINTING
0050          *

```



```

0051      *   SCREEN COMMANDS IN A STRING
0052      *
0053 0056 C0C7  SCRNG  MOV  R7,R3          ;SAVE IOB POINTER
0054 0058 C1C2          MOV  R2,R7          ;SET FOR CONVERSION
0055      *
0056 005A 0201  SCRNG1 LI   R1,1          ;DEFAULT TO 1
          005C 0001
0057 005E 0420          BLWP @CVD1Z        SEE IF THERE IS A NUMBER
          0060 0000
0058 0062 1013          JMP  ERR15         FP - ERROR
0059 0064 0587          INC  R7           NO NUMBER, INC OVER DELIMITER
0060 0066 C141          MOV  R1,R5
0061      *
0062 0068 D000          MOVB R0,R0         DONE ?
0063 006A 130C          JEQ  SCRNG5        Y, EXIT
0064 006C 0202          LI   R2,SCRNG6     GET TABLE ADDRESS
          006E 01B0
0065 0070 C132  SCRNG2 MOV  *R2+,R4       GET CODE
0066 0072 130B          JEQ  ERR15        O, NOT FOUND - ERROR
0067 0074 9100          CB   R0,R4        FOUND ?
0068 0076 16FC          JNE  SCRNG2       N, LOOP
0069 0078 06C4          SWPB R4           Y, POSITION IT
0070 007A D004  SCRNG3 MOVB R4,R0         GET BYTE
0071 007C 0000          DATA TYP0$      OUTPUT IT
0072 007E 0605          DEC  R5           DONE?
0073 0080 16FC          JNE  SCRNG3       N, LOOP
0074 0082 10EB          JMP  SCRNG1       Y, CONTINUE CMD STRING
0075      *
0076 0084 0608  SCRNG5 DEC  R8           BACKUP OVER DELIMITER
0077 0086 C1C3          MOV  R3,R7        RESTORE IOB POINTER
0078 0088 100A          JMP  PRT2         ;CONTINUE PRINTING
0079 008A 2F8F  ERR15  DATA ERROR+15    ;INVALID SCREEN COMMAND
    
```

```

0081      *
0082      *PRINT COMMAND
0083      *
0084 008C 020E PRTY   LI R14,LNSZ      ;GET LINE SIZE
          008E 0000
0085 0090 C020      MOV @MODE,R0 ;LOOK AT MODE
          0092 0000
0086 0094 1601      JNE PRTO        ;RUN
0087 0096 093E      SRL R14,3        ;IDLE, LOAD COMMA SIZE
0088      *
0089 0098 C1E0 PRT0   MOV @IOB,R7      ;SET R7 (WSBC)
          009A 0000
0090      *
0091 009C 070F PRT1   SET0 R15         ;SET FOR CRLF
0092      *
0093 009E 058F PRT2   INC R15
0094 00A0 06A0      BL @JMPROA        ;SWITCH BOARD
          00A2 0000
0095 00A4 1B PRTTB   BYTE PRTE-PRTTB/2,>00 ;NULL
0096 00A6 1B        BYTE PRTE-PRTTB/2,>3C   ;:
0097 00AB 1B        BYTE PRTE-PRTTB/2,>47   ;!
0098 00AA 1A        BYTE PRTF-PRTTB/2,>3E   ;£
0099 00AC 49        BYTE PRTC-PRTTB/2,>3F   ;,
0100 00AE 51        BYTE PRTSC-PRTTB/2,>40  ;.
0101 00B0 5B        BYTE PRTAB-PRTTB/2,>39  ;TAB
0102 00B2 AE        BYTE PRTAT-PRTTB/2,>3D  ;@
0103 00B4 0000      DATA 0
0104 00B6 060B      DEC R8            ;TRY STRINGS
0105 00B8 0420      BLWP @EVSdz
          00BA 0002
0106 00BC 102D      JMP PRTG          ;"
0107 00BE 102C      JMP PRTG          ;$
0108 00C0 0420      BLWP @EVERZ       ;NEITHER, TRY NUMBER
          00C2 0000
0109 00C4 C120      MOV @FFLG,R4 ;FORMATTING?
          00C6 0000
0110 00C8 1602      JNE $+6          ;Y, IGNOR SPACE
0111 00CA DDE0      MOV @B20,*R7+    ;N, OUT SPACE
          00CC 0000
0112 00CE 2F12      OUTFP *R2        ;OUTPUT £
0113      *
0114 00D0 060B PRT4   DEC R8          ;BACKUP TO DELIMITER
0115 00D2 10E4      JMP PRT1
0116      *
0117 00D4 0460 PRTE   B @PRTEND      ;END PRINT
          00D6 0000
  
```

```

0119      *PRINT FORMATTING
0120      *
0121 00D8 0420 PRTF      BLWP @EVSZDZ      ;EVALUATE STRING
          00DA 00BA'
0122 00DC 101A      JMP PRTF1      ;"
0123 00DE 1019      JMP PRTF1      ;$
0124 00E0 0202      LI R2,5      ;MAYBE HEX
          00E2 0005
0125 00E4 9818      CB *R8,@B3F      ;£,?
          00E6 0000
0126 00E8 1603      JNE PRTH1      ;N
0127 00EA 0588      INC R8
0128 00EC 2EC1      EVFIX R1      ;Y, GET £
0129 00EE 1007      JMP PRTH2
0130      *
0131 00F0 9818 PRTH1    CB *R8,@B40      ;£,?
          00F2 0000
0132 00F4 1608      JNE PRTH3      ;N
0133 00F6 0588      INC R8
0134 00F8 2EC1      EVFIX R1      ;Y, GET £
0135 00FA 0642      DECT R2
0136 00FC 0A81      SLA R1,8      ;LEFT JUSTIFY
0137      *
0138 00FE 06A0 PRTH2    BL @HOUTE      ;OUT £
          0100 01A4'
0139 0102 0607      DEC R7      ;BACKUP ON "H
0140 0104 10E5      JMP PRT4
0141      *
0142 0106 2EC1 PRTH3    EVFIX R1      ;FORMAT FREE
0143 0108 DDE0      MOVW @B20,*R7+      ;OUT SPACE
          010A 00CC'
0144 010C 06A0      BL @HOUT      ;OUT £
          010E 018A'
0145 0110 10DF      JMP PRT4
0146      *
0147 0112 C802 PRTF1    MOV R2,@FFLG      ;SET FLAG
          0114 00C6'
0148 0116 10DC      JMP PRT4
0149      *
0150 0118 C14E PRTG     MOV R14,R5      ;GET NUMBER OF BYTES AVAILABLE
0151 011A A160      A @IOB,R5
          011C 009A'
0152 011E 6147      S R7,R5
0153 0120 06A0      BL @BMVE      ;MOVE INTO OUTPUT STRING
          0122 0000
0154 0124 1005      JMP PRTG1      ;OK
0155 0126 06A0      BL @OUTL1      ;LIST LINE
          0128 0000
0156 012A A803      A R3,@CCNT      ;UPDATE BACKSPACES
          012C 0000
0157 012E 10F4      JMP PRTG
0158      *
0159 0130 A803 PRTG1    A R3,@CCNT      ;UPDATE COLUMN COUNT
          0132 012C'
0160 0134 10CD      JMP PRT4
0161      *
0162      *COMMA
0163      *
0164 0136 06A0 PRTC     BL @PRTCK      ;GET CURRENT £ OF BYTES
          0138 017A'
    
```

```
0165 013A 0240      ANDI R0,>7      ;MOD 8
      013C 0007
0166 013E 0220      AI R0,-8        ;GET -£ OF BLANKS
      0140 FFF8
0167 0142 06A0      BL @PRTSP       ;OUT SPACES
      0144 0170'
0168                *
0169                *SEMI-COLON OPERATER
0170                *
0171 0146 06A0 PRTSC BL @PRTCK      ;CHECK BUFFER OVERFLOW
      0148 017A'
0172 014A 600E      S R14,R0        ;EXCEEDED?
0173 014C 11A8      JLT PRT2        ;N
0174 014E 06A0      BL @OUTL1       ;Y, PRINT LINE
      0150 0128'
0175 0152 10A5      JMP PRT2
```

```

0177          *TAB FUNCTION
0178          *
0179 0154 2EC1 PRTAB EVFIX R1          ;GET TAB
0180 0156 0608          DEC R8          ;BACKUP OVER DELIMITER
0181 0158 0241          ANDI R1,>7F     ;LIMIT TABS TO 127
          015A 007F
0182 015C 06A0          BL @PRTCK       ;GET CURRENT CHARACTER POSITION
          015E 017A'
0183 0160 6001          S R1,R0         ;NEED SPACES?
0184 0162 159D          JGT PRT2        ;N, PAST POINT
0185 0164 C060          MOV @MODE,R1     ;CHECK MODE
          0166 0092'
0186 0168 139A          JEQ PRT2        ;KEYBOARD MODE, IGNORE
0187 016A 06A0          BL @PRTSP       ;Y, OUT SPACES
          016C 0170'
0188 016E 1097          JMP PRT2
0189          *
0190          *OUT -RO SPACES
0191          *
0192 0170 0580 PRTSP INC R0             ;DONE?
0193 0172 1508          JGT PRTCK1      ;Y
0194 0174 DDE0          MOVB @B20,*R7+ ;N, OUT SPACE
          0176 010A'
0195 0178 10FB          JMP PRTSP
0196          *
0197 017A C020 PRTCK MOV @CCNT,R0 ;GET LINE SIZE
          017C 0132'
0198 017E A007          A R7,R0
0199 0180 6020          S @IOB,R0       ;GET CURRENT CHARACTER POSITION
          0182 011C'
0200 0184 045B PRTCK1 RT
  
```

```

0202          *PRINT HEX
0203          *
0204 0186 0700 HOUT$   SET0 R0           ; SET ZERO FLAG
0205 0188 1001          JMP HOUT$1
0206 018A 04C0 HOUT    CLR R0           ; RESET ZERO FLAG
0207 018C DDE0 HOUT$1  MOV0 @HTXT,*R7+ ; OUT "0
          018E 01C0'
0208 0190 0202          LI R2,4         ; DO 4 DIGITS
          0192 0004
0209          *
0210 0194 C101 HOUT1   MOV R1,R4
0211 0196 0A41          SLA R1,4         ; ISOLATE DIGIT
0212 0198 09C4          SRL R4,12
0213 019A 1602          JNE HOUT2        ; NON-ZERO
0214 019C C000          MOV R0,R0        ; FLAG SET?
0215 019E 1303          JEQ HOUT3        ; N
0216 01A0 DDE4 HOUT2   MOV0 @HTXT(4),*R7+
          01A2 01C0'
0217          *
0218 01A4 0700 HOUTE   SET0 R0           ; SET FLAG
0219          *
0220 01A6 0602 HOUT3   DEC R2           ; DONE?
0221 01A8 15F5          JGT HOUT1        ; N
0222 01AA DDE0          MOV0 @B48,*R7+   ; Y, OUT "H
          01AC 01BC'
0223 01AE 045B          RT
0224          *
0225 01B0          EVEN
0226 01B0 55  SCRNQ6  BYTE 'U',>08      UP
0227 01B2 44          BYTE 'D',>0A      DOWN
0228 01B4 4C          BYTE 'L',>08      LEFT
0229 01B6 52          BYTE 'R',>09      RIGHT
0230 01B8 42          BYTE 'B',>0D      BEGINNING
0231 01BA 43          BYTE 'C',>0C      CLEAR SCREEN
0232 01BC 48  B48     BYTE 'H',>1E      HOME
0233 01BE 0000          DATA 0
0234          *
0235 01C0 30  HTXT     TEXT '0123456789ABCDEF'
          01C1 31
          01C2 32
          01C3 33
          01C4 34
          01C5 35
          01C6 36
          01C7 37
          01C8 38
          01C9 39
          01CA 41
          01CB 42
          01CC 43
          01CD 44
          01CE 45
          01CF 46
0236          *B40     BYTE >40
0237          *B48     BYTE >48
0238 01D0          EVEN
0239          END
    
```

NO ERRORS,

NO WARNINGS

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.PURGE
OBJECT ACCESS NAME= ADHOC.OBJ.PURGE
LISTING ACCESS NAME= ADHOC.LST.PURGE
ERROR ACCESS NAME=
OPTIONS= BUNLST, DUNLST, MUNLST
MACRO LIBRARY PATHNAME=

```

0002          IDT  'PURGE'
0003          *
0004          *
0005          *
0006          DEF  PURGY
0007          REF  EMV,PLC,PRDY,SLN,SLT,VNT
0008          REF  GETC$,MODEOK,ESCFLG
0009          DXOP EVFIX,11
0010          *
0011          2F80  ERROR  EQU  >2F80
0012          2FA0  ERROR2 EQU  ERROR+>20
0013          *
0014  0000  06A0  PURGY  BL   @MODEOK          ABORT IF RUNNING
0015          0002  0000
0016          0004  C0A0          MOV  @PLC,R2          SAVE CURRENT LINE + 1
0017          0006  0000
0018          0008  1303          JEQ  PRG0
0019          000A  C822          MOV  @-6(2),@SLN
0020          000C  FFFA
0021          000E  0000
0022  0010  2EC1  PRG0   EVFIX R1          GET START LINE &
0023  0012  0280          CI    R0,>3800          'TO' ?
0024  0014  3800
0025  0016  161C          JNE  ERR37          N, ILLEGAL DELIMITER
0026  0018  2EC3          EVFIX R3          GET STOP LINE &
0027  001A  C3C3          MOV  R3,R15          SAVE END
0028  001C  C3A0  PRG2   MOV  @VNT,R14          GET TABLE ADR
0029  001E  0000
0030  0020  064E          DECT  R14          BACKUP
0031  0022  880E  PRG1   C     R14,@SLT          DONE?
0032  0024  0000
0033  0026  1212          JLE  PRG3          Y
0034  0028  022E          AI    R14,-4          N, MOVE TO NEXT ENTRY
0035  002A  FFFC
0036  002C  87B1          C     R1,*R14          SAME ?
0037  002E  15F9          JGT  PRG1          N, CONTINUE
0038  0030  880E          C     R14,@SLT          DONE?
0039  0032  0024'
0040  0034  1A0B          JL    PRG3          Y
0041  0036  83DE          C     *R14,R15          REACHED END?
0042  0038  1509          JGT  PRG3          Y
0043  003A  04C0          CLR  R0
0044  003C  C05E          MOV  *R14,R1          GET LINE &
0045  003E  0720          SETO @ESCFLG          DISABLE ESCAPE KEY
0046  0040  0000
0047  0042  06A0          BL    @EMV          DELETE LINE
0048  0044  0000
0049  0046  0000          DATA GETC$          CHARACTER ?
0050  0048  10E9          JMP  PRG2          N, CONTINUE
0051  004A  10EB          JMP  PRG2          Y, NORMAL CHARACTER - IGNORE
0052  004C  0460  PRG3   B     @PRDY          ESCAPE, EXIT
0053  004E  0000
0054          *
0055  0050  2FA0  ERR37  DATA ERROR2,37          ILLEGAL DELIMITER
0056          0044          END

```

NO ERRORS, NO WARNINGS

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.PUTB
OBJECT ACCESS NAME= ADHOC.OBJ.PUTB
LISTING ACCESS NAME= ADHOC.LST.PUTB
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

```
0002          IDT 'PUTB'
0003          REF UFT          ;USER FUNCTION TABLE
0004          REF CCNT         ;COLUMN COUNTER
0005          REF IOB          ;IO BUFFER
0006          REF TYP11$       ;OUTPUT IMMEDIATE BYTE
0007      *
0008      *          PUTB TRANSFERS LEFT BYTE IN R0 INTO
0009      *          BUFFER (R7). A CHECK IS MADE FOR
0010      *          CARRIAGE RETURN AND CONTROL CHARACTERS.
0011      *
0012      * CALLING SEQUENCE:
0013      *
0014      *          BL @PUTB
0015      *          CARRIAGE RETURN
0016      *          CONTROL CHARACTER
0017      *          CHARACTER
0018      *
0019      *          IN (R7) = BUFFER
0020      *          R0 = CHARACTER
0021      *          OUT R0 = CHARACTER OR BACKSPACE OR BELL
```

```

0023      *
0024      * ENTRY POINT:
0025      *
0026      DEF  PUTB
0027      *
0028 0000 0240  PUTB  ANDI  R0,>FF00      ; REMOVE LS BYTE
      0002 FF00
0029 0004 0280      CI   R0,>0D00      ; CR?
      0006 0D00
0030 0008 1602      JNE  PUTB1          ; N
0031 000A 75D7      SB   *R7,*R7        ; Y, NULL TERMINATE
0032 000C 045B      RT                    ; EXIT
0033      *
0034 000E 0280  PUTB1 CI   R0,>2000      ; CONTROL ?
      0010 2000
0035 0012 1402      JHE  PUTB2          ; N
0036 0014 046B      B    @2(11)        ; Y, EXIT
      0016 0002
0037      *
0038 0018 0280  PUTB2 CI   R0,>7F00      ; RUBOUT?
      001A 7F00
0039 001C 1607      JNE  PUTB4          ; N
0040 001E 0584      INC  R4              ; Y, ALLOW 1 MORE CHARACTER
0041 0020 8807      C    R7,@I0B        ; RUBOUT, EMPTY?
      0022 0000
0042 0024 1B08      JH   PUTB5          ; N
0043 0026 0200  PUTB3 LI   R0,>0700      ; Y, OUT BELL
      0028 0700
0044 002A 100F      JMP  PUTB6          ; EXIT
0045      *
0046 002C 8807  PUTB4 C    R7,@UFT      ; BUFFER FULL?
      002E 0000
0047 0030 14FA      JHE  PUTB3          ; Y
0048 0032 DDC0      MOVB R0,*R7+        ; N, STORE BYTE
0049 0034 100A      JMP  PUTB6          ; EXIT
0050      *
0051 0036 0607  PUTB5 DEC  R7            ; BACKUP
0052 0038 0584      INC  R4              ; ALLOW CHARACTER
0053 003A 0620      DEC  @CCNT          ; ADJUST BACKSPACE COUNT
      003C 0000
0054 003E 0000      DATA TYP11$,>0800 ; OUT BS
0055 0042 003E'     DATA TYP11$,>2000 ; OUT SP
0056 0046 0200      LI   R0,>0800      ; OUT BS
      0048 0800
0057 004A 046B  PUTB6 B    @4(11)      ; EXIT
      004C 0004
0058      END
NO ERRORS,      NO WARNINGS

```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.RANDOM
OBJECT ACCESS NAME= ADHOC.OBJ.RANDOM
LISTING ACCESS NAME= ADHOC.LST.RANDOM
ERROR ACCESS NAME=
OPTIONS= BUNLST, DUNLST, MUNLST
MACRO LIBRARY PATHNAME=

```
0002          IDT 'RANDOM'
0003          *
0004          *
0005          *
0006          REF NLIN          ;EXIT TO MULTIPLEXOR
0007          DXOP EVFIX,11     ;EVALUATE AND FIX
0008          REF RANDS        ;RANDOM NUMBER SEED
0009          *
0010          DEF RANY
0011          * ABSTRACT:
0012          *
0013          *     THE RANDOM COMMAND WILL SET THE RANDOM
0014          *     SEED.
0015          *
0016          * CALLING SEQUENCE:
0017          *
0018          *     B @RANY
0019          *
0020          *     EXIT TO NLIN
0021          *
0022          * EXCEPTIONS AND CONDITIONS: (NONE)
0023          *
0024          *
0025 0000 2EE0 RANY EVFIX @RANDS          ; GET RANDOM SEED
          0002 0000
0026 0004 0460 B @NLIN
          0006 0000
0027          END
NO ERRORS,      NO WARNINGS
```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.READ
OBJECT ACCESS NAME= ADHOC.OBJ.READ
LISTING ACCESS NAME= ADHOC.LST.READ
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

```

0002          IDT 'READ'
0003          *
0004          *
0005          *
0006          REF BMVE          ; MOVE BYTE
0007          REF CKEX          ; CHECK FOR EXPRESSION
0008          REF EVARZ          ; GET VARIABLE
0009          REF EVERZ          ; EVALUATE EXPRESSION
0010          REF EVSDZ          ; EVALUATE STRING
0011          REF NLIN           ; EXIT TO MULTIPLEXOR
0012          REF NLINO          ; EXIT TO MULTIPLEXOR
0013          REF REST           ; RESTORE DATA PTR
0014          REF SFSN           ; SEARCH FOR STATEMENT NUMBER
0015          DXOP EVFIX,11      ; EVALUATE AND FIX
0016          2F80 ERROR EQU >2F80 ; XOP XX,14 (ERROR CALL)
0017          *
0018          REF BUS            ; BEGINNING OF USER STORAGE
0019          REF DATXB          ; DATA INTERNAL BYTE CODE
0020          REF DDM            ; DATA DELIMITER
0021          REF DLC            ; DATA LINE COUNTER
0022          REF DLIM           ; LINE DELIMITER
0023          REF SLT            ; STATEMENT LOCATION TABLE
0024          REF LNSZ
0025          DEF RDDY,RNWy
0026          *
0027          * ABSTRACT:
0028          *
0029          * READ COMMAND USES FDATA TO PICK
0030          * UP NEXT ITEM FROM DATA STATEMENTS.
0031          * SINCE THE EVALUATOR IS USED, ANY
0032          * EXPRESSION CAN APPEAR IN THE DATA
0033          * STATEMENT.
0034          *
0035          * CALLING SEQUENCE:
0036          *
0037          * B @RDDY
0038          * B @RNWy
0039          *
0040          * EXIT TO NLIN
0041          *
0042          * EXCEPTIONS AND CONDITIONS:
0043          *
0044          * ERROR 24 FOR READING INTO STRING CONSTANT
0045          * ERROR 23 FOR READ WITHOUT DATA

```

```

0047      *FIND DATA
0048      *      BL @FDATA
0049      *      STRING
0050      *      £
0051      *
0052 0000 C188 FDATA MOV R8,R6      ;SAVE PBC
0053 0002 C160      MOV @DLIM,R5    ;SAVE DLIM
      0004 0000
0054 0006 C220      MOV @DDM,R8     ;GET DELIMITER
      0008 0000
0055 000A 1307      JEQ FDATA1      ;MOVE TO NEXT LINE
0056 000C 04C0      CLR R0
0057 000E D038      MOVB *R8+,R0    ;LOOK AT DELIMITER
0058 0010 1304      JEQ FDATA1      ;EOF, MOVE TO NEXT
0059 0012 0280      CI R0,>3F00     ;.?
      0014 3F00
0060 0016 1310      JEQ FDATA2      ;Y, EVALUATE
0061      *
0062 0018 2F97 ERR23 DATA ERROR+23 ;READ W/O DATA
0063      *
0064 001A C220 FDATA1 MOV @DLC,R8   ;GET DLC
      001C 0000
0065 001E 0228      AI R8,-4        ;MOVE TO NEXT LINE
      0020 FFFC
0066 0022 8808      C R8,@SLT      ;ANY MORE LINES?
      0024 0000
0067 0026 1AF8      JL ERR23        ;N, ERROR
0068 0028 C808      MOV R8,@DLC     ;Y, SAVE DLC
      002A 001C
0069 002C C218      MOV *R8,R8     ;GET PBC
0070 002E A220      A @BUS,R8       ;MAKE DISPLACEMENT INTO POINTER
      0030 0000
0071 0032 9838      CB *R8+,@DATXB ;DATA STATEMENT?
      0034 0000
0072 0036 16F1      JNE FDATA1      ;N, CONTINUE LOOKING
0073      *
0074 0038 0420 FDATA2 BLWP @EVSdz   ;EVALUATE
      003A 0000
0075 003C 1004      JMP FDATA3      ;"
0076 003E 1003      JMP FDATA3      ;$
0077 0040 0420      BLWP @EVERZ     ;£
      0042 0000
0078 0044 05CB      INCT R11        ;RETURN 2(11)
0079      *
0080 0046 0608 FDATA3 DEC R8        ;BACKUP TO DELIMITER
0081 0048 C808      MOV R8,@DDM     ;SAVE IN DELIMITER PTR
      004A 0008
0082 004C C206      MOV R6,R8       ;RESTORE R8
0083 004E C805      MOV R5,@DLIM    ;RESTORE DLIM
      0050 0004
0084 0052 045B      RT

```



```

0086      *READ COMMAND
0087      *
0088 0054 0420 RDDY BLWP @EVSDZ ; SEE IF STRING
      0056 003A'
0089 0058 2F98 ERR24 DATA ERROR+24 ; ", ERROR
0090 005A 1010 JMP RDD2 ; $
0091 005C 0420 BLWP @EVARZ ; WANTS £
      005E 0000
0092 0060 C1C2 MOV R2,R7 ; SAVE
0093 0062 06A0 BL @FDATA ; FIND DATA
      0064 0000'
0094 0066 10F8 JMP ERR24 ; STRING, ERROR
0095 0068 CDF2 MOV *R2+,*R7+ ; STORE NUMBER
0096 006A CDF2 MOV *R2+,*R7+
0097 006C C5D2 MOV *R2,*R7
0098      *
0099 006E C020 RDD1 MOV @DLIM,R0 ; LOOK AT DELIMITER
      0070 0050'
0100 0072 0280 CI R0,>3F00 ; , ?
      0074 3F00
0101 0076 13EE JEQ RDDY ; Y, DO AGAIN
0102 0078 0460 B @NLIN ; N
      007A 0000
0103      *
0104 007C C1C2 RDD2 MOV R2,R7 ; SAVE ADR
0105 007E 06A0 BL @FDATA ; LOOKING FOR STRING DATA
      0080 0000'
0106 0082 1001 JMP RDD3 ; FOUND
0107 0084 10E9 JMP ERR24
0108      *
0109 0086 0205 RDD3 LI R5,LNSZ ; MAX £ OF CHARACTERS
      0088 0000
0110 008A 06A0 BL @BMVE ; MOVE FROM *R2 TO *R7
      008C 0000
0111 008E 10EF JMP RDD1
  
```

```
0113      *
0114      *RESTOR COMMAND
0115      *
0116 0090 06A0 RNWY BL @REST ; RESTORE TO PROGRAM BEGINNING
      0092 0000
0117 0094 06A0 BL @CKEX ; CHECK FOR EXPRESSION
      0096 0000
0118 0098 100A JMP RNWY1 ; NONE
0119 009A 2EC1 EVFIX R1 ; FIX PARAMETER
0120 009C C0B8 MOV R8,R2 SAVE PBC
0121 009E 06A0 BL @SFSN ; LOOK FOR STATEMENT £
      00A0 0000
0122 00A2 8E38 C *R8+,*R8+ ; MOVE BACK 1 LINE
0123 00A4 C808 MOV R8,@DLC ; SAVE PLC
      00A6 002A'
0124 00AB C202 MOV R2,R8 RESTORE PBC
0125 00AA 0460 B @NLIN
      00AC 007A'
0126 00AE 0460 RNWY1 B @NLINO
      00B0 0000
0127      END
NO ERRORS, NO WARNINGS
```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.RENUM
OBJECT ACCESS NAME= ADHOC.OBJ.RENUM
LISTING ACCESS NAME= ADHOC.LST.RENUM
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

```

0002          IDT 'RENUM'
0003      *
0004      *
0005      *
0006      DEF  RENUMS,NINE
0007      *
0008      REF  TYPB$,CRLF,EDIT1,LSTL
0009      REF  MOVEBN,MOVELN
0010      REF  NVD,NVS,BUS,VNT,SLT,DCNT,IOB
0011      REF  ESCFLG,RENCMP,RENLINE,RENRET
0012      REF  RNGOTO,RNGSUB,RNRSTR,RNON,CKEX
0013      REF  RNINP,RNIF,RNELSE,RNERR,RNREM
0014      *
0015      REF  RENSTA,RENINC          START & INCREMENT HOLDERS
0016      REF  MODEOK
0017      *
0018      DXOP OUTINT,13
0019      DXOP EVFIX,11
0020      *
0021      2F80 ERROR EQU >2F80
0022      *
0023      * RENUMBER COMMAND
0024      * FORM :                      NEW LINES :-
0025      *
0026      * RENUM                      FROM 10 IN STEPS OF 10
0027      * RENUM <ARG1>                FROM ARG1 IN STEPS OF 10
0028      * RENUM STEP <ARG1>           FROM 10 IN ARG1 STEPS
0029      * RENUM <ARG1> STEP <ARG2>    FROM ARG1 IN ARG2 STEPS
0030      *
0031      *
0032      * REPLACES ALL REFERENCES TO STATEMENTS IN
0033      *      GOTO
0034      *      GOSUB
0035      *      RESTOR
0036      *      ON
0037      *      INPUT
0038      *      ERROR
0039      * WITH THE NEW STATEMENT NUMBER
0040      *
0041      * METHOD:
0042      * 1. GO THROUGH PROGRAM AND BUILD UP A TABLE OF ALL
0043      *    STATEMENTS CONTAINING REFERENCES TO OTHER STATEMENTS.
0044      *    TABLE CONTAINS STATEMENT NUMBER REFERENCED AND ADDRESS
0045      *    IN PSEUDO CODE OF THIS REFERENCE.
0046      * 2. GO THROUGH PROGRAM AGAIN AND CHANGE OLD STATEMENT
0047      *    NUMBERS TO NEW. CHECK EACH OLD STATEMENT NUMBER
0048      *    AGAINST ALL NUMBERS IN THE TABLE BUILT UP IN STEP 1.
0049      *    IF A MATCH IS FOUND, CHANGE THE STATEMENT REFERENCE
0050      *    AT THE ADDRESS GIVEN IN THE TABLE FROM OLD TO NEW
0051      *    STATEMENT NUMBER. THERE MAY BE MORE THAN ONE
0052      *    ENTRY IN THE TABLE FOR EACH OLD STATEMENT NUMBER.
0053      *
0054      * DESIGN CONSIDERATIONS:
0055      * THIS PROGRAM IS NOT OPTIMISED FOR SPEED, AS IT IS NOT
0056      * AN EXECUTION TIME ROUTINE. IT IS DESIGNED TO USE THE
0057      * MINIMUM DATA STORAGE CONSISTENT WITH GOOD PROGRAM
0058      * STRUCTURE.
0059      * DATA AREA FOR THE TABLE IS TAKEN FROM THE TOP OF THE
0060      * VARIABLE DEFINITION TABLE (@NVD+2) TO (MAXIMALLY)
0061      * THE BOTTOM OF THE VARIABLE STORAGE AREA (@NVS-2).
  
```

```

0062      *
0063      * ERRORS:
0064      *   IF THE PROGRAM RUNS OUT OF MEMORY WHILE TRYING TO
0065      *   BUILD THE TABLE (STEP 1), THE POWER BASIC SYSTEM
0066      *   RETURNS AN ERROR 10 - STORAGE OVERFLOW
0067      *
0068      * NOTE:
0069      *   BEFORE RUNNING THE RENUM ROUTINE IT IS FIRST NECESSARY
0070      *   TO MODIFY THE WAY VALUES 1 TO 9 ARE STORED INTERNALLY
0071      *   (>64 - >6C) - THEY NEED TO BE STORED AS >6D >0001 TO
0072      *   >6D >0009. WHEN RENUMBERING IS COMPLETE THE REVERSE
0073      *   OPERATION MUST BE PERFORMED.
0074      *   IT IS NOT NECESSARY TO DO THIS FOR VALUES -1 AND 0
0075      *   (>62 AND >63) AS THEY ARE NOT VALID LINE NUMBERS.
0076      *   THIS REQUIRES EACH LINE IN THE PROGRAM TO BE 'LISTED
0077      *   TO THE IOB' AND THE EDITOR INVOKED ON THE IOB (WITH
0078      *   THE APPROPRIATE COMPRESSION/UNCOMPRESSION FLAG SET).
0079      *
0080      * CALLING SEQUENCE:
0081      *
0082      *       B       @RENUM
0083      *
0084      * DATA:
0085      *
0086      *       R0-R2    GENERAL
0087      *       R3       CURRENT OPCODE
0088      *       R4       TABLE POINTER
0089      *       R5       SLT POINTER
0090      *       R6       END OF AVAILABLE STORAGE
0091      *       R7       NEW STATEMENT NUMBER
0092      *       R8       PSEUDO CODE BYTE POINTER (PBC)
0093      *
0094      * STORE REF'D OPCODES (DEF'D IN EDIT)
0095      *
0096      0001' GOTO   EQU  $+1
0097 0000 0000      DATA RNGOTO
0098      0003' GOSUB  EQU  $+1
0099 0002 0000      DATA RNGSUB
0100      0005' RSTR   EQU  $+1
0101 0004 0000      DATA RNRSTR
0102      0007' ON    EQU  $+1
0103 0006 0000      DATA RNON
0104      0009' INP   EQU  $+1
0105 0008 0000      DATA RNINP
0106      000B' ERR   EQU  $+1
0107 000A 0000      DATA RNERR
0108      000D' REM   EQU  $+1
0109 000C 0000      DATA RNREM
0110      000F' IF    EQU  $+1
0111 000E 0000      DATA RNIF
0112      0011' ELSE  EQU  $+1
0113 0010 0000      DATA RNELSE
0114      *
0115 0012      45  QUOTE  BYTE >45
0116 0013      44  DQUOTE BYTE >44
0117 0014      47  EXCLAM BYTE >47
0118 0015      6D  INT2B  BYTE >6D
0119 0016      6E  HINT2B BYTE >6E
0120 0017      6F  FP6B   BYTE >6F
0121 0018      3C  CONCAT BYTE >3C
  
```

0122	0019	3E	HASH	BYTE	>3E
0123	001A	3F	COMMA	BYTE	>3F
0124	001B	41	QUEST	BYTE	>41
0125	001C	40	SEMICOL	BYTE	>40
0126	001D	38	TO	BYTE	>38
0127	001E	3A	STEP	BYTE	>3A
0128	001F	3B	THEN	BYTE	>3B
0129	0020				EVEN

```

0131      *
0132      * BEFORE EXECUTING THE RENUMBERING ROUTINE, UNCOMPRESS
0133      * THE STORED PROGRAM FOR VALUES 1 TO 9.
0134      *
0135      0020' RENUMS EQU $
0136      0020 06A0      BL      @MODEOK      ABORT IF RUNNING !
0137      0022 0000
0138      0024 0201      LI      R1,10      DEFAULT INCREMENT = 10
0139      0026 000A
0140      0028 C0B1      MOV     R1,R2      DEFAULT NEW START = 10
0141      002A 983B      CB      *RB+,@STEP ; 'STEP' ?
0142      002C 001E'
0143      002E 130B      JEQ     REN$1      ; Y, GET STEP
0144      0030 060B      DEC     R8        ; N, BACKUP
0145      0032 06A0      BL      @CKEX     ; EXPRESSION ?
0146      0034 0000
0147      0036 1005      JMP     REN$2      ; N, TAKE DEFAULTS
0148      0038 2EC2      EVFIX  R2      ; Y, GET START LINE &
0149      003A 9800      CB      R0,@STEP ; 'STEP' ?
0150      003C 001E'
0151      003E 1601      B3A     EQU     $-2
0152      0040 2EC1      REN$1  EVFIX  R1      ; N, TAKE DEFAULT STEP
0153      0042 C801      REN$2  MOV     R1,@RENINC ; Y, GET INCREMENT
0154      0044 0000
0155      0046 60B1      S       R1,R2      ; SAVE STEP SIZE
0156      0048 C802      MOV     R2,@RENSTA    ; BACKUP START
0157      004A 0000
0158      004C 0720      *      SETO   @ESCFLG      NO ESCAPE DURING RENUMBER
0159      004E 0000
0160      0050 06A0      BL      @UNCOMP      UNCOMPRESS PROGRAM
0161      0052 02A0'
0162      0054 0000      DATA  0
0163      0056 C120      MOV     @NVD,R4      INITIALISE TABLE POINTER
0164      0058 0000
0165      005A C1A0      MOV     @NVS,R6      SET R6 TO END OF AVAILABLE MEMO
0166      005C 0000
0167      005E 0226      AI      R6,-6      - 4 (FOR OVERFLOW TEST)
0168      0060 FFFA
0169      0062 C0A0      MOV     @RENSTA,R2     INITIALISE STATEMENT &
0170      0064 004A'
0171      0066 C160      MOV     @VNT,R5      INITIALISE SLT PTR
0172      0068 0000
0173      006A 0645      *
0174      006C 0645      * END OF LINE ENCOUNTERED - GET PBC OF NEXT LINE
0175      006E B160      *
0176      0070 0000      ENDLIN DECT R5
0177      0072 1A4E      DECT R5      TO NEXT LINE
0178      0074 04D4      C       @SLT,R5      END OF PROGRAM?
0179      0076 C1E0      JL      SETUP
0180      0078 0000      *
0181      007A 0000      * END OF PROGRAM - UPDATE ALL PBC ENTRIES IN THE TABLE THAT
0182      007C 0000      * CORRESPOND TO ACTUAL PROGRAM LINE &S; RENUMBER THE SLT
0183      007E 0000      * ENTRIES; CHECK FOR NON-EXISTENT LINE &S; CLEAR DOWN THE
0184      0080 0000      * TABLE AREA AND RETURN TO THE INTERPRETER
0185      0082 0000      *
0186      0084 0000      CLR     *R4      Y - PUT ZERO TO MARK END OF TAB
0187      0086 0000      MOV     @RENSTA,R7    INITIALISE NEW STATEMENT &
0188      0088 0000
0189      008A 0000
0190      008C 0000
0191      008E 0000
0192      0090 0000
0193      0092 0000
0194      0094 0000
0195      0096 0000
0196      0098 0000
0197      009A 0000
0198      009C 0000
0199      009E 0000
0200      00A0 0000
0201      00A2 0000
0202      00A4 0000
0203      00A6 0000
0204      00A8 0000
0205      00AA 0000
0206      00AC 0000
0207      00AE 0000
0208      00B0 0000
0209      00B2 0000
0210      00B4 0000
0211      00B6 0000
0212      00B8 0000
0213      00BA 0000
0214      00BC 0000
0215      00BE 0000
0216      00C0 0000
0217      00C2 0000
0218      00C4 0000
0219      00C6 0000
0220      00C8 0000
0221      00CA 0000
0222      00CC 0000
0223      00CE 0000
0224      00D0 0000
0225      00D2 0000
0226      00D4 0000
0227      00D6 0000
0228      00D8 0000
0229      00DA 0000
0230      00DC 0000
0231      00DE 0000
0232      00E0 0000
0233      00E2 0000
0234      00E4 0000
0235      00E6 0000
0236      00E8 0000
0237      00EA 0000
0238      00EC 0000
0239      00EE 0000
0240      00F0 0000
0241      00F2 0000
0242      00F4 0000
0243      00F6 0000
0244      00F8 0000
0245      00FA 0000
0246      00FC 0000
0247      00FE 0000
0248      00FF 0000
0249      0100 0000
0250      0102 0000
0251      0104 0000
0252      0106 0000
0253      0108 0000
0254      010A 0000
0255      010C 0000
0256      010E 0000
0257      0110 0000
0258      0112 0000
0259      0114 0000
0260      0116 0000
0261      0118 0000
0262      011A 0000
0263      011C 0000
0264      011E 0000
0265      0120 0000
0266      0122 0000
0267      0124 0000
0268      0126 0000
0269      0128 0000
0270      012A 0000
0271      012C 0000
0272      012E 0000
0273      0130 0000
0274      0132 0000
0275      0134 0000
0276      0136 0000
0277      0138 0000
0278      013A 0000
0279      013C 0000
0280      013E 0000
0281      0140 0000
0282      0142 0000
0283      0144 0000
0284      0146 0000
0285      0148 0000
0286      014A 0000
0287      014C 0000
0288      014E 0000
0289      0150 0000
0290      0152 0000
0291      0154 0000
0292      0156 0000
0293      0158 0000
0294      015A 0000
0295      015C 0000
0296      015E 0000
0297      0160 0000
0298      0162 0000
0299      0164 0000
0300      0166 0000
0301      0168 0000
0302      016A 0000
0303      016C 0000
0304      016E 0000
0305      0170 0000
0306      0172 0000
0307      0174 0000
0308      0176 0000
0309      0178 0000
0310      017A 0000
0311      017C 0000
0312      017E 0000
0313      0180 0000
0314      0182 0000
0315      0184 0000
0316      0186 0000
0317      0188 0000
0318      018A 0000
0319      018C 0000
0320      018E 0000
0321      0190 0000
0322      0192 0000
0323      0194 0000
0324      0196 0000
0325      0198 0000
0326      019A 0000
0327      019C 0000
0328      019E 0000
0329      01A0 0000
0330      01A2 0000
0331      01A4 0000
0332      01A6 0000
0333      01A8 0000
0334      01AA 0000
0335      01AC 0000
0336      01AE 0000
0337      01B0 0000
0338      01B2 0000
0339      01B4 0000
0340      01B6 0000
0341      01B8 0000
0342      01BA 0000
0343      01BC 0000
0344      01BE 0000
0345      01C0 0000
0346      01C2 0000
0347      01C4 0000
0348      01C6 0000
0349      01C8 0000
0350      01CA 0000
0351      01CC 0000
0352      01CE 0000
0353      01D0 0000
0354      01D2 0000
0355      01D4 0000
0356      01D6 0000
0357      01D8 0000
0358      01DA 0000
0359      01DC 0000
036
```

0078 0064'				
0176 007A C160		MOV	@VNT, R5	INITIALISE SLT POINTER
007C 0068'				
0177 007E 0645		DECT	R5	POINT TO "FIRST LINE £"
0178 0080' RENST		EGU	#	RENUMBER STATEMENT
0179 0080 A1E0		A	@RENINC, R7	INCREMENT STATEMENT £
0082 0044'				
0180 0084 0225		AI	R5, -4	POINT TO NEXT SLT ENTRY
0086 FFFC				
0181 0088 8805		C	R5, @SLT	END OF PROGRAM?
008A 0070'				
0182 008C 1A11		JL	ENDRNS	
0183 008E C120		MOV	@NVD, R4	N - INITIALISE TABLE POINTER
0090 0058'				
0184 0092' SCHTAB		EGU	#	SEARCH TABLE FOR STMT £ MATCH
0185 0092 C514		MOV	*R4, *R4	END OF TABLE?
0186 0094 130B		JEG	ENDSTB	
0187 0096 8D15		C	*R5, *R4+	N - MATCH?
0188 0098 1607		JNE	SCHINC	
0189 009A 0644		DECT	R4	Y - BACK UP TO STATEMENT £
0190 009C 0534		NEG	*R4+	MARK AS DONE
0191 009E C014		MOV	*R4, R0	PUT NEW STMT NO IN STMT REF
0192 00A0 DC07		MOVB	R7, *R0+	(MAY CROSS WORD BOUNDARY)
0193 00A2 06C7		SWPB	R7	
0194 00A4 D407		MOVB	R7, *R0	
0195 00A6 06C7		SWPB	R7	
0196 00AB 05C4	SCHINC	INCT	R4	POINT TO NEXT TABLE ENTRY
0197 00AA 10F3		JMP	SCHTAB	LOOP
0198	*			
0199	* ALL TABLE ENTRIES CORRESPONDING TO THAT LINE £ DONE.			
0200	* NOW UPDATE THE SLT ENTRY			
0201	*			
0202 00AC' ENDSTB		EGU	#	
0203 00AC C547		MOV	R7, *R5	PUT NEW STMT NO IN SLT
0204 00AE 10E8		JMP	RENT	LOOP
0205	*			
0206	* ALL SLT ENTRIES DONE. NOW CHECK FOR NON-EXISTENT LINE £S			
0207	* CLEAR DOWN TABLE AREA IN THE PROCESS			
0208	*			
0209 00B0' ENDRNS		EGU	#	
0210 00B0 C120		MOV	@NVD, R4	REF TABLE
00B2 0090'				
0211 00B4 C014	CHECK	MOV	*R4, R0	GET STATEMENT £ ENTRY
0212 00B6 1124		JLT	CHECKA	STATEMENT £ = -VE
0213 00B8 1326		JEG	CHCKND	STATEMENT £ = 0 (FINISHED)
0214 00BA C064		MOV	@2(R4), R1	GET PBC ENTRY
00BC 0002				
0215 00BE C160		MOV	@SLT, R5	GET END OF SLT
00C0 00BA'				
0216 00C2 C0B5	CHECKB	MOV	*R5+, R2	GET STATEMENT £
0217 00C4 C235		MOV	*R5+, R8	GET DISPLACEMENT INTO BUS
0218 00C6 A220		A	@BUS, R8	REF START OF STMT IN BUS
00CB 0000				
0219 00CA 8201		C	R1, R8	PBC ENTRY IN THIS LINE?
0220 00CC 1AFA		JL	CHECKB	N - TRY NEXT LINE
0221 00CE 06A0		BL	@MOVEBN	Y - OUTPUT MSG TO IOB
00D0 0000				
0222 00D2 0D0A		DATA	>0D0A	CR LF
0223 00D4 42		TEXT	- 'Bad line No. ('	
00D5 61				

00D6	64		
00D7	20		
00D8	6C		
00D9	69		
00DA	6E		
00DB	65		
00DC	20		
00DD	4E		
00DE	6F		
00DF	2E		
00E0	D8		
0224	00E2	2F40	OUTINT R0
0225	00E4	06A0	BL @MOVELN
	00E6	0000	OUT 'OFFENDING' STMT & TO IOB
0226	00EB	29	OUT MSG TO IOB
	00E9	69	
	00EA	6E	TEXT -')in new line '
	00EB	20	
	00EC	6E	
	00ED	65	
	00EE	77	
	00EF	20	
	00F0	6C	
	00F1	69	
	00F2	6E	
	00F3	65	
	00F4	E0	
0227	00F6	2F42	OUTINT R2
0228	00F8	06A0	BL @MOVELN
	00FA	00E6	OUT LINE & TO IOB
0229	00FC	0000	NULL TERMINATE
0230	00FE	0000	DATA 0
0231	0100	04F4	CHECKA CLR *R4+
0232	0102	04F4	DATA TYPB*
0233	0104	10D7	CLR *R4+
0234	0106	06A0	JMP CHECK
	0108	02A0	CHCKND BL @UNCOMP
0235	010A	0009	OUTPUT IOB
0236	010C	0460	CLEAR STMT & ENTRY IN TABLE
	010E	0000	CLEAR PBC ENTRY IN TABLE
0237			BACK FOR NEXT ENTRY
0238			COMPRESS PROGRAM
0239			
0240	0110	A0A0	NINE DATA 9
	0112	00B2	B @CRLF
0241	0114	C215	RETURN TO INTERPRETER
0242	0116	A220	
	0118	00C8	
0243	011A	011A	* PROGRAM NOT EXHAUSTED - CONTINUE BUILDING TABLE ENTRIES
0244	011A	D618	* SETUP A @RENINC, R2
0245	011C	13A6	INCREMENT STATEMENT &
0246	011E	D0D8	MOV *R5, R8
0247			SET UP PSEUDO CODE POINTER
0248			ADD PSEUDO CODE BASE
0249			
0250	0120	9818	GOTOTS EQU #
	0122	0001	WHILE NOT END OF LINE
0251	0124	1308	END OF LINE (0)?
0252	0126	9818	MOV *R8, R8
	0128	0003	JEQ ENDLIN
			N - SAVE OPCODE
			* GOTO OR GOSUB
			CB *R8, @GOTO
			GOTO?
			JEQ T2COMB
			Y - CHECK VALIDITY OF CODE FOLL
			GOSUB?
			CB *R8, @GOSUB

0253	012A	1305	JEG	T2COMB	Y - CHECK VALIDITY OF CODE FOLL
0254	012C	1012	JMP	TEST3	JUMP ROUND 'TRAP'
0255	012E	'	TS2LPE	EGU	\$
0256	012E	0588	T2COMA	INC	RB
0257			*		
0258			*	TEST 2 COMMON CODE - ALSO USED BY TESTS 1, 3 & 6	
0259			*	(CHECKS FOR VALID DELIMITER - 0, :, !)	
0260			*		
0261	0130	9818	T2COM	CB	*RB, @INT2B
	0132	0015			2 BYTE INTEGER?
0262	0134	166E	JNE	T2WARN	N - PRINT WARNING MSG
0263	0136	0588	T2COMB	INC	RB
0264	0138	05C8		INCT	RB
0265	013A	D618		MOVB	*RB, *RB
0266	013C	1306	JEG	TS2OK	0?
0267	013E	9818	CB	*RB, @CONCAT	::?
	0140	0018			
0268	0142	1303	JEG	TS2OK	
0269	0144	9818	CB	*RB, @EXCLAM	!?
	0146	0014			
0270	0148	1664	JNE	T2WARN	N - PRINT WARNING MSG
0271	014A	0648	TS2OK	DECT	RB
0272	014C	06A0		BL	@STORE
	014E	0282			STORE IN TABLE
0273	0150	1052	JMP	TSTEND	
0274			*		
0275			*	RESTOR	
0276			*		
0277	0152	9818	TEST3	CB	*RB, @RSTR
	0154	0005			RESTOR?
0278	0156	160D	JNE	TEST4	
0279	0158	0588		INC	RB
0280	015A	D618		MOVB	*RB, *RB
0281	015C	134C	JEG	TSTEND	RESTOR W/O LINE NO?
0282	015E	9818	CB	*RB, @CONCAT	::?
	0160	0018			
0283	0162	1349	JEG	TSTEND	
0284	0164	9818	CB	*RB, @EXCLAM	!?
	0166	0014			
0285	0168	1346	JEG	TSTEND	
0286	016A	9818	CB	*RB, HASH	£?
	016C	0019			
0287	016E	1343	JEG	TSTEND	
0288	0170	10DF	JMP	T2COM	MUST BE STMT EXPRESSION
0289			*		
0290			*	ON AND IF	
0291			*		
0292	0172	9818	TEST4	CB	*RB, @ON
	0174	0007			ON?
0293	0176	1303	JEG	TS4LP	
0294	0178	9818	CB	*RB, @IF	IF?
	017A	000F			
0295	017C	1629	JNE	TEST5	
0296	017E	0588	TS4LP	INC	RB
0297	0180	9818	FIX2	CB	*RB, @THEN
	0182	001F			THEN?
0298	0184	131A	JEG	TS4LPE	
0299	0186	9818	CB	*RB, @CONCAT	::?
	0188	0018			
0300	018A	1317	JEG	TS4LPE	Y, TREAT AS A 'THEN'

```

0301      *      SKIP NOS IN CASE THEY CONTAIN "THEN" P-CODE
0302 018C 9818      CB      *RB,@INT2B      2-BYTE INTEGER?
        018E 0015'
0303 0190 1309      JEQ      TS4INT
0304 0192 9818      CB      *RB,@HINT2B      2-BYTE HEX INTEGER?
        0194 0016'
0305 0196 1306      JEQ      TS4INT
0306 0198 9818      CB      *RB,@FP6B      6-BYTE FP NUMBER?
        019A 0017'
0307 019C 1605      JNE      FIX
0308 019E 0228      AI      RB,6      SKIP 6 BYTE FP NUMBER
        01A0 0006
0309 01A2 10ED      JMP      TS4LP
0310 01A4 05C8      TS4INT INCT RB      SKIP 2 BYTE INTEGER
0311 01A6 10EB      JMP      TS4LP
0312 01A8 9818      FIX      CB      *RB,@QUOTE      '?'
        01AA 0012'
0313 01AC 1303      JEQ      FIX1      Y
0314 01AE 9818      CB      *RB,@DQUOTE      "?"
        01B0 0013'
0315 01B2 16E5      JNE      TS4LP      N - SKIP OVER CODE
0316      *      STRING - SKIP OVER CHARACTERS UNTIL THE NULL
0317      * NOTE: ASCII ';' SAME AS ENCODED 'THEN'
0318 01B4 D038      FIX1      MOVB *RB+,R0
0319 01B6 16FE      JNE      FIX1      NOT NULL - CONTINUE SKIP OP
0320 01B8 10E3      JMP      FIX2      NULL - CHECK NEXT CHAR
0321 01BA 0588      TS4LPE INC RB      SKIP "THEN"
0322 01BC 9803      CB      R3,@IF      WAS THIS AN IF?
        01BE 000F'
0323 01C0 13AC      JEQ      GOTO TS
0324 01C2 0588      TES4LP INC RB      Y - TEST STMT
0325 01C4 06A0      BL      @STORE      N (ON) - SKIP GOTO
        01C6 0282'      STORE IN TABLE
0326 01C8 9818      CB      *RB,@COMMA      ,?
        01CA 001A'
0327 01CC 1614      JNE      TSTEND      N - END OF STMT
0328 01CE 10F9      JMP      TES4LP      Y - INC PAST
0329      *
0330      * INPUT
0331      *
0332 01D0 9818      TEST5 CB      *RB,@INP      INPUT?
        01D2 0009'
0333 01D4 130F      JEQ      SKPINC
0334      *
0335      * ERROR
0336      *
0337 01D6 9818      CB      *RB,@ERR      ERROR STATEMENT?
        01D8 000B'
0338 01DA 13A9      JEQ      T2COMA
0339      *
0340      * REM
0341      *
0342 01DC 9818      CB      *RB,@RNREM      REM?
        01DE 000C'
0343 01E0 1603      JNE      TESTB
0344 01E2 D038      STRING MOVB *RB+,R0      SKIP TO END OF LINE (0 BYTE)
0345 01E4 16FE      JNE      STRING
0346 01E6 1006      JMP      SKPINC
0347      *
0348      * ELSE

```

```

0349      *
0350 01E8 9818 TEST8 CB *R8,@ELSE ELSE?
      01EA 0011'
0351 01EC 1604 JNE TSTEND
0352 01EE 0588 SKPNDA INC R8 SKIP PAST
0353 01FO 1094 SKPEND JMP GOTOTS TEST SUBORDINATE STMT
0354      *
0355      *
0356      *
0357 01F2 05C8 SKP2 INCT R8
0358 01F4 0588 SKPINC INC R8
0359      01F6' TSTEND EQU $
0360      *
0361      01F6' SKIP EQU $ SKIP TO END OF CURRENT STMT
0362 01F6 D618 MOV8 *R8,*R8 0 (EOL)?
0363 01F8 13FB JEQ SKPEND Y - TERMINATE SKIP
0364 01FA 9818 CB *R8,@CONCAT :: (ED STMT) ?
      01FC 0018'
0365 01FE 13F7 JEQ SKPNDA N
0366      *
0367      * CODE FOR INPUT'S ? - THEN CHECK FOR VALID DELIMITER
0368      *
0369 0200 9818 CB *R8,@QUEST ??
      0202 001B'
0370 0204 1618 JNE NTQST N
0371 0206 9803 CB R3,@INP INPUT STMT?
      0208 0009'
0372 020A 1615 JNE NTQST N
0373 020C 0588 INC R8 ? FOUND - CHECK STMT REF
0374 020E 9818 CB *R8,@INT2B 2 BYTE INTEGER?
      0210 0015'
0375 0212 1626 T2WARN JNE WARN N - PRINT WARNING
0376 0214 0588 INC R8
0377 0216 05C8 INCT R8 POINT TO DELIMITER
0378 0218 D618 MOV8 *R8,*R8 0?
0379 021A 1397 JEQ TS20K
0380 021C 9818 CB *R8,@CONCAT ::?
      021E 0018'
0381 0220 1394 JEQ TS20K
0382 0222 9818 CB *R8,@EXCLAM !?
      0224 0014'
0383 0226 1391 JEQ TS20K
0384 0228 9818 CB *R8,@COMMA ,?
      022A 001A'
0385 022C 138E JEQ TS20K
0386 022E 9818 CB *R8,@SEMICOL :?
      0230 001C'
0387 0232 1616 JNE WARN
0388 0234 108A JMP TS20K
0389      *
0390      * TEST FOR STRINGS
0391      *
0392      0236' NTQST EQU $
0393 0236 9818 CB *R8,@QUOTE '?
      0238 0012'
0394 023A 13D3 JEQ STRING Y
0395 023C 9818 CB *R8,@DQUOTE "?"
      023E 0013'
0396 0240 13D0 JEQ STRING Y
0397 0242 9818 CB *R8,@EXCLAM !? (STORED SAME AS STRING)

```

```
0244 0014'
0398 0246 13CD      JEQ  STRING
0399                *
0400                * TEST FOR NUMBERS
0401                *
0402 0248 9818      CB   *RB,@INT2B      2 BYTE INTEGER?
        024A 0015'
0403 024C 13D2      JEQ  SKP2            Y
0404 024E 9818      CB   *RB,@HINT2B     2 BYTE HEX INTEGER?
        0250 0016'
0405 0252 13CF      JEQ  SKP2            Y
0406 0254 9818      CB   *RB,@FP6B       6 BYTE FP NUMBER?
        0256 0017'
0407 0258 16CD      JNE  SKPINC          N
0408 025A 0228      AI   RB,6            Y - SKIP NUMBER
        025C 0006
0409 025E 10CB      JMP  SKIP
```

```
0411      *
0412      * LINE NUMBERS AS EXPRESSIONS - OUTPUT WARNING MESSAGE
0413      *
0414 0260 06A0  WARN  BL  @MOVEBN          PUT MESSAGE IN BUFFER
      0262 00D0'
0415 0264 0D0A          DATA >0D0A          CR LF
0416 0266 50          TEXT -'Problem with new line '
      0267 72
      0268 6F
      0269 62
      026A 6C
      026B 65
      026C 6D
      026D 20
      026E 77
      026F 69
      0270 74
      0271 68
      0272 20
      0273 6E
      0274 65
      0275 77
      0276 20
      0277 6C
      0278 69
      0279 6E
      027A 65
      027B E0
0417 027C 2F42          OUTINT R2          OUT LINE £ INTO IOB
0418 027E 00FE'        DATA TYPB$        OUTPUT BUFFER
0419 0280 10BA          JMP TSTEND         FIND END OF STATEMENT
```

0421	*		
0422	* STORE STATEMENT NUMBER AND PBC ADDRESS IN TABLE		
0423	*		
0424 0282 8106	STORE C R6,R4	STORAGE OVERFLOW?	
0425 0284 1A06	JL STOVFL		
0426 0286 DD38	MOVB *R8+,*R4+	N - STORE STMT NO	
0427 0288 DD18	MOVB *R8,*R4+	(CAN BE ODD BOUNDARY)	
0428 028A 0608	DEC R8		
0429 028C CD08	MOV R8,*R4+	STORE CURRENT PBC	
0430 028E 05C8	INCT R8	POINT TO NEXT BYTE IN P-CODE	
0431 0290 045B	B *R11	RETURN	
0432	*		
0433	* OVERFLOW - CLEAR TABLE AREA THEN OUTPUT OVERFLOW MESSAGE.		
0434	* (RETURNS TO INTERPRETER)		
0435	*		
0436 0292 C120	STOVFL MOV @NVD,R4	INITIALISE TABLE POINTER	
0294 00B2'			
0437 0296 05C6	INCT R6	SET R6 TO LAST WORD TO BE CLEAR	
0438 0298 04F4	CLRLP CLR *R4+		
0439 029A 8106	C R6,R4	DONE?	
0440 029C 14FD	JHE CLRLP		
0441 029E 2F8A	DATA ERROR+10	STORAGE OVERFLOW (ERROR 10)	

```

0443      *
0444      * THIS ROUTINE PERFORMS THE NECESSARY COMPRESSION/
0445      * UNCOMPRESSION OF THE STORED PROGRAM
0446      *
0447      *      BL      @UNCOMP
0448      *      DATA [FLAG]
0449      *
0450      * FLAG = 0 THEN UNCOMPRESS
0451      *      = 9 THEN COMPRESS
0452      *      ANYTHING ELSE WILL CAUSE PROBLEMS!      (BIG ONES !!!)
0453      *
0454      * RENLNE - LAST STATEMENT LINE NUMBER THAT WAS LISTED
0455      * RENCMP - COMPRESSION/UNCOMPRESSION FLAG (USED IN EDIT)
0456      *
0457 02A0 C83B UNCOMP MOV *R11+,@RENCMP      GET COMPRESSION FLAG
      02A2 0000
0458 02A4 C80B      MOV R11,@RENRET      SAVE RETURN ADDRESS
      02A6 0000
0459 02A8 04E0      CLR @RENLNE      START AT THE FIRST LINE £
      02AA 0000
0460 02AC 0720      SETD @DCNT      CLEAR INDENTATION COUNTER
      02AE 0000
0461 02B0 C220 UNCMP1 MOV @VNT,R8
      02B2 007C'
0462 02B4 0648      DECT R8
0463 02B6 880B UNCMP2 C R8,@SLT      BACK UP TO SLT TERMINATOR
      02B8 00C0'      REACHED END OF SLT?
0464 02BA 120F      JLE UNCMP3      Y - RETURN TO CALLER
0465 02BC 0228      AI R8,-4      N - GET NEXT LINE £
      02BE FFFC
0466 02C0 8620      C @RENLNE,*R8      RENLNE => LINE £?
      02C2 02AA'
0467 02C4 14F8      JHE UNCMP2      Y - GOTO NEXT LINE
0468 02C6 C818      MOV *R8,@RENLNE      N - SAVE CURRENT LINE £
      02C8 02C2'
0469      *
0470      * FOUND NEXT LINE TO BE COMPRESSED/UNCOMPRESSED
0471      *
0472 02CA C1E0      MOV @IOB,R7      REF IOB
      02CC 0000
0473 02CE C078      MOV *R8+,R1      GET LINE £; R8 REFS PBC ENTRY
0474 02D0 06A0      BL @LSTL      LIST THIS LINE TO IOB
      02D2 0000
0475 02D4 06A0      BL @EDIT1      EDIT LINE IN IOB
      02D6 0000
0476      *
0477      * SLT/VNT/ ETC MAY HAVE CHANGED - FIND THE NEXT LINE TO
0478      * BE OPERATED ON (WHEN STATEMENT £ ENTRY IN SLT > RENLNE)
0479      *
0480 02D8 10EB      JMP UNCMP1
0481 02DA C2E0 UNCMP3 MOV @RENRET,R11      RESTORE RETURN ADDRESS
      02DC 02A6'
0482 02DE 045B      B *R11      RETURN TO CALLING ROUTINE
0483      END
  
```

NO ERRORS,

NO WARNINGS

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.SGNF
OBJECT ACCESS NAME= ADHOC.OBJ.SGNF
LISTING ACCESS NAME= ADHOC.LST.SGNF
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

```
0002          IDT  'SGNF'
0003          *
0004          DEF  SGNF
0005          *
0006          REF  EVSFR$,FPAC2
0007          *
0008          * SIGN FUNCTION
0009          *
0010 0000 0032 SGNF  MOV  *R2+,R0          ; GET SIGN
0011 0002 1602          JNE  SGNF1
0012 0004 0012          MOV  *R2,R0          ; INTEGER, GET SIGN
0013 0006 1306          JEQ  SGNF3          ; 0, RETURN 0
0014 0008 1503 SGNF1 JGT  SGNF2          ; +
0015 000A 0620          DEC  @FPAC2          ; -, RETURN -1
0016 000C 0000
0016 000E 1002          JMP  SGNF3          ; GET ADR & RETURN
0017          *
0018 0010 05A0 SGNF2 INC  @FPAC2          ; +, RETURN 1
0019 0012 000C'
0019 0014 0460 SGNF3 B    @EVSFR$          ; RETURN
0019 0016 0000
0020          END
NO ERRORS,      NO WARNINGS
```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.SIZE
OBJECT ACCESS NAME= ADHOC.OBJ.SIZE
LISTING ACCESS NAME= ADHOC.LST.SIZE
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

