SYM• · PG•LN• IDENT•

137BRS	1	8	M	1LBL	9	33	M	10RG	1	3	TS84A				
A1	10	4	TS84A	A2	10	5	TS84A	A3	10	6	TS84A	ACA	6	28	S
ACDATA	2	20	TYM	ACST	6	21	S	ACTIME	6	39	S	ACTLST	13	39	PAC
ACTR	1	7	I	ADMSK	4	25	S	ADRPMT	5	19	S	ADRSMT	5	17	S
AExSMT	5	20	S	AFSV4	10	32	TS84A	AFSV5	4	39	TS84A	AFSV6	10	34	TS84A
AFSV7	10	35	TS84A	AIRWD	6	38	PAC	ALARM	2	26	M	ALØG	1	7	M
ALTERM	29	7	PAC	ANN	29	18	PAC	ANN	22	9	PAC	ANØ	4	17	TS84A
APMTE	12	21	B	ARD	11	19	D	ARD1	11	20	D	ARD2	11	21	D
ARDD	11	12	D	ARG1	25	31	RUP	ARG2	25	32	RUP	ARM205	11	12	R
ARMI	9	20	M	ARMØT	8	9	S	ARWDD	11	15	D	AST	6	27	I
AST2	6	30	I	AUNN	6	19	S	AWD	11	18	D	AWDD	11	14	D
B	5	33	R	BØ	8	7	D	BØA	8	8	D	BØC	8	11	D
B129A	15	25	TYM	B135A	28	22	PAC	B135B	28	29	PAC	B135C	28	34	PAC
B137A	15	36	TYM	B137B	16	1	TYM	B138A	16	14	TYM	B157A	3	28	D
B2	8	21	D	B2A	9	2	D	B2B	9	10	D	B2C	9	11	D
B2R	8	38	D	B2W	8	34	D	B2W1	8	37	D	B300	6	13	NET
B300A	16	2	I	B40A	14	13	TYM	B40B	14	14	TYM	B66A	16	26	D
B81A	29	1	PAC	B81B	28	38	PAC	B9	9	21	D	B99	9	29	D
B9A	9	26	D	B=BRS	2	36	PMT	BASE	3	22	TS84A	BE	2	13	M
BFC	1	18	D	BFCN	1	19	D	BFP	1	17	D	BGET	21	8	D
BGET1	21	11	D	BGRAB	8	17	D	BGRABA	8	19	D	BINC	10	28	D
BIØ	1	34	I	BIØX	1	35	I	BIØX1	7	29	D	BITSUM	12	5	D
BJØB	19	27	I	BLKBSY	4	25	W	BMTST	23	29	B	BPCT	10	37	D
BPØPX	4	30	PAC	BPT	2	4	M	BPT2	5	17	I	BPTEST	5	14	I
BPUT	21	15	D	BRKTBL	4	30	TS84A	BRMC	6	14	I	BRMC1	6	17	I
BRMW1	13	3	I	BRMW2	13	4	I	BRMW3	13	5	I	BRS	2	5	B
BRS1	1	23	D	BRS11	12	34	TYM	BRS112	14	28	TYM	BRS113	15	15	TYM
BRS114	15	18	TYM	BRS12	12	37	TYM	BRS129	15	23	TYM	BRS12B	13	14	TYM
BRS13	13	16	TYM	BRS132	15	27	TYM	BRS135	28	20	PAC	BRS137	15	32	TYM
BRS138	16	5	TYM	BRS139	16	26	TYM	BRS14	13	20	TYM	BRS147	3	18	D
BRS151	3	32	D	BRS152	16	30	TYM	BRS153	16	33	TYM	BRS154	16	39	TYM
BRS155	17	6	TYM	BRS156	17	11	TYM	BRS157	3	23	D	BRS159	17	14	TYM
BRS160	17	19	TYM	BRS161	17	22	TYM	BRS162	17	23	TYM	BRS163	17	25	TYM
BRS164	17	27	TYM	BRS166	17	30	TYM	BRS167	19	7	I	BRS168	17	33	TYM
BRS169	17	34	TYM	BRS17	3	16	D	BRS172	17	36	TYM	BRS174	4	12	D
BRS175	11	13	B	BRS177	18	5	TYM	BRS178	18	8	TYM	BRS20	3	13	D

BRS24	13	23	TYM	BRS29	13	29	TYM	BRS40	14	6	TYM	BRS45	28	36	PAC
BRS66	16	20	D	BRS7	25	7	B	BRS77	3	19	D	BRS8	4	1	D
BRS81	28	37	PAC	BRS85	14	18	TYM	BRS86	14	21	TYM	BRS86A	14	23	TYM
BRS87	14	25	TYM	BRS92	17	3	D	BRS94	29	18	B	BRSRET	5	12	B
BRSTV	8	33	S	BRSTVA	8	34	S	BS	2	6	B	BS1	2	8	B
BS10	4	11	B	BS112A	14	37	TYM	BS112B	15	2	TYM	BS112C	15	6	TYM
BS112D	15	12	TYM	BS112E	15	13	TYM	BS12	4	12	B	BS14	4	10	B
BS153A	16	37	TYM	BS158	5	23	I	BS16	4	13	B	BS2	4	7	B
BS20B	4	14	B	BS22	4	17	B	BS3	4	3	B	BS4	4	8	B
BS6	4	9	B	BSET	21	22	D	BSEZ	10	32	D	BSFT	2	9	B
BST	2	28	B	BSTU	4	2	B	BSX	1	39	B	BT1	11	2	D
BT2	11	3	D	BT3	11	4	D	BT4	11	5	D	BTC	10	10	D
BTC1	10	17	D	BTCA	10	24	D	BTCB	10	25	D	BTCX	10	26	D
BUF	1	26	I	BUF3	1	27	I	BUFF	8	12	M	BUFN	1	30	I
BUG	1	5	B	BUG1	1	21	R	BUMP	9	3	R	BVC0PY	6	10	I
BVIATS	6	6	I	BVLDA	6	11	I	BVSTA	6	12	I	BWI0	9	36	D
BWI01	10	6	D	BZ1	10	35	D	BZSIZE	4	12	NET	C1	22	3	D
C2	4	24	D	C256	4	16	D	C=CHAI	2	37	PMT	CAC0	14	3	PAC
CAC1	14	4	PAC	CAC10	14	10	PAC	CAC11	14	11	PAC	CAC2	14	5	PAC
CAC3	14	6	PAC	CAC4	14	7	PAC	CAC6	14	8	PAC	CAC9	14	9	PAC
CACK	14	2	PAC	CACLST	13	32	PAC	CADD	11	1	R	CADDC	11	38	R
CAE	22	17	B	CAE1	22	36	B	CAE10	22	34	B	CAE11	22	18	B
CAE2	22	37	B	CAE3	22	38	B	CAE4	22	22	B	CAE5	22	20	B
CAE6	22	29	B	CAE7	22	31	B	CAE8	22	26	B	CAE9	22	27	B
CALLD	4	6	I	CBF0	7	8	M	CBF1	7	9	M	CBF2	7	10	M
CBF3	7	11	M	CBF4	7	12	M	CBIP	8	14	M	CB0F	7	16	M
CC	7	33	M	CCKWRD	7	19	M	CCLR	22	11	D	CCLR1	22	14	D
CCLRX	22	16	D	CCLS	3	34	D	CCT	19	29	I	CDEF4	7	25	TS84A
CDEF5	7	26	TS84A	CDEL1	24	16	D	CDEL3	24	22	D	CDEL3A	24	28	D
CDEL3B	24	29	D	CDMS	18	27	D	CDMS1	4	26	I	CDMS10	4	28	I
CDMS2	4	30	I	CDSA	8	25	M	CDSB	8	26	M	CDSX	8	27	M
CDTC	22	25	D	CDTC1	22	34	D	CDTC9	23	3	D	CDTCA	22	39	D
CDTCB	23	1	D	CDTCX	23	2	D	CE	25	6	D	CE1	25	21	D
CE9	25	15	D	CE0F	7	15	M	CE0F1	7	32	M	CE0FP	21	25	D
CE0FP1	21	29	D	CE0FP2	21	27	D	CE0FP3	21	28	D	CERR	19	11	D
CERR1	19	10	D	CERR2	19	9	D	CERR3	19	8	D	CERR4	19	7	D
CERR5	19	6	D	CERRC	19	12	D	CESW	25	17	D	CESW1	25	19	D
CEX	23	5	B	CEX1	23	10	B	CEX2	23	8	B	CEX3	23	11	B
CEX4	23	6	B	CFI1	9	14	TS84A	CFI2	9	15	TS84A	CFLG	7	29	M

CFLGEX	4	11	I	CG	24	39	D	CGRB1	24	31	D	CGRB1A	24	36	D
CGRB1B	24	37	D	CHFD	9	11	TS84A	CHR0	3	16	I	CHR1	3	17	I
CHR2	3	18	I	CHR3	25	24	PAC	CHREL	24	33	PAC	CHRW	2	39	I
CHRW0	3	7	I	CHRW1	3	8	I	CHRW2	3	9	I	CHRW3	3	11	I
CHRW4	3	13	I	CHW0	3	21	I	CHW1	3	23	I	CHW2	3	25	I
CI2	2	20	I	CI3	2	32	I	CI4	2	30	I	CI5	2	29	I
CIE	2	35	I	CIE3	2	36	I	CIN	3	29	TS84A	CI0	2	34	I
CI0T1	4	37	TYM	CI0T1A	4	36	TYM	CIT	2	14	I	CITS	2	15	I
CJP	12	39	PAC	CKQI	26	13	PAC	CK205	2	10	SET	CKCPU	14	26	M
CKCPU1	14	27	M	CKERR	18	21	D	CKF	1	26	M	CKMEM	2	34	SET
CKN	1	25	M	CKRD	2	20	SET	CKRDBF	2	32	SET	CKS8	18	6	I
CKS9	18	5	I	CKSUM	17	36	I	CKXMA	2	14	SET	CKXMA1	2	17	SET
CKXMA2	2	18	SET	CL1	4	4	D	CL2	4	8	D	CLFK	10	17	B
CLG0	6	30	R	CLIB	19	25	TYM	CLINT	25	35	PAC	CLINT1	25	36	PAC
CLINT2	26	1	PAC	CLINT4	25	37	PAC	CLINTC	25	39	PAC	CLINTT	26	21	PAC
CLINTU	26	26	PAC	CLINTV	26	27	PAC	CLINTW	26	28	PAC	CLLA	11	14	R
CLLB	11	15	R	CLLP	6	28	R	CLLX	11	16	R	CLM2	13	38	TYM
CLM2A	14	2	TYM	CLM2B	14	3	TYM	CLM2C	14	4	TYM	CL0CK2	7	18	SET
CL0CK3	25	33	PAC	CL0UT	26	3	PAC	CLP1	23	38	D	CLP2	24	6	D
CLP3	24	4	D	CLP4	24	13	D	CLP5	24	14	D	CLP6	24	12	D
CLP7	24	11	D	CLP8	24	10	D	CLP9	24	9	D	CLRPRT	23	35	D
CLRPS	17	7	B	CLS	20	28	D	CLS5	20	34	D	CLSALL	4	2	D
CLTS	11	17	R	CLTSA	11	18	R	CLWRL	8	15	I	CMND	5	11	TS84A
CMPG0	11	10	PAC	CMPG02	11	14	PAC	CMPG03	11	13	PAC	CMPIN	11	2	PAC
CMPIN3	11	4	PAC	CMPQT	13	9	PAC	CMPST	13	4	PAC	CMRL1	3	26	TS84A
CMRL2	3	27	TS84A	CMRRL1	12	36	PAC	CMRRL2	12	37	PAC	CMRRL3	12	38	PAC
CMSK	1	13	D	CMSK1	1	16	D	CMSTPS	17	13	B	CN3	2	6	D
CN4	2	8	D	CN5	2	15	D	CN6	3	4	D	CN6A	3	5	D
CN7	2	17	D	CN8	2	23	D	CN8A	2	36	D	CN8B	3	1	D
CNERR	3	10	D	C0K	2	35	D	C0PY1	9	7	TS84A	C0PY2	9	8	TS84A
C0PY7	9	9	TS84A	C0PYT	9	10	TS84A	C0UNT	26	33	PAC	C0UT	3	30	TS84A
C0V	7	34	M	CP	7	21	M	CP1	8	15	M	CP2	8	16	M
CPARW	6	16	S	CP0RT	18	30	TYM	CP0RT2	18	34	TYM	CP0RT9	18	37	TYM
CP0S	20	16	D	CPRIV	19	14	D	CPS1	20	19	D	CPS2	20	21	D
CPS4	20	24	D	CPSTMS	17	22	B	CPSTPS	17	33	B	CPT0P	7	17	M
CPUI0P	23	19	B	CPUP	23	16	B	CPX	18	34	D	CPX1	18	37	D
CPX2	18	38	D	CPY	19	1	D	CPY1	19	4	D	CQ	7	18	M
CR	20	9	D	CR1	20	14	D	CR3	4	23	TS84A	CRA1	33	17	PAC
CRA2	33	22	PAC	CRA5	33	27	PAC	CRACT	9	10	S	CRASH	33	26	PAC

CRET0	8	17	M	CRET1	8	18	M	CRET2	8	19	M	CRET3	8	20	M
CRET4	8	21	M	CRL6	5	25	W	CRSW	15	30	TYM	CRTA	13	3	B
CRTA1	13	6	B	CRTA2	13	5	B	CRW	19	20	D	CRW9	19	27	D
CSH	1	14	D	CSH1	1	15	D	CSIZE	7	14	M	CSW	7	30	M
CSW1	7	31	M	CSYS2	7	24	TS84A	CT1	8	22	M	CT2	8	23	M
CT3	8	24	M	CTL10	5	22	W	CTLI	6	6	W	CTLI2	6	7	W
CTLX	5	38	W	CTLX2	5	39	W	CTLX3	6	4	W	CTMCA	8	31	I
CTRL	5	15	W	CTRU	11	2	B	CTRU	10	35	B	CTRU1	11	6	B
CTRU1	10	38	B	CTRU2	10	39	B	CTTAB	9	17	W	CW	19	29	D
CW1	19	35	D	CW2	19	37	D	CW3	20	1	D	CW8	20	3	D
CW9	20	2	D	CWALL	20	5	D	CX1	22	18	D	CZ0NE	9	7	S
D(1)•Δ	9	29	M	D(1)•Δ	9	28	M	D(1)•Δ	3	24	M	D(1)•Δ	3	23	M
D(PSPJ	1	19	S	D1	9	15	M	D2M1	23	39	B	D=L0WE	2	38	PMT
DBF	5	22	TS84A	DBITS	7	26	S	DBT0P	8	7	M	DCT	13	12	I
DDIFP	1	15	M	DEFINE	9	14	M	DEFL	1	35	M	DEL1	7	16	TS84A
DEL2	7	17	TS84A	DEL3	7	18	TS84A	DET	13	11	I	DEVCH	8	35	I
DEVICE	8	34	I	DEXP	9	26	TS84A	DF01	27	23	B	DF08	27	12	B
DF1	7	13	TS84A	DF2	7	14	TS84A	DFK	27	4	B	DFLG	10	29	TS84A
DFR	6	3	B	DGET	8	28	M	DGTCNT	9	21	TS84A	DIR	1	36	M
DIS205	11	13	R	DISC	1	17	M	DISCN	5	33	S	DISCRW	11	22	D
DL0CK	19	22	I	DL0CKD	18	17	I	DL0CKI	18	8	I	DL0CKR	18	25	I
DLR1	18	27	I	DMIN	9	3	S	DM0D	12	24	I	DMS	4	9	PAC
DMS111	1	33	PAC	DMSK	13	23	I	DNTERM	25	28	PAC	D0P1	24	4	B
D0P2	24	3	B	D0P3	24	2	B	D0WN	6	29	S	DP2IBM	15	39	D
DPC	12	25	I	DPMTE	12	27	B	DPMTE1	12	31	B	DPMTE2	12	30	B
DR	1	5	SET	DRAG0	9	17	R	DRC	5	26	S	DREAL	10	19	I
DRMSET	1	13	SET	DRMTRY	12	39	I	DRQ	10	27	I	DRQU	12	3	I
DRSW9	7	37	TS84A	DRT	13	10	I	DRT1	15	4	I	DRT2	15	8	I
DSCTBL	5	21	SET	DSCT0P	19	25	I	DSET1	1	24	SET	DSKMET	9	1	M
DSTRCT	10	30	TS84A	DSTSW	9	11	S	DSW	13	1	I	DTAG	24	1	B
DTC	15	18	D	DTC	13	20	D	DTC1	15	27	D	DTC2	13	25	D
DTC3	13	26	D	DTC8	14	6	D	DTC9	14	3	D	DTCA	14	8	D
DTCB	14	9	D	DTCBSV	15	37	D	DTCSW	14	7	D	DTCX	14	10	D
DTH	5	33	I	DTS	16	15	D	DTS	14	12	D	DTS1	14	16	D
DTXS1	13	6	I	DTXS2	13	7	I	DUMP1	7	32	TS84A	EACST	6	22	S
EAUNN	6	20	S	EBASE	5	14	TS84A	ECA	1	9	S	ECB	1	15	S
ECC	1	21	S	ECDFIL	5	36	TS84A	ECDMP6	7	33	TS84A	ECRFIL	5	34	TS84A
ECRV6	7	34	TS84A	EDBF	5	23	TS84A	EDCL	12	27	I	EEE	9	6	TS84A
EFGL	7	5	S	EF0RKD	6	25	TS84A	EGFLG	1	24	I	EIDER	19	17	I

FFLT	39	13	RUP	FFLT	29	39	RUP	FFLT9	39	20	RUP	FFLTE	25	15	RUP
FFLTE	25	10	RUP	FFMPE	20	39	RUP	FFMPE1	21	7	RUP	FFMPE1	21	2	RUP
FFMPEF	21	10	RUP	FFNAE	24	29	RUP	FFNAE	24	25	RUP	FFSBE	22	19	RUP
FFSBE1	22	26	RUP	FFSBE1	22	21	RUP	FFSBEF	22	29	RUP	FFSIE	22	37	RUP
FFSIE1	23	5	RUP	FFSIE1	22	39	RUP	FFSIEF	23	13	RUP	FFX	9	39	TS84A
FG940A	5	35	S	FG940B	5	36	S	FGLIST	7	4	S	FILE	8	10	M
FILINT	4	15	I	FI0W	10	11	TS84A	FIVE	2	11	S	FIXBIG	39	7	RUP
FIXBIG	29	33	RUP	FIXBIG	29	10	RUP	FIXCH	2	32	R	FIXEND	39	2	RUP
FK01	33	3	B	FK04	27	1	B	FK07A	31	29	B	FK08	32	3	B
FK08B	32	5	B	FK09	31	34	B	FK09A	32	14	B	FK20	32	8	B
FK21	32	10	B	FK22	32	12	B	FK23	31	18	B	FK24	31	16	B
FK25	31	17	B	FKG0	32	17	B	FKG02	32	24	B	FKG03	32	21	B
FKST	31	9	B	FKSTAT	26	7	B	FKSTW	32	7	B	FKWT	33	1	B
FKWT1	33	2	B	FLAC	12	8	RUP	FLAD	13	5	RUP	FLAF	12	20	RUP
FLAG	3	18	M	FLAG	17	39	RUP	FLAGM	12	37	RUP	FLANZ	12	27	RUP
FLA0A	12	36	RUP	FLA0F	12	29	RUP	FLA0K	12	30	RUP	FLA0S	13	19	RUP
FLA0T	12	21	RUP	FLAX	12	31	RUP	FLAX1	12	33	RUP	FLCA	14	21	RUP
FLC9M	14	20	RUP	FLEND	34	22	RUP	FLEND	16	26	RUP	FLINT1	4	24	I
FLINT2	4	21	I	FLMX	14	39	RUP	FLNA	36	25	RUP	FLNA	15	2	RUP
FLNB	14	38	RUP	FLNB1	14	36	RUP	FLND2	34	28	RUP	FLND2	16	34	RUP
FL0F	13	26	RUP	FL0J	13	31	RUP	FL0J1	13	36	RUP	FLP0S	5	8	TS84A
FLSS	13	14	RUP	FLX	5	9	TS84A	FMIN1	24	27	RUP	FMK	8	39	I
FMK2	9	1	I	FMP	15	28	RUP	FMPE	33	27	RUP	FMPE	15	29	RUP
FN1	7	6	TS84A	FN2	7	7	TS84A	FNA	35	32	RUP	FNA	14	4	RUP
FNA1	36	31	RUP	FNA0F2	14	27	RUP	FNA0FL	14	26	RUP	FNA0V	36	5	RUP
FNA0VR	36	12	RUP	FNAS	14	7	RUP	FNASX	18	1	RUP	FNAUDR	36	23	RUP
FNPTR	7	8	TS84A	FNPTR1	7	9	TS84A	F0RGET	14	34	M	F0UR	2	10	S
F0VER	33	6	RUP	FPH	1	16	M	FPLST	14	28	PAC	FP0V01	28	39	RUP
FP0VI	28	31	RUP	FPULST	19	7	PAC	FQJ	9	13	PAC	FQJ2	9	14	PAC
FQJ3	9	17	PAC	FQJ4	9	32	PAC	FQJ5	9	33	PAC	FRLP	4	28	SET
FRP2	19	6	B	FRPMT	19	4	B	FSAVE	4	38	I	FSAVE1	5	6	I
FSAVE2	5	7	I	FSB	13	38	RUP	FSB1	32	27	RUP	FSB2	32	7	RUP
FSB3	32	6	RUP	FSBE	32	1	RUP	FSBE	13	39	RUP	FSIZE	8	29	M
FULST	6	12	S	FUNDER	33	22	RUP	G=F0RK	3	2	PMT	GA	11	38	D
GA1P	12	2	D	GAN	5	13	TS84A	GAP	12	1	D	GATT	18	26	TYM
GB1	14	29	D	GB1	12	16	D	GB2	14	30	D	GB2	12	20	D
GB3	14	33	D	GB3	12	17	D	GB6	14	25	D	GB6	12	12	D
GB6A	14	39	D	GB6A	12	33	D	GB8	15	1	D	GB8	12	30	D
GB9	15	2	D	GB9	12	31	D	GBI02	8	4	TS84A	GBI04	8	7	TS84A

GBI0C2	8	6	TS84A	GBI0CR	8	5	TS84A	GBI0FN	8	2	TS84A	GBI0PT	8	1	TS84A
GBI0SW	8	3	TS84A	GBRAN	12	3	D	GBUSY	7	3	S	GCD	5	9	RUP
GCD1	5	11	RUP	GCDP	5	10	RUP	GCI	5	25	RUP	GCI1	10	20	RUP
GCI2	6	14	RUP	GCI3	5	37	RUP	GCI4	10	21	RUP	GCI7	10	22	RUP
GCI77	5	27	RUP	GCI8	6	1	RUP	GCI9	5	31	RUP	GCIP	5	26	RUP
GCU	6	19	RUP	GDC	11	27	I	GDC1	11	29	I	GET1	8	8	TS84A
GET2	8	10	TS84A	GET3	8	12	TS84A	GET4	8	14	TS84A	GET5	8	17	TS84A
GET6	8	20	TS84A	GETBLK	14	22	D	GETBLK	12	9	D	GETFN	8	22	TS84A
GETNM	37	34	RUP	GETNM1	38	25	RUP	GETN01	1	26	TS84A	GETN02	1	27	TS84A
GETN03	1	28	TS84A	GETN04	38	9	RUP	GETN05	38	16	RUP	GETN06	38	3	RUP
GETN07	1	29	TS84A	GETNUM	37	29	RUP	GETPL	7	15	TS84A	GETSTL	6	23	TS84A
GETSTR	1	29	RUP	GFILE	8	11	M	GFK	26	24	B	GFLG	1	12	I
GFN1	7	1	TS84A	GIVEB	16	36	D	G0FN1	7	2	TS84A	G0FN2	7	3	TS84A
G0FN3	7	4	TS84A	G0FNFT	7	5	TS84A	G0REMA	10	37	TS84A	GRABB1	16	28	D
GRABB2	16	33	D	GRBM2	7	12	I	GRBMW	7	10	I	GRCT	5	10	TS84A
GRP	5	12	TS84A	GRU	4	21	TS84A	GSIN1	1	19	TS84A	GSIN2	1	20	TS84A
GSIN3	1	21	TS84A	GSIN4	1	39	RUP	GW	11	37	D	H3	4	1	B
H4	4	4	B	H=TIME	3	3	PMT	HEAD	13	20	M	HFAD1	27	1	RUP
HFADE	28	11	RUP	HFADE	26	33	RUP	HFDV	14	5	M	HFDVE	28	20	RUP
HFDVE	27	39	RUP	HFDVI	14	11	M	HFEX	27	5	RUP	HFK	26	20	B
HFK1	26	21	B	HFLD	13	8	M	HFMP	13	38	M	HFMPE	28	17	RUP
HFMPE	27	22	RUP	HFNA	28	24	RUP	HF0VM1	27	6	RUP	HFSB	13	26	M
HFSBE	28	14	RUP	HFSBE	27	8	RUP	HFSBI	13	32	M	HFST	13	14	M
HLDA	9	4	B	HLDB	9	5	B	HLDC	9	6	B	HRLRC	7	26	R
IABIT	23	1	B	ID01	12	31	I	ID02	12	32	I	ID03	12	33	I
ID05	12	34	I	ID06	12	35	I	ID1	16	12	I	ID1	14	13	I
ID124	18	29	I	ID124A	19	4	I	ID1A	14	19	I	ID2	16	15	I
ID3	14	28	I	ID4	13	23	I	ID5	13	36	I	ID6	13	37	I
ID6A	14	4	I	ID7	15	14	I	ID7A	15	19	I	IDA	16	4	I
IDB	16	5	I	IDBPR	17	18	I	IDBPW	17	19	I	IDBR	17	16	I
IDBUFF	19	20	I	IDBW	17	17	I	IDCADD	12	38	I	IDCL	12	28	I
IDCL1	12	29	I	IDCL2	12	30	I	IDDR	16	32	I	IDDR	14	30	I
IDDR1	16	39	I	IDDR1	14	38	I	IDDW	17	1	I	IDDW	14	39	I
IDE1	13	18	I	IDE2	13	16	I	IDER	19	12	I	IDER1	19	15	I
IDER2	19	16	I	IDFAIL	17	27	I	IDFILE	19	19	I	IDFRET	17	4	I
IDFRET	15	21	I	IDJBB	19	18	I	IDLBL	17	31	I	IDLBL1	17	34	I
IDM	16	9	I	IDM	13	9	I	IDM2	15	25	I	IDM22	15	32	I
IDMC1	16	7	I	IDMCK	12	23	I	IDMG0	12	16	I	IDMW	12	9	I
IDMWA	12	12	I	IDMWB	12	13	I	IDMWX	12	14	I	IDSW1	13	13	I

IDSW1A	12	36	I	IDSW1B	12	37	I	IDT	17	8	I	IDTRET	16	26	I
IDTRET	13	35	I	IDTX	17	13	I	IDX	16	6	I	IFAD	10	35	M
IFDV	12	1	M	IFDVI	12	12	M	IFLD	10	24	M	IFMP	11	29	M
IFSB	11	7	M	IFSBI	11	18	M	IFST	12	23	M	IFSTUP	26	15	RUP
IIH	1	14	M	IIR	32	22	PAC	IIR1	32	33	PAC	IIR2	32	34	PAC
IIR3	32	28	PAC	IIR4	32	35	PAC	IIR5	32	30	PAC	IMSHI	10	13	M
IMSH0	10	2	M	INCBRS	23	25	B	INCRQ	16	29	I	IND	26	17	PAC
INFIL	5	33	TS84A	INT31	10	5	I	I0D	10	14	TS84A	I0DEXP	9	27	TS84A
I0FILE	8	23	TS84A	I0FMT	9	25	TS84A	I0I	18	8	D	I0I1	18	16	D
I0I2	18	18	D	I0LINK	10	7	TS84A	I0P	23	21	B	I0PTR	7	10	TS84A
I0PTR1	7	11	TS84A	I0RD	7	25	I	I0RDW	2	11	M	I0SDW	2	10	M
I0SP1	8	9	TS84A	I0SP2	8	11	TS84A	I0SP3	8	13	TS84A	I0SP4	8	15	TS84A
I0SP5	8	18	TS84A	I0W	9	24	TS84A	ISC	18	5	RUP	ISC	15	10	RUP
ISC1	18	6	RUP	ISC1	15	11	RUP	ISC2	18	4	RUP	ISC2	15	9	RUP
J0B	5	3	S	K100	2	34	S	K1000	3	2	S	K1000D	3	3	S
K13	2	17	S	K137	2	35	S	K14	2	18	S	K15	2	19	S
K16	2	20	S	K17	2	21	S	K177	2	36	S	K1777	3	4	S
K1B4	3	9	S	K1B5	3	15	S	K1B6	3	21	S	K1B7	3	27	S
K1S4	3	10	S	K1S5	3	16	S	K1S6	3	22	S	K1S7	3	28	S
K20	2	22	S	K200	2	37	S	K2000	3	5	S	K21	2	23	S
K22	2	24	S	K23	2	25	S	K24	2	26	S	K25	2	27	S
K26	2	28	S	K27	2	29	S	K2B4	3	11	S	K2B5	3	17	S
K2B6	3	23	S	K2B7	3	29	S	K30	2	30	S	K37	2	31	S
K377	2	38	S	K3777	3	6	S	K3B7	3	32	S	K3S4	3	12	S
K3S5	3	18	S	K3S6	3	24	S	K3S7	3	30	S	K40	2	32	S
K400	2	39	S	K4000	3	7	S	K4B4	3	13	S	K4B5	3	19	S
K4B6	3	25	S	K4B7	3	31	S	K5B7	3	33	S	K6B7	3	34	S
K6S7	3	35	S	K77	2	33	S	K777	3	1	S	K7777	3	8	S
K777B5	3	38	S	K77B6	3	37	S	K7B7	3	36	S	K=UPPE	3	5	PMT
KM1	4	3	S	KM10	4	22	S	KM100	4	14	S	KM10D	4	12	S
KM2	4	4	S	KM200	4	15	S	KM3	4	5	S	KM300	4	16	S
KM4	4	6	S	KM40	4	13	S	KM400	4	17	S	KM4B3	4	18	S
KM4B4	4	19	S	KM4B5	4	20	S	KM4B6	4	21	S	KM5	4	7	S
KM6	4	8	S	KM7	4	9	S	KM8	4	10	S	KM9	4	11	S
KP06	11	9	B	KP24	11	8	B	KP96	11	7	B	KS4B4	3	39	S
KS5	3	14	S	KS5B3	4	1	S	KS6	3	20	S	KS6B2	4	2	S
KS7	3	26	S	L=L0CA	3	6	PMT	LABEL	24	29	PAC	LB	15	12	D
LB	13	11	D	LB8	13	10	D	LB9	13	9	D	LBL	9	32	M
LCDFIB	8	16	S	LDFACT	6	38	S	LDFLE	19	21	RUP	LDFLE	19	18	RUP

LDFME	18	33	RUP	LDFME1	19	1	RUP	LDFME1	18	35	RUP	LDFMEF	19	9	RUP
LDP	5	2	RUP	LDPP	5	3	RUP	LDTEMP	4	27	TS84A	LESMT	6	19	M
LETP	5	39	TS84A	LFADR	5	25	TS84A	LFDC1	7	12	TS84A	LGM0N	9	6	S
LGRS	6	1	TS84A	LISTAB	1	19	M	LISTM	1	32	M	LMCPU	11	23	TS84A
L0C	8	6	M	L0CBIT	15	4	D	L0CBIT	12	35	D	L0CBLK	16	39	D
L0C0RG	1	4	TS84A	L0G	10	5	R	L0G2	10	6	R	L0G9	10	12	R
L0GA	10	2	R	L0GB	10	3	R	L0GG	9	19	M	L0GG	1	36	R
L0GP	10	1	R	L0GX	10	4	R	L0GXX	9	39	R	LP	25	16	B
LPT	26	16	PAC	LQ	8	30	M	LREAL	19	28	PAC	LRR1	1	27	M
LRR2	1	28	M	LRR3	1	29	M	LUDL	4	19	TS84A	LVL	8	13	M
M128	11	4	I	M48K	5	22	S	M=ADD	3	7	PMT	M=PR0G	2	39	PMT
MACH	5	32	S	MAPBIT	5	25	S	MAPLBL	12	4	D	MAPNUM	7	25	S
MAXFIL	7	3	M	MAXNU	5	23	S	MAYDMS	4	35	I	MBR0	10	3	B
MBR03	10	9	B	MBR05	10	10	B	MBR06	10	12	B	MBR07	10	8	B
MBUFX	6	31	M	MBUFX	6	29	M	MCR0	33	31	PAC	MCRA	33	28	PAC
MCRB	33	29	PAC	MCRC	33	33	PAC	MCRI	33	32	PAC	MCRX	33	30	PAC
MDEL	7	38	B	MDELC	8	28	B	MFD2	9	12	TS84A	MFD21	9	13	TS84A
MGET	6	37	B	MGET1	6	28	B	MGET10	7	9	B	MGET11	7	1	B
MGET12	7	17	B	MGET13	7	18	B	MGET20	7	16	B	MGET4	6	29	B
MGTS2	7	28	B	MINUS	9	4	TS84A	MISC	7	23	S	MLABEL	2	1	M
MLVL	7	5	M	M0NCR	33	8	PAC	M0NTAB	25	4	B	MP1	2	37	SET
MP2	2	38	SET	MP2A	2	39	SET	MP7	3	8	SET	MP8	3	7	SET
MP9	3	6	SET	MPDSC	25	14	PAC	MPGC	6	20	SET	MP0PX	22	1	D
MPPACT	25	10	PAC	MPSDEX	9	5	M	MPT	12	36	B	MPTYM	25	17	PAC
MPUT2	7	35	B	MPUTA	7	31	B	MPWB	25	20	PAC	MS01	9	21	B
MS03	9	22	B	MSZ2	4	26	TS84A	MTIME	5	28	TS84A	MTPAN	22	4	B
MTRAP	22	3	B	MUX15	6	27	PAC	MX00	9	23	B	MX01	9	12	B
MX03	9	19	B	MX04	9	18	B	MX05	9	24	B	MX06	9	25	B
MX07	9	26	B	MX08	9	29	B	MX09	9	30	B	MX14	9	27	B
MXSA	11	20	B	N:	12	25	M	N:	12	14	M	N:	12	3	M
N:	11	31	M	N:	11	20	M	N:	11	9	M	N:	10	37	M
N:	10	26	M	N:	10	15	M	N:	10	4	M	N=PANI	3	8	PMT
NADMSK	4	26	S	NAST	4	33	TS84A	NB112	8	13	S	NBUF	6	28	M
NCMEM	6	21	M	NCMEM	6	15	M	NC0NV	9	29	TS84A	NDBS	7	38	M
NDBW	7	37	M	NDCL	12	26	I	NDMSK	19	24	I	NDRQ	6	32	M
NEXEC	5	8	S	NFILE	6	27	M	NF0RK	8	39	M	NGRP	4	22	TS84A
NINE	2	15	S	NINT2	9	6	I	NJ0B	6	8	M	NJ0B1	6	7	M
NLQU	6	3	M	NLVL	7	4	M	NMCHX	5	1	TS84A	NMEM	6	10	M
NMISC	8	24	S	NMSMT	1	28	PMT	N0ACT	7	1	S	N0C0MP	10	29	PAC

N0G0	10	1	PAC	N0P24	29	13	RUP	NPAC	6	4	M	NPC	31	30	PAC
NPCA	32	6	PAC	NPCB	32	7	PAC	NPGMSK	4	28	S	NP0P	2	14	M
NP0PDS	4	10	PAC	NP0Px	4	26	PAC	NP0PXQ	4	24	PAC	NP0PXB	4	23	PAC
NP0RT	2	38	M	NPPAR	6	5	M	NPUG0	16	15	PAC	NPUNXT	16	16	PAC
NPUQ	6	25	M	NR0UT	5	33	B	NSMEM	6	11	M	NSMT	6	22	M
NSMT	6	16	M	NSQU	6	2	M	NSRT	16	6	B	NSSMT	6	12	M
NSYS	5	11	S	NT19	5	37	S	NTAPE	2	21	M	NTBI	2	32	M
NTB0	2	33	M	NTERM	29	4	PAC	NTRTRY	2	22	M	NTWTRY	2	23	M
NU	7	36	S	NUMEM	6	24	M	NUSMT	6	13	M	NXP0P	4	22	PAC
NXSMT	6	17	M	0FFINT	14	25	M	0HM	1	4	RUP	0K	26	20	PAC
0KX	26	19	PAC	0NE	2	7	S	0PDSTR	10	31	TS84A	0PNCN	5	38	TS84A
0PSS1	10	6	SET	0PSS10	10	31	SET	0PSS11	10	36	SET	0PSS2	10	8	SET
0PSS3	10	10	SET	0PSS4	10	17	SET	0PSS5	10	21	SET	0PSS6	10	26	SET
0PSS7	10	28	SET	0PTM	10	39	TS84A	0UTFIL	5	35	TS84A	0UTMSG	2	13	RUP
0UTN01	1	14	TS84A	0UTN02	1	15	TS84A	0UTN03	1	16	TS84A	0UTN04	1	17	TS84A
0UTN05	1	18	TS84A	0UTN06	37	7	RUP	0UTN07	37	9	RUP	0UTNUM	36	39	RUP
0UTSTR	2	29	RUP	0UTSx	10	1	TS84A	0VDGTS	10	13	TS84A	0VFL	9	23	TS84A
0VFP	5	24	TS84A	0VFP1	5	26	TS84A	0VFP2	5	27	TS84A	PA	8	31	M
PAC	2	23	PMT	PAC1C	6	16	PAC	PAC2B	8	11	PAC	PAC9A	8	24	PAC
PACAC2	9	37	PAC	PACAC3	9	36	PAC	PACAC4	10	7	PAC	PACACT	9	10	PAC
PACDEL	10	4	PAC	PACDMB	1	32	PAC	PACDMS	14	14	PAC	PACG21	7	11	PAC
PACG22	7	12	PAC	PACG0	7	3	PAC	PACG01	6	21	PAC	PACG02	7	6	PAC
PACG04	7	15	PAC	PACG05	7	22	PAC	PACG06	7	17	PAC	PACG07	7	13	PAC
PACG08	7	35	PAC	PACG09	8	14	PAC	PACG0A	6	20	PAC	PACINT	13	19	PAC
PACJ0B	13	14	PAC	PACLVL	14	16	PAC	PAC0MP	12	34	PAC	PAC0VF	7	37	PAC
PACPTR	5	2	S	PACQ2	1	34	PAC	PACQ3	1	35	PAC	PACQ4	1	36	PAC
PACQE	4	39	PAC	PACQM1	4	38	PAC	PACQT	13	13	PAC	PACQUE	14	15	PAC
PACSCN	9	3	PAC	PACSRT	7	36	PAC	PACST	6	26	S	PACSW	13	10	PAC
PACSWP	13	12	PAC	PACSWQ	13	11	PAC	PACT	3	20	PMT	PACT1	3	21	PMT
PACTP2	13	30	PAC	PACTRP	13	28	PAC	PACWT	10	35	PAC	PACWT2	10	36	PAC
PACWT3	10	39	PAC	PAGES	8	4	S	PAGEW	8	7	S	PB	8	32	M
PCE	1	37	I	PCEX	1	38	I	PCEX1	4	31	D	PDATA	3	22	PMT
PDC	10	19	TS84A	PDL	10	20	TS84A	PDMS2	5	8	PAC	PDMS3	5	11	PAC
PDMS4	5	10	PAC	PDP	10	18	TS84A	PE	23	5	D	PE1	23	12	D
PE2	23	22	D	PE3	23	25	D	PE4	23	26	D	PE5	23	27	D
PE6	23	9	D	PE7	23	11	D	PE9	23	28	D	PE9A	23	33	D
PEB2	25	34	B	PEBRS	25	26	B	PERI0D	9	5	TS84A	PEST	9	5	PAC
PEST2	9	29	PAC	PEST3	9	21	PAC	PEST4	9	9	PAC	PEST5	9	6	PAC
PG44	13	1	PAC	PGET	19	34	PAC	PGET1	20	5	PAC	PGMSK	4	27	S
PH2	1	12	M	PI1	13	23	PAC	PIGGY	9	2	R	PIM	3	30	PMT
PIM:	2	33	PMT	PJ	5	5	TS84A	PJ2	5	6	TS84A	PJCSW	10	9	TS84A
PL	3	24	PMT	PL4	4	2	PMT	PL:	2	25	PMT	PLMSK	4	23	S
PLUS		3	TS84A	PMG5	6	30	B	PMGET		25	B	PMI11	11	39	B
PMI2		10	B	PMI3	12	6	B	PMI4		11	B	PMI5	12	16	B

PMTE	4	36	PMT	PMTI	11	38	B	PMTJ0B	5	16	S
PMTP	4	39	S	PMTR	11	30	B	PNEXT	4	23	PMT
PNX3	4	1	PMT	PNXTP1	3	39	PMT	P0FFA	28	7	PAC
P0FFX	28	9	PAC	P0PD9	5	20	PAC	P0PD9A	5	35	PAC
P0PINT	5	18	PAC	P0PR	4	12	PAC	P0PR2	4	13	PAC
P0PR5	4	17	PAC	P0PR6	4	20	PAC	P0PST	5	13	PAC
P0PTB	7	32	SET	P0PX	4	28	PAC	P0PX0	4	25	PAC
P0SL	8	26	W	PPAN	18	38	B	PPB	8	38	M
PPREV	14	17	PAC	PPTEST	14	18	PAC	PPTR	3	27	PMT
PPTR:	2	28	PMT	PPTRU	4	4	PMT	PRD	8	3	W
PRD4	8	29	W	PRD6	9	3	I	PRDI	8	33	W
PRDI3	8	35	W	PRDI4	9	4	W	PRDMS	8	22	W
PRGRL	4	14	TS84A	PRGRL1	3	38	TS84A	PRGRL2	3	39	TS84A
PRMA1	7	23	TS84A	PRMSK	4	24	S	PR0TEC	1	31	SET
PRSK	8	31	W	PRWC	9	5	W	PSDEX	9	4	M
PSDEX2	17	38	B	PSP	1	7	S	PSPJ	1	18	S
PSPJ	1	12	S	PSPJ	1	10	S	PSPN	1	8	S
PTAB	3	29	PMT	PTAB:	2	32	PMT	PTCKS	19	30	I
PTEST:	2	31	PMT	PTESTS	14	26	PAC	PTICKB	1	20	M
PTR3	7	29	TS84A	PTR5	7	30	TS84A	PTRAP	19	26	B
PTRL1	20	4	B	PTRL2	20	5	B	PTRL3	20	6	B
PTSTAT	7	28	TS84A	PU124A	17	22	PAC	PU124B	17	27	PAC
PUACT	18	2	PAC	PUACT1	18	8	PAC	PUACT2	18	1	PAC
PUBPTR	19	9	PAC	PUCLST	16	21	PAC	PUCPTR	19	12	PAC
PUCR2	18	25	PAC	PUCSET	18	22	PAC	PUCT	19	31	I
PUCTR1	19	11	PAC	PUDEAD	19	15	PAC	PUDMS	15	4	PAC
PUER2	18	30	PAC	PUG0	16	1	PAC	PUG02	15	34	PAC
PUG03A	15	24	PAC	PUG04	15	37	PAC	PUG05	16	6	PAC
PUINIT	14	39	PAC	PUINT	17	32	PAC	PUJ0B	19	27	PAC
PUL0	17	5	PAC	PUNXT	16	18	PAC	PUPAC	19	14	PAC
PURBT	17	8	PAC	PURBT1	17	17	PAC	PURBT2	17	19	PAC
PURBTA	17	15	PAC	PURL1	19	20	PAC	PURL2	19	21	PAC

PMTM1	1	34	PMT
PNEXT:	2	24	PMT
P0FFB	28	8	PAC
P0PDMS	5	4	PAC
P0PR4	4	19	PAC
P0PST2	5	14	PAC
P0PXB	4	32	PAC
PP0FF	15	16	PAC
PPTR2	4	3	PMT
PRD2	8	23	W
PRDI2	8	37	W
PRDMS1	9	4	I
PRILEV	5	13	S
PRSET	8	24	W
PSDEX1	17	37	B
PSPJ	1	16	S
PSSTK	9	3	M
PTEST	3	28	PMT
PTR2	7	27	TS84A
PTRL	24	18	PAC
PTRPA	20	7	B
PU3SEC	15	7	PAC
PUACTA	18	15	PAC
PUCR1	17	38	PAC
PUCTR	19	10	PAC
PUEPTR	19	13	PAC
PUG03	15	12	PAC
PUG06	15	25	PAC
PULIM	19	17	PAC
PUPACP	19	18	PAC
PURBT3	17	20	PAC
PURL3	19	22	PAC

PURRL1	19	23	PAC	PURRL2	19	24	PAC	PURRL3	19	25	PAC	PURUN	13	15	PAC
PUSCN	16	7	PAC	PUSET	16	9	PAC	PUST1	15	29	PAC	PUST2	15	31	PAC
PUSW	19	19	PAC	PUSWER	16	27	PAC	PUTIM	16	25	PAC	PUTIM1	16	33	PAC
PUTIM2	16	34	PAC	PUTIM3	16	37	PAC	PUTIM8	17	2	PAC	PUTIM9	17	1	PAC
PUTIME	19	26	PAC	PUTST	19	16	PAC	PUTTY	19	8	PAC	PUXSV	19	4	PAC
PVTSW	5	37	TS84A	PWFI	28	2	PAC	PWFL	7	24	S	PWNI	28	6	PAC
PX	8	33	M	Q0	2	7	PMT	Q1	2	18	PMT	Q25	33	35	PAC
Q26	33	36	PAC	QCK	29	22	PAC	QCK2	29	25	PAC	QCK3	29	24	PAC
QCK4	29	23	PAC	QCK5	29	26	PAC	QD	3	32	PMT	QGET	31	2	PAC
QGET	30	3	PAC	QGET2	30	15	PAC	QGET3	30	16	PAC	QGET4	30	17	PAC
QGET5	30	14	PAC	QGET6	30	18	PAC	QGET7	30	19	PAC	QGET8	30	20	PAC
QI0	2	10	PMT	QI0C	2	11	PMT	QI0Q	3	34	PMT	QPUT	30	27	PAC
QPUT	29	30	PAC	QPUT3	30	28	PAC	QPUT4	29	38	PAC	QPUT5	29	39	PAC
QQC	2	14	PMT	QQCC	2	15	PMT	QQCQ	3	36	PMT	QQE	2	16	PMT
QQEC	2	17	PMT	QQEDMS	5	1	PAC	QQEQ	3	37	PMT	QS	10	14	B
QSCH	31	10	PAC	QSCH1	31	21	PAC	QSCH2	31	16	PAC	QSCH3	31	19	PAC
QSCH4	31	22	PAC	QSCH5	31	18	PAC	QSCH6	31	23	PAC	QSQ	2	12	PMT
QSQC	2	13	PMT	QSQQ	3	35	PMT	QTI	2	8	PMT	QTIC	2	9	PMT
QTIQ0	33	34	PAC	QTIQ	3	33	PMT	QTPUT	30	36	PAC	RAQ1	9	21	R
RAA	11	2	R	RAAL	8	23	R	RAAV	4	2	R	RAAW	9	9	R
RAAWA	9	11	R	RAAWS	9	14	R	RAAWX	9	15	R	RAB	11	3	R
RABC	7	4	R	RABD	8	13	R	RABD1	8	18	R	RABD2	8	20	R
RABE	8	39	R	RABN	8	3	R	RAB0L	10	38	R	RABR	8	36	R
RABRQ	8	35	R	RABS	8	2	R	RABW	8	33	R	RABWQ	8	31	R
RACHH	10	34	R	RACHKW	10	14	R	RACKWD	11	32	R	RACL	6	18	R
RACLGH	11	6	R	RACUR	10	30	R	RACURX	10	31	R	RADDR	10	39	R
RADDRC	11	36	R	RADEL	5	5	R	RADI2	9	16	R	RADIA	2	3	R
RADIF	2	4	R	RADIG	2	5	R	RADRCC	11	37	R	RADTS	5	2	R
RADTYM	3	14	SET	RAE0D	10	16	R	RAE0DR	10	17	R	RAER	23	35	PAC
RAER1	23	36	PAC	RAER2	23	37	PAC	RAER3	23	38	PAC	RAER4	23	39	PAC
RAER5	24	1	PAC	RAER6	24	2	PAC	RAER7	24	3	PAC	RAERCT	3	1	R
RAERL	2	38	R	RAERLP	2	39	R	RAER0R	23	34	PAC	RAERT	1	7	SET
RAERX	2	23	R	RAEW	3	3	R	RAEX	3	9	R	RAEXIT	11	7	R
RAEXL	4	33	R	RAEXX	3	7	R	RAF0N	5	16	R	RAF0NX	5	17	R
RAFR	10	37	R	RAG0	9	18	R	RAG01	9	19	R	RAG0H	3	36	R
RAG0L	3	37	R	RAGQS	7	2	R	RAGRL	9	35	R	RAGS	4	12	R
RAGT	8	6	R	RAGTX	8	11	R	RAGTY	8	8	R	RAI1	1	37	R
RAIND	6	3	R	RAINS	5	11	R	RAJ	3	38	R	RAJ0B	23	25	PAC
RAJ0B1	23	26	PAC	RAJ0B2	23	27	PAC	RAJ0B3	23	28	PAC	RAJ0B4	23	29	PAC

RAJ0B5	23	30	PAC	RAJ0B6	23	31	PAC	RAJ0B7	23	32	PAC	RAJ0BE	23	33	PAC
RALIT	3	24	R	RALITX	3	25	R	RALL	3	23	R	RAL00P	3	15	R
RAMARK	6	22	R	RANCX	3	34	R	RANE	11	34	R	RAN0CR	3	33	R
RAN0P	10	15	R	RAPH	10	29	R	RAPHTB	9	36	R	RAPMT	10	24	R
RAP0ST	3	14	R	RAPSTS	11	28	R	RAPUT	6	38	R	RAQS	7	6	R
RAQSH	8	27	R	RAQSH1	8	28	R	RARE	2	34	R	RARELB	4	36	R
RARE0D	10	19	R	RARET	8	25	R	RAREW	2	36	R	RARFQ	11	8	R
RARHP	10	32	R	RARLB	7	18	R	RARLBX	7	19	R	RARLC	7	10	R
RARLP	10	33	R	RARLPH	10	28	R	RARLQS	6	8	R	RARLQT	6	16	R
RARLQT	6	13	R	RARLR	7	24	R	RARLRQ	7	21	R	RARLS	7	9	R
RARLX	10	21	R	RARM	6	24	R	RARN	11	10	R	RARNX	11	11	R
RARQ	11	9	R	RARSB	7	31	R	RARSC	7	28	R	RARSD	7	30	R
RARSE	7	39	R	RARSI	8	1	R	RARSRQ	7	34	R	RARSTS	7	32	R
RARSW	7	33	R	RARSWQ	7	37	R	RASCT	10	22	R	RASQH	11	5	R
RASS	3	28	R	RASW	1	38	R	RASWA	3	11	R	RASWB	2	17	R
RASWBX	2	14	R	RATS	10	25	R	RATSA	10	26	R	RAV0ID	6	23	R
RAWB	9	7	R	RAWE0D	10	18	R	RAW0RK	10	27	R	RAWSC	10	23	R
RAWWQ	9	8	R	RAX	11	4	R	RAXE0D	10	20	R	RAZ	9	25	R
RAZAP	8	24	R	RAZZ	9	31	R	RBER	8	14	S	RCER	11	29	R
RCJ	32	10	PAC	RCL	6	25	R	RCP	2	5	I	RCPARW	25	20	B
RCPX	2	6	I	RCPX1	5	17	D	RDCLEC	6	18	B	RDER	11	30	R
RDP	2	15	M	RDRL	13	13	B	RE	1	13	M	REAL	7	38	S
REDBRS	23	26	B	REL	5	25	R	RELMEM	12	20	PAC	RELPG	12	14	PAC
RELPG2	12	27	PAC	RELPG3	12	26	PAC	RELPG4	12	21	PAC	RELPG5	12	24	PAC
RELPG6	12	25	PAC	RELPG7	12	23	PAC	RELPG8	12	28	PAC	REMAC	4	16	TS84A
RES	5	29	R	RESBRS	23	24	B	RESMT	10	28	B	RESMT2	10	32	B
RF02	30	30	B	RF03	30	12	B	RF04	30	28	B	RF06	29	30	B
RF07	29	37	B	RF08	30	31	B	RF10	29	36	B	RF11	29	39	B
RF12	29	24	B	RF13	30	32	B	RF14	30	33	B	RF15	30	34	B
RF16	30	21	B	RF17	30	35	B	RF32	30	36	B	RF33	30	37	B
RFC01	30	38	B	RFK	29	22	B	RFP0V	12	34	M	RFR	11	31	R
RFRL1	30	39	B	RFRL2	31	1	B	RFRL3	31	2	B	RIIR	32	39	PAC
RL1	3	25	PMT	RL1:	2	26	PMT	RL2	3	26	PMT	RL2:	2	27	PMT
RL3	6	2	S	RLABEL	6	3	I	RLCT	9	3	B	RLDSC	25	2	PAC
RLHLDA	8	37	B	RLHLDB	9	2	B	RLHLDD	8	38	B	RLH0LD	8	36	B
RLI01	9	30	TS84A	RLI02	9	31	TS84A	RLI04	9	32	TS84A	RLI05	9	33	TS84A
RLI06	9	34	TS84A	RLI07	9	35	TS84A	RLITEX	10	3	TS84A	RLPACT	25	1	PAC
RLTS	24	39	PAC	RLTYM	25	4	PAC	RLV4	10	26	TS84A	RLWB	25	3	PAC
RM1	3	19	SET	RMC	11	20	R	RMFF	9	25	M	RML	11	22	R