```

0002          IDT 'SIZE'
0003          *
0004          * DEFINE FLOATING POINT XOPS
0005          *
0006          DXOP LOADF,0          LOAD FPAC
0007          DXOP STORE,1         STORE FPAC
0008          DXOP FADD,2         ADD TO FPAC
0009          DXOP FSUB,3         SUBTRACT FROM FPAC
0010          DXOP FMUL,4         MULTIPLY FPAC
0011          DXOP FDIV,5         DIVIDE FPAC
0012          DXOP SCALE,6        SCALE FPAC
0013          DXOP NORMAL,7       NORMALIZE FPAC
0014          DXOP CLEAR,8        CLEAR FPAC
0015          DXOP NEGATE,9       NEGATE FPAC
0016          DXOP FLOATF,10      FLOAT FPAC
0017          DXOP EVFIX,11       EVALUATE AND FIX
0018          DXOP OUTFP,12       OUT FLOATING POINT &
0019          DXOP OUTINT,13      OUT INTEGER
0020          2F80 ERROR EQU >2F80          XOP XX,14 (ERROR CALL)
0021          2FA0 ERROR2 EQU ERROR+>20
0022          *
0023          DEF SIZE,EVENRT
0024          DEF OUTR1U,MOVEB,MOVEL
0025          DEF MOVEBN,MOVELN
0026          REF VDT,BUS,NVS,NVD,VNT
0027          REF CRLF,TYPBE$,RTSTOR
0028          REF FPAC,C4A00,IOB
0029          *
0030          * SIZE ROUTINE
0031          *
0032          0000 06A0 SIZE BL @MOVEBN          ; OUT 'PRGM: '
0033          0002 009C'
0034          0004 0A0D DATA >0A0D
0035          0006 50 TEXT - 'PRGM: '
0036          0007 52
0037          0008 47
0038          0009 4D
0039          000A C6
0040          000C C060 MOV @NVD,R1          ; PRGM=((NVD-VDT)+VNT)-BUS
0041          000E 0000
0042          0010 6060 S @VDT,R1
0043          0012 0000
0044          0014 A060 A @VNT,R1
0045          0016 0000
0046          0018 6060 S @BUS,R1
0047          001A 0000
0048          001C 06A0 BL @OUTR1U          OUT R1 UNSIGNED
0049          001E 0064'
0050          0020 06A0 BL @MOVBY          ; OUT 'BYTES'+ 'VARS: '
0051          0022 0078'
0052          0024 56 TEXT - 'VARS: '
0053          0025 41
0054          0026 52
0055          0027 53
0056          0028 C6
0057          002A C060 MOV @IOB,R1          ; VARS=IOB-NVS+NVD-VDT
0058          002C 0000
0059          002E 6060 S @NVS,R1
0060          0030 0000
0061          0032 A060 A @NVD,R1

```

```

0034 000E'
0045 0036 6060      S      @VDT,R1
0038 0012'
0046 003A 0221      AI      R1,-8      ; REMOVE EPROM OVERHEAD
003C FFF8
0047 003E 06A0      BL      @OUTR1U    ; OUT R1 UNSIGNED
0040 0064'
0048 0042 06A0      BL      @MOVBY
0044 0078'
0049 0046 46        TEXT - 'FREE: '
0047 52
0048 45
0049 45
004A C6
0050 004C C060      MOV      @NVS,R1      ; FREE=NVS-NVD
004E 0030'
0051 0050 6060      S      @NVD,R1
0052 0034'
0052 *
0053 0054 06A0 SIZE1 BL      @OUTR1U    ; OUT R1 UNSIGNED
0056 0064'
0054 0058 06A0      BL      @MOVBY
005A 0078'
0055 005C 00        BYTE 0      ; NO FOLLOWING TEXT
0056 005E          EVEN
0057 005E 0000      DATA TYPBE#    ; OUTPUT BUFFER
0058 0060 0460      B      @CRLF      ; EXIT
0062 0000
0059 *
0060 *      OUT R1 UNSIGNED      USES R0
0061 *
0062 0064 0200 OUTR1U LI      R0,FPAC    GET THE FPAC
0066 0000
0063 0068 CC20      MOV      @C4A00,*R0+  SET EXPONENT
006A 0000
0064 006C 04F0      CLR      *R0+      RESET 2ND WRD
0065 006E C401      MOV      R1,*R0      LOAD UP R1
0066 0070 2DC0      NORMAL R0      NORMALISE IT
0067 0072 2F20      OUTFP @FPAC      OUT THE FPAC
0074 0066'
0068 0076 045B      RT      EXIT
0069 *
0070 * OUT 'Bytes<CR><LF>'      THEN INLINE TEXT
0071 *      (-VE/NULL TERMINATOR)
0072 *
0073 0078 C04B MOVBY MOV      R11,R1      SAVE RETURN
0074 007A 06A0      BL      @MOVELN      OUT 'BYTES'
007C 00A0'
0075 007E 20        TEXT ' Bytes'
007F 42
0080 79
0081 74
0082 65
0083 73
0076 0084 0D        BYTE >0D,->0A
0077 0086 C2C1      MOV      R1,R11      RESTORE RETURN ADDRESS
0078 0088 100B      JMP      MOVELN      & OUT INLINE TEXT

```

```

0080      *
0081      * MOVE STRING ROUTINES - NULL TERMINATOR
0082      *
0083      *      BL @MOVEB      MOVE INTO IOB
0084      *      BL @MOVEL      MOVE INTO R7
0085      *
0086      *      IN  0(11) = STRING
0087      *              R7 = OBC
0088      *      OUT   R7 = UPDATED OBC
0089      *
0090 008A C1E0 MOVEB  MOV  @IOB,R7      GET BUFFER ADR
      008C 002C'
0091 008E DDFB MOVEL  MOVB *R11+,*R7+  MOVE CHARACTER
0092 0090 16FE      JNE  #-2          UNTIL NULL
0093 0092 0607 NTERM  DEC  R7          BACKUP OVER NULL
0094      *
0095      * FORCE THE RETURN ADDRESS TO AN EVEN WORD BOUNDARY
0096      *
0097 0094 058B EVENRT INC  R11
0098 0096 024B      ANDI R11,>FFFE
      0098 FFFE
0099 009A 045B      RT
0100      *
0101      * MOVE STRING ROUTINES - NULL/NEGATED CHARACTER TERMINATION
0102      *
0103      *      BL @MOVEBN      MOVE INTO IOB
0104      *      BL @MOVELN      MOVE INTO R7
0105      *
0106      *      IN  0(11)' = STRING
0107      *              R7 = OBC
0108      *      OUT   R7 = UPDATED OBC
0109      *
0110 009C C1E0 MOVEBN MOV  @IOB,R7      GET BUFFER ADR
      009E 008C'
0111 00A0 DDFB MOVELN MOVB *R11+,*R7+  MOVE CHARACTER
0112 00A2 13F7      JEQ  NTERM          NULL, TERMINATE
0113 00A4 15FD      JGT  MOVELN          +VE, CONTINUE COPY
0114      * R7 NOW POINTS TO THE BYTE AFTER THE STRING END
0115 00A6 C80B      MOV  R11,@RTSTOR    SAVE R11
      00A8 0000
0116 00AA 04CB      CLR  R11          READY R11
0117 00AC D2E7      MOVB @-1(R7),R11    GET BACK THE CHARACTER
      00AE FFFF
0118 00B0 050B      NEG  R11          NEGATE IT TO GET REAL VALUE
0119 00B2 D9CB      MOVB R11,@-1(R7)    RE-STORE IT
      00B4 FFFF
0120 00B6 C2E0      MOV  @RTSTOR,R11    GET RETURN ADDRESS
      00BB 00AB'
0121 00BA 10EC      JMP  EVENRT        EXIT
0122      END
    
```

NO ERRORS, NO WARNINGS

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.TICF
OBJECT ACCESS NAME= ADHOC.OBJ.TICF
LISTING ACCESS NAME= ADHOC.LST.TICF
ERROR ACCESS NAME=
OPTIONS= BUNLST, DUNLST, MUNLST
MACRO LIBRARY PATHNAME=

```

0002          IDT 'TICF'
0003          *
0004          *
0005          *
0006          REF  FNRM          ;FIXES ARGUMENT
0007          *
0008          DXOP LOADF,0        ;LOAD FPAC
0009          DXOP STORE,1       ;STORE FPAC
0010          DXOP FADD,2        ;ADD TO FPAC
0011          DXOP FSUB,3        ;SUBTRACT FROM FPAC
0012          DXOP FMUL,4        ;MULTIPLY FPAC
0013          DXOP FDIV,5        ;DIVIDE FPAC
0014          DXOP SCALE,6       ;SCALE FPAC
0015          DXOP NORMAL,7      ;NORMALIZE FPAC
0016          DXOP CLEAR,8       ;CLEAR FPAC
0017          DXOP NEGATE,9      ;NEGATE FPAC
0018          DXOP FLOATF,10     ;FLOAT FPAC
0019          DXOP EVFIX,11      ;EVALUATE AND FIX
0020          DXOP OUTFP,12      ;OUT FLOATING POINT &
0021          DXOP OUTINT,13      ;OUT INTEGER
0022          *
0023          REF  FPWP          ;FLOATING POINT WORKSPACE POINT
0024          REF  TEMP          ;3 WRD TEMPORARY STORAGE
0025          REF  CLKTO1
0026          REF  CLKTO2
0027          REF  EVSFR$        ;EXIT ADDRESS
0028          DEF  GTICZ,TICF
0029          *
0030          * ABSTRACT:
0031          *
0032          * GET AND FLOAT THE NUMBER OF ELAPSED
0033          * TICS OF THE CLOCK IN FPAC.
0034          *
0035          * CALLING SEQUENCE:
0036          *
0037          * BLWP @GTICZ
0038          *
0039          * OUT FPAC = CLOCK TICS
0040          *

```



```

0042      *
0043 0000 0000 GTICZ DATA FPWP,GTIC      ; ACCESS VECTOR
0044      *
0045 0004 0300 GTIC  LIM1 0                ; DISABLE INTERRUPTS
      0006 0000
0046 0008 C060      MOV @CLKT01,R1
      000A 0000
0047 000C C0A0      MOV @CLKT02,R2
      000E 0000
0048 0010 0300      LIM1 15                ; ENABLE INTERRUPTS
      0012 000F
0049 0014 0200      LI  R0,>4A00           ; GET EXPONENT
      0016 4A00
0050 0018 0460      B    @FNRM             ; NORMALIZE
      001A 0000

0051      *
0052      * ABSTRACT:
0053      *
0054      * GET DELTA TICS. (SUBTRACT FROM THE
0055      * FUNCTION ARGUMENT THE NUMBER OF ELAPSED
0056      * TICS OF THE CLOCK.)
0057      *
0058      * CALLING SEQUENCE:
0059      *
0060      * B @TICF
0061      *
0062      * IN (R2) = ARGUMENT
0063      *
0064      * OUT (R2) = ARG - TICS
0065      *
0066      * NORMAL EXIT TO EVSFR
0067      *
0068 001C 2C12 TICF  LOADF *R2                ; LOAD FPAC
0069 001E 2E80      FLOATF 0                 ; FLOAT IF NECESSARY
0070 0020 2C60      STORE @TEMP             ; STORE IN TEMP
      0022 0000
0071 0024 0420      BLWP @GTICZ             ; GET TICS IN FPAC
      0026 0000'
0072 0028 2CE0      FSUB @TEMP              ; FPAC=FPAC-TEMP
      002A 0022'
0073 002C 0460      B    @EVSFR$            ; RETURN
      002E 0000
0074      END
NO ERRORS,      NO WARNINGS

```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.TIME
OBJECT ACCESS NAME= ADHOC.OBJ.TIME
LISTING ACCESS NAME= ADHOC.LST.TIME
ERROR ACCESS NAME=
OPTIONS= BUNLST, DUNLST, MUNLST
MACRO LIBRARY PATHNAME=

```
0002          IDT 'TIME'
0003      *
0004      * THIS MODULE CONTAINS THE ROUTINE THAT SETS UP THE
0005      * TIME STORAGE AREA FOR BASIC AND ALSO RETURNS TO BASIC THE
0006      * CURRENT TIME
0007      *
0008      *          TIME          - OUTPUT THE TIME
0009      *          TIME <EXP>, <EXP>, <EXP> - SET THE TIME
0010      *          TIME <VAR>          - GET REAL TIME CLOCK IN <VAR>
0011      *
0012      * MODULE DEFINITIONS:
0013      *
0014      *          DEF TIMY
0015      *
0016      * EXTERNAL ROUTINES:
0017      *
0018      *          REF CKEX, EVSDZ, TYPB$, NLIN
0019      *
0020      * EXTERNAL DATA:
0021      *
0022      *          REF B20, CLKADR, FPWP, IOB
0023      *          REF B3A
0024      *
0025      * XOP EQUATES
0026      *
0027      *          DXOP EVFIX, 11
```

0029 *
0030 * TIME STATEMENT
0031 *
0032 0000 0203 TIMY LI R3,CLKADR
0002 0000
0033 0004 020C LI R12,>1EE2
0006 1EE2

; GET CLOCK'S WP ADDRESS.

; SELECT 9995'S FLAG1

*
* DETERMINE TYPE
*

BLWP @EVSDZ

JMP TIM5

JMP TIM3
BL @CKEX

JMP TIM2

; LOOK FOR STRING

; " OR ' - PROBLEM
; *

; LOOK FOR EXPRESSION

; NONE

004

0042		*			
0043		*	* SET REAL TIME CLOCK WITH A 40MS INTERVAL		
0044		*			
0045	0016 0204	TIMO	LI	R4,3	; ONLY 3 <EXP>S ALLOWED
	0018 0003				
0046	001A 0300		LIMI	0	; NO INTERRUPTS
	001C 0000				
0047	001E 2EF3	TIMOA	EVFIX	*R3+	; STORE RESULT
0048	0020 0280		CI	RO, >3F00	; ?
	0022 3F00				
0049	0024 161A		JNE	TIMR	; N - SET 9901 INTERVAL
0050	0026 0604		DEC	R4	; Y - ANY MORE <EXP>S ALLOWED?
0051	0028 16FA		JNE	TIMOA	; Y - GET NEXT
0052	002A 1017		JMP	TIMR	; N, EXIT
0053		*			
0054	002C 058B	TIM2	INC	R8	; INCREMENT OVER DLIM
0055	002E C0A0		MOV	@IOB, R2	; GET IOB ADDRESS
	0030 0000				
0056	0032 DCA0		MOVB	@B20, *R2+	; OUT SPACE
	0034 0000				
0057	0036 0706		SETD	R6	; SET TO PRINT
0058	0038 1001		JMP	TIM4	
0059		*			
0060	003A 04C6	TIM3	CLR	R6	; STORE IN VARIABLE
0061		*			
0062	003C 0420	TIM4	BLWP	@TIMVEC	; GET TIME
	003E 0082				
0063	0040 06A0		BL	@TIM6	; DO HOURS
	0042 0062				
0064	0044 C004		MOV	R4, R0	
0065	0046 06A0		BL	@TIM6	; DO MINUTES
	0048 0062				
0066	004A C005		MOV	R5, R0	
0067	004C 06A0		BL	@TIM6	; DO SECONDS
	004E 0062				
0068	0050 0602		DEC	R2	; END STRING
0069	0052 7492		SB	*R2, *R2	
0070	0054 C186		MOV	R6, R6	; TYPE?
0071	0056 1301		JEQ	TIMR	; N - RETURN
0072		*			
0073	0058 0000	TIM5	DATA	TYPB\$; Y - OUTPUT BUFFER
0074	005A 0300	TIMR	LIMI	15	
	005C 000F				

```

0075 005F 0460      B      @NLIN
      0060 0000
0076      *
0077      * FIX RO TO XX:
0078      *
0079 0062 04C1  TIM6   CLR   R1      ; CLEAR COUNTER
0080      *
0081 0064 0220  TIM7   AI    RO,-10   ; NEGATIVE
      0066 FFF6
0082 0068 1102      JLT   TIM8      ; Y
0083 006A 0581      INC   R1        ; N, COUNT
0084 006C 10FB      JMP   TIM7
0085      *
0086 006E 06C1  TIM8   SWPB  R1
0087 0070 D001      MOVB  R1,R0      ; APPEND
0088 0072 0220      AI    RO,>2F3A   ; ADD ASCII CODES
      0074 2F3A
0089 0076 DC80      MOVB  RO,*R2+    ; MOVE INTO OUTPUT STREAM
0090 0078 06C0      SWPB  RO
0091 007A DC80      MOVB  RO,*R2+
0092 007C DCA0      MOVB  @B3A,*R2+  ; OUT ":
      007E 0000
0093 0080 045B      RT
0094      *
0095 0082 0000  TIMVEC DATA FPWP,TIM9
0096      *
0097 0086 0300  TIM9   LIM1  0      ; MASK INTRPTS
      0088 0000
0098 008A 0203      LI    R3,CLKADR  ; GET SOURCE
      008C 0002
0099 008E C773      MOV   *R3+,*R13  ; SET HOURS
0100 0090 CB73      MOV   *R3+,@8(R13) ; SET MIN
      0092 0008
0101 0094 CB53      MOV   *R3,@10(13) ; SET SECONDS
      0096 000A
0102 0098 0380      RTWP             ; RETURN
0103      END
    
```

NO ERRORS, NO WARNINGS

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.TRACE
OBJECT ACCESS NAME= ADHOC.OBJ.TRACE
LISTING ACCESS NAME= ADHOC.LST.TRACE
ERROR ACCESS NAME=
OPTIONS= BUNLST, DUNLST, MUNLST
MACRO LIBRARY PATHNAME=

```
0002          IDT  'TRACE'
0003          *
0004          *
0005          *
0006          REF  TRAFLG,NLINO
0007          DEF  TONY,TOFY
0008          *
0009          *  TRACE ON (TON)
0010 0000 0720  TONY  SETD @TRAFLG          ; SET TRACE FLAG
          0002 0000
0011 0004 0460  TONY1 B    @NLINO          ; EXIT TO NLIN
          0006 0000
0012          *
0013          *  TRACE OFF (TOF)
0014          *
0015 0008 04E0  TOFY  CLR  @TRAFLG          ; RESET TRACE FLAG
          000A 0002'
0016 000C 10FB          JMP  TONY1          ; EXIT
0017          END
NO ERRORS,      NO WARNINGS
```


ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.UNIT
OBJECT ACCESS NAME= ADHOC.OBJ.UNIT
LISTING ACCESS NAME= ADHOC.LST.UNIT
ERROR ACCESS NAME=
OPTIONS= BUNLST, DUNLST, MUNLST
MACRO LIBRARY PATHNAME=

```
0002          IDT 'UNIT'
0003          *
0004          *
0005          *
0006          REF NLINO          ;RETURN TO MULTIPLEXOR
0007          DXOP EVFIX,11      ;EVALUATE AND FIX
0008          REF UNIT          ;INDEX TO UNIT
0009          REF ERROR2
0010          DEF UNTY
0011          *
0012          *      THE UNIT COMMAND IS USED TO SELECT OUTPUT DEVICES
0013          *      ACCORDING TO BIT POSITIONS WITHIN @UNIT.
0014          *
0015          *      BIT:   0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
0016          *      PORT: 16 15 14 13 12 11 10  9  8  7  6  5  4  3  2  1
0017          *
0018          *      FORMAT :
0019          *      UNIT <ARG 1>
0020          *
0021          *      IF ARG 1 IS 0 THEN ALL PORTS ARE DISABLED
0022          *      IF ARG 1 IS -VE THEN THE SPECIFIED PORT IS DISABLED
0023          *      IF ARG 1 IS +VE THEN THE SPECIFIED PORT IS ENABLED
0024          *
0025          *      VALID RANGE OF ARG 1 IS 1..16
0026          *      CALLING SEQUENCE:
0027          *
0028          *      B @UNTY
0029          *
0030          *      NORMAL EXIT TO NLINO
0031          *
```

```

0033 0000 2EC1 UNTY EVFIX R1 GET THE UNIT &
0034 0002 0202 LI R2,UNIT POINT TO STORAGE
      0004 0000
0035 0006 0204 LI R4,>E481 R4='SOC R1,*R2'
      0008 E481
0036 000A C001 MOV R1,R0 PUT UNIT & IN R0
0037 000C 1310 JEQ NOUNIT O, RESET UNIT FLAG
0038 000E 0740 ABS R0
0039 0010 1502 JGT UNITON +VE, R4 OK TO SET BIT
0040 0012 0204 LI R4,>4481 -VE, R4='SZC R1,*R2'
      0014 4481
0041 *
0042 0016 0201 UNITON LI R1,>0001 SET BIT MASK
      0018 0001
0043 001A 0600 DEC R0 MAKE O..15 FOR SHIFT
0044 001C 1304 JEQ UDN1 O, MASK OK
0045 001E 0280 CI R0,16 VALID?
      0020 0010
0046 0022 1407 JHE ERR N, ERROR IT
0047 0024 0A01 SLA R1,0 Y, POSITION MASK
0048 0026 0484 UDN1 X R4 SET BIT/RESET BIT
0049 *
0050 0028 0608 UEXIT DEC R8 BACKUP POINTER
0051 002A 0460 B @NLINO EXIT
      002C 0000
0052 *
0053 002E 04D2 NOUNIT CLR *R2 RESET ALL UNIT FLAGS
0054 0030 10FB JMP UEXIT AND EXIT
0055 *
0056 0032 0000 ERR DATA ERROR2,46
0057 END

```

NO ERRORS,

NO WARNINGS

ACCESS NAMES TABLE .

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.BOOT
OBJECT ACCESS NAME= ADHOC.OBJ.BOOT
LISTING ACCESS NAME= ADHOC.LST.BOOT
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

0002 IDT 'BOOT'

0003 *

0004 *

0005 * ROUTINE LIST:

0006 *

0007 *

0008 *

0009

0010

0011

0012

0013

0014

0015

0016

0017

0018

0019

0020

0021

0022

0023

0024

0025

0026

0027

0028

0029

0030

0031

0032

0033

0034

0035

BOOTY

; BOOT STATEMENT

DEF BOOTY

REF TMPBUF

REF WPR1

1K scratch buffer

System workspace

Track 0 Sector 1 Data Format

		0	2	4	6	8	A	C	E
		+	+	+	+	+	+	+	+
00		WP	PC	----	----	----	----	----	----
		+	+	+	+	+	+	+	+
10			----	LN	SA	----	----	----	CS
		+	+	+	+	+	+	+	+
20		CO	PY	RI	GH	T^	T.	I.	L^
		+	+	+	+	+	+	+	+
30		BO	OT	^L	DA	DE	R^	V1	.0
		+	+	+	+	+	+	+	+
		~	~	~	~	~	~	~	~

WP = Entry WP

PC = Entry PC

LN = Two times number of bytes to transfer

SA = Load address for absolute code program

CS = Checksum on words hex00 to hex1C inclusive

^ = Space Character

All other locations are reserved for vector storage.

```

0037      *
0038      *      MEMORY MAPPED I/O EQUATES
0039      *
0040      F140  FDC      EQU  >F140      FDC memory address
0041      *
0042      *      CRU BASE EQUATES
0043      *
0044      01C0  DMAC     EQU  >01C0      DMAC cru address
0045      *
0046      *      CRU I/O BITS
0047      *
0048      0002  SIZEI    EQU  2          Drive size  1 = 8"    0 = 5.25"
0049      0003  DNSTY    EQU  3          Drive density 1 = DD  0 = SD
0050      0004  FDCINT    EQU  4          FDC interrupt flag
0051      0004  SIZED     EQU  4          Drive size  1 = 8"    0 = 5.25"
0052      *
0053      *      SECTOR BUFFER EQUATES
0054      *
0055  0000      DORG  0
0056  0000      BSS  20          Vectors
0057  0014      LENGTH BSS  2          Twice number of bytes to xfer
0058  0016      SADDR  BSS  8          Start address for load
0059  001E      SUM    BSS  2          Checksum
  
```

0061	0000		RDRG 0	
0062		*		
0063		*	BOOT COMMAND ENTRY POINT	
0064		*		
0065	0000 02E0	BOOTY	LWPI WPR1	Use internal RAM for wp
	0002 0000			
0066	0004 0300		LIMI 0	Mask all interrupts
	0006 0000			
0067	0008 0201		LI R1, TABLE	Load pointer to cmd table
	000A 00EB			
0068	000C 0206		LI R6, WAIT	Load pointer to wait routine
	000E 00C4			
0069	0010 0208		LI R8, ENCODE	Load pointer to 5.25" values
	0012 00E4			
0070	0014 0209		LI R9, FDC	Load pointer to FDC
	0016 F140			
0071	0018 04CC		CLR R12	Load CRU base of i/o
0072	001A D671		MOVB *R1+, *R9	Send RSTC
0073	001C 0696		BL *R6	Call wait
0074		*		
0075		*	CHECK SIZE JUMPER	
0076		*		
0077	001E 1E04		SBZ SIZED	Set 5.25" drive
0078	0020 1F02		TB SIZEI	Read jumper
0079	0022 1602		JNE MINI	Skip if 5.25"
0080	0024 8E38		C *R8+, *R8+	Bump pointer by 4 to 8" values
0081	0026 1D04		SBD SIZED	Set 8" drive
0082		*		
0083		*	RE-CALIBRATE DRIVE	
0084		*		
0085	0028 C289	MINI	MOV R9, R10	Copy FDC pointer
0086	002A 0207		LI R7, 7	Load byte count
	002C 0007			
0087	002E DEB1	LOOP1	MOVB *R1+, *R10+	Send byte
0088	0030 0607		DEC R7	Done ?
0089	0032 16FD		JNE LOOP1	No, loop
0090	0034 D6A8		MOVB @2(R8), *R10	Send last byte
	0036 0002			
0091	0038 0696		BL *R6	Call wait

0093	*			
0094	*		CHECK DENSITY JUMPER	
0095	*			
0096	003A	1F03	TB DNSTY	Read jumper
0097	003C	1603	JNE SD	Skip if single density
0098	003E	0588	INC R8	Bump pointer to double d value
0099	0040	0201	LI R1,DOUBLE	Change pointer for double d
	0042	0106		
0100	0044	C289	SD MOV R9,R10	Copy FDC pointer
0101	0046	DEB1	MOVB *R1+,*R10+	Send AFAS command
0102	0048	DEB1	MOVB *R1+,*R10+	
0103	004A	DEB1	MOVB *R1+,*R10+	
0104	004C	D6B1	MOVB *R1+,*R10	
0105	*			
0106	*		SET UP DMAC	
0107	*			
0108	004E	020C	LI R12,DMAC	Load cru base of DMAC
	0050	01C0		
0109	0052	1D1F	SBO 31	Reset
0110	0054	1D19	SBO 25	Select channel 1
0111	0056	1E14	SBZ 20	Select byte xfers
0112	0058	1E13	SBZ 19	Set memory write
0113	005A	3020	LDCR @ERAM,0	Send stop address
	005C	00E2		
0114	005E	1D10	SBO 16	Enable load of start address
0115	0060	0202	LI R2,TMPBUF	Load buffer start
	0062	0000		
0116	0064	3002	LDCR R2,0	Send start address
0117	0066	1D11	SBO 17	Enable channel
0118	*			
0119	*		SET UP FDC FOR READ	
0120	*			
0121	0068	C289	MOV R9,R10	copy FDC pointer
0122	006A	0207	LI R7,8	Load byte count
	006C	0008		
0123	006E	DEB1	LOOP2 MOVB *R1+,*R10+	Send byte
0124	0070	0607	DEC R7	Done ?
0125	0072	16FD	JNE LOOP2	No , loop
0126	0074	C289	MOV R9,R10	Copy FDC pointer
0127	0076	DEB1	MOVB *R1+,*R10+	Send cmd byte
0128	0078	DE98	MOVB *R8,*R10+	Send encode & xfer rate
0129	007A	0207	LI R7,6	Load byte count
	007C	0006		
0130	007E	DEB1	LOOP3 MOVB *R1+,*R10+	Send byte
0131	0080	0607	DEC R7	Done ?
0132	0082	16FD	JNE LOOP3	No , loop
0133	0084	0696	BL *R6	Call wait

0135	*			
0136	*		CALCULATE CHECKSUM AND VERIFY VALUE	
0137	*			
0138		0086 04C3	CLR R3	Clear
0139		0088 0204	LI R4,SUM-2	Load index
		008A 001C		
0140		008C A0E4	ADD A @TMPBUF(R4),R3	Add entry
		008E 0062		
0141		0090 0644	DECT R4	Decrement index
0142		0092 18FC	JOC ADD	Done ? No , loop
0143		0094 8883	C R3,@SUM(R2)	Checksum ok ?
		0096 001E		
0144		0098 16B3	JNE BOOTY	No , re-try
0145	*			
0146	*		COPY TRACK 0 INTO MEMORY	
0147	*			
0148		009A D660	MOVB @SORD,*R9	Send read cmd
		009C 00FF		
0149		009E DA62	MOVB @LENGTH(R2),@3(R9)	Send no of sectors to xfer
		00A0 0014		
		00A2 0003		
0150		00A4 020C	LI R12,DMAC	Load cru base
		00A6 01C0		
0151		00AB C0E2	MOV @SADDR(R2),R3	Fetch start address
		00AA 0016		
0152		00AC 3003	LDCR R3,0	Send new start address
0153	*			
0154	*		LOAD INTERNAL RAM WITH CODE	
0155	*			
0156		00AE 04CC	CLR R12	Load CRU base of i/o
0157		00B0 0200	LI R0,>1F04	Load "TB 4 " instruction
		00B2 1F04		
0158		00B4 0201	LI R1,>13FE	Load "JEG \$-2" instruction
		00B6 13FE		
0159		00BB 0202	LI R2,>0413	Load "BLWP *R3" instruction
		00BA 0413		
0160		00BC DA69	MOVB @7(R9),@7(R9)	Re-send last byte
		00BE 0007		
		00C0 0007		
0161		00C2 0440	B R0	Execute from internal RAM

0163	*			
0164	*	WAIT FOR INTERRUPT OR TIMEOUT		
0165	*			
0166 00C4 04C0	WAIT	CLR R0		Set timeout count
0167 00C6 0705		SET0 R5		Set count out
0168 00C8 04CC	WAIT1	CLR R12		Load CRU base for i/o
0169 00CA 0600		DEC R0		Timeout ?
0170 00CC 1602		JNE CHKINT		No , check interrupt
0171 00CE 0585		INC R5		Count out ?
0172 00D0 1697		JNE BOOTY		Yes , re-try
0173 00D2 1F04	CHKINT	TB FDCINT		Interrupt active ?
0174 00D4 13F9		JEG WAIT1		No , loop
0175 00D6 D660		MOVB @CINT,*R9		Send CINT
00DB 00E7'				
0176 00DA 9819		CB *R9,@OK		Status ok ?
00DC 00E3'				
0177 00DE 1690		JNE BOOTY		No , re-try
0178 00E0 045B		RT		

0180	*				
0181	*		MISC DATA		
0182	*				
0183	00E2	FFFF	ERAM	DATA >FFFF	Last dram location
0184		00E3	OK	EQU \$-1	Ok status is hex FF
0185	*				5.25" encode & drive rates
0186	00E4	50	ENCODE	BYTE >50,>A1	FM 125K BPS , MFM 250K BPS
0187	00E6	A5		BYTE >A5	Drives 0,1=Rate B 2,3=Rate A
0188	00E7	00	CINT	BYTE >00	
0189	*				8" encode & drive rates
0190	00E8	60		BYTE >60,>B1	FM 250K BPS , MFM 500K BPS
0191	00EA	5A		BYTE >5A	Drives 0,1=Rate A 2,3=Rate B
0192	*				
0193	*		SINGLE DENSITY TABLE		
0194	*				
0195	00EB	20	TABLE	BYTE >20	RSTC
0196	00EC	41		BYTE >41	RDAR
0197	00ED	15		BYTE >15	A. head step time 10 ms
0198	00EE	11		BYTE >11	A. head settle time 8 ms
0199	00EF	47		BYTE >47	A. head load time 35 ms
0200	00F0	64		BYTE >64	B. head step time 50 ms
0201	00F1	46		BYTE >46	B. head settle time 35 ms
0202	00F2	00		BYTE >00	B. head load time 0 ms
0203	*				entry supplied by code
0204	00F3	C0		BYTE >C0	AFAS
0205	00F4	FF		BYTE >FF	fill
0206	00F5	00		BYTE >00	sync
0207	00F6	06		BYTE >06	number of sync before AM
0208	00F7	30		BYTE >30	AIDA
0209	00F8	C7		BYTE >C7	id am clock pattern
0210	00F9	FE		BYTE >FE	id am data pattern
0211	00FA	00		BYTE >00	id byte 0 .. track number
0212	00FB	00		BYTE >00	id byte 1 .. side
0213	00FC	01		BYTE >01	id byte 2 .. sector
0214	00FD	00		BYTE >00	id byte 3 .. record length
0215	00FE	A5		BYTE >A5	crc , 1 am , byte 2, 5 bytes
0216	00FF	88	SORD	BYTE >88	SORD
0217	*				entry supplied by code
0218	0100	81		BYTE >81	record length
0219	0101	01		BYTE >01	no lwcir , no pre , 1 sector
0220	0102	00		BYTE >00	track 0
0221	0103	C7		BYTE >C7	data am clock pattern
0222	0104	FB		BYTE >FB	data am data pattern
0223	0105	6B		BYTE >6B	1 am , xfer , crc , 11 bytes
0224	*				
0225	*		DOUBLE DENSITY TABLE		
0226	*				
0227	0106	C0	DOUBLE	BYTE >C0	AFAS
0228	0107	FF		BYTE >FF	fill
0229	0108	00		BYTE >00	sync
0230	0109	0C		BYTE >0C	number of sync before am
0231	010A	30		BYTE >30	AIDA
0232	010B	0A		BYTE >0A	id am clock pattern
0233	010C	A1		BYTE >A1	id am data pattern
0234	010D	00		BYTE >00	id byte 0 .. track number
0235	010E	00		BYTE >00	id byte 1 .. side
0236	010F	01		BYTE >01	id byte 2 .. sector
0237	0110	01		BYTE >01	id byte 3 .. record length
0238	0111	E7		BYTE >E7	crc , 3 am , byte 2, 7 bytes
0239	0112	88		BYTE >88	SORD

0240		*	.	entry supplied by code
0241 0113	03		BYTE >03	record length
0242 0114	01		BYTE >01	no lwcir , no pre , 1 sector
0243 0115	00		BYTE >00	track 0
0244 0116	0A		BYTE >0A	data am clock pattern
0245 0117	A1		BYTE >A1	data am data pattern
0246 0118	F6		BYTE >F6	3 am , xfer , crc , 22 bytes
0247			END	
NO ERRORS,		NO WARNINGS		

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.SFSN
OBJECT ACCESS NAME= ADHOC.OBJ.SFSN
LISTING ACCESS NAME= ADHOC.LST.SFSN
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

```

0002          IDT 'SFSN'
0003          *
0004          *          SFSN          ; SEARCH FOR STATEMENT £
0005          *
0006 2F80 ERROR EQU >2F80          ; XOP XX,14 (ERROR CALL)
0007          REF SLT          ; STATEMENT LOCATION TABLE
0008          *
0009          *          SEARCH IN STATEMENT-LOCATION-TABLE FOR THE LINE NUMBE
0010          *          THE SLT IS ORGANIZED AS FOLLOWS FOR STATEMENTS 100,20
0011          *
0012          *          SLT          300
0013          *          PLC
0014          *
0015          *          200
0016          *          PLC
0017          *
0018          *          100
0019          *          PLC
0020          *
0021          *          0
0022          *
0023          * CALLING SEQUENCE:
0024          *
0025          *          BL @SFSN
0026          *
0027          *          IN  - R1 = £
0028          *          OUT - R8 = PLC
0029          *
0030          *          NORMAL EXIT - RETURN
0031          *          ERROR EXIT TO ERROR ROUTINE
0032          *
0033          * EXCEPTIONS AND CONDITIONS:
0034          *
0035          *          ERROR 13 RESULTS IF LINE IS NOT FOUND
  
```

```
0037      * ENTRY POINT:
0038      *
0039      DEF  SFSN
0040      *
0041 0000 C220 SFSN  MOV  @SLT,R8      ; START AT TOP
      0002 0000
0042      *
0043 0004 C338 SFSN1 MOV  *R8+,R12    ; DONE?
0044 0006 1305      JEQ  SFSN3      ; Y
0045 0008 804C      C    R12,R1     ; FOUND?
0046 000A 1302      JEQ  SFSN2      ; Y
0047 000C 05C8      INCT R8         ; N, CONTINUE
0048 000E 10FA      JMP  SFSN1
0049      *
0050 0010 045B SFSN2 B    *R11      ; RETURN
0051      *
0052 0012 2F8D SFSN3 DATA ERROR+13 ; NO SUCH LINE £
0053      END
NO ERRORS,      NO WARNINGS
```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.SWAP
OBJECT ACCESS NAME= ADHOC.OBJ.SWAP
LISTING ACCESS NAME= ADHOC.LST.SWAP
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

LINE	KEY	NAME
------	-----	------

0010	A	ADHOC.SRC.IOBITS =>ADHOC.SRC.IOBITS
------	---	--


```
0002          IDT  'SWAP '  
0003          *  
0004          DEF  SWAPY  
0005          *  
0006          REF  VMODE,NLINO,SWPTBL,SENDAD  
0007          *  
0008      2F80  ERROR  EQU  >2F80  
0009      2FA0  ERROR2 EQU  ERROR+>20  
0010          COPY ADHOC.SRC.10BITS
```

```

A0002      *
A0003      * THIS MODULE IS INCLUDED IN SOURCE MODULES
A0004      * BY USING A 'COPY' DIRECTIVE.
A0005      *
A0006      *
A0007      *****
A0008      *
A0009      * CRU DEFINITIONS
A0010      *
A0011      *****
A0012      0000 PIO EQU >0000 PARALLEL I/O
A0013      * PARALLEL I/O - READ BITS
A0014      0000 KDS EQU PIO+0 KEYBOARD DATA STROBE
A0015      * EQU PIO+1 UNUSED
A0016      0002 D1$SIZ EQU PIO+2 DS01 SIZE (1=8",0=5.25")
A0017      0003 D1$DEN EQU PIO+3 DS01 DENSITY (0=SD,1=DD)
A0018      0004 FDCINT EQU PIO+4 FDC INTERRUPT-
A0019      0005 KBDINT EQU PIO+5 KBD INTERRUPT-
A0020      0006 VDPINT EQU PIO+6 VDP INTERRUPT-
A0021      0007 BUSINT EQU PIO+7 BUS INTERRUPT-
A0022      0008 KEYBRD EQU PIO+8 KEYBOARD DATA (8 BITS)
A0023      * PARALLEL I/O - WRITE BITS
A0024      0000 CLKLED EQU PIO+0 CLOCK LED
A0025      0001 KBDACK EQU PIO+1 KEYBOARD ACKNOWLEDGE-
A0026      0002 BUSACK EQU PIO+2 BUS INTERRUPT RESET-
A0027      0003 BTENBL EQU PIO+3 BUS TIMEOUT FLAG ENABLE
A0028      0004 DRVSIZ EQU PIO+4 DRIVE SIZE (1=8",0=5.25")
A0029      0005 ROMON EQU PIO+5 0=ROM ON,1=ROM OFF
A0030      0006 BELLON EQU PIO+6 BELL ENABLE BIT
A0031      * EQU PIO+7 UNUSED
A0032      *
A0033      * EQU >0020 UNUSED
A0034      0040 EIA02 EQU >0040 PRINTER HARDWARE BASE ADDRESS
A0035      * EQU >0060 UNUSED
A0036      * EQU >0080 UNUSED
A0037      * EQU >00A0 UNUSED
A0038      00C0 CASS02 EQU >00C0 CASSETTE HARDWARE BASE ADDRESS
A0039      00E0 DMAC EQU >00E0 DMAC HARDWARE BASE ADDRESS
A0040      *
A0041      * RESERVED OFF-BOARD CRU LOCATIONS
A0042      0200 PRMPOP EQU >200 PROM PROGRAMMER BOARD
A0043      * READ BITS
A0044      * EQU PRMPOP+0
A0045      * EQU PRMPOP+1
A0046      * EQU PRMPOP+2
A0047      * EQU PRMPOP+3
A0048      0204 PRORDY EQU PRMPOP+4
A0049      0205 PGM EQU PRMPOP+5
A0050      0206 PGMFUL EQU PRMPOP+6 * TEST
A0051      0207 V30 EQU PRMPOP+7
A0052      0208 EDATA EQU PRMPOP+8
A0053      * WRITE BITS
A0054      0200 PRORST EQU PRMPOP+0
A0055      0201 EPTYPE EQU PRMPOP+1
A0056      0204 VCCON EQU PRMPOP+4
A0057      *PGM EQU PRMPOP+5
A0058      0206 PROERR EQU PRMPOP+6
A0059      * EQU PRMPOP+7
A0060      *EDATA EQU PRMPOP+8
A0061      0210 EPADR EQU PRMPOP+16

```

```

A0062      *
A0063      *   CENTRONICS PARALLEL PRINTER PORT
A0064      *
A0065      *   BIT       0       1       2       3       4       5       6       7       8       9
A0066      *   READ
A0067      *   WRITE    D7     D6     D5     D4     D3     D2     D1     D0     DS
A0068      *                   (LSB)                                (MSB)
A0069      *
A0070      0400 PPRINT EQU  >400
A0071      *
A0072      0408 PPDS  EQU  PPRINT+8          DATA STROBE  +_+
A0073      *
A0074      *
A0075      0409 PPBUSY EQU  PPRINT+9          BUSY          +-----+
A0076      *
A0077      *****
A0078      *
A0079      *   MEMORY MAPPED I/O DEFINITIONS
A0080      *
A0081      *****
A0082      F100 MAPPER EQU  >F100          MEMORY MAPPER LOCATION
A0083      F100 M$REG0 EQU  MAPPER+0        MAPPER REGISTERS 0..15
A0084      F102 M$REG1 EQU  MAPPER+2
A0085      F104 M$REG2 EQU  MAPPER+4
A0086      F106 M$REG3 EQU  MAPPER+6
A0087      F108 M$REG4 EQU  MAPPER+8
A0088      F10A M$REG5 EQU  MAPPER+10
A0089      F10C M$REG6 EQU  MAPPER+12
A0090      F10E M$REG7 EQU  MAPPER+14
A0091      F110 M$REG8 EQU  MAPPER+16
A0092      F112 M$REG9 EQU  MAPPER+18
A0093      F114 M$REGA EQU  MAPPER+20
A0094      F116 M$REGB EQU  MAPPER+22
A0095      F118 M$REGC EQU  MAPPER+24
A0096      F11A M$REGD EQU  MAPPER+26
A0097      F11C M$REGE EQU  MAPPER+28
A0098      F11E M$REGF EQU  MAPPER+30
A0099      *
A0100      F120 VDP    EQU  >F120
A0101      F120 VRAM   EQU  VDP+0          VDP VRAM ACCESS ADDRESS
A0102      F121 VDPREG EQU  VDP+1          VDP REGISTER ACCESS ADDRESS
A0103      *
A0104      *   TEXT / GRAPHICS 1 MODE STORAGE
A0105      0400 NTBA   EQU  >400          NAME TABLE
A0106      0700 CTBA   EQU  >700          COLOUR TABLE
A0107      0800 PGBA   EQU  >800          PATTERN GENERATOR TABLE
A0108      0780 SNTBA  EQU  >780          SPRITE NAME TABLE
A0109      0000 SPGBA  EQU  >000          SPRITE PATTERN GENERATOR TBL.
A0110      *   GRAPHICS 2 MODE STORAGE
A0111      1800 NTBA1  EQU  >1800         NAME TABLE
A0112      2000 CTBA1  EQU  >2000         COLOUR TABLE
A0113      0000 PGTBA1 EQU  >0000         PATTERN GENERATOR TABLE
A0114      1B00 SNTBA1 EQU  >1B00         SPRITE NAME TABLE
A0115      3800 SPGBA1 EQU  >3800         SPRITE PATTERN GENERATOR TBL.
A0116      *
A0117      F140 FDC    EQU  >F140          TMS9909 FLOPPY DISC CONTROLLER
A0118      *
0011      *
0012      *   THIS ROUTINE READS THE COLOUR TABLE OF
0013      *   THE VDP AND SWAPS THE FGND & BGND COLOURS

```

```

0014      *      WITH THE COLOUR SET UP IN THE 16 BYTE
0015      *      COLOUR TABLE 'SWPTBL'
0016      *      THIS TABLE IS COPIED INTO INTERNAL RAM FOR SPEED
0017      *
0018 0000 C020  SWAPY  MOV  @VMODE,R0      GRAPHICS MODE?
      0002 0000
0019 0004 1602      JNE  SWAP1          Y, EVERYTHING OK
0020      *
0021 0006 2FA0      DATA ERROR2,48      ILLEGAL IN CURRENT MODE
0022      *
0023      *      COPY THE SWAP TABLE INTO INTERNAL RAM
0024      *      TO SPEED UP THE ROUTINE (12K ACCESSSES TO IT !!)
0025      *
0026 000A 0201  SWAP1  LI   R1,SWPTBL      REF THE SWAP TABLE
      000C 0000
0027 000E 0202      LI   R2,>F000      PUT IT IN INTERNAL RAM
      0010 F000
0028 0012 0703      SETD  R3          SET COUNT FOR 16 BYTES
0029 0014 CCB1  SWAP1A MOV  *R1+,*R2+    COPY IT
0030 0016 0923      SRL  R3,2          DONE?
0031 0018 16FD      JNE  SWAP1A        N, LOOP
0032      *
0033 001A C048      MOV  R8,R1          SAVE PBC
0034 001C 0208      LI   R8,CTBA1      REF THE COLOUR TABLE
      001E 2000
0035 0020 0202      LI   R2,>1800      SET UP TABLE SIZE
      0022 1800
0036      *
0037 0024 06A0  SWAP2  BL   @SENDAD      GIVE VDP THE ADDRESS
      0026 0000
0038 0028 0588  SWAP2A INC  R8          ADJUST R8
0039 002A D0E0  SWAP2B MOVB @VRAM,R3    READ THE BYTE
      002C F120
0040 002E C143      MOV  R3,R5          SAVE FOR LATER
0041 0030 0983      SRL  R3,8          PUT IT IN LSB
0042 0032 C103      MOV  R3,R4          SAVE IT
0043 0034 0243      ANDI R3,>000F      ISOLATE BACKGROUND COLOUR
      0036 000F
0044 0038 D023      MOVB @>F000(3),R0   GET ITS NEW COLOUR
      003A F000
0045 003C 0940      SRL  R0,4          POSITION IT
0046 003E 0944      SRL  R4,4          POSITION FGND COLOUR CODE
0047 0040 D024      MOVB @>F000(4),R0   GET ITS NEW COLOUR
      0042 F000
0048 0044 0A40      SLA  R0,4          POSITION COLOUR BYTE
0049 0046 9140      CB   R0,R5          SAME?
0050 0048 13EF      JEQ  SWAP2A        Y, DONT UPDATE
0051 004A 0228      AI   R8,>4000-1    SET WRITE BIT
      004C 3FFF
0052 004E 06A0      BL   @SENDAD      GIVE ADDRESS TO VDP
      0050 0026
0053 0052 0228      AI   R8,-(>4000-1) SET VDP FOR READ OF NEXT BYTE
      0054 C001
0054 0056 D800      MOVB R0,@VRAM      WRITE IT BACK TO THE VDP
      0058 F120
0055 005A 0602      DEC  R2          DONE?
0056 005C 16E3      JNE  SWAP2        N, LOOP
0057      *
0058 005E C201      MOV  R1,R8          RESTORE PBC
0059 0060 0460      B    @NLINO        NEXT LINE

```

0062 0000

NO ERRORS, NO WARNINGS

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.RELOPS
OBJECT ACCESS NAME= ADHOC.OBJ.RELOPS
LISTING ACCESS NAME= ADHOC.LST.RELOPS
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

```
0002          IDT  'RELOPS'
0003          *
0004          *
0005          *
0006          DEF  ANDF          ; AND FUNCTION
0007          DEF  ORF           ; OR FUNCTION
0008          DEF  NOTF          ; NOT FUNCTION
0009          *
0010          DXOP CLEAR,B      ; CLEAR FPAC
0011          *
0012          REF  FPAC,FPAC2    ; FLOATING POINT ACCUMULATOR
0013          *
0014          * ABSTRACT:
0015          *
0016          *     PERFORM 'AND' FUNCTION ON TWO OPERANDS
0017          *     AND RETURN A VALUE OF 1 OR 0 ACCORDINGLY.
0018          *
0019          *
0020          * CALLING SEQUENCE:
0021          *
0022          *     BL @ANDF
0023          *
0024          *     IN (R1) = OPERAND 1
0025          *     (R2) = OPERAND 2
0026          *
0027          *     OUT (R2) = 1 OR 0 IN FPAC
0028          *
0029          *     NORMAL EXIT - RETURN
0030          *
```

```

0032      *
0033      *AND OPERATOR
0034      *
0035 0000 2E00  ANDF  CLEAR 0          ; CLEAR FPAC
0036 0002 C0F1      MOV  *R1+,R3      ; INTEGER?
0037 0004 1602      JNE  ANDF1        ; N
0038 0006 C0D1      MOV  *R1,R3       ; Y, GET INTEGER
0039 0008 1306      JEQ  ANDF3        ; O, RESULT=0
0040      *
0041 000A C0F2  ANDF1  MOV  *R2+,R3    ; INTEGER?
0042 000C 1602      JNE  ANDF2        ; RESULT=1
0043 000E C0D2      MOV  *R2,R3       ; Y
0044 0010 1302      JEQ  ANDF3        ; O, RESULT=0
0045      *
0046 0012 05A0  ANDF2  INC  @FPAC2     ; RESULT=1
      0014 0000
0047      *
0048 0016 0202  ANDF3  LI   R2,@FPAC
      0018 0000
0049 001A 045B      RT
    
```



```

0051      * ABSTRACT:
0052      *
0053      *      PERFORM AN 'OR' OPERATION ON THE
0054      *      2 OPERANDS AND RETURN A VALUE OF
0055      *      1 FOR TRUE AND 0 FOR FALSE.
0056      *
0057      * CALLING SEQUENCE:
0058      *
0059      *      BL @ORF
0060      *
0061      *      IN (R1) = OPERAND 1
0062      *      (R2) = OPERAND 2
0063      *
0064      *      OUT (R2) = 1 OR 0 IN FPAC
0065      *
0066      *      NORMAL EXIT - RETURN
0067      *
0068      *
0069      *OR OPERATOR
0070      *
0071 001C 2E00  ORF      CLEAR 0                ; CLEAR FPAC
0072 001E C0F1      MOV   *R1+,R3
0073 0020 16F8      JNE   ANDF2                ; RESULT=1
0074 0022 C0D1      MOV   *R1,R3
0075 0024 16F6      JNE   ANDF2
0076 0026 C0F2      MOV   *R2+,R3
0077 0028 16F4      JNE   ANDF2                ; RESULT=1
0078 002A C0D2      MOV   *R2,R3
0079 002C 16F2      JNE   ANDF2
0080 002E 10F3      JMP   ANDF3                ; RESULT=0
    
```

```

0082      * ABSTRACT:
0083      *
0084      *      PERFORM A 'NOT' FUNCTION ON THE OPERAND
0085      *
0086      * CALLING SEQUENCE:
0087      *
0088      *      BL @NOTF
0089      *
0090      *      IN (R1) = OPERAND
0091      *
0092      *      OUT (R2) = 1 OR 0 IN FPAC
0093      *
0094      *
0095      0031' NOTF    EQU    $+1          ; INDICATE AS UNARY OPERATION
0096 0030 2E00 NOT    CLEAR 0           ; CLEAR FPAC
0097 0032 C0F1      MOV    *R1+,R3
0098 0034 16F0      JNE    ANDF3
0099 0036 C0D1      MOV    *R1,R3
0100 0038 13EC      JEQ    ANDF2
0101 003A 10ED      JMP    ANDF3
0102      END
NO ERRORS,      NO WARNINGS
    
```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.EDIT
OBJECT ACCESS NAME= ADHOC.OBJ.EDIT
LISTING ACCESS NAME= ADHOC.LST.EDIT
ERROR ACCESS NAME=
OPTIONS= BUNLST, DUNLST, MUNLST
MACRO LIBRARY PATHNAME=

LINE	KEY	NAME
------	-----	------

0130	A	ADHOC.SRC.IOBITS =>ADHOC.SRC.IOBITS
------	---	--

0002 IDT 'EDIT'

0003 *
 0004 * COMMANDS :-
 0005 *
 0006 * RUN
 0007 * SIZ(E)
 0008 * CON(TINUE)
 0009 * MON(ITOR)
 0010 * <SPARE>
 0011 * <SPARE>
 0012 * <SPARE>
 0013 * <SPARE>
 0014 * <SPARE>
 0015 * <SPARE>
 0016 * <SPARE>
 0017 *
 0018 *

	FUNCTIONS	DELIMITERS	OPERATORS	VARIABLES
0019	* *01-1A FNx	*38 TO	*4E OR	*62 -1
0020		*39 TAB	*4F LOR	*63 0
0021	* 1B ABS	*3A STEP	*50 AND	*64 1
0022	* 1C ADR	*3B THEN	*51 LAND	*65 2
0023	* 1D ASC	*3C :	*52 NOT	*66 3
0024	* 1E ATN	*3D @	*53 LNOT	*67 4
0025	* 1F COS	*3E £	*54 LXOR	*68 5
0026	* 20 EXP	*3F ,	*55 ==	*69 6
0027	* 21 FRA	*40 ;	*56 =	*6A 7
0028	* 22 INT	*41 ?	*57 >	*6B 8
0029	* 23 LOG	*42 %	*58 >=	*6C 9
0030	* 24 KEY	*43 \$	*59 <	*6D 2B INT.
0031	* 25 SIN	*44 "	*5A <=	*6E 2B HEX
0032	* 26 SQR	*45 '	*5B <>	*6F 6B FP.
0033	* 27 SYS	*46 \ (NU)	*5C -	*70 DUMMY
0034	* 28 TIC	*47 !	*5D +	*71 DUMMY
0035	* 29 SGN	*48 &	*5E /	*72 DUMMY
0036	* 2A BIT	*49 _ (NU)	*5F *	*73 RND
0037	* 2B CRB	*4A [*60 ^	74-FF USER
0038	* 2C CRF	*4B]	*61 UN. -	
0039	* 2D MEM	*4C (
0040	* 2E MWD	*4D)		
0041	* 2F LEN			
0042	* 30 MCH			
0043	* 31 POS			
0044	* 32 COL			
0045	* 33 MOD			
0046	* 34			
0047	* 35			
0048	* 36			
0049	* 37			
0050	*			
0051	*			
0052	*			
0053	* () = SPECIAL		* = TABLE POSITION FIXED	

	*	STATEMENTS	
	*	=====	
0055	*		
0056	*		
0057	*		
0058	*		
0059	*	00	20 TEXT 40 ON
0060	*	01 GOTO*	21 WAIT 41 IF
0061	*	02 GOSUB*	22 CHAR 42 DEF
0062	*	03 ELSE*	23 NUMBER 43 NEW
0063	*	04 REM*	24 LIST 44 END
0064	*	05 FOR*	25 RENUM 45 (?) (PRINT)
0065	*	06 (LET)*	26 SPRITE 46 (*) (EXTEND)
0066	*	07 DATA	27 SHAPE 47 BIT
0067	*	08 NEXT	28 SPUT 48 CRB
0068	*	09 ERROR	29 SGET 49 CRF
0069	*	0A PRINT	2A BOOT 4A MEM
0070	*	0B CALL	2B SWAP 4B MWD
0071	*	0C LOAD	2C 4C
0072	*	0D INPUT	2D MOTOR 4D
0073	*	0E READ	2E 4E
0074	*	0F RESTOR	2F 4F
0075	*	10 RETURN	30
0076	*	11 STOP	31
0077	*	12 UNIT	32
0078	*	13 TIME	33
0079	*	14 SAVE	34
0080	*	15 BASE	35
0081	*	16 ESCAPE	36
0082	*	17 NOESC	37
0083	*	18 RANDOM	38
0084	*	19 BAUD	39 MAG
0085	*	1A ENTER	3A TDF
0086	*	1B PLOT	3B TON
0087	*	1C UNPLOT	3C POP
0088	*	1D COLOUR	3D DIM
0089	*	1E PURGE	3E LET
0090	*	1F GRAPH	3F () (PRINT)
0091	*		
0092	*	() = SPECIAL	* = TABLE POSITION FIXED

```

0094      *
0095      *
0096      *
0097      DEF  EDIT, LST, LSTL, MODEOK, EMV
0098      DEF  LCN1, LCN2, LCN3, LCN4
0099      DEF  LNXT, LDEF, EDIT1
0100      DEF  RNGOTO, RNGSUB, RNRSTR
0101      DEF  RNON, RNINP, RNERR, RNREM, RNELSE, RNIF
0102      *
0103      REF  EDTMP
0104      REF  AINC, ESCFLG
0105      REF  DEBUG$, FTM$
0106      REF  B2A, B2O, B01
0107      REF  BUS, DDM, DLC, EBP
0108      REF  CKEX, CVDB2O, C1
0109      REF  CVDIFZ, CVDIZ, EDER
0110      REF  FNS, GSC, GSS, LNUM
0111      REF  FOR2, JMPRO, R8STOR
0112      REF  FPAC, SLT, SLN, IOB
0113      REF  LINE2
0114      REF  MODE, NVD, NVS, EFLG
0115      REF  RUNP, SIZE
0116      REF  SSP, UFT, VDT, VNT
0117      REF  TYPC$, TYPBE$
0118      REF  HALTO$, DCNT, GOS2
0119      REF  CRLF, NINE, RENCMP
0120      REF  HOUT
0121      DXOP  EVFIX, 11
0122      DXOP  OUTFP, 12
0123      DXOP  OUTINT, 13
0124      *
0125      2F80  ERROR  EQU  >2F80
0126      2FA0  ERROR2 EQU  ERROR+>20
0127      1600  SYN    EQU  >1600          TAPE SYNC CHARACTER
0128      0200  STX    EQU  >0200          TAPE START OF TEXT CHARACTER
0129      0300  ETX    EQU  >0300          TAPE END OF TEXT CHARACTER
0130      COPY  ADHOC. SRC. IOBITS

```

```

A0002      *
A0003      * THIS MODULE IS INCLUDED IN SOURCE MODULES
A0004      * BY USING A 'COPY' DIRECTIVE.
A0005      *
A0006      *
A0007      *****
A0008      *
A0009      * CRU DEFINITIONS
A0010      *
A0011      *****
A0012      0000 PIO EQU >0000 PARALLEL I/O
A0013      * PARALLEL I/O - READ BITS
A0014      0000 KDS EQU PIO+0 KEYBOARD DATA STROBE
A0015      * EQU PIO+1 UNUSED
A0016      0002 D1$SIZ EQU PIO+2 DS01 SIZE (1=8",0=5.25")
A0017      0003 D1$DEN EQU PIO+3 DS01 DENSITY (0=SD,1=DD)
A0018      0004 FDCINT EQU PIO+4 FDC INTERRUPT-
A0019      0005 KBDINT EQU PIO+5 KBD INTERRUPT-
A0020      0006 VDPINT EQU PIO+6 VDP INTERRUPT-
A0021      0007 BUSINT EQU PIO+7 BUS INTERRUPT-
A0022      0008 KEYBRD EQU PIO+8 KEYBOARD DATA (8 BITS)
A0023      * PARALLEL I/O - WRITE BITS
A0024      0000 CLKLED EQU PIO+0 CLOCK LED
A0025      0001 KBDACK EQU PIO+1 KEYBOARD ACKNOWLEDGE-
A0026      0002 BUSACK EQU PIO+2 BUS INTERRUPT RESET-
A0027      0003 BTENBL EQU PIO+3 BUS TIMEOUT FLAG ENABLE
A0028      0004 DRVSIZ EQU PIO+4 DRIVE SIZE (1=8",0=5.25")
A0029      0005 ROMON EQU PIO+5 0=ROM ON,1=ROM OFF
A0030      0006 BELLON EQU PIO+6 BELL ENABLE BIT
A0031      * EQU PIO+7 UNUSED
A0032      *
A0033      * EQU >0020 UNUSED
A0034      0040 EIA02 EQU >0040 PRINTER HARDWARE BASE ADDRESS
A0035      * EQU >0060 UNUSED
A0036      * EQU >0080 UNUSED
A0037      * EQU >00A0 UNUSED
A0038      00C0 CASS02 EQU >00C0 CASSETTE HARDWARE BASE ADDRESS
A0039      00E0 DMAC EQU >00E0 DMAC HARDWARE BASE ADDRESS
A0040      *
A0041      * RESERVED OFF-BOARD CRU LOCATIONS
A0042      0200 PRMPOP EQU >200 FROM PROGRAMMER BOARD
A0043      * READ BITS
A0044      * EQU PRMPOP+0
A0045      * EQU PRMPOP+1
A0046      * EQU PRMPOP+2
A0047      * EQU PRMPOP+3
A0048      0204 PRORDY EQU PRMPOP+4
A0049      0205 PGM EQU PRMPOP+5
A0050      0206 PGMPUL EQU PRMPOP+6 * TEST
A0051      0207 V30 EQU PRMPOP+7
A0052      0208 EDATA EQU PRMPOP+8
A0053      * WRITE BITS
A0054      0200 PRORST EQU PRMPOP+0
A0055      0201 EPTYPE EQU PRMPOP+1
A0056      0204 VCCON EQU PRMPOP+4
A0057      *PGM EQU PRMPOP+5
A0058      0206 PROERR EQU PRMPOP+6
A0059      * EQU PRMPOP+7
A0060      *EDATA EQU PRMPOP+8
A0061      0210 EPADR EQU PRMPOP+16
    
```

```

A0062      *
A0063      *   CENTRONICS PARALLEL PRINTER PORT
A0064      *
A0065      *   BIT       0       1       2       3       4       5       6       7       8       9
A0066      *   READ
A0067      *   WRITE    D7      D6      D5      D4      D3      D2      D1      D0      D5
A0068      *                   (LSB)                                (MSB)
A0069      *
A0070      0400 PPRINT EQU  >400
A0071      *
A0072      0408 PPDS  EQU  PPRINT+8          DATA STROBE  +_+
A0073      *
A0074      *
A0075      0409 PPBUSY EQU  PPRINT+9          BUSY          +-----+
A0076      *
A0077      *****
A0078      *
A0079      *   MEMORY MAPPED I/O DEFINITIONS
A0080      *
A0081      *****
A0082      F100 MAPPER EQU  >F100          MEMORY MAPPER LOCATION
A0083      F100 M$REG0 EQU  MAPPER+0        MAPPER REGISTERS 0..15
A0084      F102 M$REG1 EQU  MAPPER+2
A0085      F104 M$REG2 EQU  MAPPER+4
A0086      F106 M$REG3 EQU  MAPPER+6
A0087      F108 M$REG4 EQU  MAPPER+8
A0088      F10A M$REG5 EQU  MAPPER+10
A0089      F10C M$REG6 EQU  MAPPER+12
A0090      F10E M$REG7 EQU  MAPPER+14
A0091      F110 M$REG8 EQU  MAPPER+16
A0092      F112 M$REG9 EQU  MAPPER+18
A0093      F114 M$REGA EQU  MAPPER+20
A0094      F116 M$REGB EQU  MAPPER+22
A0095      F118 M$REGC EQU  MAPPER+24
A0096      F11A M$REGD EQU  MAPPER+26
A0097      F11C M$REGE EQU  MAPPER+28
A0098      F11E M$REGF EQU  MAPPER+30
A0099      *
A0100      F120 VDP    EQU  >F120
A0101      F120 VRAM   EQU  VDP+0          VDP VRAM ACCESS ADDRESS
A0102      F121 VDPREG EQU  VDP+1          VDP REGISTER ACCESS ADDRESS
A0103      *
A0104      *   TEXT / GRAPHICS 1 MODE STORAGE
A0105      0400 NTBA   EQU  >400          NAME TABLE
A0106      0700 CTBA   EQU  >700          COLOUR TABLE
A0107      0800 PGBA   EQU  >800          PATTERN GENERATOR TABLE
A0108      0780 SNTBA  EQU  >780          SPRITE NAME TABLE
A0109      0000 SPGBA  EQU  >000          SPRITE PATTERN GENERATOR TBL.
A0110      *   GRAPHICS 2 MODE STORAGE
A0111      1800 NTBA1  EQU  >1800          NAME TABLE
A0112      2000 CTBA1  EQU  >2000          COLOUR TABLE
A0113      0000 PGTBA1 EQU  >0000          PATTERN GENERATOR TABLE
A0114      1B00 SNTBA1 EQU  >1B00          SPRITE NAME TABLE
A0115      3800 SPGBA1 EQU  >3800          SPRITE PATTERN GENERATOR TBL.
A0116      *
A0117      F140 FDC     EQU  >F140          TMS9909 FLOPPY DISC CONTROLLER
A0118      *

```



```

0132      *
0133      * CONTINUE COMMAND
0134      *
0135 0000 0720  CONT   SET0 @MODE                SET TO RUN
          0002 0000
0136 0004 0000      DATA TYP0$                OUT CRLF
0137 0006 0060      MOV  @SLN,R1                GET LINE £
          0008 0000
0138 000A 0460      B    @GDS2
          000C 0000
0139      *
0140      * CHECK TO SEE THAT WE ARE IN KEYBOARD MODE
0141      *
0142 000E 0820  MODEOK MOV  @MODE,@MODE          ; ARE WE RUNNING
          0010 0002'
          0012 0010'
0143 0014 1601      JNE  ERR48                    ; Y, NOT ALLOWED
0144 0016 045B      RT
0145      *
0146 0018 2FA0  ERR48  DATA ERROR2,48          ; WRONG MODE
  
```

```

0148      * EDIT INPUT STRING
0149      *
0150      *      R6 = SUBSCRIPT STACK PTR
0151      *      R7 = INPUT BYTE PTR
0152      *      R8 = OUTPUT BYTE PTR
0153      *      R10 = RETURN ADR
0154      *      R12 = BEGINNING OF COMMAND PTR
0155      *      R14 = SAVE FOR LINE COUNTER
0156      *
0157 001C C820 EDIT MOV @NINE,@RENCMP ; COMPRESSED FLAG=9 (FULL)
      001E 0000
      0020 0000
0158 0022 C28B EDIT1 MOV R11,R10 ; SAVE RETURN
0159 0024 C806 MOV R6,@EDTMP ; SAVE LINE COUNT FOR EDER ROUTI
      0026 0000
0160 0028 04E0 CLR @EFLG ; KILL ERROR FLAG
      002A 0000
0161 002C C1E0 MOV @IOB,R7 ; GET I/O PTR
      002E 0000
0162 0030 0206 LI R6,SSP ; GET SUBSCRIPT STACK PTR
      0032 0000
0163 0034 0716 SETO *R6 ; MARK SUBSCRIPT STACK
0164 0036 0208 LI R8,EBP ; GET EDIT BUFFER PTR
      0038 0000
0165 003A C160 MOV @VNT,R5 ; SET UP RESERVED WORDS
      003C 0000
0166 003E 04F5 CLR *R5+ ; CLEAR DUMMY VARIABLES
0167 0040 04F5 CLR *R5+
0168 0042 04F5 CLR *R5+
0169 0044 0200 LI R0,>11D2 ; 'RND'
      0046 11D2
0170 0048 C540 MOV R0,*R5
0171      *
0172      * GET LINE NUMBER
0173      *
0174 004A 0420 BLWP @CVDIZ ; CONVERT LINE NUMBER
      004C 0000
0175 004E 1042 JMP EDER3 ; OVERFLOW
0176 0050 04C1 CLR R1 ; NO NUMBER
0177 0052 C801 MOV R1,@LNUM ; STORE LINE NUMBER
      0054 0000
0178 0056 113E JLT EDER3
0179 0058 1301 JEQ $+4 ; LINE 1
0180 005A 0607 DEC R7 ; Y, BACKUP OVER DELIMITER
0181 005C C000 MOV R0,R0 ; DONE?
0182 005E 1608 JNE ED3 ; N
0183      *
0184      * IF AN EMPTY LINE IS ENTERED IN AUTO-INC MODE THEN
0185      * THE AUTO-INCREMENT MODE IS TERMINATED
0186      *
0187 0060 C2E0 MOV @AINC,R11 ; AUTO-INC MODE ?
      0062 0000
0188 0064 1602 JNE NOAINC ; Y, KILL IT
0189 0066 0460 B @EMVO ; Y, DELETE LINE
      0068 041A'
0190 006A 04E0 NOAINC CLR @AINC ; KILL AUTO-INC FLAG
      006C 0062'
0191 006E 045A B *R10 ; EXIT
0192      *
0193      *DECODE COMMAND WORD

```

```

0194      *
0195 0070 06A0 ED3   BL   @EDGLS           ; GET FIRST LETTER
      0072 05C0'
0196 0074 1038      JMP  EDSCI           ; LET OR ";
0197 0076 C0C7      MOV  R7,R3           ; MARK
0198 0078 C001      MOV  R1,R0
0199 007A 06A0      BL   @EDGL           ; GET SECOND LETTER
      007C 05C8'
0200 007E 1014      JMP  EDLET
0201 0080 A001      A     R1,R0
0202      *
0203      *CHECK FOR TWO LETTER COMMANDS HERE
0204      *
0205 0082 0280      CI    R0,>1920       ; IF?
      0084 1920
0206 0086 1329      JEQ   EDIF           ; Y
0207 0088 0280      CI    R0,>39E0       ; ON?
      008A 39E0
0208 008C 1329      JEQ   EDON           ; Y
0209 008E 06A0      BL   @EDGL           ; N, GET 3RD LETTER
      0090 05C8'
0210 0092 100A      JMP  EDLET           ; LET
0211 0094 A001      A     R1,R0
  
```

```

0213      *SEARCH COMMAND LIST
0214      *
0215 0096 0204      LI    R4,EDSCL      ;Y, GET COMMAND LIST PTR
      0098 0630'
0216      *
0217 009A C074  ED5    MOV    *R4+,R1      ;GET FIRST COMMAND
0218 009C 0911      SRL    R1,1          ;MOVE 2ND WORD INDICATOR INTO C
0219 009E 8040      C      R0,R1          ;FOUND?
0220 00A0 1307      JEQ    ED6            ;MAYBE
0221 00A2 0284      CI     R4,EDCLE       ;N, MORE COMMANDS?
      00A4 06DC'
0222 00A6 1AF9      JL     ED5            ;Y
0223      *
0224 00A8 C1C3  EDLET  MOV    R3,R7          ;RESTORE R7
0225      *
0226 00AA DE20  EDLETA MOV    @LETB,*R8+      ;INSERT >6 (IMPLIED LET)
      00AC 08D0'
0227 00AE 105B      JMP     ED13
0228      *
0229 00B0 C080  ED6    MOV    R0,R2          ;SAVE IN CASE NOT FOUND
0230 00B2 C3C7      MOV    R7,R15
0231 00B4 C024      MOV    @EDCS(4),R0      ;GET SECOND WORD
      00B6 00AA
0232 00B8 1733      JNC     ED9            ;FOUND
0233      *
0234 00BA C140  ED7    MOV    R0,R5
0235 00BC 0245      ANDI    R5,>3E          ;MASK CHARACTER
      00BE 003E
0236 00C0 132F      JEQ     ED9            ;FOUND
0237 00C2 06A0      BL     @EDGL          ;GET LETTER
      00C4 05C8'
0238 00C6 1003      JMP     ED8            ;PROBLEM
0239 00C8 0991      SRL    R1,9            ;POSITION
0240 00CA 8045      C      R5,R1          ;LETTER SAME?
0241 00CC 13F6      JEQ     ED7            ;Y
0242      *
0243 00CE C002  ED8    MOV    R2,R0          ;PROBLEM, CONTINUE TO LOOK
0244 00D0 C1CF      MOV    R15,R7
0245 00D2 10E3      JMP     ED5
0246      *
0247 00D4 06A0  EDER3  BL     @EDER          ;INVALID LINE NUMBER
      00D6 0000
0248 00D8 30      TEXT  '03'
      00D9 33
0249      *
0250 00DA 0204  EDIF    LI     R4,IFB        ;IF ENTRY
      00DC 0082
0251 00DE 1029      JMP     ED10
0252      *
0253 00E0 0204  EDON    LI     R4,ONB        ;ON ENTRY
      00E2 0080
0254 00E4 1026      JMP     ED10
0255      *
0256 00E6 9817  EDSCI  CB     *R7,@B2A      ;'*' ?
      00E8 0000
0257 00EA 160D      JNE     EDSC10         ;N, TRY OTHER SPECIALS
0258 00EC 0587      INC     R7              ;BUMP PTR
0259 00EE 0204      LI     R4,PASBX*128     ;LOAD EXTENDED COMMAND BYTE
      00F0 4600
0260 00F2 DE04      MOV    R4,*R8+          ;STORE IT

```

0261	00F4	D037	ECH#0	MOVB	*R7+,R0	GET CHARACTER
0262	00F6	1305		JEQ	ECH#1	NULL, EXIT LOOP
0263	00F8	9800		CB	R0,@B20	' ' ?
	00FA	0000				
0264	00FC	1302		JEQ	ECH#1	Y, END OF COMMAND
0265	00FE	DE00		MOVB	R0,*R8+	N, COPY OVER BYTE
0266	0100	10F9		JMP	ECH#0	AND LOOP
0267	0102	7E18	ECH#1	SB	*R8,*R8+	END, PUT A NULL
0268	0104	1031		JMP	ED14	AND CONTINUE
0269			*			
0270	0106	9837	EDSCI0	CB	*R7+,@B3F	'?' ?
	0108	0802				
0271	010A	1603		JNE	EDSCI1	N, TRY ','
0272	010C	0204		LI	R4,PQMBX	Y, LOAD '?' ENTRY
	010E	008A				
0273	0110	1010		JMP	ED10	
0274	0112	0607	EDSCI1	DEC	R7	; BACKUP POINTER
0275	0114	9837		CB	*R7+,@B3B	;' ?
	0116	0801				
0276	0118	16C8		JNE	EDLETA	; N
0277	011A	0204		LI	R4,PSCBX	;' ENTRY
	011C	007E				
0278	011F	1009		JMP	ED10	

```

0280      *COMMAND TYPE DONE
0281      *CHECK FOR GOTO OR GOSUB
0282      *
0283 0120 0201 ED9   LI   R1,EDCL      AI   R4,-EDCL (NOT ALLOWED)
      0122 0646'
0284 0124 6101      S    R1,R4      SYSTEM COMMAND?
0285 0126 1505      JGT  ED10      N, CONTINUE TRANSLATION
0286 0128 C000      MOV  R0,R0      Y, COMMAND INSTALLED ?
0287 012A 1301      JEQ  SYSERR     N, SYSTEM ERROR
0288 012C 0450      B    *R0      Y, EXECUTE COMMAND
0289      *
0290 012E 2F80      SYSERR DATA ERROR, -1  SYSTEM ERROR
0291      *
0292 0132 EB20      ED10  SOC  @C1,@ESCFLG ; NO ESCAPE IN EDIT !!!
      0134 0000
      0136 0000
0293 0138 0A74      SLA  R4,7        ; POSITION BYTE
0294 013A DE04      MOVB R4,*R8+     ; STORE IN STREAM
0295 013C 0284      CI   R4,>300    ; CHECK SPECIAL TYPES
      013E 0300
0296 0140 1108      JLT  ED12      ; GOTO, GOSUB
0297 0142 1396      JEQ  ED3       ; ELSE
0298 0144 0284      CI   R4,>400    ; REM?
      0146 0400
0299 0148 160F      JNE  ED14      ; N
0300 014A 0460      B    @EDREM     ; Y
      014C 035E'
0301      *
0302 014E DE20      ED11  MOVB @B3F,*R8+ ; STORE ,
      0150 0802'
0303      *
0304 0152 0420      ED12  BLWP @CVDIZ ; GET INTEGER
      0154 004C'
0305 0156 1000      NOP
0306 0158 10BD      JMP  EDER3
0307 015A DE01      MOVB R1,*R8+     ; STORE INTEGER
0308 015C 06C1      SWPB R1
0309 015E DE01      MOVB R1,*R8+
0310 0160 0280      CI   R0,>2C00    ; ", ?
      0162 2C00
0311 0164 13F4      JEQ  ED11      ; Y
0312      *
0313 0166 0607      ED13  DEC  R7     ; FALL THRU TO ED20
0314      *
0315 0168 C308      ED14  MOV  R8,R12 ; MARK

```

```

0317      *PROCESS REST OF LIST
0318      *
0319 016A 0420 ED20 BLWP @CVDIFZ      ; LOOK FOR NUMBER
      016C 0000
0320 016E 1014      JMP ED23      ; OVERFLOW, FP
0321 0170 101D      JMP ED30      ; NO NUMBER
0322 0172 1009      JMP ED20H     ; HEX
0323 0174 02B1      CI R1,-1      ; <-1?
      0176 FFFF
0324 0178 1109      JLT ED21      ; Y
0325 017A 8801      C R1,@RENCMP   FULL COMPRESSION?
      017C 0020'
0326 017E 1506      JGT ED21      ; N
0327 0180 0221      AI R1,>63     ; GET CODE (62-6C)
      0182 0063
0328 0184 1006      JMP ED22      ; INSERT
0329      *
0330 0186 DE20 ED20H MOVB @B6E,*R8+ ; MARK AS HEX
      0188 0A45'
0331 018A 1002      JMP ED21A
0332      *
0333 018C DE20 ED21 MOVB @B6D,*R8+ ; 2 BYTE INTEGER, INSERT >6D
      018E 0810'
0334 0190 DE01 ED21A MOVB R1,*R8+
0335 0192 06C1 ED22 SWPB R1
0336 0194 DE01      MOVB R1,*R8+
0337 0196 1009      JMP ED24      ; LOOK FOR OPERATOR
0338      *
0339 0198 DE20 ED23 MOVB @B6F,*R8+ ; FP £, INSERT >6F
      019A 0812'
0340 019C 0203      LI R3,6
      019E 0006
0341 01A0 0204      LI R4,FPAC    ; GET FPAC ADR
      01A2 0000
0342      *
0343 01A4 DE34      MOVB *R4+,*R8+ ; MOVE IN NUMBER
0344 01A6 0603      DEC R3        ; DONE?
0345 01A8 16FD      JNE #-4      ; N
0346      *
0347 01AA 0607 ED24 DEC R7        ; Y, BACKUP OVER DELIMITER
0348      *
0349      *VARIABLE OR KEY WORD
0350      *GET 3 CHARACTERS - EXIT TO EDOP IF NON-LETTER
0351      *
0352 01AC 06A0 ED30 BL @EDGWD     ; CHECK FOR WORD OPERATER
      01AE 05E0'
0353 01B0 06A0      BL @EDGLS     ; PROCESS COMMAND LIST
      01B2 05C0'
0354 01B4 105E      JMP EDOP      ; NOT LETTER, OPERATOR
0355 01B6 C001      MOV R1,R0
0356 01B8 06A0      BL @EDGL      ; GET NEXT LETTER
      01BA 05C8'
0357 01BC 101D      JMP EDV1      ; NOT LETTER, 1 CHARACTER VARIAB
0358 01BE A001      A R1,R0      ; ADD NEW LETTER
0359 01C0 02B0      CI R0,>3BC0   ; 'FN'?
      01C2 3BC0
0360 01C4 1602      JNE ED31      ; N
0361 01C6 0460      B @EDFN      ; Y, FUNCTION
      01C8 03A8'
0362      *

```

0363 01CA 06A0 ED31 BL @EDGL
01CC 05C8'
0364 01CE 1025 JMP EDV2
0365 01DO A001 A R1,R0

;N, GET 3RD LETTER

;NOT LETTER, 2 LETTER VARIABLE


```

0367      *LOOK FOR 3 LETTER KEY WORDS
0368      *
0369 01D2 0204      LI      R4,EDIL      ; GET LIST PTR
      01D4 0762'
0370      *
0371 01D6 C074      ED32      MOV      *R4+,R1      ; GET CHARACTERS
0372 01D8 0911      SRL      R1,1      ; REMOVE TERMINATOR
0373 01DA 8040      C      R0,R1      ; SAME?
0374 01DC 1609      JNE      ED34      ; N
0375 01DE 020B      LI      R11,EDIL
      01E0 0762'
0376 01E2 0224      AI      R4,>1A*2
      01E4 0034
0377 01E6 610B      S      R11,R4
0378 01EB 0914      SRL      R4,1      ; SYSTEM FUNCTION
0379      *
0380 01EA 06C4      ED33      SWPB R4      ; POSITION
0381 01EC DE04      MOV      R4,*R8+      ; INSERT
0382 01EE 10BD      JMP      ED20
0383      *
0384 01F0 0284      ED34      CI      R4,EDILE      ; N, LIST DONE?
      01F2 079C'
0385 01F4 1AF0      JL      ED32      ; N
0386 01F6 1011      JMP      ED35
0387      *
0388      *1 CHARACTER VARIABLE
0389      *
0390 01FB C0C0      EDV1      MOV      R0,R3      ; LOOK FOR £
0391 01FA 06A0      BL      @CVDB20      ; CHECK FOR DIGIT
      01FC 0000
0392 01FE 100B      JMP      EDV1A      ; N
0393 0200 0607      DEC      R7      ; Y, BACKUP
0394 0202 0420      BLWP      @CVDIZ      ; CONVERT
      0204 0154'
0395 0206 101D      JMP      EDER4      ; FP £, ILLEGAL VARIABLE NAME
0396 0208 1007      JMP      EDV1B      ; NO NUMBER
0397 020A 2460      CZC      @CFF80,R1      ; TOO LARGE?
      020C 06FA'
0398 020E 1619      JNE      EDER4      ; Y, ILLEGAL NAME
0399 0210 A0C1      A      R1,R3      ; COMBINE
0400 0212 0263      ORI      R3,>0380      ; INDICATE AS NUMBER
      0214 0380
0401      0214' C380      EQU      $-2
0402      *
0403 0216 0607      EDV1A      DEC      R7      ; BACKUP OVER DELIMITER
0404      *
0405 0218 C003      EDV1B      MOV      R3,R0      ; RESTORE R0

```

```

0407          *2 OR 3 CHARACTER VARIABLE
0408          *
0409          021A' EDV2   EQU   $
0410 021A 06A0 ED35   BL    @EDGP      ; LOOK FOR DIMENSION
          021C 0616'
0411 021E 1002          JMP   ED35A      ; Y, DIMENSIONED
0412 0220 0607          DEC   R7        ; BACKUP
0413 0222 1003          JMP   ED36
0414          *
0415 0224 0500 ED35A   NEG   R0        ; SET DIMENSION
0416 0226 05C6          INCT  R6        ; STACK -1
0417 0228 0716          SETD  *R6
0418          *
0419          *SAVE VARIABLE NAME
0420          *
0421 022A 0204 ED36   LI     R4,>FF00+>70 ; -# OF VARIABLES
          022C FF70
0422 022E C160          MOV   @VNT,R5    ; GET VARIABLE TABLE ADR
          0230 003C'
0423          *
0424 0232 8805 ED37   C      R5,@VDT    ; DONE?
          0234 0000
0425 0236 1409          JHE   ED38      ; Y, MAKE NEW VARIABLE
0426 0238 8D40          C      R0,*R5+  ; VARIABLE SAME?
0427 023A 1317          JEQ   ED39      ; Y, FOUND
0428 023C 0584          INC   R4        ; STILL ROOM?
0429 023E 16F9          JNE   ED37      ; Y
0430          *
0431 0240 2F85 ERR5   DATA ERROR+5    ; N, TOO MANY VARIABLES
0432          *
0433 0242 06A0 EDER4   BL    @EDER      ; ILLEGAL VARIABLE NAME
          0244 00D6'
0434 0246 30          TEXT '04'
          0247 34
0435          *
0436 0248 2F8A ERR10  DATA ERROR+10    ; STORAGE OVERFLOW

```

```

0438      *DEFINE NEW VARIABLE
0439      *
0440 024A C0A0 ED38 MOV @NVD,R2      ;GET END OF TABLE
      024C 0000
0441 024E 8CB2 C *R2+,*R2+      ;ADD 4
0442 0250 8802 C R2,@NVS      ;ROOM FOR NEW VARIABLE?
      0252 0000
0443 0254 14F9 JHE ERR10      ;N
0444 0256 C802 MOV R2,@NVD      ;Y, UPDATE NVD
      0258 024C
0445      *
0446 025A 0222 ED38A AI R2,-4      ;MOVE POINTERS UP 2 BYTES
      025C FFFC
0447 025E C4B2 MOV *R2+,*R2
0448 0260 8142 C R2,R5      ;DONE?
0449 0262 1BFB JH ED38A      ;N
0450 0264 C540 MOV R0,*R5      ;Y, STORE NEW NAME
0451 0266 05E0 INCT @VDT      ;UPDATE VDT
      0268 0234
0452      *
0453 026A C000 ED39 MOV R0,R0      ;DIMENSIONED?
0454 026C 11BE JLT ED33      ;Y
0455 026E 06C4 SWPB R4      ;N, INSERT VARIABLE CODE
0456 0270 DE04 MOVB R4,*R8+
0457      *
0458      * TRANSLATE CHARACTER INTO CODE
0459      *
0460 0272 04C0 EDOP CLR R0      ;GET OPERATER
0461 0274 D037 MOVB *R7+,R0      ;GET BYTE
0462 0276 1375 JEQ ED90      ;DONE
0463 0278 C040 MOV R0,R1
0464 027A 06C1 SWPB R1
0465 027C 0280 CI R0,>4100      ;<"A?
      027E 4100
0466 0280 1A08 JL EDOP1      ;Y
0467 0282 0280 CI R0,>5B00      ;>"Z?
      0284 5B00
0468 0286 1A47 JL ED49      ;N, LETTER, TRY WORD OPERATER
0469 0288 0221 AI R1,-26      ;SKIP ALPHABET
      028A FFE6
0470 028C 0280 CI R0,>5E00      ;>"^?
      028E 5E00
0471 0290 1B3F JH EDER6      ;Y, INVALID CHARACTER
0472      *
0473 0292 D061 EDOP1 MOVB @EDSL->21(1),R1 ;GET CODE
      0294 077D
0474 0296 133C JEQ EDER6      ;INVALID CHARACTER
0475 0298 D601 MOVB R1,*R8      ;STORE BYTE
0476 029A 06A0 BL @JMPRO      ;SWITCHBOARD ON SPECIAL CHARACT
      029C 0000
0477 029E EA EDOPTB BYTE EDOP-EDOPTB/2,>20 SP
0478 02A0 42 BYTE ED50-EDOPTB/2,>28 "("
0479 02A2 42 BYTE ED50-EDOPTB/2,>5B "["
0480 02A4 55 BYTE ED51-EDOPTB/2,>29 ")"
0481 02A6 55 BYTE ED51-EDOPTB/2,>5D "]"
0482 02A8 0F BYTE ED42-EDOPTB/2,>22 ""
0483 02AA 0F BYTE ED42-EDOPTB/2,>27 "'
0484 02AC 30 BYTE ED47-EDOPTB/2,>3A ":"
0485 02AE 5F BYTE ED54-EDOPTB/2,>21 "!"
0486 02B0 16 BYTE ED43-EDOPTB/2,>3D "="
    
```

```

0487 02B2 27          BYTE ED45-EDOPTB/2,>3E ">
0488 02B4 0000        DATA 0
0489 02B6 0588 ED40    INC R8          ; MOVE OVER CODE
0490 02B8 0460 ED41    B @ED20        ; CONTINUE PARSE
      02BA 016A'
0491                *
0492 02BC 0588 ED42    INC R8          ; PROCESS " OR '
0493 02BE D637        MOVB *R7+,*R8   ; GET NEXT BYTE
0494 02C0 1370        JEQ EDER2       ; PROBLEM
0495 02C2 9600        CB R0,*R8      ; CLOSE?
0496 02C4 16FB        JNE ED42       ; N
0497 02C6 7E18        SB *R8,*R8+    ; Y, TERMINATE WITH NULL
0498 02C8 10F7        JMP ED41
0499                *
0500 02CA 0608 ED43    DEC R8          ; PROCESS "="
0501 02CC 9818        CB *R8,@B57    ; >=?
      02CE 07BB'
0502 02D0 130A        JEQ ED44       ; Y
0503 02D2 9818        CB *R8,@B59    ; <=?
      02D4 07B9'
0504 02D6 1307        JEQ ED44       ; Y
0505 02D8 9838        CB *R8+,@B56   ; ==?
      02DA 07BA'
0506 02DC 16EC        JNE ED40
0507 02DE 7A20        SB @B01,@-1(8) ; Y, MAKE ==
      02E0 0000
      02E2 FFFF
0508 02E4 10E9        JMP ED41
0509                *
0510 02E6 BE20 ED44    AB @B01,*R8+   ; MODIFY CODE
      02E8 02E0'
0511 02EA 10E6        JMP ED41       ; GOTO ED20
0512                *
0513 02EC 9828 ED45    CB @-1(8),@B59 ; PROCESS ">
      02EE FFFF
      02F0 07B9'
0514 02F2 16E1        JNE ED40
0515 02F4 BA20        AB @B02,@-1(8) ; MAKE <>
      02F6 06FF'
      02F8 FFFF
0516 02FA 10DE ED41P  JMP ED41
0517                *
0518 02FC 0587 ED46    INC R7          ; PROCESS :, THEN
0519 02FE 9817 ED47    CB *R7,@B20   ; SPACE
      0300 00FA'
0520 0302 13FC        JEQ ED46       ; Y, IGNOR
0521 0304 9817        CB *R7,@B3A    ; N, IS IT ANOTHER COLON ?
      0306 07FD'
0522 0308 13F9        JEQ ED46       ; Y, IGNORE IT THEN
0523 030A 0588        INC R8          ; N, PROCESS NEW COMMAND
0524 030C 0460        B @ED3
      030E 0070'
0525                *
0526 0310 06A0 EDER6  BL @EDER       ; ILLEGAL CHARACTER
      0312 0244'
0527 0314 30          TEXT '06'
      0315 36
0528                *
0529                *LOOK FOR WORD OPERATER
0530                *

```

0531 0316 0607 ED49 DEC R7
0532 0318 06A0 BL @EDGWD
031A 05E0'
0533 031C 06A0 BL @EDER ; EXPECTING OPERATER
031E 0312'
0534 0320 30 TEXT '07'
0321 37

```

0536          *( OR I ENTRY
0537          *
0538 0322 8308 ED50 C R8,R12 ; COMMAND BYTE?
0539 0324 1B05 JH ED50A ; N
0540 0326 9828 CB @-1(8),@BCPB ; MEM, CRB, CRF, BIT?
        0328 FFFF
        032A 06FB'
0541 032C 1A0B JL ED50B ; N, LEAVE (
0542          * FOLLOWING LINE NEEDED AS CMD BYTES BIT - MWD CODE >B3B
0543 032E 1006 JMP ED50A1 ; Y, FORCE '['
0544 0330 D068 ED50A MOVB @-1(R8),R1 ; IS LAST CHAR STORED
        0332 FFFF
0545          * STRING TERMINATOR (0)?
0546 0334 1307 JEQ ED50B Y
0547 0336 9801 CB R1,@B3B N - FUNCTION ARGUMENTS?
        0338 0813'
0548 033A 1404 JHE ED50B ; N, LEAVE (
0549 033C 05C6 ED50A1 INCT R6 ; Y, MAKE I
0550 033E 04D6 CLR *R6 ; INSERT -1 INTO STACK
0551 0340 7620 SB @B02,*R8 ; *R8=>4A
        0342 06FF'
0552          *
0553 0344 0616 ED50B DEC *R6 ; INCREMENT TOP ITEM ON STACK
0554 0346 10B7 JMP ED40 ; LEAVE
0555          *
0556          *) OR I ENTRY
0557          *
0558 0348 0596 ED51 INC *R6 ; NEED I?
0559 034A 1106 JLT ED52 ; N, LEAVE >4D
0560 034C 0646 DECT R6 ; Y, DECREMENT STACK
0561 034E 0286 CI R6,SSP ; UNMATCHED?
        0350 0032'
0562 0352 1A27 JL EDER2 ; Y
0563 0354 7620 SB @B02,*R8 ; N, INSERT I CODE (>4B)
        0356 06FF'
0564          *
0565 0358 0588 ED52 INC R8 ; LEAVE CODE
0566 035A 108B ED52A JMP EDOP ; LOOK FOR OPERATER
    
```

```

0568          *PROCESS REMARKS
0569          *
0570 035C 0588 ED54   INC   R8          ; PROCESS TAIL REMARK
0571          *
0572 035E DE37 EDREM  MOVB  *R7+,*R8+   ; MOVE REMARK
0573 0360 16FE          JNE   EDREM
0574          *
0575          *END PARSE, CHECK FINAL ERRORS
0576          *
0577 0362 0286 ED90   CI    R6,SSP      ; SUBSCRIPT ERROR?
          0364 0350'
0578 0366 161D          JNE   EDER2     ; Y
0579 0368 0596          INC   *R6       ; PAREN ERROR?
0580 036A 161B          JNE   EDER2     ; Y
0581 036C 7E18          SB     *R8,*R8+  ; N, MARK OUTPUT
0582 036E 7618          SB     *R8,*R8   ; DOUBLE NULL
0583 0370 0700          SETO  R0         ; SET TO INSERT
0584 0372 C060          MOV   @LNUM,R1    ; GET LINE NUMBER
          0374 0054'
0585 0376 1651          JNE   EMVO      ; INSERT OR CHANGE
0586 0378 C1E0          MOV   @IOB,R7    ; NO LINE NUMBER
          037A 002E'
0587 037C 0227          AI     R7,30     ; MOVE INTO IOB
          037E 001E
0588 0380 C007          MOV   R7,R0      ; MARK
0589 0382 0203          LI     R3,EBP    ; **NOTE** ALL PTRS ON WRD BOUND
          0384 0038'
0590          *
0591 0386 CDF3          MOV   *R3+,*R7+  ; MOVE
0592 0388 8203          C      R3,R8     ; DONE?
0593 038A 1AFD          JL     #-4       ; N
0594          *
0595 038C 04E0          CLR   @ESCFLG    ; Y, ENABLE ESCAPE
          038E 0136'
0596 0390 C200          MOV   R0,R8      ; SET R8
0597 0392 9818          CB     *R8,@B01  ; CHECK TYPE, GOTO?
          0394 02E8'
0598 0396 1B03          JH     ED91      ; N
0599 0398 0720          SETO  @MODE     ; Y, SET TO RUN MODE
          039A 0012'
0600 039C 0004'        DATA TYPC$     ; OUT CRLF
0601          *
0602 039E 0460 ED91   B      @LINE2
          03A0 0000
0603          *
0604 03A2 06A0 EDER2  BL     @EDER      ; UNMATCHED PARENTHESIS
          03A4 031E'
0605 03A6 30          TEXT '02'
          03A7 32

```

```

0607      *PROCESS FN-
0608      *
0609 03A8 06A0 EDFN BL @EDGLS      ; GET LETTER
      03AA 05C0'
0610 03AC 102F      JMP EDER8      ; NO LETTER
0611 03AE 0921      SRL R1,2      ; POSITION
0612 03B0 DE01      MOV B R1,*R8+ ; INSERT INLINE
0613 03B2 C0C8      MOV R8,R3     ; CHECK PREVIOUS BYTE
0614 03B4 0643      DECT R3       ; LOOK FOR OPERATOR
0615 03B6 0283      CI R3,EBP     ; BEGINNING OF BUFFER?
      03BB 03B4'
0616 03BA 16CF      JNE ED52A     ; N, DISREGUARD
0617 03BC 9813      CB *R3,@DEFXB ; 'DEF'?
      03BE 0814'
0618 03C0 16CC      JNE ED52A     ; N, CONTINUE
0619 03C2 06A0      BL @EDGP      ; Y, LOOK FOR ( OR [
      03C4 0616'
0620 03C6 1002      JMP EDFN0     ; "( OR "I, OK
0621 03CB 0607      DEC R7        ; NEITHER, LOOK FOR "="
0622 03CA 1017      JMP EDFN3
0623      *
0624 03CC 0203 EDFN0 LI R3,3      ; Y, ALLOW 3 DUMMY VARIABLES
      03CE 0003
0625 03D0 C120      MOV @VNT,R4   ; GET STORAGE ADR
      03D2 0230'
0626      *
0627 03D4 06A0 EDFN1 BL @EDGLS    ; GET DUMMY VARIABLE
      03D6 05C0'
0628 03D8 101C      JMP EDER9     ; PROBLEM
0629 03DA DE01      MOV B R1,*R8+ ; INSERT IN CODE
0630 03DC CD01      MOV R1,*R4+   ; STORE CODE
0631 03DE 06A0      BL @EDGP      ; GET NEXT BYTE
      03E0 0616'
0632 03E2 1017      JMP EDER9     ;
0633 03E4 0603      DEC R3        ; ROOM FOR MORE?
0634 03E6 1303      JEQ EDFN2     ; N, SEE IF ")"
0635 03E8 0281      CI R1,>2C00   ; Y, DELIMITER ".,?
      03EA 2C00
0636 03EC 13F3      JEQ EDFN1     ; Y, LOOP
0637      *
0638 03EE 0281 EDFN2 CI R1,>2900   ; N, CLOSING PAREN?
      03F0 2900
0639 03F2 1303      JEQ EDFN3     ; Y, LOOK FOR "="
0640 03F4 0281      CI R1,>5D00   ; N, "I"?
      03F6 5D00
0641 03F8 16D4      JNE EDER2     ; N, PROBLEM
0642 03FA 95E0 EDFN3 CB @B3D,*R7  ; Y, "=?
      03FC 0817'
0643 03FE 13AD      JEQ ED52A     ; Y
0644 0400 9DE0      CB @B20,*R7+ ; SPACE?
      0402 0300'
0645 0404 13FA      JEQ EDFN3     ; Y, TRY AGAIN
0646 0406 06A0      BL @EDER      ; MISSING ASSIGNMENT
      0408 03A4'
0647 040A 33      TEXT '36'
      040B 36
0648      *
0649 040C 06A0 EDFN8 BL @EDER     ; ILLEGAL FUNCTION NAME
      040E 0408'
0650 0410 30      TEXT '08'

```


0411 38
0651 *
0652 0412 06A0 EDER9 BL @EDER , ILLEGAL FUNCTION ARGUMENT
0414 040E'
0653 0416 30 TEXT '09'
0417 39

```

0655      *FINISH EDIT PROCESS
0656      *          BL @EMV
0657      *
0658      *          IN  R0 = 0 FOR DELETE, <>0 FOR CHANGE OR INSERT
0659      *          R1 = LINE NUMBER
0660      *
0661 0418 C28B  EMV   MOV   R11,R10      ;SAVE RETURN
0662 041A C041  EMV0  MOV   R1,R1        ;DELETE 0?
0663 041C 1375      JEQ   EMVR          ;Y, ACTION COMPLETE
0664 041E 0202      LI    R2,EBP
        0420 03B8'
0665 0422 6202      S     R2,R8          ;GET £ OF BYTES
0666 0424 0588      INC   R8             ;GET £ OF WORDS (NEXT HIGHEST)
0667 0426 0818      SRA   R8,1
0668 0428 C088      MOV   R8,R2          ;GET £ OF BYTES
0669 042A A082      A     R2,R2
0670 042C C1A0      MOV   @SLT,R6        ;GET START OF STATEMENT TABLE
        042E 0000
0671 0430 C1C6      MOV   R6,R7
0672 0432 61E0      S     @BUS,R7        ;R7=POINT OF QUESTION IN PSEUDO
        0434 0000
0673      *
0674      *UPON EXIT OF EMV1, R7 DISPLACES INTO PSEUDO SOURCE
0675      * AND R6 POINTS INTO SLT
0676      *
0677 0436 C116  EMV1  MOV   *R6,R4        ;DONE?
0678 0438 1307      JEQ   EMV2            ;Y
0679 043A 8581      C     R1,*R6          ;N, FOUND?
0680 043C 1366      JEQ   EMV5            ;Y, CHANGE OR DELETE
0681 043E 1504      JGT   EMV2            ;Y, NEW LINE
0682 0440 05C6      INCT  R6
0683 0442 C1F6      MOV   *R6+,R7        ;GET NEW POINT OF QUESTION
0684 0444 10F8      JMP   EMV1
0685      *
0686 0446 2F8D  ERR13 DATA ERROR+13      ;NO SUCH LINE NUMBER
0687      *
0688 0448 C000  EMV2  MOV   R0,R0          ;NEW £, DELETE?
0689 044A 13FD      JEQ   ERR13          ;Y, PROBLEM
0690      *
0691      *INSERT NEW LINE ENTRY
0692      *
0693 044C C0C6      MOV   R6,R3          ;GET SOURCE
0694 044E 06A0      BL    @EMVA          ;ADJUST
        0450 053E'
0695 0452 0004      DATA 4              ;INSERT 4 BYTES IN SLT
0696 0454 CD81      MOV   R1,*R6+        ;INSERT NEW LINE £
0697 0456 C587      MOV   R7,*R6        ;INSERT DISPLACEMENT
0698      *
0699 0458 0204  EMV3  LI    R4,EBP        ;MOVE IN SOURCE LINE
        045A 0420'
0700 045C C196      MOV   *R6,R6          ;GET ADR
0701 045E A1A0      A     @BUS,R6        ;MAKE DISPLACEMENT, POINTER
        0460 0434'
0702 0462 C1C6      MOV   R6,R7          ;MOVE IN STRING
0703      *
0704 0464 CDF4      MOV   *R4+,*R7+      ;MOVE
0705 0466 0608      DEC   R8             ;DONE?
0706 0468 16FD      JNE   $-4            ;N
    
```

```

0708      *ADJUST GOSUB
0709      *
0710      *      IN  R2 = PBC ADJUSTMENT
0711      *      R3 = PLC ADJUSTMENT
0712      *      R6 = START
0713      *      R7 = END
0714      *
0715 046A C120 EMV4  MOV  @GSC,R4      ; GET GOSUB STACK PTR
      046C 0000
0716      *
0717 046E 8804 EMV4A C    R4,@GSS      ; DONE?
      0470 0000
0718 0472 1217      JLE  EMV4E      ; Y
0719 0474 0224      AI   R4,-4      ; N, BACKUP
      0476 FFFC
0720 0478 8506      C    R6,*R4      ; LESS THAN INSERTED LINE?
0721 047A 1B0E      JH   EMV4C1      ; Y
0722 047C 8507      C    R7,*R4      ; GREATER THAN?
0723 047E 120A      JLE  EMV4C      ; Y
0724 0480 C144      MOV  R4,R5      ; N, DELETE ENTRY
0725      *
0726 0482 CD65 EMV4B MOV  @4(5),*R5+  ; DELETE ENTRY
      0484 0004
0727 0486 8805      C    R5,@GSC      ; DONE?
      0488 046C'
0728 048A 1AFB      JL   EMV4B
0729 048C AB20      A    @CM4,@GSC    ; BACKUP PTR
      048E 051E'
      0490 048B'
0730 0492 10ED      JMP  EMV4A
0731      *
0732 0494 A502 EMV4C A    R2,*R4      ; Y, ADJUST PBC
0733 0496 1002      JMP  EMV4C2
0734 0498 A903 EMV4C1 A   R3,@2(4)    ; PLC=PLC+PLC ADJUSTOR
      049A 0002
0735 049C A902 EMV4C2 A   R2,@2(4)    ; PLC=PLC+PBC ADJUSTOR
      049E 0002
0736 04A0 10E6      JMP  EMV4A
0737      *
0738      *ADJUST FOR/NEXT STACK
0739      *
0740 04A2 C120 EMV4E MOV  @FNS,R4      ; DO FOR/NEXT STACK
      04A4 0000
0741      *
0742 04A6 C014 EMV4F MOV  *R4,R0      ; DONE?
0743 04A8 1313      JEQ  EMV4J      ; Y
0744 04AA C144      MOV  R4,R5      ; N
0745 04AC 0225      AI   R5,14
      04AE 000E
0746 04B0 8546      C    R6,*R5      ; LESS THAN INSERTED LINE?
0747 04B2 1B07      JH   EMV4I      ; Y
0748 04B4 8547      C    R7,*R5      ; >=?
0749 04B6 1203      JLE  EMV4H      ; Y
0750 04B8 06A0      BL   @FOR2      ; DELETE ENTRY
      04BA 0000
0751 04BC 10F4      JMP  EMV4F      ; LOOK AGAIN
0752      *
0753 04BE A542 EMV4H A    R2,*R5      ; Y, ADJUST PBC
0754 04C0 1002      JMP  EMV4I1
0755 04C2 A943 EMV4I A    R3,@2(R5)    ; PLC=PLC+PLC ADJUSTOR

```

04C4 0002
0756 04C6 A942 EMV4I1 A R2,@2(R5) ;PLC=PLC+PBC ADJUSTOR
04C8 0002
0757 04CA 0224 AI R4,18 ;MOVE TO NEXT
04CC 0012
0758 04CE 10EB JMP EMV4F

```

0760      *ADJUST DATA POINTERS
0761      *
0762 04D0 0204 EMV4J LI R4,DDM ;GET DELIMITER PTR
      04D2 0000
0763 04D4 8506 C R6,*R4 ;CHECK DATA PTRS
0764 04D6 1B09 JH EMV4L ;OK
0765 04D8 8507 C R7,*R4 ;DELETED OR CHANGED?
0766 04DA 1202 JLE EMV4K ;N, INSERTED
0767 04DC 04D4 CLR *R4 ;Y, SET TO LOOK FURTHER
0768 04DE 1003 JMP EMV4KA
0769      *
0770 04E0 C014 EMV4K MOV *R4,R0 ;DEFINED?
0771 04E2 1303 JEQ EMV4L ;N, DON'T WORRY ABOUT IT
0772 04E4 A502 A R2,*R4 ;Y, ADJUST PBC
0773      *
0774 04E6 AB03 EMV4KA A R3,@DLC ;ADJUST PLC
      04EB 0000
0775      *
0776      *ADJUST FUNCTION DEFINITION STACK
0777      *
0778 04EA C120 EMV4L MOV @UFT,R4 ;GET POINTER
      04EC 0000
0779      *
0780 04EE C014 EMV4M MOV *R4,R0 ;DEFINED?
0781 04F0 1307 JEQ EMV4D ;N
0782 04F2 8506 C R6,*R4 ;LESS THAN CHANGED LINE?
0783 04F4 1B05 JH EMV4D ;Y
0784 04F6 8507 C R7,*R4
0785 04F8 1202 JLE EMV4N ;N
0786 04FA 04D4 CLR *R4 ;UNDEFINE
0787 04FC 1001 JMP EMV4D
0788      *
0789 04FE A502 EMV4N A R2,*R4 ;ADJUST PBC
0790 0500 8D34 EMV4D C *R4+,*R4+ ;MOVE TO NEXT
0791 0502 8804 C R4,@GSS ;DONE?
      0504 0470
0792 0506 1AF3 JL EMV4M ;N
0793      *
0794 0508 045A EMVR B *R10 ;RETURN
  
```

```

0796      *DELETE OR CHANGE LINE
0797      *
0798 050A C2E6 EMV5   MOV   @2(6),R11      ; GET BOL
      050C 0002
0799 050E 62C7      S     R7,R11          ; GET -LINE LENGTH
0800 0510 C000      MOV   R0,R0          ; DELETE LINE?
0801 0512 1607      JNE   EMV6          ; N
0802      *
0803      *DELETE LINE
0804      *
0805 0514 C08B      MOV   R11,R2          ; Y
0806 0516 C0C6      MOV   R6,R3
0807      *        AI     R3,4          ; GET SLT SOURCE
0808 0518 8CF3      C     *R3+,*R3+      LESS CODE THAN AI R3,4
0809 051A 06A0      BL    @EMVA          ; ADJUST
      051C 053E'
0810 051E FFFC CM4   DATA -4          ; DELETE 4 BYTES FROM SLT
0811 0520 10A4      JMP   EMV4
0812      *
0813      *CHANGE LINE
0814      *
0815 0522 A08B EMV6   A      R11,R2      ; GET DELTA CHANGE
0816 0524 C020      MOV   @NVD,R0      ; CHECK STORAGE
      0526 025B'
0817 0528 A002      A      R2,R0
0818 052A 8800      C      R0,@NVS      ; ROOM?
      052C 0252'
0819 052F 1406      JHE   EMVE10        ; N
0820 0530 020D      LI    R13,EMV6A     ; GET RETURN ADR
      0532 0536'
0821 0534 1017      JMP   EMVA1
0822      *
0823 0536 0000 EMV6A DATA 0          ; LEAVE PLC ALONE
0824 0538 05C6      INCT  R6          ; MOVE TO LINE ADR
0825 053A 108E      JMP   EMV3
0826      *
0827 053C 2FBA EMVE10 DATA ERROR+10 ; STORAGE OVERFLOW

```

```

0829      *ALTER SOURCE CODE
0830      *      BL @EMVA
0831      *      DATA (SLT ADJUSTMENT)
0832      *      R3=SLT SOURCE PTR
0833      *
0834 053E C34B EMVA MOV R11,R13 ;SAVE RETURN
0835 0540 C120 MOV @NVD,R4 ;GET THRU POINTER (END OF VARIA
      0542 0526'
0836 0544 C143 MOV R3,R5 ;GET DESTINATION
0837 0546 A15D A *R13,R5 ; (4=INSERT,-4=DELETE,0=CHANGE)
0838 0548 C004 MOV R4,R0 ;CHECK FOR SIZE
0839 054A A01D A *R13,R0
0840 054C A002 A R2,R0 ;ADD LINE
0841 054E 8800 C R0,@NVS
      0550 052C'
0842 0552 14F4 JHE EMVE10 ;OVERFLOW
0843 0554 AB1D A *R13,@VNT ;ADJUST POINTERS
      0556 03D2'
0844 0558 AB1D A *R13,@VDT
      055A 0268'
0845 055C AB1D A *R13,@NVD
      055E 0542'
0846 0560 06A0 BL @MOVE ;DO FIRST MOVE
      0562 05A0'
0847      *
0848 0564 C0C7 EMVA1 MOV R7,R3 ;MAKE HOLE IN PSEUDO SOURCE
0849 0566 A0E0 A @BUS,R3 ;MAKE DISPLACEMENT INTO POINTER
      0568 0460'
0850 056A C120 MOV @NVD,R4 ;THRU
      056C 055E'
0851 056E C143 MOV R3,R5
0852 0570 A142 A R2,R5 ;DESTINATION
0853 0572 A182 A R2,R6
0854 0574 AB02 A R2,@SLT ;ADJUST POINTERS
      0576 042E'
0855 0578 AB02 A R2,@VNT
      057A 0556'
0856 057C AB02 A R2,@VDT
      057E 055A'
0857 0580 AB02 A R2,@NVD
      0582 056C'
0858 0584 AB02 A R2,@RBSTOR ;UPDATE RBSTOR FOR ENTER.
      0586 0000
0859 0588 06A0 BL @MOVE ;DO SECOND MOVE
      058A 05A0'
0860      *
0861      *ADJUST SLT
0862      *
0863 058C C0C6 MOV R6,R3 ;GET POINTER
0864      *
0865 058E 8803 EMVA2 C R3,@SLT ;DONE?
      0590 0576'
0866 0592 1204 JLE EMVA3 ;Y
0867 0594 0643 DECT R3 ;N
0868 0596 A4C2 A R2,*R3 ;ADJUST PTR
0869 0598 0643 DECT R3
0870 059A 10F9 JMP EMVA2
0871      *
0872 059C C0FD EMVA3 MOV *R13+,R3 ;GET PLC ADJUSTMENT
0873 059E 045D B *R13 ;RETURN

```

```

0875      *MOVE
0876      *          BL @MOVE
0877      *
0878      *          IN  R3 = SOURCE
0879      *          R4 = SOURCE END
0880      *          R5 = DESTINATION
0881      *
0882 05A0 8143  MOVE  C    R3,R5          ; THERE?
0883 05A2 131D      JEQ  MOVE4          ; Y
0884 05A4 1A04      JL   MOVE2          ; N, SCD
0885      *
0886 05A6 8103  MOVE1 C    R3,R4          ; SCD, DONE?
0887 05A8 1B1A      JH   MOVE4          ; Y
0888 05AA CD73      MOV  *R3+,*R5+      ; MOVE DATA
0889 05AC 10FC      JMP  MOVE1
0890      *
0891 05AE C004  MOVE2 MOV  R4,R0          ; SCD
0892 05B0 6003      S    R3,R0          ; GET £ OF WORDS
0893 05B2 A140      A    R0,R5
0894      *
0895 05B4 C554  MOVE3 MOV  *R4,*R5      ; MOVE DATA
0896 05B6 8103      C    R3,R4          ; DONE?
0897 05B8 1312      JEQ  MOVE4          ; Y
0898 05BA 0644      DECT R4             ; N, BACKUP
0899 05BC 0645      DECT R5
0900 05BE 10FA      JMP  MOVE3
  
```



```

0902      *GET LETTER
0903      *      BL @EDGL
0904      *      NO LETTER, R7 UNCHANGED
0905      *      LETTER, R7 UPDATED
0906      *
0907 05C0 9DE0 EDGLS CB @B20,*R7+      ; GET LETTER, SPACE?
      05C2 0402'
0908 05C4 13FD      JEQ EDGLS      ; Y
0909 05C6 0607      DEC R7      ; N, BACKUP
0910      *
0911 05C8 9817 EDGL CB *R7,@B40      ; <"A?
      05CA 07FE'
0912 05CC 1208      JLE EDGL1      ; Y
0913 05CE 9817      CB *R7,@B5B      ; >"Z?
      05D0 080B'
0914 05D2 1405      JHE EDGL1      ; Y
0915 05D4 04C1      CLR R1      ; LETTER
0916 05D6 D077      MOV B *R7+,R1
0917 05D8 0A21      SLA R1,2      ; REMOVE UPPER BITS
0918 05DA 0950      SRL R0,5      ; ADJUST R0
0919 05DC 05CB      INCT R11
0920 05DE 045B EDGL1 RT
0921      *
0922      05DE' MOVE4 EQU EDGL1
0923      *
0924      *GET WORD OPERATER
0925      *
0926 05E0 C14B EDGWD MOV R11,R5
0927 05E2 06A0      BL @EDGLS      ; GET LETTER
      05E4 05C0'
0928 05E6 0455      B *R5      ; NO LETTER, RETURN
0929 05EB 0607      DEC R7      ; OK, PROCESS
0930 05EA 0204      LI R4,EDGWOL      ; GET WORD OPERATER LIST
      05EC 07C2'
0931      *
0932 05EE C0C7 EDGW01 MOV R7,R3      ; MARK
0933      *
0934 05F0 D074 EDGW02 MOV B *R4+,R1      ; GET CHARACTER, FOUND?
0935 05F2 1308      JEQ EDGW04      ; Y
0936 05F4 9073      CB *R3+,R1      ; N, SAVE LETTER?
0937 05F6 13FC      JEQ EDGW02      ; Y
0938 05F8 D074      MOV B *R4+,R1      ; N, MOVE TO NEXT
0939 05FA 16FE      JNE #-2
0940 05FC 0584      INC R4      ; MOVE OVER CODE
0941 05FE D054      MOV B *R4,R1      ; DONE?
0942 0600 16F6      JNE EDGW01      ; N, KEEP TRYING
0943 0602 0455      B *R5      ; Y, RETURN
0944      *
0945 0604 D614 EDGW04 MOV B *R4,*R8      ; RESERVED WORD, GET CODE
0946 0606 C1C3      MOV R3,R7      ; UPDATE R7
0947 0608 981B      CB *R8,@B3B      ; THEN?
      060A 0801'
0948 060C 1302      JEQ EDGW05      ; Y, PROCESS :
0949 060E 0460      B @ED40      ; N, INSERT OPERATER
      0610 02B6'
0950      *
0951 0612 0460 EDGW05 B @ED47
      0614 02FE'

```

```

0953          *GET CHARACTER
0954          *
0955          *      BL @EDGP
0956          *      "( OR "[
0957          *      OTHER
0958          *
0959 0616 04C1  EDGP  CLR  R1
0960 0618 D077      MOVW *R7+,R1      ; GET CHARACTER
0961 061A 02B1      CI   R1,>2000    ; SP?
        061C 2000
0962 061E 13FB      JEQ  EDGP        ; Y
0963 0620 02B1      CI   R1,>2800    ; "(?
        0622 2800
0964 0624 1304      JEQ  EDGP1       ; Y
0965 0626 02B1      CI   R1,>5B00    ; "[?
        0628 5B00
0966 062A 1301      JEQ  EDGP1       ; Y
0967 062C 05CB      INCT R11         ; N, RETURN 2(11)
0968 062E 045B  EDGP1 RT
  
```

```

0970      *
0971      *  COMMAND LIST
0972      *
0973      *  SYMBOLS STORED AS:
0974      *    3333 3222 2211 1115
0975      *    WHERE S=0  3 LETTERS
0976      *             S=1  4-6 LETTERS
0977      *
0978      *           0 1 2 3 4 5 6 7
0979      *           0 @ A B C D E F G
0980      *           1 H I J K L M N O
0981      *           2 P Q R S T U V W
0982      *           3 X Y Z
0983      *

```

0984	0630	7564	EDSCL	DATA >7564	RUN	A
0985	0632	D266		DATA >D266	SIZ(E)	B
0986	0634	73C6		DATA >73C6	CON(TINUE)	C
0987	0636	73DA		DATA >73DA	MON(ITOR)	D
0988	0638	0000		DATA >0000		E
0989	063A	0000		DATA >0000		F
0990	063C	0000		DATA >0000		G
0991	063E	0000		DATA >0000		H
0992	0640	0000		DATA >0000		I
0993	0642	0000		DATA >0000		J
0994	0644	0000		DATA >0000		K
0995			*			
0996	0646	A3CF	EDCL	DATA >A3CF	GOTO*	01
0997		0001	RNGOTO	EGU (\$-EDCL)/2	RENUMBER OPCODE	
0998	0648	9BCF		DATA >9BCF	GOSUB*	02
0999		0002	RNGSUB	EGU (\$-EDCL)/2	RENUMBER OPCODE	
1000	064A	9B0B		DATA >9B0B	ELSE*	03
1001		0003	RNELSE	EGU (\$-EDCL)/2	RENUMBER OPCODE	
1002	064C	6964		DATA >6964	REM*	04
1003		0004	RNREM	EGU (\$-EDCL)/2	RENUMBER OPCODE	
1004	064E	93CC		DATA >93CC	FOR*	05
1005	0650	0000		DATA 0	(LET*)	06
1006	0652	A049		DATA >A049	DATA	07
1007	0654	C15D	NXTX	DATA >C15D	NEXT	08
1008	0656	948B		DATA >948B	ERROR	09
1009		0009	RNERR	EGU (\$-EDCL)/2	RENUMBER OPCODE	
1010	0658	4CA1	PRTX	DATA >4CA1	PRINT	0A
1011	065A	6047		DATA >6047	CALL	0B
1012	065C	0BD9		DATA >0BD9	LOAD	0C
1013	065E	8393		DATA >8393	INPUT	0D
1014		000D	RNINP	EGU (\$-EDCL)/2	RENUMBER OPCODE	
1015	0660	0965		DATA >0965	READ	0E
1016	0662	9965		DATA >9965	RESTOR	0F
1017		000F	RNRSTR	EGU (\$-EDCL)/2	RENUMBER OPCODE	
1018	0664	A165		DATA >A165	RETURN	10
1019	0666	7D27		DATA >7D27	STOP	11
1020	0668	4BAB		DATA >4BAB	UNIT	12
1021	066A	6A69		DATA >6A69	TIME	13
1022	066C	B067		DATA >B067	SAVE	14
1023	066E	9845		DATA >9845	BASE	15
1024	0670	1CCB		DATA >1CCB	ESCAPE	16
1025	0672	2BDD		DATA >2BDD	NOESC	17
1026	0674	7065		DATA >7065	RANDOM	18
1027	0676	AB45		DATA >AB45	BAUD	19
1028	0678	A38B		DATA >A38B	ENTER	1A
1029	067A	7B21		DATA >7B21	PLOT	1B

1030	067C	83AB	DATA	>83AB	UNPLOT	1C
1031	067E	63C7	DATA	>63C7	COLOUR	1D
1032	0680	9561	DATA	>9561	PURGE	1E
1033	0682	0C8F	DATA	>0C8F	GRAPH	1F
1034	0684	C169	DATA	>C169	TEXT	20
1035	0686	486F	DATA	>486F	WAIT	21
1036	0688	0A07	DATA	>0A07	CHAR	22
1037	068A	6D5D	DATA	>6D5D	NUMBER	23
1038	068C	9A59	DATA	>9A59	LIST	24
1039	068E	7165	DATA	>7165	RENUM	25
1040	0690	9427	DATA	>9427	SPRITE	26
1041	0692	0A27	DATA	>0A27	SHAPE	27
1042	0694	AC27	DATA	>AC27	SPUT	28
1043	0696	29E7	DATA	>29E7	SGET	29
1044	0698	7BC5	DATA	>7BC5	BOOT	2A
1045	069A	0DE7	DATA	>0DE7	SWAP	2B
1046	069C	0000	DATA	>0000		2C
1047	069E	A3DB	DATA	>A3DB	MOTOR	2D
1048	06A0	0000	DATA	>0000		2E
1049	06A2	0000	DATA	>0000		2F
1050	06A4	0000	DATA	>0000		30
1051	06A6	0000	DATA	>0000		31
1052	06A8	0000	DATA	>0000		32
1053	06AA	0000	DATA	>0000		33
1054	06AC	0000	DATA	>0000		34
1055	06AE	0000	DATA	>0000		35
1056	06B0	0000	DATA	>0000		36
1057	06B2	0000	DATA	>0000		37
1058	06B4	0000	DATA	>0000		38
1059	06B6	385A	DATA	>385A	MAG	39
1060	06B8	33E8	DATA	>33E8	TOF	3A
1061	06BA	73E8	DATA	>73E8	TON	3B
1062	06BC	83E0	DATA	>83E0	POP	3C
1063	06BE	6A48	DATA	>6A48	DIM	3D
1064	06C0	A15B	DATA	>A15B	LET	3E
1065		007E	PSCBX	EQU	\$-EDCL+2	
1066	06C2	0000		DATA	0	' ' (PRINT) 3F
1067	06C4	73C0	ONX	DATA	>73C0	ON 40
1068		0040	RNON	EQU	(\$-EDCL)/2	RENUMBER OPCODE
1069	06C6	3240	IFX	DATA	>3240	IF 41
1070		0041	RNIF	EQU	(\$-EDCL)/2	RENUMBER OPCODE
1071	06C8	3148	DEFX	DATA	>3148	DEF 42
1072	06CA	B95C		DATA	>B95C	NEW 43
1073	06CC	238A		DATA	>238A	END 44
1074		008A	PQMBX	EQU	\$-EDCL+2	
1075	06CE	0000		DATA	0	(?) (PRINT) 45
1076		008C	PASBX	EQU	\$-EDCL+2	
1077	06D0	0000		DATA	0	(*) (EXTEND) 46
1078		0047	BCP	EQU	\$-EDCL/2+1	***** INSERTS BEFORE HERE
1079	06D2	A244		DATA	>A244	BIT (SEE ED50) 47
1080	06D4	1486		DATA	>1486	CRB 48
1081	06D6	3486		DATA	>3486	CRF 49
1082	06D8	695A		DATA	>695A	MEM 4A
1083	06DA	25DA		DATA	>25DA	MWD 4B
1084		06DC	EDCLE	EQU	\$	
1085			*			
1086		0082	IFB	EQU	IFX-EDCL+2	
1087		0080	ONB	EQU	ONX-EDCL+2	
1088		0014	PTRB	EQU	PRTX-EDCL+2	

```

1090      *
1091      * SECOND HALF OF PRIMITIVE TABLE
1092      *
1093 06DC 0000      DATA RUNP      RUN      A
1094 06DE 0000      DATA SIZE      SIZ(E)      B
1095 06E0 0000      DATA CONT      CON(TINUE)      C
1096 06E2 0000      DATA DEBUG$      MON(ITDR)      D
1097 06E4 0000      DATA >0000      <SPARE>      E
1098 06E6 0000      DATA >0000      <SPARE>      F
1099 06E8 0000      DATA >0000      <SPARE>      G
1100 06EA 0000      DATA >0000      <SPARE>      H
1101 06EC 0000      DATA >0000      <SPARE>      I
1102 06EE 0000      DATA >0000      <SPARE>      J
1103 06F0 0000      DATA >0000      <SPARE>      K
1104      *
1105      00AA EDCS EGU $-EDCL-2
1106 06F2 001E      DATA >001E      GOTO      01
1107 06F4 00AA      DATA >00AA      GOSUB      02
1108      06F7' BOA EGU $+1
1109 06F6 000A      DATA >000A      ELSE*      03
1110 06F8 47 BCPB BYTE BCP,0      REM*      04
1111 06FA FF80 CFF80 DATA >FF80      FOR*      05
1112 06FC 0000      DATA >0000      (LET)*      06
1113      06FF' B02 EGU $+1
1114 06FE 0002      DATA >0002      DATA      07
1115 0700 0028      DATA >0028      NEXT      08
1116 0702 049E      DATA >049E      ERROR      09
1117 0704 051C      DATA >051C      PRINT      0A
1118 0706 0018      DATA >0018      CALL      0B
1119 0708 0008      DATA >0008      LOAD      0C
1120 070A 052A      DATA >052A      INPUT      0D
1121 070C 0008      DATA >0008      READ      0E
1122 070E 93E8      DATA >93E8      RESTOR      0F
1123 0710 74AA      DATA >74AA      RETURN      10
1124 0712 0020      DATA >0020      STOP      11
1125 0714 0028      DATA >0028      UNIT      12
1126 0716 000A      DATA >000A      TIME      13
1127 0718 000A      DATA >000A      SAVE      14
1128 071A 000A      DATA >000A      BASE      15
1129 071C 2C02      DATA >2C02      ESCAPE      16
1130 071E 00E6      DATA >00E6      NOESC      17
1131 0720 6BC8      DATA >6BC8      RANDOM      18
1132 0722 0008      DATA >0008      BAUD      19
1133 0724 048A      DATA >048A      ENTER      1A
1134 0726 0028      DATA >0028      PLOT      1B
1135 0728 A3D8      DATA >A3D8      UNPLOT      1C
1136 072A 955E      DATA >955E      COLOUR      1D
1137 072C 014E      DATA >014E      PURGE      1E
1138 072E 0220      DATA >0220      GRAPH      1F
1139 0730 0028      DATA >0028      TEXT      20
1140 0732 0028      DATA >0028      WAIT      21
1141 0734 0024      DATA >0024      CHAR      22
1142 0736 9144      DATA >9144      NUMBER      23
1143 0738 0028      DATA >0028      LIST      24
1144 073A 036A      DATA >036A      RENUM      25
1145 073C 2D12      DATA >2D12      SPRITE      26
1146 073E 0160      DATA >0160      SHAPE      27
1147 0740 0028      DATA >0028      SPUT      28
1148 0742 0028      DATA >0028      SGET      29
1149 0744 0028      DATA >0028      BOOT      2A
    
```

1150	0746	0020	DATA >0020	SWAP	2B
1151	0748	0000	DATA >0000		2C
1152	074A	049E	DATA >049E	MOTOR	2D
1153	074C	0000	DATA >0000		2E
1154	074E	0000	DATA >0000		2F
1155	0750	0000	DATA >0000		30
1156	0752	0000	DATA >0000		31
1157	0754	0000	DATA >0000		32
1158	0756	0000	DATA >0000		33
1159	0758	0000	DATA >0000		34
1160	075A	0000	DATA >0000		35
1161	075C	0000	DATA >0000		36
1162	075E	0000	DATA >0000		37
1163	0760	0000	DATA >0000		38
1164					