RML32	11	23	R	RML33	11	24	R	RMR	11	25	R	RMR32	11	26	R
RMR33	11	27	R	RMT	11	19	R	RNDFLG	9	36	TS84A	RNDX	10	2	TS84A
RP	1	18	M	RPAGE	8	6	S	RPAGEH	10	35	R	RPAGEL	10	36	R
RPCHG	11	26	TS84A	RPMAX	11	27	TS84A	RPN	11	28	TS84A	RPRPS	17	2	B
RPTM	10	38	TS84A	RQ1	15	38	I	RQ1A	15	37	I	RQ2	15	39	I
RQ3	16	1	I	RREAL	28	16	PAC	RRL1	24	36	PAC	RRL2	24	37	PAC
RRL3	24	38	PAC	RRRL	8	16	B	RS	7	27	S	RSEND	6	3	TS84A
RSMET	10	22	B	RSP	2	19	M	RSP	2	11	I	RSPX	2	12	I
RSPX1	5	30	D	RSR	2	17	M	RSREAL	6	2	TS84A	RST	5	33	M
RSYS2	7	39	TS84A	RTEX	11	16	B	RTR	2	18	M	RURL	13	16	B
RW1	1	35	SET	RW2	1	38	SET	RW3	2	2	SET	RW4	2	5	SET
RWD	5	28	S	RWP	2	16	M	RWR	5	29	S	RWTAB	23	38	B
RWTAB1	24	10	B	RWTABN	25	1	B	RX1	8	32	B	RX11	8	30	B
RX12	8	19	B	RX13	8	22	B	RX2	8	29	B	RX3	8	21	B
RX32	8	27	B	RX33	8	2	B	RX4	8	13	B	RX5	8	15	B
RX6	8	8	B	RX7	8	9	B	RX8	8	35	B	RXNTS	8	7	B
S44T10	16	9	B	S44T11	14	22	B	S44T12	16	11	B	S44T13	14	26	B
S44T14	14	6	B	S44T15	14	33	B	S44T17	14	31	B	S44T18	14	30	B
S44T19	14	27	B	S44T2	14	10	B	S44T20	14	38	B	S44T21	14	37	B
S44T22	15	2	B	S44T23	15	3	B	S44T24	15	7	B	S44T25	15	31	B
S44T27	13	32	B	S44T28	15	11	B	S44T29	15	20	B	S44T3	13	26	B
S44T30	13	33	B	S44T31	13	37	B	S44T33	15	10	B	S44T34	15	23	B
S44T35	15	27	B	S44T4	13	29	B	S44T41	15	15	B	S44T42	15	16	B
S44T43	16	8	B	S44T44	15	19	B	S44T5	16	10	B	S44T51	14	21	B
S44T53	14	18	B	S44T6	14	23	B	S44T61	14	20	B	S44T62	14	17	B
S44T7	13	28	B	S44T8	14	24	B	S44T9	14	12	B	SA	26	30	PAC
SAFE	1	22	R	SAIR	33	13	B	SAVEFL	8	37	TS84A	SAVELL	8	36	TS84A
SAVEOR	8	38	TS84A	SAVESL	8	39	TS84A	SB	26	31	PAC	SB	1	27	R
SBCT	9	10	W	SBRM	1	21	B	SBRME	1	22	B	SBRR	1	26	B
SBRS	1	23	M	SBRS	1	10	RUP	SBRSRT	1	12	TS84A	SC	1	25	R
SC1	18	14	RUP	SC1	15	19	RUP	SCAN	9	2	PAC	SCEOF1	21	36	D
SCF01	28	6	B	SCF02	28	7	B	SCF03	28	8	B	SCF04	28	9	B
SCF05	28	5	B	SCF06	27	38	B	SCF07	28	1	B	SCFK	27	36	B
SCH1	10	33	RUP	SCH10	10	37	RUP	SCH11	10	38	RUP	SCH20	10	39	RUP
SCH4	10	35	RUP	SCH5	10	34	RUP	SCH52	11	1	RUP	SCH9	10	36	RUP
SCHLP	13	3	PAC	SCIO	1	9	RUP	SCIT	1	7	RUP	SCP	2	1	I
SCPX	2	2	I	SCPX1	5	3	D	SCQ	1	34	R	SCRL	18	7	B
SD	1	26	R	SD1	11	31	SET	SD2	11	39	SET	SDBM	17	7	D
SDBM8	5	24	S	SDBMA	17	12	D	SDBMA1	17	23	D	SDBMA2	17	27	D

SDBMA3	17	25	D	SDBMA4	17	28	D	SDBMB	17	30	D	SDBMBD	17	10	D
SDBMC	17	36	D	SDSKMT	8	8	S	SDUMP	11	25	SET	SE	1	32	R
SET	5	30	M	SET1	4	14	SET	SET2	5	30	SET	SET2	5	8	SET
SET3	4	20	SET	SET4	4	35	SET	SET4	4	30	SET	SET6	4	24	SET
SET7	3	34	SET	SET9	3	32	SET	SETBIP	22	6	D	SETCC	21	31	D
SETCC1	21	34	D	SETFRE	6	24	SET	SETINT	9	23	M	SETPAC	11	25	TYM
SETPAR	7	7	I	SETPC2	11	33	TYM	SETREL	11	23	PAC	SETRL	1	26	W
SETSA	3	26	SET	SETSET	29	14	PAC	SETSW	26	2	B	SETSWP	12	6	PAC
SETTB	7	39	S	SETXX	1	25	W	SEVEN	2	13	S	SFPEX	25	34	RUP
SFPEX1	25	36	RUP	SFPEX2	25	39	RUP	SFPFG	25	30	RUP	SFP0VI	27	28	PAC
SFSTUP	26	4	RUP	SG	1	20	R	SG1	11	12	SET	SHFAN	13	1	M
SHFK	27	28	B	SI	1	33	R	SIC	18	11	RUP	SIC	15	16	RUP
SIC1	18	12	RUP	SIC1	15	17	RUP	SIC2	18	10	RUP	SIC2	15	15	RUP
SIGN	9	26	TS84A	SIX	2	12	S	SJB	31	36	PAC	SKC0	9	10	RUP
SKC01	9	15	RUP	SKC02	9	19	RUP	SKC03	10	11	RUP	SKC04	9	26	RUP
SKC05	10	15	RUP	SKFNE	23	26	RUP	SKFNE	23	21	RUP	SKFUZE	24	14	RUP
SKFUZE	24	10	RUP	SKFZE	23	38	RUP	SKFZE	23	34	RUP	SKL	5	30	I
SKLX	5	31	I	SKP	5	28	I	SKPX	5	29	I	SKR2	5	37	B
SKR0UT	5	35	B	SKS1	10	26	RUP	SKS2	10	27	RUP	SKS3	10	28	RUP
SKS4	10	29	RUP	SKS5	10	30	RUP	SKS6	10	31	RUP	SKS7	10	32	RUP
SKSE	7	35	RUP	SKSE1	8	21	RUP	SKSE2	8	15	RUP	SKSEP	7	36	RUP
SKSG	8	26	RUP	SKSGP	8	27	RUP	SKXEC	19	12	B	SLDAR	9	34	R
SLLP	22	25	PAC	SLLP2	22	19	PAC	SLLP3	22	21	PAC	SLLP4	22	22	PAC
SLPGC	23	23	PAC	SLRET	22	29	PAC	SLSYS	22	30	PAC	SMFF	9	26	M
SMT	1	22	PMT	SMTAD	6	18	SET	SMTE	1	30	PMT	SMTPG7	1	21	PMT
SN	1	23	R	SNE	5	26	I	SNEX	5	27	I	SN0P	14	29	PAC
SNTRP	14	30	PAC	S0V	7	6	S	SP	6	7	B	SP1	6	8	B
SPACES	9	22	TS84A	SPARE1	7	30	S	SPARE2	7	31	S	SPR	13	6	PAC
SPT	8	3	S	SPURL	16	12	PAC	SR	1	31	R	SRB	1	27	B
SRIN	22	6	PAC	SRIR	33	9	B	SR0UT	5	39	B	SRQ	1	30	R
SRSU1	8	30	TS84A	SRSU20	8	31	TS84A	SRSU21	8	33	TS84A	SRSU3	8	34	TS84A
SRSU4	8	35	TS84A	SRT	23	15	PAC	SRT44E	23	16	PAC	SRTE	23	17	PAC
SS	1	24	R	SSQ1	2	1	B	SSQ2	2	2	B	SSQ3	2	3	B
SS9	9	36	SET	SS9A	10	4	SET	SS9B	10	3	SET	SSA	6	17	TS84A
SSB	6	18	TS84A	SSET	4	7	SET	SSID	5	32	TS84A	SSKSE	1	11	RUP
SSL	6	16	TS84A	SSL0OP	9	24	SET	SSM	6	22	TS84A	SSP	2	8	I
SSPX	2	9	I	SSPX1	5	20	D	SSPX2	5	28	D	SSR1	6	20	TS84A
SSR2	6	21	TS84A	SSRET	6	12	B	SSRL	4	15	TS84A	SSRL1	4	1	TS84A
SSRL1	2	23	TS84A	SSRL2	4	2	TS84A	SSRL2	2	24	TS84A	SSS	9	22	SET

SST	5	10	S	SSTIME	5	31	TS84A	SSWC	8	5	S	SSX	6	19	TS84A
SSYSTL	10	25	TS84A	ST	1	9	M	STACT	9	9	S	STATUS	5	14	S
STC	6	9	B	STDISC	6	11	B	STDMS	16	1	B	STFK	26	30	B
STFK2	26	38	B	STFK3	26	39	B	STFLE	20	14	RUP	STFLE	20	11	RUP
STFLE2	20	12	RUP	STFME	19	27	RUP	STFME1	19	34	RUP	STFME1	19	29	RUP
STFME2	19	32	RUP	STFMEF	20	3	RUP	STG0	6	5	B	STG01	11	3	SET
STIME	8	2	S	STKPS	16	26	B	STOP	33	24	PAC	STP	4	32	RUP
STP1	10	19	RUP	STP2	11	21	SET	STPP	4	33	RUP	STRL	13	21	B
STRL1	16	14	B	STRL10	18	16	B	STRL11	18	23	B	STRL2	16	15	B
STRL3	18	9	B	STRL4	18	21	B	STRL41	18	18	B	STRL5	18	26	B
STRL51	18	27	B	STRL52	18	28	B	STRL53	18	29	B	STRL6	18	25	B
STRL61	18	31	B	STRL7	18	20	B	STRL7A	18	11	B	STRL8	18	19	B
STRL9	18	30	B	STRLC	16	13	B	STRLD	16	12	B	STRLQ	13	23	B
STR01	1	22	TS84A	STR02	1	23	TS84A	STR021	1	24	TS84A	STR03	1	25	TS84A
STR04	2	38	RUP	STR05	3	7	RUP	STR06	2	31	RUP	STR07	3	10	RUP
STR07A	3	28	RUP	STR08	3	5	RUP	STSTAT	8	27	PAC	STSTOP	6	14	B
STSTP1	11	15	SET	STSW	6	10	B	SUPAGE	2	27	M	SURL	16	18	B
SUSR1	8	25	TS84A	SUSR2	8	26	TS84A	SUSR3	8	28	TS84A	SVSSL	10	23	TS84A
SVST	11	24	TS84A	SW	1	29	R	SW	10	8	TS84A	SW16	5	27	S
SW205	5	30	S	SW60	4	28	TS84A	SWAP	20	22	PAC	SWAPI	22	39	PAC
SWAPL	22	16	PAC	SWAPR	22	1	PAC	SWAPR2	22	7	PAC	SWCI	1	8	RUP
SWCT	1	35	R	SWEX	6	24	S	SWFD	6	35	S	SWIN	21	7	PAC
SWIN10	21	32	PAC	SWIN11	21	30	PAC	SWIN12	21	29	PAC	SWIN13	21	25	PAC
SWIN14	21	6	PAC	SWIN2	21	13	PAC	SWIN3	21	9	PAC	SWIN4	21	24	PAC
SWIN5	21	27	PAC	SWIN6	21	17	PAC	SWIN7	21	33	PAC	SWIN8	21	8	PAC
SWIN9	21	28	PAC	SWINIT	4	32	TS84A	SWIX	23	13	PAC	SWJOB	23	6	PAC
SWLET	6	25	S	SWLP	20	32	PAC	SWLP1	20	28	PAC	SWLS	23	22	PAC
SWMEL	6	32	S	SWMBM	6	33	S	SWNCM	23	21	PAC	SWOFF	5	3	TS84A
SWPB	23	14	PAC	SWPCNT	19	28	I	SWPMT	23	8	PAC	SWPMTI	23	9	PAC
SWPRP	4	29	TS84A	SWQ	1	28	R	SWR1	23	18	PAC	SWR2	23	19	PAC
SWR3	23	20	PAC	SWRAB	23	11	PAC	SWRAI	23	12	PAC	SWRAJ	23	7	PAC
SWREL	11	30	PAC	SWRL1	24	24	PAC	SWRL3	12	3	PAC	SWRL4	11	37	PAC
SWRL5	11	38	PAC	SWRL6	11	39	PAC	SWRL7	11	33	PAC	SWRL8	11	35	PAC
SWRL9	11	36	PAC	SWSK0	20	39	PAC	SWSKOA	21	1	PAC	SWSK2	21	3	PAC
SWSMT	23	10	PAC	SWT3	24	15	PAC	SWTM	5	4	TS84A	SWXMA	5	31	S
SX	26	32	PAC	SXWR31	10	36	TS84A	SYMBOL	1	28	I	SYSRRL	7	38	TS84A
SYSS	5	12	S	SYSTL	6	24	TS84A	T	1	34	B	T1	1	35	B
T1	3	10	TS84A	T10	3	19	TS84A	T11	3	20	TS84A	T12	5	15	TS84A
T128	3	4	M	T128A	11	8	I	T13	5	16	TS84A	T14	5	17	TS84A

T2	1	36	B	T2	3	11	TS84A	T3	1	37	B	T3	3	12	TS84A
T4	1	38	B	T4	3	13	TS84A	T5	3	14	TS84A	T6	3	15	TS84A
T7	3	16	TS84A	T8	3	17	TS84A	T9	3	18	TS84A	TABLEC	6	38	M
TABLEC	6	36	M	TABLEN	7	1	M	TABLEN	11	35	D	TBACKP	19	32	TYM
TBI	2	34	M	TBØ	2	36	M	TBRK	18	15	TYM	TBRX1X	4	9	NET
TBRX2X	4	10	NET	TBSF	7	8	W	TBSFI	7	11	W	TBSFI3	7	18	W
TBSR	5	33	W	TC	3	6	M	TCA	8	20	I	TCI1	2	35	TYM
TCICIØ	2	34	TYM	TCIT	2	37	TYM	TCIX	2	27	TYM	TCL2	7	15	I
TCLDMS	7	19	I	TCØ1	5	1	TYM	TCØ2	5	5	TYM	TCØ3	5	7	TYM
TCØ3A	5	10	TYM	TCØ3B	5	12	TYM	TCØCHR	4	34	TYM	TCØX	4	29	TYM
TCZN	11	2	TS84A	TD2	7	28	I	TDELAY	18	24	TYM	TDS	7	17	I
TDT	9	24	M	TEN	2	16	S	TEØF	6	20	W	TEØF3	6	25	W
TEØF4	6	24	W	TEØFI	6	26	W	TERS	6	13	W	TERS3	6	17	W
TF01	29	9	B	TF02	29	1	B	TF05	29	13	B	TF06	29	14	B
TF07	29	15	B	TF09	29	8	B	TF1	3	1	M	TF10	29	11	B
TF12	29	10	B	TF13	29	12	B	TF2	2	22	TYM	TF3	3	2	M
TF3X1	20	11	TYM	TF3X2	20	17	TYM	TF3XMA	20	5	TYM	TF4	3	3	M
TFC01	29	16	B	TFC02	29	17	B	TFC03	8	22	PAC	TFC04	8	23	PAC
TFICTR	7	29	S	TFILE	9	19	TS84A	TFIN	7	30	I	TFIP	12	9	TYM
TFIP1	12	15	TYM	TFIP2	12	14	TYM	TFIP3	12	16	TYM	TFK	28	14	B
TFKA	28	29	B	TFREE	2	23	TYM	TFREEC	2	24	TYM	TFSF	6	31	W
TFSF2	6	34	W	TFSFI	6	35	W	TFSFI3	7	4	W	TG1	3	2	NET
TG1A	3	4	NET	TG1C	4	7	NET	TG2	3	14	NET	TG3	3	16	NET
TG5	3	19	NET	TG5C	4	8	NET	TG6	3	29	NET	TG6X	4	11	NET
TGBRX1	3	13	NET	TGBRX2	3	28	NET	TGC	1	8	NET	TGC1	1	9	NET
TGC3	1	16	NET	TGC5	1	17	NET	TGCE	1	19	NET	TGCEO	1	22	NET
TGCE7	1	28	NET	TGCINC	1	32	NET	TGCK	2	30	NET	TGCKO	2	32	NET
TGCK1	2	33	NET	TGCK2	2	34	NET	TGCK3	2	35	NET	TGØBBL	18	39	TYM
TGSUM	2	37	NET	TGSUM1	2	38	NET	TH1	12	28	TYM	THARD	9	26	TYM
THIRD	2	11	TYM	THREE	2	9	S	TIC1	11	5	TYM	TIIBAD	7	33	S
TIICTR	7	32	S	TIIR	4	20	TYM	TIIRX	4	27	TYM	TIME	25	29	PAC
TIME1	25	31	PAC	TIME1	9	16	TS84A	TIME2	9	17	TS84A	TIME3	9	18	TS84A
TINC1	1	33	NET	TINC2	1	34	NET	TINC2X	1	36	NET	TINC3	1	37	NET
TINC3X	2	1	NET	TIP	11	2	TYM	TIPIX	11	6	TYM	TIX	4	29	NET
TJØB	5	15	S	TLABEL	9	27	M	TLØG	1	6	NET	TLØGP	6	8	NET
TLV0	5	18	TS84A	TLV1	6	28	TS84A	TLV11	6	29	TS84A	TLV2	6	30	TS84A
TLV21	6	31	TS84A	TLV22	6	32	TS84A	TLV23	6	33	TS84A	TLV3	6	34	TS84A
TLV4	10	27	TS84A	TMAR	8	24	I	TMCA	8	30	I	TMK1	9	12	S
TMK2	9	13	S	TNICTR	7	28	S	TNR	8	22	I	TNUM	8	26	I

T0	5	23	TYM	T0102	5	35	TYM	T0105	5	37	TYM	T0135	6	2	TYM
T0152	6	7	TYM	T0152A	6	10	TYM	T0155	6	13	TYM	T0155A	6	15	TYM
T0IBAD	7	35	S	T0ICTR	7	34	S	T0N	5	29	TYM	T0N2	5	33	TYM
T0N3	5	32	TYM	T0X	6	4	NET	TP	3	5	M	TPAIR1	5	5	NET
TPAIR2	5	6	NET	TPAIRX	5	7	NET	TPAR	8	25	I	TPDMS	8	28	I
TPDMS2	2	33	W	TPDMS3	8	29	I	TPE	8	36	I	TP0RT	2	25	TYM
TPR22	20	27	B	TPR25	20	21	B	TPR26	21	4	B	TPR3	21	2	B
TPR4	21	5	B	TPSET	2	38	W	TPSET2	3	7	W	TPUH	12	24	TYM
TPUS	12	18	TYM	TQ	6	32	TYM	TQ1	6	36	TYM	TQA	6	21	TYM
TQB	6	22	TYM	TQC	6	20	TYM	TQCHR	6	29	TYM	TQCHRF	6	30	TYM
TQD	7	4	TYM	TQD1	7	12	TYM	TQD12	7	16	TYM	TQD15	7	18	TYM
TQDCHR	7	14	TYM	TQP0RT	6	27	TYM	TQRET	6	25	TYM	TQRL	6	24	TYM
TQTAB	7	20	TYM	TQTYPE	6	28	TYM	TQX	6	23	TYM	TR00	21	6	B
TR01	21	7	B	TR02	21	8	B	TR03	21	9	B	TRA	8	23	I
TRAP	9	18	M	TRAP	20	10	B	TRAPI	19	18	B	TRAPM	21	16	B
TRAPR	20	14	B	TRAPS	20	9	B	TRAPT	27	33	PAC	TRAPT	27	1	PAC
TRAPT1	27	36	PAC	TRAPT1	27	14	PAC	TRAPT2	27	20	PAC	TRATE	2	39	M
TRC	10	16	TYM	TRC1	10	19	TYM	TRC2	10	28	TYM	TRC3	10	30	TYM
TRC3X	10	33	TYM	TRCTIM	10	34	TYM	TRD	7	22	I	TRD2	7	24	I
TRD4	2	32	W	TRD5	2	28	W	TREAD	2	14	W	TREAL	8	37	I
TREW	5	28	W	TREW2	5	29	W	TREW3	5	30	W	TRI	4	14	NET
TRI1	3	29	W	TRI1	4	25	NET	TRI10	3	33	W	TRI2	3	30	W
TRI3	3	31	W	TRI5	3	21	W	TRI6	3	25	W	TRI8	3	20	W
TRI9	3	32	W	TRIC1	4	30	NET	TRINT	3	10	W	TR0	5	20	NET
TR01	5	23	NET	TR02	6	1	NET	TR02A	5	39	NET	TR09	6	5	NET
TR0M	5	16	NET	TR0PAI	4	32	NET	TR0SW	5	20	TYM	TR0SW1	5	21	TYM
TR0TY1	5	9	NET	TR0X	6	3	NET	TRP	9	30	M	TRPXMA	21	10	B
TRQ	32	4	PAC	TS	6	26	SET	TSA	6	27	SET	TSMAPS	8	4	M
TSMAPS	8	2	M	TSN	1	24	M	TS0FT	9	16	TYM	TS0FT1	9	22	TYM
TS0N	11	8	TYM	TS0N2	11	17	TYM	TS0NC1	11	22	TYM	TS0NC2	11	18	TYM
TS0NC3	11	19	TYM	TS0NC4	11	20	TYM	TS0NI	11	35	TYM	TS0NT1	11	21	TYM
TSS0FT	9	15	TYM	TSTD2	7	23	W	TSTD3	7	26	W	TSTRDY	7	31	W
TT	2	21	TYM	TT01	22	10	B	TT0M	9	15	B	TTIME	25	30	PAC
TTN0	6	10	S	TTRAP	7	28	W	TTYASG	2	18	TYM	TTYS	2	7	M
TTYSKS	2	8	M	TU36	4	3	TYM	TUCHR	3	1	TYM	TUCHRF	3	2	TYM
TUDMS	3	22	TYM	TUDMS1	3	28	TYM	TUE	4	9	TYM	TUE8	4	18	TYM
TUE9	4	17	TYM	TUECH0	3	3	TYM	TUGC	3	5	TYM	TUGC1	3	7	TYM
TUGC2	3	8	TYM	TUGC3	3	18	TYM	TUGC4	3	20	TYM	TUIGNR	4	7	TYM
TULC	4	5	TYM	TUN	3	31	TYM	TUN1	3	32	TYM	TUN2	3	33	TYM

TUN8	3	35	TYM	TUN8A	4	1	TYM	TURL	8	38	I	TUZN	11	1	TS84A
TVDMS	5	16	TYM	TWA	2	24	NET	TWA0	4	5	NET	TWC	2	3	NET
TWC0	4	6	NET	TWC1	2	5	NET	TWC2	2	18	NET	TWC3	2	14	NET
TWC5	2	26	NET	TWCNT	8	21	I	TWI	4	28	W	TWI1	5	3	W
TWI10	5	9	W	TWI2	5	7	W	TWI3	5	6	W	TWI6	5	4	W
TWI7	5	8	W	TWMASK	2	27	NET	TWMSG	9	20	TS84A	TW0	2	8	S
TWR4	4	22	W	TWR5	4	23	W	TWRT	4	5	W	TWX	2	25	NET
TX00	22	5	B	TX01	22	6	B	TX02	22	7	B	TX03	22	8	B
TX05	21	39	B	TY1	7	22	TYM	TY10	10	2	TYM	TY11	10	4	TYM
TY12	10	12	TYM	TY18	9	1	TYM	TY1B	8	32	TYM	TY1E0	7	27	TYM
TY1E0F	7	33	TYM	TY1E1	7	35	TYM	TY1E1A	8	1	TYM	TY1E2	8	4	TYM
TY1E3	8	6	TYM	TY1E4	8	15	TYM	TY1E5	8	18	TYM	TY1E5A	8	22	TYM
TY1E6	8	24	TYM	TY1E7	8	28	TYM	TY1N	9	4	TYM	TY1N1	9	8	TYM
TY1N2	9	12	TYM	TY1N9	9	13	TYM	TY36	9	32	TYM	TY5	9	35	TYM
TY7	9	39	TYM	TYEL	19	38	TYM	TYM2	2	21	RUP	TYMPAG	2	31	M
TYMSET	10	29	I	TZAP	19	19	TYM	TZAP1	19	12	TYM	TZAP2	19	14	TYM
TZAPR	19	7	TYM	U=USER	3	14	PMT	UA	3	23	TS84A	UAST	6	38	I
UAST2	7	1	I	UB	3	24	TS84A	UBA	2	6	TS84A	UBB	2	7	TS84A
UBE	2	11	TS84A	UBRL1	2	9	TS84A	UBRL2	2	10	TS84A	UBRSET	4	24	B
UBRSRT	3	21	TS84A	UBX	2	8	TS84A	UCIN	2	12	TS84A	UC0UT	2	13	TS84A
UE	3	28	TS84A	UEXFLG	2	15	TS84A	UEXL6	2	27	TS84A	UGCI	1	12	RUP
UMSZ	6	9	M	UMTF	27	29	PAC	UMTJ0B	27	38	PAC	UNIT	1	6	I
UNM	4	34	TS84A	UNM2	4	35	TS84A	UNMP	4	36	TS84A	UNMP1	4	37	TS84A
UNMPCF	4	38	TS84A	UN0	3	31	TS84A	UN01	7	31	TS84A	UN0SV	25	32	PAC
UNPTR	4	24	TS84A	UNPTR1	4	25	TS84A	UPFL	2	28	M	UPL	2	5	TS84A
UPRL	24	9	PAC	UPRL44	15	34	B	UPRRL1	2	21	TS84A	UPRRL2	2	22	TS84A
UPSHUT	8	15	S	UREAL	4	3	TS84A	UREAL	2	25	TS84A	USMT	1	32	PMT
USN0	7	2	S	USTKPS	16	33	B	UTIME	7	37	S	UTTY	5	4	S
UUN0	2	14	TS84A	UX	3	25	TS84A	UZ0NE	11	29	TS84A	VERS	5	34	S
W	5	37	R	WB	9	38	I	WBIU	8	32	I	WCD	4	20	RUP
WCD1	4	22	RUP	WCDP	4	21	RUP	WCH	6	33	RUP	WCH1	10	23	RUP
WCH2	10	24	RUP	WCH3	10	25	RUP	WCH5	7	11	RUP	WCH77	6	35	RUP
WCHP	6	34	RUP	WCI	4	8	RUP	WCI1	4	10	RUP	WCIP	4	9	RUP
WCL2	8	16	I	WCL3	8	6	I	WCL4	8	7	I	WCL5	8	14	I
WCL6	8	4	I	WCL7	8	9	I	WCLEAN	7	35	I	WD31	6	39	SET
WD33	7	1	SET	WDCP	7	13	SET	WDIDM	7	16	SET	WDIMT	6	36	SET
WDI0P	7	14	SET	WDPIT	6	35	SET	WDPWFI	7	12	SET	WDPWNI	7	11	SET
WDRMT	6	37	SET	WDTB	6	33	SET	WDTE	7	28	SET	WDUMT	6	38	SET
WERIS	6	4	S	WETTB	8	1	S	WETTB	33	17	B	WEX	4	4	I

WEX1	4	7	I	WEX2	4	8	I	WEX3	4	9	I	WH0Q	13	7	PAC
WH0G2	13	8	PAC	WHT	7	4	I	WIE	3	30	I	WINB	9	14	W
WI0	3	29	I	WJ0B	9	2	I	WPAGE	9	19	I	WPMT	8	27	I
WPTR	8	33	I	WR1	1	30	TS84A	WR10	1	39	TS84A	WR11	2	1	TS84A
WR12	2	2	TS84A	WR13	2	3	TS84A	WR2	1	31	TS84A	WR3	1	32	TS84A
WR31	1	33	TS84A	WR4	1	13	TS84A	WR5	1	34	TS84A	WR6	1	35	TS84A
WR7	1	36	TS84A	WR8	1	37	TS84A	WR9	1	38	TS84A	WRDB	9	16	W
WREL	10	2	I	WRET	9	27	I	WRJ0B	27	37	PAC	WRSB	9	12	W
WRW	3	34	I	WRW1	3	39	I	WRW2	4	3	I	WS1	9	25	I
WSA	9	15	I	WSAVE	9	22	I	WSB	9	16	I	WSRRL3	9	18	I
WSW	9	20	I	WSX	9	17	I	X	1	6	TS84A	X0	4	29	S
X1	4	30	S	X2	4	31	S	X3	4	32	S	X4	4	33	S
X5	4	34	S	X6	4	35	S	X7	4	36	S	X=EXEC	3	16	PMT
XARM0T	19	21	I	XBERR	8	10	S	XBP	1	9	I	XDIR	5	36	I
XEIR	5	39	I	XMA41	21	11	B	XMA43	21	12	B	XPB	5	5	S
XP0P	4	37	PAC	XP0PX	4	34	PAC	XSMTEM	2	3	PMT	XT1	11	4	TS84A
XT10	11	13	TS84A	XT11	11	14	TS84A	XT12	11	15	TS84A	XT13	11	16	TS84A
XT14	11	17	TS84A	XT2	11	5	TS84A	XT3	11	6	TS84A	XT4	11	7	TS84A
XT5	11	8	TS84A	XT6	11	9	TS84A	XT7	11	10	TS84A	XT8	11	11	TS84A
XT9	11	12	TS84A	XTE	11	18	TS84A	XWR1	3	6	TS84A	XWR2	3	7	TS84A
XWR3	3	8	TS84A	XWR31	3	9	TS84A	XWR4	2	34	TS84A	XWR41	2	35	TS84A
XWR42	2	36	TS84A	XWR43	2	37	TS84A	XWR44	2	38	TS84A	XWR45	2	39	TS84A
XWR46	3	1	TS84A	XWR47	3	2	TS84A	XWR4E	3	3	TS84A	XWR4E8	3	4	TS84A
XX	4	37	S	XX0	3	9	M	XX13	3	11	M	XX20	3	12	M
XX25	3	13	M	XX6	3	10	M	Y=SYST	3	17	PMT	YEAR	9	5	S
YPIM	14	25	PAC	YPL	14	19	PAC	YPPTR	14	22	PAC	YPTAB	14	24	PAC
YPTEST	14	23	PAC	YRL1	14	20	PAC	YRL2	14	21	PAC	ZDAY	8	21	TS84A
ZE	35	27	RUP	ZE	17	37	RUP	ZER0	2	6	S	ZH0UR	8	24	TS84A
ZM	35	28	RUP	ZM	17	38	RUP	ZMIN	8	27	TS84A	ZM0NTH	8	19	TS84A
ZMS	8	32	TS84A	ZSEC	8	29	TS84A	ZYEAR	8	16	TS84A	]=PR0G	3	9	PMT
√=ACTI	3	13	PMT	√=ACTI	2	35	PMT	√=N0N=	3	10	PMT	√=N0N=	3	4	PMT
√=0VER	3	15	PMT	√=SUBS	3	12	PMT	√=TERM	3	11	PMT				

```
M       IDENT                                                   6/21/72  14:51     PAGE   1
        NOLIST EXT,STAT

*       SYSTEM FLAGS, OPDS, PARAMETERS, AND MACROS

* ASSEMBLY FLAGS
ALOG    EQU 1;* ASSEMBLE LOG ROUTINE AND CALLS.
137BRS  EQU -1;* SUPRESS SIDEWAYS FORKS.
ST      EQU     0                   STATISTICS, 0=OFF, 1=ON
EXEC    EQU     -1                  TS BLOCK ASSEMBLY SWITCH
EXPMT   EQU  1   EXPANDED PMT'S.
PH2     EQU  1   1= PHASE 2 MUX
RE      EQU PH2
IIH     EQU PH2 INPUT OVERRUN INHIBIT (ALTMODES TOO).
DDIFP   EQU     1                   DDI CHANGES FOR FLOATING POINT
FPH     EQU     -1                SIMULATE FLOATING POINT HARDWARE WITH -1
DISC    EQU 2  DISC =1 FOR NORMAL DP, 2 FOR EXTRA DP DISCS, 3 FOR 2314
RP      EQU     -1                  -1 = CPU, 1 = TRU
LISTAB  EQU     0                   ABSOLUTE LISTING PARAMETER
PTICKB  EQU     0                   PTICK BUG FLAG

* OPDS
SBRS    OPD     17385               SYSTEM MODE BRS
TSN     OPD     002220008,1         GO FROM NORMAL TO MONITOR MODE
CKN     OPD     002201008,1         TURN ON THE CLOCK
CKF     OPD     002202008,1         TURN OFF THE CLOCK
LRR1    OPD     00220400B,1         LOAD RELABELLING REGISTER 1
LRR2    OPD     00221000B,1         LOAD RELABELLING REGISTER 2
LRR3    OPD     00221400B,1         LOAD RELABELLING REGISTER 3

*       MACROS
LISTM   MACRO; IF LISTAB; NOLIST ALL
        LIST    LCT,SRC,COM,EXP,EXT,GO
        ELSE; NOLIST EXT,STAT; ENDF; ENDM
DEFL    LMACRO D;D(0) EQU LOC;LOC EQU LOC+D(1); ENDM
DIR     MACRO; BRM XDIR; ENDM   DEBUGGING AID

EIR     MACRO; BRM XEIR; ENDM
```

```
* BREAKPOINT TEST
BPT     0PD     04020000B,1
*
* I/0 DEVICE 0PD'S
TTYS    MACR0; DATA 20277777B; ENDM
TTYSKS  EQU     240770008
E0D     0PD 655
I0SDW   EQU     2142008
I0RDW   EQU     214000B
* PARAMETERS
BE      EQU     123             LAST BERKELEY BRS.
NP0P    EQU     44B             NUMBER 0F SYSP0PS IN USE.
RDP     EQU     10B             REAL DISC PAGE
RWP     EQU     11B             REAL W PAGE
RSR     EQU     0607B           SET R0UTINE RELABELLING
RTR     EQU     12B             TYMNET BUFFERS
RSP     EQU     7               REAL STATISTICS PAGE.
* W BUFFER DEVICE PARAMETERS
NTAPE   EQU 2
NTRTRY  EQU     10              NUMBER 0F REREADS
NTWTRY  EQU     3               NUMBER 0F REWRITES
*       TYMNET SYMB0LS
*       M0ST 0F THE TYMNET BUFFERS ARE IN PAGE 12B
ALARM   EQU 210B                WHEN N0N ZER0 THE BASE S0UNDS AN ALARM
SUPAGE  EQU 211B                P0INTS T0 SUPERVIS0RS PAGE 0
UPFL    EQU 212B AND 213   SEE PACG01. TYMNET CL0BBERS UPFL PERI0DICALLY.
* BASE PUTS BRM 214B T0 INDUCE M0NIT0R CRASH.
* 216B GETS PC 0F BASE UP0N INDUCED CRASH.
TYMPAGE EQU 12B
NTBI    EQU 200B                NUMBER 0F CELLS F0R TYMNET BFR INPUT
NTB0    EQU 400B
TBI     EQU 34000B              TYMNET BUFFER INPUT   (RING BUFFER)
ETBI    EQU TBI+NTBI
TB0     EQU ETBI                0UPUT RING BUFFER
ETB0    EQU TB0+NTB0
NP0RT   EQU 64
TRATE   EQU ETB0
```

```
TF1     EQU TRATE+NPORT
TF3     EQU TF1+NPORT
TF4     EQU TF3+NPORT
T128    EQU TF4+NPORT
TP      EQU T128+64+1
TC      EQU TP+NPORT+2
*       PORT  FLAG DEFINITIONS     ****************************************

XX0     EQU 1;XX1 EQU 2;XX2 EQU 4;XX3 EQU 10B;XX4 EQU 20B;XX5 EQU 40B
XX6     EQU 1B2;XX7 EQU 2B2;XX10 EQU 4B2;XX11 EQU 1B3;XX12 EQU 2B3
XX13    EQU 4B3;XX14 EQU 1B4;XX15 EQU 2B4;XX16 EQU 4B4;XX17 EQU 1B5
XX20    EQU 2B5;XX21 EQU 4B5;XX22 EQU 1B6;XX23 EQU 2B6;XX24 EQU 4B6
XX25    EQU 1B7;XX26 EQU 2B7;XX27 EQU 4B7
        FRGT    XX0,XX1,XX2,XX3,XX4,XX5,XX6,XX7
        FRGT    XX10,XX11,XX12,XX13,XX14,XX15,XX16,XX17
        FRGT    XX20,XX21,XX22,XX23,XX24,XX25,XX26,XX27

FLAG    MACRO D GENERATE 2 FLAG SYMBOLS.ONE FOR MASKING,ONE FOR SHIFTING.
*     EXAMPLE    BETA IS A 6 BIT FIELD IN BITS 7-12 WHICH NEEDS TO BE
*  SHIFTED 12 PLACES TO BE NORMALIZED TO THE LOW PART OF SOME REGISTOR.
*             FLAG BETA,77,11 GENERATES (BETAY EQU 11B) FOR SHIFTING
*                              AND (BETAZ EQU 77B3) FOR MASKING
D(1).AZ EQU D(3).AB             SHIFT OPERAND
D(1).AY EQU XX.AD(3)*D(2).AB     MASK OR BIT
        FRGT D(1).AZ,D(1).AY
        ENDM


*       FLAGS IN TF1

        FLAG    CON,1,0         CONDEMNED CIRCUIT
        FLAG    AUX,1,1         AUXILIARY CIRCUIT
        FLAG    DI,1,2          DISMISSED ON INPUT
        FLAG    DE,1,3          DEFERRED ECHO IN EFFECT
        FLAG    GREEN,1,4       GREEN BALL TO BE RETURNED
        FLAG    7I,1,5          CHARACTER 7 IN USERS INPUT BFR
        FLAG    IIH,1,6         INPUT INHIBITED ON AUXILIARY CIRCUIT
        FLAG    135,1,7         135 SWITCH COCKED TO OUTPUT BLANKS
        FLAG    531,1,10        SUPRESS 135 SWITCH
```

```
        FLAG    152,1,11        OUTPUT 212,215,377 FOR TCO =152 AND
*                               OUTPUT 215,212 FOR TCO =155
        FLAG    102,1,12        OUTPUT 212 OR 342 FOR TCO =102 AND
*                               OUTPUT 214 OR 345 FOR TCO =105
        FLAG    GOB,1,13        CHR GOBBLER RECEIVED
        FLAG    MAR,1,14        AT LEFT MARGIN
        FLAG    MARS,1,15       SUPRESS LEFT MARGIN FLAG
        FLAG    XOFF,1,16       TERMINAL NOT INPUTTING IF XOFFY=1
        FLAG    LCO,1,17        LOWER CASE OUTPUT
        FLAG    8O,1,23         8 LEVEL OUTPUT
        FLAG    LCI,1,25        MAKE LOWER CASE INTO UPPER CASE
        FLAG    X,1,26          TERMINAL CAN UNDERSTAND X-ON, X-OFF CHRS
        FLAG    8I,1,27         8 LEVEL INPUT


*       FLAGS IN TF2
        FLAG    JB,77,0         JOB
        FLAG    RC,3,6          RATE CONTROLLER BITS.00 NO ACTIVATION
*                       DELAY.01 NO DELAY.10 DELAY 2 SEC.11 DELAY 3 SEC.
        FLAG    BRK,7,11        3 BIT FIELD. ONE BIT ON FOR BREAK TABLE.
        FLAG    B113,1,14       IGNORE SOFT AND HARD ESCAPES, BRS 113,114.
        FLAG    SOFT,1,15       SOFT ESCAPE ON PHANTOM USER
        FLAG    HARD,1,16       HARD ESCAPE ON PU
        FLAG    B161,1,17       BRS 161 HAS DISARMED HARD + SOFT ESCAPES
        FLAG    B168,1,20       BRS 168 HAS DISARMED HARD ESCAPES
        FLAG    DIAL,1,23       ALLOW REMOTE DIALUP ON THIS PORT
        FLAG    YEL,1,24        SEND OUT A YELLOW BALL AT DISMISS TIME IF SET.
        FLAG    5O,1,25         TYPE 5 OUT. TY5 CODE ACTIVATED BY CAC7.
        FLAG    ORA,1,26        WAITING FOR ORANGE BALL. SEE CAC9.
        FLAG    OIH,1,27        OUTPUT INHIBITED. USED BY SCHEDULAR.


*       FLAGS IN TF3

*       TF3 DESCRIBES THE BITS THAT DESCRIBE THE TERMINAL
* CHARACTERISTICS. THEY ARE PASSED FROM THE 940 TO THE REMOTE AS
* CHARACTER PAIRS. THE FIRST CHARACTER OF EACH PAIR IS A 1. THE
* SECOND CHARACTER IS A CHARACTER WITH TWO 4 BIT FIELDS. THE
* FIRST FIELD IS A TYPE FIELD AND THE SECOND FIELD IS DATA BITS.
```

```
* BITS ARE NUMBERED FROM LEFT TO RIGHT.
* IF THE DATA BIT IS ON, THE CONDITION IT DESCRIBES IS ACTIVE.
* TYPE 0: 0 ECHO ON,1 ECHO CONTROL I,2 ECHO CR,RB TO LINE FEED,
*  3 ECHO LF TO CARRIAGE RETURN.
* TYPE 1: 0 CR DELAY,1-3 INPUT BAUD RATE
* TYPE 2: 0-2 OUTPUT BAUD RATE,3 REQUIRES PARITY.
* TYPE 3: 0-2 PARAMETER A,3 HALF DUX
* TYPE 4: 0-2 PARAMETER B
* TYPE 5: 0-3 PARAMETER C
*    PARAMETERS A, B, AND C ARE USED TO COMPUTE CARRIAGE RETURNS
* AS FOLLOWS: F(N)=MIN[N/(2^A)+B,C]
*    PARAMETERS A, B, AND C ARE USED TO COMPUTE LINE FEED DELAYS
* AS FOLLOWS: IF N=1,F(N)=A
* OTHERWISE, IF C>N, F(N)=C-N+B.  OTHERWISE F(N)=B

        FLAG    ECHO,1,27       PORT CAN ECHO
        FLAG    TAB,1,26        ECHO CONTROL I
        FLAG    ELF,1,25        ECHO LF AS CR,RB
        FLAG    HDUX,1,10       ON IF TERMINAL IS HALF DUPLEX
        FLAG    ECR,1,24        ECHO CR AS LF

*       FLAGS IN TF4

        FLAG    TERMC,777,0     8 LEVEL INPUT TERMINATING CHR
        FLAG    5A,7,15         A REGISTER FOR USER FROM INPUT TYPE 5
        FLAG    135C,377,20     135 BLANK COUNTER

SET     MACRO D   SET ONE OF THE PORT FLAGS
        LDA D(1); SKA D(2),2; BRU *+2; ADM D(2),2; ENDM

RST     MACRO D   RESET ONE OF THE PORT FLAGS
        LDA D(1); COPY AB,N; SKB D(2),2; ADM D(2),2; ENDM

*       END OF PORT FLAG DEFINITIONS ************************************

ETBUF   EQU 377768
```

```
* PAC TABLE PARAMETERS
NSQU   EQU 4    NUMBER OF CLOCK CYCLES IN SHORT QUANTUM.
NLQU   EQU      125              FULL QUANTUM SIZE. 2 SEC.
NPAC   EQU      130              NUMBER OF PACT SLOTS
NPPAR  EQU 8    LENGTH OF PACT ENTRY
* JOB AND MEMORY PARAMETERS
NJOB1  EQU      38                NUMBER OF JOBS WITHOUT P.U.
NJOB   EQU      NJOB1+1          NUMBER OF JOBS
UMSZ   EQU      15               INITIAL MACHINE SIZE
NMEM   EQU 32   NUMBER OF PAGES
NSMEM  EQU 7    NUMBER OF PAGES USED BY SYSTEM
NSSMT  EQU      15               NO. OF SYSTEM PAGES IN SMT
NUSMT  EQU 6    NUMBER OF NONPROPRIETARY SMT PAGES+1.
       IF EXPMT
NCMEM  EQU 43B      COMMON PART OF USER MACHINE.
NSMT   EQU 43B      SIZE OF SMT.
NXSMT  EQU 116B     NO. EXPANDED SMT'S.
FESMT  EQU 100B     FIRST EXPANDED SMT NUMBER.
LESMT  EQU NXSMT+FESMT    LAST EXPANDED SMT NUMBER.
       ELSE
NCMEM  EQU      60B              COMMON PART OF USER MACHINE
NSMT   EQU      60B              SIZE OF SMT
       ENDF
NUMEM  EQU 100B-NCMEM           NUMBER OF PRIVATE USER PAGES
NPUC   EQU NJOB   NUMBER OF PUCT ENTRIES
* DISC PARAMETERS
NFILE  EQU 35+5*DISC  NUMBER OF FILES THAT CAN BE OPEN AT ONCE.
NBUF   EQU 4    NUMBER OF FILES THAT A USER CAN HAVE OPENED AT ONCE.
MBUFX  EQU 0    MBUFX DEFINES THE FILE IN USE BIT MAP. SEE TSONI..1I
       RPT NBUF
MBUFX  EQU MBUFX/2+2B7; ENDR
NDRQ   EQU      70               NO. OF JOBS IN DISC QUEUE.
       IF DISC=1
*      20000B SECTORS PER DISC=400B DATA BLOCKS. 3600B ARE MAPPED.
*      8 DISCS ARE MAPPED. 3600B*8 BLOCKS ARE MAPPED.3600B*8=36000B
TABLEC EQU      36000B           NUMBER OF BITS IN BIT MAP.
       ELSE
TABLEC EQU 8*20*200  8 RECORDS,20 HEADS,200 CYLINDERS TO MAP A 2314 PACK
       ENDF
```

```
MAXFIL EQU 3B7 MAXIMUM FILE SIZE IN CHARACTERS
NLVL   EQU     5
MLVL   EQU     NLVL-1

* BUFFER MAP
CBF0   EQU     0
CBF1   EQU     1                20 POINTERS AND FILE PARAMETERS.
CBF2   EQU     41B              40 POINTERS.
CBF3   EQU     101B             40 POINTERS.
CBF4   EQU     141B             DATA AREA.

CSIZE  EQU     CBF1+37B         NO. OF 256 WORD BLOCKS. CHARGE SIZE.
CEOF   EQU     CSIZE-1          END OF FILE.
CBOF   EQU     CEOF-1           BEGINNING OF FILE.
CPTOP  EQU     CBOF-1           HIGHEST LOC. EVER WRITTEN. FOR INPUT.
CQ     EQU     CPTOP-1          PHYSICAL FILE SIZE QUANTUM
CCKWRD EQU CQ-1       CHECKWORD PUT INTO HIGHEST XBLOCK TO REDUCE ERRORS

CP     EQU     CBF4+400B        CURSOR POSITION.
*      BIT 22,23 OF CP TELL THE CHAR. POSITION IN A BUFFER WORD.
*      BITS 14-23   CHAR. POSITION FOR DATA BLOCK (BUFFER LEVEL=MLVL).
*      BITS  9-13   POINTS TO 1 OF 40B DISC ADDRESSES IN BFR MLVL-1
*      BITS  4-8    POINTS TO 1 OF 40B DISC ADDRESSES IN BFR MLVL-2
*      BITS  0-3    POINTS TO 1 OF 20B DISC ADDRESSES IN BFR MLVL-3
*      BIT 0   ALWAYS ZERO

CFLG   EQU     CP+1             POINTER TO GFLG.
CSW    EQU     CFLG+1           =1 FOR OUTPUT. 0 FOR INPUT.
CSW1   EQU     CSW+1            =1 NORMALLY,0 FOR BRS 128, =1 FOR BRS 66
CEOF1  EQU     CSW1+1            LOCAL EOF FOR WIO AND CIO. 0=EMPTY FILE.
CC     EQU     CEOF1+1          CURSOR CHANGED. 1 WORD PER BUFFER.
COV    EQU     CC+NLVL          COUNTER FOR TOO MANY WIO'S, ETC.
* END OF BUFFER MAP
*
NDBW   EQU     COV+1;NDBW EQU NDBW-CBF0     NDBW=BUFFER SIZE.
NDBS   EQU     NDBW*NBUF
* TS BLOCK MAP
```

```
        IF      DDIFP
TSMAPS  EQU     65
        ELSE
TSMAPS  EQU 55  SPACE FOR TS MAP. CAN BE CHANGED.
        ENDF
LOC     EQU 37777B-NDBS-TSMAPS;* START ASSIGNMENT
DBTOP   DEFL 0  1ST WORD AFTER PRSYMS
FBADR   DEFL NDBS       FIRST BUFFER ADDRESS
FBWRD   DEFL 1      BUFFER AVAILABILTY BIT WORD
FILE    DEFL 1
GFILE   DEFL 1     GLOBAL FILE NO.
BUFF    DEFL 1          BUFFER ADDRESS.
LVL     DEFL 1    LVL TELLS WHICH TS BFR IN A FILE IS PERTINENT
CBIP    DEFL 1           POINTS TO DISC ADRS. SEE :SETBIP:
CP1     DEFL 1           WORKING STORAGE.
CP2     DEFL 1           WORKING STORAGE.
CRET0   DEFL 1  START UP LOCATION (LOC. 0) FOR DISMISSED FORK.
CRET1   DEFL 1         REENTRANT MARK LOCATION FOR :CDMS:
CRET2   DEFL 1
CRET3   DEFL 1
CRET4   DEFL 1
CT1     DEFL 1           REENTRANT TEMPORARY STORAGE
CT2     DEFL 1
CT3     DEFL 1
CDSA    DEFL 1        A REGISTOR WHEN :CDMS: WAS CALLED
CDSB    DEFL 1
CDSX    DEFL 1
DGET    DEFL 1        WHERE TO START LOOKING IN BIT MAP FOR BITS
FSIZE   DEFL 1        DEFAULT FILE SIZE QUANTUM. SEE TSONI, BRS92.
LQ      DEFL 1        LONG QUANTUM
PA      DEFL 8
PB      DEFL 8
PX      DEFL 8
        IF      DDIFP
FAC1    DEFL 1
FAC2    DEFL 1
        ENDF
PPB     DEFL 1     POINTER TO PB CHAIN
NFORK   DEFL 1           NUMBER OF FORKS COUNTER
```

```
DSKMET DEFL 1           DISK METER.                                    M          PAGE   9
       IF 137BRS
PSSTK  DEFL 36
PSDEX  DEFL 36
MPSDEX EQU     30                MAX OF FIVE NESTED BRS 137'S
       ENDF
       IF LOC>40000B; DISASTER, TS PAGE FULL; ENDF
*      CHANGE :TSMAPS: AND THIS COMMENT IF MORE THAN 100 LOCATIONS ARE
*      ALLOCATED BETWEEN :TSMAPS: AND HERE
 FRGT TSMAPS
*      TTY BUFFERS OCCUPY REAL PAGE 12B AND 13B

* MACROS
DEFINE MACRO D  (DEFINE CAUSES MDBG SYMBOLS TO BE PUT INTO BINARY FILE)
D1     NARG D;D2 EQU 0
       RPT D1;D2 EQU D2+1;D(D2) EQU D(D2); ENDR; FRGT D1,D2; ENDM

TRAP   MACRO D; BRM TRAP; ENDM
LOGG   MACRO D; IF ALOG; BRM LOG; D(1) D(2); ENDF; ENDM
ARMI   MACRO D; AIR; POT D(1); ENDM
ENTRY  MACRO L;ENTCNT NARG; RPT ENTCNT;L(ENTCNT) EXT
ENTCNT EQU ENTCNT-1; ENDR; ENDM
SETINT MACRO A; LDA =A(1); STA BLK31; ENDM
TDT    MACRO L;$L(1)•AW EQU *; RPT NTAPE; L(2) L(3)•AB+**-L(1)•AW; ENDR; ENDM
RMFF   MACRO; ENDM
SMFF   MACRO; ENDM
TLABEL MACRO   D                 GEN. ROUTINE TO LABEL IN TTY BUFFERS
D(1)•AL1 ZRO
D(1)•AL ZRO; XMA RRL3; LRR3; POT RRL3; STA D(1)•AL1; BRR D(1)•AL; ENDM
TRP    MACRO L;ENTCNT NARG; RPT ENTCNT;L(ENTCNT) EQU TRAP
 FRGT L(ENTCNT);ENTCNT EQU ENTCNT-1; ENDR; ENDM
LBL    MACRO D;1LBL EQU D(2); RPT D(2); LDA D(1)+1LBL-1; LRSH 6
1LBL   EQU 1LBL-1; ENDR; ENDM
*
       IF      DDIFP
       IF      FPH
*
*      FLOATING POINT HARDWARE MACROS
*
```

```
*       INITIATE OUTPUT OF MOST SIGNIFICANT HALF
IMSHO   MACRO   D
        EOM*    30721B,2
N:      NARG
        IF      N:=1
        POT     D(1)
        ELSE
        POT     D(1),D(2)
        ENDF
        ENDM
*
*       INITIATE INPUT OF MOST SIGNIFICANT HALF
IMSHI   MACRO   D
        EOM*    30731B,2
N:      NARG
        IF      N:=1
        PIN     D(1)
        ELSE
        PIN     D(1),D(2)
        ENDF
        ENDM
*
*       INITIATE FLOATING LOAD
IFLD    MACRO   D
        EOM*    30720B,2
N:      NARG
        IF      N:=1
        POT     D(1)
        ELSE
        POT     D(1),D(2)
        ENDF
        ENDM
*
*       INITIATE FLOATING ADD
IFAD    MACRO   D
        EOM*    30722B,2
N:      NARG
        IF      N:=1
        POT     D(1)
```

```
        ELSE
        P0T     D(1),D(2)
        ENDF
        ENDM
*
*       INITIATE FL0ATING SUBTRACT
IFSB    MACR0   D
        E0M*    30723B,2
N:      NARG
        IF      N:=1
        P0T     D(1)
        ELSE
        P0T     D(1),D(2)
        ENDF
        ENDM
*
*       INTIATE FL0ATING INVERSE SUBTRACT
IFSBI   MACR0   D
        E0M*    30724B,2
N:      NARG
        IF      N:=1
        P0T     D(1)
        ELSE
        P0T     D(1),D(2)
        ENDF
        ENDM
*
*       INITIATE FL0ATING MULTIPLY
IFMP    MACR0   D
        E0M*    30725B,2
N:      NARG
        IF      N:=1
        P0T     D(1)
        ELSE
        P0T     D(1),D(2)
        ENDF
        ENDM
*
*       INITIATE FL0ATING DIVIDE
```

```
IFDV    MACRO   D
        EOM*    30726B,2
N:      NARG
        IF      N:=1
        POT     D(1)
        ELSE
        POT     D(1),D(2)
        ENDF
        ENDM
*
*       INITIATE FLOATING INVERSE DIVIDE
IFDVI   MACRO   D
        EOM*    30727B,2
N:      NARG
        IF      N:=1
        POT     D(1)
        ELSE
        POT     D(1),D(2)
        ENDF
        ENDM
*
*       INITIATE FLOATING STORE
IFST    MACRO   D
        EOM*    30730B,2
N:      NARG
        IF      N:=1
        PIN     D(1)
        ELSE
        PIN     D(1),D(2)
        ENDF
        ENDM
*
*       RESET FLOATING OVERFLOW
RFPOV   MACRO   D
        SKS*    30720B,2
        BRU     *+1
        ENDM
*
*       SKIP IF FLOATING AC NEGATIVE
```

```
SHFAN   MACR0   D
        IFST    T
        SkS*    30721B,2
        ENDM
*       FL0ATING P0INT MACR0S
*
*       L0AD FL0ATING ACCUMULAT0R
HFLD    MACR0   D
        IFLD    D(2)
        IMSH0   D(1)
        ENDM
*
*       ST0RE FL0ATING ACCUMULAT0R
HFST    MACR0   D
        IFST    D(2)
        IMSHI   D(1)
        ENDM
*
*       FL0ATING ADD
HFAD    MACR0   D
        IFAD    D(2)
        IMSH0   D(1)
        ENDM
*
*       FL0ATING SUBTRACT
HFSB    MACR0   D
        IFSB    D(2)
        IMSH0   D(1)
        ENDM
*
*       FL0ATING SUBTRACT INVERSE
HFSBI   MACR0   D
        IFSBI   D(2)
        IMSH0   D(1)
        ENDM
*
*       FL0ATING MULTIPLY
HFMP    MACR0   D
        IFMP    D(2)
```

```
        IMSH8   D(1)
        ENDM
*
*       FLOATING DIVIDE
HFDV    MACR8   D
        IFDV    D(2)
        IMSH8   D(1)
        ENDM
*
*       FLOATING DIVIDE INVERSE
HFDVI   MACR8   D
        IFDVI   D(2)
        IMSH8   D(1)
        ENDM
        ENDF
        ENDF
*
*
*
*
* EXEC ENTRY POINTS

EXECI   EQU     10000B
EXECP   EQU     10001B
OFFINT  EQU     10002B
CKCPU   EQU     10003B  CPU TIME OVERRUN.
CKCPU1  EQU     10004B  CPU TIME OVERRUN IF IN EXEC.
*
*
*


FORGET  MACR8
        FRGT CRXF,OPB,ORB,RTR,SOV
        FRGT MBUFX,PNXF,LPXF
        FRGT RTCNT,PNCNT,TCNT,PLCNT,RTWT,PNWT,TXWT
        FRGT NTAPE,NLINK,NBUFX,NBUF
        FRGT NFILE,UMSZ,TTYEWM,NPAC,NPPAR,NJOB,NJOB1,NCMEM
```

```
FRGT EXEC,FCB,C181,940M,RELCHN,BE,RDP,RWP,RSR,RTR,N0I,NLQU,LQ
FRGT NF0RK,CAW,DAW,CADSC,DADSC,CASET,CAEXEC,DASET,DAEXEC
FRGT DEW,1LBL,ECHV0,TABLEC,M37C,N0EK0,DBB,EXECI,FSIZE,DGET
ENDM
FREEZE
END
```

```
        LISTM


$UNIT   0;*     ACTIVE W BUFFER DEVICE NUMBER 0R -1
$ACTR   0;*     ACTIVATION C0UNTER

$XBP    BSS NFILE  TABLE F0R INDEX BL0CK P0INTERS. INITIALIZED T0 -1.
$EXBP   EQU *

$GFLG   BSS NFILE               FILE FLAGS INDEXED BY GFILE.
*       GFLAG F0RMAT.GFLAG IS  INDEXED BY A GL0BAL FILE NUMBER.
*       GFLAG C0NTAINS J0B AND L0CAL FILE NUMBERS. THE L0CAL FILE
*       BUFFER AREA C0NTAINS THE CELL CFLG,2 WHICH P0INTS T0 THAT
*       FILE'S GFLAG W0RD. HENCE, IT IS P0SSIBLE T0 GET GL0BAL INF0 FR0M
*       L0CAL INF0 0R T0 GET L0CAL INF0 FR0M GL0BAL INF0.
*       BITS 0-6. INITIALIZED T0 -1. INCREMENTED F0R EACH DISC J0B
*       THAT IS STARTED. DECREMENTED BY INTERRUPT R0UTINE. GFLG (0R
*       INDIRECT CFLG,2) IS P0SITIVE WHEN FILE IS BUSY D0ING I/0.
*       BITS 7,8 - L0CAL FILE NUMBER-3.    BIT 9 - ERR0R DETECTED IN IDM
*       BITS 10-15 - J0B NUMBER
*       BITS 16-23 - FILE PRIVELEGE FLAGS
$EGFLG EQU *

$BUF    EQU *   TABLE 0F L0CAL BUFF ADDRESSES IN TS BL0CK
$BUF3   EQU BUF-3
SYMB0L  EQU FBADR
        RPT NBUF; DATA SYMB0L;SYMB0L EQU SYMB0L+NDBW; ENDR; FRGT SYMB0L
$BUFN   EQU     BUF+NBUF

*       CURS0R P0PS  *********

BI0     P0PD    57685
$BI0X   BRM* FSAVE1; BRU BI0X1

PCE     P0PD    53385           P0SITI0N CURS0R AND ERASE
$PCEX   BRM* FSAVE1; BRU PCEX1
```

```
SCP     P0PD    53285           SET CURS0R P0SITI0N
$SCPX   SKN 0; BRM M0NCR (KEEP M0NITER FR0M EXECUTING MARK L0CATI0NS);
        BRM* FSAVE1; BRU SCPX1

RCP     P0PD    53185           READ CURS0R P0SITI0N
$RCPX   BRM FSAVE; BRU RCPX1

SSP     P0PD    52685           SET PHYSICAL SIZE LIMIT F0R A FILE
$SSPX   BRM* FSAVE1; BRU SSPX1

RSP     P0PD    52585           READ PHYSICAL SIZE, SIZE LIMIT, AND CPT0P
$RSPX   BRM FSAVE; BRU RSPX1

CIT     P0PD    53485           CHR INPUT AND TEST
$CITS    STA SS01; STB SS02; STX SS03
        LDA* 0; SKA =37770B; BRM TRAP; AXC; EXU *+1,2
        BRU TCIT; BRM TRAP; BRM TRAP; BRU CI2
        BRU CI2; BRU CI2; BRU CI2; BRM TRAP

CI2     LDX BUF3,2; STX BUFF; SKE CSW,2; BRM TRAP
        SKE CFLG,2; SKN* CFLG,2; BRU CFLGEX
        LDA CPT0P,2; SKG CP,2; BRU CI5
        LDA CP,2; RSH 2; ETR K377; ADD BUFF; AXC
        LSH 2 (A CONTIANS CHAR. P0INTER); LDX CBF4,2 (DATA W0RD)
        COPY AX,XA,XB; EXU CI3,2 (P0SITI0N CHARACTER); ETR K377
        SKE SS01; BRU P0PX0
*       D0 A CI0 AND MIN 0 AFTER THE CI0. SEE WEX3 AND C256 C0DE.
        LDA WEX3; STA WEX; BRU CIE3
CI5     LDA K137; XMA SS01; SKE K137; BRU P0PX; BRU P0PX0
CI4     MIN 0   CI0 COMPLETED WITH0UT CALLING DISC DRIVERS.
        LDB WEX2; STB WEX; BRU P0PXB
CI3     LSH 8; RSH 8; N0P 0; BRM M0NCR

CI0     P0PD    56185           CHR INPUT/0UTPUT
$CIE    STA SS01; STB SS02; STX SS03
CIE3    LDA* 0; SKA =37770B; BRU CI0T1A; AXC; EXU *+1,2
        BRU CI0T1; BRU CI0T1; BRU P0PX; BRU CHRW
        BRU CHRW; BRU CHRW; BRU CHRW; BRM TRAP
CHRW    EQU *   CHARACTER READ/WRITE L0GIC
```

```
        LDX BUF3,2; STX BUFF; SKE CFLG,2; SKN* CFLG,2; BRU CFLGEX
        LDA CP,2; ETR THREE; SKN CSW,2; BRU *+3; MIN CC+MLVL,2; ADD FOUR
        XXA; LDB CHRW2,2; STB CHRW1; LDX CHRW3,2; XXA
        ADD CP,2; SKG CE0F1,2; BRU CHRW0
        SKN TIME; BRU *+4; SKN TTIME; SKN ACTR; BRU CHRW4
        STA CP1; LDA WEX1; STA WEX; LDA CP,2
CHRW0   XMA CP,2; RSH 2; ETR K377; ADD BUFF; COPY AX,A,B
CHRW1   0;* MODIFIED TO ONE OF THE FOLLOWING INSTRUCTIONS.
CHRW2   BRU CHR0; BRU CHR1; BRU CHR2; BRM MONCR
        BRU CHW0; BRU CHW1; BRU CHW2; BRM MONCR
CHRW3   DATA 1,1,2,0,1,1,2,0

CHRW4   LDB PACOMB; LDX =QQE; BRU POPOMS

* CIO INPUT ROUTINES DEPENDING ON CHAR. POSITION.
CHR0    LDA CBF4,2; LRSH 16; BRU WEX
CHR1    LDA CBF4,2; RSH 8; ETR K377; BRU WEX
CHR2    LDA CBF4,2; ETR K377; BRU WEX

* CIO OUTPUT ROUTINES DEPENDING ON CHAR. POSITION.
CHW0    LDB SS01; RCY 8; XMA CBF4,2; ETR K1S5; ADM CBF4,2
        LDA SS01; BRU WEX
CHW1    LDA SS01; ETR K377; LSH 8; XMA CBF4,2; ETR =77600377B; ADM CBF4,2
        LDA SS01; BRU WEX
CHW2    LDA SS01; ETR K377; XMA CBF4,2; ETR KM400; ADM CBF4,2
        LDA SS01; BRU WEX

WIO     POPD    56085           WORD INPUT/OUTPUT
$WIE    STA SS01; STB SS02; STX SS03
        LDA* 0; SKA =37770B; BRM TRAP; AXC; EXU *+1,2
        BRM TRAP; BRM TRAP; BRU POPX; BRU WRW
        BRU WRW; BRU WRW; BRU WRW; BRM TRAP
WRW     EQU *   WORD READ/WRITE LOGIC
        LDX BUF3,2; STX BUFF; SKE CFLG,2; SKN* CFLG,2; BRU CFLGEX
        LDA CP,2; SKA THREE; BRM TRAP (CURSOR MUST POINT TO A WORD.)
        ADD FOUR; SKG CE0F1,2; BRU WRW1
        STA CP1; LDA WEX1; STA WEX; LDA CP,2
WRW1    XMA CP,2; RSH 2; ETR K377; ADD BUFF; SKN CSW,2
```

```
         BRU WRW2 (INPUT)
         MIN CC+MLVL,2; CAX; LDA SS01; STA CBF4,2; BRU WEX
WRW2     CAX; LDA CBF4,2
WEX      BRU P0PX0 (CHANGED T0 N0P IF CP GT CE0F1)
         STA SS01; LDA WEX2; STA WEX
CALLD    BRM MPDSC; BRU C256
WEX1     N0P
WEX2     BRU P0PX0
WEX3     BRU CI4                    USED BY CIT T0 MIN 0 AFTER CI0 RUNS.

CFLGEX EQU *   CFLG EXCEPTI0N. TRAP IF FILE UN0PENED. 0THERWISE, ASSUME
*        BUFFER IS BUSY AND DISMISS EVEN TH0UGH BFR MAY HAVE BEC0ME FREE
         SKE CFLG,2; BRM CDMS1; BRM TRAP  DURING THE LAST FEW MICR0SEC0NDS

$FILINT EQU *  FILE EXCEPTI0N INTERRUPT C0DE. FILE EXCEPTI0N FLAGS IN B.
         CBA; MRG FILE; STA FLINT1
         SKN 0; BRU FLINT2 (D0NT FLAG USER'S FILE W0RD UNLESS IT IS IN
         CAB; LDX 0              HIS MEM0RY).
         LDA 0,6; ETR =77B5; LDX SS03; SKE =73B5 (IS HE EXCUTING A BRS);
         STB* 0  (N0, IT MUST BE A FILE SYSP0P. FLAG HIS FILE W0RD.);
FLINT2   BRM MPPACT; LDX PACPTR; LDA FLINT1; XMA SS01
         STA FLINT1; LDA K2B5; BRM RIIR; LDA FLINT1; XMA SS01
         SKA K2B5; BRU P0PX (E0F); BRM TRAP (FATAL EXCEPTI0N)
FLINT1   0;* FILE W0RD FLAGS.402B5=E0F,404B5=ERR,410B5=QUANTUM DEPLETI0N.

$CDMS1   0;*  DISMISS A USER UNTIL ALL HIS w BFR W0RK IS D0NE.
         LDA J0B; ADD CDMS10; CAB; LDX =GI0; BRU NP0PDS
CDMS10   10B TTN0 ACTIVATI0N C0NDITI0N

$CDMS2 EQU *   RETURN HERE F0R R0UTINES DISMISSED BY :CDMS:
         STA SS01; STB SS02; STX SS03
         LDA CRET0; STA 0; BRM MPDSC
         LDA CDSA; LDB CDSB; LDX CDSX; BRR CRET1

$MAYDMS EQU *  BRS 55. DISMISS IF USER IS RUNNING THE DISC.
         LDX J0B; SKN TTN0,2; BRU P0PX; MIN 0; BRM CDMS1

FSAVE    0;* SAVE REGISTERS AND SET UP F0R FILE SYSP0P. CALLED BY
*                          BRM FSAVE 0R BRM* FSAVE1 F0R SKIPABLE SYSP0PS.
```

```
        STA SS01; STB SS02; STX SS03; BRM MPDSC
        SKN 0; BRR FSAVE
        LDA* 0; BRM I8I         VALIDATE FILE NUMBER
        SKG TW8; BRU FSAVE2     FILE =0,1, 8R 2;
        LDX SS03; STA* 0; LDX BUFF
FSAVE1  BRR FSAVE
FSAVE2  LDX* FSAVE              WAS FSAVE CALLED INDIRECTLY.
        8RX NP8PX0              MAKE SKIP RETURN F8R PCE,SCP,SSP, AND BI8
        BRM NP8PX                   N8N-SKIP F8R RCP,RSP

* BRS 130   BREAK P8INT TEST
* SKIPS IS BREAK P8INT SWITCH IS D8WN.
* INPUT: X=SWITCH NUMBER.
$BPTEST SKN EXEC1; BRM TRAP;  LDX SS03
        LDA <1000; RSH 0,2; XMA BPT2; ETR KS5B3
        ADM BPT2
BPT2    BPT; MIN 0; BRU P8PX

* BRS 158   CHARGE R8YALTY PR8GRAM USER
* INPUT; A=NUMBER 8F CHARGE UNITS
* N8 SKIP: CHARGE EXCEEDED MAXIMUM 8R CAUSED AN 8VERFL8W
* SKIP: UNITS ADDED T8 CHARGE
$BS158 ADD RPCHG; SKG RPMAX; SKG RPCHG; BRU P8PX
        STA RPCHG; MIN 0; BRU P8PX

SNE     P8PD 53685                  SKIP IF N8T EQUAL
$SNEX   SKE* 0; MIN 0; BRR 0
SKP     P8PD 54485                  SKIP IF P8SITIVE
$SKPX   SKN* 0; MIN 0; BRR 0
SKL     P8PD 54585                  SKIP IF LESS
$SKLX   SKG* 0; BRU *+2; BRR 0; SKE* 0; MIN 0; BRR 0

$DTH     ZR8; ETR ADMSK; RSH 11; ST8 T; MUL THREE; CBX
        LDA RRL2; LDB RRL1; LCY 6,2; ETR K37
        LDB T; LCY 11; STA T; BRR DTH
$XDIR   0; E8M 200048; BRR XDIR   DISABLE INTERRUPTS


$XEIR   0; E8M 20002B; BRR XEIR
```

```
$RLABEL ZR0;*   SET RRL3=A. SAVE OLD RRL3 IN T4
        MLABEL; STA T4; BRR RLABEL

$BVIATS ZR0;* BIO BY WAY OF TS BLOCK. MOVE ONE DATA BLOCK.
        MIN WR4; LDX FILE; LDA BUF3,2; ADD =CBF4
        LDB* WR4; SKA WR5; XAB
        ADD BVLDA; STA BVC0PY; CBA; ADD BVSTA; STA BVC0PY+1; LDX KM400
BVC0PY  0; 0; BRX BVC0PY; BRR BVIATS
BVLDA   LDA 400B,2
BVSTA   STA 400B,2

$BRMC   ZR0;
        LDA* WR4; ETR =34000B; MUL =60000B; CAX; LDB WR11; LDA WR12
        LCY 6,2; ETR K37; STA BRMC1; LCY 6; ETR K37; CAX; BRR BRMC
$BRMC1  ZR0

*       **********  W BUFFER LOGIC

* BRS 100 ASSIGN DEVICE TO CHANNEL N
* DEVICE NUMBERS: 0=TAPE 0, 1=TAPE 1, 2=PRINTER
* INPUT: A=DEVICE NUMBER, X=CHANNEL NUMBER OR -1.
* RETURNS: NO SKIP=ERROR. DEVICE ALREADY ASSIGNED.
*          SKIP=ASSIGNMENT OK.
$AST    SKN WBIU; BRU P0PX
        SKN SYSS; BRU *+2; BRU AST2
        LDB X1; LDX J0B; SKB CPARW,2; BRU *+2; BRM TRAP
AST2    SKG TWO; SKG KM1; BRM TRAP; LDX SS03
        XXA; SKE KM1; BRU *+2; LDA UTTY; XXA
        SKN TTYASG,2; BRM TRAP
        STA DEVICE; STX DEVCH
        LDX TTYASG,2; LDA PTAB,2; LRSH 15; ETR K77; STA WJ0B
        LDA KM1; STA TPAR; MIN 0
        BRU P0PX
* BRS 101   UNASSIGN DEVICE TO CHANNEL N
$UAST   SKN SYSS; BRU *+2; BRU UAST2
        LDB X1; LDX J0B; SKB CPARW,2; BRU *+2; BRM TRAP
```

```
UAST2  LDA KM1; STA DEVICE; STA DEVCH; BRU P0PX
* BRS 104    WH0 HAS DEVICE
* OUTPUT: A=DEVICE, X=CHANNEL
$WHT   LDA DEVICE; LDX DEVCH; BRU XP0P
* BRS 107    SET PARITY
* INPUT: A NEG. FOR BCD (EVEN PARITY), A POSIT. FOR BINARY (ODD PARITY)
$SETPAR LDA =77776777B; SKN SS01; LDA KM1; STA TPAR; BRU P0PX
*
* GRAB MEMORY FOR W ROUTINES.
$GRBMW ZR0; STA GRBM2; BRM MPPACT; LDB 0,6; STB 0,6; BRM MPWB
       LDA GRBM2; BRR GRBMW
GRBM2  ZR0 0
*

$TCL2  LDA TRTW,2; STA TCLDMS; LDB TDS; LDX =GTI
       BRM MPPACT; LDA KM1; STA TNUM; STA WBIU; BRU P0PDMS
TDS    4 TCLDMS
*
TCLDMS SKS 10410B (TEST TAPE READY); BRU PACACT; BRU PEST

* TAPE READ/WRITE DRIVER
$TRD   ZR0; LDX TNUM; LDA =685; EXU ETTW,2; BRR TRD (END OF TAPE)
       EXU TRTW,2; BRU TRD2 (TAPE READY); LDA =785; BRR TRD
$TRD2  NOP 0 (READ/WRITE TAPE BINARY)
$I0RD  ZR0 0   I0RD W0RD. EOM 160000.
       PUT TD2
       MIN TRD; CLA; BRR TRD
$TD2   ZR0 0   INTERLACE PUT WORD. 0-9=WORD COUNT. 10-23=ADDR.

$TFIN  ZR0; LDA REAL; ADD =180; STA TREAL
       LDA PACPTR; STA WPTR; BRR TFIN

* THIS CHECKS TO SEE THAT A TAPE INTERRUPT HAS NOT
* BEEN DELAYED FOR MORE THAN 3 SECONDS.
$WCLEAN 0
       IF DISC=3
       BRR WCLEAN
       ELSE
       LDA REAL; DIR; SKG TREAL; BRU WCL5; EOM* 0; EIR
```

```
         LDX WPTR; LDA WCL2; STA PTEST,2; SKR PL,2; NOP
         LDA WPAGE; MRG =RWP*64; MLABEL; STA CLWRL
         LDA <4B5; ADM* TMCA
WCL6     LDA KM1; STA WBIU; STA UNIT
         SKN TNUM; BRU WCL3; STA PRDMS1; BRU WCL4
WCL3     XMA TNUM; CAX; SKR TPDMS,2; NOP
WCL4     LDA XX; STA TREAL; SKN WPAGE; BRU *+2; BRU WCL7
         DIR; BRM WREL; EIR
WCL7     LDX WJBB; LDA K2B5; ADM TTN0,2
         IF DISC<3
         SKN DSW; BRM IDMG0
         ENDF
         LDA CLWRL; STA RRL3; LRR3; POT RRL3
WCL5     EIR; BRR WCLEAN
CLWRL    ZRO 0
WCL2     226 PACDMB
         ENDF DISC=3

$TCA     DATA 0      CORE ADDRESS
$TWCNT   DATA 0      WORD COUNT
$TNR     DATA 0      NUMBER OF RECORDS
$TRA     DATA 0      RECORD ADDRESS IN PAGE 7
$TMAR    DATA 0      CHANNEL MAR FROM PIN
$TPAR    DATA 0      TAPE PARITY. 1B3=BINARY
$TNUM    DATA -1     TAPE NUMBER
$WPMT    ZRO 0    PMT ADDRESS
$TPDMS   DATA -1,-1 TAPE DISMISS. ACTIVATE WHEN NEGATIVE.
$TPDMS3  12B UNIT     WAIT FOR UNIT TO CLEAR
$TMCA    ZRO 0    MONITOR COMMUNICATIONS WORD IN PAGE 7.
$CTMCA   ZRO 0    TMCA FOR CONTROLS.
$WBIU    DATA -1    W BUFFER IN USE IF POSIT. DO NOT CHANGE DEV ASSIGN.
$WPTR    ZRO 0    PACPTR OF CONTROL FORK
$DEVICE  DATA -1   DEVICE IN USE ON W BUFFER
$DEVCH   DATA -1    CHANNEL ASSIGNED TO W BUFFER.
$TPE     DATA 0,0    MIN FOR EACH EOF AFTER TAPE END.
$TREAL   DATA 37777777B    TIME WHIN TAPE MUST FINISH. (REAL+3 SEC.)
$TURL    ZRO 0    USER PAGE NUMBER.
$FMK     HLT* FMK2                  POT WORD FOR EOF
```

```
FMK2    DATA 1786                   FILE MARK
$WJ0B   ZR0  0                      USER J0B N0.
$PRD6   ZR0 0   PRINTER P0T W0RD
$PRDMS1 DATA -1   PRINTER ACTIVATI0N W0RD

$NINT2 MIN ACTR; BRM WREL
        IF DISC<3
        SKN DSW; BRM IDMW
        ENDF
        LDX WJ0B; LDA K285; ADM TTN0,2; BRU WRET


*       DISC AND N0N-DISC W BFR INTERFACE R0UTINES  *********

WSA     0;*      W BUFFER INTERRUPT SAVE CELL
WSB     0
WSX     0
WSRRL3  0
$WPAGE  0;*      USER'S L0CKED PAGE
$WSW    0;*      RETURN SWITCH F0R WRET. 0 F0R BRR, -1 F0R BRI.

WSAVE   0;* SAVE REGISTERS AND RELABELLING F0R ANY W BUFFER INTERRUPT
        SKN WS1; SKR WS1; BRR WSAVE
        STA WSA; STB WSB; STX WSX; LDA RRL3; STA WSRRL3; BRR WSAVE
WS1     0;* SAVE 0NLY 0NCE SWICTH

$WRET   EQU *   RESTORE THE REGISTERS AND EXIT FR0M W BUFFER INTERRUPT
        SKN WSW; BRU *+3; IDT; EIR
        MIN WS1; LDA WSRRL3; MLABEL
        LDA WSA; LDB WSB; LDX WSX
        SKN WSW; SKR WSW; BRI* 31B
        IF DISC<3
        SKN IDMW; BRI INT31; BRR* 31B
        ELSE
        BRI INT31
        ENDF

$WB     0;*      D0 A BRING 0N THE 16 BIT ADRS IN A
        ETR =174B3; RSH 11; STA WPAGE; CAX; BRM DL0CKI; BRR WB
```

```
WREL    0;*     RELEASE WPAGE
        LDX WPAGE; BRM DLOCKD; BRR WREL

$INT31  0;*     THIS IS THE GENERAL W BUFFER INTERRUPT ENTRY POINT
        BRM WSAVE; LDA WPAGE; MRG =RWP*64; MLABEL
        LDA KM1; XMA UNIT; CAX
        EXU *+2,2; BRM MONCR; BRM MONCR
        BRJ TRINT       (UNIT = 1)
        BRJ TWI         (UNIT = 2)
        BRJ CTLI        (UNIT = 3)
        BRJ TEOFI       (UNIT = 4)
        BRJ TFSFI       (UNIT = 5)
        BRJ TBSFI       (UNIT = 6)
        BRJ PRDI        (UNIT = 7)

*       DISC DRIVER AND ASSOCIATED ROUTINES  **************