*

```

1166      *
1167      * SYSTEM FUNCTION TABLE
1168      *
1169 0762 9882 EDIL DATA >9882 ABS 1B
1170 0764 9102 DATA >9102 ADR 1C
1171 0766 1CC2 DATA >1CC2 ASC 1D
1172 0768 7502 DATA >7502 ATN 1E
1173 076A 9BC6 DATA >9BC6 COS 1F
1174 076C 860A DATA >860A EXP 20
1175 076E 0C8C DATA >0C8C FRA 21
1176 0770 A392 DATA >A392 INT 22
1177 0772 3BD8 DATA >3BD8 LOG 23
1178 0774 C956 DATA >C956 KEY 24
1179 0776 7266 DATA >7266 SIN 25
1180 0778 9466 DATA >9466 SQR 26
1181 077A 9E66 DATA >9E66 SYS 27
1182 077C 1A68 DATA >1A68 TIC 28
1183 077E 71E6 DATA >71E6 SGN 29
1184      *
1185      * ASSIGNABLE FUNCTIONS
1186      *
1187 0780 A244 DATA >A244 BIT 2A
1188 0782 1486 DATA >1486 CRB 2B
1189 0784 3486 DATA >3486 CRF 2C
1190 0786 695A DATA >695A MEM 2D
1191 0788 25DA DATA >25DA MWD 2E
1192      *
1193      * CHARACTER FUNCTIONS
1194      *
1195 078A 7158 DATA >7158 LEN 2F
1196 078C 40DA DATA >40DA MCH 30
1197 078E 9BE0 DATA >9BE0 POS 31
1198 0790 63C6 DATA >63C6 COL 32
1199 0792 23DA DATA >23DA MOD 33
1200 0794 0000 DATA 0 34
1201 0796 0000 DATA 0 35
1202 0798 0000 DATA 0 36
1203 079A 0000 DATA 0 37
1204      079C' EDILE EQU *
1205      *
1206      * TRANSLATION TABLE INDEXED BY ASCII CODE.
1207      * NULLS ARE ILLEGAL.
1208      *
1209 079C FFFF DATA >FFFF !! FIX FOR ' ' CHARACTER !!
1210 079E 4744 EDSL DATA >4744,>3E43 ! " £ $
1211 07A2 4248 DATA >4248,>454C % & ' (
1212 07A6 4D5F DATA >4D5F,>5D3F ) * + ,
1213 07AA 5C00 DATA >5C00,>5E00 - . / 0
1214 07AE 0000 DATA >0000,>0000 1 2 3 4
1215 07B2 0000 DATA >0000,>0000 5 6 7 8
1216 07B6 003C DATA >003C,>4059 9 : ; <
1217 07BA 5657 B56 DATA >5657,>413D = > ? @
1218 07BC 4C46 DATA >4C46,>4D60 [ \ ] ^
1219      *
1220      07B9' B59 EQU EDSL+27
1221      07BB' B57 EQU EDSL+29
1222      07BF' B46 EQU EDSL+33

```

					WORD OPERATORS
1224	07C2'	EDGWOL	EGU	\$	
1225	07C2	54	LSTO	TEXT	'TO'
	07C3	4F			
1226	07C4	00		BYTE	0,>38
1227	07C6	54	LSTB	TEXT	'TAB'
	07C7	41			
	07C8	42			
1228	07C9	00		BYTE	0,>39
1229	07CB	53	LSST	TEXT	'STEP'
	07CC	54			
	07CD	45			
	07CE	50			
1230	07CF	00		BYTE	0,>3A
1231	07D1	54	LSTH	TEXT	'THEN'
	07D2	48			
	07D3	45			
	07D4	4E			
1232	07D5	00		BYTE	0,>3B
1233	07D7	4F	LSOR	TEXT	'OR'
	07D8	52			
1234	07D9	00		BYTE	0,>4E
1235	07DB	4C	LSLOR	TEXT	'LOR'
	07DC	4F			
	07DD	52			
1236	07DE	00		BYTE	0,>4F
1237	07E0	41	LSAN	TEXT	'AND'
	07E1	4E			
	07E2	44			
1238	07E3	00		BYTE	0,>50
1239	07E5	4C	LSLAN	TEXT	'LAND'
	07E6	41			
	07E7	4E			
	07E8	44			
1240	07E9	00		BYTE	0,>51
1241	07EB	4E	LSNT	TEXT	'NOT'
	07EC	4F			
	07ED	54			
1242	07EE	00		BYTE	0,>52
1243	07F0	4C	LSLNT	TEXT	'LNOT'
	07F1	4E			
	07F2	4F			
	07F3	54			
1244	07F4	00		BYTE	0,>53
1245	07F6	4C	LSLXO	TEXT	'LXOR'
	07F7	5B			
	07F8	4F			
	07F9	52			
1246	07FA	00		BYTE	0,>54,0


```

1248      *
1249      * LIST TRANSLATION TABLE INDEXED BY PSEUDO CODE.
1250      *      "_ ARE UNDEFINED.
1251      *
1252      07FD' EDLC      EQU      $
1253      07FD      3A      B3A      BYTE >3A      :
1254      07FE      40      B40      BYTE >40      @
1255      07FF      23      BYTE >23      f
1256      0800      2C      B2C      BYTE >2C      ,
1257      0801      3B      B3B      BYTE >3B      ;
1258      0802      3F      B3F      BYTE >3F      ?
1259      0803      25      BYTE >25      %
1260      0804      24      BYTE >24      $
1261      0805      22      BYTE >22      "
1262      0806      27      BYTE >27      '
1263      0807      5C      BYTE >5C      \
1264      0808      21      BYTE >21      !
1265      0809      26      BYTE >26      &
1266      080A      03      B03      BYTE >03      _      (UNUSED)
1267      080B      5B      B5B      BYTE >5B      [
1268      080C      5D      B5D      BYTE >5D      ]
1269      080D      28      BYTE >28      (
1270      080E      29      BYTE >29      )
1271      080F      1B      B1B      BYTE >1B      OR      (UNUSED)
1272      0810      6D      B6D      BYTE >6D      LOR      (UNUSED)
1273      0811      4E      B4E      BYTE >4E      AND      (UNUSED)
1274      0812      6F      B6F      BYTE >6F      LAND      (UNUSED)
1275      0813      38      B38      BYTE >38      NOT      (UNUSED)
1276      0814      42      DEF XB      BYTE DEF X-EDCL/2+1      LNOT      (UNUSED)
1277      0815      0B      N XTXB      BYTE N XTX-EDCL/2+1      LXOR      (UNUSED)
1278      0816      3D      BYTE >3D      ==
1279      0817      3D      B3D      BYTE >3D      =
1280      0818      3E      BYTE >3E      >
1281      0819      3E      BYTE >3E      >=
1282      081A      3C      BYTE >3C      <
1283      081B      3C      BYTE >3C      <=
1284      081C      3C      BYTE >3C      <>
1285      081D      2D      BYTE >2D      -
1286      081E      2B      BYTE >2B      +
1287      081F      2F      BYTE >2F      /
1288      0820      2A      BYTE >2A      *
1289      0821      5E      BYTE >5E      ^
1290      *
1291      0822      EVEN
1292      07C1' LCN1      EQU      EDLC->3C
1293      0644' LCN2      EQU      EDCL-2
1294      072C' LCN3      EQU      EDIL->36
1295      00AC      LCN4      EQU      EDCS+2
1296      0010      LNXT      EQU      N XTX-EDCL+2
1297      00B4      LDEF      EQU      DEF X-EDCL+2
    
```

```

1299      *
1300      * LIST COMMAND
1301      *   FORM :                               LISTS LINES :-
1302      *
1303      *   LIST                               FIRST LINE ==> LAST LINE
1304      *   LIST <ARG 1>                       ARG 1 ==> ARG 1
1305      *   LIST TO <ARG 1>                     FIRST LINE ==> ARG 1
1306      *   LIST <ARG 1> TO <ARG 2>             ARG 1 ==> ARG 2
1307      *
1308      0822' LST      EQU $
1309      0822 06A0      BL      @MODEOK          ;ABORT IF RUNNING
1310      0824 000E'
1311      0826 039C'      DATA TYPC$           ;OUT 'CRLF'
1312      0828 04C1      CLR      R1             ;DEFAULT START=1ST LINE
1313      082A 0706      SETO     R6             ;DEFAULT STOP = LAST LINE
1314      082C 9838      CB      *RB+,@B38      ; 'TO' ?
1315      082E 0813'
1316      0830 1309      JEQ      GSTOP          ;Y, GET STOP
1317      0832 0608      DEC      RB             ;N, BACKUP
1318      0834 06A0      BL      @CKEX          ;EXPRESSION ?
1319      0836 0000
1320      0838 1006      JMP      LSTOB          ;N, LIST WHOLE PROGRAM
1321      083A 2EC1      EVFIX     R1             ;Y, GET START ADDRESS
1322      083C C181      MOV      R1,R6          ;MAKE IT THE DEFAULT STOP TOO
1323      083E 0280      CI      R0,>3800        ; 'TO' ?
1324      0840 3800
1325      0842 1601      JNE      LSTOB          ;N, TAKE DEFAULT STOP
1326      0844 2EC6      GSTOP     EVFIX R6      ;Y, GET STOP LINE
1327      *
1328      * LIST PROGRAM LINES
1329      *   R1= START LINE NUMBER
1330      *   R6= STOP   LINE NUMBER
1331      0846 C220      LSTOB     MOV      @VNT,RB ;GET TABLE ADR
1332      0848 057A'
1333      084A 0648      DECT     RB             ;BACKUP
1334      084C 8B08      LSTO      C      RB,@SLT ;DONE?
1335      084E 0590'
1336      0850 1215      JLE      LST2          ;Y
1337      0852 0228      AI      RB,-4          ;MOVE TO NEXT ENTRY
1338      0854 FFFC
1339      0856 8601      C      R1,*RB          ;SAME?
1340      0858 15F9      JGT      LSTO          ;
1341      085A 0720      SETO     @DCNT          ;RESET INDENT COUNTER
1342      085C 0000
1343      085E 8B08      LST1      C      RB,@SLT ;DONE?
1344      0860 084E'
1345      0862 1A0C      JL      LST2          ;Y
1346      0864 C078      MOV      *RB+,R1        ;N, GET LINE NUMBER
1347      0866 8181      C      R1,R6          ;PAST LAST LINE ?
1348      0868 1B09      JH      LST2          ;Y, END LIST
1349      * MOV      @IOB,R7 ;GET BUFFER ADR !DONE IN LSTL
1350      1341 086A 06A0      BL      @LSTL      ;LIST LINE
1351      086C 0898'
1352      *
1353      * TEST FOR HALT OUTPUT
1354      *
1355      1345 086E 0420      BLWP     @HALTO$
1356      0870 0000
1357      1346 0872 0228      AI      RB,-6      ;PREPARE FOR NEXT LINE
1358      0874 FFFA

```

1347	0876	0000		DATA TYPBE\$; TERMINATE & OUTPUT LINE
1348	0878	0826'		DATA TYPC\$; OUT 'CRLF'
1349	087A	10F1		JMP LST1	; LOOP TILL DONE
1350	087C	0460	LST2	B @CRLF	; EXIT TO CRLF
	087E	0000			

```

1352          *LIST GOTO'S AND GOSUB'S
1353          *
1354 0880 DDE0 LSTG1  MOVB @B2C,*R7+          ;OUT ",
          0882 0800'
1355 0884 0583          INC R3          ;SKIP ','
1356          *
1357 0886 D073 LSTG  MOVB *R3+,R1          ;GET LINE NUMBER
1358 0888 06C1          SWPB R1
1359 088A D073          MOVB *R3+,R1
1360 088C 06C1          SWPB R1
1361 088E 2F41          OUTINT R1          ;CONVERT
1362 0890 9813          CB  *R3,@B3F          ;CHECK NEXT BYTE FOR ",
          0892 0802'
1363 0894 13F5          JEQ  LSTG1          ;Y, ANOTHER GOTO, OR GOSUB
1364 0896 1050          JMP  LSTLX          ;N, PROCESS
    
```

```

1366      *LIST LINE
1367      *      BL @LSTL
1368      *
1369      *      IN  R1 = LINE #
1370      *      *R8 = PBC
1371      *      R15 = SP
1372      *
1373      *      OUT R7 = IOB
1374      *      PRESERVE R6,R8
1375      *
1376 0898 C28B  LSTL  MOV  R11,R10      ;SAVE RETURN
1377 089A 0000      DATA FTM$      ;FORCE TO TEXT MODE
1378 089C C1E0      MOV  @IOB,R7      ;GET IO BUFFER POINTER
1379      089E 037A'
1379 08A0 C0D8      MOV  *R8,R3      ;GET PBC
1380 08A2 A0E0      A      @BUS,R3      ;MAKE DISPLACEMENT INTO POINTER
1380      08A4 0568'
1381 08A6 020F      LI   R15,>2000      ;GET SPACE
1381      08A8 2000
1382 08AA DDCF      MOV  R15,*R7+      ;OUT SPACE
1383 08AC 2F41      OUTINT R1      ;CONVERT LINE NUMBER
1384 08AE C0A0      MOV  @DCNT,R2      ;GET INDENT COUNT
1384      08B0 085C'
1385 08B2 9813      CB   *R3,@B03      ;ELSE?
1385      08B4 080A'
1386 08B6 1601      JNE  LSTL1      ;N
1387 08B8 0642      DECT R2      ;Y, INDENT 2 MORE SPACES
1388      *
1389 08BA DDCF  LSTL1 MOV  R15,*R7+      ;OUT SPACE
1390 08BC 0582      INC  R2      ;MORE?
1391 08BE 11FD      JLT  LSTL1      ;Y
1392 08C0 9813      CB   *R3,@NXTXB      ;NEXT?
1392      08C2 0815'
1393 08C4 1601      JNE  LSTL2      ;N
1394 08C6 0607      DEC  R7      ;Y, INDENT 1 LESS
1395      *
1396 08C8 DDCF  LSTL2 MOV  R15,*R7+      ;OUT SPACE
1397      *
1398 08CA 04C0  LSTL3 CLR  R0
1399 08CC D033      MOV  *R3+,R0      ;GET TYPE
1400 08CE 0280      CI   R0,>0600      ;IMPLIED LET?
1400      08D0 0600
1401      08D0'  LETB  EQU  #-2
1402 08D2 131E      JEQ  LSTL4      ;Y
1403      *
1404 08D4 0280      CI   R0,PASBX*128      ;(*) EXTENDED COMMAND ?
1404      08D6 4600
1405 08D8 1606      JNE  LETP#1      ;N, TRY PRINT
1406 08DA DDE0      MOV  @B2A,*R7+      ;Y, OUT A '*'
1406      08DC 00E8'
1407 08DE DDF3  OUTEXT MOV  *R3+,*R7+      COPY OVER CHARACTER
1408 08E0 16FE      JNE  OUTEXT      LOOP TILL NULL FOUND
1409 08E2 0607      DEC  R7      BACKUP
1410 08E4 1008      JMP  LETB#2      OUT ' ' AND CONTINUE
1411 08E6 0280  LETP#1 CI   R0,PQMBX*128      ;(?) COMPRESSED PRINT ?
1411      08E8 4500
1412 08EA 1603      JNE  LETB#1      ;N, TRY ( )
1413 08EC DDE0      MOV  @B3F,*R7+      ;Y, OUT A '?'
1413      08EE 0802'
1414 08F0 1005      JMP  LETB#2      ;OUT SPACE & CONTINUE

```

```

1415 08F2 0280 LETB#1 CI RO,PSCBX*128 ; ( ) COMPRESSED PRINT ?
      08F4 3F00
1416 08F6 1604 JNE LSTL3A ; N, NORMAL CODE
1417 08F8 DDE0 MOVB @B3B,*R7+ ; Y, OUT ';'
      08FA 0801'
1418 08FC 0870 LETB#2 SRA RO,7 ; ADJUST RO
1419 08FF 1003 JMP LSTL3B ; OUT SPACE & CONTINUE
1420 0900 06A0 LSTL3A BL @LWRD ; LIST COMMAND TYPE
      0902 0AAE'
1421 0904 0644' DATA LCN2 ; EDCL-2
1422 0906 DDCF LSTL3B MOVB R15,*R7+ ; OUT SPACE
1423 0908 0280 CI RO,BCP*2 ; BIT,CRB,CRF,MEM?
      090A 008E
1424 090C 1A01 JL LSTL4 ; N
1425 090E 0607 DEC R7 ; Y, ELIMINATE SPACE
1426 *
1427 0910 C380 LSTL4 MOV RO,R14 ; SAVE TYPE
1428 0912 0280 CI RO,>3*2 ; CHECK TYPE
      0914 0006
1429 0916 11B7 JLT LSTG ; GOTO OR GOSUB
1430 0918 13D8 JEQ LSTL3 ; ELSE
1431 091A 0280 CI RO,>5*2
      091C 000A
1432 091E 115D JLT LSTRM ; LT - REMARK STATEMENT
1433 0920 1502 JGT LSTL5 ; GT - CONTINUE LOOKING
1434 0922 0620 DEC @DCNT ; FOR, DECREMENT COUNTER
      0924 08B0'
1435 *
1436 0926 0280 LSTL5 CI RO,LNXT ; NEXT?
      0928 0010
1437 092A 1605 JNE LSTL6 ; N
1438 092C 05A0 INC @DCNT ; Y, INCREMENT COUNTER
      092E 0924'
1439 0930 1102 JLT LSTL6 ; COUNTER OK
1440 0932 0720 SETO @DCNT ; RESET COUNTER
      0934 092E'
1441 0936 C380 LSTL6 MOV RO,R14 ; SAVE TYPE

```

1443	0938	D033	LSTLX	MOVB	*R3+,R0	; GET CODE
1444	093A	1336		JEG	LSTLE	; DONE
1445	093C	9800		CB	RO,@B1B	; USER FUNCTION?
	093E	080F				
1446	0940	1A59		JL	LSTFNP	; Y
1447	0942	9800		CB	RO,@B3B	; SYSTEM FUNCTION?
	0944	0813				
1448	0946	1A5D		JL	LSTSF	; Y
1449	0948	0980		SRL	RO,B	; READY RO,R4
1450	094A	C100		MOV	RO,R4	
1451	094C	0280		CI	RO,>3C	; TO, TAB, STEP, THEN?
	094E	003C				
1452	0950	1A62		JL	LSTTT	; Y
1453	0952	0280		CI	RO,>4E	; < OR?
	0954	004E				
1454	0956	1A03		JL	LSTLX0	; Y
1455	0958	0280		CI	RO,>54	; > LXOR?
	095A	0054				
1456	095C	125A		JLE	LSTTI	; N
1457			*			
1458	095E	0280		LSTLX0	CI RO,>62	; CHARACTER?
	0960	0062				
1459	0962	1A24		JL	LSTCR	; Y
1460	0964	0280		CI	RO,>6F	; CONSTANT?
	0966	006F				
1461	0968	1A61		JL	LSTCN	; Y, INTEGERS
1462	096A	1373		JEG	LSTCNF	; Y, FLOATING POINT
1463			*			
1464	096C	C160		MOV	@VNT,R5	; N, VARIABLE
	096E	0848				
1465	0970	A000		A	RO,RO	; X 2
1466	0972	A140		A	RO,R5	; INDEX & GET VARIABLE
1467	0974	04C2		CLR	R2	
1468	0976	C165		MOV	@-2*>70(5),R5	
	0978	FF20				
1469	097A	1503		JGT	LSTVN	; DIMENSIONED?
1470	097C	0505		NEG	R5	; Y
1471		0981	B4A	EQU	#+3	
1472	097E	0202		LI	R2,>4A	; OUT [
	0980	004A				
1473			*			
1474	0982	2160		LSTVN	CDC @C3B0,R5	; LETTER + NUMBER?
	0984	0214				
1475	0986	1608		JNE	LSTVN1	; N
1476	0988	06A0		BL	@LWRD0	; Y, OUT CHARACTER
	098A	0AD4				
1477	098C	0921		SRL	R1,2	
1478	098E	C045		MOV	R5,R1	; OUT £
1479	0990	0241		ANDI	R1,>7F	; MASK
	0992	007F				
1480	0994	2F41		OUTINT	R1	; CONVERT
1481	0996	1005		JMP	LSTVN2	
1482			*			
1483	0998	0A15		LSTVN1	SLA R5,1	; REGULAR VARIABLE
1484	099A	020D		LI	R13,LSTVN2	
	099C	09A2				
1485	099E	0460		B	@LWRD2	; LIST
	09A0	0AB8				
1486			*			
1487	09A2	C102		LSTVN2	MOV R2,R4	; "£ NEEDED?

```
1488 09A4 162B      JNE  LSTCR1      ; Y
1489 09A6 10C8      JMP  LSTLX
1490                *
1491 09A8 75D7      LSTLE SB  *R7,*R7      ; PUT NULL ON END
1492 09AA 045A      B    *R10
```



```

1494 09AC DDE4 LSTCR MOV B @LCN1(4),*R7+ ;EDLC->3C
      09AE 07C1'
1495 09B0 06C0 SWPB R0
1496 09B2 06A0 BL @JMPRO ;LOOK FOR CODES
      09B4 029C'
1497 09B6 20 LSTCRT BYTE LSTCR0-LSTCRT/2,>44 "
1498 09B8 20 BYTE LSTCR0-LSTCRT/2,>45 '
1499 09BA 0C BYTE LSTTR-LSTCRT/2,>47 !
1500 09BC 1D BYTE LSTL2P-LSTCRT/2,>3C :
1501 09BE 0000 DATA 0
1502 09C0 06A0 BL @LSTCR2 ;LOOK FOR DOUBLES (>AAII)
      09C2 09E4'
1503 * == >= <= <>
1504 09C4 3D55 DATA >3D55,>3D5B,>3D5A,>3E5B,>0000
1505 *
1506 *TAIL REMARK
1507 *
1508 09CE 0607 LSTTR DEC R7 ;BACKUP OVER !
1509 09D0 DDCF MOV B R15,*R7+ ;OUT SPACE
1510 09D2 DDCF MOV B R15,*R7+
1511 09D4 DDE4 MOV B @LCN1(4),*R7+
      09D6 07C1'
1512 09D8 1001 JMP LSTRM1
1513 *
1514 *REMARK ENTRY
1515 *
1516 09DA 0607 LSTRM DEC R7 ;REM, MOVE BACK 1 CHAR
1517 *
1518 09DC DDF3 LSTRM1 MOV B *R3+,*R7+ ;MOVE INTO LINE
1519 09DE 16FE JNE #-2
1520 09E0 0607 DEC R7 ;BACKUP
1521 09E2 045A B *R10 ;RETURN
  
```

```

1523      *
1524 09E4 D5FB LSTCR2 MOVB *R11+, *R7      ; MOVE CHARACTER INTO LINE
1525 09E6 13A8      JEQ LSTLX              ; NOT FOUND
1526 09E8 9ECO      CB R0, *R11+          ; FOUND?
1527 09EA 16FC      JNE LSTCR2             ; N, KEEP LOOKING
1528 09EC 05B7      INC R7                 ; Y, SKIP OVER CHARACTER
1529 09EE 10A4      JMP LSTLX              ; CONTINUE
1530 09F0 0460 LSTL2P B @LSTL2
      09F2 08C8'

1531      *
1532 09F4 1032 LSTFNP JMP LSTFN
1533 09F6 DDF3 LSTCR0 MOVB *R3+, *R7+      ; MOVE IN CHARACTER STRING
1534 09F8 16FE      JNE #-2                ; LOOP UNTIL NULL
1535 09FA 0607      DEC R7                  ; BACKUP OVER NULL
1536      *
1537 09FC DDE4 LSTCR1 MOVB @LCN1(4), *R7+   ; EDLC->3C
      09FE 07C1'
1538 0A00 109B      JMP LSTLX
1539      *
1540 0A02 06A0 LSTSF BL @LWRD              ; LIST SYSTEM FUNCTION
      0A04 0AAE'
1541 0A06 072C'      DATA LCN3            ; EDIL->36
1542 0A08 9813 LSTSF1 CB *R3, @B4A         ; [?
      0A0A 09B1'
1543 0A0C 1395      JEQ LSTLX              ; Y, NO SPACE
1544 0A0E DDCF LSTSP MOVB R15, *R7+        ; OUT SPACE
1545 0A10 1093 LSTLXP JMP LSTLX
1546      *
1547 0A12 0224 LSTTI AI R4, ->12           ; MOVE BACK BY THEN
      0A14 FFEE

1548      *
1549 0A16 DDCF LSTTT MOVB R15, *R7+        ; OUT SPACE
1550 0A18 A104      A R4, R4                ; DOUBLE INDEX
1551 0A1A C164      MOV @LSTTL->70(4), R5  3B*2
      0A1C 0A2B'

1552      *
1553 0A1E DDF5 LSTT1 MOVB *R5+, *R7+      ; MOVE LETTER
1554 0A20 D015      MOVB *R5, R0           ; CHECK END
1555 0A22 16FD      JNE LSTT1              ; LOOP
1556 0A24 02B4      CI R4, >3B*2          ; THEN?
      0A26 0076
1557 0A28 13E3      JEQ LSTL2P             ; Y
1558 0A2A 10F1      JMP LSTSP              ; OUT SPACE

```

```

1560 0A2C 0280 LSTCN CI RO,>6D ; -1 THRU 9?
      0A2E 006D
1561 0A30 1404 JHE LSTCN1 ; N
1562 0A32 0220 AI RO,->63 ; Y
      0A34 FF9D
1563 0A36 C040 MOV RO,R1
1564 0A38 1007 JMP LSTCN2 ; CONVERT
1565 *
1566 0A3A D073 LSTCN1 MOVB *R3+,R1 ; INTEGER
1567 0A3C 06C1 SWPB R1
1568 0A3E D073 MOVB *R3+,R1
1569 0A40 06C1 SWPB R1
1570 0A45' B6E EQU $+3
1571 0A42 0280 CI RO,>6E ; HEX?
      0A44 006E
1572 0A46 1302 JEQ LSTCN3 ; Y
1573 *
1574 0A48 2F41 LSTCN2 OUTINT R1 ; N, CONVERT
1575 0A4A 10E2 JMP LSTLXP
1576 *
1577 0A4C 06A0 LSTCN3 BL @HOUT ; OUT HEX
      0A4E 0000
1578 0A50 10DF JMP LSTLXP
1579 *
1580 0A52 2F13 LSTCNF OUTFP *R3 ; OUTPUT FP &
1581 0A54 0223 AI R3,6 ; INCREMENT OVER NUMBER
      0A56 0006
1582 0A58 10DB JMP LSTLXP
1583 *
1584 *USER FUNCTION ENTRY
1585 *
1586 0A5A DDE0 LSTFN MOVB @B46,*R7+ ; OUT "F"
      0A5C 07BF'
1587 0A5E DDE0 MOVB @B4E,*R7+ ; OUT "N"
      0A60 0811'
1588 0A62 0220 AI RO,>4000
      0A64 4000
1589 0A66 DDC0 MOVB RO,*R7+ ; OUT LETTER
1590 0A68 02BE CI R14,LDEF ; DEF?
      0A6A 00B4
1591 0A6C 16CD JNE LSTSF1 ; N, CHECK FOR (
1592 0A6E C3A0 MOV @VNT,R14 GET TABLE ADDR & CLEAR 'LDEF'
      0A70 096E'
1593 0A72 9813 CB *R3,@B56 ; Y, ARGUMENTS?
      0A74 07BA'
1594 0A76 13CC JEQ LSTLXP ; N
1595 0A78 DDE0 MOVB @B5B,*R7+ ; Y, OUT "I"
      0A7A 080B'
1596 *
1597 0A7C D7B3 LSTFD1 MOVB *R3+,*R14 ; SAVE DUMMY NAME
1598 0A7E C07E MOV *R14+,R1
1599 0A80 0821 SRA R1,2
1600 0A82 06A0 BL @LWRD01 ; MAKE LETTER & STORE
      0A84 0ADE'
1601 0A86 9813 CB *R3,@B56 ; =?
      0A88 07BA'
1602 0A8A 1303 JEQ LSTFD2 ; Y
1603 0A8C DDE0 MOVB @B2C,*R7+ ; N, OUT ', '
      0A8E 0800'
1604 0A90 10F5 JMP LSTFD1 ; LOOP AGAIN

```

```
1605      *
1606 0A92 DDE0 LSTFD2 MOVB @B5D, *R7+      ; OUT '1'
      0A94 080C'
1607 0A96 10BC      JMP LSTLXP
1608      *
1609 0A98 07C2' LSTTL DATA LSTD, LSTB
1610 0A9C 07CB'      DATA LSST, LSTH
1611 0AA0 07D7'      DATA LSOR, LSLOR
1612 0AA4 07E0'      DATA LSAN, LSLAN
1613 0AAB 07EB'      DATA LSNT, LSLNT
1614 0AAC 07F6'      DATA LSLXD
```

```

1616          *LIST WORD
1617          *      BL      @LWRD
1618          *      ADDR
1619          *
1620 0AAE C13B  LWRD  MOV  *R11+,R4      ; GET ADR OF WORD
1621 0AB0 C34B      MOV  R11,R13      ; SAVE RETURN
1622 0AB2 0B70      SRA  R0,7        ; SWAP AND X 2
1623 0AB4 A100      A    R0,R4        ; INDEX
1624          *
1625 0AB6 C154  LWRD1 MOV  *R4,R5      ; GET WORD
1626 0AB8 06A0  LWRD2 BL    @LWRD0    ; OUT FIRST CHARACTER
      0ABA 0AD4'
1627 0ABC 0A71      SLA  R1,7
1628 0ABE 06A0      BL    @LWRD0      ; OUT 2ND OR 5TH CHARACTER
      0AC0 0AD4'
1629 0AC2 0A21      SLA  R1,2
1630 0AC4 06A0      BL    @LWRD0      ; OUT 3RD OR 6TH CHARACTER
      0AC6 0AD4'
1631 0AC8 0931      SRL  R1,3
1632 0ACA 0224      AI    R4,@LCN4    ; MOVE TO NEXT HALF (EDCS+2)
      0ACC 00AC
1633 0ACE 0915      SRL  R5,1        ; ANOTHER HALF?
1634 0AD0 1BF2      JOC  LWRD1      ; Y
1635 0AD2 045D      B    *R13        ; N, RETURN
1636          *
1637 0AD4 C045  LWRD0 MOV  R5,R1      ; LOAD TEMP
1638 0AD6 04BB      X    *R11+      ; EXECUTE SHIFT
1639 0ADB 0241      ANDI R1,>1F00    ; MASK
      0ADA 1F00
1640 0ADC 1303      JEQ  LWRD02      ; RETURN
1641          *
1642 0ADE 0221  LWRD01 AI    R1,>4000  ; ADD LETTER BITS
      0AEO 4000
1643 0AE2 DDC1      MOVB R1,*R7+    ; MOVE OUT
1644 0AE4 045B  LWRD02 B    *R11     ; RETURN
1645          END
    
```

NO ERRORS,

NO WARNINGS

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.RUN
OBJECT ACCESS NAME= ADHOC.OBJ.RUN
LISTING ACCESS NAME= ADHOC.LST.RUN
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

```

0002          IDT 'RUN'
0003          *
0004          * THIS MODULE CAUSES THE STORED BASIC PROGRAM TO BE
0005          * INTERPRETED. ON ENTRY IT CAUSES THE USER VARIABLE
0006          * SPACE TO BE CLEARED.
0007          *
0008          * A LIST OF ALL BASIC STATEMENT ENTRY POINTS IS MAINTAINED
0009          * IN THIS MODULE.
0010          *
0011          * ALSO CONTAINED ARE THE ROUTINES: CLRV (CLEAR USER VARIABLE
0012          * SPACE), REST (RESET THE DLC AND DDM SYSTEM POINTERS) AND
0013          * THE BASIC STATEMENT HANDLER 'ELSE'.
0014          *
0015          DEF LINE, LINE0, LINE2, LINE5
0016          DEF NLIN, NLINO, RUNP, RUNP1, CLRV, REST
0017          DEF NXTXB, DATXB
0018          REF B2E, PLF, ESCFLG, POPY, ENTERY, MSPY
0019          REF MAGY, LST, BAUD, GOTY
0020          REF EXTNDY
0021          REF SLN, RENUMS, BASY, BITY, CLLY, CRBY
0022          REF SGETY, SPUTY, CRFY, DEFY, DIMY
0023          REF SPRITY, SHAPEY, ERRY, ESCY, FORY
0024          REF BOOTY, GOSY, IFY, INPY, LDPY
0025          REF SWAPY, LETY, MEMY, NEWY, NOEY
0026          REF NXTY, ONY, PRTY
0027          REF RANY, RDDY, RNWY, RTNY
0028          REF SAVY, STPY, TIMY, UNTY
0029          REF MOTORY, RANDS
0030          REF NUMY, PLOTY, UPLLOTY, COLORY
0031          REF GRAPHY, TEXTY, WAITY, LDOSY
0032          REF CRLF, TYPC$, STPYA, MWDY
0033          REF NVS, GSS, VNT, VDT, DDM, DLC, GSC
0034          REF BUS, DLIM, MODE, PLC, SLT, ELSF
0035          REF PURGY, STACNT, IOB, RBSTOR, TRAF LG
0036          REF TOFY, TONY, TYPBE$, MOVEB, MOVEL
0037          REF F$WHO
0038          *
0039          * XOP EQUATES
0040          *
0041          2FB0 ERROR EQU >2FB0
0042          2FA0 ERROR2 EQU ERROR+>20
0043          DXOP OUTINT, 13
  
```

```

0045      *
0046      * CLR V - CLEAR R1 THRU EVS
0047      *
0048 0000 C820 CLR V   MOV  @IOB,@NVS      ; SET NVS TO IOB POINTER
      0002 0000
      0004 0000
0049 0006 C820      MOV  @GSS,@GSC      ; RESET GDSUB STACK
      0008 0000
      000A 0000
0050 000C 04F1      CLR  *R1+          ; CLEAR MEMORY
0051 000E 0281      CI   R1,RANDS      ; REACHED EVS YET?
      0010 0000
0052 0012 1AFC      JL   #-6          ; N - BACK FOR NEXT WORD
0053 0014 04E0      CLR  @RBSTOR      ; CLEAR ENTER'S R8 STORE
      0016 0000
0054      *
0055      * REST - RESET DATA PTR
0056      *
0057 0018 C820 REST  MOV  @VNT,@DLC      ; RESET DLC TO VNT
      001A 0000
      001C 0000
0058 001E 04E0      CLR  @DDM          ; ZERO DDM
      0020 0000
0059 0022 045B      RT
  
```



```

0061      *
0062      * RUN STATEMENT
0063      *
0064 0024 0000  RUNP  DATA TYPC#           ; OUT CRLF
0065      *
0066 0026 0720  RUNP1 SET0 @MODE           ; SET MODE TO RUN
0067      0028 0000
0067 002A 04E0          CLR  @ESCFLG       ; ENABLE ESCAPE
0068      002C 0000
0068 002E 04E0          CLR  @PLF         ; FLAG AS NOT LOAD/SAVE
0069      0030 0000
0069 0032 C820          MOV  @VNT,@PLC     ; SET PLC TO START
0070      0034 001A'
0070      0036 0000
0070 0038 C060          MOV  @VDT,R1       ; CLEAR VDT THRU EUS
0071      003A 0000
0071 003C 06A0          BL   @CLRV
0072      003E 0000'
0072 0040 1013          JMP  LINE
0073      *
0074      * MULTIPLEXOR
0075      *
0076 0042 0280  NLIN1  CI   R0,>3C00       ; ' ' ?
0077      0044 3C00
0077 0046 1603          JNE  NLIN1A        ; N, TRY OTHER DELIMITERS
0078 0048 05A0          INC  @STACNT       ; Y, COUNT
0079      004A 0000
0079 004C 1021          JMP  LINE2         ; GET REST OF LINE
0080 004E 0280  NLIN1A CI   R0,>3B00       ; 'THEN' ?
0081      0050 3B00
0081 0052 131E          JEQ  LINE2         ; Y, GET REST OF LINE
0082 0054 0280          CI   R0,>4700     ; '!' ?
0083      0056 4700
0083 0058 1307          JEQ  LINE         ; Y, NEW LINE
0084      *
0085 005A 2FA0  ERR37  DATA ERROR2,37     ; N - ILLEGAL DELIMITER
0086      *
0087      * NEXT LINE OR ":
0088      *
0089 005E D838  NLINO  MOVB *R8+,@DLIM     ; SAVE & SKIP DELIMITER
0090      0060 0000
0090      *
0091 0062 C020  NLIN   MOV  @DLIM,R0        ; LOOK AT DELIMITER
0092      0064 0060'
0092 0066 16ED          JNE  NLIN1        ; NOT EOL, CHECK IT
0093      *
0094      * NEXT LINE
0095      *
0096 0068 C020  LINE   MOV  @MODE,R0       ; LOOK AT MODE
0097      006A 0028'
0097 006C 133F          JEQ  LINE3        ; IDLE - THEN FINISHED
0098      006E 006E'  LINE5 EQU  $
0099 006E C220          MOV  @PLC,R8       ; GET PROGRAM LINE COUNTER
0100      0070 0036'
0100 0072 0228          AI   R8,-4        ; MOVE TO NEXT LINE
0101      0074 FFFC
0101 0076 8808          C    R8,@SLT      ; ANY MORE STATEMENTS?
0102      0078 0000
0102 007A 1A3A          JL   LINE4        ; N
0103 007C C808  LINE0  MOV  R8,@PLC       ; Y - UPDATE PLC

```

```

007E 0070'
0104 0080 C828      MOV @-2(R8),@SLN      ; SAVE LINE #
      0082 FFFE
      0084 0000
0105 0086 C218      MOV *R8,R8            ; GET PBC
0106 0088 A220      A @BUS,R8             ; GET POINTER TO BOL
      008A 0000
0107 008C 04E0      CLR @STACNT           ; ZERO STATEMENT COUNTER
      008E 004A'
0108
      *
0109 0090 04E0 LINE2 CLR @F$WHO           ; FLAG IN BASIC
      0092 0000
0110 0094 04C2 NOTD CLR R2
0111 0096 D0B8      MOVB *R8+,R2          ; GET STATEMENT CODE
0112 0098 0972      SRL R2,7              ; GET INDEX
0113 009A 0282      CI R2;2*MAXSTA        ; VALID ?
      009C 0096
0114 009E 1B24      JH SYSERR             ; N, ERROR !!!
0115 00A0 C262      MOV @MXLST-2(2),R9    ; Y, GET STATEMENTS ENTRY PT
      00A2 00FC'
0116 00A4 1321      JEQ SYSERR            ; 'O' SYSTEM ERROR
0117
      *
0118 00A6 C2E0      MOV @TRAFLG,R11       ; IN TRACE ?
      00A8 0000
0119 00AA 131D      JEQ NOTRC             ; N, DONT OUT TRACE INFO
0120 00AC C2E0      MOV @MODE,R11         ; Y, IDLE?
      00AE 006A'
0121 00B0 131A      JEQ NOTRC            ; Y, DONT BOTHER THEN
0122
      *
0123
      * OUT TRACE INFORMATION IN THE FORM ' {line No.} '
0124
      *
0125 00B2 06A0      BL @MOVEB             ; GET IOB & ENTER TEXT
      00B4 0000
0126 00B6 20        TEXT ' Statement No. '
      00B7 53
      00B8 74
      00B9 61
      00BA 74
      00BB 65
      00BC 6D
      00BD 65
      00BE 6E
      00BF 74
      00C0 20
      00C1 4E
      00C2 6F
      00C3 2E
      00C4 20
0127 00C5 00        BYTE 0

0
0128 00C6 C2E0      MOV @PLC,R11          GET PLC
      00C8 007E'
0129 00CA 2F6B      OUTINT @-2(R11)       OUT LINE #
      00CC FFFE
0130 00CE C2E0      MOV @STACNT,R11      GET STATEMENT INDENT COUNT
      00D0 008E'
0131 00D2 1303      JEQ DTRC1            0, LEAVE LINE No.
0132 00D4 DDE0      MOV @B2E,*R7+        OUT ' '
      00D6 0000
0133 00D8 2F4B      OUTINT R11           OUT INDENT COUNT
0134 00DA 06A0      DTRC1 BL @MOVEB      ADD IN REST OF TEXT

```

```
00DC 0000
0135 00DE 20      TEXT ' '
00DF 20
00E0 20
0136 00E1 00      BYTE 0
0137 00E2 0000     DATA TYPBE$      OUT I/O BUFFER
0138 00E4 0024'     DATA TYPC$      OUT CRLF
0139              *
0140              *      !!!!! WARNING !!!!!
0141              *
0142              *      R2 MUST STILL CONTAIN 2*PCODE
0143              *
0144 00E6 0459     NOTRC B      *R?      ; GOTO STATEMENT HANDLER .
0145              *
0146 00E8 2FA0     SYSERR DATA ERROR2,-1      ; SYSTEM ERROR
0147              *
0148              *      ONLY HAD 1 LINE TO EXECUTE - ORIGINALLY ENTERED VIA EDIT
0149              *
0150 00EC 0460     LINE3 B      @CRLF
00EE 0000
0151              *
0152              *      FINISHED PROGRAM
0153              *
0154 00F0 0460     LINE4 B      @STPY      ; ESCAPE
00F2 0000
```

```
0156      *
0157      * ELSE STATEMENT
0158      *
0159 00F4 C020 ELSY  MOV  @ELSF,R0      ;CHECK ELSE FLAG
      00F6 0000
0160 00F8 16CB      JNE  LINE2      ;SET - CONTINUE TO EXECUTE
0161 00FA 10B6      JMP  LINE      ;RESET - IGNORE LINE
```

```

0163      *
0164      * RUN TABLE
0165      *
0166 00FC 00F2'      DATA STPY      00 TRAP NULLS
0167 00FE 0000      MXLST DATA GOTY      GOTO      01
0168 0100 0000      DATA GOSY      GOSUB      02
0169 0102 00F4'      DATA ELSY      ELSE      03
0170 0104 0068'      DATA LINE      REM      04
0171 0106 0000      DATA FORY      FOR      05
0172 0108 0000      DATA LETY      (LET)      06
0173 010A 0068'      DATA LINE      DATA      07
0174      0007      DATB EQU $-MXLST/2
0175 010C 0000      DATA NXTY      NEXT      08
0176      0008      NXTB EQU $-MXLST/2
0177 010E 0000      DATA ERRY      ERROR      09
0178 0110 0000      DATA PRTY      PRINT      0A
0179 0112 0000      DATA CLLY      CALL      0B
0180 0114 0000      DATA LDPY      LOAD      0C
0181 0116 0000      DATA INPY      INPUT      0D
0182 0118 0000      DATA RDDY      READ      0E
0183 011A 0000      DATA RNWY      RESTOR      0F
0184 011C 0000      DATA RTNY      RETURN      10
0185 011E 0000      DATA STPYA      STOP      11
0186 0120 0000      DATA UNTY      UNIT      12
0187 0122 0000      DATA TIMY      TIME      13
0188 0124 0000      DATA SAVY      SAVE      14
0189 0126 0000      DATA BASY      BASE      15
0190 0128 0000      DATA ESCY      ESCAPE      16
0191 012A 0000      DATA NOEY      NOESC      17
0192 012C 0000      DATA RANY      RANDOM      18
0193 012E 0000      DATA BAUD      BAUD      19
0194 0130 0000      DATA ENTERY      ENTER      1A
0195 0132 0000      DATA PLOTY      PLOT      1B
0196 0134 0000      DATA UPLOTY      UNPLOT      1C
0197 0136 0000      DATA COLORY      COLOUR      1D
0198 0138 0000      DATA PURGY      PURGE      1E
0199 013A 0000      DATA GRAPHY      GRAPH      1F
0200 013C 0000      DATA TEXTY      TEXT      20
0201 013E 0000      DATA WAITY      WAIT      21
0202 0140 0000      DATA LDOSY      CHAR      22
0203 0142 0000      DATA NUMY      NUMBER      23
0204 0144 0000      DATA LST      LIST      24
0205 0146 0000      DATA RENUMS      RENUM      25
0206 0148 0000      DATA SPRITY      SPRITE      26
0207 014A 0000      DATA SHAPEY      SHAPE      27
0208 014C 0000      DATA SPUTY      SPUT      28
0209 014E 0000      DATA SGETY      SGET      29
0210 0150 0000      DATA BOOTY      BOOT      2A
0211 0152 0000      DATA SWAPY      SWAP      2B
0212 0154 0000      DATA >0000      2C
0213 0156 0000      DATA MOTORY      MOTOR      2D
0214 0158 0000      DATA >0000      2E
0215 015A 0000      DATA >0000      2F
0216 015C 0000      DATA >0000      30
0217 015E 0000      DATA >0000      31
0218 0160 0000      DATA >0000      32
0219 0162 0000      DATA >0000      33
0220 0164 0000      DATA >0000      34
0221 0166 0000      DATA >0000      35
0222 0168 0000      DATA >0000      36

```

0223	016A	0000	DATA >0000		37
0224	016C	0000	DATA >0000		38
0225	016E	0000	DATA MAGY	MAG	39
0226	0170	0000	DATA TOFY	TOF	3A
0227	0172	0000	DATA TONY	TON	3B
0228	0174	0000	DATA POPY	POP	3C
0229	0176	0000	DATA DIMY	DIM	3D
0230	0178	0108'	DATA LETY	LET	3E
0231	017A	0110'	DATA PRTY	()	3F
0232	017C	0000	DATA ONY	ON	40
0233	017E	0000	DATA IFY	IF	41
0234	0180	0000	DATA DEFY	DEF	42
0235	0182	0000	DATA NEWY	NEW	43
0236	0184	00FC'	DATA STPY	END	44
0237	0186	017A'	DATA PRTY	(?)	45
0238	0188	0000	DATA EXTNDY	(*) (EXTEND)	46
0239	018A	0000	DATA BITY	BIT(47
0240	018C	0000	DATA CRBY	CRB(48
0241	018E	0000	DATA CRFY	CRF(49
0242	0190	0000	DATA MEMY	MEM(4A
0243	0192	0000	DATA MWDY	MWD(4B
0244			*		
0245	0194'		LASTCM EQU \$		
0246	004B		MAXSTA EQU \$-MXLST/2	MAX. STATEMENT CODE	
0247			*		
0248			* INTERNAL DATA		
0249			*		
0250	0194	08	NXTXB BYTE NXTB		
0251	0195	07	DATXB BYTE DATB		
0252			END		

NO ERRORS, NO WARNINGS

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.CVBD
OBJECT ACCESS NAME= ADHOC.OBJ.CVBD
LISTING ACCESS NAME= ADHOC.LST.CVBD
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

```

0002          IDT  'CVBD'
0003          *
0004          *
0005          *
0006          DEF  CVBD, CVBI, CVBFR, B30
0007          REF  FPAC, FPAC2
0008          REF  CVHD, CVHD01, CVHD15, CVHD12
0009          REF  FFLG
0010          REF  FADDI, FSUBI
0011          REF  CVGCN, B01, B05
0012          REF  CVBF
0013          REF  B2D, B2E, B31, B3A, B45
0014          *
0015          *  DEFINE FLOATING POINT XOPS FOR THIS MODULE
0016          *
0017          DXOP  LOADF, 0          ; LOAD FPAC
0018          DXOP  STORE, 1         ; STORE FPAC
0019          DXOP  FADD, 2          ; ADD TO FPAC
0020          DXOP  FSUB, 3          ; SUBTRACT FROM FPAC
0021          DXOP  FMUL, 4          ; MULTIPLY FPAC
0022          DXOP  FDIV, 5          ; DIVIDE FPAC
0023          DXOP  SCALE, 6         ; SCALE PFAC
0024          DXOP  NORMAL, 7        ; NORMALIZE FPAC
0025          DXOP  CLEAR, 8         ; CLEAR FPAC
0026          DXOP  NEGATE, 9        ; NEGATE FPAC
0027          DXOP  FLOATF, 10       ; FLOAT FPAC
0028          *
0029          DXOP  EVFIX, 11         ; EVALUATE AND FIX
0030          DXOP  OUTFP, 12         ; OUT FLOATING POINT £
0031          DXOP  OUTINT, 13        ; OUT INTEGER
0032          2F80  ERROR  EQU  >2F80 ; XOP XX, 14 (ERROR CALL)
0033          2FA0  ERROR2 EQU  ERROR+>20
  
```



```

0035 0000 2FA0 ERR34 DATA ERROR2,34 ; UNNORMALIZED £
0036 *
0037 *OUTPUT INTEGER
0038 *
0039 0004 2E00 CVBI CLEAR 0 ; CLEAR FPAC
0040 0006 C81B MOV *R11,@FPAC2
0008 0000
0041 000A 100B JMP CVBD0
0042 *
0043 *OUTPUT FLOATING POINT £
0044 *
0045 000C 020A CVBD LI R10,FPAC ; GET FPAC ADDRESS
000E 0000
0046 0010 DEBB MOV *R11+,*R10+
0047 0012 DEBB MOV *R11+,*R10+
0048 0014 DEBB MOV *R11+,*R10+
0049 0016 DEBB MOV *R11+,*R10+
0050 0018 DEBB MOV *R11+,*R10+
0051 001A D69B MOV *R11,*R10
0052 *
0053 001C 2E80 CVBD0 FLOATF 0 ; FLOAT FPAC IF NECESSARY
0054 001E 04CC CLR R12 ; CLEAR SIGN FLAG
0055 0020 0207 LI R7,CVHD ; GET HOLD ADR
0022 0000
0056 0024 7DD7 SB *R7,*R7+ ; CLEAR FIRST BYTE
0057 0026 C060 MOV @FPAC,R1 ; CHECK FOR ZERO
0028 000E'
0058 002A 1106 JLT CVBD1 ; NEGATIVE
0059 002C 1507 JGT CVBD2 ; POSITIVE
0060 002E DDE0 MOV *R30,*R7+ ; ZERO, OUT "0"
0030 007F'
0061 0032 020A LI R10,12 ; SET DIGIT COUNT
0034 000C
0062 0036 103B JMP CVBD12
0063 *
0064 0038 2E40 CVBD1 NEGATE 0 ; NEGATE FPAC
0065 003A 070C SETO R12 ; SET FLAG
0066 *
0067 003C 0241 CVBD2 ANDI R1,>00F0 ; NORMALIZED?
003E 00F0
0068 0040 13DF JEQ ERR34 ; N
0069 0042 04CA CLR R10 ; CLEAR DECIMAL ADJUST COUNTER
0070 *
0071 0044 D060 CVBD3 MOV *R1,@FPAC,R1 GET EXPONENT
0046 002B'
0072 0048 0981 SRL R1,B
0073 004A 0208 LI R8,>4A GET EXP=4A (16^10)
004C 004A
0074 004E 6201 S R1,R8
0075 0050 1310 JEQ CVBD6 EXP=4A
0076 0052 1105 JLT CVBD4 EXP<4A (MUL)
0077 *
0078 0054 06A0 BL @CVGCN FIX
0056 0000
0079 0058 A28B A R8,R10 UPDATE DECIMAL ADJUSTOR
0080 005A 2D10 FMUL *R0 MULTIPLY
0081 005C 10F3 JMP CVBD3
0082 *
0083 005E 0508 CVBD4 NEG R8 RB=-RB
0084 0060 06A0 BL @CVGCN

```

0062	0056'				
0085	0064	6288	S	R8,R10	UPDATE DECIMAL ADJUSTOR
0086	0066	2D50	FDIV	*R0	DIVIDE
0087	0068	10ED	JMP	CVBD3	
0088		*			
0089	006A	06A0	CVBD5	BL @FADDI	WENT NEG, ADD BACK
	006C	0000			
0090	006E	058A	INC	R10	
0091	0070	1007	JMP	CVBD7	
0092		*			
0093	0072	2C40	CVBD6	STORE R0	LOAD FPAC IN R0,R1,R2
0094	0074	0240		ANDI R0,>00FF	CLEAR EXPONENT
	0076	00FF			
0095	0078	0203	LI	R3,CVTB0	GET TABLE ADR
	007A	01BE'			
0096	007C	0209	LI	R9,>30	"0
	007E	0030			
0097		007F'	B30	EQU \$-1	
0098		*			
0099	0080	C133	CVBD7	MOV *R3+,R4	GET 1ST £
0100	0082	C173		MOV *R3+,R5	
0101	0084	C1B3		MOV *R3+,R6	
0102	0086	06A0		BL @FSUBI	R0,R1,R2=R0,R1,R2-R4,R5,R6
	0088	0000			
0103	008A	11EF		JLT CVBD5	WENT NEG, ADD BACK
0104		*			
0105	008C	0589	CVBD8	INC R9	COUNT
0106	008E	06A0		BL @FSUBI	/R0,R1,R2=R0,R1,R2-R4,R5,R6
	0090	008B'			
0107	0092	13FC		JEQ CVBD8	
0108	0094	15FB		JGT CVBD8	
0109	0096	06A0		BL @FADDI	WENT NEGATIVE, ADD BACK
	0098	006C'			
0110	009A	06C9		SWPB R9	READY BYTE FOR STORING
0111	009C	DDC9		MOVB R9,*R7+	MOVE INTO BUFFER
0112	009E	0209		LI R9,>2F	RELOAD R9
	00A0	002F			
0113	00A2	C133		MOV *R3+,R4	GET NEXT CONSTANT
0114	00A4	C173		MOV *R3+,R5	
0115	00A6	C1B3		MOV *R3+,R6	
0116	00A8	0283		CI R3,CVTBOE	DONE?
	00AA	020C'			
0117	00AC	12EF		JLE CVBD8	N

```

0119      *CONVERSION DONE, ROUND AND SUPPRESS TRAILING ZERO'S
0120      *
0121      *          R10 = F = .001
0122      *          E = .01
0123      *          D = .1
0124      *          C = 1
0125      *          B = 10
0126      *          A = 100
0127      *
0128 00AE 7DD7 CVBD12 SB  *R7,*R7+      OUT NULL
0129 00B0 02B7      CI  R7,CVHD15      DONE?
      00B2 0000
0130 00B4 1AFC      JL  CVBD12      N
0131      *
0132 00B6 C1ED      MOV  @14(13),R7      GET OUTPUT BUFFER PTR
      00B8 000E
0133 00BA 0200      LI  * R0,10      GET ROUNDING DIGIT COUNT
      00BC 000A
0134 00BE C260      MOV  @FFLG,R9      FORMATTING?
      00C0 0000
0135 00C2 160A      JNE  CVBFP      Y
0136 00C4 06A0      BL   @CVBFR      N, ROUND
      00C6 0170
0137 00C8 060A      DEC  R10      NEW DIGIT
0138 00CA 04C4      CLR  R4      CLEAR TRAILING ZEROES
0139      *
0140 00CC 9560 CVBD16 CB  @B30,*R5      "0?"
      00CE 007F
0141 00D0 1605      JNE  CVBD17      N
0142 00D2 D544      MOVB R4,*R5      Y, MAKE NULL
0143 00D4 0605      DEC  R5      BACKUP 1 DIGIT
0144 00D6 10FA      JMP  CVBD16
0145      *
0146 00D8 0460 CVBFP  B    @CVBF      DO FORMATTING
      00DA 0000
0147      *
0148      *SET SIGN AND GET INITIAL PTRS
0149      *
0150 00DC C30C CVBD17 MOV  R12,R12      NEGATIVE?
0151 00DE 1302      JEQ  $+6      N
0152 00E0 DDE0      MOVB @B2D,*R7+      Y, OUT "-"
      00E2 0000

```

0154		*PROCESS FORMAT FREE £	
0155		*	
0156	00E4 064A	DECT R10	>E11?
0157	00E6 1127	JLT CVBD27	Y
0158	00EB 028A	CI R10,16	N, <E05?
	00EA 0010		
0159	00EC 1524	JGT CVBD27	Y
0160	00EF 022A	AI R10,-11	N, "0.?
	00F0 FFF5		
0161	00F2 1510	JGT CVBD22	Y
0162	00F4 130F	JEG CVBD22	Y
0163		*	
0164	00F6 DDF3	CVBD19 MOVB *R3+,*R7+	N, MOVE NUMBER
0165	00F8 1307	JEG CVBD20	DONE
0166	00FA 058A	INC R10	TIME FOR ".?
0167	00FC 11FC	JLT CVBD19	N
0168	00FE D013	MOVB *R3,R0	Y, ". NEEDED?
0169	0100 1315	JEG CVBD25	N, DONE
0170	0102 DDE0	MOVB @B2E,*R7+	Y, OUT "
	0104 0000		
0171	0106 100F	JMP CVBD24	
0172		*	
0173	0108 0607	CVBD20 DEC R7	
0174		*	
0175	010A DDE0	MOVB @B30,*R7+	OUT "0 UNTIL R10=0
	010C 007F'		
0176	010E 058A	INC R10	DONE?
0177	0110 11FC	JLT #-6	N
0178	0112 100C	JMP CVBD25	Y
0179		*	
0180	0114 DDE0	CVBD22 MOVB @B30,*R7+	OUT "0
	0116 007F'		
0181	0118 DDE0	MOVB @B2E,*R7+	OUT "
	011A 0104'		
0182		*	
0183	011C 060A	CVBD23 DEC R10	TIME FOR DIGIT?
0184	011E 1103	JLT CVBD24	Y
0185	0120 DDE0	MOVB @B30,*R7+	N, OUT "0
	0122 007F'		
0186	0124 10FB	JMP CVBD23	
0187		*	
0188	0126 DDF3	CVBD24 MOVB *R3+,*R7+	MOVE REST OF STRING
0189	0128 16FE	JNE #-2	
0190	012A 0607	DEC R7	BACKUP OVER NULL
0191		*	
0192	012C CB47	CVBD25 MOV R7,@14(13)	RETURN UPDATE PTR
	012E 000E		
0193	0130 0380	RTWP DONE	
0194		*	
0195	0132 C1C6	CVBD26 MOV R6,R7	FORMATTING OVERFLOW
0196	0134 10FB	JMP CVBD25	

```

0198          *EXPONENTIAL FORM
0199          *
0200 0136 DDF3 CVBD27 MOVB *R3+,*R7+      OUT FIRST DIGIT
0201 0138 D113          MOVB *R3,R4      ANOTHER DIGIT?
0202 013A 1302          JEG $+6          N, NO ".
0203 013C DDE0          MOVB @B2E,*R7+    Y, ".
        013E 011A'
0204          *
0205 0140 DDF3          MOVB *R3+,*R7+    MOVE £
0206 0142 16FE          JNE $-2
0207 0144 0607          DEC R7
0208 0146 DDE0          MOVB @B45,*R7+    "E
        0148 0000
0209 014A 022A          AI R10,-10
        014C FFF6
0210 014E 050A          NEG R10
0211 0150 1503          JGT CVBD28        POSITIVE
0212 0152 DDE0          MOVB @B2D,*R7+    -, OUT "-
        0154 00E2'
0213 0156 050A          NEG R10
0214          *
0215 0158 04C9 CVBD28 CLR R9              ; CLEAR UPPER PART
0216 015A 3E60          DIV @C000A,R9     ; R9,R10/10
        015C 0204'
0217 015E 0229          AI R9,>30         ; ADD BITS
        0160 0030
0218 0162 022A          AI R10,>30
        0164 0030
0219 0166 06C9          SWPB R9           ; POSITION
0220 0168 06CA          SWPB R10
0221 016A DDC9          MOVB R9,*R7+      ; MOVE INTO STREAM
0222 016C DDCA          MOVB R10,*R7+
0223 016E 10DE          JMP CVBD25        ; RETURN

```

```

0225      *
0226      *ROUND CHARACTER STRING TO 10 TH POSITION
0227      *
0228 0170 0206  CVBFR  LI   R6,CVHD01      GET STRING ADR
          0172 0000
0229 0174 D820      MOVB @B30,@CVHD12      FORCE 12TH DIGIT TO '0'
          0176 007F'
          0178 0000
0230 017A A180      A      R0,R6          INDEX TO ROUNDING POSITION
0231 017C D000      MOVB R0,R0          ABLE TO ROUND?
0232 017E 1618      JNE  CVBFR3          N
0233 0180 C146      MOV  R6,R5          Y, MARK
0234 0182 0586      INC  R6
0235      *
0236 0184 D0D6      MOVB *R6,R3          GET ROUNDING DIGIT
0237 0186 1314      JEQ  CVBFR3          DONE
0238 0188 D580      MOVB R0,*R6          SET TO NULL
0239 018A B0E0      AB   @B05,R3          ADD 5
          018C 0000
0240 018E 90E0      CB   @B3A,R3          CARRY?
          0190 0000
0241 0192 1B0E      JH   CVBFR3          N, DONE
0242      *
0243 0194 0606  CVBFR1 DEC  R6          Y, HANDLE CARRY
0244 0196 D016      MOVB *R6,R0          EDS?
0245 0198 1308      JEQ  CVBFR2          Y, INSERT NEW DIGIT
0246 019A B5A0      AB   @B01,*R6        N, ADD CARRY
          019C 0000
0247 019E 95A0      CB   @B3A,*R6        CARRY?
          01A0 0190'
0248 01A2 1B06      JH   CVBFR3          N, DONE
0249 01A4 D5A0      MOVB @B30,*R6        Y, INSERT "0"
          01A6 007F'
0250 01A8 10F5      JMP  CVBFR1          CONTINUE
0251      *
0252 01AA 049B  CVBFR2 X    *R11          NEW DIGIT, ADJUST R10
0253 01AC D5A0      MOVB @B31,*R6        INSERT "1"
          01AE 0000
0254      *
0255 01B0 0203  CVBFR3 LI   R3,CVHD      GET INITIAL PTR
          01B2 0022'
0256 01B4 D1B3      MOVB *R3+,R6          NULL?
0257 01B6 1301      JEQ  CVBFR4          Y
0258 01B8 0603      DEC  R3              N, MOVE BACK
0259 01BA 046B  CVBFR4 B    @2(11)        RETURN
          01BC 0002
0260      *
0261      *
0262 01BE 00E8  CVTBO  DATA >00E8,>D4A5,>1000
0263 01C4 0017      DATA >0017,>4876,>E800
0264 01CA 0002      DATA >0002,>540B,>E400
0265 01D0 0000      DATA >0000,>3B9A,>CA00
0266 01D6 0000      DATA >0000,>05F5,>E100
0267 01DC 0000      DATA >0000,>0098,>9680
0268 01E2 0000      DATA >0000,>000F,>4240
0269 01E8 0000      DATA >0000,>0001,>86A0
0270 01EE 0000      DATA >0000,>0000,>2710
0271 01F4 0000      DATA >0000,>0000,>03EB
0272 01FA 0000      DATA >0000,>0000,>0064
0273      0204' C000A  EQU  $+4

```

0274 0200 0000 DATA >0000,>0000,>000A

0275 0206 0000 DATA >0000,>0000,>0001

0276 020C' CVTBOE EQU \$

0277 END

NO ERRORS, NO WARNINGS

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.CVDB
OBJECT ACCESS NAME= ADHOC.OBJ.CVDB
LISTING ACCESS NAME= ADHOC.LST.CVDB
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=


```

0002          IDT  'CVDB'
0003          *
0004          *      CVDIZ  ; CONVERT DECIMAL TO BINARY INTEGER
0005          *      CVDIFZ ; CONVERT DECIMAL TO FLOATING POINT
0006          *      CVDB20 ; GET NEXT DIGIT
0007          *
0008          REF  CVGCN1          ; GET FLOATING POINT CONSTANT
0009          REF  CVCH            ; CONVERSION HOLDING AREA
0010          REF  WPR2            ; SECONDARY WORKSPACE
0011          REF  CVC10          ; FLOATING POINT CONSTANTS
0012          REF  FPAC            ; FLOATING POINT ACCUMULATOR
0013          REF  B20             ; SPACE CHARACTER
0014          REF  C000A
0015          *
0016          DEF  CVDIZ,CVDIFZ    ENTRY POINTS
0017          DEF  CVDB20
0018          *
0019          DXOP LOADF,0          ; LOAD FPAC
0020          DXOP STORE,1          ; STORE FPAC
0021          DXOP FADD,2           ; ADD TO FPAC
0022          DXOP FSUB,3           ; SUBTRACT FROM FPAC
0023          DXOP FMUL,4           ; MULTIPLY FPAC
0024          DXOP FDIV,5           ; DIVIDE FPAC
0025          DXOP SCALE,6          ; SCALE FPAC
0026          DXOP NORMAL,7         ; NORMALIZE FPAC
0027          DXOP CLEAR,8          ; CLEAR FPAC
0028          DXOP NEGATE,9         ; NEGATE FPAC
0029          DXOP FLOATF,10        ; FLOAT FPAC
0030          2F80  ERROR  EQU  >2F80          ; XOP XX,14 (ERROR CALL)
0031          2FA0  ERROR2 EQU  ERROR+>20

```

```
0033      *      CVDIZ AND CVDIFZ CONVERT A STRING OF
0034      *      ASCII DECIMAL DIGITS TO A BINARY NUMBER.
0035      *      CVDIZ IS CALLED IF ONLY AN INTEGER WILL
0036      *      BE ACCEPTED. CVDIFZ IS CALLED IF A
0037      *      FLOATING POINT NUMBER CAN BE ALLOWED.
0038      *      CVDI IS FIRST EXECUTED TO GET AN
0039      *      INTEGER. IF IT FAILS, THEN A RETURN
0040      *      IS MADE OR CVDIFZ IS EXECUTED DEPENDING
0041      *      ON WHICH ROUTINE WAS INITIALLY CALLED.
0042      *
0043      *
0044      * CALLING SEQUENCE:
0045      *
0046      *      BLWP @CVDIZ
0047      *      OR
0048      *      BLWP @CVDIFZ
0049      *
0050      *      IN - R7 =PTR
0051      *
0052      *      OUT -   R0 = DELIMITER
0053      *              R1 = 16-BIT 2'S COMPLEMENT INTEGER
0054      *              FPAC = FLOATING POINT NUMBER
0055      *              R7 = NEW PTR
0056      *
0057      *      NORMAL EXIT - RTWP
0058      *      ERROR EXIT
```

```

0060      * CONVERT DECIMAL TO INTEGER
0061      *      BLWP @CVDIZ
0062      *      16-BIT OVERFLOW - FP & IN FPAC
0063      *      NO NUMBER
0064      *      (HEX ON CVDIF)
0065      *      NUMBER
0066      *
0067      * IN   R7 = PTR
0068      * OUT  R0 = DELIMITER
0069      *      R1 = 16-BIT 2'S COMPLEMENT INTEGER
0070      *      R7 = NEW PTR
0071      *
0072 0000 0000 CVDIZ DATA WPR2,CVDI
0073 0004 0000' CVDIFZ DATA WPR2,CVDIF
0074      *
0075 0008 04CB CVDI   CLR  R8              GET INTEGER ONLY
0076 000A 1001      JMP  $+4
0077 000C 0708 CVDIF  SETD R8              ALLOW EXPONENT
0078      *
0079 000E C1ED      MOV  @14(13),R7        GET PTR
0080      0010 000E
0081 0012 04C2      CLR  R2              CLEAR RESULT
0082 0014 04C4      CLR  R4              CLEAR SIGN FLAG
0083 0016 06A0      BL   @CVDB25         LOOK FOR SIGN
0084      0018 0194'
0085 001A 103A      JMP  CVDI4            NO NUMBER, RETURN
0086 001C 0704      SETD R4              NEGATIVE
0087 001E C187      MOV  R7,R6
0088      *
0089 0020 D036 CVDIH1 MOVB *R6+,R0
0090      0022 0980      SRL  R0,8        ; POSITION
0091 0024 0220      AI   R0,->30
0092      0026 FFD0
0093 0028 1114      JLT  CVDIH4            ; NOT HEX
0094 002A 0280      CI   R0,>09
0095      002C 0009
0096 002E 1208      JLE  CVDIH2            ; HEX 0-9
0097 0030 0220      AI   R0,->07
0098      0032 FFF9
0099 0034 0280      CI   R0,>0A
0100      0036 000A
0101 0038 110C      JLT  CVDIH4            ; NOT HEX
0102 003A 0280      CI   R0,>0F
0103      003C 000F
0104 003E 1B03      JH   CVDIH3            ; NOT HEX, "H MAYBE?"
0105      *
0106 0040 0A42 CVDIH2 SLA  R2,4            ; ADD NEW HEX DIGIT
0107 0042 A080      A    R0,R2
0108 0044 10ED      JMP  CVDIH1
0109      *
0110 0046 0220 CVDIH3 AI   R0,->11        ; "H?"
0111      0048 FFEF
0112 004A 1603      JNE  CVDIH4            ; N
0113 004C D036      MOVB *R6+,R0          ; Y, GET DELIMITER
0114 004E C1C6      MOV  R6,R7            ; UPDATE R7
0115 0050 1017      JMP  CVDI3A
0116      *
0117 0052 04C2 CVDIH4 CLR  R2            ; NOT HEX, CLEAR R2
0118      *
0119 0054 06A0 CVDI1 BL   @CVDB20        GET DIGIT

```

```

0056 0174'
0112 0058 100A      JMP  CVDI2      N
0113 005A C042      MOV  R2,R1      SET FOR MULTIPLICATION
0114 005C 3860      MPY  @C000A,R1  R1,R2=R1*10
      005E 0000
0115 0060 C041      MOV  R1,R1      OVERFLOW?
0116 0062 161B      JNE  CVDB      Y
0117 0064 C082      MOV  R2,R2      OVERFLOW?
0118 0066 1119      JLT  CVDB      ;Y
0119 0068 A080      A     R0,R2      ADD NEW DIGIT, OVERFLOW?
0120 006A 1117      JLT  CVDB      Y, OVERFLOW
0121 006C 10F3      JMP  CVDI1      LOOP
0122
      *
0123 006E 0280      CVDI2 CI  R0,>2E00  ". ?
      0070 2E00
0124 0072 1311      JEG  CVDI5      Y
0125 0074 0280      CI   R0,>4500  "E?
      0076 4500
0126 0078 130E      JEG  CVDI5      Y - SEE IF LEGAL
0127
      *
0128 007A C208      CVDI3 MOV  R8,R8      ;SKIP HEX RETURN?
0129 007C 1301      JEG  CVDI3A     ;Y
0130 007E 05CE      INCT R14      ;N
0131
      *
0132 0080 C104      CVDI3A MOV  R4,R4      N, NEGATIVE?
0133 0082 1301      JEG  $+4      N
0134 0084 0502      NEG  R2      Y, NEGATE
0135 0086 CB42      MOV  R2,@2(13)  RETURN NUMBER IN R1
      0088 0002
0136 008A CB47      MOV  R7,@14(13)  RETURN PTR IN R7
      008C 000E
0137 008E 05CE      INCT R14      RETURN 4(14)
0138 0090 05CE      CVDI4 INCT R14      RETURN 2(14) - NO NUMBER
0139 0092 C740      MOV  R0,*R13     RETURN DELIMITER IN R0
0140 0094 0380      RTWP RETURN      0(14) - OVERFLOW
0141
      *
0142 0096 C208      CVDI5 MOV  R8,R8      EXPONENT ALLOWED?
0143 0098 13F3      JEG  CVDI3A     N
0144
      *
0145      * FALL THRU TO CVDB

```

0147		* CONVERT DECIMAL TO FP	
0148		* BLWP @CVDBZ	
0149		* NUMBER	
0150		* NO NUMBER	
0151		*	
0152		* IN R7 = STRING ADDRESS	
0153		* OUT R0 = DELIMITER	
0154		* R7 = NEW PTR	
0155		* FPAC = NUMBER	
0156		*	
0157	009A 2E00	CVDB CLEAR 0	CLEAR FPAC
0158	009C C1ED	MOV @14(13),R7	GET STRING ADR
	009E 000E		
0159	00A0 0702	SET0 R2	SET DECIMAL FLAG
0160	00A2 04C6	CLR R6	DECIMAL ADJUSTMENT=0
0161		*	
0162	00A4 04CC	CLR R12	CLEAR SIGN FLAG
0163	00A6 04C4	CLR R4	CLEAR SIGNIFICANT DIGITS COUNT
0164	00AB 06A0	BL @CVDB25	PROCESS SIGN
	00AA 0194		
0165	00AC 10F1	JMP CVDI4	NO NUMBER
0166	00AE 070C	SET0 R12	NEG, SET TO -1
0167		*	
0168	00B0 06A0	CVDB1 BL @CVDB20	GET CHARACTER
	00B2 0174		
0169	00B4 1017	JMP CVDB4	NOT NUMBER
0170	00B6 C0B2	MOV R2,R2	AFTER DECIMAL?
0171	00BB 1101	JLT #+4	N
0172	00BA 05B6	INC R6	Y, COUNT
0173	00BC C000	MOV R0,R0	'0'?
0174	00BE 1602	JNE CVDB1A	
0175	00C0 C104	MOV R4,R4	Y - LEADING '0'?
0176	00C2 13F6	JEG CVDB1	
0177	00C4 05B4	CVDB1A INC R4	Y - INCREMENT SIG DIGITS COUNT
0178	00C6 02B4	CI R4,11	11 SIG DIGITS?
	00CB 000B		
0179	00CA 1201	JLE CVDB1C	
0180	00CC 04C0	CLR R0	Y - SET DIGIT TO 0
0181	00CE 2D20	CVDB1C FMUL @CVC10	;OK, FPAC=FPAC*10
	00D0 0000		
0182	00D2 0A40	SLA R0,4	00X0
0183	00D4 13ED	JEG CVDB1	IGNOR IF ZERO
0184	00D6 0220	AI R0,>4100	41X0 (FP & NOW)
	00DB 4100		
0185	00DA CB00	MOV R0,@CVCH	SAVE
	00DC 0000		
0186	00DE 2CA0	FADD @CVCH	ADD NEW DIGIT
	00E0 00DC		
0187	00E2 10E6	JMP CVDB1	
0188		*	
0189	00E4 02B0	CVDB4 CI R0,>2E00	". ?
	00E6 2E00		
0190	00EB 1606	JNE CVDB5	N
0191	00EA C0B2	MOV R2,R2	Y - FIRST PERIOD?
0192	00EC 1319	JEG CVDB10	N - END CONVERSION
0193	00EE C208	MOV R8,R8	EXPONENT ALLOWED?
0194	00F0 1317	JEG CVDB10	N - FINISH
0195	00F2 04C2	CLR R2	Y, SET FLAG
0196	00F4 10DD	JMP CVDB1	
0197		*	

0198	00F6	0280	CVDB5	CI	R0, >4500	"E?"
	00F8	4500				
0199	00FA	1612		JNE	CVDB10	N
0200	00FC	0205		LI	R5, >0501	Y, (NEG R1)
	00FE	0501				
0201	0100	06A0		BL	@CVDB25	CHECK FOR SIGN
	0102	0194				
0202	0104	10C5		JMP	CVDB14	PROBLEM, RETURN NO NUMBER
0203	0106	0AC5		SLA	R5, 12	NEG, LOAD NOOP (1000)
0204	0108	04C1		CLR	R1	
0205			*			
0206	010A	06A0	CVDB6	BL	@CVDB20	GET DIGIT
	010C	0174				
0207	010E	1006		JMP	CVDB7	NO DIGIT, DONE
0208	0110	C0C1		MOV	R1, R3	
0209	0112	3BE0		MPY	@C000A, R3	X 10
	0114	005E				
0210	0116	A100		A	R0, R4	ADD NEW DIGIT
0211	0118	C044		MOV	R4, R1	RESTORE R1
0212	011A	10F7		JMP	CVDB6	LOOP AGAIN
0213			*			
0214	011C	0485	CVDB7	X	R5	DO EXPONENT CHANGE
0215	011E	A181		A	R1, R6	ADD EXPONENT ADJUSTOR
0216			*			
0217	0120	C740	CVDB10	MOV	R0, *R13	DONE, RETURN DELIMITER
0218	0122	CB47		MOV	R7, @14(13)	RETURN NEW PTR
	0124	000E				
0219			*			
0220	0126	C206	CVDB11	MOV	R6, R8	CHECK ADJUSTMENT
0221	0128	1106		JLT	CVDB12	DO ADJUSTMENT NECESSARY
0222	012A	130B		JEG	CVDB13	" " "
0223	012C	06A0		BL	@CVGCN1	GET CONSTANT
	012E	0000				
0224	0130	618B		S	R8, R6	
0225	0132	2D50		FDIV	*R0	DIVIDE BY 10^R6
0226	0134	10F8		JMP	CVDB11	
0227			*			
0228	0136	0508	CVDB12	NEG	R8	
0229	0138	06A0		BL	@CVGCN1	GET CONSTANT
	013A	012E				
0230	013C	A18B		A	R8, R6	
0231	013E	2D10		FMUL	*R0	MULTIPLY BY 10^R6
0232	0140	10F2		JMP	CVDB11	
0233			*			
0234	0142	C30C	CVDB13	MOV	R12, R12	CHECK FOR NEGATIVE &
0235	0144	1312		JEG	CVDB14	POSITIVE
0236	0146	2E40		NEGATE	0	
0237			*			
0238			*			
0239			*			
0240			*			
0241			*			
0242	0148	020C		LI	R12, FPAC	REF FPAC
	014A	0000				
0243	014C	C0CC		MOV	R12, R3	
0244	014E	0206		LI	R6, CC480	REF -32768 (IN FL PT FORMAT)
	0150	016C				
0245	0152	8DBC		C	*R12+, *R6+	1ST WORD SAME?
0246	0154	160A		JNE	CVDB14	
0247	0156	8DBC		C	*R12+, *R6+	Y - 2ND WORD SAME?

0248 0158 1608	JNE	CVDB14	
0249 015A 8D9C	C	*R12, *R6+	Y - 3RD WORD SAME?
0250 015C 1606	JNE	CVDB14	
0251 015E 04F3	CLR	*R3+	Y - CLEAR 1ST WORD
0252 0160 CCD6	MOV	*R6, *R3+	2ND WORD TO >8000
0253 0162 C096	MOV	*R6, R2	R2=RESULT REGISTER FOR CVDI
0254 0164 04D3	CLR	*R3	CLEAR 3RD WORD
0255 0166 04C4	CLR	R4	CLEAR NEGATE FLAG
0256 0168 1088	JMP	CVDI3	
0257 016A 0380	CVDB14	RTWP	
0258	*		
0259 016C C480	CC480	DATA >C480, 0, 0	
0260 0172 8000		DATA >8000	

```

0262      * CVDB20: - GETS THE NEXT DIGIT IF THERE
0263      *           IS ONE. LEADING BLANKS ARE SKIPPED.
0264      *           IF THE NEXT CHARACTER IS A NON-DIGIT
0265      *           THEN RETURN IS MADE TO THE FIRST
0266      *           INSTRUCTION FOLLOWING THE CALL. IF
0267      *           THE NEXT CHARACTER IS A DIGIT, A
0268      *           RETURN IS TAKEN TO THE SECOND
0269      *           INSTRUCTION FOLLOWING THE CALL.
0270      *
0271      * CALLING SEQUENCE:
0272      *
0273      *           BL @CVDB20
0274      *
0275      *           IN - R7 = POINTER
0276      *           OUT - R0 = CHARACTER
0277      *
0278      *           NORMAL EXIT - RETURN + 1 ON DIGIT
0279      *                           RETURN ON NON-DIGIT
0280      *
0281 0174 04C0 CVDB20 CLR R0           GET DIGIT
0282 0176 D037      MOVB #R7+,R0      GET NEXT CHARACTER
0283 0178 130C      JEQ CVDB22        EOL
0284 017A 0280      CI R0,>2000       BLANK?
0285      017C 2000
0286 017E 13FA      JEQ CVDB20        Y
0287 0180 0280      CI R0,>3000       <"0?
0288      0182 3000
0289 0184 1A06      JL CVDB22        Y
0290 0186 0280      CI R0,>3900       >"9?
0291      0188 3900
0292 018A 1B03      JH CVDB22        Y
0293 018C 0A40      SLA R0,4
0294 018E 09C0      SRL R0,12        RIGHT JUSTIFY CHARACTER
0295 0190 05CB      INCT R11
0296 0192 045B CVDB22 RT RETURN

```



```

0295      * PROCESS SIGN
0296      *      BL @CVDB25
0297      *      NO NUMBER
0298      *      -NUMBER
0299      *      NUMBER OR +NUMBER
0300      *
0301 0194 9837 CVDB25 CB *R7+, @B20 SPACE?
      0196 0000
0302 0198 13FD JEG CVDB25 ?
0303 019A 0607 DEC R7 BACKUP
0304 019C 0047 MOV R7, R1 MARK
0305 019E 028B MOV R11, R10
0306 01A0 06A0 BL @CVDB20 GET NUMBER
      01A2 0174'
0307 01A4 1004 JMP CVDB28 NO NUMBER
0308 01A6 05CA CVDB26 INCT R10 NUMBER, RETURN 4(10)
0309 01A8 05CA INCT R10 -NUMBER, RETURN 2(10)
0310 01AA 01C1 CVDB27 MOV R1, R7 RESTORE R7
0311 01AC 045A B *R10 RETURN
0312      *
0313 01AE 0280 CVDB28 CI R0, >2B00 "+?"
      01B0 2B00
0314 01B2 1307 JEG CVDB29 Y
0315 01B4 0280 CI R0, >2D00 N, "-?"
      01B6 2D00
0316 01B8 1307 JEG CVDB30 Y
0317 01BA 0280 CI R0, >2E00 ".?"
      01BC 2E00
0318 01BE 16F5 JNE CVDB27 N, NO NUMBER
0319 01C0 0607 DEC R7 Y, BACKUP OVER PERIOD
0320      *
0321 01C2 06A0 CVDB29 BL @CVDB31 PROCESS POSITIVE NUMBER
      01C4 01CE'
0322 01C6 10EF JMP CVDB26 OK
0323      *
0324 01C8 06A0 CVDB30 BL @CVDB31 PROCESS NEGATIVE NUMBER
      01CA 01CE'
0325 01CC 10ED JMP CVDB26+2 OK
0326      *
0327 01CE 00CB CVDB31 MOV R11, R3 SAVE RETURN
0328 01D0 0047 MOV R7, R1 MARK
0329 01D2 06A0 BL @CVDB20 DIGIT?
      01D4 0174'
0330 01D6 1001 JMP CVDB32 N, LOOK FOR PERIOD
0331 01D8 0453 B *R3 Y, NUMBER
0332      *
0333 01DA 0280 CVDB32 CI R0, >2E00 ".?"
      01DC 2E00
0334 01DE 16E5 JNE CVDB27 N, NO NUMBER
0335 01E0 06A0 BL @CVDB20 Y, LOOK FOR DIGIT
      01E2 0174'
0336 01E4 10E2 JMP CVDB27 NO NUMBER
0337 01E6 0453 B *R3 NUMBER OK
0338      *
0339      END
NO ERRORS, NO WARNINGS

```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.CVGC
OBJECT ACCESS NAME= ADHOC.OBJ.CVGC
LISTING ACCESS NAME= ADHOC.LST.CVGC
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

```
0002          IDT 'CVGC'
0003          *
0004          *      CVGCN          ; GET CONSTANT
0005          *
0006          DEF CVC10          ; FLOATING POINT CONSTANT 10.
0007          DEF CVGCN, CVGCN1
0008          *
0009          *
0010          *      CVGCN WILL GET THE POWER OF TEN CLOSEST
0011          *      TO THE POWER OF 16 IN R8.  IE:
0012          *
0013          *          16^1  >  10^1
0014          *          16^2  >  10^2
0015          *          16^3  >  10^3
0016          *          16^4  >  10^4
0017          *          16^5  >  10^6
0018          *          16^6  >  10^7
0019          *          .....
0020          *
0021          *  CALLING SEQUENCE:
0022          *
0023          *      BL @CVGCN
0024          *
0025          *      IN      R8 = POWER OF 16
0026          *      OUT (R0) = FP CONSTANT
```

```

0028 0000 0288  CVGCN  CI  RB,5
      0002 0005
0029 0004 1A06          JL  CVGCN2
0030 0006 0588          INC RB                      EXP CHANGE >=5
0031 0008 0288  CVGCN1 CI  RB,10
      000A 000A
0032 000C 1A02          JL  CVGCN2
0033 000E 0208          LI  RB,9                      EXP CHANGE >9, USE 9
      0010 0009
0034          *
0035 0012 04BB  CVGCN2 X   *R11+                      ADJUST DECIMAL COUNTER
0036 0014 0200          LI  R0,CVTB1                    GET 10'S TABLE ADR
      0016 0022'
0037 0018 0A18          SLA  RB,1                      MAKE WORD INDEX
0038 001A A008          A    RB,R0                      INDEX
0039 001C 0A18          SLA  RB,1                      RB X 2
0040 001E A008          A    RB,R0                      R3=R3+3*RB
0041 0020 045B          B    *R11                      RETURN
0042          *
0043 0022 4110  CVTB1  DATA >4110,>0000,>0000 10^0
0044 002B 41A0  CVC10  DATA >41A0,>0000,>0000 10^1
0045 002E 4264          DATA >4264,>0000,>0000 10^2
0046 0034 433E          DATA >433E,>8000,>0000 10^3
0047 003A 4427          DATA >4427,>1000,>0000 10^4
0048 0040 4518          DATA >4518,>6A00,>0000 10^5
0049 0046 45F4          DATA >45F4,>2400,>0000 10^6
0050 004C 469B          DATA >469B,>9680,>0000 10^7
0051 0052 475F          DATA >475F,>5E10,>0000 10^8
0052 0058 483B          DATA >483B,>9ACA,>0000 10^9
0053          END
NO ERRORS,      NO WARNINGS

```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.EVAL
OBJECT ACCESS NAME= ADHOC.OBJ.EVAL
LISTING ACCESS NAME= ADHOC.LST.EVAL
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

```

0002          IDT 'EVAL'
0003          *
0004          *      EVARZ          ;EVALUATE VARIABLE
0005          *      EVERZ          ;EVALUATE EXPRESSION
0006          *      EVSDZ          ;EVALUATE STRING
0007          *      EVFX          ;EVALUATE AND FIX
0008          *      CKEX          ;CHECK FOR EXPRESSION
0009          *      EVAL          ;RECURSIZE EVALUATOR
0010          *      EVALS2        ;2ND ENTRY IN RECURSIVE EVALUATOR
0011          *      ADDF          ;ADD TWO VARIABLES
0012          *      SUBF          ;SUBTRACT TWO VARIABLES
0013          *
0014          DXOP LOADF,0          ;LOAD FPAC
0015          DXOP STORE,1         ;STORE FPAC
0016          DXOP FADD,2          ;ADD TO FPAC
0017          DXOP FSUB,3          ;SUBTRACT FROM FPAC
0018          DXOP FMUL,4          ;MULTIPLY FPAC
0019          DXOP FDIV,5          ;DIVIDE FPAC
0020          DXOP SCALE,6         ;SCALE FPAC
0021          DXOP NORMAL,7        ;NORMALIZE FPAC
0022          DXOP CLEAR,8         ;CLEAR FPAC
0023          DXOP NEGATE,9        ;NEGATE FPAC
0024          DXOP FLOATF,10       ;FLOAT FPAC
0025          DXOP EVFIX,11        ;EVALUATE AND FIX
0026          DXOP OUTFP,12        ;OUT FLOATING POINT &
0027          DXOP OUTINT,13       ;OUT INTEGER
0028          2F80 ERROR EQU >2F80 ;XOP XX,14 (ERROR CALL)
0029          2FA0 ERROR2 EQU ERROR+>20
0030          *
0031          REF ABSF              ;ABSOLUTE VALUE FUNCTION
0032          REF ASCF              ;CONVERT ASCII FUNCTION
0033          REF ATNF              ;ARC-TANGENT FUNCTION
0034          REF COSF              ;COSINE FUNCTION
0035          REF EXPF              ;EXPONENTIAL FUNCTION
0036          REF FRAF              ;FRACTIONAL PART FUNCTION
0037          REF INTF              ;INTEGER PART FUNCTION
0038          REF LOGF              ;LOG FUNCTION
0039          REF NKYF              ;KEY FUNCTION
0040          REF SIN F              ;SINE FUNCTION
0041          REF SGRF              ;SQUARE ROOT FUNCTION
0042          REF SYSF              ;SYSTEM FUNCTION
0043          REF TICF              ;TIC FUNCTION
0044          REF SGNF              ;SIGN FUNCTION
0045          REF BITF              ;BIT FUNCTION
0046          REF CRBF              ;CRB FUNCTION
0047          REF CRFF              ;CRF FUNCTION
0048          REF MEMF              ;MEM FUNCTION
0049          REF MWDF              ;MEMORY WORD FUNCTION
0050          REF LENF              ;STRING LENGTH FUNCTION
0051          REF MCHF              ;STRING MATCH FUNCTION
0052          REF POSF              ;STRING SEARCH FUNCTION
0053          REF COLF              ;PIXEL COLOUR TEST
0054          REF MODF              ;MOD FUNCTION
0055          REF POWF              ;POWER FUNCTION
0056          REF LORF              ;LOGICAL OR FUNCTION
0057          REF LXORF             ;LOGICAL EXCLUSIVE OR FUNCTION
0058          REF LANDF             ;LOGICAL AND FUNCTION
0059          REF ANDF              ;AND FUNCTION
0060          REF ORF               ;OR FUNCTION
0061          REF NOTF             ;NOT FUNCTION

```

0062	REF	LNOTF	; LOGICAL NOT FUNCTION
0063	REF	FIX	; FIX ROUTINE
0064	REF	FLOAT	; FLOAT ROUTINE
0065	REF	RANDZ	; GET RANDOM NUMBER ROUTINE
0066	REF	ADRF	; RETURN ADDRESS OF VARIABLE FUC
0067	REF	EVSKB, EVSKE	; EVALUATION STACK
0068	REF	FPAC, FPAC2	; FLOATING POINT ACCUMULATOR
0069	REF	TEMP	; TEMP FLOATING POINT REGISTER
0070	REF	E\$TEMP, TEMP2, TEMP4	
0071	REF	FUZZ	; FUZZ VALUE
0072	REF	UFT	; USER-FUNCTION-TABLE INDEX
0073	REF	VDT	; VARIABLE-DEFINITION-TABLE INDE
0074	REF	VNT	; NVARIABLE-NAME-TABLE INDEX
0075	REF	IOB	; I/O-BUFFER INDEX
0076	REF	DLIM	; DELIMITER INDEX
0077	REF	NVD	; NEXT-VARIABLE-DEFINITION INDEX
0078	REF	NVS	; NEXT-VARIABLE-STORAGE INDEX
0079	REF	WPR2	; SECONDARY WORKSPACE
0080	REF	RTSTOR	; TEMP. STORE FOR R11
0081	*		
0082	DEF	EVSRF	; RETURN FOR FUNCTIONS
0083	DEF	EVSRF#	; RETURN WITH R2 RELOAD
0084	DEF	EVOP3A	; RETURN FOR OPERATORS
0085	DEF	B01, B05	; BYTES
0086	DEF	C1, C4, C6	; WORDS
0087	DEF	EVSDZ	
0088	DEF	EVARZ	
0089	DEF	CKEX	
0090	DEF	EVERZ	
0091	DEF	EVFX	
0092	DEF	EVAL	; FULL EXPRESSION EVALUATION
0093	DEF	EVALS2	; PARTIAL EXPRESSION EVALUATION
0094	DEF	SUBF	
0095	DEF	ADDF	

```
0097      *      EVSD EVALUATES FOR A STRING CONSTANT OR VARIABLE
0098      *      BEGINNING AT THE PBC (R8).  MULTIPLE RETURNS ARE
0099      *      USED ACCORDING TO WHAT IS FOUND.  IF NO STRING
0100      *      OR VARIABLE IS FOUND, THE PBC (PROGRAM BYTE COUNTER)
0101      *      IS LEFT UNCHANGED.
0102      *
0103      * CALLING SEQUENCE:
0104      *
0105      *      BLWP @EVSDZ
0106      *      STRING CONSTANT RETURN ("----")
0107      *      STRING VARIABLE RETURN ($VAR)
0108      *      NEITHER
0109      *
0110      *      IN  R8 = PBC (PROGRAM BYTE COUNTER)
0111      *      OUT R0 = DLIM (DELIMITER
0112      *           R2 = PTR TO STRING
0113      *           R8 = PBC UPDATED
0114      *
0115      * EXCEPTIONS AND CONDITIONS:
0116      *
0117      *      ERRORS INCLUDE EXPECTING VARIABLE, SYNTAX ERROR,
0118      *      UNDEFINED FUNCTION, ILLEGAL DELIMITER, STORAGE
0119      *      OVERFLOW, UNDEFINED VARIABLE, EXPRESSION TOO
0120      *      COMPLEX, SUBSCRIPT ERROR, TOO MANY SUBSCRIPTS,
0121      *      UNDIMENSIONED VARIABLE, AND ALL ARITHMETIC ERRORS.
```



```

0123 0000 0000 EVSDZ DATA WPR2, EVSD
0124 *
0125 0004 C22D EVSD MOV @16(13), R8 ; GET PBC
      0006 0010
0126 0008 9818 CB *R8, @B43 ; $?
      000A 0569'
0127 000C 130C JEQ EVSD2 ; Y
0128 000E 1A09 JL EVSD1 ; NOT STRING
0129 0010 9838 CB *R8+, @B45 ; " OR '?'
      0012 056A'
0130 0014 1B06 JH EVSD1 ; N
0131 0016 C088 MOV R8, R2 ; Y
0132 *
0133 0018 D038 MOVB *R8+, R0 ; LOOK FOR "0
0134 001A 16FE JNE #-2
0135 *
0136 001C 04C0 CLR R0
0137 001E D038 MOVB *R8+, R0 ; RETURN DELIMITER
0138 0020 101D JMP EVER1
0139 *
0140 0022 8FBE EVSD1 C *R14+, *R14+ ; NOT STRING, RETURN 4(14)
0141 0024 0380 RTWP
0142 *
0143 0026 05CE EVSD2 INCT R14
0144 0028 05AD INC @16(13) ; SKIP $
      002A 0010
0145 *
0146 * THIS SHOULD FALL THROUGH INTO EVAR
    
```

```
0148      *      EVAR CALLS THE EXPRESSION EVALUATOR FOR
0149      *      ONE ITEM ONLY, NAMELY A VARIABLE.  WITH
0150      *      R5 ZEROED, THE EVALUATOR WILL EVALUATE
0151      *      UP TO THE FIRST OPERATER OR DELIMITER.
0152      *      UPON RETURNING, EVAR WILL CHECK TO SEE
0153      *      IF THE RESULT IS IN THE EXPRESSION STACK
0154      *      IN WHICH CASE AN EXCEPTION WOULD OCCUR.
0155      *
0156      * CALLING SEQUENCE:
0157      *
0158      *      BLWP @EVARZ
0159      *      NORMAL RETURN
0160      *
0161      *      IN  R8 = PBC
0162      *      OUT R0 = DLIM
0163      *      R2 = ADR
0164      *      R8 = PBC UPDATED
0165      *
0166      * EXCEPTIONS AND CONDITIONS:
0167      *
0168      *      SAME AS EVSD
```

0170	002C	04C5	EVAR	CLR	R5	; SET FOR VARIABLE ONLY
0171	002E	06A0		BL	@EVALS1	; EVALUATE
	0030	01D8				
0172	0032	0282		CI	R2, EVSKB	; £ IN STACK?
	0034	0000				
0173	0036	1A12		JL	EVER1	; N, RETURN
0174			*			
0175	0038	2F96	ERR22	DATA	ERROR+22	; EXPECTING VARIABLE
0176	003A	0000	EVARZ	DATA	WPR2, EVAR	

```

0178      *      CKEX IS CALLED TO SEE IF AN EXPRESSION
0179      *      COULD OCCUR NEXT.  THE EXCEPTION WOULD
0180      *      OCCUR IF A DELIMITER WAS THE NEXT ITEM.
0181      *
0182      * CALLING SEQUENCE:
0183      *
0184      *      BL @CKEX
0185      *      NO EXPRESSION
0186      *      EXPRESSION
0187      *
0188      *      IN  R8 = PBC
0189      *
0190 003E D818  CKEX  MOVB *R8,@DLIM
      0040 0000
0191 0042 1307      JEQ  CKEX2          ;EOL
0192 0044 9818      CB   *R8,@B38      ;FUNCTION?
      0046 0568'
0193 0048 1A03      JL   CKEX1          ;Y
0194 004A 9818      CB   *R8,@B4C      ;OPERATOR OR VARIABLE?
      004C 056C'
0195 004E 1A01      JL   CKEX2          ;N, TAKE ERROR RETURN
0196      *
0197 0050 05CB  CKEX1  INCT R11
0198 0052 045B  CKEX2  RT
    
```

```
0200      *      EVER WILL EVALUATE A FULL EXPRESSION AND
0201      *      WILL EXIT THRU THE ERROR CALL
0202      *      WITH ANY EXCEPTION.
0203      *
0204      * CALL SEQUENCE:
0205      *
0206      *      BLWP @EVERZ
0207      *
0208      *      IN  R8 = PBC
0209      *      OUT R0 = DLIM
0210      *      R2 = ADR
0211      *      R8 = PBC UPDATED
0212      *
0213      * EXCEPTIONS AND CONDITIONS:
0214      *
0215      *      SAME AS EVSD
0216      *
0217 0054 003A' EVERZ  DATA WPR2,EVER
0218 005B 06A0 EVER   BL   @EVALS           ; EVALUATE
0219      005A 01D6'
0219 005C CB42 EVER1  MOV   R2,@4(13)       ; RETURN PARAMETERS
0219      005E 0004
0220 0060 C800 EVER2  MOV   R0,@DLIM        ; RETURN DELIMITER
0220      0062 0040'
0221 0064 C740      MOV   R0,*R13
0222 0066 CB48      MOV   R8,@16(13)       ; UPDATE PBC
0222      0068 0010
0223 006A 0380      RTWP
```

```

0225      *      EVFX USES AN XOP CALL SINCE IT IS
0226      *      USED SO OFTEN AND RETURNS A 1 WORD,
0227      *      2'S COMPLEMENT INTEGER IN THE OPERAND
0228      *      OF THE EXOP.  EVFX DOES A FULL
0229      *      EVALUATION (R5 <> 0) AND RETURNS
0230      *      R0, R8, AND THE RESULT.
0231      *
0232      *      SINCE R0 IS ALTERED, EVFX R0 WOULD
0233      *      BE ILLEGAL.
0234      *
0235      * CALLING SEQUENCE:
0236      *
0237      *      XOP XX,11
0238      *
0239      *      IN  R8 = PBC
0240      *      OUT R0 = DLIM
0241      *      R8 = PBC UPDATED
0242      *      *R11 = £
0243      *
0244      * EXCEPTIONS AND CONDITIONS:
0245      *
0246      *      SAME AS EVSD WITH THE ADDITION OF
0247      *      A FIX ERROR POSSIBILITY.
0248      *
0249      *      R0,R8 CANNOT BE USED AS THE RETURNING
0250      *      FIELD.
0251      *
0252 006C C80B  EVFX  MOV  R11,@RTSTOR      SAVE ADDRESS
0253      006E 0000
0253 0070 06A0  BL   @EVALS              EVALUATE
0253      0072 01D6'
0254 0074 06A0  BL   @FIX                FIX RESULT
0254      0076 0000
0255 0078 C320  MOV  @RTSTOR,R12          RESTORE RETURN ADDRESS
0255      007A 006E'
0256 007C C701  MOV  R1,*R12              RETURN £
0257 007E 10F0  JMP  EVER2
    
```

```

0259      *      EVAL IS A RECURSIVE STACK EXPRESSION EVALUATOR.
0260      *      R6 POINTS TO THE BOTTOM OF THE STACK WHERE
0261      *      OPERANDS AND OTHER DATA ARE STACKED
0262      *      WHILE R7 POINTS TO THE TOP OF THE STACK WHERE
0263      *      OPERATORS ARE STACKED. BOTH ENDS OF THE STACK
0264      *      ARE MARKED WITH NULLS TO CHECK FOR PROPER
0265      *      EXPRESSION TERMINATION. THE ROUTINE CONSISTS
0266      *      MAINLY OF A MULTIPLEXOR AND A PRECEDENCE ORIENTED
0267      *      PROCESSOR.
0268      *
0269      *      VARIABLES ARE STORED ON A STACK AS AN ADDRESS.
0270      *      CONSTANTS GO DIRECTLY ON THE STACK FOLLOWED BY
0271      *      A -1 TO INDICATE A CONSTANT VALUE. IE:
0272      *
0273      *      * (PTR) * ----- /XXXX XXXX XXXX/
0274      *      * 4110 *
0275      *      * 0000 *
0276      *      * 0000 *
0277      *      * -1 *
0278      *
0279      *      USER FUNCTIONS, SYSTEM FUNCTIONS, AND DIMENSIONED
0280      *      VARIABLES ALL RECURSE ON THE EVALUATOR TO
0281      *      OBTAIN THE ARGUMENTS.
0282      *
0283      *      THE LEFT BYTE OF R5 CONTROLS WHETHER A OPERATOR
0284      *      IS UNARY OR NOT. WHEN NULL AND AN OPERATOR IS
0285      *      ENCOUNTERED, IT MUST BE UNARY! UPON ENCOUNTERING
0286      *      A VARIABLE, THIS LEFT BYTE IS SET INDICATING
0287      *      THE NEXT OPERATOR WILL NOT BE UNARY.
0288      *
0289      *      THE RIGHT BYTE OF R5 CONTROLS HOW FAR THE
0290      *      EVALUATION IS TO GO. IE, ONE ONLY WANTS THE
0291      *      FIRST VARIABLE FOR AN ASSIGNMENT STATEMENT OR
0292      *      ANYWHERE ONE WANTS TO STORE A VALUE. THUS,
0293      *      WHEN THE RIGHT BYTE OF R5 IS ZERO, EVALUATION
0294      *      WILL TERMINATE AT THE FIRST OPERATOR.
0295      *
0296      *      WHEN A DELIMITER IS ENCOUNTED, THE RIGHT BYTE
0297      *      OF R5 IS SET TO ZERO THUS TERMINATING THE
0298      *      EVALUATION OF THE EXPRESSION.
0299      *
0300      *      THE LEAST SIGNIFICANT BIT OF THE OPERATOR
0301      *      ADDRESS IS USED TO INDICATE A UNARY OPERATION.
0302      *
0303      *      USER FUNCTIONS USE THE STACK QUITE EXTENSIVELY
0304      *      PUSHING FUNCTION ADDRESS, DUMMY ARGUMENTS, PLC,
0305      *      R5, AND A RETORE-STACK-POINTER (R4) FOR EACH
0306      *      LEVEL OF FUNCTIONS.
0307      *
0308      *      CALLING SEQUENCE:
0309      *
0310      *      BL @EVALS      ; START EVALUATION
0311      *
0312      *      IN R5 = EXPRESSION FLAG
0313      *
0314      *      (FALL THRU TO EVAL)
0315      *
0316      *      BL @EVAL      ; RECURSIVE EVALUATOR
0317      *
0318      *      R5 = EXPRESSION FLAG

```

```
0319      *           R6 = BOTTOM OF STACK
0320      *           R7 = TOP OF STACK
0321      *           R8 = PLC
0322      *
0323      * EXCEPTIONS AND CONDITIONS:
0324      *
0325      *           IF R6 >= R7, THEN AN EXPRESSION TOO
0326      *           COMPLEX OCCURS.
```



```

0328      *USER FUNCTION EVALUATION
0329      *
0330      *STACK: R5
0331      *      FUNC ADR
0332      *      DUMMY £1
0333      *      DUMMY £2
0334      *      DUMMY £3
0335      *
0336      *      R8
0337      *      R4 = STACK+2
0338      *
0339 0080 2FA0 ERR38 DATA ERROR2,38      ; UNDEFINED FUNCTION
0340      *
0341 0084 CDB5 EVAFN MOV R5,*R6+      ; SAVE R5
0342 0086 C106      MOV R6,R4      ; MARK
0343 0088 C0E0      MOV @UFT,R3      ; GET TABLE ADR
0344 008C 0960      SRL R0,6      ; GET INDEX
0345 008E A0C0      A R0,R3      ; INDEX
0346 0090 CDB3      MOV *R3+,*R6+      ; STACK ADR
0347 0092 13F6      JEQ ERR38      ; NOT DEFINED, ERROR
0348 0094 C053      MOV *R3,R1      ; GET COUNT
0349 0096 132C      JEQ EVAFN5      ; NO ARGUMENTS
0350 0098 C0E0      MOV @VDT,R3
0351 009A 0000
0352 009C CDB3      MOV *R3+,*R6+      ; SAVE OLD DUMMIES
0353 009E CDB3      MOV *R3+,*R6+
0354 00A0 CD93      MOV *R3,*R6+
0355 00A2 0223      AI R3,-4
0356 00A4 FFFC
0357 00A6 0705      SETO R5      ; DO FULL EVAL IF NOT [
0358 00A8 9838      CB *R8+,@B4A      ; [?
0359 00AA 056B'
0360 00AC 1303      JEQ EVAFN1      ; Y
0361 00AE 0608      DEC R8      ; N
0362 00B0 04C5      CLR R5
0363 00B2 0701      SETO R1      ; SET COUNT TO -1
0364      *
0365 00B4 CDB1 EVAFN1 MOV R1,*R6+      ; PUSH COUNT
0366 00B6 CDB3      MOV R3,*R6+
0367 00B8 CDB4      MOV R4,*R6+
0368 00BA 06A0      BL @EVAL      ; EVALUATE NEXT PARAMETER
0369 00BC 01E4'
0370 00BE 0646      DECT R6
0371 00C0 C116      MOV *R6,R4      ; POP R4
0372 00C2 0646      DECT R6
0373 00C4 C0D6      MOV *R6,R3      ; POP SYMBOL TABLE PTR
0374 00C6 0646      DECT R6
0375 00CB C056      MOV *R6,R1      ; POP COUNT
0376 00CA CCC6      MOV R6,*R3+      ; LOAD NEW DUMMY ADR
0377 00CC CDB2      MOV *R2+,*R6+      ; STACK
0378 00CE CDB2      MOV *R2+,*R6+
0379 00D0 CD92      MOV *R2,*R6+
0380 00D2 05B1      INC R1      ; LOOKING FOR VARIABLE ONLY?
0381 00D4 1308      JEQ EVAFN2      ; Y
0382 00D6 0641      DECT R1      ; N, DONE?
0383 00D8 1308      JEQ EVAFN4      ; Y
0384 00DA 0705      SETO R5      ; N, DO FULL
0385 00DC 02B0      CI R0,>3F00      ; ?
0386 00DE 3F00

```

0382	00E0	13E9	JEQ	EVAFN1	; Y
0383	00E2	2FA0	EVER37	DATA ERROR2, 37	; ILLEGAL DELIMITER
0384			*		
0385	00E6	0608	EVAFN2	DEC R8	; BACKUP OVER DELIMITER
0386	00E8	1003	JMP	EVAFN5	
0387			*		
0388	00EA	0280	EVAFN4	CI R0, >4B00	; 1?
	00EC	4B00			
0389	00EE	16F9	JNE	EVER37	; N, ERROR
0390			*		
0391	00F0	CD84	EVAFN5	MOV R4, *R6+	; PUSH R4
0392	00F2	CD88		MOV R8, *R6+	; PUSH R8
0393	00F4	C214		MOV *R4, R8	; GET DEF ADR
0394	00F6	0705		SET0 R5	
0395	00F8	06A0		BL @EVAL	; EVALUATE
	00FA	01E4			
0396	00FC	0646		DECT R6	
0397	00FE	C216		MOV *R6, R8	; POP R8
0398	0100	0646		DECT R6	
0399	0102	C196		MOV *R6, R6	; POP R6 (R4)
0400	0104	C106		MOV R6, R4	
0401	0106	05C4		INCT R4	; MOVE TO DUMMY STORAGE
0402	0108	C0E0		MOV @VDT, R3	
	010A	009A			
0403	010C	CCF4		MOV *R4+, *R3+	; RESTORE OLD VARIABLE ADDRESSES
0404	010E	CCF4		MOV *R4+, *R3+	
0405	0110	C4D4		MOV *R4, *R3	
0406	0112	1051		JMP EVSFR	; RETURN

```
0408      *DO SYSTEM STRING OPERAND
0409      *
0410 0114 9818 EVSFD CB *R8,@B43 ;$?
      0116 0569'
0411 0118 130B JEQ EVSFD2 ;Y
0412 011A 1A09 JL EVSFD1
0413 011C 9818 CB *R8,@B45 ;" OR '?'
      011E 056A'
0414 0120 1B06 JH EVSFD1 ;N
0415 0122 0588 INC R8 ;Y
0416 0124 C088 MOV R8,R2 ;SAVE ADR
0417      *
0418 0126 D038 MOVB *R8+,R0 ;MOVE TO END OF STRING
0419 0128 16FE JNE $-2
0420 012A D038 MOVB *R8+,R0 ;GET DELIMITER
0421 012C 05CB INCT R11
0422      *
0423 012E 045B EVSFD1 RT
0424      *
0425 0130 0588 EVSFD2 INC R8 ;MOVE TO VARIABLE
0426 0132 05CB INCT R11 ;SET RETURN
0427 0134 04C5 CLR R5
0428 0136 0460 B @EVAL ;DO EVALUATION
      0138 01E4'
```

```

0430      *SYSTEM FUNCTION EVALUATION
0431      *
0432 013A CD85 EVSF MOV R5,*R6+ ;STACK R5,R0
0433 013C CD80 MOV R0,*R6+
0434 013E 0201 LI R1,1 ;DEFAULT SECOND PARAMETER TO 1
      0140 0001
0435 0142 9838 CB *R8+,@B4A ;[?
      0144 056B'
0436 0146 1306 JEQ EVSF1 ;Y
0437 0148 04C5 CLR R5 ;N, DO VARIABLE ONLY
0438 014A 0608 DEC R8
0439 014C 06A0 BL @EVAL
      014E 01E4'
0440 0150 0608 DEC R8
0441 0152 1029 JMP EVSF2
0442      *
0443 0154 06A0 EVSF1 BL @EVSFO ;CHECK FOR STRING
      0156 0114'
0444 0158 1009 JMP EVSF1A ;N
0445 015A CD82 MOV R2,*R6+ ;STACK ADR
0446 015C 0280 CI R0,>3F00 ;.?
      015E 3F00
0447 0160 161D JNE EVSF1B ;N
0448 0162 06A0 BL @EVSFO ;GET SECOND STRING
      0164 0114'
0449 0166 10BD JMP EVER37
0450 0168 C042 MOV R2,R1
0451 016A 1018 JMP EVSF1B
0452      *
0453 016C 0705 EVSF1A SETO R5 ;DO FULL EVALUATION
0454 016E 06A0 BL @EVAL
      0170 01E4'
0455 0172 0280 CI R0,>4B00 ;.?
      0174 4B00
0456 0176 1317 JEQ EVSF2 ;Y
0457 0178 0280 CI R0,>3F00 ;.?
      017A 3F00
0458 017C 16B2 JNE EVER37 ;N, ERROR
0459      *
0460      * 2 PARM FUNCTIONS LEAVE PARM1 ON THE STACK UNPROTECTED
0461      * IF IT IS A CONSTANT (AND FORGETS ABOUT IT !)
0462      *
0463 017E 8182 C R2,R6 ;CONST ON STACK ?
0464 0180 1A07 JL E##1 ;N, LEAVE ALONE
0465 0182 0208 LI R11,E$TEMP ;Y, GET ITS STOREAGE
      0184 0000
0466 0186 C142 MOV R2,R5 ;GET ITS PLACE ON STACK
0467 0188 C08B MOV R11,R2 ;SAVE STOREAGE ADR
0468 018A CEF5 MOV *R5+,*R11+ ;COPY IT
0469 018C CEF5 MOV *R5+,*R11+
0470 018E C6D5 MOV *R5,*R11
0471 0190 CD82 E##1 MOV R2,*R6+ ;STACK ADR
0472 0192 0705 SETO R5
0473 0194 06A0 BL @EVAL ;GET SECOND PARAMETER
      0196 01E4'
0474 0198 06A0 BL @FIX
      019A 0076'
0475      *
0476 019C 0646 EVSF1B DECT R6 ;POP 1ST OPERAND
0477 019E C096 MOV *R6,R2
  
```

```

0478 01A0 0280          CI    R0,>4B00          ;J?
      01A2 4B00
0479 01A4 169E          JNE    EVER37          ;N, ERROR
0480          *
0481 01A6 0646  EVSF2  DECT  R6          ;POP ID
0482 01AB C0D6          MOV    *R6,R3
0483 01AA 0973          SRL    R3,7          ;GET INDEX
0484 01AC C2E3          MOV    @SFUNP->36(3),R11
      01AE 03DE
0485 01B0 1308          JEQ    SYSERR          ;O, SYSTEM ERROR
0486 01B2 2E00          CLEAR  0          ;CLEAR FPAC
0487 01B4 069B          BL     *R11          ;GOTO ROUTINE
0488          *
0489 01B6 0646  EVSFR  DECT  R6          ;POP R5
0490 01B8 C156          MOV    *R6,R5
0491          *
0492 01BA CDB2  EVSF4  MOV    *R2+,*R6+          ;MOVE RESULT ON STACK
0493 01BC CDB2          MOV    *R2+,*R6+
0494 01BE CD92          MOV    *R2,*R6+
0495 01C0 1069          JMP     EVAL6          ;CONTINUE
0496          *
0497 01C2 2FA0  SYSERR DATA ERROR2,-1          ;SYSTEM ERROR
0498          *
0499          * FUNCTION RETURN WITH R2 RELOAD
0500          *
0501 01C6 0202  EVSFR$ LI    R2,FPAC          ;RELOAD R2
      01CB 0000
0502 01CA 10F5          JMP     EVSFR          ;RETURN
0503          *
0504          *GET RANDOM NUMBER
0505          *
0506 01CC 0420  EVRND  BLWP  @RANDZ          ;GET RANDOM £ IN FPAC
      01CE 0000
0507 01D0 0202          LI     R2,@FPAC
      01D2 01CB
0508 01D4 10F2          JMP     EVSF4
    
```

```

0510      *
0511 01D6 0705 EVALS SETD R5 ; FULL EXPRESSION
0512      *
0513 01D8 C22D EVALS1 MOV @16(13),R8 ; GET PBC
      01DA 0010
0514      *
0515 01DC 0206 EVALS2 LI R6,EVSKB ; GET BEGINNING STACK PTR
      01DE 0034
0516 01E0 0207 LI R7,EVSKE ; GET END STACK PTR
      01E2 0000
0517      *
0518      *EVALUATE SIMPLE EXPRESSION
0519      *
0520 01E4 CD8B EVAL MOV R11,*R6+ ; STACK RETURN
0521 01E6 04F6 CLR *R6+ ; MARK OPERAND STACK
0522 01E8 04C0 CLR R0 ; MARK OPERATOR STACK
0523 01EA D140 MOVVB R0,R5 ; IF OPERATOR: MUST BE UNARY
0524      *
0525 01EC 0607 EVAL1 DEC R7 ; PUSH ON OPERATOR STACK
0526 01EE D5C0 MOVVB R0,*R7
0527 01F0 81C6 C R6,R7 ; STACK OVERFLOW?
0528 01F2 1457 JHE EVER27 ; Y, ERROR
0529      *
0530 01F4 D038 EVAL2 MOVVB *R8+,R0 ; PARSE
0531 01F6 1376 JEQ EVADL ; DONE
0532 01F8 0280 CI R0,>6F00 ; VARIABLE?
      01FA 6F00
0533 01FC 1B22 JH EVAV ; Y
0534 01FE 130F JEQ EVFP ; N, FP £
0535 0200 0280 CI R0,>6200 ; CONSTANT?
      0202 6200
0536 0204 1413 JHE EVAL3 ; Y
0537 0206 0280 CI R0,>4C00 ; "( OR OPERATER?
      0208 4C00
0538 020A 1B64 JH EVOP ; OPERATER
0539 020C 13EF JEQ EVAL1 ; "(, PUSH
0540 020E 0280 CI R0,>3800 ; DELIMITER?
      0210 3800
0541 0212 1468 JHE EVADL ; Y
0542 0214 0280 CI R0,>1B00 ; SYSTEM FUNCTION?
      0216 1B00
0543 0218 1490 JHE EVSF ; Y
0544 021A 0460 B @EVAFN ; FUNCTION
      021C 00B4
0545      *
0546 021E DDB8 EVFP MOVVB *R8+,*R6+ ; FP CONSTANT
0547 0220 DDB8 MOVVB *R8+,*R6+
0548 0222 DDB8 MOVVB *R8+,*R6+
0549 0224 DDB8 MOVVB *R8+,*R6+
0550 0226 DDB8 MOVVB *R8+,*R6+
0551 0228 DDB8 MOVVB *R8+,*R6+
0552 022A 1034 JMP EVAL6 ; MARK
0553      *
0554 022C 0280 EVAL3 CI R0,>6D00 ; 1 WORD INTEGER?
      022E 6D00
0555 0230 142A JHE EVAL4 ; Y
0556 0232 06C0 SWPB R0
0557 0234 0220 AI R0,->63 ; N, -1 THRU 9
      0236 FF9D
0558 0238 102A JMP EVAL5 ; STORE £

```

0559

*

0560 023A 2F8A EVER10 DATA ERROR+10

; STORAGE OVERFLOW

```

0562      *UNARY MINUS
0563      *
0564 023C 0200 EVAL9 LI R0,>6100      ; STACK -
      023E 6100
0565 0240 10D5 EVAL91 JMP EVAL1
0566      *
0567      *VARIABLE
0568      *
0569 0242 0280 EVAV CI R0,>7300      ; RND?
      0244 7300
0570 0246 13C2 JEQ EVRND      ; Y
0571 0248 C060 MOV @VNT,R1      ; GET VARIABLE TABLE
      024A 0000
0572 024C 0970 SRL R0,7      ; SWAP & X 2
0573 024E A040 A R0,R1      ; INDEX
0574 0250 C061 MOV @->70*2(1),R1      ; DIMENSIONED
      0252 FF20
0575 0254 117F JLT EVADV      ; Y
0576 0256 C060 MOV @VDT,R1
      0258 010A
0577 025A A040 A R0,R1
0578 025C C021 MOV @->70*2(1),R0      ; DEFINED?
      025E FF20
0579 0260 161A JNE EVAL7      ; Y
0580 0262 06C5 SWPB R5      ; POSITION EVALUATION TYPE LJ IN
0581 0264 D145 MOVB R5,R5      ; FULL OR VARIABLE EVALUATION ?
0582 0266 06C5 SWPB R5      ; RESTORE EVALUATION TYPE TO ORI
0583 0268 160C JNE EVER40      ; NE - FULL EVAL - DO NOT DEFINE
0584 026A C020 MOV @NVS,R0      ; VARIABLE EVALUATION ONLY - DEF
      026C 0000
0585 026E 0220 AI R0,-6
      0270 FFFA
0586 0272 8800 C R0,@NVD      ; OK?
      0274 0000
0587 0276 1AE1 JL EVER10      ; N, STORAGE OVERFLOW
0588 0278 C840 MOV R0,@->70*2(1)
      027A FF20
0589 027C C800 MOV R0,@NVS      ; UPDATA NVS
      027E 026C
0590 0280 100A JMP EVAL7
0591 0282 2FA0 EVER40 DATA ERROR2,40      ; UNDEFINED VARIABLE
0592      *
0593      *CONSTANTS
0594      *
0595 0286 D038 EVAL4 MOVB *R8+,R0      ; 1 WRD CONSTANT
0596 0288 06C0 SWPB R0
0597 028A D038 MOVB *R8+,R0
0598 028C 06C0 SWPB R0
0599      *
0600 028E 04F6 EVAL5 CLR *R6+      ; MOVE INTO STACK
0601 0290 CD80 MOV R0,*R6+      ; INSERT 0,£,0
0602 0292 04F6 CLR *R6+
0603      *
0604 0294 0700 EVAL6 SETD R0      ; MARK AS CONSTANT
0605      *
0606 0296 CD80 EVAL7 MOV R0,*R6+      ; MARK STACK
0607 0298 04C0 CLR R0
0608      *
0609 029A D160 EVAL8 MOVB @BFF,R5      ; IF OPERATOR: MUST NOT BE UNARY
      029C 056D
    
```


0610	029E	81C6	C	R6,R7	; STACK OVERFLOW?
0611	02A0	1AA9	JL	EVAL2	; N, OK
0612		*			
0613	02A2	2F9B	EVER27	DATA ERROR+27	; Y, EXPRESSION TOO COMPLEX

```

0615      *OPERATOR
0616      *
0617      *HANDLE UNARY OPERATORS
0618      *
0619 02A4 0280  EVOPA  CI   R0,>5D00      ; "+?
          02A6 5D00
0620 02A8 13A5      JEQ  EVAL2          ; Y, IGNOR
0621 02AA 0280      CI   R0,>5C00      ; N, "-?
          02AC 5C00
0622 02AE 13C6      JEQ  EVAL9          ; Y, STACK UNARY -
0623 02B0 0280      CI   R0,>5200      ; NOT?
          02B2 5200
0624 02B4 139B      JEQ  EVAL1          ; Y, STACK
0625 02B6 0280      CI   R0,>5300      ; LNOT?
          02B8 5300
0626 02BA 139B      JEQ  EVAL1          ; Y, STACK
0627 02BC 2FB1  EVERO  DATA ERROR+1    ; N, SYNTAX ERROR
0628      *
0629      *HANDLE OPENING AND CLOSING PAREN'S
0630      *
0631 02BE 0280  EVOP0  CI   R0,>4D00      ; ")?
          02C0 4D00
0632 02C2 1612      JNE  EVOP1          ; N
0633 02C4 9817  EVOP0A CB   *R7,@B4C    ; (?
          02C6 056C
0634 02C8 160F      JNE  EVOP1          ; N
0635 02CA 0587      INC  R7              ; Y, POP
0636 02CC 1093      JMP  EVAL2
0637      *
0638 02CE D057  EVOP0B MOVB *R7,R1      ; NULL?
0639 02D0 16F9      JNE  EVOP0A        ; N
0640 02D2 100B      JMP  EVADL         ; Y, TERMINATE
0641      *
0642      *OPERATER ENTRY
0643      *
0644 02D4 D045  EVOP   MOVB R5,R1      ; NEED UNARY OPERATER?
0645 02D6 13E6      JEQ  EVOPA         ; Y
0646 02D8 0280      CI   R0,>4D00      ; ))?
          02DA 4D00
0647 02DC 13F8      JEQ  EVOP0B        ; Y
0648 02DE 0245      ANDI R5,>FF        ; N, ALLOW UNARY ONLY TO FOLLOW
          02E0 00FF
0649 02E2 1602      JNE  EVOP1        ; FULL EXPRESSION
0650      *
0651      *DELIMITER ENTRY
0652      *
0653 02E4 04C5  EVADL  CLR  R5          ; SET TO TERMINATE
0654 02E6 04C0      CLR  R0          ; DO CR
0655      *
0656 02E8 D057  EVOP1  MOVB *R7,R1      ; LOOK AT TOP OPERATOR
0657 02EA C100      MOV  R0,R4
0658 02EC 0244      ANDI R4,>FE00      ; LAP OFF LOW BIT
          02EE FE00
0659 02F0 9044      CB   R4,R1        ; HIGHER PRECEDENCE?
0660 02F2 1BA6      JH   EVAL91        ; Y, STACK
0661 02F4 D0F7      MOVB *R7+,R3      ; N, POP OPERATER
0662 02F6 131C      JEQ  EVOP4         ; DONE
0663 02FB 0973      SRL  R3,7          ; SWAP & X 2
0664 02FA C2E3      MOV  @EVATB->9C(3),R11
          02FC 0342

```

```

0665      *
0666 02FE 0646      DECT R6      ; POP VARIABLE
0667 0300 C056      MOV  *R6,R1  ; EMPTY?
0668 0302 13DC      JEQ  EVER0    ; Y
0669 0304 0596      INC  *R6      ; CONSTANT?
0670 0306 1603      JNE  EVOP2    ; N
0671 0308 0226      AI   R6,-6    ; Y
      030A FFFA
0672 030C C046      MOV  R6,R1
0673      *
0674 030E 22E0 EVOP2 CDC  @C1,R11 ; UNARY OPERATION?
      0310 0408
0675 0312 1308      JEQ  EVOP3    ; Y
0676 0314 0646      DECT R6      ; N, GET SECOND OPERAND
0677 0316 C096      MOV  *R6,R2  ; EMPTY?
0678 0318 13D1      JEQ  EVER0    ; Y
0679 031A 0596      INC  *R6      ; CONSTANT?
0680 031C 1603      JNE  EVOP3    ; N
0681 031E 0226      AI   R6,-6    ; Y
      0320 FFFA
0682 0322 C086      MOV  R6,R2
0683      *
0684 0324 069B EVOP3 BL   *R11    ; DO OPERATION
0685      *
0686 0326 CDB2 EVOP3A MOV  *R2+,*R6+ ; MOVE RESULTS ON STACK
0687 0328 CDB2      MOV  *R2+,*R6+
0688 032A CD92      MOV  *R2,*R6+
0689 032C 0736      SETO *R6+    ; MARK AS CONSTANT
0690 032E 10C7      JMP  EVOP0    ; CONTINUE
0691      *
0692 0330 D02B EVOP4 MOVB @-1(8),R0 ; GET DELIMITER
      0332 FFFF
0693 0334 0646      DECT R6      ; POP VARIABLE
0694 0336 C096      MOV  *R6,R2  ; EMPTY?
0695 0338 13C1      JEQ  EVER0    ; Y, ERROR
0696 033A 0596      INC  *R6      ; CONSTANT?
0697 033C 1603      JNE  EVOP5    ; N
0698 033E 0226      AI   R6,-6    ; Y
      0340 FFFA
0699 0342 C086      MOV  R6,R2
0700      *
0701 0344 0226 EVOP5 AI   R6,-4    ; DONE, POP STACK
      0346 FFFC
0702 0348 C2D6      MOV  *R6,R11
0703 034A C066      MOV  @2(6),R1 ; STACK EMPTIED?
      034C 0002
0704 034E 16B6      JNE  EVER0    ; N, ERROR
0705 0350 045B      RT

```

```

0707 0352 10A3 EVALBP JMP EVALB
0708 *
0709 *DIMENSIONED VARIABLE
0710 *
0711 0354 C060 EVADV MOV @VDT,R1
      0356 0258'
0712 0358 A040 A R0,R1
0713 035A C121 MOV @->70*2(1),R4 ;DEFINED?
      035C FF20
0714 035E 133B JEQ ERR39 ;N, UNDIMENSIONED VARIABLE
0715 0360 CD85 MOV R5,*R6+ ;STACK R5
0716 0362 04F6 CLR *R6+ ;ZERO INDEX & STACK
0717 *
0718 0364 CD84 EVADV1 MOV R4,*R6+ ;STACK R4
0719 0366 0705 SETO R5 ;DO FULL EXPRESSION
0720 0368 06A0 BL @EVAL ;RECURSE
      036A 01E4'
0721 036C 06A0 BL @FIX ;FIX £
      036E 019A'
0722 0370 0646 DECT R6 ;POP BASE ADR
0723 0372 C116 MOV *R6,R4
0724 0374 0646 DECT R6 ;READY FOR INDEX POP
0725 0376 8D01 C R1,*R4+ ;EXCEED DIMENSION?
0726 0378 1B2C JH ERR17 ;Y, SUBSCRIPT ERROR
0727 037A C174 MOV *R4+,R5 ;GET DEL MULTIPLIER
0728 037C 0285 CI R5,-1 ;DONE?
      037E FFFF
0729 0380 1306 JEQ EVADV2 ;Y
0730 0382 3845 MPY R5,R1 ;MULTIPLY
0731 0384 AD82 A R2,*R6+ ;ADD TO INDEX & STACK
0732 0386 0280 CI R0,>3F00 ;?
      0388 3F00
0733 038A 13EC JEQ EVADV1 ;Y, DO AGAIN
0734 *
0735 038C 2F92 ERR18 DATA ERROR+18 ;N, TOO FEW SUBSCRIPTS
0736 *
0737 038E A056 EVADV2 A *R6,R1 ;ADD FINAL DIMENSION
0738 0390 3860 MPY @C6,R1 ;MULTIPLY BY 6
      0392 0412'
0739 0394 A084 A R4,R2 ;ADD BASE
0740 0396 0646 DECT R6 ;POP R5
0741 0398 C156 MOV *R6,R5
0742 039A CD82 MOV R2,*R6+ ;STACK ADDRESS
0743 039C 0280 CI R0,>4B00 ;?
      039E 4B00
0744 03A0 13D8 JEQ EVALBP ;Y
0745 03A2 0280 CI R0,>4000 ;?
      03A4 4000
0746 03A6 1616 JNE ERR19 ;N, TOO MANY SUBSCRIPTS
0747 03AB CD85 MOV R5,*R6+ ;Y, STACK R5
0748 03AA 0705 SETO R5 ;DO FULL EVAL
0749 03AC 06A0 BL @EVAL ;RECURSE
      03AE 01E4'
0750 03B0 06A0 BL @FIX ;GET INTEGER
      03B2 036E'
0751 03B4 0280 CI R0,>4B00 ;?
      03B6 4B00
0752 03B8 1610 JNE EVADVE ;N, ERROR
0753 03BA 0646 DECT R6 ;Y, POP R5
0754 03BC C156 MOV *R6,R5

```

0755	03BE	0646	DECT	R6		; POP ADR
0756	03C0	C016	MOV	*R6, R0		
0757	03C2	0601	DEC	R1		; INDEX=COUNT-1
0758	03C4	1106	JLT	ERR17		; ERROR
0759	03C6	A001	A	R1, R0		; INDEX
0760	03C8	8800	C	R0, @IOB		; EXCEED MEMORY?
	03CA	0000				
0761	03CC	1402	JHE	ERR17		; Y, ERROR
0762	03CE	0460	B	@EVAL7		; N, CONTINUE
	03D0	0296				
0763			*			
0764	03D2	2F91	ERR17	DATA ERROR+17		; SUBSCRIPT ERROR
0765			*			
0766	03D4	2F93	ERR19	DATA ERROR+19		; TOO MANY SUBSCRIPTS
0767			*			
0768	03D6	2FA0	ERR39	DATA ERROR2, 39		; UNDIMENSIONED VARIABLE
0769			*			
0770	03DA	2FA0	EVADVE	DATA ERROR2, 37		; INVALID DELIMITER
0771			*			
0772	03DE	0000	EVATB	DATA ORF, LORF		
0773	03E2	0000		DATA ANDF, LANDF		
0774	03E6	0000		DATA NOTF, LNOTF		
0775	03EA	0000		DATA LXORF		
0776	03EC	0516		DATA FUZF		
0777	03EE	0516		DATA EQUF, GTHF		
0778	03F2	0516		DATA GEGF, LTHF		
0779	03F6	0516		DATA LEGF, NEQUF		
0780	03FA	0498		DATA SUBF, ADDF		
0781	03FE	04DA		DATA DIVF, MULF		
0782	0402	0000		DATA POWF, UMNF+1		
0783			*			
0784		0028	EVPR	EQU \$-EVATB		
0785	0406	0000		DATA 0		
0786		0409	B01	EQU \$+1		
0787	0408	0001	C1	DATA 1		
0788	040A	0004	C4	DATA 4		
0789		040D	B05	EQU \$+1		
0790	040C	0005		DATA 5		
0791	040E	0002		DATA 2		
0792	0410	0003		DATA 3		
0793	0412	0006	C6	DATA 6		
0794			*			
0795	0414	0000	SFUNP	DATA ABSF	1B	
0796	0416	0000		DATA ADRF	1C	
0797	0418	0000		DATA ASCF	1D	
0798	041A	0000		DATA ATNF	1E	
0799	041C	0000		DATA COSF	1F	
0800	041E	0000		DATA EXPF	20	
0801	0420	0000		DATA FRAF	21	
0802	0422	0000		DATA INTF	22	
0803	0424	0000		DATA LOGF	23	
0804	0426	0000		DATA NKYF	24	
0805	0428	0000		DATA SINP	25	
0806	042A	0000		DATA SGRF	26	
0807	042C	0000		DATA SYSP	27	
0808	042E	0000		DATA TICF	28	
0809	0430	0000		DATA SGNF	29	
0810	0432	0000		DATA BITF	2A	
0811	0434	0000		DATA CRBF	2B	
0812	0436	0000		DATA CRFF	2C	

0813	0438	0000	DATA MEMF	2D
0814	043A	0000	DATA MWDF	2E
0815	043C	0000	DATA LENF	2F
0816	043E	0000	DATA MCHF	30
0817	0440	0000	DATA POSF	31
0818	0442	0000	DATA COLF	32
0819	0444	0000	DATA MODF	33
0820	0446	0000	DATA 0	34
0821	0448	0000	DATA 0	35
0822	044A	0000	DATA 0	36
0823	044C	0000	DATA 0	37