$DREAL  DATA 37777777B   SEE PJ 3 SECOND ROUTINE

* DRQ MAP
*    WORD 0: DISC ADDR.
*    WORD 1: 0-7=FILE OR JOB NUMBER  8-23=CORE ADDR.
*    WORD 2: 0=R/W. 1-4=INT. ROUTINE. 5-10=JOB
*    WORD 2 CONTINUED     11-16=GLOBAL FILE NUMBER.17-23=WORD COUNT/40B
*       WORD 2 NEGATIVE FOR WRITE.
$DRQ    EQU *

$TYMSET 0;* INITIALIZE TYMNET. CALLED FROM SET.
        LDA RLTYM; BRM RLABEL
        LDX =-2-NPORT; LDA =TBUF; MUL THREE; LSH 23
        STA TP+NPORT+1,2; STB TC+NPORT+1,2; BRX *-2
        LDA =ETBUF*3-1; STA TP+NPORT
        LDX =-NTBI
        LDA TFREEC; LDB TFREE; STA ETBI,2; BRX *+1; STB ETBI,2; BRX *-3
        LDX =-NTBO
        STA ETBO,2; BRX *+1; STB ETBO,2; BRX *-3
        LDX KM100; LDA T128A+100B,2; STA T128+100B,2; BRX *-2
        LDA =2285; STA ETBI-1; EOR KM1; STA ETBI-2; LDA KM2; STA TIX
```

```
        LDA T4; BRM RLABEL
        BRR TYMSET

M128    MACRO D
 DATA D(2)*^B4+D(1)*^B,D(4)*^B4+D(3)*^B,D(6)*^B4+D(5)*^B,D(8)*^B4+D(7)*^B
        ENDM

T128A   EQU *
        M128 0121,7011,7011,7011,7011,7011,7011,7011     0-7
        M128 7011,7071,7421,7011,7011,7431,7011,7011
        M128 7011,7011,7011,7011,7011,7011,7011,7011
        M128 7011,7011,7011,7041,7041,7041,7061,7051
        M128 4411,7411,6411,6411,6411,6411,6411,6411      40-47
        M128 6411,6411,6411,6411,6411,6411,6411,6411
        M128 4411,4411,4411,4411,4411,4411,4411,4411
        M128 4411,4411,6411,6411,6411,6411,6411,6411
        M128 6411,4411,4412,4411,4411,4413,4411,4411     100-107
        M128 4411,4411,4411,4411,4411,4411,4411,4411
        M128 4411,4411,4411,4411,4411,4411,4411,4411
        M128 4411,4411,4411,6411,6411,6414,6411,6411
        M128 4501,4501,4501,4501,4501,4501,4501,4501      140-147
        M128 4501,4501,4505,4501,4501,4506,4501,4501
        M128 4501,4501,4501,4501,4501,4501,4501,4501
        M128 4501,4501,4501,6501,6511,6511,6511,0121

        IF DISC<3
$GDC    0;* GET DISC TO CORE. B = CORE ADRS, A = DISC ADRS.
        STB T; STA T1
GDC1    ARMI =6B5; LDA T; RSH 14
        LDA K4000  (WORD COUNT); LCY 14; STA T2
        LSH 19; LDA T; ETR =1484; CBX
        RSH 14; CXB; LSH 5; MRG =IORDW; STA T3
        EXU DRT; BRU *-1
        EOM 10026B (ALERT DISC); POT T1
        EOM* 184; EXU T3; POT T2
        EOM 26266   DISC READ
        EXU DRT; BRU *-1; EXU DET; BRU GDC1; EXU DCT; BRU *-1
        BRR GDC
        ENDF
```

```
        BSS 3*NDRQ+DRQ-*    RESERVE 3*NDRQ CELLS BETWEEN DRQ AND DRQU
$DRQU   EQU *   END OF DRQ
        IF DISC=3; ZRO 0    THIS CELL MODIFIED AT MUX15 AND BY DIM; ENDF

        IF DISC<3


$IDMw   DATA -1    CALLED TO START DISC AFTER NON-DISC W BFR INTERRUPT
        STA IDMwA; STB IDMwB; STX IDMwX
        BRM IDMGO; BRU WRET
IDMwA   0;* RESTORE THESE REGISTORS AFTER THE DISC IS DONE. SEE :IDM2:
IDMwB   0
IDMwX   0

$IDMGO  0;*      GIVE THE DISC CONTROL OF THE W BUFFER
        LDA BRMW2; STA 31B; LDA BRMW3; STA 33B; LDA KM1; STA DSW
        CLA; STA UNIT; MIN WSW; BRM IDM; BRR IDMGO

* 'IDM'  4/16/66
* THIS IS THE DISC INTERRUPT ROUTINE FOR ALL DISC I/O
*   IDM MAKES ALL READS AT LEAST 64 WORDS.
IDMCK   ZRO;* 1 IF AN INTERRUPT IS PENDING. OTHERWISE 0.
DMOD    DATA 6    LAST OCTAL DIGIT OF CURRENT DP DEVICE ADRS.
$DPC    0;* NUMBER OF DP DISCS. SEE DRMSET.
$NDCL   ZRO;* COUNT OF DISC COMMANDS IN LIST
$EDCL   ZRO;* CURRENT END OF LIST
$IDCL   ZRO;* CURRENT INTERRUPT POINTER
$IDCL1  ZRO;* CURRENT COMMAND EXECUTING POINTER.
IDCL2   ZRO
ID01    ZRO     0                   WORKING CELL
ID02    EOM     17200B              I/O CONTROL EOM
ID03    DATA    -200B               READ DIFFERENCE
ID05    EOM     36660B              WRITE EOM
ID06    DATA    -1040B              READ  DIFFERENCE
IDSW1A  EOM     10026B              ALERT DISK
IDSW1B  BRU     ID3
IDCADD  ZRO     0
DRMTRY  ZRO;* TRY-AGAIN COUNTDOWN
```

```
$DSW    DATA -1 THIS SWITCH IS POSITIVE WHEN DISC WAITING FOR W BUFFER
*               TO BE FREED FROM ANOTHER W BUFFER UNIT
BRMW1   BRM  INT31
BRMW2   BRM IDM
BRMW3   BRM IDM2
$DTXS1   ZRO;* COMMAND COUNT
$DTXS2   ZRO;* LAST COMMAND LOC

$IDM    ZRO     0
DRT     SKS 10026B; BRU DRT1    DISC READY TEST
DET     SKS 11026B; BRU ID7     DISC ERROR TEST
DCT     SKS 11000B; BRU ID7     DISC CHANNEL ERROR TEST
IDSW1   BRU     ID3             ALERT DISK OR BRU ID3
        POT*    IDCL            DISK ADDRESS
        EOM*    10000B          ALERT CHNL
IDE2    BRU     *               I/O CONTROL EOM
        POT     IDCADD          CORE ADDRESS
IDE1    BRU     *               R/W BUC
        BRM wSAVE; SKR NDCL; NOP 0      (SAVE REG.,DECR. COMMAND COUNT)
        LDA REAL; STA DREAL; MIN IDMCK
        LDA IDSW1B; STA IDSW1
        LDA IDCL; ADD THREE; SKE =DRQU; BRU *+2; LDA =DRQ; XMA IDCL (NXT CMD
ID4     XMA IDCL1; STA IDCL2; SKG KM1; BRU ID5 (IDCL1 IS EXECUTING)
        MIN XARMOT
        CAX; LDA 1,2; RSH 11; ETR K37; CAX; BRM DLOCKD
        LDX IDCL2; LDA 2,2; RSH 13; ETR K77; STA IDJOB
        CAX; LDA K265; ADM TTNO,2; SKN TTNO,2; MIN ACTR
        LDX IDCL2; LDB 2,2; LSH 5; ETR K17; CAX; LDA IDT,2; ETR XX
        STA IDTX; SKG IDT,2; EXU IDTX
        CLA; LSH 12; ETR K77; ADD =GFLG; STA IDFILE
        SKG =GFLG; BRM MONCR
        LDA* IDFILE; RSH 15; ETR THREE; CAX; LDA BUF,2; STA IDBUFF
        EXU IDTX

IDTRET LDA THREE; STA DRMTRY            (DISC TRIES 4 TIMES)
ID5     LDA IDCL; SKN NDCL; BRU ID6; BRU ID1 (SETUP NXT CMND OR WINDUP)
ID6     EQU *       SETUP NEXT COMMAND
        STA IDCL; CAX
        LDB 0,2; LSH 6; ETR THREE; SKG DPC; BRU *+2; BRM MONCR
```

```
         SKG 8NE; ADD SIX; SKE DM8D; BRU *+2; BRU ID6A
         SKN IDCL1; BRU ID1; SUB DM8D; ADM DM8D
         ADM DRT; ADM DET; ADM IDSW1A; ADM ID05
ID6A     LDA 1,2; LRSH 14; ETR THREE; STA ID01  (2 HIGH 8RDER C8RE ADR BITS)
         CLA; RSH 5; LDA 2,2; ETR K177   (W8RD C8UNT/40B)
         SKG 8NE; LDA Tw8 (W8RD C8UNT IS 40B. MAKE IT 100B);
         RSH 5; STB IDCADD
         ADD ID02; SKN 2,2; ADD ID03; STA IDE2
         LDB ID01; LSH 29; ADM IDE2
         LDA IDSW1A; STA IDSW1          (SET SWITCH T8 XEQ C8MMANDS)
         LDA ID05; SKN 2,2; ADD ID06; STA IDE1 (MAKE READ/WRITE E8M)
         SKN IDCL1; BRU ID1; BRU IDSW1      (EXIT IF CMND ACTIVE)
ID1      SKN wSw; BRU wRET
         IF ST
         SKN STSw; BRU ID1A
         LDA =RSR; BRM DL8CKR; BRM SSS
         2 =5B5+2
         1 REAL
ID1A     EQU *
         ENDF
         SKN IDCL1; BRU *+2, BRU wRET
*        RESTORE STATE OF MACHINE, THEN CHECK FOR LOST INTERRUPT. SEE IF
*        IDCL1 COMMAND HAS FINISHED.
         MIN WS1; LDA WSRRL3; MLABEL; LDA WSA; LDB WSB; LDX WSX
         CZTw   SKIP IF W BFR W8RD C8UNT REGISTER = 0; BRI IDM
         BRU IDM+1 (N8TE THAT IT IS ASSUMED THAT IDM WAS CALLED AS AN
*        INTERRUPT (DIDNT CHECK wSW).
ID3      BRM WSAVE; LDA KM1; BRU ID4 (SETUP T8 INDICATE N8 CMND EXECUTING)

IDDR     EQU *    PR8CESS DATA BL8CK READ INTERRUPT
         LDX IDCL2; LDA 2,2; SKA =176B; BRU IDDW
*        W8RD C8UNT IS 40B. CHECK CHECKSUM.
         BRM IDLBL
         LDx IDCL2; LDB 0,2; SKB =74B6; BRU *+2; BRU IDDR1
         LDA 1,2; MRG =34000B; C8PY Ax,BA; BRM CKSUM
         LDB =74B6; SKM* IDCL2; BRU *+2; BRU IDDR1
         MIN DF; LDA* IDCL2; BRM IDER; BRM IDFAIL
IDDR1    LDA IDLBL1; MLABEL
IDDW     EQU     *
```

```
        LDA =-485; ADM* IDFILE
        BRU IDFRET

DRT1    EQU *  DISC WASNT READY WHEN IT SHOULD HAVE BEEN
        MIN DXX; EXU DRT; BRU *+2 (STILL NOT READY);
        BRU DET (A FEW MICROSECONDS LATE, MAY LOSE A REVOLUTION);
        BRM WSAVE
DRT2    LDX =-1000/7   SET UP FOR 1 MILISECOND LOOP
        EXU DRT; BRX *-1   7 MICROSECONDS PER ITERATION
        BRX DET  BRANCH IF DISC BECAME READY IN LESS THAN A MILISECOND
        LDA DM8D; SKA FOUR; SUB SIX; CAX; MIN D26,2 (RECORD SERIOUS FAILURE
        BRU DRT2

ID7     EQU *   ERRORS
        SKN IDCL1; BRU *+2; BRU ID3
        BRM WSAVE; LDA* IDCL1; BRM IDER   RECORD DISC ADDRESS
        EXU DET; MIN D5; EXU DCT; MIN DC
        SKR DRMTRY; BRU ID7A; MIN DU; BRU IDSW1
ID7A    MIN NDCL; LDA KM1; XMA IDCL1; BRU ID6

IDFRET  EQU *   CHECK FOR HARDWARE ERRORS AND RETURN
        SKN DRMTRY; BRU IDTRET
        BRM IDFAIL; LDA* IDCL2; BRM IDER; BRU IDTRET

IDM2    ZRO; SKN IDCL1, BRU IDM22
        BRM WSAVE; LDA BRMW1; STA 31B; STA 33B
        LDA KM1; STA UNIT; XMA IDMW    RESET IDMW
        XMA IDM2; STA INT31  MOVE IDM2 TO INT31 AND SAVE IDMW
        SKN IDM2; BRU *+2 (RETURN TO NON-DISC W BFR ACTIVITY); BRU WRET
        LDA WPAGE; MRG =RWP*1003; MLABEL
        LDA IDMWA; LDB IDMWB; LDX IDMWX; BRR IDM2
IDM22   EXU DCT; BRU *+4; EXU DET; BRU *+2; BRI IDM2
        MIN WSW; BRM IDM; BRU IDM2+1

        ELSF DISC=3

$RQ1A   EQU 30B HEAD OF LIST CREATED BY DTC
$RQ1    EQU 32B HEAD OF LIST THAT 620 IS PROCESSING. SET UP BY DTS
$RQ2    EQU 34B CURRENT JOB BEING PROCESSED BY 620 UNLESS RQ1 = RQ2
```

```
$B300A 0 B300  SEE TLOG, DRMSET AND DIM.

IDA     0
IDB     0
IDX     0
IDMC1   0

$IDM    0; MIN IDMC1
        STA IDA; STB IDB; STX IDX
        LDA REAL; STA DREAL;* STORE REAL TIME IN DREAL
ID1     LDA RQ3; SKE RQ2; BRU ID2
        LDA IDA; LDB IDB; LDX IDX
        BRI IDM
ID2     MIN XARMST
        LDX RQ3; LDA 2,2; RSH 13; ETR K77; STA IDJOB
        COPY AB,AX; LDA K2B5; SKB KM1; ADM TTNO,2; SKN TTNO,2; MIN ACTR
        LDX RQ3; LDA 1,2; RSH 11; ETR K37; CAX; BRM DLOCKD
        LDX RQ3; LDB 2,2; LSH 5; ETR K17; CAX; LDA IDT,2; ETR XX
        STA IDTX; SKN IDT,2; EXU IDTX
        CLA; LSH 12; ETR K77; ADD =GFLG; STA IDFILE
        SKG =GFLG; BRM MONCR
        LDA* IDFILE; RSH 15; ETR THREE; CAX; LDA BUF,2; STA IDBUFF
        EXU IDTX

IDTRET  LDX RQ3; SKN 1,2; BRU *+4; LDA 0,2; MIN DU; BRM IDER
        EAX RQ3; BRM INCRQ; BRU ID1

$INCRQ  0;* INCREMENT DRQ POINTER AND WRAP AROUNDD IF NECESSARY
        LDA 0,2; ADD THREE; SKE =DRQU; BRU *+2; LDA =DRQ; STA 0,2; BRR INCRQ

IDDR    EQU *
        LDX RQ3; LDA 2,2; SKA =176B; BRU IDDW
        BRM IDLBL
        LDX RQ3; LDA 1,2; MRG =340000B; AXC
        LDB* RQ3; SKB X7; BRM CKSUM (CHECKSUM BFR IF CKSM IS KNOWN)
        LDB X7; SKM* RQ3; BRU *+2; BRU IDDR1
        MIN DF; LDA* RQ3; BRM IDER; BRM IDFAIL
IDDR1   LDA IDLBL1; MLABEL
```

```
IDDW    EQU *
        LDA =-485; ADM* IDFILE

IDFRET  EQU *
        LDX RQ3; LDA 0,2; SKN 1,2; BRU IDTRET; BRM IDFAIL; BRU IDTRET

        ENDF
IDT     EQU *   DISC INTERRUPT DISPATCH TABLE WITH FLAGS
* FLAGS         BIT 0 FOR FILE OPERATION,BITS 1 AND 2 UNUSED
        BRU IDTRET; BRU IDDR,4; BRU IDDW,4; BRM MONCR
        BRU ID124; BRM MONCR; BRM MONCR; BRM MONCR
        BRU IDBPR,4; BRU IDBPW,4; BRU IDBR,4; BRU IDBW,4
IDTX    ZRO 0   INSTR FROM IDT MINUS FLAGS


IDBR    EQU     IDDR            BIO READ INTERRUPT
IDBW    EQU     IDDW            BIO WRITE INTERRUPT
IDBPR   EQU *   BIO READ INTERRUPT FOR PAGE BOUNDARY
IDBPW   EQU *   BIO WRITE INTERRUPT FOR PAGE BOUNDARY
        BRM IDLBL; LDX WR4; SKN 1,2; BRU IDDR1
        LRR1; PUT WR11; LRR2; PUT WR12
        BRM SVIATS; BRM BRMC; BRM DLOCKD; LDX BRMC1; BRM DLOCKD
        LDA RRL3; ETR K37; CAX; BRM DLOCKD
        EIR; LRR1; PUT RRL1; LRR2; PUT RRL2
        BRU IDDR1

IDFAIL  0;* FAILURE DURING FILE I/O. SET FLAG TO CAUSE TRAP IN IOI.
        LDA* IDFILE; MRG =484; STA* IDFILE; BRR IDFAIL


IDLBL   ZRO;*   LABEL TS BLOCK
        LDA =0600B; MLABEL; STA IDLBL1; LDX IDJOB
        LDA* PMTP,2; ETR K37; ADM RRL3; LRR3; PUT RRL3; BRR IDLBL
IDLBL1  ZRO

$CKSUM  0;* COMBINE AND FOLD DISC ADR AND WORDS 0,1,2, AND 37B OF BFR
*       A CONTAINS DISC ADR. X CONTAINS CORE BFR ADR.
        IF DISC=3; LDA 0,2; ELSE; ETR DMSK; EOR 0,2; ENDF
        EOR 1,2; EOR 2,2; EOR 37B,2
```

```
        STA CKS9; LSH 8; STA CKS8; LSH 8; EOR CKS9; EOR CKS8
        STA CKS9; LSH 4; EOR CKS9
        IF DISC<3; ETR =7486; ELSE; ETR X7; ENDF
        BRR CKSUM
CKS9    0;* TEMP FOR FOLDING OPERATION
CKS8    0

$DLOCKI ZRO;*     LOCK PAGE FOR DISC
        DIR
        LDA =6006; ADD RLTS; BRM DLOCKR
        LDA RMC,2; SKG =775B; SKN XPB; BRU *+3; MIN DSKMET; MIN SDSKMT
        LDA K777; ADM RMC,2; LDA RMT,2; BRM B; LDA KM4; STA DLOCK
        LDX =7-NMEM; LDA =775B; SKG ERMC,2; MIN DLOCK; BRX *-2
        LDA DLR1; BRM DLOCKR
        EIR; BRR DLOCKI

$DLOCKD ZRO;*     DECREMENT PAGE LOCK FOR DISC
        DIR
        LDA =0682; BRM DLOCKR
        LDA RMC,2; SUB K777; STA RMC,2; SKG =776B; SKR DLOCK; NOP
        LDA RMT,2; BRM REL; BRM DRAGO
        LDA DLR1; BRM DLOCKR
        EIR; BRR DLOCKD

DLOCKR  ZRO;* SET RELABELLING FOR DLOCK ROUTINES
        XMA RRL3; LRR3; POT RRL3; STA DLR1; BRR DLOCKR
DLR1    ZRO

ID124   EQU *   INTERRUPT ROUTINE FOR BRS 122,123,124,125
        IF DISC<3; SKN DRMTRY; ELSE; LDX RQ3; SKN 1,2; ENDF; BRU IDTRET
        LDA =6006; BRM DLOCKR
        LDX IDJOB; SET =484,TTNO SET BRS 122,123,124,125 ERROR BIT
        LDA TTNO,2; ETR K77; CAX; LDA TTYASG,2    FIND FORK DOING I-O
        CAX; LDA PPTR,2; RSH 12; MRG PLMSK; SKE PLMSK; BRU *-5
        LDA IDJOB; ADD CDMS10; SKE PTEST,2   (IS FORK DISMISSED ON IO)
        BRU ID124A   NO, OR YES BUT FORK IS ALREADY SET TO RUN AT PACTRP
        SKR PL,2; NOP 0
        ADD =1285; STA PTEST,2    SET PTEST TO ACTIVATE AT PACTRP
        SUB =2085  (MAKE PUTST WORD); XXA; RSH 12  (PUPAC IS BO TO B11);
```

```
        LDA =-2B5; ADM 0,2  DECR. PTEST SO FORK WONT RUN BEFORE PU
        LDA 0,2   CHANNEL NUMBER IN X REQUIRED BY PU; XXA
        BRM EPU
ID124A  LDA DLR1; BRM DLOCKR
        BRU IDTRET

$BRS167 EQU * READ AND RESET BRS122,123,124,125 ERROR FLAG
        LDX JOB; LDA =4B4; SKA TTNO,2; BRU *+2; BRU POPX
        CNA; ADM TTNO,2; BRU POPXO

* ROUTINE TO SAVE DISC ADDRESS OF ERROR
$IDER   0;* PUT DISC ADR OF FAILURE INTO TABLE
        ETR KS7; STA* IDER1; MIN IDER1; SKN* IDER1; BRR IDER
        LDA EIDER; ADM IDER1; BRR IDER
IDER1   ZRO *+1
$IDER2  BSS 10
EIDER   DATA    -10
IDJOB   0
IDFILE  0
IDBUFF  ZRO     0                       TS BLOCK BUFFER ADDRESS
$XARMOT 0;* MINNED IN IDM FOR EVERY DISC TRANSFER COMPLETED
$DLOCK  DATA    -4 4 LESS THAN THE NUMBER OF PAGES LOCKED FOR DISC
$DMSK   DATA 3777777B   DISC ADRS MASK
$NDMSK  DATA 7406
$DSCTOP DATA 8*128000-1   MAXIMUM 8 PAC 2314 ADDRESS ALLOWED BY DTC

$BJOB   ZRO
$SWPCNT BSS NJOB;* NUMBER OF PAGES SWAPPED IN FOR THIS JOB.
$CCT    BSS NJOB;* NUMBER OF CHARACTERS FOR THIS JOB.
$PTCKS  BSS NJOB;* PACE*TICKS, CORE CHARGE.
$PUCT   BSS NPUQ*4
$EPUCT3 EQU PUCT+NPUQ*4+4
$EPUCT  EQU PUCT+NPUQ*4
$EPUCM3 EQU PUCT+NPUQ*4-4
        FORGET
ENDI    EQU *
        END
```

```
S       IDENT                                                   6/21/72  14:55     PAGE   1
        LISTM


PSP     MACRO D   CREATES ASC CELLS AND DATA CELLS FOR SYSTEM COUNTERS
PSPN    NARG D
$ECA    EQU *   DEFINES THE BEGINNING OF THE ASC AREA
PSPJ    EQU 0
        RPT PSPN
PSPJ    EQU PSPJ+1
        ASC '•&D(PSPJ)•&'
        ENDR
$ECB    EQU *   DEFINES THE BEGINNING OF THE PSP COUNTERS
PSPJ    EQU 0
        RPT PSPN
PSPJ    EQU PSPJ+1
$D(PSPJ) ZRO
        ENDR
$ECC    DATA -PSPN
        ENDM; FRGT PSP,PSPN,PSPJ


        ENTRY   FGLIST,FBMAP,EFGL,MACH
        ENTRY   PRILEV,NSYS,SST,EXECS,EX,NEXEC,SYSS
        ENTRY   VERS,STATUS
        ENTRY   NU,MAXNU,UTIME,RPAGE,DBITS
        ENTRY   SDBM8,DRC,RWD,RWR,SW205,RS
        ENTRY   XBERR,FDERR,TNICTR,TFICTR,ARMBT
        ENTRY   TIIBAD,T0IBAD,EXTPU,NB112
        ENTRY   SWXMA,DISCN,FATIG
        ENTRY   PAGES,TIICTR,T0ICTR
        ENTRY   MERIS
        ENTRY   BRSTV,BRSTVA,PWFL,M48K
        ENTRY   PACPTR,J0B,UTTY,TJ0B,PMTJ0B
        ENTRY   STIME,REAL,SETTB,ETTB,WETTB
        ENTRY   PMTP,RL3,ADRSMT
        ENTRY   CPARW,AUNN,EAUNN,DMIN,ACST,PACST
```

```
        ENTRY   XPB,EXEC1
        ENTRY   MISC
*
* CONSTANTS
*
$ZERO DATA 0
$ONE DATA 1
$TWO DATA 2
$THREE DATA 3
$FOUR DATA 4
$FIVE DATA 5
$SIX DATA 6
$SEVEN DATA 7
$EIGHT DATA 8
$NINE DATA 9
$TEN DATA 10
$K13 DATA 13B
$K14 DATA 14B
$K15 DATA 15B
$K16 DATA 16B
$K17 DATA 17B
$K20 DATA 20B
$K21 DATA 21B
$K22 DATA 22B
$K23 DATA 23B
$K24 DATA 24B
$K25 DATA 25B
$K26 DATA 26B
$K27 DATA 27B
$K30 DATA 30B
$K37 DATA 37B
$K40 DATA 40B
$K77 DATA 77B
$K100 DATA 100B
$K137 DATA 137B
$K177 DATA 177B
$K200 DATA 200B
$K377 DATA 377B
$K400 DATA 400B
```

```
$K777 DATA 777B
$K1000 DATA 1000B
$K1000D DATA 1000
$K1777 DATA 1777B
$K2000 DATA 2000B
$K3777 DATA 3777B
$K4000 DATA 4000B
$K7777 DATA 7777B
$K1B4   DATA 1B4
$K1S4   DATA 17777B
$K2B4   DATA 2B4
$K3S4   DATA 37777B
$K4B4   DATA 4B4
$KS5    DATA 77777B
$K1B5 DATA 1B5
$K1S5 DATA 177777B
$K2B5 DATA 2B5
$K3S5 DATA 377777B
$K4B5 DATA 4B5
$KS6 DATA 777777B
$K1B6 DATA 1B6
$K1S6 DATA 1777777B
$K2B6 DATA 2B6
$K3S6 DATA 3777777B
$K4B6 DATA 4B6
$KS7 DATA 7777777B
$K1B7 DATA 1B7
$K1S7 DATA 17777777B
$K2B7 DATA 2B7
$K3S7 DATA 37777777B
$K4B7 DATA 4B7
$K3B7  DATA 3B7
$K5B7 DATA 5B7
$K6B7 DATA 6B7
$K6S7 DATA 67777777B
$K7B7 DATA 7B7
$K77B6 DATA 77B6
$K777B5 DATA 777B5
$KS4B4 DATA 7777B4
```

```
$KS5B3 DATA 77777B3
$KS6B2 DATA 77777700B
$KM1 DATA -1
$KM2 DATA -2
$KM3 DATA -3
$KM4 DATA -4
$KM5 DATA -5
$KM6 DATA -6
$KM7 DATA -7
$KM8  DATA -8
$KM9  DATA -9
$KM10D DATA -10
$KM40  DATA -40B
$KM100 DATA -100B
$KM200 DATA -200B
$KM300 DATA -300B
$KM400 DATA -400B
$KM4B3 DATA -4000B
$KM4B4 DATA -40000B
$KM4B5 DATA -400000B
$KM4B6 DATA -4000000B
$KM10 EQU KM8
$PLMSK EQU <S4B4
$PRMSK EQU K7777
$ADMSK EQU <3S4
$NADMSK EQU KM4B4
$PGMSK EQU <3777
$NPGMSK EQU KM4B3
$X0    EQU ZERB
$X1    EQU <1B7
$X2    EQU <2B7
$X3    EQU <3B7
$X4    EQU <4B7
$X5    EQU <5B7
$X6    EQU K6B7
$X7    EQU K7B7
$XX    EQU <3S7
*
* SYSTEM TABLES AND WBRDS
```

```
*
PACPTR  ZRB
JOB     ZRB
UTTY    ZRB
XPB     ZRB 0   PB POINTER
EXEC1   ZRB 0   NEGATIVE FOR SUBSYSTEM STATUS
EX      BSS 0   EXEC STATUS TRAP
NEXEC   ZRB 0   EXEC STATUS TRAP WITH RELABELING
EXECS   ZRB 0   NEG. FOR EXEC STATUS
SST     BSS 0   SYSTEM STATUS TRAP
NSYS    ZRB 0   SYSTEM STATUS TRAP WITH RELABELING.
SYSS    ZRB 0   NEG. FOR SYSTEM STATUS
PRILEV  DATA 0;* 1B7 FOR NO STATUS, 2B7 FOR SUBSYS, 3B7 FOR SYS, 3786 EX.
STATUS  ZRB 1   STATUS CONTROL WORD.
TJOB    ZRB 0                   ELAPSED TIME POINTER
PMTJOB  ZRB 0                   POINTER TO CURRENT PMT
ADRSMT  ZRB 0   ADDRESS OF SMT
        IF EXPMT
$ADRPMT ZRB PMT
$AEXSMT ZRB EXSMT
        ENDF
M48K    DATA    0               -1 FOR 48K. 0 FOR 64K.
MAXNU   DATA    NJOB-1          MAXIMUM NUMBER OF USERS. SEE TY11.
SDBM8   DATA    -1              -1 IF BIT MAP NOT SET,0 OTHERWISE
$MAPBIT 0;* A SINGLE BIT WHICH TELL DISC ZONE.REDUNDANT TO MAPNUM.
DRC     DATA    400000B         CONSTANT FOR LOCATING EXEC ON DISC
$SW16   0;* SET TO -1 BY DRMSET IF BPT2 SET FOR 16 DISC SYSTEM
RWD     DATA    -1              MINNED BY SET IF DISC WRITE PROTECTED
RWR     DATA    -1              MINNED BY SET IF RAD WRITE PROTECTED
SW205   DATA    -1              MINNED BY SET IF 205 INTRPT IS NOT HIGH
SWXMA   DATA -1
MACH    ZRB 0;*  MACHINE NUMBER
DISCN   ZRB 0;* DISC NUMBER
VERS    ASC     '23  '          EXEC VERSION IS CALLED SYSTEM
$FG940A DATA    40516337B   940 UP FLAGS
$FG940B DATA 51506262B      940 UP FLAGS
$NT19   EQU *-ADRSMT            SIZE OF TABLE 19 FOR BRS 7
* JOB INDEXED TABLES
PMTP    BSS NJOB POINTER TO FIRST PMT ENTRY
```

```
$EPMTP EQU PMTP+NJ0B
RL3    BSS NJ0B M0NIT0R MAP. SH0ULD C0NTAIN 0NLY TS BL0CK MAP
$ERL3  BSS 0
WERIS  EQU *                    USER ASSIGNED T0 TTY
       RPT NP0RT; DATA -1; ENDR
$EWERIS EQU    *
* J0B - TELETYPE INDEX
* BITS 0-7=DISC BUSY BITS, 1B5 IS PUTIM BUSY BIT, 4B4 F0R ID124 ERR0RS
*      BIT 15-17 (700B) F0R RAD ERR C0UNT
$TTN0  0; RPT NJ0B-2; DATA *+1-TTN0; ENDR
$ETTN0 DATA 2-NJ0B
$FULST DATA 1

*
* EXEC PARAMETERS
CPARW  BSS     NJ0B   USER'S C0NTR0L PARAMETERS.
*                     BITS: 0=SUBSYSTEM, 1=DEVICE, 2=0PERAT0R, 3=SYSTEM
*                           4=ACCT SUPERVIS0R, 5=EXEC
AUNN   BSS     NJ0B             ACCT/USER N0S.
EAUNN  EQU     *
ACST   BSS     64
$EACST EQU     *
* EXEC SWITCHES. SET BY BRS 136. D0 N0T REORDER.
$SWEX  BSS     0
$SWLET DATA    0                LETTER R0UTINE SWITCH
$PACST ZR0     ACST
       ZR0     0                FILL CELL F0R TABLE.
$ACA   ZR0     0                ACC0UNTING AREA
D0WN   DATA    0                N0 L0NGER REFERENCED BY M0NIT0R 0R EXEC
       BSS 1
FATIG  DATA -1;* MINNED IF REAL IS T00 LARGE.
$SWMBL DATA    -1               MAIL B0X LIST BUSY
$SWMBM DATA    -1               MAIL B0X MESSAGES BUSY
$ENMSK DATA    -1               SUBSYSTEM ENABLE MASK.
$SWFD  DATA    0                FILE DIR. BUSY F0R USER SPECIFIED.
* END 0F TABLE
*
$LDFACT        DATA NJ0B        L0AD FACT0R
$ACTIME DATA 0                  LAST TIME ACC0UNTING WAS WRITTEN.
```

```
$NOACT DATA    0                ACCOUNTING SWITCH. =1 FOR ACCOUNTING.
$USNO  DATA 0   USER NO. LOCKED OUT BY EXEC
$GBUSY BSS NJOB   ASSOCIATED WITH FGLIST IN THE EXEC
FGLIST BSS NJOB   FILE GROUP LIST
EFGL   BSS 0
SOV    EQU 40000B  CHANGE WHEN OVERFLOW FD AREA MOVES.
FBMAP  BSS 60000B/24-SOV/24+1


* SYSTEM PARAMETERS
*   RC = ERRORS DETECTED BY SKS 51000 PLUS ERRORS DETECTED BY SKS 51026
*   RS = ERRORS DETECTED BY SKS 51000
*   RA=RAD ADDRESSING, RU=RAD UNCORRECTABLE
*   RAT=RAD ANGLE, DC=DISC CHANNEL, DS=DISC SEEK, DF=DISC FAIL
*   DU=DISC UNCORRECTABLE, CPC=CPU PARITY
*      DXX IS MINNED WHENEVER A DP DISC WAS NOT READY ON ENTERRING IDM
*      D26,D27,D22,D23 ARE MINNIED FOR EVERY MILISECOND OF CONTINUOUS
*                NOT READINESS. SEE DRT1 CODE.
*      BZ = NUMBER OF TTY BUFFERS ZAPPED BY TG6 CODE IN NET
       PSP     RC,RA,RU,RAT,DC,DS,DF,DU,DXX,D26,D27,D22,D23,CPC,BZ

MISC   BSS     0
PWFL   ZRO     0                NO. OF POWER FAILURES.
$MAPNUM 0;* AN INTEGER WHICH TELLS WHICH ZONE OF DISC IS MAPPED
DBITS  DATA    TABLEC-1         NO. OF FREE BITS IN BIT MAP
RS     ZRO     0                RAD CHANNEL ERRORS.
TNICTR ZRO     0                TTY ON INTERRUPTS
TFICTR ZRO     0                TTY OFF INTERRUPTS
SPARE1 0
SPARE2 0
TIICTR ZRO     0                TTY INPUT INTERRUPTS
TIIBAD ZRO     0                BAD CHANNEL ON INPUT INTERRUPT
TOICTR ZRO     0                TTY OUTPUT INTERRUPTS
TOIBAD ZRO     0                BAD CHANNEL ON OUTPUT INTERRUPT
NU     ZRO     0                NO. OF USERS ON
UTIME  ZRO     0                NO. OF USER ON SECONDS
REAL   ZRO     0                NO. OF CLOCK TICKS
SETTB  ZRO     0                UNSOLD TIME
```

```
WETTB   ZRØ     Ø               CPU NØT USED TØ SPEED UP DISK IØ
STIME   ZRØ     Ø               TIME IN SYSTEM MØDE.
$SPT    ZRØ     Ø               PAGE TICKS FØR ALL USERS
PAGES   ZRØ     Ø               NØ. ØF PAGES SWAPPED
$SSWC   ZRØ     Ø               SWAP CØUNT FØR ALL USERS
RPAGE   ZRØ     Ø               NØ. ØF READS
$PAGEW  ZRØ     Ø               NØ. ØF WRITES
$SDSKMT ZRØ     Ø               DISK ACCESS UNITS FØR ALL USERS
ARMØT   ZRØ     Ø               DISC ARM MØTIØN
XBERR   ZRØ     Ø               NØ. ØF X-BLØCK ERRØRS WHILE SETTING MAP
FDERR   ZRØ     Ø   NØ. ØF FILE DIR. ERRØRS WHILE SETTING DISC BIT MAP
EXTPU   ZRØ     Ø               EXTRA PU ENTRIES AT BRS 112
NB112   ZRØ     Ø               NUMBER ØF LØGØUTS
$RBER   ZRØ     Ø               RUBØUT ERRØR ØN PU.
$UPSHUT Ø Ø    UP,SHUT FLAG. 2 FØR UP, 1 FØR SHUT.
$LCDFIB ZRØ Ø;*  LØST CHARACTERS DUE TØ FULL INPUT BUFFER.
        BSS 6
$EQPT   EQU  *  BITS IN THIS CELL TELL THE STATE ØF MØNITØR IF SWITCHES
        IF FPH,DUM EQU 1; ELSE;DUM EQU Ø; ENDF
        IF PH2;DUM EQU DUM+2; ENDF
        IF DISC=3;DUM EQU DUM+4; ELSF DISC=2;DUM EQU DUM+8; ENDF
        IF RP;DUM EQU DUM+20B; ENDF
        DATA DUM; FRGT DUM
$NMISC EQU *-MISC               SIZE ØF MISC TABLE FØR RWTAB MACRØ
*       MISC TABLE ENDS HERE. LENGTH ØF TABLE IS 45B. MAINTAIN THIS
*       CØMMENT AND A CELL IN RWTAB FØR BRS 7 WHEN CHANGING MISC TABLE.
*       THE LENGTH ØF THE MISC TABLE CAN BE FØUND BY SAYING:
*       NMISC= IN DDT AFTER LØADING S. THE SIZE ØF TABLE 19 CAN
*       BE FØUND BY SAYING: NT19=

* BRS MØDULE TRANSFER VECTØR
BRSTV   DATA Ø,16B,13B6,131484,131415B2,17B
BRSTVA DATA 40010005B
```

```
* TIMING TABLES
DMIN    DATA -3                    SYS. START-UP DATE/TIME IN MINS.
*                        1ST 6 BITS FOR MONTH. REST FOR MINUTES OF MONTH.
$YEAR   DATA    10B   YEAR NUMBER
$LGMON  ZR0     0 LAST TIME MONITOR DATA ENTERED IN ACCOUNTING
$CZONE  ZR0     0 CURRENT TIME ZONE FOR EXEC.
ETTB    BSS NJOB                    COMPUTE TIME COUNTER
$STACT  0;*     START ACCOUNTING CURSOR POSITION
$CRACT  0;*     CURRENT ACCOUNTING CURSOR POSITION
$DSTSW  0;*     SWITCH FOR GMT TIME CONVERSION
$TMK1   0;*     CONSTANT FOR LOCAL TIME CONVERSION (BRS 42)
$TMK2   0;*     CONSTANT FOR LOCAL TIME CONVERSION

ENDSYM  BSS 0
        FORGET
        END
```

```
        LISTM

*       MDBG DEFINITIONS THAT ARE USEFUL TO HAVE  IN THE SYMBOL TABLE
        DEFINE ALARM,TBI,TBO,TP,TC,TRATE,TF1,TF3,TF4,T128,NPORT,NTBI,NTBO


*       CHARACTER PROPERTIES            ************************************
*       CHARACTER PROPERTIES ARE LOOKED UP IN AN ARRAY OF 12 BIT
*       CHARACTER DESCRIPTOR BYTES. ON INPUT THE INDEX IS EXTERNAL ASCI,
*       ON OUTPUT IT IS INTERNAL ASCI. HENCE THE INPUT AND OUTPUT
*       PROPERTIES OF A SINGLE CHR ARE IN 2 DIFFERENT DESCRIPTOR BYTES.
*       A CHR DESCRIPTOR TELLS WHICH BREAK TABLES THE CHR BREAKS ON IN
*       THE FIRST 3 BITS. THE NEXT BIT SAYS WHETHER CHR IS ECHOABLE.
*       THERE IS AN UNUSED BIT FOLLOWED BY A 4 BIT INPUT TYPE AND A 3 BIT
*                               OUTPUT TYPE.

*  CHARACTERS ARE DIVIDED INTO CLASSES FOR INPUT AND OUTPUT PROCESSING

*       7 LEVEL OUTPUT CHARACTER TYPES (INTERNAL ASCI)
*       MOST    TYPE 1          NORMAL PROCESSING
*       102     TYPE 2          TO102 PROCESSING
*       105     TYPE 3
*       135     TYPE 4
*       152     TYPE 5
*       155     TYPE 6

* 7 LEVEL INPUT DIVISIONS ACCORDING TO EXTERNAL ASCI CODE
*       0       TYPE 12         DISCARD
*       1-10    TYPE 1          NORMAL
*       11      TYPE 7          PROCESS TAB CHR
*       12      2               TI12 PROCESSING
*       13-14   1               NORMAL PROCESSING
*       15      3               TI15 PROCESSING
*       16-32   1               NORMAL PROCESSING
```

```
*       33-35   4               SOFT ESCAPE
*       36      1       CONTROL SHIFT N. USED TO MAP TO 04 SOMETIMES.
*       37      5               HARD ESCAPE
*       40-137  1               NORMAL PROCESSING
*       140-173 10              LOWER CASE
*       174-176 11              SOFT ESCAPE OR LOWER CASE
*       177     12              DISCARD

*       END OF CHARACTER PROPERTIES   ************************************

$THIRD DATA 12525270B  MULTIPLY A BY THIS AND A IS DIVIDED BY 3 AND
*       B0 AND B1 CONTAIN REMAINDER. IF A STARTS OUT SIGNIFICANTLY MORE
*       THAN A LEGITIMATE CHR ADRS <0 TO 140000B> THEN THE QUOTIENT AND
*       THE 2 BIT REMAINDER BECOME ERRONEOUS. B0,B1 BECOME 3
*       ABOUT 1 FOURTH OF THE TIME WHEN A CHR ADRS IS CLOBBERED. THIS
*       CAUSES CRASHES ON EXU TGC5,2 INSTRUCTIONS.

$TTYASG EQU *; RPT NPORT; 0 37777B; ENDR

$ACDATA BSS NJOb ACTIVATION DATA. SEE BRS14,BRS81, BRS164.
$TT     BSS NPORT
$TF2    BSS NPORT  TF2 NOT IN PAGE 12 SO SCHEDULAR CAN CHECK OIH FLAG
$TFREE DATA 2*285               VALUE OF FREE ENTRY IN TBI OR TBO
$TFREEC DATA -2*285-1           COMPLEMENT OF TFREE
$TPORT 0

$TCIX   STA SS01; STB SS02; STX SS03
        LDA* 0; STA* 0
        BRM MPTYM
        LDX UTTY; STX TPORT
        BRM TUGC; BRU TUDMS; BRM TUE
        BRM* TGCINC
        LDX SS03; STA* 0; BRM NPOPXB
TCICIO  BRM TUGC; BRU TUDMS; BRM TUE; STA SS01
TCI1    BRM* TGCINC; BRM NPOPX

$TCIT   BRM MPTYM; LDX UTTY; STX TPORT; BRM TUGC; BRU TUDMS
        XMA SS01; SKE SS01; BRM NPOPX; MIN 0; BRM TUE; BRU TCI1
```

```
TUCHR  0
TUCHRF 0
TUECH8 0

TUGC   0
       LDX TP8RT
TUGC1  LDB TF1,2; SKB =G8BY+IIHY+X8FFY; BRU TUGC3
TUGC2  BRM TGC; BRR TUGC; STA TUCHR; LDB TF1,2; SKB =8IY; BRU TUN8
       ETR K177; RCY 1; XXA; LDX T128,2; XXA
       SKB X4; RCY 12; STA TUCHRF
       LDB TUCHR; DIR; BRM TQD; EIR; STA TUECH8
       LDA TUCHR; LDA TUCHRF; RCY 3; ETR K17
       ADD =2300004B+*; STA *+2; LDA TUCHR; EXU *
       BRM M8NCR
       BRU TUN; BRU TUN; BRU TUN; BRU TUN
       BRU TUN; BRU TU36; BRU TUN; BRU TULC
       BRU TUN; BRU TUIGNR
TUGC3  SKB =G8BY; BRU TUGC4 (CHR G8BBLER RECEIVED 8R BUFFER ZAPPED BY TWC);
       DIR; LDA TC,2; SKG K40; BRM TBACKP; EIR; BRU TUGC2
TUGC4  LDA =-G8BY; ADM TF1,2; LDB K1B4; BRM TIIR; BRU TUGC1

TUDMS  LDX TP8RT; LDA XX; STA TT,2
       LDB TF1,2; SKB =DIY; BRU *+3; LDA =DIY; ADM TF1,2
       SKB =GREENY; BRU *+2 (RESET GREEN BALL FLAG, RETURN GREEN BALL);
       BRU *+5; LDA =-GREENY; ADM TF1,2; LDA FIVE; BRM TR8M
       LDA =YELY; DIR; E8R TF2,2; SKA =YELY; BRU TUDMS1; STA TF2,2
       C8PY XA,8; LCY 8; MRG =200376B; STA TPAIR2; E8R K377; BRM TR8PAIR
TUDMS1 EIR; LDB K4000; BRM TIIR
       LDA TP8RT; ADD =6B5+TT; CAB; LDX =QTI; BRU NP8PDS

TUN    SUB K40
TUN1   ETR K177
TUN2   MIN TUGC; BRR TUGC

TUN8   LCY TERMCZ; E8R TF4,2; SKA =TERMCY; BRU TUN8A
       LDA =-1-8IY; DIR
       ETR TF1,2; STA TF1,2
       LDA =-1-BRKY; ETR TF2,2; MRG BRS12B+3; STA TF2,2
       EIR
```

```
TUN8A  CLA; STA TUECH0; LDA TUCHR; BRU TUN2

TU36   LDA F0UR; BRU TUN1

TULC   LDB TF1,2; SUB  K40; SKB =LCIY; BRU TUN1; BRU TUN

TUIGNR BRM* TGCINC; BRU TUGC1

TUE    0
       LDB TF1,2; SKB =DEY; SKB =7IY; BRR TUE
       LDB TUECH0; SKB KM1; SKN TF3,2; BRR TUE
       SKN TF2,2; SKN TR0SW; BRU TVDMS
       STA TUE9; CBA
       STA TUE8; BRM TR0TY1; LDA TUE8; LRSH 8; SKE ZER0; BRU *+5
       RST =MARY,TF1
       LDA TUE9; BRR TUE
TUE9   0
TUE8   0

TIIR   0;* GENERATE S0FTWARE INTERRUPT AT 210,211 0R 212 IF ARMED.
*      210     BFR CLEARED BUT CIRCUIT STILL VALID    (1B4)
*      211     DISMISS WILL 0CCUR 0N CI0 IN 0R 0UT    (4B3)
*      212     BAD FILE NUMBER 0N CI0. C0ULD BE A ZAPPED CIRCUIT. (2B3)
       STX TIIRX
       BRM MPPACT; LDX PACPTR; CBA; BRM RIIR
       BRM MPTYM; LDX TIIRX; BRR TIIR
TIIRX  0

$TC0X  EQU *
       STA SS01; STB SS02; STX SS03
       LDB* 0   GET CHARACTER T0 BE 0UTPUT
       BRM MPTYM
       LDX UTTY; STX TP0RT; CBA; BRU TC01
TC0CHR 0

$CI0T1A CAX
$CI0T1  BRM CP0RT
       SKN CP0RT9; BRU TCICI0 (INPUT CI0); LDA SS01;*BRU TC01 (0UTPUT);
```

```
TC01    ETR K377; STA TC0CHR
        SKN TF2,2; SKN TR0SW; BRU TVDMS
        LDB TF1,2; SKB =80Y+135Y; BRU TC02
        BRM T0; BRM NP0PX
TC02    SKB =135Y; BRU TC03;* SKIP IF 8 LVL OUPUT
        BRM TR0TY1; BRM NP0PX
TC03    LDB TF4,2; SKB =135CY; BRU TC03A (COUNTING IN PROGRESS)
*       COUNTER AT 0. SET COUNTER AND OUTPUT BLANKS
        LSH 135CZ; ETR =135CY; ADM TF4,2
TC03A   LDA =135CY; SKA TF4,2; BRU TC03B
        LDA =-135Y; ADM TF1,2; BRM NP0PX
TC03B   SKN TF2,2; SKN TR0SW; BRU TVDMS
        LDA =2408; BRM TR0TY1
        LDB KM1; LSH 24+135CZ; ADM TF4,2; BRU TC03A

TVDMS   LDB K4000; BRM TIIR
        LDA =TF2; ADD TP0RT; CAB; SKN TF2,2
        LDB TR0SW1; LDX =0TI; BRU NP0PDS

$TR0SW DATA =1
$TR0SW1 1 TR0SW

T0      0
        ETR K177; RCY 1; XXA; LDX T128,2; XXA; SKB X4; RCY 12
        LDB TF1,2; ETR SEVEN; ADD =2300003B+*; STA **1; EXU *
        BRM M0NCR; BRU T0N; BRU T0102; BRU T0105
        BRU T0135; BRU T0152; BRU T0155; BRM  M0NCR

T0N     LDA K40
        ADD TC0CHR; MRG K200
        BRM TR0TY1
T0N3    LDA =MARY; C0PY AB,N; SKB TF1,2
T0N2    ADM TF1,2; BRR T0

T0102   SKB =LC0Y; BRU T0N; LDA =212B; BRM TR0TY1; BRR T0

T0105   SKB =LC0Y; BRU T0N; SKB =MARY; BRR T0
        LDA =MARY; ADM TF1,2
        LDA =215B; BRM TR0TY1; BRR T0
```

```
T0135   SKB =531Y; BRU T0N
        LDA =135Y; SKA TF1,2; BRM M0NCR; ADM TF1,2
        LDA =135CY; ETR TF4,2; CNA; ADM TF4,2
        BRU T0N3

T0152   LDA =212B; BRM TR0TY1
        LDB TF1,2; SKB =MARY; BRR T0
        LDA =215B; BRM TR0TY1; LDA K377; BRM TR0TY1
T0152A  LDB TF1,2; SKB =MARY+MARSY; BRR T0
        LDA =MARY; BRU T0N2

T0155   SKB =MARY; BRU T0155A
        LDA =215B; BRM TR0TY1
T0155A  LDA =212B; BRM TR0TY1; BRU T0152A


*       204 INTERRUPT PR0CESSING ****************************************

TQC     0
TQA     0
TQB     0
TQX     0
TQRL    0
TQRET   LDA TRCTIM; SKG REAL; BRU TRC
        LDA TQRL; MLABEL; LDA TQA; LDB TQB; LDX TQX; BRI TQ
$TQP0RT 0
$TQTYPE 0
$TQCHR  0
TQCHRF  0

$TQ     0
        MIN TQC; STA TQA; STB TQB; STX TQX
        LDA RLTYM; MLABEL; STA TQRL

TQ1     BRM TRI; XMA TQTYPE  (SAVE CHR, GET TYPE);
        ADD =2300003B+*; XMA TQTYPE; EXU* TQTYPE
        BRM M0NCR
        BRU TY1; BRU TQRET; BRM M0NCR; BRM M0NCR
```

```
        BRU TY5; BRM M0NCR; BRU TY7; BRU TY10
        BRU TY11; BRU TY12

TQD     0;* PUT ECH0 CHR IN A WITH SIGN BIT 0R 0 IF SH0ULDNT ECH0
        SKA <400; SKN TF3,2; BRU TQD1
        STB TQDCHR; RCY 3
        LDB TF1,2; SKB =AUXY; BRU TQD1
        ETR <17; ADD =2300003B+*; XMA TQDCHR; EXU* TQDCHR
        BRM M0NCR; BRR TQD; BRU TQD12; BRU TQD15
        BRM M0NCR; BRM M0NCR; BRM M0NCR; BRU TQTAB
        BRR TQD; BRR TQD; BRM M0NCR
TQD1    CLA; BRR TQD

TQDCHR  0

TQD12   LDB TF3,2; LDA =212B; SKB =ELFY; MRG =215B*4B2+377B*2B5; BRR TQD

TQD15   LDB TF3,2; LDA =215B; SKB =ECRY; MRG =212B*4B2; BRR TQD

TQTAB   LDB TF3,2; SKB =TABY; BRR TQD; BRU TQD1

TY1     LDB TF1,2; SKB =C0NY; BRU TQ1; SKA =*10B; BRU TY1B
        ADD =2300003B+*; STA *+1; EXU *
        BRU TY1E0; BRU TY1E1; BRU TY1E2; BRU TY1E3
        BRU TY1E4; BRU TY1E5; BRU TY1E6; BRU TY1E7

TY1E0   EQU *   CHARACTER PAIR. PUT B0TH IN USER BUFFER.
        CLA; BRM TWC   FIRST THE ZER0
        CLA; STA TY1E0F    (SET FLAG F0R TWC T0 INDICATE CHAR PAIR)
        BRM TRI; CXA; E0R TWX; SKA ADMSK, BRM M0NCR (WR0NG P0RT);
        LDA TQTYPE; SKE 0NE; BRM M0NCR
        LDA TQCHR; LDB TF1,2; BRU TY1B