```

0825      *      ARITHMETIC OPERATIONS SUCH AS ADD, SUBTRACT,
0826      *      MULTIPLY, DIVIDE, UNARY MINUS, AND RELATIONAL
0827      *      OPERATORS SUCH AS FUZZ, EQUAL, NOT EQUAL,
0828      *      LESS THAN, GREATER THAN, LESS THAN OR EQUAL,
0829      *      GREAT THAN OR EQUAL, AND NOT EQUAL USE
0830      *      OPERANDS POINTED TO BY R1,R2 AND RETURN A
0831      *      RESULT POINTED TO BY R2.
0832      *
0833      *      PREP DETERMINES IF BOTH OPERANDS ARE INTEGER
0834      *      VALUES (IN WHICH CASE AN ATTEMPT IS MADE TO
0835      *      DO AN INTERATION OPERATION) OR IF 1 OR BOTH
0836      *      OF THE OPERANDS IS A FLOATING POINT NUMBER
0837      *      (IN WHICH CASE BOTH ARE FLOATED AND A FLOATING
0838      *      POINT OPERATION IS PERFORMED.)
0839      *

```

```

0840      *      RELATIONAL OPERATORS ARE GIVEN A VALUE CORRE-
0841      *      SPONDING TO THE COMPARISON TO BE MADE. BIT 0
0842      *      IS THE EQUAL BIT, BIT 1 IS THE LESS-THAN BIT,
0843      *      AND BIT 2 IS THE GREATER-THAN BIT.  HENCE:
0844      *

```

```

0845      *      0000 = FUZZ
0846      *      0001 = EQUAL
0847      *      0002 = LESS-THAN
0848      *      0003 = LESS-THAN OR EQUAL
0849      *      0004 = GREATER-THAN
0850      *      0005 = GREATER-THAN OR EQUAL
0851      *      0006 = NOT EQUAL (LESS-THAN, GREATER-THAN)
0852      *

```

```

0853      *      CALLING SEQUENCE:
0854      *

```

```

0855      *      BL @ADDF
0856      *      BL @SUBF
0857      *      BL @MULF
0858      *      BL @DIVF
0859      *
0860      *      IN (R1) = OPERAND 1
0861      *      (R2) = OPERAND 2
0862      *

```

```

0863      *      OUT (R2) = RESULT
0864      *
0865      *
0866      *      B @FUZF          ;==
0867      *      B @EQUF          ;=
0868      *      B @LTHF          ;<
0869      *      B @LEQF          ;<=
0870      *      B @GTHF          ;>
0871      *      B @GEQF          ;>=
0872      *      B @NEQF          ;<>
0873      *

```

```

0874      *      IN (R1) = OPERAND 1
0875      *      (R2) = OPERAND 2
0876      *

```

```

0877      *      OUT (R2) = RESULT
0878      *      EXITS TO EVOP3A
0879      *

```

```

0880      *      B @UMNF          ; - (UNARY MINUS)
0881      *

```

```

0882      *      IN (R1) = OPERAND
0883      *

```

```

0884      *      OUT (R2) = RESULT

```

0885

*

EXISTS TO EVOP3A


```

0887      *ARITHMETIC PREPARATION
0888      *      BL @PREP
0889      *      VECTOR
0890      *      BOTH INTEGERS
0891      *
0892      *      IN      R1 = SOURCE 1
0893      *      R2 = SOURCE 2
0894      *
0895      *      INT OUT R3 = S1
0896      *      R1 = S2
0897      *
0898 044E C811  PREP  MOV  *R1,@TEMP      ; LOAD TEMP
          0450 0000
0899 0452 C821      MOV  @2(1),@TEMP2
          0454 0002
          0456 0000
0900 0458 C821      MOV  @4(1),@TEMP4
          045A 0004
          045C 0000
0901 045E 2C12      LOADF *R2      ; LOAD FPAC
0902 0460 C0F1      MOV  *R1+,R3    ; INTEGER?
0903 0462 160C      JNE  PREP4      ; N
0904 0464 C0F2      MOV  *R2+,R3    ; INTEGER?
0905 0466 160A      JNE  PREP4      ; N
0906 0468 C0D1      MOV  *R1,R3     ; BOTH INTEGERS
0907 046A C052      MOV  *R2,R1     ; GET INTEGERS
0908 046C 046B      B     @2(11)
          046E 0002
  
```

0910	0470	C28B	ADDF	MOV	R11,R10	; SAVE RETURN
0911	0472	06A0		BL	@PREP	; PREPARE OPEANDS
	0474	044E'				
0912	0476	2CA0		DATA	2*>40+>2C20	
0913	0478	A043		A	R3,R1	; BOTH INTEGERS
0914	047A	1907		JND	PREP5A	; NO OVERFLOW
0915	047C	0420	PREP4	BLWP	@FLTMZ	; FLOAT TEMP
	047E	0494'				
0916	0480	2E80		FLOATF	0	; FLOAT FPAC
0917	0482	0498		X	*R11	; DO FP OPERATION
0918	0484	0450'		DATA	TEMP	
0919	0486	1003		JMP	PREP5B	; RETURN
0920			*			
0921	0488	0501	PREP5	NEG	R1	; NEGATE RESULT
0922			*			
0923	048A	C801	PREP5A	MOV	R1,@FPAC2	; SAVE RESULT
	048C	0000				
0924			*			
0925	048E	0202	PREP5B	LI	R2,FPAC	; GET ADDRESS OF RESULT
	0490	01D2'				
0926	0492	045A		B	*R10	; RETURN
0927			*			
0928	0494	0484'	FLTMZ	DATA	TEMP,FLOAT	
0929			*			
0930			* SUBTRACT			
0931			*			
0932			*			
0933	0498	C28B	SUBF	MOV	R11,R10	; SAVE RETURN
0934	049A	06A0		BL	@PREP	
	049C	044E'				
0935	049E	2CE0		DATA	3*>40+>2C20	
0936	04A0	6043		S	R3,R1	; INTEGERS
0937	04A2	19F3		JND	PREP5A	
0938	04A4	10EB		JMP	PREP4	; OVERFLOW

```

0940      *
0941      *MULTIPLY
0942      *
0943 04A6 C28B MULF  MOV  R11,R10      ; SAVE RETURN
0944 04AB 06A0      BL   @PREP
      04AA 044E
0945 04AC 2D20      DATA 4*>40+>2C20
0946 04AE C0C3      MOV  R3,R3      ; R3<0?
0947 04B0 1108      JLT  MULF5      ; Y
0948 04B2 C041      MOV  R1,R1      ; R1<0?
0949 04B4 1108      JLT  MULF6      ; Y
0950      *
0951 04B6 3843 MULF2 MPY  R3,R1      ; R1,R2 = R3 X R1
0952 04B8 C041      MOV  R1,R1      ; OVERFLOW?
0953 04BA 16E0      JNE  PREP4      ; Y
0954 04BC C042      MOV  R2,R1      ; OVERFLOW?
0955 04BE 11DE      JLT  PREP4      ; Y
0956 04C0 10E4      JMP  PREP5A     ; N
0957      *
0958 04C2 0503 MULF5 NEG  R3      ; R3<0
0959 04C4 C041      MOV  R1,R1      ; R1<0
0960 04C6 1503      JGT  MULF7      ; N
0961 04C8 0501      NEG  R1      ; Y
0962 04CA 10F5      JMP  MULF2      ; Y
0963      *
0964 04CC 0501 MULF6 NEG  R1      ; SIGNS DIFFERENT
0965 04CE 3843 MULF7 MPY  R3,R1      ; R1,R2 = R3 X R1
0966 04D0 C041      MOV  R1,R1      ; OVERFLOW?
0967 04D2 16D4      JNE  PREP4      ; Y
0968 04D4 C042      MOV  R2,R1      ; OVERFLOW?
0969 04D6 11D2      JLT  PREP4      ; Y
0970 04D8 10D7      JMP  PREP5      ; RETURN
  
```

```

0972      *
0973      *DIVIDE
0974      *
0975 04DA C28B DIVF MOV R11,R10 ;SAVE RETURN
0976 04DC 06A0 BL @PREP
      04DE 044E'
0977 04E0 2D60 DATA 5*>40+>2C20
0978 04E2 C0C3 MOV R3,R3 ;SOURCE<0?
0979 04E4 110A JLT DIVF5 ;Y
0980 04E6 1316 JEQ ERR28 ;DIVISION BY ZERO
0981 04E8 C081 MOV R1,R2 ;DESTINATION<0?
0982 04EA 110C JLT DIVF6 ;Y
0983      *
0984 04EC 04C1 DIVF2 CLR R1 ;POSITIVE RESULT
0985 04EE 3C43 DIV R3,R1 ;DIVIDE
0986 04F0 C082 MOV R2,R2 ;REMAINDER?
0987 04F2 16C4 JNE PREP4 ;Y, FLOAT
0988 04F4 C041 MOV R1,R1 ;OVERFLOW?
0989 04F6 11C2 JLT PREP4 ;Y, FLOAT
0990 04F8 10C8 JMP PREP5A ;N, STORE RESULTS
0991      *
0992 04FA 0503 DIVF5 NEG R3 ;SOURCE<0
0993 04FC C081 MOV R1,R2 ;DESTINATION<0?
0994 04FE 1503 JGT DIVF7 ;N
0995 0500 0502 NEG R2 ;Y, POSITIVE RESULT
0996 0502 10F4 JMP DIVF2
0997      *
0998 0504 0502 DIVF6 NEG R2 ;SIGNS DIFFERENT
0999 0506 04C1 DIVF7 CLR R1
1000 0508 3C43 DIV R3,R1 ;DIVIDE
1001 050A C082 MOV R2,R2 ;REMAINDER?
1002 050C 16B7 JNE PREP4 ;Y, OVERFLOW
1003 050E C041 MOV R1,R1 ;OVERFLOW?
1004 0510 11B5 JLT PREP4 ;Y
1005 0512 10BA JMP PREP5 ;RETURN
1006      *
1007 0514 2F9C ERR28 DATA ERROR+28 ;DIVISION BY ZERO
  
```

```

1009      *
1010      *RELATIONAL OPERATORS
1011      *
1012      0516' FUZF   EQU   $
1013      0516' EQUF   EQU   $
1014      0516' LTHF   EQU   $
1015      0516' LEQF   EQU   $
1016      0516' GTHF   EQU   $
1017      0516' GEGF   EQU   $
1018      0516' NEQUF  EQU   $
1019      0516 C123    MOV   @EVATB->AA+EVPR(3),R4
          0518 035C'
1020      051A 06A0    BL     @SUBF
          051C 0498'
1021      051E 0203    LI     R3,4           ; GET MATCH CONSTANT
          0520 0004
1022      0522 C104    MOV   R4,R4           ; FUZZ?
1023      0524 1609    JNE   EQUF1          ; N
1024      0526 0584    INC   R4              ; Y, R4=0001
1025      0528 C052    MOV   *R2,R1          ; GET EXPONENT
1026      052A 1306    JEQ   EQUF1          ; INTEGER
1027      052C 0241    ANDI  R1,>7F00        ; ISOLATE
          052E 7F00
1028      0530 0281    CI     R1,FUZZ        ; LESS THAN FUZZ?
          0532 0000
1029      0534 1A06    JL     EQUF2A         ; Y, EQUAL
1030      0536 1007    JMP   EQUF3          ; N, UNEQUAL, SET R3 TO -2
1031      *
1032      0538 C072    EQUF1 MOV   *R2+,R1    ; GET SIGN OF RESULT
1033      053A 1601    JNE   EQUF2
1034      053C C052    MOV   *R2,R1
1035      053E 1503    EQUF2 JGT   EQUF3
1036      0540 1601    JNE   $+4
1037      0542 0603    EQUF2A DEC   R3
1038      0544 0643    DECT  R3
1039      *
1040      0546 2E00    EQUF3 CLEAR 0          ; CLEAR FPAC
1041      0548 2103    COC   R3,R4           ; CORRESPONDING BITS SET?
1042      054A 160A    JNE   UMNFB
1043      054C 05A0    INC   @FPAC2          ; RETURN 1
          054E 048C'
1044      0550 1007    JMP   UMNFB
  
```

```

1046      *
1047      *UNARY MINUS
1048      *
1049 0552 2C11  UMNFB  LOADF *R1      ; LOAD FPAC
1050 0554 C051      MOV  *R1,R1      ; INTEGER
1051 0556 1302      JEQ  UMNFB1      ; Y
1052 0558 2E40      NEGATE 0         ; N, NEGATE FPAC
1053 055A 1002      JMP  UMNFB3
1054      *
1055 055C 0520  UMNFB1  NEG  @FPAC2
      055E 054E'
1056      *
1057 0560 0202  UMNFB3  LI   R2,FPAC      ; RETURN
      0562 0490'
1058 0564 0460  UMNFB4  B    @EVOP3A
      0566 0326'
1059 0568      38  B38      BYTE >38
1060 0569      43  B43      BYTE >43
1061 056A      45  B45      BYTE >45
1062 056B      4A  B4A      BYTE >4A
1063 056C      4C  B4C      BYTE >4C
1064 056D      FF  BFF      BYTE >FF
1065 056E      EVEN
1066      END
NO ERRORS,      NO WARNINGS
  
```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.EXPF
OBJECT ACCESS NAME= ADHOC.OBJ.EXPF
LISTING ACCESS NAME= ADHOC.LST.EXPF
ERROR ACCESS NAME=
OPTIONS= BUNLST, DUNLST, MUNLST
MACRO LIBRARY PATHNAME=

```

0002          IDT 'EXPF'
0003          *
0004          *          EXPF          ;EXPONENTIAL FUNCTION
0005          *
0006          REF FUNFX          ;FIXES ARGUMENT
0007          REF PLYX,PLYXX      ;EVALUATE POLYNOMIALS
0008          REF FUNBK          ;BREAK FP &
0009          REF GETP2          ;GET POWER OF 2
0010          REF FPAC          ;FLOATING POINT ACCUMULATOR
0011          REF TEMP          ;3 WRD TEMPORARY STORAGE
0012          REF DS1          ;3 WRD TEMPORARY STORAGE
0013          REF DS            ;3 WRD TEMPORARY STORAGE
0014          DEF EXPF
0015          *
0016          DXOP LOADF,0        ;LOAD FPAC
0017          DXOP STORE,1        ;STORE FPAC
0018          DXOP FADD,2         ;ADD TO FPAC
0019          DXOP FSUB,3         ;SUBTRACT FROM FPAC
0020          DXOP FMUL,4         ;MULTIPLY FPAC
0021          DXOP FDIV,5         ;DIVIDE FPAC
0022          2F80 ERROR EQU >2F80 ;XOP XX,14 (ERROR CALL)
0023          2FA0 ERROR2 EQU ERROR+>20
  
```



```
0025      *
0026      *      CALCULATE THE EXPONENTIAL VALUE OF
0027      *      E RAISED TO (ARG)
0028      *
0029      * CALLING SEQUENCE:
0030      *
0031      *      B @EXPF
0032      *
0033      *      IN (R2) = ARG
0034      *      OUT (R2) = E ^ ARG
0035      *
0036      * EXCEPTIONS AND CONDITIONS:
0037      *
0038      *      ERROR 33 = EXP ARGUMENT TOO LARGE
0039      *      FLOATING POINT ERRORS
```

0041	0000	2FA0	ERR33	DATA ERROR2,33	; EXP ARG TOO LARGE
0042			*		
0043	0004	C28B	EXPF	MOV R11,R10	
0044	0006	06A0	BL	@FUNFX	; FIX SIGN & FPAC
	0008	0000			
0045	000A	1339	JEG	EXPF2	; EXP(0)=1
0046	000C	2D20	FMUL	@EXPC0	; F=F*LN 2
	000E	0088			
0047	0010	06A0	BL	@FUNBK	; BREAK TO INTEGER AND FRACTION
	0012	0000			
0048	0014	0281	CI	R1,>7D	; ARGUMENT TOO LARGE?
	0016	007D			
0049	0018	15F3	JGT	ERR33	; Y, ERROR
0050	001A	C081	MOV	R1,R2	; N, SAVE
0051	001C	2CE0	FSUB	@EXPC1	; F=(F*LN 2)-C1
	001E	008E			
0052	0020	2C60		STORE @DS	; STORE IN DS
	0022	0000			
0053	0024	06A0	BL	@PLYXX	; EVALUATE
	0026	0000			
0054	0028	0094		DATA EXPC2	
0055	002A	2D20	FMUL	@DS	; * DS
	002C	0022			
0056	002E	2C60		STORE @DS	; STORE IN DS AGAIN
	0030	002C			
0057	0032	06A0	BL	@PLYX	; EVALUATE
	0034	0000			
0058	0036	00AB		DATA EXPC3	
0059	0038	2C60		STORE @TEMP	; MOVE TO TEMP
	003A	0000			
0060	003C	2CE0	FSUB	@DS	; -DS
	003E	0030			
0061	0040	2C60		STORE @DS1	; SAVE IN DS1
	0042	0000			
0062	0044	2C20	LOADF	@TEMP	; MOVE TEMP TO FPAC
	0046	003A			
0063	0048	2CA0	FADD	@DS	; ADD DS
	004A	003E			
0064	004C	2D60	FDIV	@DS1	; / DS1
	004E	0042			
0065	0050	2D20	FMUL	@EXPC4	; * C4
	0052	00C2			
0066	0054	C042	MOV	R2,R1	; FIX EXPONENT
0067	0056	0921	SRL	R1,2	
0068	0058	06C1	SWPB	R1	
0069	005A	A801	A	R1,@FPAC	; ADD TO EXPONENT
	005C	0000			
0070	005E	C043	MOV	R3,R1	
0071	0060	0242	ANDI	R2,>3	; NEED POWER OF 2?
	0062	0003			
0072	0064	1303	JEG	EXPF1	; N
0073	0066	C0C2	MOV	R2,R3	
0074	0068	06A0	BL	@GETP2	; MULTIPLY BY POWER OF 2
	006A	0000			
0075			*		
0076	006C	C041	EXPF1	MOV R1,R1	; NEGATIVE?
0077	006E	1309	JEG	EXPF3	; N, RETURN
0078	0070	2C60		STORE @TEMP	; Y, MOVE TO TEMP
	0072	0046			
0079	0074	2C20		LOADF @FP1	; GET INVERSE, LOAD FPAC

```
0076 00AA'
0080 007B 2D60      FDIV @TEMP
007A 0072'
0081 007C 1002      JMP  EXPF3          ; DONE
0082                *
0083 007E 2C20  EXPF2  LOADF @FP1      ; EXP(0)=1
0080 00AA'
0084                *
0085 0082 0202  EXPF3  LI   R2,FPAC
0084 005C'
0086 0086 045A      B     #R10
0087                *
0088 0088 4117  EXPC0  DATA >4117,>1547,>652C 1.442695 LOG 2 (E)
0089 008E 4080  EXPC1  DATA >4080,>0000,>0000 1/2
0090 0094 0002  EXPC2  DATA 2
0091 0096 423C      DATA >423C,>9D67,>06A2 60.614853
0092 009C 4476      DATA >4476,>4EF8,>C12A
0093 00A2 461F      DATA >461F,>BE80,>58C1
0094 00AB 0003  EXPC3  DATA 3
0095 00AA 4110  FP1    DATA >4110,>0000,>0000
0096 00B0 436D      DATA >436D,>549A,>5FE1
0097 00B4 4550      DATA >4550,>02D2,>6DCF
0098 00BC 465B      DATA >465B,>9820,>5C39
0099 00C2 4116  EXPC4  DATA >4116,>A09E,>667F 1.4142135
0100                END
NO ERRORS,      NO WARNINGS
```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.FIX
OBJECT ACCESS NAME= ADHOC.OBJ.FIX
LISTING ACCESS NAME= ADHOC.LST.FIX
ERROR ACCESS NAME=
OPTIONS= BUNLST, DUNLST, MUNLST
MACRO LIBRARY PATHNAME=

```
0002          IDT  'FIX'
0003          *
0004          *      FIX      ;FIX 2'S COMPLEMENT INTEGER
0005          *      PFIX     ;FIX 16-BIT POSITIVE INTEGER
0006          *
0007          DXOP LOADF,0          ;LOAD FPAC
0008          DXOP SCALE,6        ;SCALE FPAC
0009          DXOP CLEAR,8        ;CLEAR FPAC
0010          2F80  ERROR  EQU  >2F80      ;XOP XX,14 (ERROR CALL)
0011          *
0012          REF  FPAC2          ;FLOATING POINT ACCUMULATOR
0013          REF  C4600
0014          DEF  FIX
0015          DEF  PFIX
```

```
0017      *      FIX RETURNS THE 2'S COMPLEMENT 16-BIT INTEGER VALUE 0
0018      *      THE 3 WORD NUMBER POINTED TO BY R2.  IF THE NUMBER IS
0019      *      AN INTEGER (1ST WORD = 0 ) THEN IT SIMPLY RETURNS THE
0020      *      IF THE VALUE IS A FLOATING POINT NUMBER (1ST WORD <>
0021      *      NUMBER IS LOADED INTO THE FLOATING POINT ACCUMULATOR,
0022      *      THE DECIMAL POINT IS AFTER THE SECOND WORD, AD THEN T
0023      *      2ND WORD OF FPAC IS RETURNED.  CARE IS TAKE TO ZERO F
0024      *      TO NEGATE THE RESULT DEPENDING UPON THE SIGN BIT.
0025      *
0026      *      PFIX RETURNS A 16-BIT RESULT RATHER THAN THE 2'S
0027      *      COMPLEMENT VALUE.
0028      *
0029      *  CALLING SEQUENCE:
0030      *
0031      *      BL @FIX
0032      *      OR
0033      *      BL @PFIX
0034      *
0035      *      IN - (R2) = 3 WORD PBASIC NUMBER.
0036      *      OUT - R1 = RESULT.
0037      *
0038      *      NORMAL EXIT - RETURN
0039      *
0040      *  EXCEPTIONS AND CONDITIONS:
0041      *
0042      *      ERROR 30 RESULTS IF THE ABSOLUTE VALUE OF THE NUMBER
0043      *      GREATER THAN 2^15
0044      *
0045      *      R0 IS PRESERVED
```

```

0047 0000 C072  FIX      MOV  *R2+,R1          ; INTEGER?
0048 0002 1602          JNE  FIX1              ; N, DO INTF
0049 0004 C052          MOV  *R2,R1            ; Y, RETURN £
0050 0006 045B          RT
0051                *
0052 0008 2C22  FIX1     LOADF @-2(2)           ; LOAD FPAC
0053                *
0054 000C C081  FIX2     MOV  R1,R2              ; SAVE
0055 000E 0241          ANDI R1,>7F80           ; MASK TO EXPONENT + 1 BIT
0056 0012 0281          CI   R1,>4400           ; TOO LARGE?
0057 0014 4400          JGT  ERR30              ; Y
0058 0018 2DA0          SCALE @C4600           ; N, GET INTEGER
0059 001A 0000          MOV  @FPAC2,R1
0060 001C 0060          MOV  @FPAC2,R1
0061 0020 2E00          CLEAR 0                ; LEAVE FPAC ZERO
0062 0022 0A12          SLA  R2,1              ; NEGATIVE?
0063 0024 1701          JNC  FIX3              ; N
0064 0026 0501          NEG  R1                ; Y
0065 0028 045B  FIX3     RT
0066 002A 2F9E  ERR30    DATA ERROR+30        ; FIX ERROR
0067                *
0068                *
0069                *
0070 002C 2C12  PFIX     LOADF *R2             ; LOAD FPAC
0071 002E C052          MOV  *R2,R1            ; INTEGER?
0072 0030 1302          JEQ  PFIX1             ; Y
0073 0032 2DA0          SCALE @C4600           ; N, SCALE
0074 0034 001A'        *
0075 0036 C060  PFIX1     MOV  @FPAC2,R1        ; GET RESULT
0076 0038 001E'        *
0077                RT
0077                END
NO ERRORS,          NO WARNINGS
  
```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.FP
OBJECT ACCESS NAME= ADHOC.OBJ.FP
LISTING ACCESS NAME= ADHOC.LST.FP
ERROR ACCESS NAME=
OPTIONS= BUNLST, DUNLST, MUNLST
MACRO LIBRARY PATHNAME=


```

0002          IDT  'FP'
0003          *
0004          *      FAD          ;ADD TO FPAC
0005          *      FSD          ;SUBTRACT FROM FPAC
0006          *      FMD          ;MULTIPLY FPAC
0007          *      FDD          ;DIVIDE FPAC
0008          *      FLDD         ;LOAD FPAC
0009          *      FSRD         ;STORE FPAC
0010          *      FNEG         ;NEGATE FPAC
0011          *      FCLR         ;CLEAR FPAC
0012          *      FNRM         ;NORMALIZE FPAC
0013          *      FSCL         ;SCALE FPAC
0014          *      FLOAT        ;FLOAT FPAC
0015          *      FADDI        ;3 WRD ADDITION
0016          *      FSUBI        ;3 WRD SUBTRACTION
0017          *
0018          *      THE FLOATING POINT ACCUMULATOR IS THE FIRST 3
0019          *      WORDS OF THE FLOATING POINT REGISTERS (R0,R1,R2).
0020          *
0021          *      ALL FLOATING POINT OPERATIONS ASSUME NORMALIZED
0022          *      NUMBERS AS INPUTS AND ALL RESULTS ARE NORMALIZED.
0023          *
0024          *      THE FORM OF A FLOATING POINT NUMBER IS AS FOLLOWS:
0025          *
0026          *      1ST WORD      SCCC CCCC MMMM MMMM
0027          *      2ND WORD      MMMM MMMM MMMM MMMM
0028          *      3RD WORD      MMMM MMMM MMMM MMMM
0029          *
0030          *      WHERE  S = SIGN BIT
0031          *                  C = 7 BIT, EXCESS >40 CHARACTERISTIC
0032          *                  M = 40 BIT UNSIGNED MAGNITUDE MANTISSA
0033          *                  0 <= M < 1
0034          *
0035          *      A NUMBER IS NORMALIZED WHEN THE 1ST HEX DIGIT
0036          *      OF THE MANTISSA IS NON-ZERO.
0037          *
0038          *      A TRUE ZERO (ALL ZERO'S) IS USED FOR ZERO.
0039          *
0040          2F80  ERROR  EQU  >2F80          ;XOP XX,14 (ERROR CALL)
0041          2FA0  ERROR2 EQU  ERROR+>20
0042          *
0043          DEF  FSRD
0044          DEF  FLDD
0045          DEF  FSD          ;SUBTRACT FROM FPAC
0046          DEF  FAD
0047          DEF  FNEG
0048          DEF  FLOAT
0049          DEF  FNRM
0050          *
0051          REF  C7F,CF0,CFF80
    
```

```
0053      *
0054      *          LOAD INTO FPAC A 3 WRD FLOATING POINT £.
0055      *
0056      * CALLING SEQUENCE:
0057      *
0058      *          XOP XX,1
0059      *
0060 0000 CECO  FSRD  MDV  R0,*R11+
0061 0002 CEC1      MDV  R1,*R11+
0062 0004 C6C2      MDV  R2,*R11
0063 0006 0380      RTWP
```

```
0065      *
0066      *      LOAD FLOATING POINT ACCUMULATOR WITH 3RD FLOATING
0067      *      POINT NUMBER.
0068      *
0069      * CALL SEQUENCE:
0070      *
0071      *      XDP XX,0
0072      *
0073 0008 C03B FLDD  MOV  *R11+,R0      ; LOAD FPAC
0074 000A C07B FLDD1 MOV  *R11+,R1
0075 000C C09B      MOV  *R11,R2
0076 000E 0380 FLDD2 RTWP
```

```

0078      *
0079      *      ADD OR SUBTRACT FROM FPAC A 3 WORD FLOATING
0080      *      POINT NUMBER.
0081      *
0082      * CALL SEQUENCE:
0083      *
0084      *      XOP XX,2      ;ADD TO FPAC
0085      *
0086      *      XOP XX,3      ;SUBTRACT FROM FPAC
0087      *
0088      * EXCEPTIONS AND CONDITIONS:
0089      *
0090      *      ERROR 29 CAN RESULT FROM OVERFLOW
0091      *
0092      * EXTERNAL ROUTINE LIST:
0093      *
0094      *      FLDD      ;LOAD FPAC
0095      *      FARS      ;SHIFT FPAC RIGHT 1 HEX DIGIT
0096      *      FADDI     ;ADD R4,R5,R6 TO R0,R1,R2
0097      *
0098      *
0099      *      SUBTRACTION IS MADE SIMPLY BY TOGGLING SIGN BIT
0100      *
0101 0010 C13B FSD      MOV  *R11+,R4      ;GET FIRST WORD, ZERO?
0102 0012 13FD      JEQ  FLDD2      ;Y, ZERO, RETURN
0103 0014 0224      AI   R4,>8000      ;N, TOGGLE SIGN BIT
0104 0016 8000
0104 0018 1002      JMP  FAD0
0105      *
0106      *
0107 001A C13B FAD      MOV  *R11+,R4      ;GET FIRST WORD, ZERO?
0108 001C 13F8      JEQ  FLDD2      ;Y, RETURN
0109 001E C000 FAD0     MOV  R0,R0      ;N, FPAC ZERO?
0110 0020 1602      JNE  FAD0A      ;N
0111 0022 C004      MOV  R4,R0      ;Y, MOVE TEMP TO FPAC
0112 0024 10F2      JMP  FLDD1
0113      *
0114 0026 C17B FAD0A    MOV  *R11+,R5      ;LOAD R4,R5,R6
0115 0028 C19B      MOV  *R11,R6
0116 002A D0C0      MOVB R0,R3      GET EXPONENTS
0117 002C D1C4      MOVB R4,R7
0118 002E 7000      SB   R0,R0      ISOLATE
0119 0030 7104      SB   R4,R4
0120 0032 0A13      SLA  R3,1      REMOVE SIGN
0121 0034 1702      JNC  FAD1      NEGATIVE?
0122 0036 06A0      BL   @FADN      Y, NEGATE
0123 0038 00B8      *
0123      *
0124 003A 0993 FAD1     SRL  R3,9
0125 003C 0A17      SLA  R7,1      REMOVE SIGN
0126 003E 1708      JNC  FAD2      NEGATIVE?
0127 0040 0506      NEG  R6      Y, NEGATE
0128 0042 1604      JNE  FAD1A
0129 0044 0505      NEG  R5
0130 0046 1603      JNE  FAD1B
0131 0048 0504      NEG  R4
0132 004A 1002      JMP  FAD2
0133 004C 0545 FAD1A    INV  R5
0134 004E 0544 FAD1B    INV  R4
0135      *
    
```

0136	0050	020A	FAD2	LI	R10,10	10 SHIFTS GIVE ZERO RESULT
	0052	000A				
0137	0054	0997		SRL	R7,9	POSITION EXPONENT
0138			*			
0139	0056	81C3	FAD2A	C	R3,R7	COMPARE EXPONENTS
0140	0058	131D		JEG	FAD4	SAME, DO ADD
0141	005A	1509		JGT	FAD3	SHIFT TEMP (IS R3>R7?)
0142	005C	06A0		BL	@FARS	N, R3<R7, SHIFT FPAC
	005E	02D2				
0143	0060	060A		DEC	R10	COUNT SHIFTS
0144	0062	16F9		JNE	FAD2A	OK
0145	0064	C004		MOV	R4,R0	FPAC=0, MOVE TEMP TO FPAC
0146	0066	C045		MOV	R5,R1	
0147	0068	C086		MOV	R6,R2	
0148	006A	C0C7		MOV	R7,R3	MOVE EXPONENT
0149	006C	100C		JMP	FAD3A	
0150			*			
0151	006E	0946	FAD3	SRL	R6,4	SHIFT TEMP 1 HEX DIGIT RIGHT
0152	0070	C245		MOV	R5,R9	
0153	0072	0AC9		SLA	R9,12	
0154	0074	A189		A	R9,R6	MOVE HEX DIGIT ACROSS
0155	0076	0945		SRL	R5,4	
0156	0078	C244		MOV	R4,R9	
0157	007A	0AC9		SLA	R9,12	
0158	007C	A149		A	R9,R5	MOVE HEX DIGIT ACROSS
0159	007E	0844		SRA	R4,4	
0160	0080	0587		INC	R7	DONE, INCREMENT EXPONENT
0161	0082	060A		DEC	R10	COUNT SHIFTS
0162	0084	16E8		JNE	FAD2A	OK
0163			*			
0164	0086	C000	FAD3A	MOV	R0,R0	TEMP=0, NEED NEGATION?
0165	0088	1553		JGT	FNRM3	N, FPAC OK
0166	008A	06A0		BL	@FADN	Y, NEGATE FPAC
	008C	00B8				
0167	008E	0223		AI	R3,>80	ADD SIGN TO EXPONENT
	0090	0080				
0168	0092	104E		JMP	FNRM3	
0169			*			
0170	0094	06A0	FAD4	BL	@FADDI	ADD
	0096	02F2				
0171	0098	1106		JLT	FAD5	NEGATIVE
0172	009A	1509		JGT	FAD6	POSITIVE
0173	009C	C041		MOV	R1,R1	
0174	009E	1607		JNE	FAD6	NON-ZERO
0175	00A0	C082		MOV	R2,R2	
0176	00A2	1605		JNE	FAD6	NON-ZERO
0177	00A4	1047		JMP	FNRM4	ZERO, RTWP
0178			*			
0179	00A6	06A0	FAD5	BL	@FADN	NEGATE FPAC
	00A8	00B8				
0180	00AA	0263		ORI	R3,>80	ADD SIGN BIT
	00AC	0080				
0181			*			
0182	00AE	D000	FAD6	MOVB	R0,R0	CHECK ADDITION OVERFLOW
0183	00B0	132E		JEG	FNRM2	NUMBER OK
0184	00B2	06A0		BL	@FARS	SHIFT NUMBER
	00B4	02D2				
0185	00B6	102B		JMP	FNRM2	NORMALIZE
0186			*			
0187	00B8	0502	FADN	NEG	R2	NEGATE FPAC

0188	00BA	1604	JNE	FADN1	
0189	00BC	0501	NEG	R1	
0190	00BE	1603	JNE	FADN2	
0191	00C0	0500	NEG	RO	
0192	00C2	045B	RT		
0193		*			
0194	00C4	0541	FADN1	INV	R1
0195	00C6	0540	FADN2	INV	RO
0196	00CB	045B	B	*R11	RETURN

```

0198      *
0199      *      GET ABSOLUTE VALUE OF FPAC.
0200      *      NEGATE FPAC.
0201      *
0202      * CALL SEQUENCE:
0203      *
0204      *      XOP XX,9      ;NEGATE FPAC
0205      *
0206      *      DEF FABS
0207      *
0208      *FABS  ANDI R0,>7FFF  CLEAR SIGN BIT
0209      *      RTWP          RETURN
0210 00CA C000  FNEG  MOV  R0,R0      ZERO?
0211 00CC 1302      JEQ  FNEG1      Y, LEAVE TRUE ZERO
0212 00CE 0220      AI   R0,>8000    N, COMPLEMENT SIGN BIT
      00D0 8000
0213 00D2 0380  FNEG1 RTWP RETURN
    
```

```

0215      *
0216      *      IF FPAC IS NOT NORMALIZED, AN ATTEMPT WILL BE
0217      *      MADE TO NORMALIZE.  IN GENERAL, FLOAT WORKS ON
0218      *      R0,R1,R2 AND HENCE IT CAN BE USED TO FLOAT
0219      *      OTHER SETS OF NUMBERS THAN FPAC.
0220      *
0221      * CALLING SEQUENCE:
0222      *
0223      *      XOP XX,10      WHERE XX IS NOT USED
0224      *
0225      * EXCEPTIONS AND CONDITIONS:
0226      *
0227      *      IF R0 IS NON-ZERO, THE ROUTINE WILL ABORT.
0228      *
0229      * EXTERNAL ROUTING LIST:
0230      *
0231      *      FNRM      ;NORMALIZE
0232      *      FCLR      ;CLEAR FPAC
0233      *
0234 00D4 C000  FLOAT  MOV  R0,R0      ALREADY FLOATING?
0235 00D6 162E      JNE  FNRM4      Y, RTWP
0236 00D8 0203      LI   R3,>44     N, GET EXPONENT
0237      00DA 0044
0237 00DC 04C2      CLR  R2          CLEAR LOW
0238 00DE C041      MOV  R1,R1      CHECK SIGN
0239 00E0 1504      JGT  FLOAT1     POSITIVE
0240 00E2 132C      JEQ  FCLRP     ZERO
0241 00E4 0501      NEG  R1         NEGATIVE
0242 00E6 0203      LI   R3,>C4     NEGATE EXPONENT
0243      00E8 00C4
0243      *
0244 00EA D001  FLOAT1 MOVB R1,R0     SHIFT 1 BYTE LEFT
0245 00EC 1303      JEQ  FLOAT2     SHIFT 2 BYTES
0246 00EE 06C0      SWPB R0
0247 00F0 0AB1      SLA  R1,8
0248 00F2 100D      JMP  FNRM2
0249      *
0250 00F4 C001  FLOAT2 MOV  R1,R0     SHIFT 2 BYTES
0251 00F6 04C1      CLR  R1
0252 00F8 0643      DECT R3         ADJUST EXPONENT
0253 00FA 1009      JMP  FNRM2
  
```



```

0255      *
0256      *      FPAC IS NORMALIZED SUCH THAT THERE EXISTS
0257      *      A NON-ZERO HEX DIGIT IN BITS 8-11 OF THE 1ST
0258      *      WORD OF FPAC.
0259      *
0260      * CALL SEQUENCE:
0261      *
0262      *      XOP XX,7      WHERE XX IS NOT USED
0263      *
0264      * EXTERNAL ROUTINE LIST:
0265      *
0266      *      FCLR          ; CLEAR FPAC
0267 00FC 04C3  FNRM  CLR  R3      CLEAR R3
0268 00FE D0C0      MOVB R0,R3    GET EXPONENT & SIGN
0269 0100 6003      S    R3,R0    REMOVE FROM NUMBER
0270 0102 1604      JNE  FNRM1    LOOK FOR ZERO
0271 0104 C041      MOV  R1,R1
0272 0106 1602      JNE  FNRM1
0273 0108 C082      MOV  R2,R2
0274 010A 1314      JEQ  FNRM4    ZERO, RETURN
0275      *
0276 010C 06C3  FNRM1 SWPB R3      READY FOR DECREMENTING
0277      *
0278 010E 2420  FNRM2 CZC  @CF0,R0  NORMALIZED?
      0110 0000
0279 0112 160E      JNE  FNRM3      Y
0280 0114 24E0      CZC  @C7F,R3    N, EXPONENT=ZERO?
      0116 0000
0281 0118 1311      JEQ  FCLRP      Y, CANNOT NORMALIZE
0282 011A 0603      DEC  R3          N, OK TO DECREMENT
0283 011C 0A40      SLA  R0,4        SHIFT R0,R1,R2 4 BITS LEFT
0284 011E C241      MOV  R1,R9
0285 0120 09C9      SRL  R9,12
0286 0122 A009      A    R9,R0      MOVE 1ST HEX DIGIT ACROSS
0287 0124 0A41      SLA  R1,4
0288 0126 C242      MOV  R2,R9
0289 0128 09C9      SRL  R9,12
0290 012A A049      A    R9,R1      MOVE 2ND HEX DIGIT ACROSS
0291 012C 0A42      SLA  R2,4
0292 012E 10EF      JMP  FNRM2
0293      *
0294 0130 06C3  FNRM3 SWPB R3      READY EXPONENT
0295 0132 D003      MOVB R3,R0      MOVE INTO FPAC
0296 0134 0380  FNRM4 RTWP        RETURN
  
```

```

0298      *
0299      *      MULTIPLY FPAC BY FLOATING POINT NUMBER
0300      *      POINTED TO BY R11.
0301      *
0302      * CALL SEQUENCE:
0303      *
0304      *      XOP XX,4
0305      *
0306      * EXCEPTIONS AND CONDITIONS:
0307      *
0308      *      ERROR 29 CAN RESULT FROM OVERFLOW
0309      *
0310      * EXTERNAL ROUTINE LIST:
0311      *
0312      *      FCLR          ; CLEAR FPAC
0313      *      FPFX          ; FIX EXPONENTS
0314      *      FAD6          ; ADD AND NORMALIZE
0315      *
0316      * ENTRY POINT:
0317      *
0318      *      DEF    FMD
0319      *
0320 0136 C000  FMD      MOV    R0,R0          ; FPAC=0?
0321 0138 1374      JEQ    FCLR          ; Y
0322 013A C13B      MOV    *R11+,R4        ; FPAC X 0?
0323 013C 1372  FCLRP  JEQ    FCLR          ; Y
0324 013E C17B      MOV    *R11+,R5        ; N, LOAD R4,R5,R6
0325 0140 C19B      MOV    *R11,R6
0326 0142 06A0      BL     @FPFX          FIX EXPONENTS
0326 0144 022A'
0327 0146 A0C7      A      R7,R3
0328 0148 FFC0      DATA >FFC0
0329      *
0330      *DO MULTIPLICATION
0331      *
0332      *      R4,R5,R6
0333      *      R0,R1,R2
0334      *      =====
0335      *      XX XX          A      R2 X R6
0336      *      XX XX          B      R2 X R5
0337      *      XX XX          C      R2 X R4
0338      *      XX XX          D      R1 X R6
0339      *      XX XX          E      R1 X R5
0340      *      XX XX          F      R1 X R4
0341      *      XX XX          G      R0 X R6
0342      *      XX XX          H      R0 X R5
0343      *      XX XX          I      R0 X R4
0344      *      -----
0345      *      R7,R8,R9,R10
0346      *
0347 014A C286      MOV    R6,R10
0348 014C 3A82      MPY    R2,R10          R10,R11 (A)
0349 014E C206      MOV    R6,R8
0350 0150 3A01      MPY    R1,R8          R8,R9 (D)
0351 0152 3980      MPY    R0,R6          R6,R7 (G)
0352      *
0353 0154 A289      A      R9,R10          SUM PARTIAL PRODUCTS (A,D,G)
0354 0156 1701      JNC    $+4
0355 0158 0588      INC    R8
0356 015A A207      A      R7,R8
    
```

0357 015C 1701	JNC	\$+4	
0358 015E 0586	INC	R6	
0359 0160 C248	MOV	R8, R9	
0360 0162 C206	MOV	R6, R8	R8, R9, R10 = A+D+G
0361	*		
0362 0164 C185	MOV	R5, R6	
0363 0166 3982	MPY	R2, R6	R6, R7 (B)
0364 0168 A287	A	R7, R10	SUM
0365 016A 1701	JNC	\$+4	
0366 016C 0586	INC	R6	
0367 016E A246	A	R6, R9	
0368 0170 1701	JNC	\$+4	
0369 0172 0588	INC	R8	(NO CARRY OUT)
0370 0174 C289	MOV	R9, R10	
0371 0176 C248	MOV	R8, R9	
0372 0178 C1C5	MOV	R5, R7	
0373 017A 39C1	MPY	R1, R7	R7, R8 (E)
0374 017C 3940	MPY	R0, R5	R5, R6 (H)
0375	*		
0376 017E A288	A	R8, R10	FINISH PARTIAL SUMS B, E, H
0377 0180 04C8	CLR	R8	
0378 0182 1701	JNC	\$+4	
0379 0184 0587	INC	R7	
0380 0186 A247	A	R7, R9	
0381 0188 1701	JNC	\$+4	
0382 018A 0588	INC	R8	
0383 018C A246	A	R6, R9	
0384 018E 1701	JNC	\$+4	
0385 0190 0588	INC	R8	
0386 0192 04C7	CLR	R7	
0387 0194 A205	A	R5, R8	
0388 0196 1701	JNC	\$+4	
0389 0198 0587	INC	R7	R7, R8, R9, R10 = A+B+D+E+G+H
0390 019A C144	MOV	R4, R5	
0391 019C 3942	MPY	R2, R5	R5, R6 (C)
0392 019E A286	A	R6, R10	SUM
0393 01A0 1705	JNC	FMD4	
0394 01A2 0589	INC	R9	
0395 01A4 1703	JNC	FMD4	
0396 01A6 0588	INC	R8	
0397 01A8 1701	JNC	FMD4	
0398 01AA 0587	INC	R7	
0399 01AC A245	FMD4 A	R5, R9	
0400 01AE 1703	JNC	FMD5	
0401 01B0 0588	INC	R8	
0402 01B2 1701	JNC	FMD5	
0403 01B4 0587	INC	R7	
0404	*		
0405 01B6 C144	FMD5 MOV	R4, R5	
0406 01B8 3941	MPY	R1, R5	R5, R6 (F)
0407 01BA A246	A	R6, R9	SUM
0408 01BC 1703	JNC	FMD6	
0409 01BE 0588	INC	R8	
0410 01C0 1701	JNC	FMD6	
0411 01C2 0587	INC	R7	
0412 01C4 A205	FMD6 A	R5, R8	
0413 01C6 1701	JNC	\$+4	
0414 01C8 0587	INC	R7	
0415 01CA 3900	MPY	R0, R4	R4, R5 (I)
0416 01CC A205	A	R5, R8	

0417 01CE 1701
0418 01D0 0587
0419 01D2 A1C4
0420 01D4 D1C8
0421 01D6 06C7
0422 01D8 D209
0423 01DA 06C8
0424 01DC D24A
0425 01DE 06C9
0426 01EO 06CA

JNC \$+4
INC R7
A R4,R7
MOVB R8,R7
SWPB R7
MOVB R9,R8
SWPB R8
MOVB R10,R9
SWPB R9
SWPB R10

PRODUCT COMPLETE
SHIFT 1 BYTE LEFT

```

0428      *MULTIPLY AND DIVIDE ENTRY
0429      *
0430      *      R10 HAS GUARD BITS FOR NORMALIZING & ROUNDING
0431      *
0432 01E2 C007 FMD7  MOV  R7,R0      MOVE INTO FPAC
0433 01E4 C048      MOV  R8,R1
0434 01E6 C089      MOV  R9,R2
0435      *
0436 01E8 2420      CZC  @CF0,R0      NORMALIZED?
      01EA 0110'
0437 01EC 1611      JNE  FMD9      Y, ROUND
0438 01EE 24E0      CZC  @C7F,R3      N, EXP=0?
      01F0 0116'
0439 01F2 1317      JEQ  FCLR      Y, CANNOT NORMALIZE
0440 01F4 0603      DEC  R3      N, OK TO DECREMENT
0441 01F6 0A40      SLA  R0,4      SHIFT R0,R1,R2,R10 LEFT 4 BITS
0442 01F8 C241      MOV  R1,R9
0443 01FA 09C9      SRL  R9,12
0444 01FC A009      A    R9,R0      MOVE 1ST HEX DIGIT ACROSS
0445 01FE 0A41      SLA  R1,4
0446 0200 C242      MOV  R2,R9
0447 0202 09C9      SRL  R9,12
0448 0204 A049      A    R9,R1      MOVE 2ND HEX DIGIT ACROSS
0449 0206 0A42      SLA  R2,4
0450 0208 C24A      MOV  R10,R9
0451 020A 09C9      SRL  R9,12
0452 020C A089      A    R9,R2      MOVE 3RD HEX DIGIT ACROSS
0453 020E 0A4A      SLA  R10,4
0454      *
0455      *DONE, ROUND USING R10
0456      *
0457 0210 0A1A FMD9  SLA  R10,1      GET GUARD BIT
0458 0212 178E      JNC  FNRM3      IF ZERO, DONE
0459 0214 0582      INC  R2      ELSE ROUND
0460 0216 178C      JNC  FNRM3
0461 0218 0581      INC  R1
0462 021A 178A      JNC  FNRM3
0463 021C 0580      INC  R0
0464 021E 0460      B    @FAD6      IF THIS FAR, POSSIBILITY
      0220 00AE'      OF NORMALIZATION
    
```

```
0466      *
0467      *          SET FPAC TO TRUE ZERO.
0468      *
0469      * CALLING SEQUENCE:
0470      *
0471      *          XOP XX,B          WHERE XX IS NOT USED
0472      *
0473      * ENTRY POINT:
0474      *
0475      *          DEF  FCLR
0476      *
0477 0222 04C0  FCLR  CLR  R0          CLEAR FPAC
0478 0224 04C1          CLR  R1
0479 0226 04C2          CLR  R2
0480 0228 0380          RTWP
```

```
0482      *
0483      *      DIVIDE FPAC BY THE FLOATING POINT NUMBER POINTED
0484      *      TO BY R11.
0485      *
0486      *      FPFx REMOVES EXPONENTS FROM R0,R1,R2 AND
0487      *      R4,R5,R6, UNBIASES THEM, PERFORMS MULTIPLICATION
0488      *      OR DIVISION (ADD OR SUBTRACT) AND THEN
0489      *      RE-BIASES THEM.
0490      *
0491      * CALL SEQUENCE:
0492      *
0493      *      XDP XX,5
0494      *      BL @FPFX
0495      *
0496      * EXCEPTIONS AND CONDITIONS:
0497      *
0498      *      ERROR 28 WILL RESULT FROM DIVISION BY ZERO
0499      *      ERROR 29 WILL RESULT FROM OVERFLOW
0500      *
0501      * EXTERNAL ROUTINE LIST:
0502      *
0503      *      FCLR      ; CLEAR
0504      *      FMD7      ; FINISH BY NORMALIZING
```

```

0506      *FIX EXPONENTS
0507      *
0508      *      BL @FPFX
0509      *      BIAS INSTRUCTION
0510      *      REBIAS £
0511      *
0512 022A D0C0  FPFX  MOV B R0,R3      GET EXPONENTS
0513 022C D1C4      MOV B R4,R7
0514 022E 7000      SB   R0,R0      ; REMOVE FROM NUMBERS
0515 0230 7104      SB   R4,R4
0516 0232 C207      MOV  R7,R8      GET SIGN
0517 0234 2A03      XOR  R3,R8      +-, -+ = NEG  ++, -- = POS
0518 0236 06C3      SWPB R3      REMOV SIGN BITS
0519 0238 0243      ANDI R3,>7F
0519 023A 007F
0520 023C 06C7      SWPB R7
0521 023E 0247      ANDI R7,>7F
0521 0240 007F
0522 0242 04BB      X      *R11+      DO ADD FOR MUL, SUB FOR DIV
0523 0244 A0FB      A      *R11+,R3    BIAS
0524 0246 24E0      CZC  @CFF80,R3    CHECK FOR OVERFLOW
0524 0248 0000
0525 024A 1642      JNE  ERR29      PROBLEM
0526 024C 0A18      SLA  R8,1      ADD SIGN
0527 024E 1702      JNC  $+6
0528 0250 0223      AI   R3,>80      NEGATIVE
0528 0252 0080
0529 0254 045B      RT      RETURN
0530      *
0531      *FLOATING POINT DIVIDE
0532      *
0533      * ENTRY POINT:
0534      *
0535      DEF  FDD
0536      *
0537 0256 C000  FDD  MOV  R0,R0      ; FPAC=ZERO?
0538 0258 13E4      JEQ  FCLR      ; Y
0539 025A C13B      MOV  *R11+,R4    ; FPAC / 0?
0540 025C 133B      JEQ  FDDER      ; Y, DIVISION BY ZERO
0541 025E C17B      MOV  *R11+,R5    ; N, LOAD R4,R5,R6
0542 0260 C19B      MOV  *R11,R6
0543 0262 06A0      BL   @FPFX      FIX EXPONENTS
0543 0264 022A'
0544 0266 60C7      S     R7,R3      SUBTRACT EXPONENTS
0545 0268 0040      DATA >40      ADD >40 TO BIAS
0546 026A 8100      C     R0,R4      CHECK FOR PROPER FRACTION
0547 026C 110A      JLT  FDD2      OK
0548 026E 0583      INC  R3      IMPROPER, INCREMENT EXPONENT
0549 0270 0A44      SLA  R4,4      SHIFT R4,R5,R6 LEFT 4 BITS
0550 0272 C245      MOV  R5,R9
0551 0274 09C9      SRL  R9,12
0552 0276 A109      A     R9,R4      MOVE 1ST HEX DIGIT ACROSS
0553 0278 0A45      SLA  R5,4
0554 027A C246      MOV  R6,R9
0555 027C 09C9      SRL  R9,12
0556 027E A149      A     R9,R5      MOVE 2ND HEX DIGIT ACROSS
0557 0280 0A46      SLA  R6,4
0558      *
0559 0282 04C7  FDD2  CLR  R7      Y, CLEAR QUOTIENT
0560 0284 04C8      CLR  R8
    
```


0561	0286	04C9		CLR	R9	
0562	0288	020A		LI	R10, 40	DO 40 TIMES
	028A	0028				
0563			*			
0564	028C	0A10	FDD3	SLA	R0, 1	SHIFT LEFT R0, R1, R2, R7, R8, R9
0565	028E	0A11		SLA	R1, 1	
0566	0290	1701		JNC	\$+4	
0567	0292	0580		INC	R0	
0568	0294	0A12		SLA	R2, 1	
0569	0296	1701		JNC	\$+4	
0570	0298	0581		INC	R1	
0571	029A	0A17		SLA	R7, 1	
0572	029C	0A18		SLA	R8, 1	
0573	029E	1701		JNC	\$+4	
0574	02A0	0587		INC	R7	
0575	02A2	0A19		SLA	R9, 1	
0576	02A4	1701		JNC	\$+4	
0577	02A6	0588		INC	R8	
0578	02A8	8100		C	R0, R4	IS R0, R1, R2<=R4, R5, R6?
0579	02AA	1A09		JL	FDD5	Y
0580	02AC	1B05		JH	FDD4	N
0581	02AE	8141		C	R1, R5	MAYBE
0582	02B0	1A06		JL	FDD5	Y
0583	02B2	1B02		JH	FDD4	N
0584	02B4	8182		C	R2, R6	MAYBE
0585	02B6	1A03		JL	FDD5	Y
0586	02B8	06A0	FDD4	BL	@FSUBI	N, R0, R1, R2=R0, R1, R2-R4, R5, R6
	02BA	0306				
0587	02BC	0589		INC	R9	ENTER BIT
0588			*			
0589	02BE	060A	FDD5	DEC	R10	DONE?
0590	02C0	16E5		JNE	FDD3	N, LOOP AGAIN
0591	02C2	0A10		SLA	R0, 1	Y, CHECK FOR ROUNDING
0592	02C4	8100		C	R0, R4	2*REMAINDER < DIVISOR?
0593	02C6	1ABD		JL	FMD7	Y, NO NEED TO ROUND
0594	02C8	020A		LI	R10, >8000	ROUND
	02CA	8000				
0595	02CC	108A		JMP	FMD7	DONE
0596			*			
0597	02CE	2F9C	FDDER	DATA	ERROR+28	; DIVISION BY ZERO
0598			*			
0599	02D0	2F9D	ERR29	DATA	ERROR+29	; FP ERROR

```

0601      *
0602      *      SHIFT R0,R1,R2 RIGHT 1 HEX DIGIT WHILE
0603      *      UPDATEING EXPONENT IN R3
0604      *
0605      * CALLING SEQUENCE:
0606      *
0607      *      BL @FARS
0608      *
0609      * EXCEPTIONS AND CONDITIONS:
0610      *
0611      *      ERROR 29 WILL RESULT ON OVERFLOW
0612      *
0613      * EXTERNAL ROUTINE LIST:
0614      *
0615      *      ERR29          ; ERROR 29
0616      *
0617 02D2 0942      FARS      SRL   R2,4          SHIFT R0,R1,R2 4 BITS RIGHT
0618 02D4 C241      MOV    R1,R9
0619 02D6 0AC9      SLA    R9,12
0620 02DB A0B9      A      R9,R2          MOVE 1ST HEX DIGIT ACROSS
0621 02DA 0941      SRL   R1,4
0622 02DC C240      MOV    R0,R9
0623 02DE 0AC9      SLA    R9,12
0624 02E0 A049      A      R9,R1          MOVE 2ND HEX DIGIT ACROSS
0625 02E2 0B40      SRA    R0,4
0626 02E4 C243      MOV    R3,R9
0627 02E6 05B3      INC    R3          INCREMENT EXPONENT
0628 02E8 2A43      XOR    R3,R9          WATCH FOR SIGN CHANGE
0629 02EA 0249      ANDI   R9,>80        SIGN CHANGE?
0630      02EC 0080
0631 02EE 16F0      JNE    ERR29          Y, OVERFLOW
0631 02F0 045B      RT

```

```

0633      *
0634      *      ADD OR SUBTRACT R4,R5,R6 FROM R0,R1,R2.
0635      *
0636      * CALL SEQUENCE:
0637      *
0638      *      BL @FADDI
0639      *      BL @FSUBI
0640      *
0641      *      ENTRY POINT :
0642      *
0643      *      DEF  FADDI
0644      *
0645 02F2 A086  FADDI  A    R6,R2      R0,R1,R2=R0,R1,R2+R4,R5,R6
0646 02F4 1703      JNC  $+8
0647 02F6 0581      INC  R1
0648 02F8 1701      JNC  $+4
0649 02FA 0580      INC  R0
0650 02FC A045      A    R5,R1
0651 02FE 1701      JNC  $+4
0652 0300 0580      INC  R0
0653 0302 A004      A    R4,R0      ADDITION COMPLETE
0654 0304 045B      RT
0655      *
0656      * ENTRY POINT:
0657      *
0658      *      DEF  FSUBI
0659      *
0660 0306 6086  FSUBI  S    R6,R2      R0,R1,R2=R0,R1,R2-R4,R5,R6
0661 0308 1803      JOC  $+8
0662 030A 0601      DEC  R1
0663 030C 1801      JOC  $+4
0664 030E 0600      DEC  R0
0665 0310 6045      S    R5,R1
0666 0312 1801      JOC  $+4
0667 0314 0600      DEC  R0
0668 0316 6004      S    R4,R0      SUBTRACTION COMPLETE
0669 0318 045B      RT
  
```

```

0671      *
0672      *
0673      *      FPAC WILL BE SHIFTED LEFT OR RIGHT IN
0674      *      ORDER TO MAKE THE EXPONENT AGREE WITH THE
0675      *      SCALE FACTOR POINTED TO BY R11. THIS
0676      *      OPERATION IS ESSENTIALLY THE OPPOSITE
0677      *      OF NORMALIZATION.
0678      *      SCALING TO >4600 PLACES THE DECIMAL POINT
0679      *      BETWEEN THE SECOND AND THIRD WORD OF THE FLOATING
0680      *      POINT NUMBER. SCALING TO >4A00 PLACES THE DECIMAL
0681      *      POINT AFTER THE THIRD WORD.
0682      *      HENCE, TO INTEGERIZE A FLOATING POINT NUMBER IN FPAC,
0683      *      ONE SCALES TO >4A00 AND THEN NORMALIZES.
0684      *
0685      * CALLING SEQUENCE:
0686      *
0687      *      'XOP XX,6
0688      *
0689      * EXCEPTIONS AND CONDITIONS:
0690      *
0691      *      ERROR 29 WILL RESULT FROM A LEFT SHIFT
0692      *
0693      * EXTERNAL ROUTINE LIST:
0694      *
0695      *      ERR29          ;ERROR 29
0696      *
0697      * ENTRY POINT:
0698      *
0699      *      DEF  FSCL
0700 031A C11B  FSCL  MOV  *R11,R4      GET SCALE FACTOR
0701 031C 04C3      CLR  R3
0702 031E C200      MOV  R0,R8      SAVE SIGN
0703 0320 D1C8      MOVB R8,R7      GET EXPONENT
0704 0322 0247      ANDI R7,>7F00   GET EXPONENT
0705      0324 7F00
0706 0326 61C4      S      R4,R7      GET DIFFERENCE
0707 0328 130B      JEQ  FSCL2      ALREADY SCALED
0708 032A D003      MOVB R3,R0      ZERO EXPONENT
0709 032C 0887      SRA  R7,8      RIGHT JUSTIFY
0710 032E 15D0      JGT  ERR29      SHIFT LEFT, FP ERROR?
0711      *
0712 0330 06A0  FSCL1 BL  @FARS      SHIFT RIGHT
0713      0332 02D2
0714 0334 0587      INC  R7          DONE?
0715 0336 16FC      JNE  FSCL1      N
0716 0338 D008      MOVB R8,R0      RESTORE SIGN BIT
0717 033A 0240      ANDI R0,>80FF   ;MASK EXPONENT
0718      033C 80FF
0719 033E B004      AB   R4,R0
0720 0340 0380  FSCL2 RTWP
0721      *
0722      END
  
```

NO ERRORS,

NO WARNINGS

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.BITF
OBJECT ACCESS NAME= ADHOC.OBJ.BITF
LISTING ACCESS NAME= ADHOC.LST.BITF
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

0002		IDT 'BITF'	
0003		*	
0004		*	
0005		* ROUTINE LIST:	
0006		*	
0007		* BITF	BIT FUNCTION
0008		* BITY	BIT COMMAND
0009		*	
0010		REF EVSFR\$	RELOAD R2 & EXIT TO EVALUATOR
0011		REF NLIN	EXIT TO MULTIPLEXOR
0012		REF EVERZ	EVALUATE EXPRESSION
0013		REF GPRM1	GET-PARAMETER ENTRY
0014		REF FPAC2	FLOATING POINT ACCUMULATOR
0015		REF B4A	
0016		DXOP STORE,1	STORE FPAC
0017		DXOP EVFIX,11	EVALUATE AND FIX
0018		DXOP OUTFP,12	OUT FLOATING POINT &
0019	2F80 ERROR	EQU >2F80	XOP XX,14 (ERROR CALL)
0020		DEF BITY	ENTRY POINT FOR BIT STATEMENT
0021		DEF BITF	ENTRY POINT FOR BIT FUNCTION
0022		DEF CB000	

```

0024      *      THE BIT STATEMENT ALLOWS A BASIC PROGRAM
0025      *      TO ALTER ANY BIT WITHIN A BASIC
0026      *      VARIABLE. THE FORM IS:
0027      *
0028      *      BIT[ <VAR> , <EXP1> ] = <EXP2>
0029      *
0030      *      WHERE: <VAR> = VARIABLE TO BE ALTERED
0031      *      <EXP1> = BIT POSITION WITHIN VARIABLE
0032      *      <EXP2> = 0 OR <>0 TO RESET OR SET
0033      *      BIT RANGES FROM 0 UPWARDS (IF -VE ASSUMES BIT 0)
0034      *
0035      *      CALLING SEQUENCE:
0036      *
0037      *      B @BITY
0038      *
0039      *      EXIT TO NLIN
0040      *
0041      *      EXCEPTIONS AND CONDITIONS:
0042      *
0043      *      EVALUATION ERRORS
0044      *      ERROR 1 IF BRACKETS MISSING OR NO ' , '
0045      *
0046 0000 9838 BITY CB *R8+, @B4A ; RIGHT BRACKET?
0047      0002 0000 ; N, ERROR
0048 0004 1616 JNE ERR1 ; GET ADDRESS OF VARIABLE
0049 0006 0420 BLWP @EVERZ ; ' , ' FOLLOWING ?
0050      0008 0000 ; NO, ERROR IT
0051 000A 0280 CI R0, >3F00 ; GET PARAMETERS
0052      000C 3F00 ; TEST SIGN OF BIT &
0053 000E 1611 JNE ERR1 ; +VE, LEAVE ALONE
0054 0010 06A0 BL @GPRM1 ; -VE, MAKE 0
0055 0012 0000 ; GET WORD INDEX
0056 0014 C041 MOV R1, R1 ; INDEX
0057 0016 1501 JGT BITY0 ; SET MASK
0058 0018 04C1 CLR R1 ; POSITION MASK
0059 001A C001 BITY0 MOV R1, R0 ; SET BIT TO ZERO
0060 001C 0931 SRL R1, 3 ; NEED 1?
0061 001E A081 A R1, R2 ; N
0062 0020 0201 LI R1, >8000 ; Y, SET BIT TO ONE
0063      0022 8000 ; RETURN
0064 0024 0B01 SRC R1, 0
0065 0026 4481 SZC R1, *R2
0066 0028 C0C3 MOV R3, R3
0067 002A 1301 JEQ BITY1
0068 002C E481 SOC R1, *R2
0069 002E 0460 BITY1 B @NLIN
0070      0030 0000
0071 0032 2F81 ERR1 DATA ERROR+1

```