$TY1E0F DATA -1   (CHARACTER PAIR FLAG F0R TWC R0UTINE)

TY1E1   BRM TRI; SKG =375B; BRU TQ1
        SKE <377; BRU TY1E1A
        LDB TF1,2; SKB =DIY; BRU *+5; LDA =YELY; MRG TF2,2; STA TF2,2; BRU TQ1
        C0PY XA,B; LCY 8; MRG =200376B; STA TPAIR2; E0R K377; BRM TR0PAIR
        BRU TQ1
```

```
TY1E1A LDA =-1-0RAY; ETR TF2,2; STA TF2,2; ETR K77; AXC; STA ACDATA,2
       BRU TQ1

TY1E2  BRM CLI0; SET =G0BY,TF1; BRU TQ1

TY1E3  EQU *   TYMNET REQUEST T0 ZAP A CIRCUIT
       MIN TFICTR; BRM TBRK
       LDB TF1,2; SKB =AUXY; BRU *+3; BRM TZAPR; BRU TQ1
*      ZAP AUXILIARY CIRCIUT. N0TE SIMILIARITY T0 TZAP SUBR0UTINE.
       LDA =C0NY; ADM TF1,2; LDA =JBY; ETR TF2,2; C0PY N,B; ADM TF2,2
       STB TC,2
       LDA THREE; BRM TR0M    SEND CIRCUIT ZAPPER T0 TYMNET
       BRU TQ1

TY1E4  EQU *   CHARACTERS F0LL0WING THIS 0NE ARE ECH0ED BY REM0TE
       RST =DEY,TF1; BRU TQ1

TY1E5  EQU *   GREEN BALLS ARRIVE HERE
       SKN TR0SW; BRU TQ1  IGN0RE GREEN BALLS IF RING NEAR FULL.
       SKB =DIY; BRU TY1E5A
       SET =GREENY,TF1; BRU TQ1
TY1E5A LDA TQCHR; BRM TR0M; BRU TQ1

TY1E6  LDA =-GREENY; SKB =GREENY; ADM TF1,2        RESET GREEN BALL FLAG
       SKN TR0SW; BRU TQ1  D0N'T SEND RED BALL BACK IF RING NEAR FULL
       BRU TY1E5A    RETURN RED BALL

TY1E7  EQU *   CHRS F0LL0WING THIS 0NE HAVENT BEEN ECH0ED BY REM0TE
       LDA =DEY; SKB =DIY; BRU *+4; LDA SEVEN; BRM TWC; LDA =7IY+DEY
       MRG TF1,2; STA TF1,2; BRU TQ1

TY1B   SKB =8IY; BRU TY18
       ETR K177; LRSH 1; XXA; LDX T128,2; XXA; SKB X4; RCY 12
       STA TQCHRF
       RCY 3; LDB TF1,2; ETR K17; ADD =2300003B+*; STA *+1; EXU *
       BRM M0NCR; BRU TY1N; BRU TY1N; BRU TY1N
       BRU TS0FT; BRU THARD; BRU TY36; BRU TY1N
       BRU TY1N; BRU TSS0FT; BRU TY1N
```

```
TY18    EQU *
        BRM TWC; BRM TBRK; BRU TQ1

TY1N    CBA; ETR =DEY+DIY; SKE =DEY+DIY; BRU TY1N1
        LDA TQCHRF; LDB TQCHR; BRM TQD; SKE ZERO; BRU *+2; BRU TY1N1
        SKN TF2,2; SKN TR0SW; BRU TY1N2
        STA TY1N9; BRM TR0TY1; LDA TY1N9; LRSH 8; SKE ZERO; BRU *-5
TY1N1   RST =MARY,TF1
        LDA TQCHR; BRM TWC
        LDA TQCHRF; ETR TF2,2; SKA =BRKY; BRM TBRK
        BRU TQ1
TY1N2   SET =7IY,TF1; LDA SEVEN; BRM TWC; BRU TY1N1
TY1N9   0

TSS0FT  SKB =LCIY; BRU TY1N
TS0FT   SKB =C0NY; BRU TQ1
        SKB =AUXY; BRU TY1N
        LDB TF2,2; SKB =S0FTY+HARDY+B161Y; BRU TQ1
        LDA TQP0RT; SKE DEVCH; BRU *+4; LDB KM1; SKN WBIU; STB DEVCH
        LDX TTYASG,2; LDA PIM,2; LDX TQP0RT; SKA K2B6; BRU *+2; BRM CLIB
        BRM TBRK; LDA =S0FTY; LDB =3B5
TS0FT1  ADM TF2,2; C0PY XB,E; LDA TIC1; BRM EPU; MIN ACTR
        BRU TQ1

THARD   SKB =C0NY; BRU TQ1
        SKB =AUXY; BRU TY1N
        LDB TF2,2; SKB =HARDY+B161Y+B168Y; BRU TQ1
        SKB =B113Y; BRU *+3; BRM TQ0BBLE; BRM TBRK; BRM CLIB
        LDA =HARDY; LDB K4B5; BRU TS0FT1

TY36    EQU *   C0NTR0L SHIFT N
        LDA =2048; STA TQCHR; BRU TY1N1

TY5     EQU *   RECEIVE RESP0NSE FR0M SUPERVIS0RY REQUEST HERE.
        LDB TF2,2; SKB =50Y; BRU *+2; BRU TQ1
        LCY 5AZ; ETR =5AY; ADM TF4,2; LDA =-50Y; ADM TF2,2; BRU TQ1

TY7     SET =0IHY,TF2; BRU TQ1
```

```
TY10    RST =0IHY,TF2; BRU TQ1

TY11    LDA MAXNU; SKG NU; BRU TQ1
        LDA WERIS,2; SKE KM1; BRU TQ1; MIN WERIS,2
        LDA =8IY; STA TF1,2; LDA =TERMCY; STA TF4,2
        CLA; STA TF2,2; STA TF3,2
        STA TC,2; STA TRATE,2
        LDB <185; COPY XB,E; LDA TIC1; BRM EPU; MIN ACTR
        LDX TQP0RT; BRU TY1E7 (ENTER DEFERRED ECH0 M0DE);

TY12    EQU *   TQP0RT IS NEW P0RT. A REG. AND TQCHR IS COMMAND P0RT.
        COPY AX,N; LDB TF2,2; SKB =50Y; SKG =-NP0RT; BRU TQ1
        LDB TQP0RT; LSH 24+135CZ; SUB =135CY; ADM TF4,2; BRU TQ1

TRC     EQU *
        LDA =90; ADM TRCTIM
        LDX =-NP0RT
TRC1    LDA TF2+NP0RT,2; ETR K77 J0B NUMBER
        SKG ZER0; BRU TRC2   N0T A VALID J0B
        ABC; XMA TRATE+NP0RT,2   CLEAR RATE C0UNTER
        XXB; ADM CCT,2; CBX  ADD A BUNDLE T0 HIS CHR C0UNT CHARGE
        LDB TC+NP0RT,2; SKB KM400; BRU TRC3   L0TS 0F CHARACTERS IN BFR
        LDB =RCY/3; SKG K15; LDB =-RCY/3; CBA  INCREMENT RC 0R DECR RC
        ADD TF2+NP0RT,2; E0R TF2+NP0RT,2  DEPENDING 0N CURRENT INPUT RATE
        SKA =-1-RCY; BRU TRC2  (0VERFL0WED THE 2 BIT FIELD);
        CBA; ADM TF2+NP0RT,2
TRC2    BRX TRC1
        BRU TQ1
TRC3    STX TRC3X; CXA; ADD =NP0RT; CAX
        LDB =RCY; LDA =-RCY/3; SKB TF2,2; ADM TF2,2
        BRM TBRK; LDX TRC3X; BRU TRC2
TRC3X   0
TRCTIM  0

*       END 0F 204 PR0CESSING ********************************************

*       R0UTINES THAT INTERFACE WITH SCHEDULAR ***************************
```

```
$TIP    LDX PUTTY; STX UTTY; LDX TIPIX
        EXU *+1,2; BRU TS0N; BRU TS0N; BRU TFIP; BRU TPUS; BRU TPUH

$TIC1   1 TIP
$TIPIX  0

TS0N    LDA FPLST; SKN FULST; SKG ZER0; BRM M0NCR
        LDA FULST; LDX PUTTY; ADM TF2,2; STA J0B
        XXA; XMA TTN0,2; STA FULST
* START UP EXEC FOR THIS USER
        BRM GFK; BRU TS0N2; LDA =NUMEM; MUL J0B; LSH 23; STX PUPAC
        LDX J0B; ADD TS0NC4; STA TS0NT1
        SUB =NCMEM+NUMEM; STA PMTJ0B; ADD =-20000000B+NCMEM; STA PMTP,2
        CLA; LDX PUPAC; BRM SETPAC
        LDA =NCMEM; LDX J0B; STA RL3,2; BRU NPUG0
TS0N2   BRM M0NCR
TS0NC2  DATA    011213B+NCMEM*1B6
TS0NC3  DATA    1415B4
TS0NC4  ZR0     PMT,2
TS0NT1  ZR0
TS0NC1  ZR0     TS0NI

* SET UP NEW PAC SLOT
SETPAC  ZR0; STA RL2,2; STA RL1,2
        LDB J0B; LSH 39; ADD =EXECLM; MRG X6; STA PTAB,2
        LDB TIPIX; LSH 44
        ADD =1B7+6B2; STA PIM,2
        LDA PUPAC; LDX PUTTY; STA TTYASG,2
        LDX =QTI; BRM QPUT; LDX PUPAC
        LDA TS0NC1; STA PL,2; LDA TS0NC2; STA RL1,2; LDA TS0NC3; STA RL2,2
        LDA SETPC2; STA PTEST,2; BRR SETPAC
SETPC2  13 PACOMB
*
TS0NI   MIN NU; LDX =NCMEM; LDA* PMTJ0B; MRG X2; STA* PMTJ0B
        LDX PACPTR; LDA X1; ADM PTAB,2; LDA X4; STA XPB
        LDA RRL3; ETR K77; STA RLTS
        LDA =NLQU; STA LQ
        LDA J0B; MUL =TABLEN*9/64; STB DGET    SEE GETBLK.
```

```
        LDA =7708; STA FSIZE   USED IN BRS1 T0 SET FILE SIZE QUANTUM.
        LDA SEVEN; STA NF0RK; LDA KM1; STA LMCPU
        LDX KM7; LDA Tw0; STA PX,2; ADD 0NE; BRX *-2
        LDA 0NE; STA PPB
        LDA =MBUFX; STA FBWRD
        LDX PACPTR; LDA PIM,2; ETR K4B6; LRSH 44
        LDA PIM,2; ETR =737777778; STA PIM,2; CBA; BRU ExECI,4
*
TFIP    LDX UTTY; LDA KM3; SKE WERIS,2; BRU TFIP2; MIN WERIS,2
        LDX TTYASG,2; STX TFIP3; BRM HFK; STX PUPAC
        LDX PUPAC; BRM RFK; LDA =QTI; BRM TFK
        BRM QPUT; LDX TF06; LDB TFIP1; STB PL,2
        BRU NPUG0
TFIP2   MIN RBER; BRU NPUG0
TFIP1   ZR0 0FFINT,4
TFIP3   ZR0 0

TPUS    EQU *   PU S0FT ESCAPE PR0CESSING
        BRM MPTYM; LDX UTTY
        LDA =S0FTY; SKA TF2,2; BRU *+2; BRU NPUG0; CNA; ADM TF2,2
        BRM MPPACT
        LDX TTYASG,2; LDA K2B6; BRM IIR; BRU TH1; BRU NPUG0

TPUH    EQU *   PU HARD ESCAPE PR0CESSING
        BRM MPTYM; LDX UTTY
        LDA =HARDY; SKA TF2,2; BRU *+2; BRU NPUG0; CNA; ADM TF2,2
        BRM CLIB; BRM MPPACT; LDX TTYASG,2; BRM HFK
TH1     STX PUPAC; CLB; SKN XPB; BRU *+2; STB UCIN
        BRM RFK; LDA =QTI; BRM TFK; BRM QPUT
        BRU NPUG0

*       TYMNET BRS S    ************************************************

$BRS11  EQU *   CLEAR INPUT BFR
        BRM GATT; BRM CLIB; BRM NP0Px

$BRS12  EQU *   SET ECH0 TABLE
        BRM GATT
        LDA SS01; SKA =-1-8IY-80Y-LCIY-LC0Y-XY-1600777B
```

```
      BRM TRAP BIT OTHER THAN 8IY,80Y,LCIY,LC0Y,TABY,ELFY, OR ECRY
      LDB =ECH0Y; ETR =8IY+777B;; SKE THREE; SKA =8IY; CLB; STB T ECH0 BIT
      LDA SS01; LCY 4; ETR =ELFY+ECRY; EOR =ELFY+ECRY
      MRG T; EOR TF3,2; ETR =ECH0Y+ELFY+ECRY; EOR TF3,2
      BRM TF3XMA     OUTPUT NEW TERMINAL CHARACTERISTICS IF ANY
      LDA TF4,2; ETR =TERMCY; CNA; ADM TF4,2
      LDA SS01; IF TERMCZ; LCY TERMCZ; ENDF; ETR =TERMCY; ADM TF4,2
      LDA SS01; ETR =8IY+80Y+LCIY+LC0Y+3   ONLY 3 IS IN EXPONENT FIELD
      COPY AX,XB,A,E    CLEAR ECH0 TABLE FROM A, CAX ETC.
      LDX BRS12B,2; XXB
      DIR; EOR TF1,2; ETR =8IY+80Y; EOR TF1,2; STA TF1,2
      CBA; EOR TF2,2; ETR =BRKY; EOR TF2,2; STA TF2,2
      BRU BS153A
BRS12B DATA BRKY/7*4,BRKY/7*2,BRKY/7,BRKY/7*4     BREAK TABLE BITS

$BRS13 EQU *  SKIP IF INPUT BFR EMPTY OR SKIP IF OUTPUT BACKPRESSURE.
      BRM GATT
      BRM TUGC; BRU NP0PX0; BRM NP0PX

$BRS14 EQU *   DISMISS UNTIL TERMINAL OUTPUT COMPLETED
      BRM GATT; BRM TYEL; BRM MPPACT; LDA XX; BRU BRS81

$BRS24 EQU *   SET LCI AND LC0 AND X-ON, X-OFF MODES
      ETR =-1-777B; SKA =-1-436B5-LCIY-LC0Y-XY; BRM TRAP; STA T
      BRM GATT; LDA TF1,2; ETR =LCIY+LC0Y+XY; CNA; ADD T; ADM TF1,2
      BRM NP0PX


$BRS29 EQU *   CLEAR OUTPUT BFR
      CXA; SKE KM1; BRU *+2; LDX ONE
      BRM CP0RT; SKN CP0RT9; BRM TRAP
      SKN TR0SW; BRU TVDMS
      BRM TG0BBLE
      BRM NP0PX

* BRS 32   CLEAN UP -2 AND -3 FOR MOST CASES.
* INPUT: X=CHANNEL NUMBER
$CLM2  STX CLM2A; LDA KM1; SKG WERIS,2; BRU P0PX
      LDX TTYASG,2; LDA PTEST,2; STA CLM2C
```

```
        LDB CLM2B; ETR =7765; SKE =765; STB PTEST,2; BRU P0PX
CLM2A   ZR0
CLM2B   BRU PACDMB
CLM2C   ZR0

$BRS40  EQU *   READ ECH0 TABLE AND ECI,ELF,ECR,8I,80,X (SEE MDBG FLAGS)
        BRM GATT
        LDA TF1,2; ETR =8IY+80Y+XY  CURRENT 8I,80,X MODES; STA SS01
        LDA TF3,2; ETR =TA0Y+ECRY+ELFY; EOR =ECRY+ELFY; LRSH 4; ADM SS01
        LDA =LCIY+LC0Y; ETR TF1,2; ADM SS01
        SKN TF1,2; BRU B40B (NOT 8 LVL);* 8 LVL
        LDA =TERMCY; ETR TF4,2; IF TERMCZ; LRSH TERMCZ; ENDF
B40A    ADM SS01; BRM NP0PX
B40B    LDA =BRKY; ETR TF2,2; SKN TF3,2; CLA
        LDX =20-BRKZ; NOD 23-BRKZ; CXA; CNA (0,1,2 OR 3,BELEIVE IT OR NOT)
        BRU B40A

$BRS85  EQU *   SET 8 LEVEL OUTPUT MODE
        BRM GATT; LDA =80Y; SKA TF1,2; BRM NP0PX; BRU BRS86A

$BRS86  EQU *   CLEAR 8 LEVEL OUTPUT MODE
        BRM GATT; LDA =80Y; COPY A0,N; SKB TF1,2
BRS86A  ADM TF1,2; BRM NP0PX

$BRS87  EQU *   CONVERT PORT NUMBER IN X TO JOB NUMBER IN A
        LDA TF2,2; ETR K77; BRM NP0PXB

$BRS112 EQU *  SUICIDE BRS. REMOVE J0B FROM SYSTEM TABLES.
        LDX J0B; LDA X4; SKR NU; SKA TTN0,2; BRM M0NCR
        COPY XA,B; STB PMTP,2
        XMA FULST; XMA TTN0,2; ETR K77
        SKE DEVCH; BRU *+3; LDB KM1; STB DEVCH
        CAX; LDA ADMSK; XMA TTYA$G,2
        ADD =PPTR; XMA FPLST; STA* FPLST

        LDB K77; LDX PUBPTR     CLEAR USER FROM P.U.
BS112A  CXA; SKE =PUBPTR; BRU *+2; BRU BS112B
        LDA 3,2; SKM UTTY; BRU *+4; LDA PUDEAD; STA 1,2; MIN EXTPU
        LDX 0,2; BRU BS112A
```

```
BS112B COPY A,X; STA RELPG8
       EAX* PMTJOB; STX RELPG3 (SAME AS LDA PMTJOB,ETR ADMSK,STA RELPG3)
       LDX PACPTR; LDA RL1,2; LDB RL2,2; CLX; BRM RELMEM
       LDA PMTJOB; ADD =NCMEM-1; STA BS112D; ADD =NUMEM; STA BS112E
BS112C CAX; DIR; BRM RES; BRM MONCR; EIR
       SKR BS112E; LDA BS112E; SKE BS112D; BRU BS112C
       CLA; LDX JOB; SKE AUNN,2; MIN NB112; STA AUNN,2
       LDA KM1; LDX UTTY; STA WERIS,2
       BRM MPTYM; BRM TZAP; BRM TZAP1; BRM MPPACT
       BRU PACGOA
BS112D 0
BS112E 0

$BRS113 EQU *  INHIBIT SOFT AND HARD ESSCAPES FOR USER
       LDX UTTY; SET =B113Y,TF2; BRU POPX

$BRS114 EQU *  UNDO BRS 113
       LDX UTTY; LDA =-B113Y; LDB TF2,2; SKB =B113Y; ADM TF2,2
       SKB =SOFTY+HARDY; BRU BRS45  (DISMISS LONG ENOUGH FOR PU TO RUN);
       BRU POPX

$BRS129 EQU *  HANG PORT IN A IF IT IS A COMMAND PORT.
       SKG =NPORT-1; SKA X4; BRM TRAP; SKG ZERO; BRM NPOPX; CAX
B129A  DIR; SKN WERIS,2; BRM TZAPR; BRU BS153A

$BRS132 EQU *  READ TERMINAL CHARACTERISTICS OUT OF TF3
       BRM GATT; LDA TF3,2; BRM NPOPXB

$CRSW  BRM NPOPX   BRS 134. IGNORE CR/LF SWITCH

$BRS137 EQU *  LIST PORT NUMBERS IN HIS MEMORY. X POINTS TO HEAD OF LIST
       COPY XA,B; ETR ADMSK; ADD =4B7+1; STA T POINTS TO 2ND ENTRY
       STB 0,6    PUT HIS COMMAND PORT AT HEAD OF LIST
       LDX =-NPORT; LDB =AUXY   APPEND AUXILIARY PORTS TO LIST
B137A  LDA TF2+NPORT,2; ETR K77; SKB TF1+NPORT,2; SKE JOB
       BRU B137B     NOT ONE OF HIS AUXILAIRY PORTS
       CXA     USER PORT NUMBER = MONITOR PORT INDEX TIMES 2 PLUS 10B.
       ADD =NPORT+4  (MONITOR INDEX+4); STA* T; ADM* T; MIN T
```

```
B137B  BRX B137A
       LDA T; SUB SS03; ETR ADMSK      LENGTH OF LIST
       BRM NP0PXB

$BRS138 EQU *  INITIATE SUPERVISORY REQUEST
       LDX UTTY; SKN TF2,2; SKN TR0SW; BRU TVDMS; SET =50Y,TF2
       LDA =B138A; XMA 0; STA SBRSRT; BRM MPTYM
       LDA =135CY+5AY; ETR TF4,2; CNA; ADD =135CY; ADM TF4,2
       LDA =-1-135Y; DIR
       ETR TF1,2; STA TF1,2
       COPY XA,B; LSH 8; MRG =5*2B5; BRM TR0 (SEND A TYPE 5); EIR
       LDA =32B5+TF2; ADD UTTY; CAB; LDX =QTI
       BRU NP0PDS
B138A  STB SS02; STX SS03
       LDA SBRSRT; STA 0
       BRM MPTYM; LDX UTTY
       LDA TF4,2; ETR =5AY; LRSH 5AZ; STA SS01   ERROR NUMBER FROM SUP
       LDA TF4,2; ETR =135CY;* A=135CY IF TYPE 12 DIDNT PRECEDE TYPE 5
       LRSH 135CZ; COPY AX,AB,N; SKG =-NP0RT; BRM NP0PX (NO TYPE 12);
       LDA TF2,2; SKA K77; BRM NP0PX (SUP ALLOCATED A BUSY PORT);
       LSH 25; ADD EIGHT; STA SS03   USERS X REG IS NEW CIRCUIT NUMBER
       STB TF3,2; LDA =8IY+AUXY; STA TF1,2; LDA J0B; STA TF2,2
       LDA =TERMCY; STA TF4,2
       BRM NP0PX

$BRS139 EQU *
       BRM GATT; LDB TF1,2; SKB =AUXY; BRU *+2; BRM TRAP
       BRM TZAP; BRM NP0PX

$BRS152 EQU *    RETURN TTY INPUT BUFFER CHARACTER COUNT IN A
       BRM GATT; LDA TC,2; BRM NP0PXB

$BRS153 EQU *    SET CELL 211 WHICH POINTS TO SUP PAGE 0
       LDA RRL1; LRSH 18; ETR K37; AXC
       SKN SVST; BRM TRAP; SKE 211B; BRM TRAP; STX 211B
       LDA RMT,2; DIR; BRM 8
BS153A EIR; BRM NP0PX

$BRS154 EQU *   RESET CELL 211
```

```
        CLA; SKE 211B; SKN SVST; BRM TRAP
        XMA 211B; CAX; LDA RMT,2
        DIR; BRM REL; BRM DRAG0
        BRU BS153A

$BRS155 EQU * (IF X IS INPUT PORT THEN SKIP IF NO CHRS RECEIVED, IF
        BRM GATT X IS OUTPUT PORT THEN SKIP IF TCO WOULD CAUSE A DISMISS)
        SKN CPORT9; BRU *+5; SKN TF2,2; SKN TR0SW; BRU NP0PX0; BRM NP0PX
        CLA; SKE TC,2; BRM NP0PX; BRU NP0PX0

$BRS156 EQU * ZAP ALL AUXILIARY CIRCUITS FOR THIS JOB.
        BRM TZAP1; BRM NP0PX

$BRS159 EQU * SET UP/SHUT STATE.
        STA UPSHUT; MIN UPSHUT (UPSHUT = 1 FOR SHUT, 2 FOR UP)
        MRG =6*285; DIR; BRM TR0 (SEND TYPE 6 TO BASE); BRU BS153A


$BRS160 EQU *       BRS 160. CHANGE TERMINAL CHARACTERISTICS.
        BRM GATT; LDA SS01; BRM TF3XMA; BRM NP0PX

$BRS161 BRM GATT; SET =B161Y,TF2; BRM NP0PX   COMPLETELY IGNORE ESCAPES
$BRS162 BRM GATT; RST =B161Y,TF2; BRM NP0PX   UNDO BRS 161

$BRS163 LDX KM4,3; LDA 0,2; STA 0,6; BRX *-2; BRM NP0PX  NET TO USER

$BRS164 EQU * ACTIVATE HIM WHEN YELLOW BALL ARRIVES OR TIME RUNS OUT.
        BRM GATT; BRM TYEL; BRM MPPACT; LDA SS01; BRU BRS81

$BRS166 EQU * HANG PORT 0
        COPY A,X; BRU B129A

$BRS168 BRM GATT; SET =B168Y,TF2; BRM NP0PX   DISARM HARD ESCAPES
$BRS169 BRM GATT; RST =B168Y,TF2; BRM NP0PX

$BRS172 EQU * SEND OUT CONTROL INFORMATION TO REMOTE. SEND A16-A23.
        BRM GATT; LDA TF2,2; SKA =DIALY; BRU *+2; BRM TRAP
        SKN TF2,2; SKN TR0SW; BRU TVDMS
        LDA SS01; RCY 8; CXA; LCY 8; MRG K285   (A TYPE 1 CHARACTER)
```

```
        DIR
        STA TPAIR2; ETR KM400; MRG ONE (TYPE 1, CHARACTER 1); BRM TRØPAIR
        BRU BS153A

$BRS177 EQU *   (ALLØW REMØTE DIALØUT ØN THIS PØRT)
        BRM GATT; LDA =DIALY; MRG TF2,2; STA TF2,2; BRM NPØPX

$BRS178 EQU *   (TURN ØFF STATUS FØR REMØTE DIALØUT)
        BRM GATT; LDA TF2,2; ETR =EDIALY; STA TF2,2; BRM NPØPX

*       END ØF TYMNET BRS S     ****************************************

*       LØW LEVEL RØUTINES  ********************************************

TBRK    0
        LDA TF2,2; ETR =RCY; LRSH RC7
        XXA; LDX TDELAY,2; XXA
        LDB TF1,2; SKE ZERØ; BRU *+3; SKB =DIY; MIN ACTR; ADD REAL
        STA TT,2
        LDA =-DIY; SKB =DIY; ADM TF1,2
        CBA; ETR =DIY+DEY; SKE =DIY+DEY; BRR TBRK
        LDA =7IY; ADM TF1,2; LDA SEVEN; BRM TWC; BRR TBRK

TDELAY DATA 0,0,90,180

GATT    0;*     VALIDATE PØRT NUMBER IN X. SET UP RRL3 AND TPØRT
        CXA; SKG KM1; CLX           =1 MEANS USERS CØMMAND PØRT
        BRM CPØRT; BRR GATT

CPØRT   0
        CXA; ETR ADMSK; LRSH 1; STB CPØRT9
        LDX UTTY; SKG ZERØ; BRU CPØRT2
        SUB FØUR; SKG =NPØRT-1; SKA X4; BRM TRAP; CAX
CPØRT2  STX TPØRT
        LDA JØB; LDB K77; SKM TF2,2; BRU *+3; BRM MPTYM; BRR CPØRT
        BRM MPPACT; LDX PACPTR; LDA K2000; BRM RIIR; BRM TRAP
CPØRT9  0;* NEGATIVE IF LAST PØRT AGRGUMENT WAS FØR ØUTPUT. PØSITIVE FØR INPUT.

TGØBBLE 0;*   SEND A CHARACTER GØBBLER ØUT TØ TYMNET
```

```
        DIR
        LDA =-1-YELY; ETR TF2,2; STA TF2,2    RST YELLOW BALL FLAG
        LDA =-1-135Y-MARY; ETR TF1,2; STA TF1,2 STOP BLNKS,RST MARGN FLG
        LDA TWO; BRM TROM   SEND CHR GOBBLE AND EIR
        BRR TGOBBLE

TZAPR   0;* TZAP REQUEST. SET UP PU TASK TO ZAP A PORT.
        LDA WERIS,2; SKG KM2; BRR TZAPR
        LDB KM3; SKG KM1; LDB KM4; STB WERIS,2
        LDB K235; COPY XB,E; LDA TIC1; BRM EPU; MIN ACTR; BRR TZAPR

TZAP1   0;*     ZAP ALL AUXILIARY CIRCUITS FOR THIS JOB.
        LDX =NPORT-1
TZAP2   LDA TF2,2; ETR K77
        SKE JOB; BRU *+4; LDB TF1,2; SKB =AUXY; BRM TZAP
        CXA; EAX -1,2; SKA ADMSK; BRU TZAP2
        BRR TZAP1

TZAP    0;*     ZAP A PORT.
        LDA =CONY; ADM TF1,2; LDA =JBY; ETR TF2,2; COPY N,B; ADM TF2,2
        STB TC,2
        LDA THREE; BRM TROM    SEND CIRCOUT CIRCUIT ZAPPER TO TYMNET
        BRR TZAP

CLIB    0
        DIR; BRM TBACKP
        LDA =-1-MARY-GOBY; ETR TF1,2; STA TF1,2
        ABC; STA TC,2
        LDA SEVEN; SKB =7IY; BRM TWC (PUT BACK 7 IN INPUT BFR);
        EIR; BRR CLIB

TBACKP  0
        COPY XA,B; LCY 8; MRG =200021B; LDB TF1,2; SKB =X0FFY; BRM TR0
        LDA =-1-IIHY-X0FFY; ETR TF1,2; XMA TF1,2
        SKA =IIHY; BRU *+2; BRR TBACKP
        COPY XA,B; LCY 8; MRG =10B*2B5; BRM TR0; BRR TBACKP

TYEL    0;* SEND YELLOW BALL TO TYMNET.
        COPY XA,B; LCY 8; MRG =200376B; LDB TF1,2
```

```
        DIR; STA TPAIR2; SKB =AUXY; MIN TPAIR2; EOR K377; BRM TROPAIR
        LDA =ORAY; MRG TF2,2; STA TF2,2
        EIR; BRR TYEL

TF3XMA  0;*     SET NEW TERMINAL CHARACTERISTICS IN TF3 OR DISMISS
        SKN TF2,2; SKN TROSW; BRU TVDMS
        STA T     NEW VALUE FOR TF3
        LDB FIVE; STB T2   COUNTER FOR TERM. CHR. FIELDS  (0-5)
        CAB; XMA TF3,2
        EOR T    USE EOR TO NOTICE CHANGED 4 BIT FIELDS
TF3X1   SKA =7486; BRU *+2 (OUTPUT A 4 BIT FIELD TO TYMNET); BRU TF3X2
        STA T1; STB T
        LDA FIVE; SUB T2; RCY 4; CXA; LCY 8
        MRG <285; DIR; STA TPAIR2; ETR KM400; MRG ONE
        BRM TROPAIR; EIR   OUTPUT A 1 FOLLOWED BY TERMINAL CHARACTERISTIC
        LDA T1; LDB T
TF3X2   LCY 4; SKR T2; BRU TF3X1
        BRR TF3XMA
        END
```

```
PAC     IDENT                                          6/21/72  14:57     PAGE   1

        LISTM

* ENTRY P0INTS

        ENTRY   PACSW,RAER3,SWIN,SWLS,SWT3,SWPMT,SWSMT
        ENTRY   SRT44E
        ENTRY   RAER0R,QTIG0,PACSWQ,PG44,P0PR4,P0PR5
        ENTRY   RLWB
        ENTRY   RAER2,RAJ0B2,CMPST,NP0PDS
        ENTRY   CMRRL1,CMRRL2,CMRRL3,PACQUE,PACJ0B,PAC0MP,RAJ0B1
        ENTRY   RAJ0B,RAJ0B3,RELPG,SWAPR,SWAPI,SWJ0B
        ENTRY   RELMEM,RELPG3,RELPG8,SETSWP,P0PX
        ENTRY   CLINTC,P0PR2,P0PR,PACDMS
        ENTRY   MPPACT,RLTS,PUPACP
        ENTRY PUDEAD,PWNI
        ENTRY   PUTIM,NP0PX,NXP0P,NP0PXB
        ENTRY   MPDSC,MPWB,NTERM,ALTERM

        ENTRY   SRT,SRTE
        ENTRY   PACDMB,P0PX,XP0P,PACWE,P0PDMS,SETSET
        ENTRY   PACG0,PEST,P0PINT,P0PST,PUG0,NPUG0,QQEDMS,PACG01
        ENTRY   FPULST,PUEPTR,PUBPTR,PUPAC
        ENTRY   JPRL,PTRL,LABEL,CHREL
        ENTRY   RRL1,RRL2,RRL3,TIME,TTIME,TIME1,UN0SV,PUCTR
        ENTRY   RREAL,EXDMS,DMS
        ENTRY   EPU,FPLST,WSCH,QPUT,IIR
        ENTRY   CLINT,PWFI,TRAPT,M0NCR,CRASH,ST0P
        ENTRY   PPTEST,PTESTS,PACLVL

PACDMB  ZR0  *
DMS111  23B  PACDMB    BRS 111 DISMISS
PACG2   16B  PACDMB   QUANTUM 0VERFL0W
PACG3   17B  PACDMB   BRS 45
PACG4   35B  PACDMB    TRAPT

*
*       SCHEDULER
```

```
* ACTIVATION CONDITIONS
*
* 0       WORD GREATER THAN 0
* 1       WORD LE 0. DMS UNTIL FREE PAGES FOR BIO (DLOCK NEG.)
* 2       WORD GE 0. DISC FILES.
* 3       WORD LE TTY EARLY WARNING. (OUTPUT BUFFER)
* 4       SPECIAL TEST. WORD POINTS TO A SPECIAL ACTIVATION TEST
*         ROUTINE. TAPE READY TEST.
* 5       INTERRUPT OCCURRED. ADDRESS HAS A BIT SET CORRESPONDING TO
*         THE INTERRUPT WHICH OCCURRED.
* 6       WORD LT REAL. (INPUT BUFFER)
* 7       SPECIAL. FORK NOT ON QUEUES.
*             ADDRESS
*                 0  DEAD
*                 1  RUNNING
*                 2  BRS 31. DMS UNTIL LOWER FORK PANICS.
*                 3  UNCORRECTABLE RAD ERRORS.
*                 4  EXEC BRS.
*                 5  BRS 109. DMS UNTIL OFF INTERRUPT PROCESSING.
*                 6  BRS 9.
* 10      WORD GT 0. BRS 124,125.
* 11      WORD (AND) 267=0. DISC BUFFER IN USE. FILES.
* 12      WORD LT 0. FILES. FW LT 0.
* 13      WORD GT 0. PLEASE LOG IN.
* 14      WORD GT 0. EXEC BRS.
* 15      WORD GT 0. BRS 9.
* 16      WORD GT 0. QUANTUM OVERFLOW.
* 17      WORD GT 0. BRS 45.
* 20      WORD GT 0. EXEC LEVEL PANIC.
* 21      WORD GT 0. ORDINARY PANIC.
* 22      WORD GE 0. DISC ERRORS OF PU 3 SEC W BUFFER TIME OUT.
*         GOES TO TRAP.
* 23      WORD GT 0. BRS 111.
* 24      WORD GT 0. BRS 44.
* 25      NOT USED. SAVE FOR BRS 44.
* 26      NOT USED. SAVE FOR RFK.
* 27      NOT USED. SAVE FOR COMPUTE JOB.
* 30      NOT USED. SAVE FOR PU.
* 31      WORD GT 0. BRS 81. DISMISSED FOR LACK OF SPACE.
```

```
* 32      BRS 138 DISMISS.
* 33      WORD LT 0. NEW TAPE AND PRINTER.
* 34      WORD LE 0. NEW TAPE.
* 35      WORD GT 0. TRAPT.
*
*
*
* *** LOG SYMBOLS
* 20 FKGB2,FKGB,PACPTR
* 21 PMTA,PMTA
* 22 POPD9A
* 23 PURBT,PUPAC,PUTTY
* 24 PGET,PGET,UTTY,JOB
* 27 CDMS1,JOB    DISMISS BRS 122,123 IF DRQ FULL
* 30 BRS153,C,LOGX=PAGE
* 32 PACGOS,JOB    ACTIVATE A FORK.
* 33 FCJ,SWJOB    FETCH COMPUTE JOB.
* 34 PEST3,SWJOB     SWAP OTI JOB.
* 35 RELPG,JOB,RELPG    RELEASE PAGES.
* 36 QPUT,QUEUE
* 37 QPUT,QPUT
* 40 QGET,QUEUE-PNEXT
* 41 QGET,QGET
* 42 POPR2,BSX    DISMISS WITHOUT RELEASE
* 43
* 44
* 45 QTPUT,QTPUT
* 46 RELMEM,RELMEM    RELEASE MEMORY
* 47 SWAP,SWAP
* 50 UBRSET,JOB
* 51 BRSRET,JOB
* 52 STRLO,0
* 53
* 54
* 55
* 56 PTRAP,PTRAP
* 57 FKST,XPB    BRS 9
* 60 SWREL,SWREL
*
```

```
*
* BRS 72   EXEC DISMISS
* INPUT: SS03=QUEUE NUMBER.  X=PACPTR.  B=PTEST.
EXDMS  EXU EX; LDX SS03
       RCH 220B (CXA,CBX); MUL THREE; LSH 23; ADD =QO
       RCH 440B (CXB,CAX); MIN 0; BRU P0PDMS
* DISMISS UNTIL INTERRUPT
* BRS 109
DMS    LDB =7000005B; STB PACDMS; BRU P0PST
NP0PDS BRM MPPACT; BRU P0PDMS
* P0P EXIT WITHOUT PAGE RELEASE
P0PR   MIN 0; LDB DMS111
P0PR2  DIR; LOGG 42B,LSX; EIR
       STB PACDMS; LDA PACPTR
       LDX =QQE; SKN PACJOB; BRU P0PR5; SKN TTIME; BRU *+2
       BRU P0PR4; STX PACQUE; BRM QTPUT; BRU P0PR6
P0PR5  SKN TTIME; BRU *+2; BRU P0PR4; LDX PACSWQ
       XXA; ADD =PNEXT; XXA
P0PR4  STX PACQUE; BRM QPUT
P0PR6  LDB PACDMS; LDX PACPTR; STB PTEST,2; STB PTESTS; BRU P0PD9
* RELABEL PAC TABLES BEFORE EXIT
NXP0P  STA SS01; BRM MPPACT; LDA SS01; BRU XP0P
NP0PXB 0; STA SS01; BRM NP0PX
$NP0PXQ BRM MPPACT
$P0PXQ MIN 0; BRU P0Px
NP0PX  0; BRM MPPACT
*SYSP0P EXIT TO RESTORE CENTRAL REGISTERS
P0PX   LDA SS01
       IF      DDIFP
$BP0PX BSS     0
       ENDF
$P0PXB  LDB SS02
       IF      DDIFP
$XP0PX BSS     0
       ENDF
       LDX     SS03
XP0P   BRK 0
$PACQM1 MIN 0
PACQE  LDB PACQ2
```

```
QQEDMS LDX =QQE; SKN PACJ0B; BRU P0PDMS (N0T C0MPUTE J0B)
       SKN TTIME; BRU *+2; BRU P0PDMS; LDX =QQC
*P0P DISMISS ENTRY P0INT
P0PDMS STB PACDMS; SKN PACJ0B; BRU PDMS2 (N0T C0MPUTE J0B)
       CXA; SKE =QI0; BRU PDMS2
       IF DISC=3; LDA KM4; ELSE; LDA KM8; ENDF
       ADM TTIME; SKN TTIME; LDX =QQC
PDMS2  C0PY XA,XB; SKN SVST; BRU PDMS4; SKE =QQC; BRU *+2
       BRU PDMS4; SKG =QTI; BRU PDMS4; LDB =QSQ
PDMS4  EQU *
PDMS3  CBX; LDA PACPTR; STX PACQUE
       BRM QPUT (PUT PAC ON QUEUE); BRU P0PST2
P0PST  CLA; STA PACQUE
P0PST2 LDB PACDMS (PICK UP DISMISS C0ND)
       LDX PACPTR, STB PTEST,2; STB PTESTS
* RELEASE PAGES OF TERMINATING F0RK
       LDA PACQUE; SKE =QQC; BRU *+2; BRU P0PD9
P0PINT LDA X4; LDB PACPTR; LDX J0B; BRM RELPG
* CHECK F0R RELEASED PAGES
P0PD9  EQU *
       IF ST
       DIR; SKN STSW; BRU P0PD9A
       LDA =RSR; BRM RLABEL; BRM GSS
       2 =185+11
       3 J0B
       4 UTTY
       5 PAGE#
       2 PPTEST
       2 PTESTS
       2* YRL1
       2* YRL2
       2 XPB
       7 UNSSV
       1 REAL
P0PD9A EIR
       ENDF

       LDX PACPTR; LDA 0; ETR =50037777B; XMA PL,2; ETR =7700000B
       ADM PL,2 (SAVE START L0C)
```

```
        DIR; LOGG 22B,PACPTR; LOGG 22B,J0B; EIR
*SAVE CENTRAL REGISTERS
        SKN XPB; BRU PACG01
        LDX XPB; LDA SS02; STA PB,2; LDA SS01; STA PA,2
        LDA SS03; STA PX,2
        IF      DDIFP
        IF      FPM
        HFST    FAC1,FAC2
        ENDF
        SKN     SFP0VI
        BRU     PAC1C
        COPY    A                   FLOATING OVERFLOW YET TO BE REPORTED
        STA     SFP0VI
        LDA PACPTR; SKE =QC; BRU *+2; BRM RCJ
        LDX PACPTR; LDA K2B4; BRM MONCR; NOP
PAC1C   BSS     0
        ENDF
*SET UP PROPER QUANTUM REMAINING
        LDA TTIME; SKG KM1; LDA =NLQU; STA LQ
$PACG0A SKN PACJ0B; BRU PACG01; LDA TTIME; BRM SJB
PACG01 LDA =SETTB; STA TJ0B (SET ACCOUNT TO SYSTEM)
        CLA; STA PACJ0B; STA SCHLP

        ARMI AIRWD;*  THIS SHOULDNT HAVE TO BE EXECUTED EVERY TIME THRU
*                               THE SCHEDULAR

MUX15   LDA FG940A; STA UPFL; LDA FG940B; STA UPFL+1
        IF DISC=3; LDA =2B5+1; STA DRQU; ENDF   UP FLAG FOR DISC 620

* TEST FOR PU ACTIVATION. DON'T RUN IF QTI JOB IS COMING IN.
        SKN PACSW; BRU *+2; BRU PACG0; LDA RAJ0B2; SKE ZERO
        BRU PACG0; LDA PUTIME; SKG REAL
        BRU PU3SEC; LDA PUCTR; SKG ZERO; BRU PACG0
        SKN PURUN; BRU PUG04
        SKN ACTR; BRU *+2; BRU PACG0; SKE PUCTR1
        BRU PUG04; BRU PACG0

$AIRWD EQU *
      IF DISC=3; DATA 61463; ELSE PH2; DATA 604B3; ELSE; DATA 77B4; ENDF
```

```
* START SCHEDULER
PACG0  LDA <M1; STA ONTERM; STA PURUN
       SKN M48K; BRU *+2; BRU PACG02
       SKN JQC; BRU FCJ (FETCH COMPUTE J0B)
PACG02 SKN PACSW; BRU FQJ (FETCH QTI J0B)
* QTI J0B EITHER IN OR COMING IN
       CLA
       DIR; SKN SCHLP; BRU PACG21; SKE RAJ0B1; BRU PACG22
       BRU PACG07
PACG21 SKN RAJ0B1; BRU *+2; BRU PACG07; MIN RAJ0B1
PACG22 EIR; BRU PACWT   (RUN COMPUTE J0B OR WAIT)
PACG07 EIR
* QTI J0B READY TO RUN
PACG04 LDX PACSW; STX PACPTR; SKN RAER1; BRU N0G0
       BRM SETSWP; BRM SWAPI
PACG06 BRM LABEL; CLA; STA PACSW; STA PACJ0B; LDX PACPTR
       LDA FQJ5; STA PTEST,2
       LDA PNEXT,2; LDB PACSWQ; LDX PACSWP; BRM QGET
       LDA PACQT; STA T; SKR RAJ0B1; BRM MONCR; LDA =NSQU; STA TIME
* ACTIVATE A FORK
PACG05 LDX <M7; LDA PACPTR; ADD =PL; STA YPIM+1,2
       ADD ONE; BRX *-2; LDX PACPTR; BRM PGET
       DIR; LOGG 32B,J0B; LOGG 32B,PACPTR; EIR
       LDA =700001B; XMA PTEST,2; STA PPTEST
       LDA =NLQU; SKN XPB; BRU *+2; LDA LQ; STA TTIME
       DIR; SKG TIME; STA TIME; EIR
       LDA REAL; STA TIME1

*CHECK FOR INTERRUPT AND SET UP START LOC (0)
       BRM STSTAT
       LDA X1; SKA PIM,2; BRU PACG08
       SKN LMCPU; BRU *+2; BRU PACG08
       BRM CTRU; LDX PACPTR; SKG LMCPU; BRU PACG08; BRU PACG09
PACG08 LDA T; LRSH 15; ADD =ACTLST; XMA T; BRU* T
PACSRT LDA PL,2; STA 0
PACOVF ROV; LCY 1; LSH 1 (SET UP OVERFLOW)
       CLEAR
       IF      DDIFP
```

```
        STA     UMTF
        ENDF
        SKN XPB; BRU PAC2B; LDA UUN0; STA UN0SV
        LDX     XPB
        IF      0DIFP
        IF      FPH
        HFLD    FAC1,FAC2
        ENDF
        ENDF
        LDA PA,2; LDB PB,2; LDX PX,2
PAC2B   MIN* TJ0B; BRU* 0
*
* RETURN TO EXEC FOR CPU TIME OVERRUN
PAC009  LDB =M1; STB LMCPU
        LDX UTTY; LDX TTYASG,2; LDA PPTR,2; SKA PRMSK; CLB
        STB PAC9A; BRM HFK; STX PUPAC
        CLB; BRM RFK; LDA =QTI; BRM TFK; BRM QPUT; LDX TF01
        LDA TFC04; SKN PAC9A; LDA TFC03
        STA PL,2; LDA X1; MRG PIM,2; STA PIM,2; LDA X4; LDB PACPTR
        LDX J0B; BRM RELPG
        BRU PAC0BA
TFC03   ZR0 CKCPU,4
TFC04   ZR0 CKCPU1,4
PAC9A   ZR0 0   NEG. IF IN EXEC
*
* SET STATUS FOR FORK
$STSTAT ZR0; LDA PTAB,2; LCY 1; RSH 23; STA EXEC1  =1 FOR SUBSYS STATUS
        LDA PIM,2; CLB; SKA K200; LDB =M1; STB SYSS
        CLB; SKA K400; LDB KM1; STB EXECS
        LDB X2; SKN EXEC1; LDB X1
        SKB SYSS; LDB X3; SKB EXECS; LDB =3786
        STB PRILEV   HIGH FOR HIGHLY PRIVELEGED FORKS
        LDA SN0P; SKN SYSS; LDA SNTRP; STA NSYS
        LDA SN0P; SKN EXECS; LDA SNTRP; STA NEXEC; BRR STSTAT
*

* SCAN SPECIFIED QUEUE FOR A READY JOB
* INPUT: X=QUEUE (QTIQ)
* NO SKIP RETURN: NO READY JOB
```

```
* SKIP RETURN: X=PACPTR OF READY JOB
SCAN    ZRO
PACSCN  STX PACLVL (SAVE QUEUE); BRU PEST5
* SCAN PACT
PEST    LDX PACPTR
PEST5   SKN PNEXT,2; BRU PEST4 (NO READY JOBS); STX PPREV
        LDX PNEXT,2; STX PACPTR; LDA PTEST,2; LRSH 15; CAX
        LDA CACLST,2; STA T; LDX PACPTR; LDA* PTEST,2; BRU* T
PEST4   BRR SCAN
$PACACT MIN SCAN; LDX PACPTR; BRR SCAN

* FETCH QUEUE JOB
FQJ     LDX =QTIQ; BRM SCAN; BRU FQJ2; BRU PEST3
FQJ2    LDX =QIBQ; BRM SCAN; BRU *+2; BRU PEST3
        SKN M48K; BRU FQJ3
        LDX =QQEQ; BRM SCAN; BRU *+2; BRU PEST3
FQJ3    MIN PURUN; LDA KM1; STA ACTR; SKN QQC; BRU PACG01
        CLA; STA TIME; SKN CMPST; BRU CMPG0
        SKE RAJOB2; BRU PACG01; BRU CMPIN3
* SWAP IN QTI JOB
PEST3   LDA =SETTB; STA TJOB
        LDX PACPTR; LDA PL,2; SKA K1B5; BRU PEST2; LDX T; STX PACQT
        LDX PACLVL; STX PACSWQ; LDX PPREV; STX PACSWP; LDX PACPTR
        LDA FQJ4; XMA PTEST,2; STA FQJ5
        STX PACSW; LDA =RAJOB1; STA SWRAJ; CLA; STA RAJ881
        LDA PTEST,2; LRSH 15; ADD =QTIG0; STA RFR
        BRM SETSWP; DIR; LOGG 34B,SWJOB; EIR; BRM SWAP
        BRU PACAC3; BRU PACG06; BRU PACWT
PEST2   ETR =-1-1B5; STA PL,2; SKN PNEXT,2; BRU PEST3
        LDA PNEXT,2; LDB PACLVL
        LDX PPREV; BRM QGET; LDA PACPTR; LDX =QTI; BRM QPUT; BRU FQJ
FQJ4    BRU PACDMB
FQJ5    ZRO 0
*
* QTI JOB HAD READ ERROR
PACAC3  LDA =FQJ; STA PACAC4
PACAC2  LDX PACPTR; LDA PTAB,2; LRSH 15; ETR K77; CAX; DIR
        LDA TTNO,2; ADD K100; SKA K400; BRU PACDEL
        STA TTNO,2; EIR; BRU* PACAC4
```

```
NOGO    BRM SETREL; BRM SWREL; CLA; STA PACSW; LDA KM1; STA RAER1
        LDA =PACG02; STA PACAC4; BRU PACAC2
* REMOVE USER
PACDEL  ETR =77777077B; STA TTNO,2; EIR; LDX PACPTR; LDA PNEXT,2
        LDB PACSWQ; LDX PACSWP; BRM QGET; LDX PACPTR
        LDA =700003B; STA PTEST,2; MIN RU; BRU* PACAC4
PACAC4  ZRO 0   EXIT LOCATION.

* FETCH COMPUTE JOB
FCJ     CLA; STA PACOMP; STA PG44; LDA ONE; STA CMPST
        MIN SPR; SKA SPR; BRU FCJ5
        LDX =QSQQ; BRM SCAN; BRU FCJ5; BRU FCJ4
FCJ5    LDX WHOQ; BRM SCAN; BRU FCJ3; BRU FCJ4
FCJ3    LDX WHOQ2; BRM SCAN; BRU PACG02 (NO COMPUTE JOB)
FCJ4    STX PACOMP; LDA WHOQ; XMA WHOQ2; STA WHOQ
        LDA PNEXT,2; LDB PACLVL; LDX PPREV; BRM QGET
        LDA T; STA CMPQT
        LDA PACOMP; LDX PPREV; SKE PACSWP; BRU *+2; STX PACSWP
        LDX =QQC; BRM QPUT
FCJ2    LDA KM1; STA CMPST (JOB COMING IN)
        LDA =RAJOB2; STA SWRAJ; LDA X4; STA SWPB; LDX PACOMP
        LDA =27B+QTIG0; STA RFR
        BRM SETSWP; DIR; LOGG 33B,SWJOB; EIR; BRM SWAP
        BRU FCJDEL; BRU *+2; BRU PACG02
        STA CMRRL1; STB CMRRL2; STX CMRRL3; MIN CMPST
        LDA SWIN14; STA CJP
        BRU PACG02
* READ ERROR. MOVE JOB TO QTI AND TRY AGAIN.
NOCOMP  LDA KM1; STA RAER2; BRM SETREL; BRM SWREL
FCJDEL  LDA =QQC; LDB =QQCQ; CBX; BRM QGET
        LDA PACOMP; LDX =QTI; BRM QPUT
        CLA; STA CMPST; STA PACOMP; BRU FCJ

* QUEUE JOB IS SWAPPING. RUN COMPUTE JOB OR WAIT.
PACWT   CLA; SKE CMPST; BRU *+2; BRU CMPGO; SKN CMPST; BRU PACWT3
PACWT2  SKE RAJOB2; BRU *+2; BRU CMPIN; SKE RAJOB1
        BRU PACWT2; BRU PACG04
* WAIT FOR QUEUE JOB. NO COMPUTE JOB.
PACWT3  SKE RAJOB1; BRU *-1; BRU PACG04
```

```
* COMPUTE JOB JUST CAME IN
CMPIN   LDA KM1; LDB =36; DIR; ADM RAJOB1; STB TIME
        EIR
CMPIN3  MIN CMPST; LDX PACOMP; SKN RAER2; BRU NOCOMP
        BRM SETSWP; BRM SWAPI
        STA CMRRL1; STB CMRRL2; STX CMRRL3
        LDA SWIN14; STA CJP; LDA CMPWT; STA T; LDA CMRRL1
        BRU CMPG02
* COMPUTE JOB READY TO RUN
CMPG0   LDA =WETTB; STA TJOB; LDA KM1; STA SCHLP
        LDX =QQCW; BRM SCAN; BRU MUX15; LDA =SETTB; STA TJOB
        LDA =36; DIR; STA TIME; SKR RAJOB1; NOP; EIR
CMPG03  LDA CMRRL1; LDB CMRRL2; LDX CMRRL3
CMPG02  BRM LABEL
        LDA CJP; LDX LW; BRM NPC
        LDX PACOMP; STX PACPTR
        LDA =QQC; LDB =QQCW; CBX; BRM UGET
        LDA KM1; STA PACJOB; BRU PACG05


*SET UP FOR SWREL
* INPUT: X=PACPTR
SETREL ZRO; LDA PTAB,2; LRSH 15; ETR K77; COPY AX,XB
        LDA PMTP,2; ADD =-NCMEM; STA SWPMT; XXB
        LDA RL1,2; STA SWR1; LDA RL2,2; STA SWR2; CBX
        LDA RL3,2; STA SWR3; BRR SETREL
* RELEASE PAGES FOR USER WITH READ ERROR
* INPUT: PSEUDO-REL IN SWR1,SWR2,SWR3.
*          PMT ADDR.-NCMEM IN SWPMT
SWREL   ZRO; LDA SWR1; LDB SWR2; LDX SWR3; BRM UPRL
        DIR; LOGG 6CB,SWREL; EIR
        LDX NINE; STX SWRL3
SWRL7   LDX SWRL3; LDA SRT,2; SKE ZERO; BRU *+2; BRU SWRL4
        SKG =NCMEM-1; BRU SWRL5; ADD SWPMT
SWRL8   CAX; LDA 0,2; ETR =77B5; SKE K4B5; BRU SWRL6
SWRL9   CXA; MRG X4; DIR; BRM REL; EIR
SWRL4   SKR SWRL3; BRU SWRL7; DIR; BRM RAGO; EIR; BRR SWREL
SWRL5   ADD =SMT; BRU SWRL8
SWRL6   SKE K2B5; BRU *+2; BRU SWRL9; SKE =12B5
```

```
        BRU *+2; BRU SWRL9
        SKE =355; BRM MONCR; BRU SWRL4
SWRL3   ZR0 0

* SET UP FOR SWAP
SETSWP  ZR0; LDA PTAB,2; LRSH 15; ETR K77; STA SWJOB
        LDB RL2,2; LDX RL1,2; XXA; LDX RL3,2; BRR SETSWP


* ROUTINE TO RELEASE PAGES.
* SUBSYSTEM PAGES GO WITH LOW PRIORITY WRITE.
* INPUT: B=PACPTR, X=JOB
*        A=0 FOR LOW PRIORITY WRITE. A=X4 FOR HIGH PRIORITY WRITE.
RELPG   ZR0; STA RELPG8; LDA PMTP,2; ADD =-NCMEM
        STA RELPG3 (PMT ADDR-NCMEM); DIR; LOGG 35B,LOGX
        LOGG 35B,RELPG; EIR
        LDX RL3,2; XXB; LDA RL1,2; LDX RL2,2; XXB
        BRM RELMEM; BRR RELPG
* RELEASE MEMORY
RELMEM  ZR0; DIR; LOGG 46B,RELMEM; EIR; BRM UPRL; LDX NINE; STX RELPG2
RELPG4  LDX RELPG2; LDA SRT,2; SKE ZERO; BRU *+2; BRU RELPG5
        SKG =NCMEM-1; BRU RELPG6; ADD RELPG3; MRG RELPG8
RELPG7  DIR; BRM REL; EIR
RELPG5  SKR RELPG2; BRU RELPG4; DIR; BRM RAGO; EIR; BRR RELMEM
RELPG6  ADD =SMT; BRU RELPG7
RELPG3  ZR0 0   PMT ADDR-NCMEM
RELPG2  ZR0 0   TEMP STORE FOR IX
RELPG8  ZR0 0   PRIORITY SWITCH. 0=LOW PRIORITY.

* RAJOB1=QTI JOB
* RAJOB2=COMPUTE JOB
* RAJOB3=RFK USER, BRS 44, AND OTHER HANG CALLS.

PACOMP  ZR0 0   PACPTR OF COMPUTE JOB

CMRRL1  ZR0 0   RRL1 FOR COMPUTE JOB
CMRRL2  ZR0 0   RRL2 FOR COMPUTE JOB
CMRRL3  ZR0 0   RRL3 FOR COMPUTE JOB
$CJP    ZR0 0;* NO. OF COMPUTE JOB PAGES/
```

```
$PG44   ZR0 0   N0. 0F PAGES READ BY BRS 44
SCHLP   ZR0 0   -1 IF WAITING F0R QQC J0B T0 BE READY T0 RUN.
CMPST   ZR0 0   C0MPUTE J0B STATE. -1=C0MING IN. 0=IN. 1=N0 J0B.

SPR     ZR0 0   SUPERVIS0R SWITCH. SUPER RUNS IF SWITCH IS EVEN.
WH0Q    DATA QQEQ   SET T0 QQEQ 0R QI0Q. WHICHEVER WE TRY NEXT.
WH0Q2   DATA QI0Q
CMPQT   0 0     ACTIVATI0N C0NDITI0N F0R C0MPUTE J0B
PACSW   ZR0 0   PACPTR 0F J0B C0MING IN FR0M QTI
PACSWQ  ZR0 0   QUEUE 0F J0B C0MING IN (PACLVL)
PACSWP  ZR0 0   PPREV 0F J0B C0MING IN
PACQT   ZR0 0   ACT. TEST 0F J0B C0MMING IN.
PACJ0B  ZR0 0   0=QTI J0B RUNNING. -1=C0MPUTE J0B RUNNING.
PURUN   DATA -1   RUN PU IF N0 QUEUE J0BS.


*CAUSE A PR0G INTERRRUPT
PACINT  LDA PPTEST; ETR K37; SKE FIVE; BRU PI1
        LDX PACPTR; LDA PTAB,2; RCY 15; ETR K77; CAX (J0B);
        LDA <185; C0PY AB,N; ADM TTN0,2; SKB TTN0,2; BRM M0NCR
        LDA FIVE; LDX PACPTR
PI1     LDB PL,2; SKN PL,2; LDB SBRSRT    GET INTERRUPTED ADDRESS
        C0PY AX,BA; ETR ADMSK; LDX 200B,6 GET MARK L0CATI0N P0INTER
        C0PY XA,AB; LDX 0,6; STB 0,6   (GET INTERRUPT RETURN ADRS)
        ETR ADMSK;  ADD =40000001B; STA 0; LDX PACPTR; BRU PAC0VF
*  CAUSE AN INSTRUCTI0N TRAP
PACTRP  LDA PL,2; STA 0; CLEAR
        SKN XPB; BRU PACTP2; LDX XPB; LDA PA,2; LDB PB,2; LDX PX,2
PACTP2  STA SSQ1; STB SSQ2; STX SSQ3; TRAP 52
*ACTIVATI0N TEST R0UTINES
CACLST  0 CAC0; 0 CAC1; 0 CAC2; 0 CAC3
        0 CAC4; 1 PACACT; 0 CAC6; 0 CACK
        0 CAC2; 0 CAC9; 0 CAC10; 0 CAC0; 0 CAC0
        0 CAC0; 0 CAC0; 0 CAC0; 0 CAC0; 0 CAC0; 2 CAC2
        0 CAC0; 0 CAC0; 0 PEST; 0 PEST; 0 PEST; 0 PEST; 0 CAC0
        0 CAC11; 0 CAC10; 0 CAC1; 0 CAC0
* SPECIAL ACTIVATI0N R0UTINES
ACTLST  BRU PACSRT; BRU PACINT; BRU PACTRP
```

```
CACK    BRM MONCR
CAC0    SKG ZERO; BRU PEST; BRU PACACT
CAC1    SKG ZERO; BRU PACACT; BRU PEST
CAC2    SKG <M1; BRU PEST; BRU PACACT
CAC3    BRM MONCR
CAC4    BRU* PTEST,2
CAC6    SKG REAL; BRU PACACT; BRU PEST
CAC9    SKA =YELY; BRU PEST; BRU PACACT
CAC10   SKG <M1; BRU PACACT; BRU PEST
CAC11   SKA =59Y; BRU PEST; BRU PACACT
*
*
PACDMS  ZRO 0
PACQUE  ZRO 0    QUEUE FOR TERMINATING FORK
PACLVL  ZRO
PPREV   ZRO
PPTEST  ZRO      0                  SAVE OF STARTUP PTEST
$YPL    ZRO 0
$YRL1   ZRO 0
$YRL2   ZRO 0
$YPPTR  ZRO 0
$YPTEST ZRO 0
$YPTAB  ZRO 0
$YPIM   ZRO 0
PTESTS  ZRO      0                  SAVE OF DISMISS PTEST

FPLST ZRO
$SNOP  NOP
SNTRP  TRAP 53
*
*


* PHANTOM USER
*
* INITIALIZE PU
PUINIT ZRO; CLA; STA PACPTR; STA PUPAC; STA XPB; STA JOB
```

```
        STA PMTJ0B; STA UTTY; STA RLTS; LDA XX; STA TIME
        LDA PURRL1; LDB PURRL2; LDX PURRL3; BRM LABEL; BRR PUINIT
*
PUDMS   LDA PUCTR; STA PUCTR1; CLA; STA XPB; EIR; STA PUSW; BRU PACG0
* THIS CHECKS TO SEE IF REAL IS NEG OR ABOUT TO BE.
* IT RUNS EVERY 3 SEC.
PU3SEC  BRM PUINIT; LDA REAL; SKA X4; BRM MONCR
        ADD =180; STA PUTIME
        ADD =60*60*60-180; SKA X4; SKN FATIG; BRU PUG03
        MIN FATIG; LDA =6*2B6; DIR; BRM TRO (SEE BRS159); EIR
        LDA ONE; ADM ALARM
PUG03   EQU *
        IF DISC<3       TURN POSITION POWER OFF ON DP DISC AFTER 1 MINUTE
        SKN IDCL1; BRU PUG03A                OF INACTIVITY
        LDA REAL; SUB =60*60; SKG DREAL; BRU PUG03A; LDA XX; STA DREAL
$PPOFF  EOM 10226B; NOP 10227B; NOP 10222B; NOP 10223B    SEE DRMSET
        ELSE
        LDA     RQ1
        SKE     RQ3             CHECK IF DISC QUEUE EMPTY
        BRU     *+3
        LDA     XX              YES, SET DREAL TO INFINITY
        STA     DREAL
        ENDF
PUG03A  EQU *
PUG06   EQU *
        BRM WCLEAN
        IF ST
*       DUMP STATISTICS ONTO DISC IF A 400 WORD BUFFER IS FULL
$PUST1  LDA SP; EOR SP1; SKG K377; BRU PUST2
        LDA =RDP*100B+RSP (DISC AND SET); BRM RLABEL; BRU SDUMP
PUST2   LDA =0640B; BRM RLABEL   (SDUMP RETURNS TO PUST1 TO RECHECK )
        ENDF

PUG02   LDA KM1; STA PUSW
        LDA REAL; RSH 6; CAB; SUB LREAL; STB LREAL
        MUL NU; LSH 23; ADM UTIME; BRU PUG05
PUG04   BRM PUINIT; BRU PUG02
* P.U. SCHEDULER
* RELEASE PAGES OF LAST PU JOB AND CLEAR PU RELABELING
```

```
PUG0    LDA =SETT6; STA TJ0B
        LDX PUJ06; LDA PMTP,2; ADD =-NCMEM; STA RELPG3
        LDA X4; STA RELPG8; LDA PURL1; LDB PURL2; LDX PURL3
        BRM RELMEM; CLA; STA PURL1; STA PURL2; STA PURL3
* SCAN F0R P.U. J0BS
PUG05   LDA PUCTR; DIR; SKG ZER0; BRU PUDMS; EIR; LDX =PUBPTR; STA PUCTR1
PUSCN   STX PUCPTR; LDX 0,2; LDA 1,2; STA PUTST; LRSH 15; ETR K77
        AXC; BRU* PUCLST,2
PUSET   ZR0; LDX PUCPTR; LDX 0,2; LDA 2,2; RSH 12; STA PUPAC
        CLA; LCY 12; STA PUTTY; BRR PUSET
* SET PU RELABELLING BACK
SPURL   ZR0; BRM MPPACT; CLA; STA J0B; STA XPB
        LDA PURRL1; LDB PURRL2; LDX PURRL3; BRM LABEL
        LDA RRL3; ETR K77; STA RLTS; BRR SPURL
NPUG0   BRM SPURL; BRU PUG0
$NPUNXT BRM SPURL
* CAN'T PR0CESS THIS ENTRY
PUNXT   LDX* PUCPTR; CXA; SKE PUEPTR; BRU PUSCN
        LDA PUCTR1; DIR; SKE PUCTR; BRU PUG0; BRU PUDMS
* PHANT0M USER ACTIVATI0N TESTS
PUCLST  DATA PUCR1,PURBT,PU124A,PUCR1,PUCR1,PUCR1
        DATA PUL0,PUTIM,PUCR1

* TEST PR0GRAM INTERRUPT TIME 0UT
PUTIM   LDX* PUCPTR; LDA REAL; SKG 2,2; BRU PUNXT
        LDA 3,2; LDB PRMSK; LCY 12; STB PUTIM9 (PACPTR 0F F0RK)
        CBX; LRSH 12; STA PUTTY
        LDX PTAB,2; XXA; LDX TTYASG,2; E0R PTAB,2; SKA =77B5
        BRU PUTIM1   WE F0UND S0ME 0LD PUTIM ENTRY F0R AN0THER J0B
        BRM PUINT; STX PUTIM8 (J0B)
        LDX PUTIM9; LDA PTEST,2; SKE =7B5 (IS F0RK DEAD)
        BRU PUTIM2 (N0)
PUTIM1  LDX* PUCPTR; LDA PUDEAD; STA 1,2; BRU PUNXT
PUTIM2  LDA K1B5; SKA PIM,2; BRU *+2; BRU PUACT1; STX T2
        LDA =PURBTA; BRM SCFK; STX PUPAC
        SKN T2; BRM RCJ; BRU PUACT
PUTIM3  LDA K1B5; LDX PUTIM9; BRM IIR; BRU NPUG0
        LDX PUTIM8; LDA K1B5; ADM TTN0,2; SKA TTN0,2; BRU NPUG0; BRM M0NCR
```

```
PUTIM9 0
PUTIM8 0

* REMOVE EXTRA PU ENTRIES
PUL0    BRU PUACT1

* TEST FOR RUBOUT
PURBT   BRM PUSET
        LDA PUPAC; LRSH 3; STA TIPIX; LDX PUTTY; SKG ONE; BRU PURBT2
        SKN TTYASG,2; BRU PURBT3; LDX TTYASG,2; STX T2; BRM PUINT; LDX T2
        LDA =PURBTA; BRM SCFK; STX PUPAC
        DIR; LOGG 23B,PUPAC; LOGG 23B,PUTTY; EIR
        SKN T2; BRM RCU; BRU PUACT
* SETS T2 TO 0 IF ANY FORK IS ON QQC
PURBTA ZRO; LDA PIM,2; SKA X1; BRU PURBT1; COPY XA,AB; SKE QQC
        BRR PURBTA; COPY BA,B; STB T2; BRR PURBTA
PURBT1  MRG X2; STA PIM,2; BRU PUNXT
* BRS 112 OR ON INTERRUPT
PURET2  SKN TTYASG,2; BRU PUACT1; BRM MONCR
PURBT3  LDA PUDEAD; STA PUTST; BRU PUACT1

PU124A  EQU *   USER HAD DISC ERROR. ACTIVATE PU124B IF TTNO COUNT IS -1.
        LDA* PUTST; SKG =-2B5-1; BRU PUNXT  (IO IN PROGRESS)
        BRM PUSET; LDX PUPAC; LDA =PURBTA; BRM SCFK; SKN T2
        BRM RCU; BRU PUACT

PU124B  EQU *   INTERRUPT FORK FOR IO ERROR IF INT 11 ARMED.
        LDA K2B5; ADM* PUTST  SET UP TO ACTIVATE AT PACTRP
        LDX PUPAC; LDA K1000; BRM IIR; BRU NPUGO  (UNARMED, CAUSE PACTRP);
        BRU NPUGO  (ARMED. PTEST OP-CODE IS 5. INT 11 WILL BE INVOKED.)

PUINT   0
        LDX PUTTY; LDA TF2,2; CAB; ETR K77; CAX
        LDA =401B5; SKA TTNO,2; BRU PUNXT
        LDA TIPIX; SKB =B113Y+RCY; SKG TWO; BRR PUINT; BRU PUNXT

* ILLEGAL PU INDEX
PUCR1   BRM MONCR
* ACTIVATE P.U. REQUEST ROUTINE
```

```
PUACT2 BRM PUSET
PUACT  LDX PUPAC; LDA PTAB,2; LRSH 15; ETR K77; STA SWJ0B
       STA PUJ0B; STA J0B
       LDA =30B+QTIG0; STA RFR
       C0PY A,5; LDX =NCMEM
       STA PURL1; STB PURL2; STX PURL3; BRM SWAPR; BRU PUSWER
       BRM LABEL; LDX PUPAC; BRM PGET
PUACT1 EQU *
       IF ST
       DIR; SKN STSW; BRU PUACTA
       LDA =RSR; BRM RLABEL; BRM SSS
       2 =3B5+5
       6* PUCPTR
       1 REAL
PUACTA EIR
       ENDF
       LDX PUCPTR; DIR; LDA FPULST; XMA* 0,2
       XMA 0,2; STA FPULST; SKE PUEPTR; BRU *+2; STX PUEPTR
       SKR PUCTR; MIN PULIM; EIR
       LDA PUTST; LRSH 15; ETR K77; CAX; BRU* PUCSET,2
* PHANT0M USER REQUEST PROCESSING PREPARATI0N
PUCSET ZR0 PUCR2; ZR0* PUTST; DATA PU124B,PUCR2,PUCR1,PUCR2
       DATA PUG0,PUTIM3,PUCR2
* ILLEGAL PU INDEX
PUCR2  BRM M0NCR
* ERR0R SWAPPING IN PU J0B
PUSWER LDX J0B; DIR; LDA TTN0,2; ADD K100; STA TTN0,2
       EIR; SKA K400
       BRU PUER2; BRU NPUNXT
PUER2  LDA =655; STA PUTST; BRU PUACT1

* MAKE P. U. ENTRY
* INPUT: A=1,2. B=2,2. X=3,2. L0W 6 BITS 0F X=TTY N0.
* 0UTPUT: X=PU PTR.
EPU    0
       DIR
       XMA* FPULST; SKR PULIM; BRU *+2; BRM M0NCR
       MIN PUCTR; XMA FPULST
       STX PUXSV; CAX; XMA* PUEPTR
```

```
        STX PUEPTR; XMA 0,2; STA 1,2; STB 2,2
        XMA PUXSV; STA 3,2; XMA PUXSV
        EIR; BRR EPU
PUXSV   ZRB 0
*
*
FPULST  ZRB 0   1ST FREE ENTRY
$PUTTY  0;* TTY BEING PROCESSED BY PHANTOM USER
PUBPTR  ZRB 0   PTR. TO 1ST ACTIVE ENTRY. LAST ENTRY POINTS TO PUBPTR
PUCTR   ZRB 0   NO. OF ENTRIES.
PUCTR1  ZRB 0   COUNTER DURING PU PROCESSING.
PUCPTR  ZRB 0   IND. PTR. TO ACTIVE ENTRY DURING PU PROCESSING.
PUEPTR  ZRB 0   LAST ACTIVE ENTRY.
PUPAC   ZRB 0   PACPTR OF ENTRY BEING PROCESSED BY PU.
PUDEAD  6   0   NULL PU ENTRY
PUTST   ZRB
PULIM   ZRB NPUQ    COUNTS NO. OF PU ENTRIES.
PUPACP  ZRB 0    PACPTR FOR PU
$PUSW   ZRB 0;* -1 WHEN IN PU. 0 OTHERWISE.
PURL1   ZRB 0    RELABELING FOR P.U. JOBS
PURL2   ZRB 0
PURL3   ZRB 0
PURRL1  DATA 40404040B
PURRL2  DATA 40404040B
PURRL3  DATA 640B
PUTIME  ZRB 0    3 SECOND ACTIVATION TIME.
PUJOB   ZRB 0
LREAL   ZRB 0    LAST REAL FOR TIME CALCULATOR
*
*

* GET USER TO CORE
* INPUT: X=PACPTR
PGET    ZRB; STX PUPAC; CLA; STA XPB
        LDA PTAB,2; LRSH 15; ETR K77; STA JOB
        CAX; ADD =ETTB; STA TJOB (SET UP TIME CHARGING)
        LDA PMTP,2; ADD =20000000B-NCMEM; STA PMTJOB (SET UP PMT PTR)
        DIR; LDA TTNO,2; ETR =77776077B; STA TTNO,2; EIR
        ETR K77; STA UTTY (SET UP USER TELETYPE)
```

```
        LDA RRL3; ETR K77; STA RLTS; LDX PUPAC
        LDA PTAB,2; SKA X1; BRU *+2; BRU PGET1 (NO TS BLOCK)
        LDA PIM,2; ETR =70B; LRSH 3; MRG X4; STA XPB
        DIR; LOGG 24B,PGET; LOGG 24B,UTTY; LOGG 24B,JOB; EIR
PGET1   BRR PGET
        ZRO 0


*
* SWAPPER
*
* PREPARE TO RUN FORK. STARTS READING PAGES IF NECESSARY.
* INPUT: PSEUDO=RELABELING IN A,B,X
*        JOB NO. IN SWJOB
*        RAJOB PTR. IN SWRAJ
*        0 IN SWPB FOR LOW PRIORITY READ.
*        X4 IN SWPB FOR HIGH PRIORITY READ.
*   SWPB IS SET TO 0 BEFORE THE SWAPPER EXITS.
* RETURNS: NO SKIP=READ ERROR. A=ERROR COUNT
*          1 SKIP=PAGES ARE IN. REAL REL. IN A,B,X
*          2 SKIP=PAGES ARE NOT IN, BUT ARE COMING.
SWAP    ZRO; DIR, LOGG 47B,SWAP; EIR; STA SWR1; STB SWR2; STX SWR3; BRM UPRL
        LDX SWJOB; LDA PMTP,2; ADD =-NCMEM; STA SWPMT
        CNA; ADD =SMT; STA SWSMT
        LDA =RAJOBE; SUB SWRAJ; CAX; LDA X4; LRSH 0,2; STA SWRAB
        LDA* SWRAJ; STA SWRAI; MIN* SWRAJ
        LDX KM10D; LDA SWJOB; STA BJOB
SWLP1   STX SWIX; LDA SRTE,2; SKE ZERO; BRU *+2; BRU SWLP
        SKG =NCMEM-1; ADD SWSMT; ADD SWPMT
        MRG SWPB; MRG SWRAB; MIN* SWRAJ; DIP; BRM B; EIR
        LDX SWIX
SWLP    BRX SWLP1; CLA; STA SWPB
        DIR; BRM RAGO; EIR
        SKR* SWRAJ; NOP; LDA* SWRAJ; SKE SWRAI
        BRU SWSK2 (PAGES NOT ALL IN)
        LDA SWRAJ; SUB =RAJOB; CAX; SKN RAER,2; BRU SWSKO
        BRM SWIN; MIN SWAP; BRR SWAP
* READ ERROR
SWSKO   STX SWSKOA; BRM SWREL; LDX SWSKOA; LDA KM1; XMA RAER,2; BRR SWAP
```

```
SWSKOA ZRO 0
* PAGES NOT ALL IN
SWSK2  MIN SWAP; MIN SWAP; BRR SWAP
*
* PAGES ALL IN. PUT REAL REL. IN A,B,X
$SWIN14 ZRO 0
SWIN   ZRO; CLA; STA SWIN14; LDX KM10D
SWIN8  LDA SRTE,2; SKE ZERO; BRU SWIN2; LDA K40; STA SRTE,2
SWIN3  BRX SWIN8; BRM PTRL; BRR SWIN
*NON-ZERO PSEUDO-REL ENTRY
* THIS SECTION HAS REDUNDANT CHECKING WHICH SHOULD BE REMOVED
* WHEN SWAPPER AND PAGER ARE DEBUGGED.
SWIN2  STX SWIX; MIN SWIN14
       SKE =NCMEM; MIN SWNCM; SKG =NCMEM-1; ADD SWSMT
       ADD SWPMT; SKN SWNCM; BRU SWIN6; STA SWIN7
       DIR; BRM W; EIR; LDA SWIN7
SWIN6  CAX; LDA 0,2
       ETR =378+177B5; SKA X1; MRG K40; SKN SWNCM
       BRU SWIN4
* TS BLOCK
       LDB =77B5; SKM K2B5; BRM MONCR; SKA K40; BRM MONCR
       BRU SWIN5
* NOT TS BLOCK
SWIN4  SKR SWNCM; NOP
SWIN13 LDB =75B5; SKM K4B5; BRU *+3; MRG K40 (STATE 4 OR 6); BRU SWIN5
       LDB =77B5; SKM K2B5; BRU SWIN9 (NOT 2)
SWIN5  ETR K77; LDX SWIX; STA SRTE,2; BRU SWIN3
SWIN9  STA SWIN10; SKM =12B5; BRM MONCR; LDX 0,2
SWIN12 LDA 0,2; SKM =12B5; BRU SWIN11; CAX; BRU SWIN12
SWIN11 ETR =37B+77B5; LDB X1; SKB SWIN10; MRG K40
       BRU SWIN13
SWIN10 ZRO 0
SWIN7  ZRO 0

* READ PAGES FOR RUNNING FORK. HANG UNTIL ALL PAGES ARE IN.
* INPUT: PSEUDO-REL IN A,B,X
*        JOB NO. IN SWJOB
* RETURNS: NO SKIP=READ ERROR. A=ERROR COUNT.
*          SKIP=PAGES IN. REAL REL. IN A,B,X
```

```
SWAPR   ZR0; STA SWR1; LDA X4; STA SWP0
        LDA =RAJ0B3; STA SWRAJ; CLA; STA RAJ0B3; LDA SWR1
        BRM SWAP; BRU SWAPR2; BRU SRIN
        LDA* SWRAJ; SKE SWRAI; BRU *-2
        SKN RAER3; BRU SWAPR2; BRM SWIN
SRIN    MIN SWAPR; BRR SWAPR
SWAPR2  BRM SWREL; LDA KM1; XMA RAER3; BRR SWAPR (READ ERR0R)

ANN     EQU 1
        IF -ANN
* RETURN REL. IF PAGES ARE IN. DOES NOT START RAD.
* INPUT: PSEUDO-REL IN A,B,X
*        JOB NO. IN SWJ0B
* RETURNS: NO SKIP=PAGES NOT IN. A=COUNT OF MISSING PAGES.
*          SKIP=PAGES IN. REAL REL. IN A,B,X
$SWAPL  ZR0; STA SWR1; STB SWR2; STX SWR3; BRM UPRL; LDA KM1; STA SLPGC
        LDX SWJ0B; LDA PMTP,2; ADD =2B7-NCMEM; STA SWPMTI
        LDX <M100
SLLP2   STX SWIX; LDA SRTE,2; SKE ZER0; BRU *+2; BRU SLLP
        SKG =NCMEM-1; BRU SLSYS; CAX; LDA* SWPMTI
SLLP3   LDB =77B5; SKM =12B5; BRU SLLP4; CAX; LDA 0,2; BRU SLLP3
SLLP4   ETR =77B5
        LRSH 15; CAX; LDA SWLS; LSH 0,2; SKA X4
        MIN SLPGC; LDX SAIX
SLLP    BRX SLLP2; SKN SLPGC; BRU SLRET
* PAGES ARE IN
        LDA SWR1; LDB SWR2; LDX SWR3; BRM SWAPR; BRR SWAPL
        MIN SWAPL; BRR SWAPL
SLRET   MIN SLPGC; LDA SLPGC; BRR SWAPL
SLSYS   ADD =SMT; CAX; LDA 0,2; BRU SLLP3

        ENDF

* CALLED WHEN ALL PAGES ARE KNOWN TO BE IN CORE AND
* NO BRING IS NEEDED.
* INPUT: PSEUDO-REL IN A,B,X
*        JOB NO. IN SWJ0B
* RETURNS: NO SKIP=READ ERR0R.  SKIP=REAL REL IN A,B,X
SWAPI   ZR0; STA SWR1; STB SWR2; STX SWR3; BRM UPRL
```

```
        LDX SWJ0B; LDA PMTP,2; ADD =-NCMEM; STA SWPMT
        CNA; ADD =SMT; STA SWSMT; BRM SWIN
        BRR SWAPI
*
* SWAPPER VARIABLES
SWJ0B   ZR0 0     J0B N0.
SWRAJ   ZR0 0     RAJ0B P0INTER
SWPMT   ZR0 0     PMT P0INTER-NCMEM
SWPMTI  ZR0 0     PMT P0INTER INDEXED
SWSMT   ZR0 0     SMT-PMT
SWRAB   ZR0 0     RAJ0B BIT MASK
SWRAI   ZR0 0     INITIAL VALUE 0F RAJ0B
SWIX    ZR0 0     TEMP STORE F0R SRT IX
SWPB    ZR0 0     HIGH PRIORITY READ SWITCH. X4 F0R HIGH PRI0R. READ.
SRT     BSS 8     STORAGE F0R UPRL
SRT44E  BSS 2
SRTE    BSS 0     END 0F SRT
SWR1    ZR0 0     RL1 INPUT
SWR2    ZR0 0     RL2 INPUT
SWR3    ZR0 0     RL3 INPUT
SWNCM   DATA -1     SWITCH. -1=NCMEM. 0=N0T NCMEM
SWLS    DATA 64237777B   IN C0RE STATE MASK. STATES 2,4,5,6,8,9 IN C0RE.
SLPGC   DATA -1     PAGE C0UNT F0R SWAPL

RAJ0B   ZR0 0
RAJ0B1  DATA -1     GTI J0B
RAJ0B2  ZR0 0     C0MPUTE J0B
RAJ0B3  ZR0 0     HANG CALLS
RAJ0B4  ZR0 0
RAJ0B5  ZR0 0
RAJ0B6  ZR0 0
RAJ0B7  ZR0 0
RAJ0BE  BSS 0
RAER0R  BSS 0     RAD ERR0R C0RRESPNDING TABLE.
RAER    DATA -1
RAER1   DATA -1
RAER2   DATA -1
RAER3   DATA -1
RAER4   DATA -1
```

```
RAER5  DATA -1
RAER6  DATA -1
RAER7  DATA -1

*
* UNPACK RELABELLING REGISTERS
* INPUT: RELABELING IN A,B,X
* OUTPUT: ALL REGISTERS CLOBBERED
UPRL   ZR9; STB SWT3; LRSH 18; STA SRT
       CLA; LCY 6; STA SRT+1; CLA; LCY 6; STA SRT+2
       CLA; LCY 6; STA SRT+3; LDA SWT3; LRSH 18; STA SRT+4
       CLA; LCY 6; STA SRT+5; CLA; LCY 6; STA SRT+6
       CLA; LCY 6; STA SRT+7; CXA; RSH 6; ETR K77; STA SRT+8
       CLA; LCY 6; STA SRT+9; BRR UPRL
SWT3   ZRB 0
*
* PACK RELABELING IN SRT INTO A,B,X
PTRL   ZRB; LDA SRT+3; LRSH 6; LDA SRT+2; LRSH 6
       LDA SRT+1; LRSH 6; LDA SRT; LRSH 6; STB SWRL1
       LDA SRT+9; LRSH 6; LDA SRT+8; LRSH 18; CBX
       LDA SRT+7; LRSH 6; LDA SRT+6; LRSH 6
       LDA SRT+5; LRSH 6; LDA SRT+4; LRSH 6
       LDA SWRL1; BRR PTRL
SWRL1  ZRB 0
*
*
* SET UP REAL RELABELING. KEEPS PAGE 6 RELABELLING FROM RRL3.
* INPUT: REAL RELABELING IN A,B,X
LABEL  ZRB; STA RRL1; STB RRL2; CXA
       ETR K77; XMA RRL3; ETR =7700B; ADM RRL3
       LRR1; PBT RRL1; LRR2; PBT RRL2; LRR3; PBT RRL3; BRR LABEL

CHREL  ZR9; LDA RL1,2; LDB RL2,2; LDX JOB; STX SWJOB
       LDX RL3,2; BRR CHREL
*
RRL1   ZRB
RRL2   ZRB
RRL3   ZR9 0   CURRENT CONTENTS OF RR3
RLTS   ZRB     0  TS BLOCK RELABELLING FOR RUNNING FORK
```

```
RLPACT DATA 6000B     PAC TABLE REAL RELABELLING
RLDSC  DATA 1000B     DISC REAL RELABELLING
RLWB   DATA 1100B     W BUFFER DRIVERS REAL RELABELLING
$RLTYM DATA 0600B+TYMPAGE       TYMNET LABELLING
*
*
* ROUTINES TO RELABEL EXTRA MONITOR PAGES.
*
* PAC TABLE RELABELING
MPPACT ZRO; LDA RLPACT; MRG RLTS; STA RRL3
       LRR3; POT RRL3; BRR MPPACT
*
* DISC RELABELING
MPDSC  ZRO; LDA RLDSC; MRG RLTS; STA RRL3
       LRR3; POT RRL3; BRR MPDSC
*      SETUP TYMNET LABELLING
$MPTYM 0; LDA RLTYM; XMA RRL3; LRR3; POT RRL3; BRR MPTYM

* W BUFFER ROUTINE RELABELING
MPWB   ZRO; LDA RLWB; MRG RLTS; STA RRL3
       LRR3; POT RRL3; BRR MPWB
*
* CHANGE RRL3 RELABELING
$CHR3  ZRO; LRR3; POT RRL3; BRR CHR3

*      CLOCK ROUTINES AND TABLES
*
$DNTERM DATA    -1              DO NOT TERMINATE FORK IF 0
TIME   ZRO      0   SHORT
TTIME  ZRO      0   LONG
TIME1  ZRO      0               TIME AT START OF FORK
UNOSV  ZRO      0               USER NUMBER SAVE CELL
$CLOCK3 DATA -1                 SKR CELL FOR INTERRUPT 74
* CLOCK INTRRUPT ROUTINE  AVG. TIME= .047 MS
CLINT  0
$CLINT1 MIN REAL      SOMETIMES BRU CLINTT   (SEE TVDMS)
CLINT4 MIN* TJOB; SKN CLINT; MIN STIME
       BPT4
CLINTC BRM MONCR
```

```
CLINT2 SKR TTIME; NOP; SKR TIME; BRI CLINT; SKN TTIME
       SKN ACTR; BRU CLOUT; BRI CLINT
CLOUT  BSS 0
       EOM     22400B
       IF      DDIFP
       SKR     UMTF
       NOP
       ELSE
       XMA J0B; STA UMTJ0B; XMA J0B
       ENDF
       SKN CLINT; BRU CKOI
       BRI CLINT
CKOI   STA SA; LDA* CLINT; ETR =20077777B; SKE =40000B
       BRU OK; LDA =100; STA COUNT; STX SX
       SKN 0; BRU OKX; LDX 0
LPT    LDA 0,6; SKA =40000B; BRU IND; BRU OKX
IND    SKA X2; ADD SX; CAX; SKR COUNT; BRU LPT
       BRI =*+1; TRAP 54
OKX    LDX SX
OK     LDA SA; BRI CLINT
CLINTT EQU *   SEE IF TYMNET OUTPUT RING (TB0) IS EMPTYING.
       MIN CLINTW; STA SA
       LRR3; POT RLTYM; LDA* TROSW; LRR3; POT RRL3
       SKE TFREEC; BRU CLINTU
       LDA <M1; STA TROSW; LDA CLINTU; STA CLINT1; LDA SA; BRU CLINT1
CLINTU MIN REAL; LDA SA; BRU CLINT4
$CLINTV BRU CLINTT
CLINTW 0

SA     ZRO 0
SB     ZRO 0
SX     ZRO 0
COUNT  ZRO 0
       IF      DDIFP
*      USER MODE TRANSITION TRAP ROUTINE
*      USED TO DETECT APPROPRIATE TIME FOR
*      DISMISSAL FOR QUANTUM OVERFLOW OR TO
*      REPORT FLOATING OVERFLOW INTERRUPTS
*      TO A USER PROGRAM.
```

```
TRAPT   ZRØ
        STA     SSØ1
        STB     SSØ2
        STX     SSØ3
        LDA     TRAPT
        STA     Ø
        SKN     SFPØVI
        BRU     TRAPT2
        COPY    A
        STA     SFPØVI
        LDA     K2ØØ4
        LDX     PACPTR
        BRM     RIIR
TRAPT1  SKR     Ø
        BRU     *+1
        SKN     UMTF
        BRU     PØPX
        EØM     2240ØB
        BRU     PØPX
TRAPT2  SKN     UMTF
        BRU     TRAPT1
        SKN     DNTERM
        BRU     *+2
        BRU     PACQE
        COPY    A
        STA     UMTF
        BRU     TRAPT1
$SFPØVI ZRØ     Ø               FLOATING ØVERFLØW FLAG (-1 FØR ØF PENDING)
$UMTF   ZRØ     Ø               QUANTUM ØVERFLØW FLAG (-1 FØR PENDING)
        ELSE
*
* USER MØDE TRANSITIØN TRAP
TRAPT   ZRØ; SKN DNTERM; BRU* TRAPT
        STA SSØ1; STB SSØ2; STX SSØ3
        LDA UMTJØB; SKE JØØ; BRU WRJØB
TRAPT1  LDA TRAPT; STA Ø; LDB PACQ4; BRU QQEDMS
WRJØB   LDA SSØ1; BRU* TRAPT
$UMTJØB ZRØ
        ENDF
```

```
* FAST CLOCK (POWER OFF INTERRUPT)
PWFI    ZR0; STA P0FFA; STB P0FFB; STX P0FFX; MIN PWFL
        LDX =100075B; BRM ST0P;* BRI NOT ISSUED
*
* POWER ON INTERRPUT
PWNI    ZR0; BRU* *
P0FFA   ZR0 0
P0FFB   ZRF 0
P0FFX   ZR9 0
*
*
* BRS'S FOR TIMING
* BRS 80   READ COMPUTE TIME.
*
* BRS 42
RREAL   LDA UZ0NE; ETR K37; LDB K40; SKB UZ0NE; SKN DSTSW; SUB 0NE
        ADD DMIN; SKG TMK1; BRU *+2; ADD TMK2; CAB; LDA REAL; LDX YEAR
        BRU XP0P

$BRS135 EQU *  INTERRUPT FORK AFTER TIME IN B IS ELAPSED
        EAX PUBPTR              LOOK FOR OLD BRS 135 ENTRIES ON PU
B135A   LDX 0,2; CXA; SKE =PUBPTR; BRU *+2; BRU B135B
        LDA 1,2; SKE B135C; BRU B135A    (NOT A PUTIM ENTRY);
*       REMOVE ENTRY IF PU PACPTR SAME AS CURRENT OR IF SAME USER
        LDA 3,2; EOR UTTY; LRSH 12; EOR PACPTR
        SKB PLMSK (CHECK UTTY; SKG PLMSK (CK PACPTR); BRU *+2 ; BRU B135A
        LDA PUDEAD; STA 1,2    CANCEL PU ENTRY AND WAIT UNTIL IT
        LDA =3000; BRU B81B    IS CLEARED FROM THE PU CHAIN
B135B   LDX PACPTR; LDA K1B5; MRG PIM,2; STA PIM,2
        LDA SS02; MUL =1727024B     CONVERT MILISECONDS TO CLOCK TICKS
        ADD REAL; COPY AX,B; LDA PACPTR; LSH 12; MRG UTTY; COPY XB,AX
        LDA B135C; BRM EPU
        BRU P0PX
B135C   7 PUTIM

$BRS45  LDA =3000       SIMULATE QUANTUM OVERFLOW
$BRS81  MIN 0; SKG =2000; LDA =2000   MINIMUM DISMISS OF 2 SECONDS
B81B    MUL =1727024B; ADD REAL; LDX J0B; STA ACDATA,2
        CXA; ADD B81A; CAB; LDX =QTI; BRU P0PDMS
```

```
* BRS 22   DO NOT TERMINATE FORK
NTERM  CLA; STA DNTERM; BRU POPX
*
* BRS 23   ALLOW FORK TO TERMINATE
ALTERM LDA =M1; STA DNTERM; SKN TIME; BRU POPX; BRU PACQE
*


*
* SYSTEM START
SETSET EOM* 0, TSN; LRR3; POT =RSR; BRU SETSA

*       QUEUE ROUTINES
*
ANN     EQU 1
        IF ANN
* CHECK NUMBER OF ITEMS ON QUEUE AGAINST COUNTER
* INPUT: X=QUEUE (QI0,X)
QCK     ZRO; STX QCK5; CLA; STA QCK2
QCK4    SKN PNEXT,2; BRU QCK3; MIN QCK2; LDX PNEXT,2; BRU QCK4
QCK3    LDX QCK5; LDA PNX3,2; SKE QCK2; BRM MONCR; BRR QCK
QCK2    ZRO 0   COUNTER
QCK5    ZRO 0   QUEUE
*
* INPUT: A=PACPTR, X=QUEUE
* OUTPUT: A=PACPTR
QPUT    ZRO; STA QPUT4; STX QPUT5; CXA; SUB =PNEXT; CAX
        BRM QCK; LDA QPUT4; LDX QPUT5
        DIR; LOGO 36B,L0GX; LOGO 37B,QPUT; EIR
        LDB 2,2; MIN 3,2
        COPY AX,BA,XB; XMA PNEXT,2; SKE ZERO; BRM MONCR; COPY BX,XA
        LDB 1,2; STA 1,2
        CBX; STA PNEXT,2
        LDA QPUT5; SUB =PNEXT; CAX; BRM QCK; LDA QPUT4; BRR QPUT
QPUT4   ZRO 0   PACPTR
QPUT5   ZRO 0   QUEUE
```

```
*
* INPUT: A=PNEXT OF FORK TO GET, B=QUEUE (QIOQ), X=PREVIOUS PACPTR (QIOQ)
QGET    ZRO; STA QGET2; STB QGET3; STX QGET4; CBX; BRM QCK
        LDB QGET3; LDX QGET4
        LDX PNEXT,2; DIR; LOGG 42B,LOGX; EIR
        STX QGET8
        CLA; XMA PNEXT,2; XMA QGET2; SKE QGET2; BRM MONCR
        LDX QGET4; XXB; SKR PNX3,2; BRU *+2; BRM MONCR; XXB
        DIR; LOGG 40B,LOGB; LOGG 41B,QGET; EIR
        XMA PNEXT,2; STA QGET7; LDA PNEXT,2
        XXB; SKG ZERO; BRM QGET5
        STX QGET6; SUB =PNEXT; SKE QGET6; BRM MONCR
        STB PNXTP1,2; BRM QGET5
QGET5   ZRO; LDX QGET3; BRM QCK; BRR QGET
QGET2   ZRO 0   INPUT A=PNEXT
QGET3   ZRO 0   INPUT B=QUEUE (QIOQ)
QGET4   ZRO 0   INPUT X=PREVIOUS PACPTR (QIOQ)
QGET6   ZRO 0   QUEUE
QGET7   ZRO 0   PEVIOUS PNEXT
QGET8   ZRO 0   PACPTR
        ENDF
*
        IF -ANN
*
* INPUT: A=PACPTR.  X=QUEUE.
* OUTPUT: A=PACPTR
QPUT    ZRO
QPUT3   DIR; LOGG 36B,LOGX; LOGG 37B,QPUT; EIR
        LDB 2,2; MIN 3,2; XXA
        STB PNEXT,2; XXA; LDB 1,2; STA 1,2
        CBX; STA PNEXT,2; BRR QPUT
        ENDF
*
* PUT JOB AT TOP OF QUE
* INPUT: A=PACPTR
QTPUT   ZRO; DIR; LOGG 45B,QTPUT; EIR
        CAX; LDA QUE; STA PNEXT,2; STX QUE; MIN QUEC; BRR QTPUT
*
        IF -ANN
```

```
* INPUT: A=PNEXT OF FORK TO GET, B=QUEUE (QIBQ), X=PREVIOUS PACPTR (QIBQ)
QGET    ZRB; XXB; SKR PNX3,2; BRU *+2; BRM MONCR; XXB
        DIR; LOGG 40B,LOGB; LOGG 41B,QGET; EIR
         STA PNEXT,2; XXB; SKG ZERO; BRR QGET; STB PNXTP1,2; BRR QGET
        ENDF
*
* INPUT: A=NEW ACT. COND.  X=PACPTR.
* QSCH CHANGES ACTIVATION CONDITION AND TAKES PAC ENTRY OFF
*   THE QUEUES.
QSCH    ZRB
        STA QSCH1; STX QSCH4
        LDA =QTI; STA QSCH6; LDA QSCH1
        XMA PTEST,2; EOR =700000B; SKA =7700000B; BRU *+2
        BRR QSCH (OLD ACT. COND. WAS 7. FORK NOT ON QUEUES.)
        LDA PNEXT,2; STA QSCH1; LDA QSCH6; BRU QSCH5
QSCH2   SKG ZERO; BRU QSCH3
        LDA =QD; ADM QSCH6; LDA QSCH6; SKG =QQE; BRU QSCH5; BRM MONCR
QSCH5   SUB =PNEXT; CAB
QSCH3   CAX; LDA PNEXT,2; SKE QSCH4; BRU QSCH2
        LDA QSCH1; BRM QGET; LDX QSCH4; BRR QSCH
QSCH1   ZRB 0    ACT. COND./ PNEXT
QSCH4   ZRB 0    PACPTR
QSCH6   ZRB 0
*
* CORE USE METER FOR COMPUTE JOB.
* NPC INPUT, A = NEW PAGE COUNT, X = REMAINING LONG TIME QUANTUM.
* SJB INPUT, A = REMAINING TIME QUANTUM. OUTPUT, A = PAGE*TICKS.
* CALL NPC EVERY TIME NUMBER OF PAGES CHANGE.
* CALL SJB AFTER EVERY COMPUTE JOB TO RESET METER AND GET TOTAL.
$NPC    ZRB; XMA NPCA; STX T2; SUB NPCA; MUL T2; LSH 23
        ADM NPCB
        IF PTICKB
        LDA T2; SKG TRQ; BRU *+2; BRM MONCR; STA TRQ
       ENDF
        BRR NPC
SJB     ZRB; COPY AX,A; BRM NPC
        CLA; XMA NPCB; CMA; LDX JOB; ADM PTCKS,2; ADM SPT
       IF PTICKB
        LDB XX; STB TRQ
```

```
        ENDF
         BRR SJB
        IF PTICKB
TRQ      DATA    377777778
        ENDF
NPCA     ZRB
NPCB     ZRB
**
* PUT COMPUTE JOB ON QUE AND RELEASE COMPUTE JOB PAGES.
RCJ      ZRB; LDA =QQC; LDB =QQCQ; CBx; BRM QGET
         LDA PACBMP; LDX =QQE; BRM QPUT
         LDA CMRRL3; XMA RRL3; LRR3; PBT RRL3; LDB LQ; STA RRL3
         LRR3; PBT RRL3; CBA; BRM SJB
         LDX PACBMP; LDA PTAB,2; LRSH 15; ETR K77; STA JBB
         COPY Ax,xB; LDA X4; BRM RELPG; CLA; STA PACBMP; STA RAJBB2
         LDA BNE; STA CMPST; BRR RCJ
*
*        INTERRUPT LOGIC
*
* IIR     MAKES ACT. COND. A PROGR. INT. AND PUTS FORK ON QTI
* INPUT: A=INT. MASK.    X=PACPTR.
IIR      ZRB; SKA PIM,2; BRU *+2; BRR IIR
         STA IIR1; STX IIR2; EOR PIM,2; STA PIM,2
         CLEAR; LDA IIR1; NOD 24; CxA; SUB =500000B-2; CNA
         STA IIR4; LDX IIR2; LDA PIM,2; SKA X1; BRR IIR (NON-TERM.)
         LDA PTEST,2; SKE =700004B; BRU IIR3; LDA PPTR,2; RSH 12
         CAx; BRM DFK; ERU IIR5
IIR3     LDA PPTR,2; SKA PLMSK; BRU *+2; BRU IIR5; RSH 12; CAX
         LDB THREE; BRM RFK; LDA =DFK; BRM SCFK
IIR5     LDA IIR4; LDX IIR2; BRM QSCH
         LDX =QTI; LDA K2B6; SKA IIR1; LDX =QTI; LDA IIR2
         BRM QPUT; LDx IIR2; MIN IIR; BRR IIR
IIR1     ZRB 0    INT. MASK
IIR2     ZRB 0    PACPTR
IIR4     ZRB 0    ACT. COND.

* IIR FOR RUNNING FORKS
$RIIR    ZRB; SKA PIM,2; BRU *+2; BRR RIIR; STA IIR1; STX IIR2; EOR PIM,2
```

```
        STA PIM,2; CLEAR; LDA IIR1; NOD 24; CXA; ADD TWO; CNA; SKE FIVE
        BRU *+2; BRM MONCR; LDB 0; COPY AX,BA; ETR ADMSK;
        LDX 2000B,6; COPY XA,AB; LDX 0,6; STB 0,6; ETR ADMSK
        ADD X4; STA 0; BRU POPX

*
* MONITOR CRASH
MONCR   ZRO
        SKS 20002B; MIN MCRI; CKF; EOM 20004B (DIR)
        EOD 12000B; PIN T3
        EOD 0; EOM 0
        STA MCRA; STB MCRB; STX MCRX; MIN MCRC
        LDA MCRC; SKE ONE; BRU CRA2
        LDA 0; STA MCRO
*       COPY USER PAGES 0-3 TO PAGES 13,15,16,17 AND TS TO PAGE 14
        LDA =1300B; LDB RRL1
CRA1    ETR =37B2; STA STOP
        SKE =14B2; BRU *+3; LDA KLTS; BRU *+2; LSH 6; ETR K37
        ADM STOP; LRR3; POT STOP
        SKG ZERO; BRU *+5; LDX KM4B3; LDA 0,2; STA 34000B,2; BRX *-2
        LDA STOP; ADD K100, SKG K1777; BRU CRA1
CRA2    LDX =100077B; BRM STOP

STOP    ZRO; STX 0,2; BRU 0,2

CRASH   BRM MONCR
CRA5    ZRO     0                       TS BLOCK RELABELLING
$MCRA   ZRO 0
$MCRB   ZRO 0
$MCRX   ZRO 0
MCRO    ZRO 0
MCRI    ZRO 0   1 IF ENABLED
MCRC    ZRO 0
$QTIGO  BSS 32
$Q25    DATA 25B+QTIGO   BRS 44 FLAG
$Q26    DATA 26B+QTIGO   RFK FLAG
ENDPAC  BSS 0
        FORGET
        END
```

```
B       IDENT                                                   6/21/72  15:02     PAGE   1

        LISTM

BUG     EQU     1               CHECK MEMORY TRAP BUG

* ENTRY POINTS
        ENTRY   TFC02
        ENTRY   CPUP,IOP
        ENTRY   T,T1,T2,T3,T4,BSX,SS01,SS02,SS03,BST,EPOPX
        ENTRY   MGET,TRAPS
        ENTRY   TRAPI,TRAPM,TRAPR
        ENTRY   RFK,TFK,FKSTW,HFK,GFK,SCFK,DFK
* ENTRIES FROM MDBG
        ENTRY   PB,PX,PPB,NFBRK,DBTOP

*
* 'SBRM','SBRR'   6/28/66
*
* SBRM  TIME=43.75 US
SBRM    PBPD    57685
$SBRME  STX SS03; EAX* 0; XXA; ETR ADMSK
        MRG =40040000B; XMA 0; EOR =40000B; STA* 0
        CXA; LDX SS03; BRR 0
*
SBRR    PBPD    571B5
$SRB    STX SS03; LDX* 0; EAX 0,6; STX T; LDX SS03; BRR* T (TIME 28 US)
*
*
* 'BRS'  9/27/65
*
* THIS ROUTINE DISPATCHES ON THE ADDRESS OF A 'BRS'
*
T       ZRO;* UNIVERSAL TEMPORARY STORAGE
T1      ZRO
T2      ZRO
T3      ZRO
T4      ZRO
BSX     ZRO;* BRS EXIT
```

```
SS01    ZR0;* SAVE (A)
SS02    ZR0;* SAVE (B)
SS03    ZR0;* SAVE (X)
*
BRS     P0PD    57385
$BS     STA SS01; STB SS02; STX SS03; EAX* 0
        C0PY xA,B; SKA =37600B; BRU BSFT
BS1     LDA BST,2; SKG PRILEV; BRU BS3; TRAP 1
BSFT    ETR ADMSK; SKG =BSTU-BST; BRU BS1; TRAP 2

*       TAG     STATUS REQUIRED
*       0       N0 STATUS
*       ,1      SUBSYSTEM
*       ,2      SYSTEM
*       ,3      EXEC
*       PRILEV IS THE F0RK LEVEL. PRILEV IS HIGH F0R HIGH STATUS F0RKS.
* TYPES:  0P C0DE  TYPE
*           0      TRAP
*           1      M0NIT0R WITH X=PACPTR
*           2      EXEC
*           3      RUP BRS'S
*           4      M0NIT0R WITH MAP DISC
*           5      M0NIT0R WITH MAP W
*           6      M0NIT0R WITH X=SS03
*           7      M0NIT0R WITH MAP DISC AND X=SS03
*          10      M0NIT0R WITH MAP W AND X=SS03
*          11       M0NIT0R WITH MAP TTY BUFFERS AND X = SS03
BST     ZR0 0 (0); 7 BRS1,3; 7 CCLS,3; 1 PMTI
        1 MPT (4); 1 FKSTAT; 2 21; 6 BRS7
        7 BRS8,3 (8); 1 FKST; 1 PPAN; 11B BRS11
        11B BRS12 (12); 11B BRS13; 11B BRS14; 2 0
        2 1 (16); 7 BRS17; 2 3; 2 4
        7 BRS20 (20); 1 HFNA; 1 NTERM,3; 1 ALTERM,3
        1 BRS24 (24); 4 GRABB2,3; 1 SKR0UT
        IF ST
        1 ST30,3 (27); 1 STST0P,3 (28)
        ELSE
        0 0 (27); 0 0 (28)
        ENDF
```

```
11B BRS29; 4 GIVEB,3; 1 FKWT
6 CLM2,3 (32); 3 GETSTR; 3 BUTMSG; 3 BUTSTR
3 BUTNUM (36); 2 20; 3 GETNUM; 1 RCPARW
11B BRS40 (40); 7 LBCBLK,2; 1 RREAL; 1 RDRL
1 STRL (44); 6 BRS45; 1 NRBUT,3; 1 SRBUT
2 6 (48); 1 SRIR; 1 FFIX; 1 FFLT
2 26+5000B (52); 2 27+5000B; 4 GRABB1,3; 6 MAYDMS
1 PMTR,3; 1 MXSA,3; 2 2; 2 5
2 9 (60); 0; 2 16; 2 17
2 18 (64); 2 19; 7 BRS66,3; 2 22
2 14 (68); 2 15; 1 FRPMT; 1 SKXEC
6 EXDMS,3 (72); 1 RDCLEC; 0 0; 6 XPOP
0 0 (76); 7 BRS77; 1 SAIR; 0
1 MBRB (80); 6 BRS81; 1 SKFNE; 1 SKFZE
1 SKFUZE (84); 11B BRS85; 11B BRS86; 6 BRS87,1
1 RTEX (88); 1 RESMT; 1 DFR; 2 28+1000B
7 BRS92 (92); 1 RSMET,3; 1 CLFX,2; 2 7,3
2 8 (96); 5 RESBRS; 5 INCBRS; 5 REDBRS
1 AST (100); 1 UAST; 5 TREAD; 5 TWRT
1 WHT (104); 5 CTRL; 5 PRD; 1 SETPAR
5 TSTD2 (108); 1 DMS; 5 TSTRDY; 1 BRSRET,3
6 BRS112,3 (112); 6 BRS113; 6 BRS114; 1 EXRETE,3
1 RURL (116); 1 SURL; 1 LDFLE; 1 STFLE
1 APMTE,3 (120); 1 DPMTE; 4 ARDD,2; 4 AWDD,3
4 ARD,2 (8E+1,124); 4 AWD,3; 6 POPXO; 1 PEBRS,2
7 SDBM,3 (8E+5,128); 11B BRS129,3; 1 BPTEST; 1 CRASH,3
11B BRS132 (132); 0; 1 CRSW; 1 BRS135
6 SETSW,3 (8E+13,136); 11B BRS137; 6 BRS138,2; 11B BRS139
2 29+1000B (8E+17,140); 2 11,3; 2 10+3000B; 1 BMTST
4 EXBGET,3 (144); 4 EXBPUT,3; 1 CS,2; 7 BRS147
1 FFNAE (148); 1 FFIXE; 1 FFLTE; 7 BRS151
11B BRS152; 6 BRS153; 6 BRS154; 11B BRS155
11B BRS156 (156); 7 BRS157,2; 1 BS158; 11B BRS159,3
11B BRS160; 11B BRS161; 11B BRS162; 11B BRS163,3
11B BRS164; 0; 11B BRS166,3; 6 BRS167
11B BRS168; 11B BRS169; 2 12+3000B (170); 2 13+3000B (171)
11B BRS172; 2 23+4000B; 7 BRS174; 1 BRS175
2 30+1000B (176); 11B BRS177,2; 11B BRS178
```

```
H3      EQU *
$BSTU   BRM TRAP
BS3     ETR K57; STA BSX; LCY 9; CBX; EXU **+1,2
H4      BRM TRAP; BRU BS2; BRU BS4; BRU BS6; BRU BS10; BRU BS12
        BRU BS14; BRU BS16; BRU BS20B; BRU BS22

BS2     LDA SS01; LDB SS02; LDX PACPTR; BRU* BSX
$BS4    LDA 0; STA SBRSRT; BRU UBRSET
BS6     LDA =UBRSET; XMA 0; STA SBRSRT
BS14    LDA SS01; LDB SS02; LDX SS03; BRU* BSX
BS10    BRM MPDSC; BRU BS2
BS12    BRM MPXB; BRU BS2
BS16    BRM MPDSC; BRU BS14
BS20B   BRM MPXB; BRU BS14

BS22    BRM MPTYM; BRU BS14

*
* CLASS 3 BRS SETUP AND RETURN ROUTINE
* EXEC BRS'S
*
UBRSET DIR; LDGG 50B,J0B; EIR; LDA NFORK; SKG ZERO; TRAP 5
        BRM GFK; BRU FKSTW; BRM STFK
        LDX PACPTR; LDA RL1,2; STA UBRL1; LDA RL2,2; STA UBRL2
        LDA BRSTVA; STA 0; LDA BSX; RCY 9; ETR K37; CAX
        LDA BRSTV,2; CLB; LRSH 18; MRG =NCMEM*1B6+1B4+12B2
        LDX FK04 (NEW PACPTR); STA RL1,2; STB RL2,2
        LDA =700B+2B6; ADM PIM,2
        LDA BSX; ETR K777; ADD 0; STA PL,2
        LDA =EXECL; RSH 15; LDA J0B; LSH 15; MRG X7; STA PTAB,2
        LDX STFK2 (NEW XPB)
        LDA SS01; STA PA,2; STA UBA
        LDA SS02; STA PB,2; STA UBB; LDA SS03; STA PX,2; STA UBX
        LDA SBRSRT; STA 0; STA UPL
        LDB =700004B; BRM FK30