```

0069      *      THE BIT FUNCTION WILL DISPLAY ANY BIT VALUE OFFSET
0070      *      FROM A BASIC VARIABLE. THE FORM IS:-
0071      *
0072      *      BIT[ <EXP1> , <EXP> ]
0073      *
0074      *      WHERE  <EXP1> = BASIC VARIABLE
0075      *               <EXP2> = BIT POSITION
0076      *
0077      * CALLING SEQUENCE:
0078      *
0079      *      B @BITF
0080      *
0081      *      EXIT TO EVSFR$
0082      *
0083 0034 C041  BITF  MOV  R1,R1          ;TEST SIGN OF BIT £
0084 0036 1501          JGT  BITF0        ;+VE, OK
0085 0038 04C1          CLR  R1           ; -VE, MAKE 0
0086 003A C001  BITF0 MOV  R1,R0
0087 003C 0931          SRL  R1,3        ;GET WORD INDEX
0088 003E A081          A    R1,R2        ;INDEX
0089 0040 0201          LI   R1,>8000     ;GET INITIAL MASK
0042 8000
0090 0044 0B01          SRC  R1,0        ;SHIFT
0091 0046 C092          MOV  *R2,R2      ;GET WORD
0092 0048 2081          COC  R1,R2        ;BIT SET?
0093 004A 1602          JNE  BITF1        ;N, RETURN 0
0094 004C 05A0          INC  @FPAC2      ;Y, RETURN 1
004E 0000
0095      *
0096 0050 0460  BITF1 B    @EVSFR$      ;RETURN ADR
0052 0000
0097      END
NO ERRORS,      NO WARNINGS

```


ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.GETLINE
OBJECT ACCESS NAME= ADHOC.OBJ.GETLINE
LISTING ACCESS NAME= ADHOC.LST.GETLINE
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

```
0002          IDT  'GETLINE'
0003          *
0004          *
0005          *
0006          DEF  GTLN, EDER, C2000, B20, B08
0007          *
0008          REF  AINC, EDTMP, CVDIZ
0009          REF  DCNT, JMPRO
0010          REF  LNUM, LSTL, NIC, SFSN
0011          REF  TYP0$, TYP11$, TYPB$, TYPC$, TYPE$
0012          REF  GETCR$, IOB
0013          REF  CRLF, NOERR, MODE
0014          REF  ERRLS2, BLSTOR
0015          2F80  ERROR  EQU  >2F80
0016          DXOP  OUTINT, 13
```

```

0018      *
0019      *^E   *** EDIT LINE ***
0020      *
0021 0000 C3A0  GTCE   MOV   @MODE,R14           ; IDLE ?
          0002 0000
0022 0004 162B      JNE   GTLLP               ; N, IGNORE CHARACTER
0023 0006 C1E0      MOV   @IOB,R7             ; GET BUFFER ADR
          0008 0000
0024 000A 0420      BLWP  @CVDIZ              ; CONVERT £
          000C 0000
0025 000E 1000      NOP
0026 0010 2F8D      DATA ERROR+13           ; NO SUCH LINE
0027 0012 06A0      BL    @SFSN              ; SEARCH FOR STATEMENT £
          0014 0000
0028 0016 C1E0      MOV   @IOB,R7             ; RESET BUFFER ADR
          0018 000B
0029 001A 0720      SETO  @DCNT               ; RESET INDENT COUNTER
          001C 0000
0030 001E 06A0      BL    @LSTL              ; LIST LINE
          0020 0000
0031 0022 0000      DATA TYPC$              ; OUT CRLF
0032 0024 1014      JMP   GTCE2
0033      *
0034      *GET INPUT LINE
0035      *      R6 = MAX £ OF CHARACTERS
0036      *      R7 = I/O POINTER
0037      *
0038 0026 C80B  GTLN   MOV   R11,@BLSTOR       SAVE RETURN ADDRESS
          0028 0000
0039 002A 0206      LI    R6,NIC              ; MAXIMUM OF NIC CHARACTERS
          002C 0000
0040 002E C1E0      MOV   @IOB,R7             ; GET IOB PTR
          0030 001B
0041 0032 C0C7      MOV   R7,R3
0042 0034 C046      MOV   R6,R1
0043      *
0044 0036 04F3      CLR   *R3+                ; CLEAR BUFFER
0045 0038 0641      DECT  R1                  ; DONE?
0046 003A 15FD      JGT   $-4                 ; N
0047      *
0048 003C C060      MOV   @AINC,R1             ; AUTO-INCREMENT
          003E 0000
0049 0040 130C      JEQ   GTLLP$              ; N
0050 0042 A060      A     @LNUM,R1            ; Y, ADD IN LAST LINE £
          0044 0000
0051 0046 112B      JLT   GTCR               ; -VE, TERMINATE AUTO-INC
0052 0048 2F41      OUTINT R1                 ; CONVERT
0053 004A DDE0      MOVB  @B20,*R7+          ; SPACE
          004C 00CC
0054      *
0055 004E 0000  GTCE2  DATA TYPB$            ; TYPE
0056 0050 0206      LI    R6,NIC              ; GET NEW COUNT
          0052 002C
0057 0054 A1A0      A     @IOB,R6             ; ADD BUFFER ADR
          0056 0030
0058 0058 6187      S     R7,R6               ; SUB CURRENT ADR

```

```

0060 005A 04CF GTLLP$ CLR R15 ;RESET INSERT FLAG
0061 005C 0000 GTLLP DATA GETCR$ ;LOOP
0062 005E 0280 CI RO,>2000 CONTROL ?
      0060 2000
0063 0062 1401 JHE $+4 N, LEAVE INSERT FLAG
0064 0064 04CF CLR R15 Y, KILL INSERT FLAG
0065 0066 06A0 BL @JMPRO ;DO JUMP ON RO
      0068 0000
0066 006A 52 GTJMP BYTE GTLD-GTJMP/2,>17 ^W(ETB)DELETE
0067 006C CB GTJMP BYTE GTCE-GTJMP/2,>05 ^E(ENG)EDIT
0068 006E 35 GTJMP BYTE GTCF-GTJMP/2,>09 ^I(HT) CURSOR RIGHT
0069 0070 22 GTJMP BYTE GTBS-GTJMP/2,>08 ^H(BS) CURSOR LEFT
0070 0072 65 GTJMP BYTE GTLI-GTJMP/2,>16 ^V(SYN)INSERT
0071 0074 18 GTJMP BYTE GTLLP3-GTJMP/2,>0C ^L(FF) CLEAR SCREEN
0072 0076 F9 GTJMP BYTE GTLF-GTJMP/2,>0A LF IGNORE
0073 0078 1A GTJMP BYTE GTCR-GTJMP/2,>0D CR
0074 007A 27 GTJMP BYTE GTRB-GTJMP/2,>7F RUBOUT
0075 007C 0000 DATA 0
0076 007E 0280 CI RO,>2000 ;<BLK?
      0080 2000
0077 0082 1A09 JL GTLB ;Y, OUT BELL
0078 0084 D157 MOV B *R7,R5 ;EXPANDING BUFFER?
0079 0086 1602 JNE GTLLP1 ;N
0080 0088 0606 DEC R6 ;Y, ROOM?
0081 008A 1104 JLT GTLLP2 ;N, OUT BELL
0082 008C C3CF GTLLP1 MOV R15,R15 NORMAL MODE ?
0083 008E 1654 JNE INS N, INSERT
0084 0090 DDC0 MOV B RO,*R7+ ;Y, STORE BYTE
0085 0092 1003 JMP GTLLP3
0086 *
0087 005C' GTLF EQU GTLLP
0088 *
0089 0094 0586 GTLLP2 INC R6 ;RESTORE R6
0090 0096 0200 GTLB LI RO,>0700 ;OUT BELL
      0098 0700
0091 009A 0000 GTLLP3 DATA TYP0$ ;ECHO CHARACTER
0092 009C 10DF JMP GTLLP
0093 *
0094 009E DDD7 GTCR MOV B *R7,*R7+ AT END OF LINE? <<<
0095 00A0 1303 JEQ GTCR1 Y, EXIT <<<
0096 00A2 0000 DATA TYP11$,>0900 N, CURSOR RIGHT <<<
0097 00A6 10FB JMP GTCR AND LOOP <<<
0098 00AB C2E0 GTCR1 MOV @BLSTOR,R11 GET RETURN ADDRESS
      00AA 0028'
0099 00AC 045B RT RETURN
  
```

```

0101      *BACKSPACE (^H)
0102      *
0103 00AE 81E0 GTBS C @IOB,R7 ; BEGINNING OF BUFFER?
      00B0 0056'
0104 00B2 13F1 JEQ GTLB ; Y, OUT BELL
0105 00B4 0607 DEC R7 ; N, BACKUP
0106 00B6 100B JMP GTRB1 ; OUT BACKSPACE
0107      *
0108      *RUBOUT
0109      *
0110 00B8 04CF GTRB CLR R15 RESET INSERT FLAG
0111 00BA 81E0 C @IOB,R7 ; BEGINNING OF BUFFER?
      00BC 00B0'
0112 00BE 13EB JEQ GTLB ; Y, OUT BELL
0113 00C0 0607 DEC R7 ; N, BACKUP
0114 00C2 D5E0 MOVB @B20,*R7 ; STORE BLANK
      00C4 00CC'
0115 00C6 00A2' DATA TYP11%,>0800 ; OUT BKSP,BLK
0116 00CA 00C6' DATA TYP11%,>2000
0117 00CC' C2000 EQU #-2
0118 00CC' B20 EQU C2000
0119      *
0120 00CE 00CA' GTRB1 DATA TYP11% ; OUT BKSP
0121 00D0' B08 EQU %
0122 00D0 0800 DATA >0800
0123 00D2 10C4 JMP GTLLP ; GOTO LOOP
0124      *
0125      *FORWARD SPACE (^I)
0126      *
0127 00D4 D037 GTCF MOVB *R7+,R0 ; GET CHARACTER
0128 00D6 16E1 JNE GTLB+4 ; OK TO SEND
0129 00DB 0607 DEC R7 ; TOO FAR
0130 00DA 10DD JMP GTLB ; OUT BELL

```

```

0132      *
0133      * EDIT ERROR RECOVERY ROUTINE
0134      *
0135      00DC' EDER EQU $
0136 00DC C0E0 MOV @NDERR,R3 ERROR TRAP ENABLED?
      00DE 0000
0137 00E0 1302 JEQ EDER1 Y, HANDLE IT
0138 00E2 0460 B @CRLF N, IGNORE TRAP
      00E4 0000
0139 00E6 C0DB EDER1 MOV *R11,R3 PICKUP ERROR CHARACTERS
0140 00EB 0607 DEC R7
0141 00EA C147 MOV R7,R5 ;Y, MARK
0142 00EC 06A0 BL @ERRLS2 OUTPUT ERROR MESSAGE
      00EE 0000
0143 00F0 C1A0 MOV @EDTMP,R6 RESTORE & 'FREE SPACES' IN IOB
      00F2 0000
0144      *
0145 00F4 0022' DATA TYP C$ ;OUT CRLF
0146 00F6 004E' DATA TYP B$ ;OUT BUFFER
0147 00F8 00CE' DATA TYP 11$, >0D00 ;OUT CR ONLY
0148 00FC C1E0 MOV @IOB,R7
      00FE 00BC'
0149 0100 04CF CLR R15 ;RESET INSERT FLAG
0150 0102 8147 GTLI4 C R7,R5 ;POSITIONED?
0151 0104 14AB JHE GTLLP ;Y, GOTO LOOP
0152 0106 04C0 CLR R0
0153 0108 D037 MOV B *R7+,R0 ;N, OUT CHARACTER
0154 010A 009A' DATA TYP 0$
0155 010C 10FA JMP GTLI4 ;DO AGAIN
  
```

```

0157      *
0158      * DELETE
0159      *
0160 010E D5D7 GTLD   MOVB *R7,*R7      ; ANYTHING TO DELETE?
0161 0110 13C2      JEQ  GTLB          ; N, OUT BELL
0162 0112 C047      MOV  R7,R1          ; Y, SAVE CURRENT POSITION
0163      *
0164 0114 DDE7 DEL1   MOVB @1(7),*R7+    ; OVERWRITE WITH FOLLOWING CHAR
      0116 0001
0165 0118 16FD      JNE  DEL1          ; LOOP TILL WE COPY THE NULL
0166 011A D451      MOVB *R1,*R1        ; EMPTY LINE ?
0167 011C 1301      JEQ  DEL1A         ; Y, DONT OUTPUT IT THEN
0168 011E 0000      DATA TYPE#        ; OUTPUT REST OF LINE
0169 0120 00F8' DEL1A DATA TYP11%,>2000 ; AND A SPACE
0170 0124 61C1      S      R1,R7       ; CALCULATE # BS TO OUTPUT
0171 0126 0120' DEL2 DATA TYP11%,>0800 ; OUTPUT A BACKSPACE
0172 012A 0607      DEC  R7            ; DONE?
0173 012C 16FC      JNE  DEL2          ; N, LOOP
0174 012E C1C1 DEL3   MOV  R1,R7       ; RESTORE POINTER
0175 0130 0586      INC  R6            ; SOMETHING TO DELETE
0176 0132 1094      JMP  GTLLP         ; AND LOOP

```

```

0178      *
0179      * INSERT
0180      *
0181 0134 070F GTLI   SETD R15           ; FLAG AS INSERTING
0182 0136 1092      JMP  GTLLP         AND LOOP
0183      *
0184      * INSERT CHARACTERS
0185      *
0186 0138 0606 INS    DEC  R6           ; ROOM ?
0187 013A 11AD      JLT  GTLB         ; N
0188 013C 13AC      JEQ  GTLB         ; N
0189 013E C047      MOV  R7,R1        ; Y, SAVE CURRENT POSITION
0190      *
0191 0140 DDD7      MOVB *R7,*R7+      ; FOUND THE END ?
0192 0142 16FE      JNE  #-2          ; N, KEEP LOOKING
0193      *
0194 0144 D5E7 INS1  MOVB @-1(7),*R7   ; COPY IN PREVIOUS CHARACTER
0195      0146 FFFF
0196 0148 0607      DEC  R7           ; BACKUP
0197 014A 8047      C    R7,R1        ; ARE WE BACK WHERE WE STARTED?
0198 014C 1BFB      JH   INS1         ; N, CONTINUE MOVEING BUFFER
0199 014E DDC0      MOVB R0,*R7+      ; Y, INSERT THE CHARACTER
0200 0150 011E'     DATA TYPE#       ; OUTPUT THE UPDATED BUFFER
0201 0152 0581      INC  R1           ; BUMP POINTER
0202 0154 D031 INS2  MOVB *R1+,R0     ; ANYMORE BACKSPACES?
0203 0156 1382      JEQ  GTLLP         ; N, DO MAIN LOOP
0204 0158 0126'     DATA TYP11$,>0800 ; Y, OUT A BS
0204 015C 10FB      JMP  INS2         ; LOOP TILL CURSOR POSITIONED

NO ERRORS,      NO WARNINGS

```


ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.GETP2
OBJECT ACCESS NAME= ADHOC.OBJ.GETP2
LISTING ACCESS NAME= ADHOC.LST.GETP2
ERROR ACCESS NAME=
OPTIONS= BUNLST, DUNLST, MUNLST
MACRO LIBRARY PATHNAME=

```

0002          IDT  'GETP2'
0003          *
0004          *      GETP2                      ; GET POWER OF 2
0005          *
0006          *      DXOP FMUL,4                ; MULTIPLY FPAC
0007          *
0008          *      REF  CVCH                    ; VARIABLE HOLDER
0009          *      DEF  GETP2                  ; ENTRY POINT
0010          *
0011          *      GETP2 WILL RETURN A 3 WORD FLOATING
0012          *      POINT POWER OF 2 AS SPECIFIED BY R2.
0013          *
0014          *  CALLING SEQUENCE:
0015          *
0016          *      BL @GETP2
0017          *
0018          *      IN R2      = 1,2 OR 3
0019          *      OUT CVCH = £
0020          *
0021          *
0022 0000 0202  GETP2  LI    R2,GETPC-2
          0002 0012'
0023 0004 0812          SRA  R2,1                ; /2
0024 0006 A0B3          A    R3,R2                ; INDEX
0025 0008 0A12          SLA  R2,1                ; MAKE WORD INDEX
0026 000A C812          MOV  *R2,@CVCH            ; MOVE INTO CONSTANT
          000C 0000
0027 000E 2D20          FMUL @CVCH                ; MULTIPLY FPAC
          0010 000C'
0028 0012 045B          RT
0029          *
0030 0014 4120  GETPC  DATA >4120,>4140,>4180 POWERS OF 2
0031          END
NO ERRORS,      NO WARNINGS
  
```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.GETPARM
OBJECT ACCESS NAME= ADHOC.OBJ.GETPARM
LISTING ACCESS NAME= ADHOC.LST.GETPARM
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

```

0002          IDT 'GETPARM'
0003          *
0004          *      GPRM, GPRM1, GPRM2          ; GET PARAMETER
0005          *
0006          DXOP EVFIX, 11          ; EVALUATE AND FIX
0007          2F80 ERROR EQU >2F80          ; XOP XX, 14. (ERROR CALL)
0008          2FA0 ERROR2 EQU ERROR+>20
0009          *
0010          REF B4A, B56
0011          DEF GPRM, GPRM1, GPRM2
0012          *
0013          *      FIX PARAMETERS SUCH THAT:
0014          *
0015          *      XXX[ (R2) , R1 ] = R3
0016          *
0017          *      DIFFERENT ENTRY POINTS WILL PICK UP
0018          *      THE EVALUATION AND CHECKING AT
0019          *      DIFFERENT POINTS.
0020          *
0021          * CALLING SEQUENCE:
0022          *
0023          *      BL @GPRM
0024          *      BL @GPRM1
0025          *      BL @GPRM2
0026          *
0027          *      IN (R2) = VARIABLE
0028          *      OUT R1 = INDEX
0029          *      R3 = ASSIGNMENT FIXED
0030          *
0031          * EXCEPTIONS AND CONDITIONS:
0032          *
0033          *      EVALUATION ERRORS, FIX ERRORS, AND
0034          *      EXPECTING OPERATOR.
0035          *
0036          *
0037          0000 9838 GPRM CB *R8+, @B4A          ; RIGHT BRACKET?
0038          0002 0000          JNE ERR1          ; N, ERROR
0039          *
0040          0006 2EC1 GPRM1 EVFIX R1          ; GET INDEX
0041          *
0042          0008 0280 GPRM2 CI R0, >4B00          ; I?
0043          000A 4B00          JNE ERR1          ; N
0044          000E 7E20 CB @B56, *R8+          ; =?
0045          0010 0000          JNE ERR36          ; N
0046          0014 2EC3 EVFIX R3          ; GET RESULT
0047          0016 045B RT
0048          *
0049          0018 2FB1 ERR1 DATA ERROR+1
0050          001A 2FA0 ERR36 DATA ERROR2, 36
0051          END
  
```

NO ERRORS, NO WARNINGS

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.LOGF
OBJECT ACCESS NAME= ADHOC.OBJ.LOGF
LISTING ACCESS NAME= ADHOC.LST.LOGF
ERROR ACCESS NAME=
OPTIONS= BUNLST, DUNLST, MUNLST
MACRO LIBRARY PATHNAME=

```

0002          IDT 'LOGF'
0003          *
0004          * ROUTINE LIST:
0005          *
0006          *      LOGF          ; NATURAL LOG FUNCTION
0007          *
0008          REF FUNFX          ; FIX ARGUMENT
0009          REF PLYX,PLYXX    ; EVALUATE POLYNOMIAL
0010          REF GETP2        ; GET POWER OF 2
0011          REF FPAC         ; FLOATING POINT ACCUMULATOR
0012          REF TEMP         ; 3 WRD TEMPORARY STORAGE
0013          REF DS,DS1       ; 3 WRD TEMPORARY STORAGE
0014          REF B40
0015          DXOP LOADF,0      ; LOAD FPAC
0016          DXOP STORE,1      ; STORE FPAC
0017          DXOP FADD,2       ; ADD TO FPAC
0018          DXOP FSUB,3       ; SUBTRACT FROM FPAC
0019          DXOP FMUL,4       ; MULTIPLY FPAC
0020          DXOP FDIV,5       ; DIVIDE FPAC
0021          DXOP NORMAL,7     ; NORMALIZE FPAC
0022          2F80 ERROR EQU >2F80 ; XOP XX,14 (ERROR CALL)

```

```

0024      *
0025      *      COMPUTE THE NATURAL LOG OF (R2)
0026      *
0027      * CALLING SEQUENCE:
0028      *
0029      *      BL @LOGF
0030      *
0031      *      IN  (R2) = ARG
0032      *      OUT (R2) = RESULT
0033      *
0034      * EXCEPTIONS AND CONDITIONS:
0035      *
0036      *      LOG OF NON-POSITIVE NUMBER
0037      *
0038      * ENTRY POINT:
0039      *
0040      *      DEF LOGF
0041      *
0042 0000 2F9A ERR26  DATA ERROR+26      ; LOG OF NON-POSITIVE NUMBER
0043      *
0044 0002 C28B LOGF   MOV R11,R10          ; SAVE RETURN
0045 0004 06A0      BL @FUNFX            ; FIX
0046      *
0046 0008 10FB      JMP ERR26            ; LOG 0
0047 000A C0C3      MOV R3,R3            ; NEGATIVE?
0048 000C 16F9      JNE ERR26            ; Y, ERROR
0049 000E D820      MOV @B40,@FPAC       ; LOAD EXPONENT
0050      *
0050 0014 0961      SRL R1,6             ; SWAP & X 4
0051 0016 0221      AI R1,->100          ; UNBIAS
0052      *
0052 001A C092      MOV *R2,R2           ; GET FPAC
0053 001C 0200      LI R0,>80
0054      *
0054 0020 2080      CDC R0,R2            ; HIGH BIT SET?
0055 0022 1308      JEQ LOGF2            ; Y
0056 0024 04C3      CLR R3               ; N, FIND POWER OF 2
0057      *
0058 0026 0583 LOGF1 INC R3               ; COUNT
0059 0028 0910      SRL R0,1
0060 002A 2080      CDC R0,R2            ; BIT SET?
0061 002C 16FC      JNE LOGF1            ; N, CONTINUE TO COUNT
0062 002E 6043      S R3,R1
0063 0030 06A0      BL @GETP2            ; GET POWER OF 2 & MULTIPLY
0064      *
0065 0034 C0B1 LOGF2 MOV R1,R2
0066 0036 2C60      STORE @TEMP          ; MOVE TO TEMP
0067      *
0067 003A 2CE0      FSUB @LOGCO
0068      *
0068 003E 0200      LI R0,FP1            ; GET FP1
0069      *
0069 0042 C060      MOV @FPAC,R1         ; CHECK SIGN
0070      *
0070 0046 1504      JGT LOGF3
0071 0048 1303      JEQ LOGF3
0072 004A 0200      LI R0,LOGC1
0073      *
0073 004C 00B0'

```

```

0073 004E 0602      DEC R2
0074                *
0075 0050 2C20      LOGF3  LOADF @TEMP      ; MOVE TEMP TO FPAC
                   0052 0038'
0076 0054 2C90      FADD *R0
0077 0056 2C60      STORE @DS      ; SAVE IN DS
                   0058 0000
0078 005A 2C20      LOADF @TEMP      ; RELOAD TEMP
                   005C 0052'
0079 005E 2CD0      FSUB *R0
0080 0060 2D60      FDIV @DS      ; DIVIDE BY DS
                   0062 0058'
0081 0064 2C60      STORE @DS      ; SAVE IN DS
                   0066 0062'
0082 0068 06A0      BL @PLYXX
                   006A 0000
0083 006C 00B6'      DATA LOGC2
0084 006E 2C60      STORE @DS1      ; STORE IN DS1
                   0070 0000
0085 0072 06A0      BL @PLYX
                   0074 0000
0086 0076 00D0'      DATA LOGC3
0087 0078 2D60      FDIV @DS1      ; DIVIDE BY DS1
                   007A 0070'
0088 007C 2D20      FMUL @DS      ; MULTIPLY BY DS
                   007E 0066'
0089 0080 2C60      STORE @TEMP      ; STORE IN TEMP
                   0082 005C'
0090 0084 0200      LI R0,>8C00
                   0086 8C00
0091 0088 C082      MOV R2,R2
0092 008A 1102      JLT LOGF4
0093 008C 0910      SRL R0,1
0094 008E 1002      JMP LOGF5
0095                *
0096 0090 0502      LOGF4  NEG R2
0097 0092 0810      SRA R0,1
0098                *
0099 0094 C042      LOGF5  MOV R2,R1
0100 0096 04C2      CLR R2
0101 0098 2C00      LOADF R0      ; LOAD FPAC
0102 009A 2DC0      NORMALIZE 0
0103 009C 2D20      FMUL @LOGC4
                   009E 00E4'
0104 00A0 2CA0      FADD @TEMP
                   00A2 0082'
0105                *
0106 00A4 0202      LOGF6  LI R2,FPAC
                   00A6 0044'
0107 00A8 045A      B *R10
0108                *
0109 00AA 40B5      LOGC0  DATA >40B5,>04F3,>33FA
0110 00B0 4080      LOGC1  DATA >4080,>0000,>0000      1/2
0111 00B6 0003      LOGC2  DATA 3
0112 00B8 4110      FP1    DATA >4110,>0000,>0000
0113 00BE C214      DATA >C214,>BBC5,>DCDB
0114 00C4 423D      DATA >423D,>C2D5,>31F0
0115 00CA C22D      DATA >C22D,>165C,>4BE0
0116 00D0 0002      LOGC3  DATA 2
0117 00D2 C212      DATA >C212,>53EF,>500E

```


0118 00D8 425D DATA >425D,>76C2,>314A
0119 00DE C25A DATA >C25A,>2CB8,>97BF
0120 00E4 40B1 LOGC4 DATA >40B1,>7217,>F7D2
0121 00EA EVEN
0122 END
NO ERRORS, NO WARNINGS

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.PLOT
OBJECT ACCESS NAME= ADHOC.OBJ.PLOT
LISTING ACCESS NAME= ADHOC.LST.PLOT
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

```
0002          IDT  'PLOT'
0003          *
0004          *
0005          *
0006      2F80  ERROR  EQU  >2F80
0007      2FA0  ERROR2 EQU  ERROR+>20
0008          DXOP  EVFIX,11
0009          *
0010          DEF   PLOTY,UPLITY,PLOT,UNPLOT
0011          REF   VDPWP1,PIXON,PIXOFF,V1R5L,V1R6L
0012          REF   V1R3L,V1R4L
0013          REF   XLOC,YLOC,NXLOC,NYLOC,NLIN,JMPROA
0014          REF   FGM$
```

PLOT 5142
UNPLOT 513E

```
0016      *
0017      *
0018      *
0019      *
0020      *
0021      *
0022      *
0023      *
0024      *
0025      *
0026      *
0027      *
0028      *
0029      *
0030      *
0031      *
0032      *
0033      *
0034      *
0035      *
0036      *
0037      *
0038      *
0039      *
```

PLOT AND UNPLOT POINT/LINE ROUTINES

FORMAT:

PLOT A,B	PLOT POINT (A,B)
PLOT TO A,B	PLOT LINE FROM CURSOR TO (A,B)
PLOT A,B TO C,D	PLOT LINE FROM (A,B) TO (C,D)
UNPLOT A,B	UNPLOT POINT (A,B)
UNPLOT TO A,B	UNPLOT LINE FROM CURSOR TO (A,B)
UNPLOT A,B TO C,D	UNPLOT LINE FROM (A,B) TO (C,D)

FOLLOWING ONE OF THESE STATEMENTS THE CURSOR IS
POSITIONED AT THE LAST CO-ORDINATE PAIR SPECIFIED.

AFTER THE LAST CO-ORDINATE PAIR A 'TO' MAY BE USED
TO EXTEND THE LINE (UN)PLOTING WITHIN THE ONE
STATEMENT. EG: -

PLOT TO A,B TO C,D TO E,F	THIS WILL PLOT THE 3 LINES
	1. FROM CURSOR TO (A,B)
	2. FROM (A,B) TO (C,D)
	3. FROM (C,D) TO (E,F)

```

0041 508E 070D UPL0TY SET0 R13 FLAG AS 'UNPLOT'
0042 509D 1001 JMP PLOTE >5144
0043 50A2 04CD PLOTY CLR R13 FLAG AS 'PLOT'
0044 *
0045 50A4 0000 PLOTE DATA FGM$>0012 FORCE GRAPH MODE
0046 50A8 06A0 BL @JMPROA >15F2 Y, GET BYTE & CHECK IT
0047 50A8 18 EOLTST BYTE PLTO-EOLTST/2,>38 'TO' - PLOT LINE
0048 50AE 16 BYTE PLTE-EOLTST/2,>00 NULL - EXIT
0049 50AE 16 BYTE PLTE-EOLTST/2,>3C ':' - EXIT
0050 50F2 16 BYTE PLTE-EOLTST/2,>47 '!' - EXIT
0051 5052 0000 DATA 0
0052 *
0053 * NO LEADING 'TO' - MUST BE AN XY PAIR
0054 *
0055 5054 0608 DEC RB BACKUP CODE PTR
0056 0018 06A0 BL @GETXY >519C GET X,Y PAIR
0057 001A 005E'
0057 001C D80F MOVB R15,@XLOC EE36 SET CURSOR X
0058 001E 0000
0058 0020 D80E MOVB R14,@YLOC EE37 SET CURSOR Y
0059 0022 0000
0059 0024 0280 CI R0,>3800 'TO' ?
0060 0026 3800
0060 0028 1309 JEQ PLTO >517A Y, LINE
0061 002A C34D MOV R13,R13 N, PIXEL SET ?
0062 002C 1303 JEQ SETIT >5172 Y,
0063 002E 0420 BLWP @PIXOFF N, TURN PIXEL OFF
0064 0030 0000
0064 0032 1002 JMP PLTE AND QUIT
0065 *
0066 0034 0420 SETIT BLWP @PIXON TURN PIXEL ON
0067 0036 0000
0067 0038 0460 PLTE B @NLIN EXIT TO NLIN
0068 003A 0000
0068 *
0069 *
0070 003C 06A0 PLTO BL @GETXY 'TO' FOUND - GET X,Y PAIR
0071 003E 005E'
0071 0040 D80F MOVB R15,@NXLOC SET STOP X
0072 0042 0000
0072 0044 D80E MOVB R14,@NYLOC SET STOP Y
0073 0046 0000
0073 0048 C34D MOV R13,R13 PLOT ?
0074 004A 1606 JNE UNPLT N, DO UNPLOT LINE
0075 004C 0420 BLWP @PLOT Y, CALL PLOT LINE
0076 004E 0080'
0076 *
0077 * CHECK FOR FOLLOWING 'TO'
0078 *
0079 0050 0280 LOOKTO CI R0,>3800 'TO' FOLLOWING ?
0080 0052 3800
0080 0054 13F3 JEQ PLTO Y, CONTINUE
0081 0056 10F0 JMP PLTE N, EXIT
0082 *
0083 0058 0420 UNPLT BLWP @UNPLOT CALL UNPLOT LINE
0084 005A 0076'
0084 005C 10F9 JMP LOOKTO CHECK FOR 'TO'
0085 *
0086 * GET A FOLLOWING X,Y PAIR AND STORE THEM IN THE

```

1838	>518C
1600	>514E
1630	>51C8
1647	>51E0

6000

```
0087      *      MS BYTE OF R15 & R14 RESPECTIVLY.
0088      *      RANGE CHECKING IS DONE IN PIXON/PIXOFF ROUTINES
0089      *
0090 005E 2ECF GETXY EVFIX R15          GET X
0091 0060 06CF      SWPB R15          POSITION IT
0092 0062 0280      CI  R0,>3F00      ', ' ?
        0064 3F00
0093 0066 1603      JNE  ERR37        N, ERROR IT
0094 0068 2ECE      EVFIX R14        Y, GET Y
0095 006A 06CE      SWPB R14        POSITION IT
0096 006C 045B      RT
0097      *
0098 006E 2FA0 ERR37 DATA ERROR2,37   INVALID DELIMITER
0099 0072 2FA0 ERR48 DATA ERROR2,48   NOT ALLOWED IN CURRENT MODE
```

0101	*	WORKSPACE	:	VDPWP1	
0102	*	ROUTINE(S)	:	PLOT, UNPLOT	
0103	*	REGISTER USAGE	:		
0104	*				GLOBAL DATA
0105	*				LABELS
0106	*	R0	:	PLOT/UNPLOT POINTER	
0107	*	R1	:		
0108	*	R2	:		
0109	*	R3	:		LSB=V1R3L
0110	*	R4	:		LSB=V1R4L
0111	*	R5	:		LSB=V1R5L
0112	*	R6	:		LSB=V1R6L
0113	*	R7	:		
0114	*	R8	:		
0115	*	R9	:		
0116	*	R10	:		
0117	*	R11	:	BL RETURN ADDRESS	
0118	*	R12	:		
0119	*	R13	:	RETURN WP	
0120	*	R14	:	RETURN PC	
0121	*	R15	:	RETURN ST	
0122	*				
0123	*				
0124	0076	0000	UNPLOT	DATA	VDPWP1, \$+2
0125	007A	0200	LI	R0, PIXOFF	
	007C	0030			
0126	007E	1004	JMP	PLOT1	
0127	0080	0076	PLOT	DATA	VDPWP1, \$+2
0128	0084	0200	LI	R0, PIXON	
	0086	0036			
0129	0088	070B	PLOT1	SETO	R11
0130	008A	070A		SETO	R10
0131	008C	0709		SETO	R9
0132	008E	04C3		CLR	R3
0133	0090	04C4		CLR	R4
0134	0092	DB20		MOVB	@XLOC, @V1R4L
	0094	001E			
	0096	0000			
0135	0098	DB20		MOVB	@NXLOC, @V1R3L
	009A	0042			
	009C	0000			
0136	009E	6103	S	R3, R4	
0137	00A0	0744	ABS	R4	
0138	00A2	1101	JLT	INITY	
0139	00A4	050A	NEG	R10	
0140	00A6	DB20	INITY	MOVB	@YLOC, @V1R3L
	00A8	0022			
	00AA	009C			
0141	00AC	04C5	CLR	R5	
0142	00AE	DB20		MOVB	@NYLOC, @V1R5L
	00B0	0046			
	00B2	0000			
0143	00B4	60C5	S	R5, R3	
0144	00B6	0743	ABS	R3	
0145	00B8	1101	JLT	XYDRYX	
0146	00BA	0509	NEG	R9	
0147	00BC	80C4	XYDRYX	C	R4, R3
0148	00BE	1508	JGT	LEEVE	
0149	00C0	2903	XOR	R3, R4	
0150	00C2	28C4	XOR	R4, R3	

```

0151 00C4 2903          XOR   R3,R4
0152 00C6 D160          MOVB  @NYLOC,R5
      00C8 00B0'
0153 00CA D1A0          MOVB  @NXLOC,R6
      00CC 009A'
0154 00CE 1008          JMP    SLOPE
0155 00D0 D160  LEEVEM  MOVB  @NXLOC,R5
      00D2 00CC'
0156 00D4 D1A0          MOVB  @NYLOC,R6
      00D6 00C8'
0157 00DB 2A89          XOR   R9,R10
0158 00DA 2A4A          XOR   R10,R9
0159 00DC 2A89          XOR   R9,R10
0160 00DE 05CB          INCT  R11
0161 00E0 0985  SLOPE  SRL   R5,8
0162 00E2 0986          SRL   R6,8
0163 00E4 0A53          SLA   R3,5
0164 00E6 04C2          CLR   R2
0165 00E8 3CB4          DIV   R4,R2
0166 00EA C1C2          MOV   R2,R7
0167 00EC C083          MOV   R3,R2
0168 00EE 04C3          CLR   R3
0169 00F0 1901          JNO   DIVLSB
0170 00F2 1003          JMP   SETREM
0171 00F4 3CB4  DIVLSB  DIV   R4,R2
0172 00F6 C0C2          MOV   R2,R3
0173 00F8 C087          MOV   R7,R2
0174 00FA 0207  SETREM  LI    R7,>10
      00FC 0010
0175 00FE 04C8          CLR   R8
0176 0100 04CC          CLR   R12
0177 0102 810C  PLOTLP  C    R12,R4
0178 0104 1B1D          JH    EXIT
0179 0106 058C          INC   R12
0180 0108 C2CB          MOV   R11,R11
0181 010A 1107          JLT   XLESS
0182 010C D820          MOVB  @V1R5L,@XLOC
      010E 00B2'
      0110 0094'
0183 0112 D820          MOVB  @V1R6L,@YLOC
      0114 0000
      0116 00A8'
0184 0118 1006          JMP   SCNDRW
0185 011A D820  XLESS  MOVB  @V1R6L,@XLOC
      011C 0114'
      011E 0110'
0186 0120 D820          MOVB  @V1R5L,@YLOC
      0122 010E'
      0124 0116'
0187 0126 0410  SCNDRW  BLWP  *R0
0188 0128 A203          A     R3,R8
0189 012A 1701          JNC   NOCARY
0190 012C 0587          INC   R7
0191 012E A1C2  NOCARY  A     R2,R7
0192 0130 0287          CI    R7,>20
      0132 0020
0193 0134 1A03          JL    EXPLOT
0194 0136 0227          AI    R7,->20
      0138 FFEO
0195 013A A18A          A     R10,R6

```



```
0196 013C A149 EXPLOT A R9,R5
0197 013E 10E1 JMP PLOTLP
0198 0140 D820 EXIT MOV B @NXLOC,@XLOC
      0142 00D2'
      0144 011E'
0199 0146 D820 MOV B @NYLOC,@YLOC
      0148 00D6'
      014A 0124'
0200 014C 0380 RTWP
0201 END
NO ERRORS, NO WARNINGS
```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.POLY
OBJECT ACCESS NAME= ADHOC.OBJ.POLY
LISTING ACCESS NAME= ADHOC.LST.POLY
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

```
0002          IDT 'POLY'
0003          *
0004          * ROUTINE LIST:
0005          *
0006          *      FUNBK          ; BREAK FPAC INTO INTEGER & FRACTION
0007          *      FUNFX          ; FIX FPAC INTO EXPONENT AND ABSOLUTE
0008          *      PLYX           ; EVALUATE P(X) WITH X IN TEMP
0009          *      PLYXX          ; EVALUATE P(X*X) WITH X IN FPAC
0010          *
0011          DXOP LOADF,0          ; LOAD FPAC
0012          DXOP STORE,1         ; STORE FPAC
0013          DXOP FADD,2          ; ADD TO FPAC
0014          DXOP FSUB,3          ; SUBTRACT FROM FPAC
0015          DXOP FMUL,4          ; MULTIPLY FPAC
0016          DXOP FDIV,5          ; DIVIDE FPAC
0017          DXOP SCALE,6         ; SCALE FPAC
0018          DXOP NORMAL,7        ; NORMALIZE FPAC
0019          DXOP CLEAR,8         ; CLEAR FPAC
0020          DXOP NEGATE,9         ; NEGATE FPAC
0021          DXOP FLOATF,10       ; FLOAT FPAC
0022          2F80 ERROR EQU >2F80 ; XOP XX,14 (ERROR CALL)
0023          *
0024          REF FPAC,FPAC4        ; FLOATING POINT ACCUMULATOR
0025          REF TEMP              ; 3 WRD TEMPORARY STORAGE
0026          REF C4A00             ; >4A00
0027          DEF FUNBK,FUNFX
0028          DEF PLYX,PLYXX
```

```

0030      * ABSTRACT:
0031      *
0032      *      BREAK FPAC INTO INTEGER IN R1 AND
0033      *      LEAVE FRACTIONAL PART IN FPAC.
0034      *
0035      * CALLING SEQUENCE:
0036      *
0037      *      BL @FUNBK
0038      *
0039      *      IN FPAC = £
0040      *      OUT R1 = INTEGER PART
0041      *      FPAC = FRACTIONAL PART
0042      *
0043      * EXCEPTIONS AND CONDITIONS:
0044      *
0045      *      ERROR 30 = FIX ERROR
0046      *
0047 0000 C060 FUNBK MOV @FPAC,R1      GET HIGH WORD
      0002 0000
0048 0004 A041      A R1,R1      REMOVE SIGN
0049 0006 02B1      CI R1,>8900    TOO BIG?
      0008 8900
0050 000A 1B0F      JH ERR30      Y, ERROR
0051 000C 2C60      STORE @TEMP    ;N, MOVE TO TEMP
      000E 0000
0052 0010 2DA0      SCALE @C4A00   ;SCALE
      0012 0000
0053 0014 C060      MOV @FPAC4,R1  LOAD LOW
      0016 0000
0054 0018 C0A0      MOV @FPAC,R2   <0?
      001A 0002'
0055 001C 1501      JGT FUNBK1     N
0056 001E 0501      NEG R1        Y, GET 2'S COMPLEMENT
0057      *
0058 0020 2DC0 FUNBK1 NORMALIZE 0   NORMALIZE
0059 0022 2E40      NEGATE 0      NEGATE
0060 0024 2CA0      FADD @TEMP     ADD TEMP TO FPAC
      0026 000E'
0061 0028 045B      B *R11        RETURN
0062      *
0063 002A 2F9E ERR30 DATA ERROR+30
    
```

```

0065      * ABSTRACT:
0066      *
0067      *      FIX FPAC INTO EXPONENT IN R1, SIGN BIT
0068      *      IN R3, AND ABSOLUTE VALUE IN FPAC
0069      *
0070      * CALLING SEQUENCE:
0071      *
0072      *      BL @FUNFX
0073      *
0074      *      IN  R2 = £
0075      *      OUT R1 = EXPONENT
0076      *      (R2) = ABS(FPAC)
0077      *      R3 = SIGN BIT
0078      *
0079 002C 2C12  FUNFX  LOADF *R2      ; LOAD FPAC
0080 002E 2E80      FLOATF 0        ; FLOAT IF NECESSARY
0081 0030 0202      LI      R2,FPAC  GET ADR
0082      0032 001A'      MOV  *R2,R3  GET SIGN
0083 0034 C0D2      JEQ  FUNFX2      ZERO
0084 0036 1306      SRL  R3,15      GET SIGN BIT
0085 0038 09F3      JEQ  FUNFX1      + OR 0
0086 003A 1301      JEQ  FUNFX1      -, NEGATE
0087 003C 2E40      NEGATE 0
0088      *
0088 003E 05CB  FUNFX1 INCT R11      RETURN @2(11)
0089 0040 04C1      CLR  R1
0090 0042 D052      MOVB *R2,R1      GET EXPONENT
0091 0044 045B  FUNFX2 RT
    
```

```
0093      * ABSTRACT:
0094      *
0095      *      EVALUATE POLYNOMIAL P(X) OR P(X*X) WITH
0096      *      X IN TEMP OR FPAC RESPECTIVELY.  THE
0097      *      POLYNOMIAL IS GIVEN BY:
0098      *
0099      *              (RO) = N,C1,C2,...,CN
0100      *
0101      *      WHERE  N = # OF COEFFICIENTS
0102      *              C = COEFFICIENTS
0103      *
0104      * CALLING SEQUENCE:
0105      *
0106      *      BL @PLYXX
0107      *      BL @PLYX
0108      *
0109      *      IN (RO) = N,C1,C2,...,CN
0110      *      OUT FPAC = RESULT
0111      *
0112      * EXCEPTIONS AND CONDITIONS:  (NONE)
0113      *
0114      *EVALUATE P(X*X) WITH X IN FPAC
0115      *
0116 0046 2C60 PLYXX  STORE @TEMP          MOVE X TO TEMP
0117      0048 0026'
0117 004A 2D20      FMUL @TEMP          FPAC=FPAC*FPAC
0118      004C 0048'
0118 004E 2C60      STORE @TEMP          MOVE X*X INTO TEMP
0119      0050 004C'
0119      *
0120      *EVALUATE P(X) WITH X IN TEMP
0121      *
0122 0052 C03B PLYX  MOV  *R11+,RO        GET CONSTANTS ADR
0123 0054 C070      MOV  *RO+,R1        GET COUNT
0124 0056 2C10      LOADF *RO          LOAD 1ST CONSTANT
0125      *
0126 0058 2D20 PLYXA FMUL @TEMP          FPAC=FPAC*TEMP
0127      005A 0050'
0127 005C 0220      AI    RO,6         MOVE TO NEXT CONSTANT
0128      005E 0006
0128 0060 2C90      FADD *RO          ADD
0129 0062 0601      DEC  R1            DONE?
0130 0064 16F9      JNE  PLYXA        N
0131 0066 045B      B    *R11         Y
0132      END
NO ERRORS,      NO WARNINGS
```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.POWF
OBJECT ACCESS NAME= ADHOC.OBJ.POWF
LISTING ACCESS NAME= ADHOC.LST.POWF
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

```
0002          IDT 'POWF'
0003          *
0004          * ROUTINE LIST:
0005          *
0006          *          POWF          ; POWER FUNCTION
0007          *
0008          REF LOGF          ; LOG FUNCTION
0009          REF EXPF          ; EXP FUNCTION
0010          REF EVOP3A        ; EXIT TO EVAL
0011          DXOP LOADF,0      ; LOAD FPAC
0012          DXOP STORE,1      ; STORE FPAC
0013          DXOP FMUL,4       ; MULTIPLY FPAC
0014          DXOP FLOATF,10    ; FLOAT FPAC
0015          REF TEMP,TEMP6    ; TEMPORARY STORAGE
0016          DEF POWF
```



```

0018      * ABSTRACT:
0019      *
0020      *      RAISE (R1) TO THE (R2) POWER USING
0021      *      LOG AND EXPONENTIAL FUNCTIONS.
0022      *
0023      *       $R1 \wedge R2 = \text{EXP}(R1 * \text{LOG}(R2))$ 
0024      *
0025      * CALLING SEQUENCE:
0026      *
0027      *      B @POWF
0028      *
0029      *      IN  (R1) = ARG1
0030      *      (R2) = ARG2
0031      *      OUT (R2) = RESULT
0032      *
0033      *      EXIT TO EVOP3A
0034      *
0035      * EXCEPTIONS AND CONDITIONS:
0036      *
0037      *      PRESERVE R0
0038      *      EXP, LOG, AND FP ERRORS
0039      *
0040      *      DEF POWF
0041      *
0042 0000 C101 POWF MOV R1,R4 ; PRESERVE R0,R1
0043 0002 C800 MOV R0,@TEMP6
0044 0004 0000
0044 0006 06A0 BL @LOGF ; GET LOG(R2)
0044 0008 0000
0045 000A 2C60 STORE @TEMP ; MOVE TO TEMP
0045 000C 0000
0046 000E 2C14 LOADF *R4 ; GET B
0047 0010 2E80 FLOATF 0 ; FLOAT IF NECESSARY
0048 0012 2D20 FMUL @TEMP ; MULTIPLY BY TEMP
0048 0014 000C
0049 0016 06A0 BL @EXPF ; GET EXP(R1*LOG(R2))
0049 0018 0000
0050 001A C020 MOV @TEMP6,R0 ; RESTORE R0
0050 001C 0004
0051 001E 0460 B @EVOP3A
0051 0020 0000
0052      END
NO ERRORS, NO WARNINGS
    
```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.RNDF
OBJECT ACCESS NAME= ADHOC.OBJ.RNDF
LISTING ACCESS NAME= ADHOC.LST.RNDF
ERROR ACCESS NAME=
OPTIONS= BUNLST, DUNLST, MUNLST
MACRO LIBRARY PATHNAME=

```

0002          IDT 'RNDF'
0003          *
0004          *      RANDG          ; RANDOM NUMBER GENERATOR
0005          *      RAND          ; GET FLOATING POINT RANDOM NUMBER
0006          *
0007          *      REF FNRM          ; FIXES ARGUMENT
0008          *      REF RANDS        ; RANDOM SEED
0009          *      REF FPWP         ; FLOATING POINT WORKSPACE POI
0010          *
0011          *      RETURNS A 16 BIT RANDOM NUMBER.  THE SEED
0012          *      IS MAINTAINED IN RANDS AND IS UPDATED
0013          *      ON EACH CALL.
0014          *
0015          *  GENERATE PSEUDO RANDOM SEQUENCE OF INTEGERS
0016          *
0017          *      LINEAR CONGRUENTIAL SEQUENCE:
0018          *       $X[N+1] = (X[N] * A + C) \text{ MOD } 2^{16}$ 
0019          *
0020          *  USE MOST SIGNIFICANT K-BITS IF <16 ARE REQUIRED
    
```

```

0022      * CALLING SEQUENCE:
0023      *
0024      *      BL @RANDG
0025      *
0026      *      OUT RO = 16 BIT RANDOM NUMBER
0027      *
0028      *
0029 0000 C020  RANDG  MOV @RANDS,RO    ; GET RANDOM SEED
      0002 0000
0030 0004 C0C0      MOV RO,R3
0031 0006 0AB3      SLA R3,11          ; N*2^11
0032 0008 A0C0      A RO,R3           ; N*2^11+N
0033 000A 0A23      SLA R3,2
0034 000C A003      A R3,RO           ; N*2^11+5N
0035 000E 0220      AI RO,13849       ; (N*A+13849.) MOD 2^16
      0010 3619
0036 0012 C800      MOV RO,@RANDS     ; STORE NEW SEED
      0014 0002
0037 0016 045B      RT
0038      *
0039      *
0040      * ABSTRACT:
0041      *
0042      *      RETURNS A FLOATING POINT RANDOM NUMBER IN FPAC.
0043      *      TWO 16-BIT RANDOM NUMBERS ARE USED TO GENERATE
0044      *      THE THREE WORD FLOATING POINT NUMBER.
0045      *
0046      * CALLING SEQUENCE:
0047      *
0048      *      BLWP @RANDZ
0049      *
0050      *      OUT FPAC = FLOATING POINT RANDOM NUMBER
0051      *
0052      *      FNRM          ; NORMALIZE NUMBER
0053      *      DEF RANDZ
0054      *
0055 0018 0000  RANDZ  DATA FWP,RAND
0056      *
0057 001C 06A0  RAND   BL @RANDG        ; GET RANDOM £
      001E 0000
0058 0020 C040      MOV RO,R1
0059 0022 06A0      BL @RANDG
      0024 0000
0060 0026 C080      MOV RO,R2          ; LOAD LOW
0061 0028 0200      LI RO,>4200        ; ADD EXPONENT
      002A 4200
0062 002C 0460      B @FNRM           ; NORMALIZE
      002E 0000
0063      *      END
NO ERRORS,      NO WARNINGS

```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.SINF
OBJECT ACCESS NAME= ADHOC.OBJ.SINF
LISTING ACCESS NAME= ADHOC.LST.SINF
ERROR ACCESS NAME=
OPTIONS= BUNLST, DUNLST, MUNLST
MACRO LIBRARY PATHNAME=

```
0002      IDT 'SINF'
0003      *
0004      *      COSF      ;COSINE FUNCTION
0005      *      SINF      ;SINE FUNCTION
0006      *
0007      REF FUNFX      ;FIX ARGUMENTS
0008      REF FUNBK      ;BREAK ARGUMENTS
0009      REF PLYX,PLYXX ;EVALUATE POLYNOMIAL
0010      REF FPAC      ;FLOATING POINT ACCUMULATOR
0011      REF TEMP      ;3 WRD TEMPORARY STORAGE
0012      REF DS,DS1    ;3 WRD TEMPORARY STORAGE
0013      *
0014      DXOP LOADF,0    ;LOAD FPAC
0015      DXOP STORE,1    ;STORE FPAC
0016      DXOP FADD,2     ;ADD TO FPAC
0017      DXOP FSUB,3     ;SUBTRACT FROM FPAC
0018      DXOP FMUL,4     ;MULTIPLY FPAC
0019      DXOP FDIV,5     ;DIVIDE FPAC
0020      DXOP NEGATE,9    ;NEGATE FPAC
```

```

0022      *
0023      *          CALCULATE SINE/COSINE FUNCTIONS OF (ARG).
0024      *
0025      * CALLING SEQUENCE:
0026      *
0027      *          BL @SINF
0028      *          BL @COSF
0029      *
0030      *          IN  (R2) = ARG
0031      *          OUT (R2) = RESULT
0032      *
0033      * EXCEPTIONS AND CONDITIONS:
0034      *
0035      *          FLOATING POINT ERRORS
0036      * ENTRY POINT:
0037      *
0038      *          DEF SINF,COSF
0039      *
0040 0000 C28B COSF  MOV R11,R10
0041 0002 06A0      BL @FUNFX          FIX SIGN
0042      *          0004 0000
0043      *          JMP COSF1          0, RETURN 1
0044      *          LI R3,1          START WITH 1
0045      *          000A 0001
0046      *          JMP COSF2
0047      *
0048      *          COSF1  LOADF @COSC2  COS 0 = 1
0049      *          0010 006B'
0050      *          JMP COSF5          RETURN
0051      *
0052      *          SINF  MOV R11,R10
0053      *          0014 C28B      BL @FUNFX          FIX EXPONENT
0054      *          0016 06A0
0055      *          0018 0004'
0056      *          JMP COSF5          ;0, RETURN 0
0057      *          001A 101F      SLA R3,1          IF +: 0, ELSE 2
0058      *          001C 0A13
0059      *          COSF2  FMUL @COSC0  FPAC=FPAC*C1
0060      *          001E 2D20
0061      *          0020 0060'
0062      *          BL @FUNBK          BREAK INTO FRAC AND INT
0063      *          0022 06A0
0064      *          0024 0000
0065      *          A R3,R1
0066      *          0026 A043      SRL R1,1          QUADRANT 2 OR 4?
0067      *          0028 0911      JNC COSF3          N
0068      *          002A 1706      JNC COSF3          Y, GET 1-FRACTION
0069      *          002C 2C60      STORE @TEMP
0070      *          002E 0000
0071      *          0030 2C20      LOADF @COSC2  LOAD FPAC WITH 1
0072      *          0032 006B'
0073      *          0034 2CE0      FSUB @TEMP      SUBTRACT FRACTION
0074      *          0036 002E'
0075      *
0076      *          COSF3  SRL R1,1          EFFECTIVE QUAD 3 OR 4?
0077      *          0038 0911      JNC COSF4          N
0078      *          003A 1701      JNC COSF4          Y, NEGATE
0079      *          003C 2E40      NEGATE 0
0080      *          003E 2C60
0081      *          COSF4  STORE @DS          STORE IN DS
0082      *          0040 0000
0083      *          BL @PLYXX          EVALUATE POLYNOMIAL X*X
0084      *          0042 06A0
0085      *          0044 0000
0086      *          DATA COSC1
0087      *          0046 0066'
0088      *          0048 2C60      STORE @DS1      SAVE IN DS1

```

```

004A 0000
0071 004C 06A0      BL @PLYX      EVALUATE POLYNOMIAL X
004E 0000
0072 0050 007A'      DATA COSC3
0073 0052 2D60      FDIV @DS1      FPAC=FPAC/DS1
0054 004A'
0074 0056 2D20      FMUL @DS      FPAC=(FPAC/DS1)*DS
0058 0040'
0075      *
0076 005A 0202 COSF5  LI R2,FPAC      RETURN
005C 0000
0077 005E 045A      B *R10
0078      *
0079 0060 40A2 COSC0  DATA >40A2,>F983,>6E4E
0080 0066 0002 COSC1  DATA 2
0081 0068 4110 COSC2  DATA >4110,>0000,>0000
0082 006E 4273      DATA >4273,>4DCA,>815D
0083 0074 4411      DATA >4411,>7825,>55B4
0084 007A 0005 COSC3  DATA 5
0085 007C BE95      DATA >BE95,>3606,>2DEE
0086 0082 4041      DATA >4041,>E3F5,>31B8
0087 0088 C1C6      DATA >C1C6,>5036,>51D0
0088 008E 4311      DATA >4311,>B7C5,>5A8F
0089 0094 C3A9      DATA >C3A9,>3BA0,>C828
0090 009A 441B      DATA >441B,>70D4,>8BB5
0091      END
NO ERRORS,      NO WARNINGS

```


ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.SGRF
OBJECT ACCESS NAME= ADHOC.OBJ.SGRF
LISTING ACCESS NAME= ADHOC.LST.SGRF
ERROR ACCESS NAME=
OPTIONS= BUNLST, DUNLST, MUNLST
MACRO LIBRARY PATHNAME=

```

0002          IDT 'SQRF'
0003          *
0004          *
0005          *
0006          REF FUNFX          ; FIXES ARGUMENT
0007          DXOP LOADF,0        ; LOAD FPAC
0008          DXOP STORE,1        ; STORE FPAC
0009          DXOP FADD,2         ; ADD TO FPAC
0010          DXOP FMUL,4         ; MULTIPLY FPAC
0011          DXOP FDIV,5         ; DIVIDE FPAC
0012          2F80 ERROR EQU >2F80 ; XOP XX,14 (ERROR CALL)
0013          REF FPAC            ; FLOATING POINT ACCUMULATOR
0014          REF TEMP            ; 3 WRD TEMPORARY STORAGE
0015          REF DS              ; 3 WRD TEMPORARY STORAGE
0016          REF EVSFR           ; EXIT ADDRESS
0017          REF CB000
0018          *
0019          0004 SQRI EQU 4          ; £ OF NEWTON ITERATIONS
0020          * ABSTRACT:
0021          *
0022          * COMPUTE THE SQUARE ROOT OF N (R2) USING A
0023          * NEWTON ITERATION DEFINED AS FOLLOWS:
0024          *
0025          *  $X(I+1) = [X(I) + N / X(I)] / 2$ 
0026          *
0027          * ITERATE FOR SQRI TIMES. 3 ITERATIONS =
0028          * 6 DIGITS, 4 ITERATION = 11 DIGITS.
0029          *
0030          * CALLING SEQUENCE:
0031          *
0032          * B @SQRF
0033          *
0034          * IN (R0) = 3 WRD FLOATING POINT £
0035          * OUT (R2) = 3 WRD FLOATING POINT ROOT OF £
0036          *
0037          * NORMAL EXIT IS TO @EVSFR
0038          * ERROR EXIT IS THRU XOP XX.14
0039          *
0040          * EXCEPTIONS AND CONDITIONS:
0041          *
0042          * USES R0-R4,R11
0043          *
0044          * ERROR 25 = SQUARE ROOT OF NEGATIVE NUMBER
0045          *
0046          SQRF DEF SQRF          ; ENTRY POINT
0047          0000 06A0 BL @FUNFX    ; MOVE INTO FPAC
0048          0002 0000
0048          0004 1022 JMP SQRF3    ; ZERO, DONE
0049          0006 C0C3 MOV R3,R3     ; NEGATIVE?
0050          0008 1626 JNE ERR25    ; Y, ERROR
0051          *****
0052          * NEWTON'S ITERATION HAS QUADRATIC CONVERGENCE GIVEN
0053          * A GOOD FIRST GUESS. PUT FIRST GUESS IN FPAC.
0054          *****
0055          000A 2C60 STORE @DS      ; MOVE INTO DS
0056          000C 0000
0056          000E 0200 LI R0,>4000  ; GET 16^0 EXPONENT
0057          0010 4000
0057          0012 6040 S R0,R1      ; UNBIAS OLD EXPONENT
0058          0014 DB00 MOVB R0,@FPAC ; LOAD EXPONENT
    
```

```

0016 0000
0059 0018 2D20      FMUL @SQRC1      ; FPAC=N*C1
001A 0058'
0060 001C 2CA0      FADD @SQRC2
001E 005E'
0061 0020 0991      SRL R1,9          ; ODD EXPONENT?
0062 0022 1702      JNC SGRF1         ; N
0063 0024 2D20      FMUL @FP4         ; Y, MULTIPLY BY 4
0026 006A'
0064
0065      *
0065 0028' SGRF1 EQU $
0066 0028 06C1      SWPB R1           ; POSITION EXPONENT
0067 002A B801      AB R1,@FPAC       ; ADJUST EXPONENT
002C 0016'
0068 002F 0201      LI R1,SQRI        ; DO SQRI ITERATIONS
0030 0004
0069      *
0070      *****
0071      * ITERATION LOOP FOLLOWS.
0072      * DO X(I+1) = [X(I) + N / X(I)] / 2
0073      *****
0074 0032' SGRF2 EQU $
0075 0032 2C60      STORE @TEMP        ; MOVE FPAC TO TEMP
0034 0000
0076 0036 2C20      LOADF @DS
0038 000C'
0077 003A 2D60      FDIV @TEMP         ; DIVIDE..N/X(I)
003C 0034'
0078 003E 2CA0      FADD @TEMP         ; ADD TEMP..X(I)+N/X(I)
0040 003C'
0079 0042 2D20      FMUL @SQRC3        ; X 0.5
0044 0064'
0080 0046 0601      DEC R1             ; SQRI TIMES?
0081 0048 16F4      JNE SGRF2         ; N
0082      *****
0083      * GET ADDRESS OF RESULT AND RETURN TO EVAL
0084      *****
0085 004A' SGRF3 EQU $
0086 004A 0202      LI R2,FPAC         ; Y, RETURN
004C 002C'
0087 004E 44A0      SZC @C8000,*R2     SQR(0.0625) GIVES -VE VALUES
0050 0000
0088 0052 0460      B @EVSFR
0054 0000
0089      *
0090 0056 2F99      ERR25 DATA ERROR+25 ; SQUARE ROOT OF NEGATIVE NUMBER
0091      *
0092 0058 40E4      SQRC1 DATA >40E4,>F92A,>F9A8 ; 0.894 427
0093 005E 4039      SQRC2 DATA >4039,>3E4E,>F028 ; 0.223 607
0094 0064 4080      SQRC3 DATA >4080,>0000,>0000 ; 0.5
0095 006A 4140      FP4 DATA >4140,>0000,>0000 ; 4.0
0096      END
NO ERRORS,      NO WARNINGS
    
```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.EXTEND
OBJECT ACCESS NAME= ADHOC.OBJ.EXTEND
LISTING ACCESS NAME= ADHOC.LST.EXTEND
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

LINE	KEY	NAME
------	-----	------

0023	A	ADHOC.SRC.IOBITS =>ADHOC.SRC.IOBITS
------	---	--

```


0002          IDT 'EXTEND'
0003          *
0004          * DEFINE FLOATING POINT XOPS
0005          *
0006          DXOP LOADF,0          LOAD FPAC
0007          DXOP STORE,1         STORE FPAC
0008          DXOP FADD,2          ADD TO FPAC
0009          DXOP FSUB,3          SUBTRACT FROM FPAC
0010          DXOP FMUL,4          MULTIPLY FPAC
0011          DXOP FDIV,5          DIVIDE FPAC
0012          DXOP SCALE,6         SCALE FPAC
0013          DXOP NORMAL,7        NORMALIZE FPAC
0014          DXOP CLEAR,8         CLEAR FPAC
0015          DXOP NEGATE,9        NEGATE FPAC
0016          DXOP FLOATF,10       FLOAT FPAC
0017          DXOP EVFIX,11        EVALUATE AND FIX
0018          DXOP OUTFP,12        OUT FLOATING POINT &
0019          DXOP OUTINT,13        OUT INTEGER
0020          2F80 ERROR EQU >2F80      XOP XX,14 (ERROR CALL)
0021          2FA0 ERROR2 EQU ERROR+>20
0022          *
0023          COPY ADHOC.SRC.IOBITS
    
```

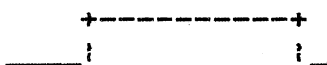
```

A0002      *
A0003      * THIS MODULE IS INCLUDED IN SOURCE MODULES
A0004      * BY USING A 'COPY' DIRECTIVE.
A0005      *
A0006      *
A0007      *****
A0008      *
A0009      * CRU DEFINITIONS
A0010      *
A0011      *****
A0012      0000 P10 EQU >0000 PARALLEL I/O
A0013      * PARALLEL I/O - READ BITS
A0014      0000 KDS EQU P10+0 KEYBOARD DATA STROBE
A0015      * EQU P10+1 UNUSED
A0016      0002 D1$SIZ EQU P10+2 DS01 SIZE (1=8",0=5.25")
A0017      0003 D1$DEN EQU P10+3 DS01 DENSITY (0=SD,1=DD)
A0018      0004 FDCINT EQU P10+4 FDC INTERRUPT-
A0019      0005 KBDINT EQU P10+5 KBD INTERRUPT-
A0020      0006 VDPINT EQU P10+6 VDP INTERRUPT-
A0021      0007 BUSINT EQU P10+7 BUS INTERRUPT-
A0022      0008 KEYBRD EQU P10+8 KEYBOARD DATA (8 BITS)
A0023      * PARALLEL I/O - WRITE BITS
A0024      0000 CLKLED EQU P10+0 CLOCK LED
A0025      0001 KBDACK EQU P10+1 KEYBOARD ACKNOWLEDGE-
A0026      0002 BUSACK EQU P10+2 BUS INTERRUPT RESET-
A0027      0003 BTENBL EQU P10+3 BUS TIMEOUT FLAG ENABLE
A0028      0004 DRVSIZ EQU P10+4 DRIVE SIZE (1=8",0=5.25")
A0029      0005 ROMON EQU P10+5 0=ROM ON,1=ROM OFF
A0030      0006 BELLON EQU P10+6 BELL ENABLE BIT
A0031      * EQU P10+7 UNUSED
A0032      *
A0033      * EQU >0020 UNUSED
A0034      0040 EIA02 EQU >0040 PRINTER HARDWARE BASE ADDRESS
A0035      * EQU >0060 UNUSED
A0036      * EQU >0080 UNUSED
A0037      * EQU >00A0 UNUSED
A0038      00C0 CASS02 EQU >00C0 CASSETTE HARDWARE BASE ADDRESS
A0039      00E0 DMAC EQU >00E0 DMAC HARDWARE BASE ADDRESS
A0040      *
A0041      * RESERVED OFF-BOARD CRU LOCATIONS
A0042      0200 PRMPOP EQU >200 FROM PROGRAMMER BOARD
A0043      * READ BITS
A0044      * EQU PRMPOP+0
A0045      * EQU PRMPOP+1
A0046      * EQU PRMPOP+2
A0047      * EQU PRMPOP+3
A0048      0204 PRORDY EQU PRMPOP+4
A0049      0205 PGM EQU PRMPOP+5
A0050      0206 PGMPUL EQU PRMPOP+6 * TEST
A0051      0207 V30 EQU PRMPOP+7
A0052      0208 EDATA EQU PRMPOP+8
A0053      * WRITE BITS
A0054      0200 PRORST EQU PRMPOP+0
A0055      0201 EPTYPE EQU PRMPOP+1
A0056      0204 VCCON EQU PRMPOP+4
A0057      *PGM EQU PRMPOP+5
A0058      0206 PROERR EQU PRMPOP+6
A0059      * EQU PRMPOP+7
A0060      *EDATA EQU PRMPOP+8
A0061      0210 EPADR EQU PRMPOP+16

```

```
A0062      *
A0063      *   CENTRONICS PARALLEL PRINTER PORT
A0064      *
A0065      *   BIT           0       1       2       3       4       5       6       7       8       9
A0066      *   READ
A0067      *   WRITE      D7      D6      D5      D4      D3      D2      D1      D0      D5
A0068      *                               (LSB)                               (MSB)
A0069      *
A0070      0400 PPRINT EQU  >400
A0071      *
A0072      0408 PPDS  EQU  PPRINT+8
A0073      *
A0074      *
A0075      0409 PPBUSY EQU  PPRINT+9
A0076      *
A0077      *****
A0078      *
A0079      *   MEMORY MAPPED I/O DEFINITIONS
A0080      *
A0081      *****
A0082      F100 MAPPER EQU  >F100
A0083      F100 M$REG0 EQU  MAPPER+0
A0084      F102 M$REG1 EQU  MAPPER+2
A0085      F104 M$REG2 EQU  MAPPER+4
A0086      F106 M$REG3 EQU  MAPPER+6
A0087      F108 M$REG4 EQU  MAPPER+8
A0088      F10A M$REG5 EQU  MAPPER+10
A0089      F10C M$REG6 EQU  MAPPER+12
A0090      F10E M$REG7 EQU  MAPPER+14
A0091      F110 M$REG8 EQU  MAPPER+16
A0092      F112 M$REG9 EQU  MAPPER+18
A0093      F114 M$REGA EQU  MAPPER+20
A0094      F116 M$REGB EQU  MAPPER+22
A0095      F118 M$REGC EQU  MAPPER+24
A0096      F11A M$REGD EQU  MAPPER+26
A0097      F11C M$REGE EQU  MAPPER+28
A0098      F11E M$REGF EQU  MAPPER+30
A0099      *
A0100      F120 VDP      EQU  >F120
A0101      F120 VRAM     EQU  VDP+0
A0102      F121 VDPREG  EQU  VDP+1
A0103      *
A0104      *   TEXT / GRAPHICS 1 MODE STORAGE
A0105      0400 NTBA     EQU  >400
A0106      0700 CTBA     EQU  >700
A0107      0800 PGBA     EQU  >800
A0108      0780 SNTBA    EQU  >780
A0109      0000 SPGBA    EQU  >000
A0110      *   GRAPHICS 2 MODE STORAGE
A0111      1800 NTBA1    EQU  >1800
A0112      2000 CTBA1    EQU  >2000
A0113      0000 PGTBA1   EQU  >0000
A0114      1800 SNTBA1   EQU  >1800
A0115      3800 SPGBA1   EQU  >3800
A0116      *
A0117      F140 FDC      EQU  >F140
A0118      *
0024      DEF  XFRPM,EXTNDY
0025      *
0026      REF  WPR2,EXTWP,C10,EXT$ID,LINE,CONFIG
```

DATA STROBE 

BUSY 

MEMORY MAPPER LOCATION
MAPPER REGISTERS 0..15

VDP VRAM ACCESS ADDRESS
VDP REGISTER ACCESS ADDRESS

NAME TABLE
COLOUR TABLE
PATTERN GENERATOR TABLE
SPRITE NAME TABLE
SPRITE PATTERN GENERATOR TBL.

NAME TABLE
COLOUR TABLE
PATTERN GENERATOR TABLE
SPRITE NAME TABLE
SPRITE PATTERN GENERATOR TBL.

TMS9909 FLOPPY DISC CONTROLLER

```

0027      *
0028      *   TRANSFER PHYSICAL MEMORY
0029      *
0030      *   BLWP @XFRPM
0031      *
0032      *   IN:  R1   PAGE £
0033      *         R2   PAGE INDEX (WORD ALIGNED)
0034      *         R3   TRANSFER LENGTH (BYTES)
0035      *         R4   LOAD POINT (WORD ALIGNED)
0036      *
0037 0000 0000 XFRPM DATA WPR2, $+2
0038 0004 C320 MOV @CONFIG, R12      GET CONFIGURATION FLAGS
      0006 0000
0039 0008 091C SRL R12, 1          MAPPER PRESENT ?
0040 000A 1722 JNC ERR47           N, ERROR
0041 000C 03A0 CKON                Y, ENABLE MAPPER
0042 000E 020C LI R12, M$REG2      POINT TO MAP REG 2
      0010 F104
0043 0012 C01C MOV *R12, R0        SAVE IT
0044 0014 C72D MOV @2*R1(R13), *R12 SET NEW PAGE
      0016 0002
0045 0018 C0AD MOV @2*R2(R13), R2   GET INDEX INTO PAGE
      001A 0004
0046 001C C0ED MOV @2*R3(R13), R3   GET £ BYTES TO XFER
      001E 0006
0047 0020 C12D MOV @2*R4(R13), R4   GET LOAD ADDRESS
      0022 0008
0048 0024 0242 ANDI R2, >0FFF      ENSURE INDEX IS VALID
      0026 0FFF
0049 0028 0583 INC R3              FORCE TRANSFER LENGTH TO WORDS
0050 002A 0913 SRL R3, 1
0051      *
0052 002C CD22 XF1 MOV @>2000(R2), *R4+ COPY THE WORD
      002E 2000
0053 0030 0603 DEC R3              COUNT IT
0054 0032 1307 JEQ XF2            0, DONE
0055 0034 05C2 INCT R2            UPDATE INDEX
0056 0036 0282 CI R2, >1000       OFF PAGE?
      0038 1000
0057 003A 1AFB JL XF1             N, LOOP
0058 003C 04C2 CLR R2            Y, RESET IT
0059 003F 0592 INC *R2           GET NEXT PAGE
0060 0040 10F5 JMP XF1           LOOP
0061      *
0062 0042 03C0 XF2 CKOF           DISABLE MAPPER
0063 0044 C700 MOV R0, *R12        RESTORE MAP REG 2
0064 0046 0380 RTWP              EXIT
    
```



```

0066      *
0067      *      EXTENDED COMMAND HANDLER
0068      *
0069      *
0070      *      EXTENDED COMMAND WORKSPACE :-
0071      *      IN      R8      PBC
0072      *      R13-R15  RETURN CONTEXT
0073      *
0074      *      NORMAL EXIT :
0075      *      RTWP
0076      *
0077      *      ERROR EXIT :
0078      *      LI      R0,????      ERROR & (RIGHT JUSTIFIED)
0079      *      INCT R14
0080      *      RTWP
0081      *
0082      *      EXTENDED COMMAND HANDLER FORMAT
0083      *
0084      *      1000 *COMMAND_NAME parameters
0085      *      ~~~~~
0086      *      EPROM FORMAT
0087      *
0088      *      DATA >94C2      IDENTIFIER WORD
0089      *      DATA LINK1      LINK TO NEXT DIR. ENTRY
0090      *      DATA ENTRY1      ENTRY PT. OF COMMAND_1 ROUTINE
0091      *      TEXT 'COMMAND_1'  NAME
0092      *      BYTE 0            TERMINATOR FOR NAME
0093      *      LINK1 DATA LINK2  LINK TO NEXT DIR. ENTRY
0094      *      DATA ENTRY2      ENTRY PT. OF COMMAND_2 ROUTINE
0095      *      TEXT 'COMMAND_2'  NAME
0096      *      BYTE 0            TERMINATOR FOR NAME
0097      *      LINK2 DATA 0      LINK TERMINATOR !!
0098      *
0099      *      EXTENDED COMMAND HANDLERS ARE LOCATED ON 4K BYTE
0100      *      BOUNDARIES STARTING FROM >010000 AND ARE MAPPED INTO
0101      *      THE CPU ADDRESS SPACE FROM >2000 TO >3FFF. ALL EXTENDED
0102      *      COMMAND HANDLER EPROMS MUST BE INSTALLED BETWEEN
0103      *      >01 0000 AND >0F FFFF
0104      *
0105 0048 C060 EXTNDY MOV @CONFIG,R1      GET CONFIG FLAG
0106      004A 0006
0106 004C 0911      SRL R1,1      MAPPER PRESENT ?
0107 004E 1802      JOC EXT1      Y, GO LOOK FOR HANDLER
0108 0050 2FA0 ERR47 DATA ERROR2,47    REQ. H/W NOT FOUND
0109      *
0110 0054 0201 EXT1  LI R1,M$REG2      POINT TO MAPPER R2 (>2000)
0111      0056 F104
0111 0058 0206      LI R6,M$REG3      POINT TO MAPPER R3 (>3000)
0112      005A F106
0112 005C C3D1      MOV *R1,R15      SAVE ITS CONTENTS
0113 005E C396      MOV *R6,R14      SAVE ITS CONTENTS
0114 0060 C460      MOV @C10,*R1      START FROM ADDRESS 010000 HEX
0115      0062 0000
0115 0064 C288      MOV R8,R10      SAVE PBC
0116 0066 03A0      CKON      MAPPER ON
0117      *
0118 0068 0202 EXT2  LI R2,>2000      POINT TO START OF WINDOW
0119      006A 2000
0119 006C 8832      C *R2+,@EXT$ID    EXPANSION FOUND?
0120      006E 0000
    
```

0120	0070	1620		JNE	EXT7	N, TRY NEXT 4K
0121	0072	C591		MOV	*R1,*R6	Y, SET UP 2ND HALF OF WINDOW
0122	0074	0596		INC	*R6	
0123	0076	C0F2	EXT3	MOV	*R2+,R3	Y, GET LINK POINTER
0124	0078	131C		JEQ	EXT7	0, END OF TABLE FOR THIS ROM
0125	007A	C132		MOV	*R2+,R4	GET ITS ENTRY POINT
0126	007C	D038	EXT4	MOVB	*R8+,R0	GET SEARCH STRING BYTE
0127	007E	1304		JEQ	EXT5	0, DONE SEARCH
0128	0080	9032		CB	*R2+,R0	MATCH ?
0129	0082	13FC		JEQ	EXT4	Y, CONTINUE
0130	0084	C0B3	EXT4A	MOV	R3,R2	N, PICK UP LINK
0131	0086	10F7		JMP	EXT3	AND LOOP
0132	0088	D012	EXT5	MOVB	*R2,R0	SOURCE NULL, MATCH ?
0133	008A	16FC		JNE	EXT4A	N, CONTINUE SEARCH
0134	008C	0203	EXT6	LI	R3,EXTWP	MATCH, SET EXTERNAL WP
	008E	0000				
0135	0090	C8C8		MOV	R8,@2*B(R3)	COPY OVER R8
	0092	0010				
0136	0094	0403		BLWP	R3	EXECUTE HANDLER
0137	0096	1008		JMP	EXT6A	<NORMAL RETURN>
0138	0098	0202		LI	R2,ERROR2	<ERROR RETURN>
	009A	2FA0				
0139	009C	C0D3		MOV	*R3,R3	PICK UP ERR &
0140	009E	0243		ANDI	R3,>003F	ISOLATE ERROR CODE
	00A0	003F				
0141	00A2	1601		JNE	EXT6B	NOT 0, OK
0142	00A4	0703		SETO	R3	0, MAKE ILLEGAL
0143	00A6	0442	EXT6B	B	R2	CALL THE ERROR HANDLER
0144			*			
0145	00A8	03C0	EXT6A	CKOF		
0146	00AA	C44F		MOV	R15,*R1	RESTORE MAPPER REGISTER
0147	00AC	C58E		MOV	R14,*R6	RESTORE MAPPER REGISTER
0148	00AE	0460		B	@LINE	EXIT TO NEXT LINE
	00B0	0000				
0149	00B2	0591	EXT7	INC	*R1	MOVE TO NEXT 4K
0150	00B4	D0A1		MOVB	@1(R1),R2	HAVE WE REACHED 10 0000
	00B6	0001				
0151	00B8	16D7		JNE	EXT2	N, LOOK FOR NEXT EPROM
0152			*			
0153			*			
0154			*			
0155	00BA	03C0	ERR41	CKOF		TURN MAPPER OFF
0156	00BC	C44F		MOV	R15,*R1	RESTORE MAPPER REGISTER
0157	00BE	C58E		MOV	R14,*R6	RESTORE MAPPER REGISTER
0158	00C0	2FA0		DATA	ERROR2,41	EXPANSION EPROM NOT FOUND
0159				END		

NO ERRORS, NO WARNINGS

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.TEXT
OBJECT ACCESS NAME= ADHOC.OBJ.TEXT
LISTING ACCESS NAME= ADHOC.LST.TEXT
ERROR ACCESS NAME=
OPTIONS= BUNLST, TUNLST, DUNLST, MUNLST
MACRO LIBRARY PATHNAME=

LINE	KEY	NAME
------	-----	------

0003	A	ADHOC.SRC.IOBITS =>ADHOC.SRC.IOBITS
------	---	--

0002
0003

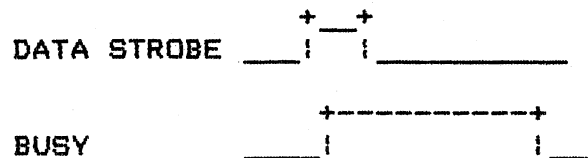
IDT 'TEXT'
COPY ADHOC.SRC.IOBITS

```

A0002      *
A0003      * THIS MODULE IS INCLUDED IN SOURCE MODULES
A0004      * BY USING A 'COPY' DIRECTIVE.
A0005      *
A0006      *
A0007      *****
A0008      *
A0009      * CRU DEFINITIONS
A0010      *
A0011      *****
A0012      0000 PIO EQU >0000 PARALLEL I/O
A0013      * PARALLEL I/O - READ BITS
A0014      0000 KDS EQU PIO+0 KEYBOARD DATA STROBE
A0015      * EQU PIO+1 UNUSED
A0016      0002 D1$SIZ EQU PIO+2 DS01 SIZE (1=8",0=5.25")
A0017      0003 D1$DEN EQU PIO+3 DS01 DENSITY (0=SD,1=DD)
A0018      0004 FDCINT EQU PIO+4 FDC INTERRUPT-
A0019      0005 KBDINT EQU PIO+5 KBD INTERRUPT-
A0020      0006 VDPINT EQU PIO+6 VDP INTERRUPT-
A0021      0007 BUSINT EQU PIO+7 BUS INTERRUPT-
A0022      0008 KEYBRD EQU PIO+8 KEYBOARD DATA (8 BITS)
A0023      * PARALLEL I/O - WRITE BITS
A0024      0000 CLKLED EQU PIO+0 CLOCK LED
A0025      0001 KBDACK EQU PIO+1 KEYBOARD ACKNOWLEDGE-
A0026      0002 BUSACK EQU PIO+2 BUS INTERRUPT RESET-
A0027      0003 BTENBL EQU PIO+3 BUS TIMEOUT FLAG ENABLE
A0028      0004 DRVSIZ EQU PIO+4 DRIVE SIZE (1=8",0=5.25")
A0029      0005 ROMON EQU PIO+5 0=ROM ON,1=ROM OFF
A0030      0006 BELLON EQU PIO+6 BELL ENABLE BIT
A0031      * EQU PIO+7 UNUSED
A0032      *
A0033      * EQU >0020 UNUSED
A0034      0040 EIA02 EQU >0040 PRINTER HARDWARE BASE ADDRESS
A0035      * EQU >0060 UNUSED
A0036      * EQU >0080 UNUSED
A0037      * EQU >00A0 UNUSED
A0038      00C0 CASS02 EQU >00C0 CASSETTE HARDWARE BASE ADDRESS
A0039      00E0 DMAC EQU >00E0 DMAC HARDWARE BASE ADDRESS
A0040      *
A0041      * RESERVED OFF-BOARD CRU LOCATIONS
A0042      0200 PRMPOP EQU >200 PROM PROGRAMMER BOARD
A0043      * READ BITS
A0044      * EQU PRMPOP+0
A0045      * EQU PRMPOP+1
A0046      * EQU PRMPOP+2
A0047      * EQU PRMPOP+3
A0048      0204 PRORDY EQU PRMPOP+4
A0049      0205 PGM EQU PRMPOP+5
A0050      0206 PGMFUL EQU PRMPOP+6 * TEST
A0051      0207 V30 EQU PRMPOP+7
A0052      0208 EDATA EQU PRMPOP+8
A0053      * WRITE BITS
A0054      0200 PRORST EQU PRMPOP+0
A0055      0201 EPTYPE EQU PRMPOP+1
A0056      0204 VCCON EQU PRMPOP+4
A0057      *PGM EQU PRMPOP+5
A0058      0206 PROERR EQU PRMPOP+6
A0059      * EQU PRMPOP+7
A0060      *EDATA EQU PRMPOP+8
A0061      0210 EPADR EQU PRMPOP+16
    
```

```

A0062      *
A0063      *   CENTRONICS PARALLEL PRINTER PORT
A0064      *
A0065      *   BIT       0       1       2       3       4       5       6       7       8       9
A0066      *   READ
A0067      *   WRITE    D7      D6      D5      D4      D3      D2      D1      D0      D5
A0068      *                               (LSB)                               (MSB)
A0069      *
A0070      0400 PPRINT EQU  >400
A0071      *
A0072      0408 PPDS  EQU  PPRINT+8
A0073      *
A0074      *
A0075      0409 PPBUSY EQU  PPRINT+9
A0076      *
A0077      *****
A0078      *
A0079      *   MEMORY MAPPED I/O DEFINITIONS
A0080      *
A0081      *****
A0082      F100 MAPPER EQU  >F100
A0083      F100 M$REG0 EQU  MAPPER+0
A0084      F102 M$REG1 EQU  MAPPER+2
A0085      F104 M$REG2 EQU  MAPPER+4
A0086      F106 M$REG3 EQU  MAPPER+6
A0087      F108 M$REG4 EQU  MAPPER+8
A0088      F10A M$REG5 EQU  MAPPER+10
A0089      F10C M$REG6 EQU  MAPPER+12
A0090      F10E M$REG7 EQU  MAPPER+14
A0091      F110 M$REG8 EQU  MAPPER+16
A0092      F112 M$REG9 EQU  MAPPER+18
A0093      F114 M$REGA EQU  MAPPER+20
A0094      F116 M$REGB EQU  MAPPER+22
A0095      F118 M$REGC EQU  MAPPER+24
A0096      F11A M$REGD EQU  MAPPER+26
A0097      F11C M$REGE EQU  MAPPER+28
A0098      F11E M$REGF EQU  MAPPER+30
A0099      *
A0100      F120 VDP    EQU  >F120
A0101      F120 VRAM   EQU  VDP+0
A0102      F121 VDPREG EQU  VDP+1
A0103      *
A0104      * TEXT / GRAPHICS 1 MODE STORAGE
A0105      0400 NTBA   EQU  >400
A0106      0700 CTBA   EQU  >700
A0107      0800 PGBA   EQU  >800
A0108      0780 SNTBA  EQU  >780
A0109      0000 SPGBA  EQU  >000
A0110      * GRAPHICS 2 MODE STORAGE
A0111      1800 NTBA1  EQU  >1800
A0112      2000 CTBA1  EQU  >2000
A0113      0000 PGTBA1 EQU  >0000
A0114      1B00 SNTBA1 EQU  >1B00
A0115      3800 SPGBA1 EQU  >3800
A0116      *
A0117      F140 FDC     EQU  >F140
A0118      *
      0004      DEF    B4A, B0D, B1B, B2E, B56, B62, CRDELY
      0005      DEF    IDRUN, MBEGN, MCRLF, MPRDY, STRTC, BOO
      0006      DEF    EXT$ID, CPYST, B3C, B3A, BFA, BF1, B4C, B47
    
```



MEMORY MAPPER LOCATION
 MAPPER REGISTERS 0..15

VDP VRAM ACCESS ADDRESS
 VDP REGISTER ACCESS ADDRESS

NAME TABLE
 COLOUR TABLE
 PATTERN GENERATOR TABLE
 SPRITE NAME TABLE
 SPRITE PATTERN GENERATOR TBL.

NAME TABLE
 COLOUR TABLE
 PATTERN GENERATOR TABLE
 SPRITE NAME TABLE
 SPRITE PATTERN GENERATOR TBL.

TMS9909 FLOPPY DISC CONTROLLER

```

0007 DEF CASRDY, CRLF1T, CRLF2T, SYSERR
0008 DEF ERR00T, ERR01T, ERR02T, ERR03T, ERR04T
0009 DEF ERR05T, ERR06T, ERR07T, ERR08T, ERR09T
0010 DEF ERR10T, ERR11T, ERR12T, ERR13T, ERR14T
0011 DEF ERR15T, ERR16T, ERR17T, ERR18T, ERR19T
0012 DEF ERR20T, ERR21T, ERR22T, ERR23T, ERR24T
0013 DEF ERR25T, ERR26T, ERR27T, ERR28T, ERR29T
0014 DEF ERR30T, ERR31T, ERR32T, ERR33T, ERR34T
0015 DEF ERR35T, ERR36T, ERR37T, ERR38T, ERR39T
0016 DEF ERR40T, ERR41T, ERR42T, ERR43T, ERR44T
0017 DEF ERR45T, ERR46T, ERR47T, ERR48T, ERR49T
0018 DEF FP5E5, TAPERR, AUTORN, INM1, INM2, INM3
0019 DEF TIMC, TMPBUF, TBEND
0020 DEF C0004, C4600, C4A00, B3F, B7F
0021 DEF BFF, B2D, B45, B31, B2A, B40, CFO, C7F, CFF80
0022 DEF BADTER, BADADR
0023 DEF BELL, SPACES, SPACE2, BPMSG
0024 DEF LOGON, SPR, REGSTR, ASKBP, EXMSG, SPBP
0025 DEF STPMSG, SYMBLC
0026 DEF PROMPT, WS, EQU5GN
0027 DEF START, PADIT, SPACES, PGMFND, IDTEG
0028 *
0029 REF P$EUS, P$FNS, P$GSS, P$UFT, P$IOB, CRASH$, WP10L
0030 REF CONFIG, PRAM, BRAM, MEMSIZ, DEVTBL, DTEND
0031 REF SRATE, NEWP, SETVEC, C1, IOB
0032 *
0033 * DECREMETER DECREMENTED EVERY 4TH CLKOUT CYCLE RUNNING
0034 * AT 3 MHZ
0035 *
0036 0000 1D4C TIMC DATA 7500 = 10 MS FOR DECREMETER
0037 *
0038 0002 4A00 C4A00 DATA >4A00
0039 0004 4600 C4600 DATA >4600
0040 0006 0004 C0004 DATA 4
0041 A5A5 IDRUN EQU >A5A5
0042 0008 A5A5 STRTC DATA IDRUN AUTO RUN CODE
0043 000A 0019 CRDELY DATA 25 250MS CR DELAY COUNT
0044 000C 94C2 EXT$ID DATA >94C2 EXTENDED CMD HEADER ID
0045 000E' B45 EQU $
0046 000E 457A FP5E5 DATA >457A, >1200, >0000 FLOATING POINT 500,000
0047 0014 00F0 CFO DATA >00F0
0048 0016 007F C7F DATA >007F
0049 0018' BFF EQU $
0050 0018 FF80 CFF80 DATA >FF80
0051 001A EVEN
0052 001A 47 B47 BYTE >47 '!' BLUE ON CYAN COL. CODE
0053 001B 0D PGMFND BYTE >0D, >0A
0054 001D 46 TEXT 'Found "'
0055 0024 00 BYTE 0
0056 *
0057 0025 4C B4C BYTE >4C
0058 0026 31 B31 BYTE >31 '1'
0059 0027 1B B1B BYTE >1B 'ESC'
0060 0028 FA BFA BYTE >FA =16MS FOR 9902 TIMER
0061 0029 3C B3C BYTE >3C '<'
0062 002A 4A B4A BYTE >4A 'J'
0063 002B 56 B56 BYTE >56 'V'
0064 002C 62 B62 BYTE >62 'b'
0065 002D F1 BF1 BYTE >F1
0066 0017' B7F EQU C7F+1 CURSOR CHARACTER
    
```

```

0067      *
0068      002E' B0D      EQU  $
0069 002E  0D  MBEGN    BYTE >0D,>0A      ; PROMPT
0070 0030  43          TEXT 'CORTEX BASIC Rev'
0071      0040' B2E      EQU  $
0072 0040  2E          TEXT ' 1.1 '
0073 0047  88          BYTE >88
0074 0048  20          TEXT ' 1982'
0075 004D  0D  MPRDY    BYTE >0D,>0A
0076 004F  2A          TEXT '*Ready'
0077 0055  0D  MCRLF    BYTE >0D,>0A
0078 0057  00  B00      BYTE 0
0079      *
0080 0058  0D  TAPERR    BYTE >0D,>0A
0081 005A  2A          TEXT '** Tape read error'
0082 006C  20  CRLF2T    TEXT -' ** '
0083 0070  0D  CASRDY    BYTE >0D,>0A
0084 0072  43          TEXT 'Cassette ready '
0085      0081' B3F      EQU  $
0086 0081  3F          TEXT -'? (Y/N) '
0087 0089  0D  AUTDRN    BYTE >0D,>0A
0088 008B  41          TEXT 'Auto'
0089      008F' B2D      EQU  $
0090 008F  2D          TEXT -'-Run ? (Y/N) '
0091 009C  0D  CRLF1T    BYTE >0D,>0A
0092      009E' B2A      EQU  $
0093 009E  2A          TEXT -'** '
0094 00A1  3F  INM2      TEXT '?'
0095 00A2  3F  INM1      TEXT '?'
0096 00A4  00          BYTE 0
0097      00A5' B3A      EQU  $
0098 00A5  3A  INM3      TEXT ':'
0099 00A7  00          BYTE 0
  
```



```

0101 *****
0102 *
0103 * MONITOR MESSAGES
0104 *
0105 00A8 07 BELL BYTE >07,0
0106 00AA EVEN
0107 00AA 0D0A PADIT DATA >0D0A
0108 00AC 20 SPACE8 TEXT ' ' !!! MUST BE WORD ALIGNED !!!
0109 00AF 20 SPACE5 TEXT ' '
0110 00B0 20 TEXT ' '
0111 00B2 20 SPACE2 TEXT ' '
0112 00B3 20 TEXT ' '
0113 00B4 00 BYTE 0
0114 00B5 0D BPMSG BYTE >D,>A,7
0115 00B8 42 TEXT 'BP'
0116 00BA 00 BYTE 0
0117 00BB 0D LOGON BYTE >D,>A
0118 00BD 4D TEXT 'Monitor Rev. 1.1'
0119 00CE 88 BYTE >88
0120 00CF 20 TEXT '1982'
0121 00D4 00 BYTE 0
0122 00D5 20 SPR TEXT 'R'
0123 00D6' REGSTR EQU $-1
0124 00D7 00 BYTE 0
0125 00D8 20 ASKBP TEXT 'Set BPs?'
0126 00E1 00 BYTE 0
0127 00E2 20 EXMSG TEXT 'Execute'
0128 00EA 00 BYTE 0
0129 00EB 20 SPBP TEXT 'BP'
0130 00EE 00 BYTE 0
0131 00EF 0D STPMSG BYTE >D,>A
0132 00F1 53 TEXT 'SS:-'
0133 00F5 07 BYTE 7,0
0134 00F7' B40 EQU $
0135 00F7 40 SYMBLC TEXT '@>'
0136 00F9 00 BYTE 0
0137 00FA 0D PROMPT BYTE >D,>A
0138 00FC 5B TEXT '['
0139 00FE 00 BYTE 0
0140 00FF 57 WS TEXT 'WP='
0141 0102 00 BYTE 0
0142 0103 50 TEXT 'PC='
0143 0106 00 BYTE 0
0144 0107 53 TEXT 'ST'
0145 0109 3D EQU8GN TEXT '='
0146 010A 00 BYTE 0
0147 010B 49 IDTEG TEXT 'IDT='
0148 010F 00 BYTE 0
0149 0110 49 BADTER TEXT -'Illegal terminator'
0150 0122 49 BADADR TEXT -'Invalid address'

```

```

0152 *****
0153 *****
0154 *****
0155 ***
0156 ***
0157 ***
0158 ***
0159 *** THE AREA BETWEEN 'TMPBUF' AND 'TBEND' IS USED ***
0160 *** DURING NORMAL PROGRAM EXECUTION. ACCESS TO THIS ***
0161 *** AREA MUST THEREFORE BE MADE WITH THE EPROM TURNED ***
0162 *** ON. ***
0163 ***
0164 *****
0165 *****
0166 *****
0167 *
0168 0131' TMPBUF EQU $ <===== START OF SCRATCH AREA
0169 *
0170 0132 0000 PTRTBL DATA P$IOB, P$EUS, P$FNS, P$GSS, P$UFT
0171 *****
0172 *
0173 * SYSTEM COLDSTART *
0174 *
0175 *****
0176 *
0177 * THIS ROUTINE MUST :-
0178 * 1. CHECKSUM THE EPROMS TO MAKE SURE THEY ARE OK
0179 * 2. COPY THE EPROMS INTO RAM AND TURN THEM OFF
0180 * 3. RESET THE KEYBOARD INTERRUPT LATCH
0181 * 4. INITIALISE THE MEMORY MAPPER REGISTERS
0182 * 5. AUTOSIZE THE RAM
0183 *
0184 *
0185 *****
0186 *
0187 * CHECKSUM THE EPROMS AND GENERATE A SYSTEM
0188 * CRASH IF THIS FAILS. THIS ALSO COPIES THE
0189 * EPROMS INTO RAM.
0190 *
0191 *****
0192 013C' START EQU $ RESET ENTRY POINT
0193 013C 04C1 CLR R1 START FROM 0
0194 013E 0202 LI R2,CHKWRD POINT TO THE CHECKWORD
0195 0140 016E'
0196 0142 C002 MOV R2,R0 INITIALISE CHECKSUM
0197 0144 C101 LO MOV R1,R4 SAVE THE COPY POINTER
0198 0146 0244 ANDI R4,>FFFE KILL LS BIT
0199 0148 FFFE
0200 014A B0B4 C R4,R2 ARE WE AT THE CHECKWORD?
0201 014C 1305 JEQ L2 Y, DONT ADD IT IN
0202 014E BB11 AB *R1,@WP10L ADD INTO R0 LSB
0203 0150 0000
0204 0152 1C01 JOP L1 ODD PARITY, MODIFY CHECKWORD
0205 0154 2B01 XOR R1,R0 MODIFY THE CHECKSUM
0206 0156 0B00 L1 SRC R0,0 FRIG THE CHECKWORD
0207 0158 DC51 L2 MOVB *R1,*R1+ COPY THE ROM WORD INTO RAM
0208 015A 02B1 CI R1,>6000 DONE?
0209 015C 6000
0210 015E 1AF2 JL L0 N, LOOP
0211 0160 04C1 CLR R1

```

```

0208          ASMIF PIO
0209          LI R12,2*PIO          POINT TO PARALLEL I/O
0210          ASMELS
0211 0162 04CC          CLR R12
0212          ASMEND
0213 0164 8480          C R0,*R2          DOES IT MATCH?
0214 0166 1304          JEQ RAMST          Y, GO TURN ROM OFF <<<<<< ✓
0215 0168 1D06          SBO BELLON-PIO    N, TURN THE BELL ON
0216 016A 0460          B @CRASH$          GO DIE!!!!
          016C 0000
0217 016E 1B70          CHKWRD DATA >1B70          REV 1.1 CHECKSUM WORD
0218          *****
0219          *
0220          *          RESET THE I/O AND TURN THE ROMS OFF          *
0221          *
0222          *****
0223 0170 0200          RAMST LI R0,>0360          R0='RSET'
          0172 0360
0224 0174 0201          LI R1,>1D00+(KBDACK-PIO) R1='SBO KBDACK-PIO'
          0176 1D01
0225 0178 0202          LI R2,>1D00+(ROMON-PIO) R2='SBO ROMON-PIO'
          017A 1D05
0226 017C 0203          LI R3,>045B          R3='RT'
          017E 045B
0227 0180 0680          BL R0          TURN OFF EPROMS
0228          *****
0229          *
0230          *          NOW INITIALISE THE MAPPER & SET CONFIG FLAGS          *
0231          *
0232          *****
0233 0182 0207          LI R7,>0001          SET FOR MAPPER PRESENT
          0184 0001
0234 0186 020C          LI R12,MAPPER          POINT TO THE MAPPER
          0188 F100
0235 018A 0708          SETMAP SETD R8          FUDGE FOR NEXT INSTRUCTION
0236 018C 0588          SETMAP INC R8          NEXT REGISTER
0237 018E CF08          MOV R8,*R12+          INIT. REGISTER
0238 0190 06C8          SWPB R8          TEST JUST IT'S MSB
0239 0192 922C          CB @-1(12),R8          WAS IT WRITTEN OK?
          0194 FFFF
0240 0196 1301          JEQ MAPOK          Y, LEAVE MAPPER FLAG
0241 0198 04C7          CLR R7          N, RESET MAPPER FLAG
0242 019A 06C8          MAPOK SWPB R8          RESTORE R8
0243 019C 028C          CI R12,M$REGF          HAVE WE DONE WITH THE MAPPER?
          019E F11E
0244 01A0 12F5          JLE SETMAP          N, LOOP
0245 01A2 C807          NOPOP MOV R7,@CONFIG          SETUP CONFIGURATION FLAGS
          01A4 0000
0246          *****
0247          *
0248          *          AUTO-SIZE RAM          *
0249          *
0250          *****
0251 01A6 02A2          STWP R2          GET START OF AUTOSIZE
0252 01A8 0204          LI R4,>AAAA          SET TEST PATTERN
          01AA AAAA
0253 01AC 0205          LI R5,PRAM          SET PRAM POINTER
          01AE 0000
0254 01B0 0642          START1 DECT R2          BACKUP POINTER
0255 01B2 C484          MOV R4,*R2          WRITE

```

```

0256 01B4 8484      C      R4,*R2      D.K. ?
0257 01B6 1605      JNE  START6      N, END OF RAM
0258 01B8 04D2      CLR  *R2      CLEAR RAM
0259 01BA 0282      CI   R2,BRAM     DOWN TO INTERPRETER
        01BC 0000
0260 01BE 1BF8      JH   START1      N, CONTINUE
0261 01C0 0642      DECT R2
0262
0263      *
0264      *      INITIALIZE SYSTEM POINTERS      *
0265      *
0266      *
0267      *
0268      *      R2=BRAM-2
0269      *      R5=PRAM
0270      *
0271 01C2 0201      START6 LI  R1,IOB      GET POINTER
        01C4 0000
0272 01C6 0205      LI   R5,PTRTBL     REF THE SYSTEM PTR TABLE
        01C8 0132
0273 01CA C802      MOV  R2,@MEMSIZ     ;SAVE MEMORY SIZE
        01CC 0000
0274 01CE 0222      AI   R2,1024       ;RESERVE ROOM FOR HEADER BLOC
        01D0 0400
0275      *
0276 01D2 CC75      MOV  *R5+,*R1+     ;SAVE IOB
0277 01D4 CC75      MOV  *R5+,*R1+     ;SAVE EUS
0278 01D6 CC75      MOV  *R5+,*R1+     ;SAVE FNS
0279 01D8 CC75      MOV  *R5+,*R1+     ;SAVE GSS
0280 01DA CC55      MOV  *R5,*R1+     ;SAVE UFT
0281 01DC CC42      MOV  R2,*R1+     ;SAVE BUS
0282 01DE CC42      MOV  R2,*R1+     ;SAVE EORBUS
0283
0284      *
0285      *      INITIALISE ALL 9902'S FOR 300 BAUD OPERATION      *
0286      *
0287      *
0288 01E0 0207      LI   R7,DEVTBL     REF. DEVICE TABLE
        01E2 0000
0289 01E4 0202      LI   R2,>04D0     SET TO 300 BAUD
        01E6 04D0
0290      *
0291 01E8 0287      NXTDEV CI  R7,DTEND  DONE ALL ENTRIES
        01EA 0000
0292 01EC 140E      JHE  SETUP      Y, SET SYSTEM POINTERS
0293 01EE C337      MOV  *R7+,*R12     N, GET ENTRY
0294 01F0 2320      CDC  @C1,R12      LS BIT SET?
        01F2 0000
0295 01F4 16F9      JNE  NXTDEV      N, NOT A 9902
0296 01F6 024C      ANDI R12,>7FFE    KILL OFF LS & MS BITS
        01F8 7FFE
0297 01FA 1F12      TB   18          IS IT INSTALLED
0298 01FC 1303      JEQ  NODEV      N, DELETE TABLE ENTRY
0299 01FE 06A0      BL   @SRATE      Y, INITIALISE IT
        0200 0000
0300 0202 10F2      JMP  NXTDEV      DO NEXT DEVICE
0301      *
0302 0204 04E7      NODEV CLR  @-2(R7)  DELETE TABLE ENTRY
        0206 FFFE
0303 0208 10EF      JMP  NXTDEV      DO NEXT DEVICE

```

```
0304      *
0305      020A' SETUP EQU $
0306 020A 020B      LI R11,NEWP      SET EXIT TO 'NEW'
      020C 0000
0307 020E 0460      B @SETVEC      SET CRITICAL SYSTEM POINTERS
      0210 0000
```

ACCESS NAMES TABLE

PAGE 0001

SOURCE ACCESS NAME= ADHOC.SRC.LAST
OBJECT ACCESS NAME= ADHOC.OBJ.LAST
LISTING ACCESS NAME= ADHOC.LST.LAST
ERROR ACCESS NAME=
OPTIONS= BUNLST,DUNLST,MUNLST
MACRO LIBRARY PATHNAME=

LINE	KEY	NAME
------	-----	------

0003	A	ADHOC.SRC.CSET =>ADHOC.SRC.CSET
------	---	------------------------------------

0001

IDT 'LAST'

0002

DEF ZZZZZ\$, BRAM

0003

COPY ADHOC. SRC. CSET

```

A0003          OPTION BUNLST,DUNLST
A0004          DEF PCHTB,PCHTE,USERCS
A0005          *****
A0006          *
A0007          *          CHARACTER SET FOR THE 9928/9929 VDP
A0008          *
A0009          *          EACH CHARACTER IS STORED IN A PACKED 6 BYTE ENTRY
A0010          *          IN THE FOLLOWING MANNER:-
A0011          *
A0012          *          TABLE ENTRY
A0013          *          =====
A0014          *
A0015          *          BYTE 0    BYTE 1    BYTE 2    BYTE 3    BYTE 4    BYTES
A0016          *
A0017          *          AAAAAAAAA BBBB BBBB CCCCCCCC DDDDDDDD EEEEEEEE FFFFFFFF
A0018          *
A0019          *
A0020          *          CHARACTER CELL
A0021          *          =====
A0022          *
A0023          *          COLUMN      0 1 2 3 4 5
A0024          *
A0025          *          A A A A A A          ROW 0
A0026          *          A A B B B B          1
A0027          *          B B B B C C          2
A0028          *          C C C C C C          3
A0029          *          D D D D D D          4
A0030          *          D D E E E E          5
A0031          *          E E E E F F          6
A0032          *          F F F F F F          7
A0033          *
A0034          *
A0035          *****
A0036          *
A0037          *          BIT PATTERN          CHARACTER    HEX CODE
A0038          *
A0039          0000' PCHTB EQU $
A0040          *
A0041          *          CONTROL CHARACTERS
A0042          *
A0043          0000 934B          DATA >934B,>2449,>2300          NUL    >00
A0044          0006 6204          DATA >6204,>08EB,>E280          SOH    >01
A0045          000C 6204          DATA >6204,>08EB,>4280          STX    >02
A0046          0012 E20E          DATA >E20E,>20EB,>E280          ETX    >03
A0047          0018 E20E          DATA >E20E,>20EB,>4100          EDT    >04
A0048          001E E20D          DATA >E20D,>2AEB,>A180          ENG    >05
A0049          0024 624F          DATA >624F,>2428,>C280          ACK    >06
A0050          002A 628C          DATA >628C,>28D0,>4180          BEL    >07
A0051          0030 628D          DATA >628D,>A8D0,>2300          BS     >08 (CUR. LEFT)
A0052          0036 A38A          DATA >A38A,>0038,>4100          HT     >09 (CUR. RIGHT)
A0053          003C 820B          DATA >820B,>A8F0,>8200          LF     >0A (CUR. DOWN)
A0054          0042 8942          DATA >8942,>0038,>4100          VT     >0B (CUR. UP)
A0055          0048 E20F          DATA >E20F,>A8B0,>8200          FF     >0C (CLS & HOME)
A0056          004E 6208          DATA >6208,>1C49,>C480          CR     >0D (BEG. LINE)
A0057          0054 6205          DATA >6205,>0AEB,>A100          SO      >0E
A0058          005A 6204          DATA >6204,>0ED0,>4380          SI     >0F
A0059          0060 C28A          DATA >C28A,>3020,>8380          DLE    >10
A0060          0066 C28A          DATA >C28A,>3010,>4100          DC1    >11
A0061          006C C28B          DATA >C28B,>3210,>8380          DC2    >12
A0062          0072 C28B          DATA >C28B,>3210,>2300          DC3    >13
  
```


A0063 0078 C28A	DATA >C28A,>3218,>E080	DC4 >14
A0064 007E D2C9	DATA >D2C9,>0828,>C280	NAK >15
A0065 0084 6204	DATA >6204,>0AD0,>4100	SYN >16
A0066 008A E20F	DATA >E20F,>2AF0,>A300	ETB >17
A0067 0090 6206	DATA >6206,>1269,>6480	CAN >18
A0068 0096 E20E	DATA >E20E,>20EB,>6A80	EM >19
A0069 009C 6207	DATA >6207,>0AF0,>A300	SUB >1A
A0070 00A2 E20E	DATA >E20E,>26E0,>8180	ESC >1B
A0071 00AB E20D	DATA >E20D,>A890,>2300	FS >1C
A0072 00AE 6209	DATA >6209,>AC91,>A300	GS >1D
A0073 00B4 E28D	DATA >E28D,>A810,>2300	RS >1E (HOME)
A0074 00BA A28B	DATA >A28B,>9810,>2300	US >1F

A0075 *
 A0076 * PRINTABLE CHARACTERS
 A0077 *

A0078 00C0 0000	DATA >0000,>0000,>0000	SPACE >20
A0079 00C6 2082	DATA >2082,>0820,>0200	! >21
A0080 00CC 5145	DATA >5145,>0000,>0000	" >22
A0081 00D2 514F	DATA >514F,>94F9,>4500	& >23
A0082 00DB 21EA	DATA >21EA,>1C2B,>C200	\$ >24
A0083 00DE C321	DATA >C321,>0842,>6180	% >25
A0084 00E4 428A	DATA >428A,>10AA,>4680	& >26
A0085 00EA 1084	DATA >1084,>0000,>0000	' >27
A0086 00F0 2108	DATA >2108,>2081,>0200	(>28
A0087 00F6 2040	DATA >2040,>8208,>4200) >29
A0088 00FC 22A7	DATA >22A7,>3E72,>A200	* >2A
A0089 0102 0082	DATA >0082,>3E20,>8000	+ >2B
A0090 0108 0000	DATA >0000,>0020,>8400	, >2C
A0091 010E 0000	DATA >0000,>3E00,>0000	- >2D
A0092 0114 0000	DATA >0000,>0000,>0200	. >2E
A0093 011A 0021	DATA >0021,>0842,>0000	/ >2F
A0094 0120 7229	DATA >7229,>AACA,>2700	0 >30
A0095 0126 2182	DATA >2182,>0820,>8700	1 >31
A0096 012C 7220	DATA >7220,>8C42,>0F80	2 >32
A0097 0132 7220	DATA >7220,>8C0A,>2700	3 >33
A0098 0138 10C5	DATA >10C5,>24F8,>4100	4 >34
A0099 013E FA0F	DATA >FA0F,>020A,>2700	5 >35
A0100 0144 3908	DATA >3908,>3C8A,>2700	6 >36
A0101 014A FB21	DATA >FB21,>0841,>0400	7 >37
A0102 0150 722B	DATA >722B,>9C8A,>2700	8 >38
A0103 0156 722B	DATA >722B,>9E08,>4E00	9 >39
A0104 015C 0002	DATA >0002,>0020,>0000	: >3A
A0105 0162 0002	DATA >0002,>0020,>8400	; >3B
A0106 0168 1084	DATA >1084,>2040,>8100	< >3C
A0107 016E 000F	DATA >000F,>80FB,>0000	= >3D
A0108 0174 4081	DATA >4081,>0210,>8400	> >3E
A0109 017A 7221	DATA >7221,>0820,>0200	? >3F
A0110 0180 722B	DATA >722B,>AABA,>0780	@ >40
A0111 0186 722B	DATA >722B,>BE8A,>2880	A >41
A0112 018C F124	DATA >F124,>9C49,>2F00	B >42
A0113 0192 722B	DATA >722B,>2082,>2700	C >43
A0114 0198 F124	DATA >F124,>9249,>2F00	D >44
A0115 019E FA08	DATA >FA08,>3C82,>0F80	E >45
A0116 01A4 FA08	DATA >FA08,>3C82,>0800	F >46
A0117 01AA 7A08	DATA >7A08,>209A,>2780	G >47
A0118 01B0 8A28	DATA >8A28,>BE8A,>2880	H >48
A0119 01B6 7082	DATA >7082,>0820,>8700	I >49
A0120 01BC 0820	DATA >0820,>820A,>2700	J >4A
A0121 01C2 8A4A	DATA >8A4A,>30A2,>4880	K >4B
A0122 01C8 8208	DATA >8208,>2082,>0F80	L >4C

A0123 01CE 8B6A	DATA >8B6A,>AA8A,>2880	M	>4D
A0124 01D4 8A2C	DATA >8A2C,>AA9A,>2880	N	>4E
A0125 01DA FA28	DATA >FA28,>A28A,>2F80	O	>4F
A0126 01E0 F228	DATA >F228,>BC82,>0800	P	>50
A0127 01E6 7228	DATA >7228,>A2AA,>4680	Q	>51
A0128 01EC F228	DATA >F228,>BCA2,>4880	R	>52
A0129 01F2 7228	DATA >7228,>1C0A,>2700	S	>53
A0130 01F8 FB82	DATA >FB82,>0820,>8200	T	>54
A0131 01FF 8A28	DATA >8A28,>A28A,>2700	U	>55
A0132 0204 8A28	DATA >8A28,>9450,>8200	V	>56
A0133 020A 8A28	DATA >8A28,>AAAA,>A500	W	>57
A0134 0210 8A25	DATA >8A25,>0852,>2880	X	>58
A0135 0216 8A25	DATA >8A25,>0820,>8200	Y	>59
A0136 021C FB21	DATA >FB21,>0842,>0F80	Z	>5A
A0137 0222 3882	DATA >3882,>0820,>8380	[>5B
A0138 0228 0204	DATA >0204,>0810,>2000	\	>5C
A0139 022E 7041	DATA >7041,>0410,>4700]	>5D
A0140 0234 2148	DATA >2148,>8000,>0000	^	>5E
A0141 023A 0000	DATA >0000,>0000,>0F80	_	>5F
A0142 0240 4081	DATA >4081,>0000,>0000	`	>60
A0143 0246 0007	DATA >0007,>22FA,>2880	a	>61
A0144 024C 000F	DATA >000F,>1271,>2F00	b	>62
A0145 0252 0007	DATA >0007,>A082,>0780	c	>63
A0146 0258 000F	DATA >000F,>1249,>2F00	d	>64
A0147 025E 000F	DATA >000F,>20E2,>0F00	e	>65
A0148 0264 000F	DATA >000F,>20E2,>0800	f	>66
A0149 026A 0007	DATA >0007,>A0BA,>2700	g	>67
A0150 0270 0008	DATA >0008,>A2FA,>2880	h	>68
A0151 0276 0007	DATA >0007,>0820,>8700	i	>69
A0152 027C 0007	DATA >0007,>0822,>8E00	j	>6A
A0153 0282 0009	DATA >0009,>28C2,>8900	k	>6B
A0154 0288 0008	DATA >0008,>2082,>0F80	l	>6C
A0155 028E 0008	DATA >0008,>B6AA,>2880	m	>6D
A0156 0294 0008	DATA >0008,>B2AA,>6880	n	>6E
A0157 029A 000F	DATA >000F,>A28A,>2F80	o	>6F
A0158 02A0 000F	DATA >000F,>22F2,>0800	p	>70
A0159 02A6 000F	DATA >000F,>A2AA,>4E80	q	>71
A0160 02AC 000F	DATA >000F,>22F2,>8900	r	>72
A0161 02B2 0007	DATA >0007,>A070,>2F00	s	>73
A0162 02B8 000F	DATA >000F,>8820,>8200	t	>74
A0163 02BE 0004	DATA >0004,>9249,>2300	u	>75
A0164 02C4 0008	DATA >0008,>A292,>8400	v	>76
A0165 02CA 0008	DATA >0008,>A2AB,>6880	w	>77
A0166 02D0 0008	DATA >0008,>9421,>4880	x	>78
A0167 02D6 0008	DATA >0008,>9420,>8200	y	>79
A0168 02DC 000F	DATA >000F,>8421,>0F80	z	>7A
A0169 02E2 3104	DATA >3104,>2041,>0300	{	>7B
A0170 02E8 2082	DATA >2082,>0020,>8200		>7C
A0171 02EE 6041	DATA >6041,>0210,>4600	}	>7D
A0172 02F4 42A1	DATA >42A1,>0000,>0000	~	>7E
A0173 02FA 8DD8	DATA >8DD8,>F7DF,>FDFF	DEL	>7F (REV. '?')
A0174	*		
A0175	* 911 VDT TYPE GRAPHICS SET		
A0176	*		
A0177 0300 0000	DATA >0000,>0000,>001C		>80
A0178 0306 0000	DATA >0000,>0000,>071C		>81
A0179 030C 0000	DATA >0000,>0001,>C71C		>82
A0180 0312 0000	DATA >0000,>0071,>C71C		>83
A0181 0318 0000	DATA >0000,>1C71,>C71C		>84
A0182 031F 0007	DATA >0007,>1C71,>C71C		>85

A0183	0324	01C7	DATA	>01C7,>1C71,>C71C		>86
A0184	032A	71C7	DATA	>71C7,>1C71,>C71C		>87
A0185	0330	7A1B	DATA	>7A1B,>69B6,>1780	COPYRIGHT	>88
A0186	0336	A95A	DATA	>A95A,>95A9,>5A95	CHECKED BLK	>89
A0187	033C	0007	DATA	>0007,>DF7D,>C71C		>8A
A0188	0342	71CF	DATA	>71CF,>3CF1,>C71C		>8B
A0189	0348	000F	DATA	>000F,>FFFD,>C71C		>8C
A0190	034E	71C7	DATA	>71C7,>DF7C,>0000		>8D
A0191	0354	0C61	DATA	>0C61,>0821,>0C20		>8E
A0192	035A	03F0	DATA	>03F0,>0003,>F000		>8F
A0193	0360	0008	DATA	>0008,>2080,>0000		>90
A0194	0366	000C	DATA	>000C,>30C0,>0000		>91
A0195	036C	000E	DATA	>000E,>38E0,>0000		>92
A0196	0372	000F	DATA	>000F,>3CF0,>0000		>93
A0197	0378	000F	DATA	>000F,>BEF8,>0000		>94
A0198	037E	000F	DATA	>000F,>FFFC,>0000		>95
A0199	0384	3124	DATA	>3124,>1C41,>0B80	POUND	>96
A0200	038A	0007	DATA	>0007,>1C70,>0000		>97
A0201	0390	71CF	DATA	>71CF,>FFFD,>C71C		>98
A0202	0396	FFFF	DATA	>FFFF,>FFFF,>FFFF		>99
A0203	039C	000F	DATA	>000F,>3CF1,>C71C		>9A
A0204	03A2	71C7	DATA	>71C7,>DF7D,>C71C		>9B
A0205	03AB	71CF	DATA	>71CF,>FFFC,>0000		>9C
A0206	03AE	71CF	DATA	>71CF,>3CF0,>0000		>9D
A0207	03B4	8304	DATA	>8304,>0820,>4183		>9E
A0208	03BA	8A28	DATA	>8A28,>A28A,>28A2		>9F

A0209 *
 A0210 *
 A0211 *

SPECIAL GRAPHICS SET

A0212	03C0	21CA	DATA	>21CA,>8820,>8200	UP	ARROW	>A0
A0213	03C6	2082	DATA	>2082,>08A9,>C200	DOWN	ARROW	>A1
A0214	03CC	0081	DATA	>0081,>3E10,>8000	RIGHT	ARROW	>A2
A0215	03D2	0084	DATA	>0084,>3E40,>8000	LEFT	ARROW	>A3
A0216	03DB	00E1	DATA	>00E1,>8A42,>0000	N. E	ARROW	>A4
A0217	03DE	0021	DATA	>0021,>28C3,>8000	S. W	ARROW	>A5
A0218	03E4	0204	DATA	>0204,>0A18,>E000	S. E	ARROW	>A6
A0219	03EA	038C	DATA	>038C,>2810,>2000	N. W	ARROW	>A7
A0220	03F0	21C2	DATA	>21C2,>2AA9,>C000	ANCHOR		>A8

A0221 *
 A0222 *
 A0223 *

03F6' USERCS EQU \$ USER DEFINED CHARACTER SET

A0224	03F6	00	BYTE	>00,>00,>00,>00,>00,>00	?	>A9
A0225	03FC	00	BYTE	>00,>00,>00,>00,>00,>00	?	>AA
A0226	0402	00	BYTE	>00,>00,>00,>00,>00,>00	?	>AB
A0227	0408	00	BYTE	>00,>00,>00,>00,>00,>00	?	>AC
A0228	040E	00	BYTE	>00,>00,>00,>00,>00,>00	?	>AD
A0229	0414	00	BYTE	>00,>00,>00,>00,>00,>00	?	>AE
A0230	041A	00	BYTE	>00,>00,>00,>00,>00,>00	?	>AF
A0231	0420	00	BYTE	>00,>00,>00,>00,>00,>00	?	>B0
A0232	0426	00	BYTE	>00,>00,>00,>00,>00,>00	?	>B1
A0233	042C	00	BYTE	>00,>00,>00,>00,>00,>00	?	>B2
A0234	0432	00	BYTE	>00,>00,>00,>00,>00,>00	?	>B3
A0235	0438	00	BYTE	>00,>00,>00,>00,>00,>00	?	>B4
A0236	043E	00	BYTE	>00,>00,>00,>00,>00,>00	?	>B5
A0237	0444	00	BYTE	>00,>00,>00,>00,>00,>00	?	>B6
A0238	044A	00	BYTE	>00,>00,>00,>00,>00,>00	?	>B7
A0239	0450	00	BYTE	>00,>00,>00,>00,>00,>00	?	>B8
A0240	0456	00	BYTE	>00,>00,>00,>00,>00,>00	?	>B9
A0241	045C	00	BYTE	>00,>00,>00,>00,>00,>00	?	>BA
A0242	0462	00	BYTE	>00,>00,>00,>00,>00,>00	?	>BB

A0243	046B	00	BYTE	>00,>00,>00,>00,>00,>00	?	>BC
A0244	046E	00	BYTE	>00,>00,>00,>00,>00,>00	?	>BD
A0245	0474	00	BYTE	>00,>00,>00,>00,>00,>00	?	>BE
A0246	047A	00	BYTE	>00,>00,>00,>00,>00,>00	?	>BF
A0247	0480	00	BYTE	>00,>00,>00,>00,>00,>00	?	>C0
A0248	0486	00	BYTE	>00,>00,>00,>00,>00,>00	?	>C1
A0249	048C	00	BYTE	>00,>00,>00,>00,>00,>00	?	>C2
A0250	0492	00	BYTE	>00,>00,>00,>00,>00,>00	?	>C3
A0251	049B	00	BYTE	>00,>00,>00,>00,>00,>00	?	>C4
A0252	049E	00	BYTE	>00,>00,>00,>00,>00,>00	?	>C5
A0253	04A4	00	BYTE	>00,>00,>00,>00,>00,>00	?	>C6
A0254	04AA	00	BYTE	>00,>00,>00,>00,>00,>00	?	>C7
A0255	04B0	00	BYTE	>00,>00,>00,>00,>00,>00	?	>C8
A0256	04B6	00	BYTE	>00,>00,>00,>00,>00,>00	?	>C9
A0257	04BC	00	BYTE	>00,>00,>00,>00,>00,>00	?	>CA
A0258	04C2	00	BYTE	>00,>00,>00,>00,>00,>00	?	>CB
A0259	04CB	00	BYTE	>00,>00,>00,>00,>00,>00	?	>CC
A0260	04CE	00	BYTE	>00,>00,>00,>00,>00,>00	?	>CD
A0261	04D4	00	BYTE	>00,>00,>00,>00,>00,>00	?	>CE
A0262	04DA	00	BYTE	>00,>00,>00,>00,>00,>00	?	>CF
A0263	04E0	00	BYTE	>00,>00,>00,>00,>00,>00	?	>D0
A0264	04E6	00	BYTE	>00,>00,>00,>00,>00,>00	?	>D1
A0265	04EC	00	BYTE	>00,>00,>00,>00,>00,>00	?	>D2
A0266	04F2	00	BYTE	>00,>00,>00,>00,>00,>00	?	>D3
A0267	04FB	00	BYTE	>00,>00,>00,>00,>00,>00	?	>D4
A0268	04FE	00	BYTE	>00,>00,>00,>00,>00,>00	?	>D5
A0269	0504	00	BYTE	>00,>00,>00,>00,>00,>00	?	>D6
A0270	050A	00	BYTE	>00,>00,>00,>00,>00,>00	?	>D7
A0271	0510	00	BYTE	>00,>00,>00,>00,>00,>00	?	>D8
A0272	0516	00	BYTE	>00,>00,>00,>00,>00,>00	?	>D9
A0273	051C	00	BYTE	>00,>00,>00,>00,>00,>00	?	>DA
A0274	0522	00	BYTE	>00,>00,>00,>00,>00,>00	?	>DB
A0275	052B	00	BYTE	>00,>00,>00,>00,>00,>00	?	>DC
A0276	052E	00	BYTE	>00,>00,>00,>00,>00,>00	?	>DD
A0277	0534	00	BYTE	>00,>00,>00,>00,>00,>00	?	>DE
A0278	053A	00	BYTE	>00,>00,>00,>00,>00,>00	?	>DF
A0279	0540	00	BYTE	>00,>00,>00,>00,>00,>00	?	>E0
A0280	0546	00	BYTE	>00,>00,>00,>00,>00,>00	?	>E1
A0281	054C	00	BYTE	>00,>00,>00,>00,>00,>00	?	>E2
A0282	0552	00	BYTE	>00,>00,>00,>00,>00,>00	?	>E3
A0283	055B	00	BYTE	>00,>00,>00,>00,>00,>00	?	>E4
A0284	055E	00	BYTE	>00,>00,>00,>00,>00,>00	?	>E5
A0285	0564	00	BYTE	>00,>00,>00,>00,>00,>00	?	>E6
A0286	056A	00	BYTE	>00,>00,>00,>00,>00,>00	?	>E7
A0287	0570	00	BYTE	>00,>00,>00,>00,>00,>00	?	>E8
A0288	0576	00	BYTE	>00,>00,>00,>00,>00,>00	?	>E9
A0289	057C	00	BYTE	>00,>00,>00,>00,>00,>00	?	>EA
A0290	0582	00	BYTE	>00,>00,>00,>00,>00,>00	?	>EB
A0291	058B	00	BYTE	>00,>00,>00,>00,>00,>00	?	>EC
A0292	058E	00	BYTE	>00,>00,>00,>00,>00,>00	?	>ED
A0293	0594	00	BYTE	>00,>00,>00,>00,>00,>00	?	>EE
A0294	059A	00	BYTE	>00,>00,>00,>00,>00,>00	?	>EF
A0295	05A0	00	BYTE	>00,>00,>00,>00,>00,>00	?	>F0
A0296	05A6	00	BYTE	>00,>00,>00,>00,>00,>00	?	>F1
A0297	05AC	00	BYTE	>00,>00,>00,>00,>00,>00	?	>F2
A0298	05B2	00	BYTE	>00,>00,>00,>00,>00,>00	?	>F3
A0299	05B8	00	BYTE	>00,>00,>00,>00,>00,>00	?	>F4
A0300	05BE	00	BYTE	>00,>00,>00,>00,>00,>00	?	>F5
A0301	05C4	00	BYTE	>00,>00,>00,>00,>00,>00	?	>F6
A0302	05CA	00	BYTE	>00,>00,>00,>00,>00,>00	?	>F7

A0303	05D0	00	BYTE	>00,>00,>00,>00,>00,>00	?	>F8
A0304	05D6	00	BYTE	>00,>00,>00,>00,>00,>00	?	>F9
A0305	05DC	00	BYTE	>00,>00,>00,>00,>00,>00	?	>FA
A0306	05E2	00	BYTE	>00,>00,>00,>00,>00,>00	?	>FB
A0307	05E8	00	BYTE	>00,>00,>00,>00,>00,>00	?	>FC
A0308	05EE	00	BYTE	>00,>00,>00,>00,>00,>00	?	>FD
A0309	05F4	00	BYTE	>00,>00,>00,>00,>00,>00	?	>FE
A0310	05FA	00	BYTE	>00,>00,>00,>00,>00,>00	?	>FF
A0311	0600'	PCHTE	EQU	\$		
0004	0600	94C2	ZZZZZ\$	DATA	>94C2	END OF CODE MARKER
0005	0602'	BRAM	EQU	\$		
0006			END			
NO ERRORS,		NO WARNINGS				