*
```

```
* CLASS 2   MONITOR BRS'S
* SAVE RETURN
*
*
* RETURN FROM CLASS 2 BRS'S
EPOPX  SKN TIME; BRR SBRSRT
       SKN TTIME; SKN ACTR; BRU *+2; BRR SBRSRT
       STA SS01; STB SS02; STX SS03; LDA SBRSRT; STA 0; BRU PACQM1
*
*
* BRS 111   RETURN FROM EXEC BRS'S  (CLASS 3/4)
BRSRET DIR; LDBB 510,JBB; EIR; LDA PPTR,2; MRG PLMSK; CAX
       LDA PTEST,2; SKE =7000046; TRAP 6
       LDA UPL; STA PL,2; MIN PL,2; STX FK04
       LDX PACPTR; BRM OFK
       BRM FK682

* BRS 115   ERROR RETURN FOR EXEC BRS'S
EXRETE LDA PPTR,2; MRG PLMSK; CAX; LDA PTEST,2; SKE =700004B
       TRAP 7; SUB THREE; STA PTEST,2; LDA UPL; STA 0
       STX EXRA; CLA; LDB PACPTR; LDX JBB; BRM RELPG
       LDX PACPTR; BRM OFK; LDX EXRA; LDA PIM,2; LRSH 3; ETR SEVEN
       MRG X4; STA XPL; CLB; STB PACJBB
       BRM STSTAT; STX PACPTR; STX PUPAC; DIR
       LDX PUBPTR
EXR2   CXA; SKE =PUBPTR; BRU EXR3; LDA UTTY; CAX; MRG =3B5
       CAB; LDA TIC1; BRM EPU; MIN ACTR; BRU P0PR
EXR3   LDA 1,2; LDB 2,2; LDX 0,2; SKE TIC1; BRU EXR2; CBA
       ETR X77; SKE UTTY; BRU EXR2; EIR; BRU P0PR
EXRA   ZRB 0

* BRS 46   SET NON-TERMINABILITY
NROUT  LDA PIM,2; MRG X1; STA PIM,2; BRU P0PX
* BRS 26   SKIP IF TERMINATION PENDING
SKROUT LDA PIM,2; SKA X2; BRU SKR2
       LDX UTTY; LDA =SOFTY+HARDY; SKA TF2,2
SKR2   MIN 0; BRU P0PX
* BRS 47   CLEAR NON-TERMINABILITY
SROUT  LDA PIM,2; SKA X1; BRU *+2; BRU P0PX
```

```
        ETR =47777777B; XMA PIM,2; SKA X2; BRU PACQE; BRU P0PX                B              PAGE    6
* BRS 90   DECLARE FORK FOR RUBOUT
DFR     CXA; LDX UTTY; STA TTYASG,2; BRU P0PX
        IF ST
STG0    EQU *   START STATISTICS
        LDA =RSR; BRM RLABEL; BRU STG01
$SP     ZR0;* STATISTICS POINTER. POINTS INTO TABLE IN REAL PAGE 7
$SP1    ZR0;*   POINTS TO NEXT STATISTICS BUFFER TO BE DUMPED
$STC    ZR0;*   MINNED AND STORED INTO EACH BUFFER AS A BUFFER NUMBER
$STSW   ZR0;* -1 IF STATISTICS ARE RUNNING. 0 IF STAT NOT RUNNING
$STDISC ZR0;* DISC POINTER FOR STATISTICS
$SSRET STA RRL3; LRR3; POT RRL3; BRU 1,2  (RETURN FROM SSS)

STSTOP EQU *  STOP STATISTICS
        LDA =RSR; BRM RLABEL; BRU STSTP1
        ENDF
*  READ AND CLEAR ERROR CODE.
RDCLEC LDA ERCODE; BRU P0PXB

*       MEMORY ALLOCATION LOGIC
*

* ASSIGN A POSITION IN PMT
* RETURNS: NO SKIP=NO BYTES. SKIP=BYTE NO. IN X
PMGET   ZR0; CLA; LDB KM1; MIN PMGET; BRM PMSCH; SKR PMGET; BRR PMGET

PMSCH   ZR0; LDX =NUMEM-1; STX PMG5; LDX =NCMEM-1
MGET1   EAX 1,2; SKM* PMTJOB; BRU MGET4; MIN PMSCH; BRR PMSCH
MGET4   SKR PMG5; BRU MGET1; BRR PMSCH
PMG5    ZR0 0
*
*
* GET A BLOCK OF MEMORY
* INPUT: A=CORE ADDRESS IN PAGE. INDIRECT BIT IF MEMORY MUST
*          COME FROM AN UPPER FORK.
*        X=PACPTR
MGET    ZR0; BRM MX01; SKE ZER0; BRU MX03
        LDA KM2; STA MX09; LDX MS03
* SCAN FOR LOCAL OR FIXED MEMORY FORK
```

```
MGET11 STX MX08; LDA RL1,2; LDB RL2,2; LDX MX07; MIN MX09
       LCY 0,2; SKA K7786; BRU MGET12; LDX MX08
       SKN PIM,2; BRU *+2; BRU MX03; SKN PTAB,2
       BRU MGET13 (CHECK HIGHER FORKS FIRST)
       LDA MX14; SKA =400008; BRU MX03
       MIN MX09; BRM PMGET; BRU MX03
       STX MGTS2; BRM PMTA; LDX MS03
* PROPAGATE NEW BYTE AS NECESSARY
MGET10 STX MX08; LDA RL1,2; LDB RL2,2; LDX MX07
       LCY 0,2; RCY 18; MRG MGTS2; LCY 18; RCY 0,2
       LDX MX08; STA RL1,2; STB RL2,2
       LDA PPTR,2; MRG PLMSK; CAX; SKR MX09; BRU MGET10
* SET UP NEW MAP
       LDX MS03; BRM CHREL; BRM SWAPI
       SKN PACJOB; BRU MGET20; STA CMRRL1; STB CMRRL2; STX CMRRL3
MGET20 BRM LABEL; BRU MX04
MGET12 LRSH 18; STA MGTS2; CAX; BRM PMTA; LDX MS03; BRU MGET10
MGET13 LDA PPTR,2; MRG PLMSK; CAX; SKA PRMSK; BRU MGET11; BRU MX03
*
* ASSIGN BLOCK IN PMT
PMTA   ZRB; LDA JOB; STA BJOB
       CXA; ADD PMTJOB; ETR ADMSK; MRG =41B6; MIN RAJOB3
       DIR; BRM B; EIR
       LDA =WETTB; XMA TJOB; STA PMTA2; LDA RAJOB3; SKE ZERB
       BRU *-2; LDA PMTA2; STA TJOB; MIN* TJOB; BRR PMTA
PMTA2  ZRB 0
*
MGTS2  ZRB 0
*
* RELEASE MEMORY. USED BY BRS 4.
MPUTA  ZRB; BRM MX01; SKG =NCMEM-1; TRAP 8
       CAX; LDA* PMTJOB; SKE ZERB; BRU *+2; TRAP 9
       SKA X2; BRU *+2; BRU MPUT2
       CXB; LDX MS03; EXU EX; CBX
MPUT2  BRM MDEL, BRU MX04
*
* INPUT: X=RELABELING BYTE.
MDEL   ZRB; LDA JOB; SKE BNE; BRU *+5; DIR; LOGG 55B,LOGX; EIR
       COPY XA,B; STA T2; STB T3; ADD PMTJOB; STA T
```

```
        CAX; LDB 0,2; STB HLDC
RX33    DIR; BRM RES; BRU RX32
        EIR
        LDX T; CLA; XMA 0,2; XMA T2; LDX JØB; CAB; SKE RL3,2; BRU RXNTS
        CLA; STA RL3,2; STA XPB; LDX PACPTR; LDA PTAB,2
        ETR =67777777B; STA PTAB,2
RXNTS   LDA PACPTR; BRM MDELC; SKN T2; BRU RRRL (NØT SHARED)
RX6     LDB =7737777b; LDA =1285; ADD T; LDX =-NUMEM*NJØB1
RX7     SKM PMTE,2; BRU RX5
        CØPY XA,B; STB PMTE,2; SUB =-NUMEM*NJØB1
        RSH 23; DIV =NUMEM; CØPY AX,BA; ADD =NCMEM; STA T4
        EAX 1,2; LDX TINØ,2; LDX TTYASG,2
RX4     LDA PPTR,2; SKA KS4B4; BRU Rx3; CXA; LDB T4
        BRM MDELC; BRU RX6
RX5     BRX RX7
RRRL    LDX PACPTR; BRM CHREL; BRM SWAPI
        SKN T3; BRU RX12; SKN PACJØB; BRU RX13
        STA CMRRL1; STB CMRRL2; STX CMRRL3
RX12    BRM LABEL; SKN PACJØB; BRR MDEL
        LDA SWIN14, STA CJP; LDX TTIME; BRM NPC; BRR MDEL
RX3     RSH 12; CAX; BRU Rx4
RX13    BRM LABEL; LDA RAJØB2; SKE ZERØ; BRU *-2
        SKN RAER2; BRR MDEL
        LDX PACØMP; BRM SETSWP; BRM SWAPI
        STA CMRRL1; STB CMRRL2; STX CMRRL3
        LDA SWIN14; STA CJP; BRR MDEL
RX32    EIR; HLT; LDA T; BRU RX33
MDELC   ZRØ; SKE PACØMP; BRU RX11; SKR T3
RX2     MRG KS4B4
RX11    CØPY AX,BA; LDB THREE; STB T1
        LDB KM1; STB RLCT; LDB K77
RX1     SKM RL1,2; BRU *+5; CNA; ADM RL1,2; CNA; MIN RLCT
        SKM RL2,2; BRU *+5; CNA; ADM RL2,2; CNA; MIN RLCT; LCY 6
        SKR T1; BRU RX1; SKN RLCT; BRU RLHØLD
RX8     LDA PPTR,2; SKA K7777; BRU Rx2; BRR MDELC
RLHØLD  CXA; SKE PACPTR; BRU RLHLDB
RLHLDA  STA HLDA; STB HLDB
RLHLDD  LDA HLDC; ETR =7785; SKE =1285; BRU RX8; LDA HLDC
        DIR; BRM REL; EIR; SKR RLCT; BRU RLHLDD
```

```
        LDX HLDA; LDB HLDB; BRU RX8
RLHLDB  SKE PACOMP; BRU *+2; BRU RLHLDA; SKE PACSW; BRU RX8; BRU RLHLDA
RLCT    ZR8
HLDA    ZR8
HLDB    ZR8
HLDC    ZR8


* COMMON ENTRY FOR MGET,MPUT,CHKRO
* GET RELABELING BYTE INTO A
* INPUT: A=ADDR. IN PAGE, X=PACPTR
MX01    ZR8; STA MX14; ETR =34000B; STA MS01; STX MS03
        LDX MX01; LDB -1,2; STB MX00; LRSH 11; STA MX05; MUL THREE
        LDX MS03
TTOM    LDA RL1,2; LDX RL2,2; XXB; STX MX07
        RCY 18; LCY 0,2; ETR K77; STA MX06; BRR MX01
*
MX04    MIN MX00
MX03    LDA MS01; LDX MS03; BRR MX00
*
MS01    ZR8     0               ADDR. OF 1ST WORD IN PAGE
MS03    ZR8     0   PACPTR
MX00    ZR8     0   RETURN FOR MGET OR MPUT
MX05    ZR8     0   PAGE NUMBER
MX06    ZR8     0   RELABELING BYTE
MX07    ZR8     0   RELABELLING BYTE INDEX
MX14    ZR8     0   IND. BIT=1 IF MEMORY COMES FROM UPPER FORK.
*
MX08    ZR8
MX09    ZR8


*
* 'MBRO','APMTE','OPMTE','MPT'  6/30/66
*
* BRS'S FOR MODIFYING THE MEMORY TABLES
*
*
* BRS 80   MAKE BLOCK READ ONLY
```

```
* INPUT: A=PMT NO.  IF A LT 0, MAKE FILE RO.  OTHERWISE, READ-WRITE
* OUTPUT: A=FORMER STATE OF PMT ENTRY
MBR9    ETR K77; STA MBR06; BRM CRTA; TRAP 10; STA CRTA
        SKA X2; EXU EX; LDA MBR06; SKG =NCMEM-1; EXU EX
        LDA 0,2; SKN SS01; BRU MBR03
        MRG X1; STA 0,2; LDX PACPTR; BRM CHREL; BRM SWAPI
        SKN PACJOB; BRU MBR07; STA CMRRL1; STB CMRRL2; STX CMRRL3
MBR07   BRM LABEL; BRU MBR05
MBR03   ETR =67777777B; STA 0,2
MBR05   LDA CRTA; LCY 2; ETR X4; XMA SS01; ETR K77
        ADM SS01; BRU POPX
MBR06   ZRE 0
*
$QS     LDA SQEC; ADD QQCC; CAX; LDA QTIC; LDB QIOC; BRU XPOP
*
* BRS 94    CLEAR FORKING STRUCTURE
CLFK    LDX JOB; SKN TTNO,2; BRU *+2; BRM TRAP; LDX PACPTR
        BRM HFK; MIN PL,2; STX FK04; LDA PPTR,2; RSH 12; CAX
        LDA =DFK; BRM SCFK; CLX;  STX PX,2; BRM FKG03
*
* BRS 93  RESET METERS.
RSMET   CLA; STA DSKMET; LDX JOB; STA SWPCNT,2; STA ETTB,2
        STA PTCKS,2; STA CCT,2; BRU POPX
* BRS 89, GET METER READINGS FOR SESSION.
* A= ADDRESS FOR READINGS.
* DISK USE, SWAP COUNT, CHARACTER COUNT, PAGE TICKS
* CPU TIME, CONNECT TIME.
RESMT   BRM CTRU; LDB SWPCNT,2; LDX SS01; STA 4,6; STB 1,6
        LDX JOB; LDA CCT,2; LDB PTCKS,2; LDX SS01; STA 2,6; STB 3,6
        LDA DSKMET; STA 0,6; LDA REAL; SUB UREAL; STA 5,6
        BRU POPX
RESMT2  DATA 7
* OUTPUT FROM CTRU.  A=EFFECTIVE CPU TIM, X=JOB.
        IF -RP
$CTRU   0; LDX JOB; LDA CCT,2; MUL CTRU1; STA CTRU2
        LDA DSKMET; MUL RESMT2; LSH 23; ADD ETTB,2
        ADD CTRU2; BRR CTRU
CTRU1   DATA 3000006;* S.B. 02314632 FOR .05/KCH.
CTRU2   DATA 0
```

```
        ELSE
$CTRU   0; LDX J0B; LDA SWPCNT,2; MUL KP96; ADD ETTB,2
        STA CTRU1; LDA CCT,2; MUL KP24; ADM CTRU1
        LDA PTCKS,2; MUL KP06; ADM CTRU1
        LDA DSKMET; MUL SIX; LSH 23; ADD CTRU1; BRR CTRU
CTRU1   DATA 0
KP96    DATA 365605108
KP24    DATA 075341228
KP06    DATA 017270248
        ENDF

* BRS 175, SAME AS BRS 89 WITH ETTB AS SEVENTH WORD
BRS175  LDX J0B; LDA ETTB,2; LDX SS01; STA 6,6; BRU RESMT

* BRS 88, READ CPU TIME
RTEX    BRM CTRU; BRU XP0PX
*
* BRS 56   MAKE EXPANDED SMT BYTE AVAILABLE TO USERS.
* INPUT: A=BYTE NO.
MXSA    SKG =LESMT; SKG =FESMT; TRAP 11; CAX; LDA X2
        MRG EXSMTD,2; STA EXSMTD,2; BRU P0PX
*
*
* BRS 56   MAKE POINTER INDIRECT FOR RECEIVER
* INPUT: A: BIT 0=1 FOR READ ONLY. BITS 17-23=PMT BYTE
*           NUMBER FOR PMT OR SMT BYTE POINTED AT.
*        B: BITS 18-23 SPECIFY THE PMT NO.
*        X: BITS 18-23 ARE CHANNEL NUMBER FOR SECOND BYTE.
* OUTPUT: A=USERS PMT BYTE NUMBER.
PMTR    CBA; ETR K77; SKG =NCMEM; TRAP 12; CAX; LDA* PMTJ0B
        SKE ZERO; TRAP 13; STX PMI2; BRU PMI11

* BRS 3   MAKE POINTER INDERICT
* INPUT: A: BIT 0 = 1 FOR READ ONLY. BITS 17-23 = PMT BYTE
*           NUMBER FOR PMT OR SMT BYTE POINTED AT.
*        X: BITS 18-23 ARE CHANNEL NUMBER FOR SECOND BYTE.
* OUTPUT: A = USERS PMT BYTE NO.
PMTI    LDA KM1; STA PMI2
PMI11   LDA SS01; ETR K377; SKG =NCMEM-1; BRU PMI3
```

```
        SKG =FESMT-1; BRU PMI6; SKN EXEC1; BRU PMI7; BRU PMI8
PMI6    EXU EX; LDX SS03; SKN TTYASG,2; TRAP 14
        LDX TTYASG,2; LDA PTAB,2; LRSH 15; ETR K77
        SKG =NJOB; BRU *+2; TRAP 15; CAX
        LDA SS01; ETR K77; SUB =NCMEM; ADD PMTP,2; BRU PMI4
PMI3    EXU EX; ADD =SMT; BRU PMI4
PMI7    CAX; LDA EXSMTD,2; SKA X2; BRU *+2 (USER PAGE); TRAP 16; CXA
PMI8    SKN SS01; EXU EX
        ADD =EXSMT; SUB =FESMT; SKG =XSMTEM; BRU PMI4; TRAP 17
PMI2    ZERO 0   BYTE NO.
PMI4    CAX; LDA 0,2; ETR XX; SKG ZERO; TRAP 18; MRG X4
        STA 0,2; CXA; MRG =12B5; SKN SS01; BRU *+2
        MRG X1; LDX PMI2; SKN PMI2; BRU PMI5; LDB K1S7
        BRM PMSCH; BRU *+2; BRU PMIX; STA PMI2
        BRM PMGET; TRAP 19; LDA PMI2
PMI5    STA* PMTJOB
PMIX    STX SS01; BRU POPX
*
* BRS 120   ADD A PAGE FOR SPECIFIED PMT ENTY.
* INPUT: A=REL. BYTE
APMTE   SKG K77; SKG =NCMEM-1; TRAP 24; AXC
        SKE* PMTJOB; TRAP 25
        ADD ONE; STA* PMTJOB; BRU POPX
*
* BRS 121   DELETES PMT ENTRY
* INPUT: A=REL. BYTE
DPMTE   ETR K77; SKG ZERO; BRU POPX; SKG =NCMEM-1; TRAP 26  (SMT ENTRY)
        STA MDEL; BRM CRTA; BRU POPX
        CBX; SKA X2; BRU DPMTE1 (EXEC PAGE)
DPMTE2  LDX MDEL; BRM MDEL; BRU POPX
DPMTE1  EXU EX; BRU DPMTE2
*
* BRS 4   DELETES A PMT ENTRY. REMOVES PMT POINTER
*       FROM ALL OTHER FORKS.
* INPUT: A=ADDR. IN PAGE TO RELEASE.
MPT     BRM MPUTA; TRAP 27; BRU POPX
*
* COMPUTE RELABELING TABLE ADDRESS
* INPUT: A=PSEUDO-REL. BYTE
```

```
* OUTPUT: A=PMT ENTRY,  B=INPUT X,  X=PMT/SMT ADDRESS
* RETURN SKIPS IF PMT IS NOT EQUAL ZERO.
CRTA    ZRO; COPY XB,AX; SKG =NCMEM-1; BRU CRTA1
        EAX* PMTJOB; LDA 0,2
CRTA2   SKE ZERO; MIN CRTA; BRR CRTA
CRTA1   ADD =SMT; CAX; LDA 0,2; BRU CRTA2
*
* 'RDRL','RURL','STRL','SURL'  6/28/66
*
* READ AND SET RELABELING
*
* BRS 43    READ FORK RELABELING
RDRL    LDA RL1,2; LDB RL2,2; LDX SS03; BRU XPOP
*
* BRS 116    READ PROGRAM RELABELING FROM TS BLOCK
RURL    LDA UPRRL1; LDB UPRRL2; LDX SS03; BRU XPOP
*
* BRS 44    SET FORK RELABELING
* CHANGES REL. IF REMAINING TIME QUANTUM IS ADEQUATE.
* SRT STORES NEW RELABELLING. NSRT STORES OLD RELABELLING.
STRL EQU *
         STA STRL1; STB STRL2; SKN PIM,2; BRU *+2; TRAP 28
STRLQ   LDX =SRT; BRM UPRL44; LDA KM1; STA S44T5; STA S44T12
        LDX PACPTR; LDA RL1,2; LDB RL2,2; LDX =NSRT; BRM UPRL44
        LDX KM8; LDB =7785
S44T3   LDA SRT44E,2; SKE ZERO; BRU S44T7; SKE ENSRT,2; MIN S44T12
        BRU S44T4
S44T7   SKE ENSRT,2; BRU S44T2
S44T4   BRX S44T3
        SKN S44T5; BRU S44T13; SKN S44T12; BRU *+2; BRU POPX
* RELEASE OLD PAGES THAT ARE DIFFERENT
S44T27  LDX KM8
S44T30  LDA ENSRT,2; SKE ZERO; BRU *+2; BRU S44T31
        SKE SRT44E,2; BRU *+2; BRU S44T31; STX S44T10
        SKG =NCMEM-1; BRM S44T25; ADD PMTJOB; ETR ADMSK
        DIR; BRM REL; EIR; LDX S44T10
S44T31  BRX S44T30; DIR; BRM RAGO; EIR
        CLA; SKE RAJOB3; BRU *-1; SKN RAER3; BRU S44T33
        LDX PACPTR; LDA STRL1; LDB STRL2; STA RL1,2; STB RL2,2
```

```
* COMPUTE NEW REAL RELABELING
        LDA PMTJ0B; ETR ADMSK; STA SwPMT; CNA; ADD =SMT; STA SWSMT
        CLA; LDX KM2; STA SRTE,2; BRx *-1; BRM SWIN
        LDX RRL3; SKN PACJ0B; BRU S44T14
        STA CMRRL1; STB CMRRL2; STX CMRRL3
S44T14  EQU *
        BRM LABEL; SKN PACJ0B; BRU P0PX; LDA SWIN14; STA CJP
        LDX TTIME; BRM NPC; BRU P0PX
* CHECK LEGALITY
S44T2   MIN S44T5; STX S44T10; SKG =NCMEM-NUSMT; BRU S44T6
        SKG =NCMEM-1; BRM S44T25; ADD PMTJ0B
S44T9   CAX; LDA 0,2; SKE ZER0; BRU *+2; TRAP 29
        SKM =12B5; BRU S44T51; ETR ADMSK
        SKG =EXSMTE; BRU *+2; TRAP 30; SKG =EXSMT1; BRU S44T53
        SKN EXEC1; BRU *+2; BRU S44T61; STA S44T62; LDA* S44T62
        SKA X2; BRU S44T61; TRAP 31
S44T62  ZR0 0
S44T53  SKG =PMTE; SKG =PMTM1; BRU *+2; EXU EX
        SKG =USMT; SKG =SMT; BRU *+2; EXU EX
S44T61  LDA 0,2; SKA X1; BRU S44T51; EXU EX
S44T51  SKA K1B7; BRU S44T8; SKA K2B7; EXU NEXEC
S44T11  LDX S44T10; BRU S44T4
S44T6   SKN EXEC1; TRAP 33; ADD =SMT; BRU S44T9
S44T8   SKA K2B7; EXU NSYS; BRU S44T11
* CHECK FOR MISSING PAGES
S44T13  LDX KM8; CLAB; STA S44T12; DIR
S44T19  LDA SRT44E,2; SKE ZER0; BRU *+2; BRU S44T15; STX S44T10
        LDB =77B5; SKG =NCMEM-1; BRM S44T25; ADD PMTJ0B;
        CAX; LDA 0,2
S44T18  SKM =12B5; BRU S44T17; CAX; LDA 0,2; BRU S44T18
S44T17  ETR =77B5; LRSH 15; COPY AX,B; LDA SWLS; LSH 0,2; SKA X4
        MIN S44T12 (COUNT MISSING PAGES); LDX S44T10
S44T15  BRX S44T19
        LDA S44T12; SKN PACJ0B; BRU S44T20; ADM PG44
        SKN RAJ0B1; BRU S44T21; SKN PACSW; BRU *+2; BRU STDMS
        SKN ACTR; BRU STDMS
S44T21  LDA STRLC; SKG PG44; BRU STDMS; BRU S44T22 (BRING PAGES)
S44T20  LSH 1; SKG TIME; BRU *+2 (BRING PAGES); BRU STDMS
        LDX PACPTR; LDA K1B5; MRG PL,2; STA PL,2
```

```
* BRING PAGES
S44T22 LDX KM8; LDA Q25; STA RFR; LDA J0B; STA BJ0B
S44T23 LDA SRT44E,2; SKE ZER0; BRU *+2; BRU S44T24; SKE ENSRT,2
       BRU *+2; BRU S44T24; STX S44T10; SKG =NCMEM-1
       BRM S44T25; ADD PMTJ0B; ETR ADMSK; MRG =4186
       MIN RAJ0B3; BRM 6; LDX S44T10
S44T24 BRX S44T23; BRM RAG0; EIR
       BRJ S44T27
* READ ERR0R
S44T33 LDA KM1; STA RAER3; LDX KM8
S44T28 LDA SRT44E,2; SKE ZER0; BRU *+2; BRU S44T29; SKE ENSRT,2
       BRU *+2; BRU S44T29; STX S44T10
       SKG =NCMEM-1; BRM S44T25; ADD PMTJ0B; ETR ADMSK; MRG X4
       STA S44T43; LDB =7765; CAX; LDA 0,2
S44T41 SKM =1265; BRU S44T42; CAX; LDA 0,2; BRU S44T41
S44T42 ETR =7765; LRSH 15; COPY AX,B; LDA SWLS; LSH 0,2
       SKA X4; BRU S44T44; LDA S44T43
       DIR; BRM REL; EIR
S44T44 LDX S44T10
S44T29 BRX S44T28; DIR; BRM RAG0; EIR
* RELEASE REMAINING OLD PAGES
       LDX KM8
S44T34 LDA ENSRT,2; SKE ZER0; BRU *+2; BRU S44T35; SKE SRT44E,2
       BRU S44T35; STX S44T10; SKG =NCMEM-1; BRM S44T25
       ADD PMTJ0B; ETR ADMSK; MRG X4; DIR; BRM REL; EIR
       LDX S44T10
S44T35 BRX S44T34; DIR; BRM RAG0; EIR; LDX PACPTR
       LDA STRL1; LDB STRL2; STA RL1,2; STB RL2,2
       LDB STRL0; MIN 0; STB PACDMS; LDA PACPTR
       LDX =QTI; BRU P0PR4
S44T25 ZR0; ADD =SMT; MIN S44T25; BRR S44T25
* UNPACK RELABELING
* INPUT: A,B=RELABELING, X=TABLE ADDRESS
UPRL44 ZR0; STB SWT3; LRSH 18; STA 0,2
       CLA; LCY 6; STA 1,2; CLA; LCY 6; STA 2,2
       CLA; LCY 6; STA 3,2; LDA SWT3; LRSH 18; STA 4,2
       CLA; LCY 6; STA 5,2; CLA; LCY 6; STA 6,2
       CLA; LCY 6; STA 7,2; BRR UPRL44
* CHANGE RL1,RL2 AND DISMISS ON OLD QUEUE.
```

```
STDMS  EIR; LDB PACPTR; LDX J0B; LDA X4; BRM RELPG
       LDX PACPTR; LDA STRL1; LDB STRL2; STA RL1,2; STB RL2,2
       LDB STRLD; MIN 0
       STB PACDMS; LDA PACPTR; LDX =QSE; SKN SVST; BRU *+2; LDX =QSQ
       SKN PACJ0B; BRU P0PR5; LDX =QTI; BRU P0PR4
NSRT   BSS 8   UNPACKED RELABELING
ENSRT  BSS 0
S44T43 ZR0   0     TEMP STORE FOR ERRORS
S44T10 ZR0 0    TEMP STORE FOR X
S44T5  DATA -1    COUNT OF NEW NON-ZERO BYTES
S44T12 DATA -1    COUNT OF NEW 0 BYTES, COUNT OF MISSING PAGES
STRLD  248 PACDM0
STRLC  ZR0 15 NO. OF BRS 44 PAGES ALLOWED FOR QQC
STRL1  ZR0      0                NEW RELABELLING WORDS FOR BRS 44
STRL2  ZR0      0
*
* BRS 117 SET PROGRAM RELABELING IN TS BLOCK
SURL   CLA; LDB UEXFLG; LSH 2; CAX; LDA =7B6; LSH 0,2; ETR X7
       CAX; LDA SS01; LDB SS02
       BRM SCRL; TRAP 34; LDA SS01; LDB SS02
       STA UPRRL1; STB UPRRL2; BRU P0PX
*
       IF BRS137
*      BRS 137  -  STACK AND CHANGE PROGRAM STATE
*
STKPS  LDA PSDEX; SKG =MPSDEX-1; BRU *+2; TRAP 35
       BRM CMSTPS; LDA SIX; ADM PSDEX
       LDA SS01; ETR ADMSK; MRG X4; BRM CPSTMS
       SKR 0; NOP; LDX PACPTR; BRU STRL
*
*      BRS 138  -  UNSTACK AND CHANGE PROGRAM STATE
*
USTKPS LDA PSDEX; SKG ZERO; TRAP 36; BRM CMSTPS
       LDA PSDEX; ADD =PSSTK-6; BRM CPSTMS; STA STRL1; STB STRL2
       LDA PSDEX; COPY AB; SUB SIX; STA PSDEX
       ADD =PSSTK; COPY AB,BA; ADD =PSSTK; BRM CPSTPS
       LDX PACPTR; LDA STRL1; LDB STRL2; BRU STRL+2
*
*      BRS 139  -  READ PREVIOUS PROGRAM STATE
```

```
*
RPRPS   LDA SS01; ETR ADMSK; MRG X4; COPY AB; LDA PSDEX; ADD =PSSTK; BRM CPSTPS
        BRU POPX
*
*       BRS 140 - CLEAR PROGRAM STATE STACK
*
CLRPS   COPY A; STA PSDEX; BRU POPX
*
*       THE FOLLOWING ROUTINES BELONG IN B PACKAGE AFTER CODE FOR BRS 139
*
*       COPY MACHINE STATE TO PROGRAM STACK
*
$CMSTPS ZRO
        LDX PACPTR; LDA RL1,2; LDB RL2,2
        LDX PSDEX; EAX PSSTK,2; STA 4,2; STB 5,2
        LDA 0; STA 0,2; LDA SS01; STA 1,2
        LDA SS02; STA 2,2; LDA SS03; STA 3,2
        BRR CMSTPS
*
*       COPY PROGRAM STACK TO MACHINE STATE
*
$CPSTMS ZRO
        ADD K2B7; STA PSDEX1; COPY X
        LDA* PSDEX1; MRG X4; STA 0; EAX 1,2
        LDA* PSDEX1; STA SS01; EAX 1,2
        LDA* PSDEX1; STA SS02; EAX 1,2
        LDA* PSDEX1; STA SS03; EAX 1,2
        LDA* PSDEX1; EAX 1,2; LDB* PSDEX1
        BRR CPSTMS
*
*       COPY PROGRAM STACK TO PROGRAM STACK
*
$CPSTPS ZRO
        ADD =6+2B7; ETR =6B7+37777B; STA PSDEX1; COPY BA
        ADD =6+2B7; ETR =6B7+37777B; STA PSDEX2; LDX KM6
        LDA* PSDEX1; STA* PSDEX2; BRX *-2; BRR CPSTPS
$PSDEX1 ZRO
$PSDEX2 ZRO
        ENDP
```

```
* UNPACK RELABELING AND CHECK FOR LEGALITY.
* INPUT: A=RL1, B=RL2
* INPUT: X=X7 FOR EXEC, X3 FOR SYSTEM, X1 FOR SUBSYSTEM, 0 FOR USER
* OUTPUT: X=PACPTR IF IT SKIPS
* NO SKIP: A PMT ENTRY=0 OR USER TRIED TO RELABEL AN EXEC PAGE.
* SKIP: RELABELING OK.
SCRL    ZRO; STX STRL5; XXA; LSH 1; STA STRL52; LSH 1; STA STRL51
        LRSH 2; XXA; CLX; BRM UPRL; LDX KM10D
STRL3   LDA SRTE,2; SKE ZERO; BRU *+2; BRU STRL7; SKG =NCMEM+NUSMT
        BRU STRL6
STRL7A  BRM CRTA; BRR SCRL; CBX; STA STRL9
        LDS =7785; SKM =1265; BRU STRL8; ETR ADMSK
        SKG =EXSMTE; BRU *+2; BRM STRL53; SKG =EXSMT1; BRU STRL10
        SKN STRL51; BRU *+2; BRU STRL41; STA S44T62; LDA* S44T62
        SKA X2; BRU STRL41; BRM STRL53
STRL10  SKG =PMTE; SKG =PMTM1; BRU *+3; SKN STRL5; BRM STRL53
        SKG =USMT; SKG =SMT; BRU *+3; SKN STRL5; BRM STRL53
STRL41  LDA STRL9; SKA X1; BRU STRL8; SKN STRL5; BRM STRL53
STRL8   SKA X2; BRU STRL4
STRL7   BRX STRL3; MIN SCRL; LDX PACPTR; BRR SCRL
STRL4   SKA X1; BRU STRL11; SKN STRL5; BRR SCRL; BRU STRL7
* PAGE IS READ ONLY
STRL11  SKN STRL52; BRR SCRL; BRU STRL7
* CHECK FOR SUBSYSTEM
STRL6   SKN STRL51; BRR SCRL; BRU STRL7A
STRL5   ZRO 0   NEG. FOR EXEC
STRL51  ZRO 0   NEG. FOR SUBSYSTEM
STRL52  ZRO 0   NEG. FOR SYSTEM
STRL53  ZRO; BRR SCRL
STRL9   ZRO 0
STRL61  ZRO 0
*


*
*
* BRS 10
PPAN    LDA 0; CLB; LDX =QTI; BRM PTRAP
*
```

```
*
* BRS 70   COUNT FREE PMT ENTRIES
* RETURNS NO. OF FREE PMT ENTRIES IN A
FRPMT  LDX JOB; LDA PMTP,2; ADD =NUMEM+25OB5; STA FRP2
       LDX =-NUMEM; CLA; STA SS01
FRP2   SKE 0,2; BRU *+2; MIN SS01; BRX *-3; BRU POPX
*
*
* BRS 71   SKIP IF EXFLG SET. SET B=UEXFLG VALUE.
* UEXFLG VALUE:   1      0      -1   -2
* EXEC TYPE:      1   NEITHER   BOTH   2
SKXEC  LDB UEXFLG, STB SS02; SKN UEXFLG; BRU POPX; MIN 0; BRU POPX
*
*.
*     TRAP ROUTINES
*
* ILLEGAL INSTRUCTION TRAP
TRAPI  ZRO; STA SS01; STB SS02; STX SS03
       LDX PACPTR; LDA RL2,2; ETR =77B2
       SKE =16B2; BRU *+4; LDA RRL2; STA RRL1; BRM MONCR
       LDA TRAPI; LDB ONE; LDX =QTI; BRM PTRAP
*
*GENERAL TRAP LOGIC
* INPUT: A=TRAP LOC., B=REASON, X=QUEUE FOR NEW FORK.
       TLABEL PTP
PTRAP  ZRO; STX PTRPA
       CAX; LDA =600B; MRG RLTS; BRM PTPL
       LDA RRL1; STA PTRL1; LDA RRL2; STA PTRL2; LDA RRL3; STA PTRL3; CXA
       LDX PACPTR; SKG KM1; BRU *+2; LDA 0; ETR =50037777B
       DIR; LOGG 56B,PTRAP; EIR
       STA PL,2; SKN XPB; BRM MONCR
       LDX XPB; LDA SS02; STA PB,2; LDA SS01; STA PA,2
       LDA SS03; STA PX,2
       IF     DDIFP
       IF     FPH
       HFST   FAC1,FAC2
       ENDF
       ENDF
       LDX    PACPTR
```

```
        BRM RFK; LDA PTRPA; BRM TFK; STA FK04
        LDX TF07; STX STFK2; LDA PTRL1; LDB PTRL2; LDX PTRL3
        BRM LABEL; BRM FKG03
PTRL1   ZRB 0
PTRL2   ZRB 0
PTRL3   ZRB 0
PTRPA   ZRB 0   QUEUE FOR NEW FORK
*ILLEGAL SYSPOP EXIT
TRAPS   ZRB; TRAP 42
$TRAP   ZRB 0; BRM MPPACT; SKN XPB; BRU *+3; LDA TRAP; STA ERCODE
        LDA SBRSRT; SKN 0; STA 0; LDA 0; LDB ONE; LDX =QTI; BRM PTRAP
*READ-ONLY TRAP
        TLABEL  TPR
TRAPR   ZRB; STA TR01; STB TR02; STX TR03
        LDA TRAPR; ETR =50037777B; STA TR00; MIN TRPXMA
        BRM CAE; NOP; SKR TRPXMA; ETR =34000B; MUL =6B4; STA TPR4
        LDA RLTS; MRG =06B2; BRM TPRL
        LDX PUPAC; LDA RL2,2; LDB RL1,2
        LDX TPR4; LCY 6,2; ETR K77; SKG =NCMEM-1; BRU TPR26
        ADD PMTJOB; ETR ADMSK
TPR25   CAX; LDA 0,2; SKA X1
        BRU TPR3 (REALLY RB); CXA
        DIR; BRM W; EIR
        LDX TPR4; LDB RRL1; LDA RRL2; LCY 6,2; ETR =77777737B
        RCY 6,2; LDX RRL3; XAB
        SKN PACJOB; BRU TPR22; STA CMRRL1; STB CMRRL2; STX CMRRL3
TPR22   BRM LABEL
        LDA TPRL1; BRM TPRL
        IF      DDIFP
        IF      FPM
        LDA*    TR00
        LDB     =157B5
        SKM     =1365
        BRU     *+2
        SKR     TX00
        BRU     *+1
        ENDF
        ENDF
        ROV; LDA TR00; LCY 1; LSH 1
```

```
        LDA TR01; LDB TR02; LDX TR03; BRU* TR00
TPR3    LDA TR01; STA SS01; LDA TR02; STA SS02; LDA TR03
        STA SS03; LDA TRAPR; STA 0; BRM MTRAP
TPR26   ADD =SMT; BRU TPR25
TPR4    ZRB 0
TR00    ZRB 0
TR01    ZRB 0
TR02    ZRB 0
TR03    ZRB 0
TRPXMA  ZRB XMA41
XMA41   XMA 41B
XMA43   XMA 43B
*MEMORY TRAP

        TLABEL TPM
TRAPM   ZRB;*   MEMORY TRAP INTERRUPT ROUTINE
        IF BUG
        STA TT01; LDA TRAPM; ETR =40037777B
        SKE =TTOM; BRU *+2, BRM MONCR; LDA TT01
        ENDF
        STA TX01; STB TX02; STX TX03
        LDA TRAPM; ETR =50037777B; STA TX00; BRM CAE
        MRG =40000B (MEMORY MUST COME FROM A HIGHER FORK)
        STA TT01; LDA =0600B; MRG RLTS; BRM TPML (MAP PMT); LDA TT01
        LDX PACPTR; BRM MGET; BRU TX05
        LDA TPML1; BRM TPML   (RESTORE RELABELLING)
        IF      DDIFP
        IF      FPH
        LDA*    TXC0
        LDB     =157B5
        SKM     =1365           CHECK FOR POT OR PIN FROM FP
        BRU     *+2
        SKP     TXC0
        BRU     *+1
        ENDF
        ENDF
        ROV; LDA TX00; LCY 1; LSH 1
        LDA TX01; LDB TX02; LDX TX03; BRU* TX00
TX05    SKM TRAPM; BRM MTRAP
```

```
        LDA TX01; STA SS01; LDA TX02; STA SS02; LDA TX03; STA SS03
        LDA TX00; SKA X4; STA 0; BRM MTRAP
MTRAP   0; LDX PACPTR; LDA K1B6; BRM RIIR
MTPAN   LDA 0; LDB TW0; LDX =QTI; BRM PTRAP
TX00    ZR0
TX01    ZR0
TX02    ZR0
TX03    ZR0
        IF      DUG
TT01    ZR0 0
        ENDF
* COMPUTE EFFECTIVE OUT-OF-BOUNDS ADDRESS
* INPUT: A=TRAP LOC.
* OUTPUT: A=ADDRESS OR X4 IF POP CAUSED TROUBLE
* NO SKIP: USER POP OR P=LOC. BRANCHED TO.
*       TIME = 114 + N CY
CAE     ZR0; STA CAE1; STX CAE3; SKN PUSW; BRU *+2; BRM M0NCR
CAE11   BRM CEX; LDA* CAE1; BRU CAE4
* ADDRESS FOUND
CAE5    LDA CAE1; BRR CAE
* ADDRESS NOT FOUND
CAE4    LDA CEX3; STA CAE2; SKA X1; BRU CAE6; LDA IABIT; ADM CAE1
        LDX CAE3; BRM CEX; EAX* CAE1; BRU CAE7
* INDIRECT ADDRESS CHAIN OUT-OF-BOUNDS
        LDA KM4B4; ADM CAE1; BRU CAE9
CAE8    LDA CAE1; ETR X4; MRG CEX3; ETR =60037777B; STA CAE1
CAE9    LDX CAE3; BRM CEX; LDA* CAE1; BRU CAE8; BRU CAE5
* POP CAUSED TROUBLE
CAE6    LDA X4; BRR CAE
* CHECK FOR EXU- OTHERWISE X CONTAINS BAD ADDRESS
CAE7    CXA; ETR =40037777B; STA CAE1; LDA CAE2; EOR EXUW
        SKA =17700000B; BRU CAE10; LDX CAE3; BRU CAE11
* NOT EXU
CAE10   MIN CAE; BRU CAE5

CAE1    ZR0
CAE2    ZR0
CAE3    ZR0
EXUW    EXU     0
```

```
* EXECUTE NEXT INSTRUCTION AND SKIP IF OUT-OF-BOUNDS
*       TIME = 31 + N CY
CEX     ZR0; MIN CEX; STA CEX1; LDA CEX2; EXU* TRPXMA; STA CEX3; EXU* CEX
CEX4    XMA CEX3; EXU* TRPXMA;  LDA CEX1; BRR CEX
* OUT-OF BOUNDS IF WE COME HERE
CEX2    BRU *+1; MIN CEX; BRU CEX4

CEX1    ZR0 0    SAVE A
CEX3    ZR0 0    CONTENTS OF A AS A RESULT OF EXECUTING NEXT INSTR.


* PARITY INTERRUPT ROUTINES
CPUP    0;* CPU PARITY. CPUP MAY CONTAIN ADRS OF FAILING INSTRUCTION.
        MIN CPC; SKN CPUP; BRU *+3; SKN EXECS; BRI CPUP
        STX MCRX; LDX =100076B
CPUIOP  CKF; EOM 20004B (DIR); STA MCRA; STB MCRB; BRM STOP

IOP     0;* IO PARITY INTERRUPT
        EOD 12000B; PIN T3; STX MCRX; LDX =100074B; BRU CPUIOP

RESBRS  BRM MPWB; BRU WRSB
INCBRS  BRM MPXB; BRU WINB
REDBRS  BRM MPWB; BRU WROB
*
* BRS 143  TEST BIT MAP SET
BMTST   SKN SDBM8; MIN 0 (MAP SET); BRU POPX
*       BRS 7      READ OR WRITE A TABLE IN THE MONITOR MEMORY
*       A=USER CORE ADRS, SIGN BIT ON FOR WRITE. X = TABLE NUMBER.
*       RWTAB MACRO SETS UP A SINGLE WORD ENTRY IN THE TABLE OF
*            ACCESSABLE MONITOR TABLES.
*       AN ENTRY HAS BITS 10-23 AS THE TABLE ADRS. BITS 4-9 ARE THE TABLE
*            SIZE. BITS 0,1 ARE THE MINIMUM STATUS REQUIRED TO WRITE
*            THE TABLE AND BITS 2,3 ARE THE READ STATUS.

RWTAB   MACRO D;* CALLED WITH 4 ARGS      TABADR,TABSIZ,WSTAT,RSTAT
D2M1    EQU D(2)-1   SIZE ARGUEMENT MINUS 1
```

```
DTAG    EQU 2*D(3)+D(4)/2  BITS 0,1 FOR WR STATUS, BITS 2,3 FOR READ STAT
D0P3    EQU D(4)-D(4)/2-D(4)/2 1ST BIT OF OP FIELD IS 2ND BIT OF RD STAT
D0P2    EQU D2M1/2+D0P3*400  CREATE OP FIELD
D0P1    OPD D0P2*1B5
        IF D2M1-D2M1/2-D2M1/2
        D0P1* D(1),DTAG; ELSE; D0P1 D(1),DTAG; ENDF
        FRST RWTAB,D2M1,DTAG,D0P1,D0P2,D0P
        ENDM

RWTAB1  EQU *    TABLE OF MONITOR TABLES ACCESSABLE TO THE USER
        RWTAB   MISC,47B,3,2     COUNTERS TYPED OUT BY *1COUNT, TABLE 0
        RWTAB   RAERL,20,3,2     RAD ERROR TABLE, TABLE 1
        RWTAB   IDER2,10,3,2     DISC ERROR LIST, TABLE 2
        RWTAB   QTIG0,32,3,2     SWAPS/ACTIV. COND, TABLE 3
        RWTAB   210B,6,3,2       BASE-HOST COMMUNICATIONS, TABLE 4
        RWTAB   RMT,NMEM,3,2     REAL MEMORY TABLE, TABLE 5
        RWTAB   GFLG,40,3,2      GLOBAL FILE NUMBERS, TABLE 6
        RWTAB   TTN0,42,3,2      JOB TO PORT CONVERSION, TABLE 7
        RWTAB   ALARM,1,3,2      ALARM, TABLE 8
        RWTAB   XBP,40,3,2       GLOBAL X BLOCK PTRS, TABLE 9
        RWTAB   DSCT0P,1,3,2     HIGHEST DISC ADDRESS, TABLE 10
        RWTAB   MACH,1,3,0       MACHINE NUMBER, TABLE 11
        RWTAB   EQPT,1,3,0       FOR THE MAP, TABLE 12
        RWTAB   STACT,2,3,0      TABLE 13
        RWTAB   WERIS,NPORT,3,2  LUD LOC TABLE, TABLE 14
        RWTAB   PMTP,NJOB,3,2    PMT PTRS FOR EACH JOB, TABLE 15
        RWTAB   CPARW,NJ0B,3,2   CONTROL PARAMETERS, TABLE 16
        RWTAB   AUNN,NJ0B,3,2    USER NUMBERS, TABLE 17
        RWTAB   FGLIST,NJ0B,3,2  F.D. GROUP LIST, TABLE 18
        RWTAB   ADRSMT,19,3,2    MISC MONITOR CELLS, TABLE 19
        RWTAB   RAFR,32,3,2      RAD BIT MAP, TABLE 20
        RWTAB   ETTB,NJ0B,3,2    COMPUTE TIME, TABLE 21
        RWTAB   PTCKS,NJ0B,3,2   PAGE TICKS, TABLE 22
        RWTAB   CCT,NJ0B,3,2     CHARACTER COUNT, TABLE 23
        RWTAB   TTYASG,NPORT,3,2  PACPTRS/PORT TABLE 24
        RWTAB   QO,20,3,2        MONITOR QUEUES, TABLE 25
        RWTAB   MONTAB,2,3,2     ADDRESSES TO PEEK AT
        RWTAB   UEXFLG,1,3,0     JOB STATUS
        RWTAB DBITS,1,3,0        NUMBER OF AVAIL DISC BLOCKS
```

```
RWTABN EQU *-RWTAB1      LENGTH OF TABLE
*
* MONTAB CONTAINS THE ADDRESS OF CELLS TO USE THE BRS 127 ON
MONTAB ZRO PNEXT; ZRO PPTR
*

BRS7    CXA; SKG =RWTABN-1; SKG KM1; TRAP 43    (ILLEGAL TABLE);
        LDA RWTAB1,2; SKN SS01; LCY 2; LRSH 1; SKG PRILEV
        BRU *+2; TRAP 44   (READ OR WRITE STATUS INSUFFICIENT)
        LDA RWTAB1,2; STA LP; RSH 14; ETR K77; ADD ONE  (TABLE SIZE)
        COPY AB,N; COPY AX,N   SIZE IN A AND B. NEGATIVE SIZE IN X.
        ADD SS01; ETR =40037777B; STA SS01 UPDATE USER'S POINTER; MRG X4
        XAB; ADD LP; ETR ADMSK    USER ADRS IN B , MONITOR ADRS IN A
        SKN SS01; XAB (HE WANTS TO READ, NOT WRITE);
        ADD =23503 (STA 0,2); STA LP+1; CBA; ADD =27605 (LDA 0,2); STA LP
LP      0; 0; BRX LP; BRU POPX


* BRS 39  READ CPARW AND AUNN
RCPARW LDX J08; LDA CPARW,2; LDB AUNN,2; LDX SS03; BRU XPOP
*
* PEBRS     BRS 127
* READS A WORD IN MEMORY.
* INPUT: X=LOC. OF WORD
* OUTPUT: A= CONTENTS OF WORD.
PEBRS   EQU *
        LDA SS03; SKG K1S5; SKG KM1; TRAP 45;
        RSH 11; ETR K37; MRG =600B; XMA RRL3; STA PEB2
        LRR3; POT RRL3; CLA; LSH 11
        LDB K77; SKB RRL3; ADD =34000B
        CAX; LDA 0,2
        STA SS01; LDA PEB2; XMA RRL3; LRR3; POT RRL3; BRU POPX
        SKE =612B; BRU POPX; BRU BPOPX (THWART PASSWORD WATCHING)
PEB2    ZRO 0    TEMP STORAGE FOR RRL3
*
*
* SETSW    BRS 136
* SET SWITCHES FOR MONITOR AND EXEC
* INPUT: A=NEW SWITCH VALUE.  X=SWITCH NUMBER.
```

ETBO+1483

SKG = [illegible]; SKG = TBI-1 +1483; BRU X+2; BRU POPX

X

```
* OUTPUT: A=OLD SWITCH VALUE.
SETSW  XMA SWEX,2; BRU POPXB
*
* BRS 5   RETURN STATUS OF CALLING FORK
* RETURN: A=1=SUBSYSTEM, A=3=SYSTEM, A=7=EXEC
*         A=0=NO STATUS
FKSTAT LDA PPTR,2; MRG PLMSK; SKA PRMSK; BRU *+2; LDA PACPTR
       COPY AX,A; LDB X2; SKB PTAB,2; MRG ONE; LDB K400
       SKB PIM,2; MRG FOUR; LDB K200; SKB PIM,2
       MRG TWO; BRU POPXB
*
*
*
*
*
*       FORK LOGIC
*
* FIND HIGHEST FORK IN STRUCTURE
HFK    ZRO; LDA XPB; ETR X4; STA XPB; STA STFK2
HFK1   LDA PPTR,2; SKA PRMSK; BRU *+2; BRR HFK
       MRG PLMSK; CAX; BRU HFK1
* GET FORK ENTRY.
GFK    ZRO; LDA FPLST; SKG ZERO; BRR GFK
       SUB =PPTR; COPY AX,A
       XMA PPTR,2; STA FPLST; MIN GFK; BRR GFK
* SET PDOWN(OLD)=NEW, PDOWN(NEW)=0
*      PFORK(NEW)=OLD, PPAR(NEW)=PDOWN(OLD)
* SET PTEST TO 7E 2.
STFK   ZRO; STX FK04; SKR NFORK; BRU *+2; BRM MONCR
       LDX PPB; LDB PB,2; STB PPB; CXB; STB STFK2
       LSH 3; LDX FK04; STB PIM,2; CXA
       LDX PACPTR; LSH 12
       ETR PLMSK; XMA PPTR,2; ETR PRMSK; ADM PPTR,2
       LDA FK04; COPY XA,AX,B; ETR PRMSK; STA PPTR,2
       LDA STFK3; STA PTEST,2
       LDX PACPTR; BRR STFK
STFK2  ZRO 0    XPB FOR NEW FORK
STFK3  7B 2   ACT. COND. BEFORE FORK RUNS.
```

```
*
* DELETE PAC ENTRY WH0SE PACT PTR IS GIVEN IN X
DFK     ZR0; DIR; L0GG 21B,DFK; L0GG 21B,PACPTR; EIR
        CxA;  STX DF01; LDA =700000B; BRM GSCH
* REM0VED FR0M QUEUE IF DISMISSED
        LDA PPTR,2; SKA PRMSK; BRU *+2; BRR DFK (ExEC N0T DELETED)
        MIN NF0RK
* REM0VE Pb P0INTER
        LDA PIM,2; RSH 3; ETR SEVEN; LDX PPB; XXA
        STA PB,2; STX PPB; LDX DF01
DF08    LDA PPTR,2; MRG PLMSK
        C0PY AB,XA; LDA UTTY
* PF0RK IN B, PACT PTR IN A
        SkE TTYASG,2; BRU *+2; STB TTYASG,2 (PR0PAGATE RUB0UT 'UP')
* PUT PF0RK IN X, PD0WN(PF0RK) IN A
        C0X; LDA PPTR,2; RSH 12; SKE DF01; BRM M0NCR
        C0PY XB,AX
* LAST F0RK 0N PD0WN T0 BE CLEARED- PUT PF0RK IN X, PACT PTR IN B
        XXB; LDA PPTR,2; ETR PRMSK; STA PPTR,2; C0PY BX,BA
* PUT PACT ENTRY 0N FPLST
        ADD =PPTR; xMA FPLST; STA PPTR,2; BRR DFK
DF01    ZR0
*

* SEARCH SUBSIDIARY F0RK STRUCTURE F0R SPECIFIED F0RK
*       SkIP IF PRESENT, N0 SkIP IF TERMINATED
SHFK    ZR0; LDA PPTR,2; SKA PLMSK; BRU *+2; BRR SHFK; RSH 12
        CAX; LDA PTAB,2; E0R SS01; SKA ADMSK; BRR SHFK
        MIN SHFK; BRR SHFK
* SCAN F0RK STRUCTURE AND 0PERATE
*       A= 0PERATION
*       X= PACPTR
*       THE 0PERATION SPEC. BY A IS CALLED WITH C0RRECT PACPTR IN X
*       PERF0RMS 0P F0R ALL L0WER AND PARALLEL F0RKS.
SCFK    ZR0; STA SCF01; STB SCF02; STX SCF03; CXA
* SCAN T0 L0CAL '30TT0M'
SCF06    SkN PPTR,2; BRU SCF07
        LDA PPTR,2; RSH 12; CAX; BRU SCF06
```

```
SCF07  CXA; SKE SCF03; BRU SCF05
* PERFORM OP AND EXIT
       BRM* SCF01; LDB SCF02; LDX SCF03; BRR SCFK
* PERFORM OP AND GO 'UP'
SCF05 LDA PPTR,2; MRG PLMSK; STA SCF04; BRM* SCF01; LDX SCF04; BRU SCF07
SCF01   ZRO 0   OPERATION
SCF02   ZRO 0   STATUS WORD
SCF03   ZRO 0   PACPTR
SCF04   ZRO
*
* TERMINATE FORK STRUCTURE
* INPUT: X=PACPTR OF LOWER FORK, A=QUEUE, B=STATUS WORD
* OUTPUT: A=TF06=PACPTR FOR NEW FORK, X=TF05=QUEUE FOR NEW FORK
TFK     ZRO
        STA TF05; STB TF12; STX TF01
        IF ST
        DIR; SKN STSW; BRU TFKA
        LDA =RSR; BRM RLABEL; BRM SSS
        2 =255+11
        3 JOB
        4 CTTY
        5 PAGEW
        2 PPTEST
        2 PTESTS
        10B TF01
        7 UNOSV
        1 REAL
        LDX TF01
TFKA    EIR
        ENDF
        LDB TF12; LDA PPTR,2; STA TF01; LDA PTAB,2; STA TF06
        LDA =DFK; BRM SCFK; LDA TF01; STX TF01; SKA PRMSK; BRU TF02
* EXEC TOP-LEVEL PANIC
        LDA TF10; STA PTEST,2; SKN XPB; BRM MONCR
        LDX XPB; LDA UEXL6; STX TF07
        STA PX,2 (SAVE STATUS); LDX TF01
        LDA TFC01; XMA PL,2; STA TFC02; MIN TFC02
        BRU TF09
* ORDINARY PANIC
```

```
TF02    MRG PLMSK; COPY AX,A8; LDA PIM,2; RSH 3; ETR SEVEN; STA TF07
        CXB
* DISMISSED ON ACT. COND. 7
        LDA TF13; STA PTEST,2; MIN PL,2
        SKN XPB; BRM MONCR; LDX TF07
        LDA TF06; ETR ADMSK; STA PA,2; LDA SCF02
        STA PX,2; CBX
TF09    CXA;  STX TF06; LDX TF05; BRR TFK
$TFC1   ZR8
TF12    ZR8
TF10    20B PACDMB    EXEC PANIC
TF13    21B PACDMB    ORDINARY PANIC
TF05    ZR8 0    QUEUE FOR NEW FORK
$TF06   ZR8 0    PANIC TABLE ADDRESS
TF07    ZR8 0    FORK NUMBER FOR HIGHER FORK
TFC01   ZR8 EXECP,4
TFC02   ZR8 0    PL BEFORE IT BECAME EXECP,4
BRS94   BRM TRAP     BRS TO ESCAPE EXEC

* READ FORK STATUS
* INPUT: B=STATUS, X=PACPTR OF FORK TO BE READ
RFK     ZR8; LDA PIM,2; EOR =300B; SKA =300B; BRU RF12 (NOT EXEC BRS)
        LDA PPTR,2; MRG PLMSK; CAX
RF12    LDA =RF06; BRM SCFK
        LDX J88; STX SWJ8B; LDX =NCMEM; CLAB; BRM SWAPI
        BRM LABEL
        LDB SCF02; LDX SCF03; BRR RFK
*
* INPUT: X=PACPTR OF LOWER OR PARALLEL FORK.
RF06    ZR8
        STX RF08; LDX J88; STX SWJ8B; LDX RL3,2; CLAB
        BRM SWAPI; BRM LABEL; LDX RF08
        LDA PL,2; SKA X4; BRU *+2; LDA SBRSRT; STA RF14
        CXA; EOR SCF03; SKA ADMSK; BRU RF10
        LDB SCF02; SKB X4; BRU *+2; BRU RF07
RF10    LDA PTEST,2; ETR ADMSK; LDB KM1; SKE PACDMB; LDB KM2
RF07    LDA PIM,2; EOR =300B; SKA =300B; BRU *+3; CXA; BRU RF11 (EXEC BRS)
        LDA PPTR,2; SKA PRMSK; BRU *+2; CXA (TOP-LEVEL EXEC)
RF11    MRG PLMSK;  STA RF13; STB RF17
```

```
        LDA PIM,2; RSH 3; ETR SEVEN; STA RF15
        LDA PTAB,2; ETR ADMSK; MRG X4; STA RF08
        CXA; ADD RFC01; STA RF16; LDA RF08
        ETR =34000B; LRSH 11; MUL THREE
        LDX RF13; LDA RL1,2; LDX RL2,2; XXB; RCY 18; LCY 0,2
        STX RF33
        ETR K77; SKE ZERO; BRU *+2; BRR RF06
        STA RF32
        LDB THREE; STB RF02; LDB Q26; STB RFR
        LDB JBB; STB SWJBB; CLB; RCY 0,2; LCY 18
        LDX =NCMEM; STA RFRL1; STB RFRL2; STX RFRL3
RF03    BRM SWAPR; BRU RF04 (PANIC TABLE PAGE AND TS)
        BRM LABEL
        LDA RF32; ADD PMTJBB; DIR; BRM W; EIR; LDX RF33
        LDB RRL1; LDA RRL2; LCY 6,2; ETR =77777737B
        RCY 6,2; LDX RRL3; XAB; BRM LABEL
        LDA RF14; STA* RF08; MIN RF08 (SAVE PL)
        LDX RF15; LDA PA,2; STA* RF08; MIN RF08
        LDA PB,2; STA* RF08; MIN RF08
        LDA PX,2; STA* RF08; MIN RF08; LDX KM2
RF16    LDA PL4,2; STA* RF08; MIN RF08; BRX *-3
        LDA RF17; STA* RF08
* RELEASE PANIC TABLE AND TS BLOCK
        LDX JBB; LDA PMTP,2; ADD =-NCMEM; STA RELPG3
        LDA X4; STA RELPG8; LDA RFRL1; LDB RFRL2; LDX RFRL3
        BRM RELMEM; BRR RF06
* CAN'T READ PANIC TABLE PAGE
RF04    SKR RF02; BRU *+2; BRR RF06
        LDA RFRL1; LDB RFRL2; LDX RFRL3; BRU RF03
RF02    ZRB 0   READ ERROR RETRY COUNT
RF08    ZRB
RF13    ZRB
RF14    ZRB
RF15    ZRB 0   PB POINTER OF FORK TO READ
RF17    ZRB 0   STATUS WORD
RF32    ZRB 0
RF33    ZRB 0
RFC01   LDA PL4,2
RFRL1   ZRB 0   PANIC TABLE AND TS BLOCK RELABELING
```

```
RFRL2   ZRO 0
RFRL3   ZRO 0
*
* BRS 9   START SUBSIDIARY FORK
* INPUT: SS01=A=BITS + PANIC TABLE ADDRESS
*        BITS: 0=EXEC, 1=PANIC TABLE REL., 2=PROP. RUBOUT
*              3=FIXED MEMORY, 4=LOCAL MEMORY, 5=SUBSYSTEM STATUS
*              6=SYSTEM STATUS, 7=SUBSYSTEM INDEX
FKST    SKN PIM,2; BRU *+2; TRAP 46
        ETR K3777; ADD SIX; SKA K4000; TRAP 47  (PANIC TABLE OVERLAP)
        DIR; LOOP 57B,XPB; EIR
        LDA SS01; BRM SHFK; BRU *+2; TRAP 48  (TRIED TO RESTART SAME FORK)
        LDA NFORK; SKG ZERO; TRAP 49
        BRM SFK; BRU FKSTW (DISMISS UNTIL PACT SLOT FREE); BRM STFK
        CLB; LDA SS01; SKA X4; BRU FK20 (EXEC STATUS)
FK24    SKA K4B5; BRU FK21
FK25    SKA K1B6; BRU FK22
FK23    STA SS01; STB FK09A
        LDX FK04; LDB SS01; LDA PIM,2
        SKB X4; MRG K400; SKB K4B5; MRG K200; SKB K4B6
        MRG X4; STA PIM,2
        CBA; ETR ADMSK; LRSH 15; LDA J0B; LSH 15
        SKB K2B6; MRG X4; SKB K1B6
        MRG X2; MRG X1; STA PTAB,2
* PROPAGATE TTYASG DOWN IF CALLED FOR AND FORK HAS TTYASG
        LDA UTTY; SKG KM1; BRU FK07A
        CAX; LDA PACPTR; SKE TTYASG,2; BRU FK07A
        LDA FK04; SKB X1; STA TTYASG,2
FK07A   LDX PACPTR; LDA SS01; SKA X2; BRU FK08
        LDA RL1,2; LDB RL2,2
        LDX FK09A; BRM SCRL; BRU FK08B; LDX PACPTR
        LDA RL1,2; LDB RL2,2
* STORE RELABELLING, INITIAL A,B,X,L
FK09    LDX FK04; STA RL1,2; STB RL2,2; LDX SS01
        LDA 2,6; LDB 3,6; LDX STFK2; STA PB,2; STB PX,2
        LDX SS01; LDA 1,6; LDX STFK2; STA PA,2
        LDX SS01; LDA KM1; STA 6,6 (SET STATUS WD TO RUNNING)
        LDA 0,6; ETR =50037777B; SKN 0; BRU *+2; MRG X4
        LDX FK04; STA PL,2
```

```
        LDB =700006B; BRM FKG8
* PICK UP RELABELLING FROM PANIC TABLE
FK08    CAX; LDA 4,6; LDB 5,6; LDX FK09A
        BRM SCRL; BRU FK08B; LDX SS01; LDA 4,6; LDB 5,6; BRU FK09
FK08B   LDX FK04; BRM DFK; TRAP 50
* DISMISS UNTIL PACT SLOT IS RELEASED
FKSTW   LDB =FPLST; LDX =QTI; BRU NPOPDS
FK20    SKN EXECS; BRU *+4; MRG =41465; LDB X7
        BRU FK23; ETR K357; BRU FK24
FK21    SKN SYSS; BRU *+4; MRG =1485; LDB X3
        BRU FK23; ETR =37377777B; BRU FK25
FK22    SKN EXEC1; BRU *+3; LDB X1; BRU FK23
        ETR =36377777B; BRU FK23
FK09A   ZRO 0   STORAGE FOR RELABELLING STATUS
*
* DO NOT DISMISS EXEC BRS AND BRS 9 FORKS.
FKG8    ZRO; LDX PACPTR; STB PTEST,2
        LDA 0; ETR =50037777B; XMA PL,2; ETR =77B5; ADM PL,2
        SKN XPB; BRM MONCR; LDX XPB; LDA SS02; STA PB,2
        LDA SS01; STA PA,2; LDA SS03; STA PX,2; BRM FKG83
FKG83   ZRO
        LDX STFK2; LDA PA,2; STA SS01; LDA PB,2; STA SS02
        LDA PX,2; STA SS03; BRM FKG82
FKG82   ZRO; LDA KM1; STA DNTERM
        LDX PACPTR; LDA RL1,2; LDB RL2,2
        LDX FK04; XMA RL1,2; STA STRL1; CBA; XMA RL2,2
        STA STRL2; STX PACPTR; STX PUPAC
        SKR PL,2; NOP; SKN PACJOB; BRU *+2; STX PACOMP
        LDA PL,2; STA 0; BRM STSTAT
        LDA PIM,2; LRSH 3; ETR SEVEN; MRG X4; STA XPB
        LDX JOB; DIR; LDA TTNO,2; ETR =77776077B; STA TTNO,2
        EIR; LDX PACPTR; LDA =700001B; STA PTEST,2
        LDA STRL1; LDB STRL2
        DIR; LOGG 20B,FKG8; LOGG 20B,PACPTR; EIR; BRU STRLG
*
* BRS 31
* WAIT FOR SPECIFIED FORK TO TERMINATE
* INPUT: A=PANIC TABLE ADDR.
* OUTPUT: X=STATUS
```

```
FKWT    BRM SHFK; BRU FK01; BRM M0NCR
FKWT1   STB PACDMS; BRU P0PST
FK01    LDX SS01; LDA 6,6; STA SS03; BRU P0PX
*
* PR0GRAMMED INTERRUPT L0GIC
*
* BRS 49    READ INTRRUPTS
SRIR    LDA PIM,2; ETR =37777000B; LDX SS03; BRU XP0P
*
*
* BRS 78    ARM INTERRPUTS
SAIR    ETR =37770008; SkN SYSS; ETR =3776000B
        XMA PIM,2; ETR =74000777B; ADM PIM,2; BRU P0PX
*
* BRS 135    INTERRUPT AFTER A SPECIFIED TIME.
$WETTB ZR0 0 M0VE T0 SYMS S0METIME
ENDBRS BSS 0
        F0RGET
        END
```

```
        LISTM
*QUE STRUCTURE
*       THERE ARE READ QUES, SPACE QUES, AND CLEAR QUES THAT HAVE 2 WORD
*       BLOCKS CHAINED LIKE PUCT.THE SECOND WORD CONTAINS THE INTERESTING
*       DATA. BITS 1-8 TELL WHICH CELLS SHOULD BE SKIP REDUCED WHEN A
*       BLOCK IS READ. THE ADRS PORTION OF THE DATA WORD CONTAINS A PMT
*       ADRS. RACHH STARTS THE READ,SPACE, AND CLEAR QUES. THEY ARE
*       INTERLEAVED. EACH QUE ENDS WITH A POINTER TO A -1.
*       THE WRITE QUE AND THE EXPENDABLE BLOCK QUE MAY HAVE ENTRIES
*       REMOVED OR ADDED EASILY. THE BLOCK NUMBER TO THE RIGHT OF
*       BLOCK X ON THE WRITE QUE IS AT RMR+XNUMBER TO THE RIGHT OF
*       BLOCK X ON THE WRITE QUE IS AT RMR+X.  THE LEFT END OF THE WRITE
*       QUE BEGINS AT RMR+40 WHICH SAYS 41 IF THE QUE IS EMPTY. THE
*       RIGHT END OF THE WRITE QUE STARTS AT RML +41. THE BEGINNING
*       OF THE EXPENDIBLES IS AT RML+40.


SG      EQU -1;* INSTALL EMPTY SWITCH FOR SPEED
BUG1    EQU 1
SAFE EQU 1
SN      EQU 0B5;* VOID, UC=0
SS      EQU 1B5;* WAITING FOR CORE BLOCK, UC IN PMT
SC      EQU 2B5;* IN CORE, UC IN RMC
SD      EQU 3B5;* ON RAD, UC=0
SB      EQU 4B5;* BOTH CORE AND RAD, UC IN RMC
SWQ     EQU 5B5;* ON WRITE QUEUE, UC=0
SW      EQU 6B5;* BEING WRITTEN, UC=0
SRQ     EQU 7B5;* ON READ QUEUE, UC IN PMT
SR      EQU 10B5;* BEING READ, UC IN RMC
SE      EQU 11B5;* CORE AND RAD BUT CORE EXPENDABLE, UC=0
SI      EQU 12B5;* INDIRECT
SCQ     EQU 13B5;* WAITING TO HAVE ITS CORE BLOCK CLEARED, UC IN RMC
SWCT    MACRO; LDX BJOB; MIN SWPCNT,2; MIN SSWC; ENDM
LOGG    MACRS; ENDM
$RAI1   ZRO; DIR; SKS* 100268; BRU *-1;* READY TEST
RASW    EOD 100268;* SOMETIMES BRU RASWA
        PUT RADDR;* RAD ADDRESS
```

```
        SKS* 11026B; MIN RDER;* RAD ERR0R
        SKS* 11000B; MIN RCER;* CHANNEL ERR0R
$RADIA E0D* 10000B
RADIF   E0D 15200B; P0T CADD;* C0RE ADDRESS
RADIG   E0D 2266B;* WRITE(S0METIMES 2226 F0R READ.)
        STA RAA; STB RAB; STX RAX
        LDA RADDR; LDB K40; SKB RADIG; MRG K4B7; XMA RADDRC; STA RADRCC
        LDA CADD; STA CADDC
        LDA =600B; XMA RRL3; LRR3; P0T RRL3; STA RARLX
        SKN RAB0L; BRU RALIT; MIN RAB0L; LDA KM1
        ADM RADIF; LDA =30003500B; ADM CADD
        LDA =35B; ADM RADDR
        LDA K40; SKA RADIG; BRU FIXCH
RASWBX  LDA RDER; ADM RC; LDA RCER; ADM RC; ADM RS; SUB 0NE
        ADD RDER; STA RAPSTS
        SKE KM1; BRU *+2; BRU RASWB; LDA RADRCC; BRM RAREW
RASWB   MIN PAGES
        SKN RACUR; BRU *+2; BRU RAEXX;* THERE WAS N0 PREVI0US
*   J0B 0R ELSE IT WAS ABAND0NED.
        LDX RACUR; LDX RMT,2; L0GG 5,L0GX
        LDX RACUR; LDA* RMT,2; LDB =7705; SKM =SR; BRU RAEW
*   THIS CLEANS UP A READ.
        LDA =SB-SR; ADM* RMT,2; LDB* RMT,2
        CXA; MRG =RSP*64; MLABEL
        LSH 19; ETR K777; CAX; LSH 7; E0R RAERT,2; ETR KM200
        E0R RAERT,2; LSH 7; LDB =77777600B; SKM* RACKWD; BRU RARE
        LDA RAERT,2; E0R* RACKWD; ETR KM200; E0R* RACKWD; STA* RACKWD
RAERX   LDA RARN; BRM RAP0ST; SKN RAPSTS; BRU *+2; BRU RAEXX
        LDA =20600B; STA RRL3; LRR3; P0T RRL3
        LDX RACUR; LDA =SD-SB; ADM* RMT,2; LDA =32; BRM RAINS
        LDA KM1; STA RMT,2; BRU RAEXX
FIXCH   SKN RACURX; BRU *+2; BRU RASWB; LDA RACURX; ADM RRL3; LRR3
        P0T RRL3; LDB RACHKW; STB* RACKWD; BRU RASWBX
RARE    MIN RAPSTS; MIN RA; BRM RAREW
        LDA* RACKWD; BRM RAREW; BRU RAERX
RAREW   ZR0; STA* RAERLP; MIN RAERLP; SKR RAERCT; BRR RAREW
        LDA =19; STA RAERCT; LDA =RAERL; STA RAERLP; BRR RAREW
$RAERL  BSS 20
RAERLP  DATA RAERL
```

```
RAERCT DATA 19
*THIS CLEANS UP A WRITE.
RAEW    EQU *
        IF SAFE; SKM =SW; BRM M0NCR; ENDF
        SKN RMC,2; BRM M0NCR
        LDA RACUR; STA *+2; BRM RARLQS; ZR0
RAEXX   LDA RACURX; STA RACUR; LDA RARNX; STA RARN
        CLA; STA RDER; STA RCER; LDA KM1; STA RAPSTS
RAEX    LDA RARLX; STA RRL3; LRR3; PUT RRL3
        LDA RAA; LDB RAB; LDX RAX; EIR; BRI RAI1
RASWA   STA RAA; STB RAB; STX RAX
        LDA =600B; XMA RRL3; LRR3; PUT RRL3; STA RARLX
        MIN RA80L; LDA RAXE0D; STA RASW; BRU RASWB
RAP0ST  ZR0; EAX 7; ETR =37785; CLB
RAL00P  SKE ZER0, CRU *+2; BRR RAP0ST; N0D 1,2; SKN RAPSTS; MIN RAER0R,2
        SKR RAJ0B,2; BRU RALL; E0M 22400B
        IF      DDIFP
        SKR     UMTF
        BRU     *+1
        ELSE
        LDB J85; STB UMTJ0B; CLB
        ENDF
RALL    SUB X2; BRU RAL00P
RALIT   SKR RA80L; BRM M0NCR
RALITX  BRM RASS; BRM RASS
        CLA; STA RAW0RK; LDA KM1; STA RACURX
        LDA RAN0P; STA RASW; BRU RAEX
RASS    ZR0; LDA RAPH; E0R TW0; STA RAPH
        CAX; ADD RARLPH; ETR THREE; CAB; LDA RML32
        SKE =33; BRU *+2; BRU RAN0CR; CLA
        SKE RARHP,2; BRU RAG0H; XXB; SKE RARHP,2; BRU RAG0H; XXB
        SKE RARLP,2; BRU RAG0L; XXB; SKE RARLP,2; BRU RAG0L; XXB
RAN0CR  LDA RML33; SKE =32; BRU *+2; BRR RASS; LDA 0NE; STA RAWSC; CLA
RANCX   RPT (J=0,7); SKE RAFR+J*4,2; BRM RAGS; ENDR
        XXB; SKR RAWSC; BRU RANCX; BRR RASS
RAG0H   SKR RARHP,2; BRU RAJ
RAG0L   SKR RARLP,2
RAJ     STX RAPH; LDX RML32; BRM RADEL; SKN RMT,2; BRU *+2; BRU RAAV
        IF SAFE; LDA* RMT,2; ETR =7785; SKE =SE; BRM M0NCR; ENDF
```

```
        LDA =SD-SE; ADM* RMT,2
RAAV    STX RACURX; CXB; LDX RAPH; LDX RACHH,2; LDA 1,2; STA RARNX
        XXB; ETR ADMSK; STA RMT,2; XXA; LDX 0,2; XXA; ETR K37
        STA RMC,2; XXB; LDA RARFQ; XMA 0,2; MIN RPAGE
        STX RARFQ; LDX RAPH; STA RACHH,2; COPY BX,B; LDX RMT,2
        IF SAFE; LDA 0,2; ETR =77B5; SKE =SRQ; BRM MONCR; ENDF
        LDA 0,2; ETR =37740B; STA RADDR; LDA RAREOD; STA RADIG
        LDA RACURX; ETR K30; LSH 2; ADD RAEODR; STA RADIF
        LDA 0,2; ETR =77B5; SKE =SRQ; BRM MONCR
        LDA RACURX; ADD =SR; EOR 0,2; ETR =7700037B
        EOR 0,2; STA 0,2; ETR SEVEN; LSH 11; ADD =64B6; STA CADD; BRU RAEXL
RAGS    ZRO; STX RAPH; CXA; LDX RAGS; ADD =1,2; SUB =RAFR; MIN PAGEW
        STA RATS; COPY Ax,5;* THESE FIVE BITS ARE CALLED 'EFGHJ'
        LDA RAFR,2; EAx 15; NOD 16; STx RATSA;* THESE BITS 'ABCD'
        LDA RATS; RSH 5; CXA; SKB K2B6; ADD K20; LSH 9
        ETR =37740B; STA RADDR; LDx RATSA; LDA =77777600B; ADM RASCT
*   RADDR NOW CONTAINS 'JABCDEFGH00000'
        LSH 0,2; LDX RATS; ADM RAFR,2;* TURN OFF FREE BIT
        LDA RML33; ETR K30; LSH 2; ADD RAEOD; STA RADIF
        LDA RML33; ETR SEVEN; LSH 11; ADD =64B6; STA CADD
        LDX RML33; SKN RMC,2; BRM MONCR
        BRM RADEL; LDA RAWEOD; STA RADIG; STX RACURX
        LDX RMT,2; LDA RADDR; EOR 0,2; ETR =37740B; EOR 0,2
        STA 0,2;* RECORD RAD ADDRESS IN PMT.
        IF SAFE; ETR =77B5; SKE =SWQ; BRM MONCR; ENDF
        LDA =SW-SWQ; ADM 0,2
        LDA RADDR; RSH 5; CAx
        LDA RACURX; MRG =RSP*64; MLABEL
        MIN RAERT,2; LDA* RACKWD; EOR RAERT,2
        ETR KM200; EOR RAERT,2; STA RAERT,2; ETR K177
        RSH 2; MRG RADDR; LSH 9; EOR* RACKWD; ETR KM200
        EOR* RACKWD; XMA* RACKWD; STA RACHKW
RAEXL   LDX RACURX; LDX RMT,2; LOGG 10,LOGX; BRU RAEX

*   RARELB ADDS OR DELETES BITS FROM THE RAD SPACE MAP.
RARELB ZRO; ETR =77740B; LCY 15; SKB K20; ADD K2B6; XAB; ETR K17
        COPY AX,A; RSH 19; STB RADTS; MIN RARELB; LDA* RARELB
        ADM RASCT; LDB K200; LSH 0,2; LDX RADTS; SKB RAFR,2
        BRU *+4; SKG ZERO; BRM MONCR; BRU *+4; SKG ZERO; BRU *+2; BRM MONCR
```

```
        ADM RAFR,2; BRR RARELB
RADTS   ZR0

* RADEL DELETES THE BL0CK INDICATED IN X.
RADEL   ZR0
        STX RADTS; IF SAFE; CXA; LDX RML,2; SKE RMR,2; BRM M0NCR
        CAX; LDX RMR,2; SKE RML,2; BRM M0NCR; CAX; ENDF; LDA RMR,2
        LDX RML,2; STA RMR,2; XXA; STA RML,2; LDX RADTS; BRR RADEL

* RAINS INSERTS BL0CK X T0 THE LEFT 0F BL0CK A.
$RAINS  ZR0
        IF SAFE; C0PY XB,AX; LDA RML,2; XXA; SKE RMR,2; BRM M0NCR
        C0PY BA,AX; ELSE; C0PY XA,XB,AX; ENDF; XMA RML,2
        XXA; STB RMR,2; XXB; STB RML,2; STA RMR,2; BRR RAINS

RAF0N   ZR5; STB RAEXIT
RAF0Nx  CAX; STX RAPMT; LDA 0,2; RSH 15; ETR K77; L0GG 7,L0GA
        IF SAFE; SKG =SI/165; BRU *+2; BRM M0NCR; ENDF
        ADD RAF0N; XXA; BRU 1,2
*
* INPUT: A=PMT ADDRESS+BITS 1-8: FLAG BITS F0R RAJ0B BACKWARDS.
*          BITS: RAJ0B(X) GETS SKR WHEN J0B IS D0NE. IF RAJ0B(X)
*                G0ES LT 0, ARM UMT, CLEAR TIME.
*          SET SIGN=1 F0R HIGH PRIORITY READ 0R WRITE.
$REL     ZR0; L0GG 0,L0GA; LDB REL; BRM RAF0N
        BRM M0NCR; BRU RARLS; BRU RARLC
        BRM M0NCR; BRU RARLB; BRM M0NCR; BRM M0NCR; BRU RARLRQ
        BRU RARLR; BRM M0NCR; BRU RAIND
$RES    ZR0; MIN RES; L0GG 1,L0GA; LDB RES; BRM RAF0N
        BRR RAEXIT; BRU RARLS; BRU RARSC
        BRU RARSD; BRU RARSB; BRU RARSWQ; BRU RARSW
        BRU RARSRQ; BRU RARLR; BRU RARSE; BRU RARSI
$B      ZR0; L0GG 2,L0GA; LDB B; BRM RAF0N
        BRU RABN; BRU RABS
        BRU RABC; BRU RABD; BRU RABC; BRU RABWQ; BRU RABW
        BRU RABRQ; BRU RABR; BRU RABE; BRU RAIND
$W      ZR0; L0GG 3,L0GA; LDB W; BRM RAF0N
        BRR RAEXIT; BRM M0NCR
        BRR RAEXIT; BRM M0NCR; BRU RAW0; BRU RAWWQ; BRM M0NCR
```

```
        BRM MONCR; BRM MONCR; BRM MONCR; BRU RAIND

RAIND   CBX; LCY 9; CXA; RCY 9; BRU RAFONX

*  CALL HERE TO RELINQUISH CORE BLOCK TO SPACE QUEUE MEMBER OR
*  INSERT ON EXPENDABLE QUEUE.  X HAS CORE BLOCK.
*  BLOCK MAY OR MAY NOT HAVE PAGE IN IT.
RARLQS  ZRO; MIN RARLQS; CXB; LDX RASQH; SKN 0,2; BRU RACL; LDA =32
        CBX; SKN RMT,2; BRU *+2; BRU RARLQT

        CAB; LDA* RMT,2; ETR =70077777B; MRG =SE; STA* RMT,2; CBA
        SKN* RARLQS; LDA RMR33
        IF BUG1
RARLQT  BRM RAINS; LDB =7705; LDA =1285
        SKN RMT,2; SKM* RMT,2; BRR RARLQS; BRM MONCR
        ELSE
RARLQT  BRM RAINS; BRR RARLQT
        ENDF
RACL    LDA 0,2; STA RASQH; LDA RACLQH; STA 0,2; STX RACLQH

        LDX 1,2; CBA; ETR K37; ADD =SCQ; EOR 0,2; ETR KS7
        EOR 0,2; XMA 0,2; ETR K37; XXB
        STA RMC,2; BRM RARM; SKN RMT,2, BRU RAVOID
RAMARK  CBA; ETR ADMSK; STA RMT,2; BRR RARLQS
RAVOID  LDA* RMT,2; ETR =70077777B; MRG =SD; STA* RMT,2; BRU RAMARK
RARM    ZRO; ARMI ARM205; BRR RARM
$RCL    0;* CLEAR A PAGE. THIS IS A 205 INTERRUPT.
        ARMI DIS205
        STA CLLA; STB CLLB; STX CLLX

CLLP    DIR; LDX RACLQH; SKN 0,2; BRU CLGO; EIR
        LDA CLLA; LDB CLLB; LDX CLLX; BRI RCL
CLGO    CXA; XMA RARFQ; XMA 0,2; STA RACLQH; LDX 1,2; STX CLTS

        LOGG 4,LOGX
        CXA; BRM RAPOST; LDX CLTS; EIR; LDA 0,2; LDB =7705
        SKM =SCQ; BRM MONCR; ADD =SC-SCQ; STA 0,2; ETR K37

        XMA RRL3; LRR3; POT RRL3; STA CLTSA; LDX =-1000B; CLA
        STA 35000B,2; STA 36000B,2; STA 37000B,2; STA 0,2; BRX *-4
        LDA CLTSA; STA RRL3; LRR3; POT RRL3; BRU CLLP
*  PMT  USE TEST
RAPUT   ZRO; CAX; LDA 0,2; SKA K37; BRU *+2; BRR RAPUT
        SUB ONE; STA 0,2; BRR RAEXIT
```

```
*  GET A SLOT IN THE QUEUE AREA AND PUT RAPMT IN IT.
RAGGS   ZR0; LDX RARFQ; SKN 0,2; BRU *+2; BRM M0NCR
        LDA 0,2; STA RARFQ; LDA RAPMT; STA 1,2; BRR RAGQS
RABC    BRM RAZAP; MIN RMC,2; LDA RAPMT; BRM RAP0ST; BRR RAEXIT
*  RAQS FINDS WHICH READ QUEUE FR0M A RAD ADDRESS
RAQS    ZR0; LDX RAPMT; LDB 0,2; CLA; SKB K40; ADD TW0
        SKB <2B4; ADD 0NE; BRR RAQS

RARLS   BRM RAPUT; LDX =RASQH; BRM RAQSH; BRM RARET; DATA SN=SS
RARLC   BRM RAZAP; SKR RMC,2; BRR RAEXIT
        LDA =33; SKN RAPMT; LDA RMR32
        IF BUG1
        BRM RAINS; IF SG; MIN RANE; ENDF; LDB =77B5; LDA =07B5
        SKE* RMT,2; BRU *+2; BRM M0NCR; BRM RARET; DATA SWQ=SC
        ELSE
        BRM RAINS; IF SG; MIN RANE; ENDF; BRM RARET; DATA SWQ=SC
        ENDF
RARLB   STA RARLBX+1; BRM RAZAP; SKR RMC,2; BRR RAEXIT
RARLBX  BRM RARLQS; ZR0; BRR RAEXIT
        IF -1
RARLRQ  BRM RAPUT; BRM RAQS; STA RATSA; ADD =RACHH; CAX; BRM RAQSH
        LDA =RARHP; SKN 1,2; LDA =RARLP; ADD RATSA; CAX
        SKR 0,2; BRU *+2; BRM M0NCR; BRM RARET; DATA SD=SRQ
RARLR   LDA RAPMT; LDB RAEXIT; SKR HRLRC; BRM M0NCR
        EIR; HLT; DIR; MIN HRLRC; BRU RAF0NX
HRLRC   DATA 40000001B
        ELSE;RARLR EQU *;RARLRQ BRM M0NCR; ENDF
RARSC   BRM RARSTS; LDX RAPMT; LDA 0,2; ETR X7; XMA 0,2
        ETR <37; CAX; LDA KM1; STA RMT,2; BRM RARLQS; DATA -1; BRR RAEXIT
RARSD   CAX; LDA 0,2; BRM RARELB; DATA 200B; BRM RARET; DATA SN=SD
RARSB   BRM RARSTS; LDX RAPMT; LDA 0,2; BRM RARELB; DATA 200B; BRU RARSC
RARSTS  ZR0; BRM RAZAP; LDA =-64; MIN RMC,2
        SKA RMC,2; BRU* RAEXIT; BRR RARSTS
RARSRQ  BRM RAQS; ADD =RACHH; CAX; BRM RAQSH; LDX RAPMT
        LDA 0,2; BRM RARELB; DATA 200B; BRM RARET; DATA SN=SRQ

RARSWQ  BRM RAZAP; BRM RADEL; BRU RARSC
RARSW   BRM RAAW; BRU RARSC
RARSE   BRM RAZAP; BRM RADEL; BRU RARS0
```

```
RARSI   CAX; LDA 0,2; ETR X7; STA 0,2; BRR RAEXIT
RABS    EAX RASQH; BRU PIGGY
RABN    BRM RAQQS; LDA =32; SKE RMR33; BRU RAGT; CXA; XMA RASQH
        STA 0,2; LDX RAPMT; LDA 0,2; ETR X7; ADD =SS; STA 0,2
        BRR RAEXIT
RAGT    CXA; XMA RACLQH; STA 0,2; BRM RARM; LDX RML32; BRM RADEL
        CXB; SKN RMT,2; BRU RAGTX
RAGTY   LDX RAPMT
        LDA 0,2; ETR X7; ADD =SCQ; COPY BA,E; STA 0,2
        COPY B,BX,XA; STB RMC,2; ETR ADMSK; STA RMT,2; BRR RAEXIT
RAGTX   LDA* RMT,2; ETR =7785; SKE =SE; BRM MONCR
        LDA =SD-SE; ADM* RMT,2; BRU RAGTY
RABD    MIN* RFR; CAX; LDA 0,2; ETR =777777408; STA 0,2
        IF SG; MIN RANE; ENDF
        BRM RAQQS; STX RATS; BRM RAQS; CAX; LDA RATS
        SKN RAPMT; BRU RABD1; MIN RARHP,2; XMA RACHH,2
        LDX RATS; MIN RPAGEH; BRU RABD2
RABD1   MIN RARLP,2; EAX RACHH,2; STX RATS; LDX 0,2
        SKN 0,2; BRU *-3; STA* RATS; XXA; MIN RPAGEL
RABD2   STA 0,2
        IF ST; SKN STSW; BRU RAAL; LDA =RSR
        BRM RLABEL; BRM SSS; 2 =685+3; 2 RAPMT; 1 RFR
RAAL    EQU *; ENDF; SWCT; BRM RARET; DATA SRQ-SD
RAZAP   ZRB; LDX RAPMT; LDA 0,2; ETR K37; CAX; BRR RAZAP
RARET   ZRB; MIN RARET; LDX RAPMT; LDA* RARET; ADM 0,2; BRR RAEXIT
* DELETE ENTRY FROM QUEUE AND RETURN WITH X POINTING TO IT.
RAQSH   ZRB; LDA RAPMT; LDB ADMSK
RAQSH1  STX RATS; LDX 0,2
        SKN 0,2; BRU *+2; BRM MONCR; SKM 1,2; BRU RAQSH1
        LDB 0,2; STB* RATS; LDB RARFQ; STB 0,2; STX RARFQ; BRR RAQSH
RABWQ   BRM RAZAP; MIN RMC,2; BRM RADEL
        LDA RAPMT; BRM RAPBST; SWCT; BRM RARET; DATA SC-SWQ
RABW    BRM RAAW; LDA RAPMT; BRM RAPBST; BRM RAZAP
        LDA =SC-SW; ADM* RMT,2; MIN RMC,2; SWCT; BRR RAEXIT
RABRQ   BRM RAQS; ADD =RACHH; CAX; BRU PIGGY
RABR    LDB ADMSK; CLX; SKM RARN; EAX 1; ETR =37785; CAB
        ETR RARN,2; XAB; MRG RARN,2; STA RARN,2; LDX RACUR,2
        MIN RMC,2; CBA; BRM RAPBST; BRR RAEXIT
RABE    BRM RAZAP; MIN RMC,2; BRM RADEL; LDA RAPMT; BRM RAPBST; SWCT
```

```
         BRM RARET; DATA SB-SE
PIGGY    LDB ADMSK; LDA RAPMT
BUMP     LDX 0,2; SKN 0,2; BRU *+2; BRM MONCR; SKM 1,2; BRU BUMP
         ETR =37765; CAB; ETR 1,2; XAB; MRG 1,2; STA 1,2
         CAX; MIN 0,2; CBA; SKA KM1; BRM RAPOST; BRR RAEXIT

RAWB     CAX; LDA 0,2; BRM RARELB; DATA 200B; BRM RARET; DATA SC-SB
RAWWG    BRM RAZAP; BRM RADEL; BRM RARET; DATA SC-SWQ
RAAW     ZRO; CAX; LDA 0,2; ETR K37; SKN RABOL; BRU RAAWB
         SKE RACUR; BRU RAAWA; LDB KM2; STB RACUR; BRU RAAWX
RAAWA    SKE RACURX; BRM MONCR; LDB KM3; STB RACURX
         XMA RRL3; LRR3; POT RRL3; LDB RACHKW; STB* RACKWD
         STA RRL3; LRR3; POT RRL3; BRU RAAWX
RAAWB    SKE RACUR; BRM MONCR; LDB KM4; STB RACUR; STB RACURX
RAAWX    LDA 0,2; BRM RARELB; DATA 200B; BRR RAAW
$RADI2   ZRO; BRM MONCR
$DRAGO   EQU *
$RAGO    ZRO; IF SG; SKN RANE; BRU RAGO1; BRR RAGO
RAGO1    LDA KM1; STA RANE; ENDF
         LOGG 6,REAL; SKN RAWORK; BRU RAO1; BRR RAGO
RAO1     LDA KM1; STA RAWORK; LDA =600B; XMA RRL3; STA RAGRL
         LRR3; POT RRL3; SKN RABOL; BRU RAZ
         LDA RAXEOD; STA RASW
         LDA *+3; STA RAEX; BRU RALITX; BRU RAZZ
RAZ      EOD 10226B; PIN RATS; EOD 12226B
         PIN RATSA; LDA RATS; SUB RATSA; CAB; LDA ONE; SKB K40
         CNA; STA RARLPH; CAX; LDB RATS; SKB K40; EAX 1,2
         LDB RATSA; SKB K40; EAX 4,2; LDA RAPHTB+1,2; STA RAPH
         LDA *+3; STA RAEX; BRU RALIT; BRU *+1
         LDA *+3; STA RAEX; BRU RAI1+1; BRU *+1
RAZZ     LDA SLDAR; STA RAEX
         LDA RAGRL; STA RRL3; LRR3; POT RRL3
         BRR RAGO
SLDAR    LDA RARLX
RAGRL    ZRO
RAPHTB   DATA 0,1,1,2,3,2,0,3

         IF ALOG
LOGXX    DATA 75
```

```
LOGP    DATA -1
$LOGA   ZRO
$LOGB   ZRO
$LOGX    ZRO
$LOG     ZRO; MIN LOG
$LOG2   STA LOGA; STB LOGB; STX LOGX
        LDA RRL3; XMA LOGXX; STA RRL3; LRR3; POT RRL3
        LDX LOG; EAX* 0,2; LDA 0,2; EOR* LOG; ETR KS5; EOR* LOG
        LCY 3; LDB REAL; RCY 3; LDX LOGP; STA ELDATA,2
        BRX *+2; LDX ELDATA; STX LOGP; LDA RRL3; XMA LOGXX
        MLABEL; LDA LOGA; LDB LOGB; LDX LOGX
$LOG9   BRR LOG
        ENDF
RACHKW  BSS 1
RANOP   BRU RASWA
RAEOD   EOD 15201B
RAEODR  EOD 15201B,4
RAWEOD  EOD 2266B
RAREOD  EOD 2226B
RAXEOD  EOD 10026B
RARLX   BSS 1
RASCT   BSS 1
RAWSC   BSS 1
RAPMT   BSS 1;* HOLDS ARGUMENT DURING CALLS TO RES, REL, B AND W
RATS    BSS 1
RATSA   BSS 1
RAWORK  BSS 1;* 0 MEANS RAD CHANNEL IDLE. -1 MEANS BUSY.
RARLPH  BSS 1;* RELATIVE PHASE OF RADS.
RAPH    BSS 1;* CHOICE OF RAD AND ANGLE OF CURRENT JOB.
RACUR   DATA -1;* CURRENT PAGE
RACURX  BSS 1;*  PAGE ABOUT TO START.
RARHP   BSS 4;*  HIGH PRIORITY READ COUNTS
RARLP   BSS 4;*  LOW  PRIORITY READ COUNTS
RACHH   DATA KM1,=-1,=-1,=-1;*  READ QUEUE HEAD
$RPAGEH ZRO
$RPAGEL ZRO
$RAFR   RPT 32; DATA 3777600B; ENDR;* BIT MAP OF RAD
RABOL   BSS 1;*  -1 DURING LITTLE PART.
RADDR   BSS 1
```

```
CADD   BSS 1;*  CORE ADDRESS
RAA    BSS 1
RAB    BSS 1
RAX    BSS 1
RASGH  ZRB <M1;* SPACE QUEUE HEAD
RACLQH ZRB KM1;* CLEAR QUEUE HEAD
RAEXIT BSS 1
RARFG  DATA RARQ;*  FREE LIST HEAD.
RARQ   RPT 19; DATA **+2; ZRB; ENDR; DATA KM1; ZRB;* READ QUEUE SPACE.
RARN   BSS 1;*  CURRENT   READ PMT POINTER
RARNX  BSS 1;*  IMPENDING  '     '     '
$ARM205 DATA 2020008              ARM 205
$DIS205 DATA 575777B              DISARM 205
CLLA   ZRB
CLLB   ZRB
CLLX   ZRB
CLTS   ZRB
CLTSA  ZRB
$RMT   BSS NMEM
$RMC   BSS NMEM
$ERMC  BSS 0
RML    BSS NMEM
RML32  ZRB 33
RML33  ZRB 32
RMR    BSS NMEM
RMR32  ZRB 33
RMR33  ZRB 32
RAPSTS ZRB;* ERROR SENTINEL FOR RAPBST.
RCER   ZRB
RDER   ZRB
$RFR   ZRB
RACKWD ZRB 34000B
       IF SG
RANE   DATA -1;* NOT EMPTY SWITCH
       ENDF
RADDRC ZRB;* FOR FE'S TO LOOK AT IN A CRASH.
RADRCC ZRB;* RAD ADDRESS OF PREVIOUS PAGE WITH READ OR WRITE IND.
CADDC  0;* ADRS OF CORE BLOCK THAT RAD CHNL LAST STARTED TO USE
$ENDR  BSS 0
```

```
FRGT SN,SS,SC,SD,SB,SWQ,SW,SRQ,SR,SE,SI,SCQ
FORGET
END
```

```
RUP     IDENT   06/16/72                                        6/21/72  15:11     PAGE   1

        LISTM
OHM     EQU =1;* SUPPRESS FANCY HASH TABLE ADDRESSING
* STRING AND FLOATING POINT SYSPOPS  7/30/66

SCIT    OPD     1346B5
SWCI    OPD     1576B5          SYSPOPS WITH BIT 0 REMOVED ...
SCIO    OPD     1616B5             ... FOR USE BY SYSTEM MODE ROUTINES
SBRS    OPD     1736B5
SSKSE   OPD     1636B5
UGCI    MACRO   A
        EAX     A(1)
        BRM     GCU
        ENDM


*               USAGE COUNTERS FOR ARITHMETIC POPS
* UTILITY BRS'S (33,34,35,37)

*       BRS 33

$GETSTR STB     GSIN1
        STX     GSIN3
        ETR     =40037777B
        CAX
        LDB     1,6
        SKA     K4B7
        STB     0,6
        EAX     0,6
        STX     GSIN2
        BRU     *+2
GSIN4   SWCI*   GSIN2
```

```
         SCI0     GSIN3
         SKE      GSIN1
         BRU      GSIN4
         LDA      0,6
         LDB      1,6
         LDX      GSIN3
         BRU      EP6Px


*        BRS 34

$0UTMSG  STB      STR03
         MUL      THREE
         LSH      23
         SUB      ONE
         SKN      STR03
         BRU      TYM2          NOT SPECIAL MODE
         LDB      =137777B
         BRU      STR06
TYM2     CAB
         ADD      STR03
         XAB
         BRU      STR06


*        BRS 35

$0UTSTR  ADD      ZER0          CLEAR X\0
         STX      STR03
STR06    ADD      ZER0          CLEAR X\0
          STX      STR01
         ETR      K1S5
         STA      STR02
         CBA
         ETR      K1S5
         STA      STR021
STR04    UGCI     STR02
         BRU      STR08
```

```
         SKN     STR03
         BRU     STR07
         SKE     K17
         BRU     STR05
STR08    LDX     STR01
         BRU     EP8Px
STR05    SKE     F8UR
         BRU     STR07
         LDA     =155B
STR07    SKG     K137
         BRU     STR07A
         SKE     =147B          BELL
         BRU     *+2
         BRU     STR07A
         SKE     =155B
         BRU     *+2
         BRU     STR07A
         SKE     =152B
         BRU     *+2
         BRU     STR07A
         SKN     UEXFLG
         BRU     STR04
         CAB
         LDA     SIx            AMPERSAND
         SCI8    STR01
         CBA
         SUB     K100
STR07A   SCI8    STR01
         BRU     STR04
```

```
*WRITE CHARACTER AND INCREMENT. THE POP ADDRESSES A STRING POINTER, AND
*THE CHARACTER IN A IS WRITTEN ONTO THE END OF THE SPECIFIED STRING.
*THE SECOND POINTER IS INCREASED BY ONE, SO THAT THE POINTER NOW
*POINTS TO THE NEW STRING. X AND A ARE PRESERVED, B DESTROYED.
WCI     POPD    157B5
$WCIP   BSS     0
WCI1    ETR     K377
        STB WCH3
        STA WCH1                SAVE CHARACTER TO BE WRITTEN
        STX WCH2                SAVE X REGISTER
        EAX* 0                  GET EFFECTIVE ADDRESS OF POP
        MIN 1,6                 INCREMENT SECOND WORD OF STRING POINTER
        LDA 1,6                 AND PICK IT UP.
        BRU WCH5                GO TO WCH CODE, WHICH IS IDENTICAL FROM
*WRITE CHARACTER AND DECREMENT: DECR. 1ST PTR.
*AND WRITE CHAR. ON BEGINNING OF STRING
WCD     POPD    13585
$WCDP   BSS     0
WCD1    ETR K377
        STB WCH3
        STA WCH1
        STX WCH2
        EAX* 0
        LDA 0,6; SKR 0,6; NOP
        BRU WCH5
*                               THIS POINT ON.
*STORE POINTER. THE A AND B REGISTERS ARE STORED IN THE WORD ADDRESSED
*BY THE POP AND THE NEXT WORD
STP POPD 167B5
$STPP   STX STP1                SAVE INDEX REGISTER
        EAX* 0                  GET EFFECTIVE ADDRESS
        STA 0,6                 DO THE
        STB 1,6                 STORE
        LDX STP1                RESTORE X
        BRR 0                   RETURN
*LOAD POINTER. THE A AND B REGISTERS ARE LOADED FROM THE WORD ADDRESSED BY THE P
```

```
*ANF THE NEXT WORD.
LDP POPD 16685
$LDPP   CXA; EAX* 0; LDB 1,6; LDX 0,6; XXA; BRR 0


*       GET CHARACTER AND DECREMENT
*    TRANSFER CHAR. FROM OPERAND STRING TO A REGISTER. DECREMENT END-STR. PTR.

GCD     POPD    13785
$GCDP   BSS     0
GCD1    STB GCI4
        STX     GCI1
        EAX*    0
        LDA     1,6
        SKG     0,6
        BRU     GCI3
        SKR     1,6
        BRU     GCI8

*GET CHARACTER AND INCREMENT. THE POP ADDRESSES A STRING POINTER. IF THE
*FIRST WORD OF THE POINTER IS+=0R= TO THE SECOND, IT RETURNS WITHOUT
*       SKIPPING. OTHERWISE, IT SKIPS AND RETURNS WITH THE FIRST
*       CHARACTER OF THE STRING IN A AFTER INCRIMENTING THE FIRST POINTER
*       BY 1. X IS PRESERVED, B DESTROYED.
GCI     POPD 16585
$GCIP   BSS     0
GCI77   STB GCI4
        STX GCI1                SAVE X
        EAX* 0                  YES. DON'T HAVE TO WORRY ABOUT WHETHER
*                               ADDRESS IS RELABLED. GET IT
GCI9    MIN 0,6                 INCREMENT POINTER
        LDA 0,6                 AND GET IT
        SKG 1,6                 IS STRING NULL (1ST POINTER +== END)
        BRU GCI8                NO
        SKR 0,6                 YES. RESTORE POINTER
        NOP 0                   ALLOW FOR POSSIBLE SKIP
GCI3    LDX GCI1                RESTORE X
        LDB GCI4
        BRR 0                   AND RETURN WITHOUT SKIPPING
```

```
GCI8   MIN 0                    INCREMENT RETURN ADDRESS FOR SKIP ON RETURN
       MUL =12525253B           MULTIPLY CHARACTER ADDRESS BY 1/3. THIS
*                               LEAVES WORD ADDRESS IN A, CHARACTER
*                               ADDRESS IN TOP TWO BITS OF B.
       RCH     412B             CAX + BAC. WORD ADDRESS TO X
       LCY     2                MOVE CHARACTER ADDRESS TO BOTTOM OF B
       LDA     0,6              GET WORD
       RCH     24B              CBX + CAB
       EXU GCI2,2               SHIFT TO GET CHARACTER IN BOTTOM OF A
       ETR K377                 REMOVE SUPERFLUOUS BITS
       LDX GCI1                 RESTORE X
       LDB GCI4
       BRR 0                    AND RETURN
GCI2   LCY 8                    SHIFT TABLE FOR EXTRACTING A CHARACTER
       LCY 16                   FROM A WORD
       CBA
       BRM TRAP
* GCI WITH POINTER IN T.S. BLOCK
GCU    ZRO
       LDA GCU
       STA 0
       MIN 0,2
       LDA 0,2
       SKG 1,2
       BRU GCI8
       SKR 0,2
       NOP
       BRR GCU
*WRITE CHARACTER ON THE END OF STRING STORAGE. THIS IS THE ONLY POP WHICH
*      CAN CAUSE A GARBAGE COLLECTION. IT EXPECTS THE ADDRESS OF THE
*      PARAMETER TABLE IN X AND THE CHARACTER TO BE WRITTEN IN A AND
*      RESTORES BOTH. B IS DESTROYED.
WCH POPD 164B5
$WCHP  BSS     0
WCH77  STB WCH3
       ETR     K377
       STA WCH1                 SAVE CHARACTER
       STX WCH2                 AND INDEX (ADDRESS OF PARAMETER TABLE)
       EAX* 0
```

```
        MIN 0,6                 INCREMENT CURRENT LAST CHARACTER TO GET
*                                 CHARACTER ADDRESS FOR THE NEXT CHARACTER
        LDA 0,6                 GET THE ADDRESS TO A
        SKE 1,6                 IS IT EQUAL TO LAST ADDRESS IN STRING
*                               STORAGE$$
        BRU WCH5
        LDA KM1
        ADM 0,6
        EAX 2,6; CXA; ETR =40037777B; STA T; LDX WCH2; LDA WCH1
        LDB 0; BRU* T
WCH5    MUL =12525253B          COMPUTE WORD AND CHARACTER ADDRESSES
*                                  SEE GCI CODE
        CAX                     SAVE WORD ADDRESS
        LCY     5               CHARACTER ADDRESS SHIFTED 3 IN AC
        ETR <30                 REMOVE EXTRANEOUS BITS
        LDB     0,6             PICK UP WORD
        XXA                     SHIFT COUNT TO X
        LCY 8,2                 SHIFT IT TO PUT CHARACTER TO BE ALTERED
*                                  AT BOTTOM
        ETR =77777400B          MASK OUT THIS CHARACTER
        MRG WCH1                AND INSERT THE NEW ONE
        RCY 8,2                 SHIFT WORD BACK
        CAX                     RESTORE WORD ADDRESS
        STB     0,6             STORE WORD
        LDA     WCH1            RESTORE A
        LDX WCH2                AND X
        LDB WCH3
        BRR 0                   AND RETURN
*SKIP ON STRING EQUAL. THIS POP COMPARES THE STRINGS WITH POINTERS IN
*       AB AND IN THE TWO WORDS ADDRESSED BY THE POP. IT  IS
*       EXACTLY IDENTICAL TO THE MACHINE COMMAND SKE. ALL REGISTERS ARE
*       PRESERVED. THIS POP CHECKS FOR EQUAL LENGTH STRINGS FIRST
*       RETURNING WITHOUT A SKIP IF THE STRINGS ARE NOT EQUAL. IF THEY
*       ARE, IT CALLS SKCO TO FIND OUT ABOUT EQUALITY OF THIER CONTENTS
SKSE POPD 163B5
$SKSEP BSS      0
        STA SKS1
        STB SKS1+1
        STA SKS5                SAVE TWO COPIES
```

```
        STB SKS5+1          OF FIRST STRING POINTER
        STX SKS2            SAVE X
        SUB SKS1+1          COMPUTE
        STA SKS6            AND SAVE LENGTH
        EAX* 0              GET ADDRESS OF SECOND STRING POINTER
        LDA 0,6             COMPUTE
        SUB 1,6             ITS LENGTH
        SKE SKS6            AND COMPARE WITH FIRST
        BRU *+2             NOT EQUAL
        BRU SKSE2           EQUAL
        LDX SKS2            NOT EQUAL. RESTORE X,
        LDA SKS5            A AND
        LDB SKS5+1          B
        BRR 0               AND RETURN WITHOUT SKIPPING
SKSE2   LDB 1,6             LENGTH EQUAL. PICK UP SECOND
        LDA     0,6         STRING POINTER
        LDX     SKS2        AND ORIGINAL X
        BRM SKC0            COMPARE STRING IN AB WITH THE ONE IN SKS1
        BRU *+2             AB LESS
        MIN 0             EQUAL. INCREMENT RETURN ADDRESS FOR SKIP
SKSE1   LDA SKS5            GREATER. RESTORE A
        LDB SKS5+1          AND B
        BRR 0             AND RETURN
*SKIP ON STRING GREATER. THIS POP IS TO SKG AS SKSE IS TO SKE. IT CALLS
*       SKC0 IMMEDIATELY WITHOUT COMPUTING LENGTHS
SKSG POPD 16285
$SKSGP BSS      0
        STA SKS1            SAVE
        STB SKS1+1          TWO COPIES
        STA SKS5            OF FIRST STRING.
        STB SKS5+1
        STX SKS6            SAVE X
        EAX* 0                GET
        LDA 0,6             SECOND STRING
        LDB 1,6             POINTER
        LDX SKS6            RESTORE X
        BRM SKC0            DO COMPARISON
        MIN 0             SECOND STRING LESS, I.E. FIRST GREATER($.)
*                             INCREMENT RETURN ADDRESS FOR SKIP
```

```
        NOP 0                   EQUAL
        LDA SKS5                FIRST LESS. RESTORE A
        LDB SKS5+1              AND B
        BRR 0                   AND RETURN
*THIS ROUTINE COMPARES THE STRINGS IN AB (FIRST STRING) AND IN SKS1
*       (SECOND STRING). IT RETURNS WITHOUT SKIPPING IF THE FIRST IS
*       LESS, SKIPS ONCE IF THEY ARE EQUAL, AND SKIPS TWICE IF THE FIRST
*       IS GREATER. NOTE THAT SEVERAL OF THE NECESSARY GCI'S ARE WRITTEN
*       OUT INSTEAD OF BEING CALLS ON THE POP.
SKC0 ZRO 0
        STA SKS4                SAVE FIRST
        STB SKS4+1              STRING POINTER
        ADD K2000; SKG SKS4+1; BRM TRAP;* ANTI HOG MEASURE.
        STX SKS7                AND X
SKC01 MIN SKS1                  BEGIN LOOP. GET A CHARACTER
        LDA SKS1                FROM SECOND STRING
        SKG SKS1+1              (IS IT NULL$$
        BRU SKC04               NO
SKC02 MIN SKC0                  YES.) FIRST STRING CANNOT BE LESS, SO
*                                  INCREMENT RETURN ADDRESS
        MIN     SKS4            IS FIRST STRING NULL
        LDA     SKS4
        SKG     SKS4+1
        MIN     SKC0            NO. THEREFORE IT IS GREATER
        BRU     SKC05           RETURN
SKC04   MUL =12525253B          SECOND STRING NOT NULL. GET CHARACTER
        RCH 401B                AS
        LRSH 22                 IN
        LDA 0,6                 GCI
        RCH 24B                 CODE
        EXU GCI2,2              ABOVE
        ETR K377
        STA SKS3                GOT IT
        MIN SKS4                TRY TO GET CHARACTER
        LDA SKS4                FROM FIRST STRING
        SKG SKS4+1              IS IT NULL$$
        BRU *+2                 NO
        BRU SKC05               YES SO IT MUST BE LESS. RETURN WITHOUT
*                                  SKIPPING
```

```
         MUL =12525253B          NOT NULL. GET
         RCH 401B                CHARACTER
         LRSH 22                 AS
         LDA 0,6                 IN
         RCH 24B                 GCI
         EXU GCI2,2              CODE
         ETR K377               ABOVE
         SKE SKS3                AND COMPARE WITH CHARACTER OF 2ND STRING
         BRU SKC03               NOT UAL
         BRU SKC01               EQUAL. LOOP
SKC03    SKG SKS3                COMPARE FURTHER
         BRU SKC05               LESS. RETURN WITOUT SKIPPING
         MIN SKC9                GREATER SKIP
         MIN SKC0                TWICE
SKC05    LDX SKS7                RETURN POINT. RESTORE X
         BRR SKC0                AND RETUR


STP1     ZR0     0
GCI1     ZR0     0
GCI4     ZR0     0
GCI7     ZR0     0
WCH1     ZR0     0
WCH2     ZR0     0
WCH3     ZR0     0
SKS1     BSS     2
SKS2     ZR0     0
SKS3     ZR0     0
SKS4     BSS     2
SKS5     BSS     2
SKS6     ZR0     0
SKS7     ZR0     0
SCH1     BSS     2
SCH5     BSS     2
SCH4     ZR0     0
SCH9     ZR0     0
SCH10    ZR0     0              MASK FOR FIRST 3 CHARACTERS OF STRING
SCH11    ZR0     0              MASK FOR LAST 3 CHARACTERS OF STRING
SCH20    ZR0     0              STARTING LOCATION FOR SCAN OF HAST TABLE
```

```
SCH52   ZR0     0               FLAG IS =1 IF N0 =1 ENTRY HAS BEEN
*                                  ENC0UNTERED DURING SCH SCAN, THE INDEX
*                                  0F THE FIRST SUCH ENTRY ENC0UNTERED
*                                  0THEWISE


        IF      =D0IFP
* FL0ATING P0INT ARITHMETIC

* FAD   FL0ATING ADD
FAD     P0PD    156B5
$FADE   STA     SS01
        CLA
FASE    STX     SS03
        STA     FLAG
        STB     ZM
        STE
        STX     ZE
        LDX     SS03
        EAX*    0
        CLA
        SKE     0,6
        BRU     FAN
        CLAB
        BRU     FLAD
FAN     SKE     SS01
        BRU     FAX
    CLAB
        BRU     FLAC
FAX     CXA
        LDB     1,6
        STE
        XXA
        SUB     ZE
        SKG     KM1
```

```
         BRU     FLAGM
         SKA     KM100
         LDA     =39
         XMA     SS01
         LDB     ZM
         RSH*    SS01
         XAB
FLAC     SKN     FLAG
         BRU     FAS
         MRG     K777
         SUB     1,6
         E0R     K777
         XAB
         SUC     0,6
         BRU     FLAF
FAS      C0PY    A,E
         ADD     1,6
         XAB
         ADC     0,6
FLAF     STE
FLA0T    0VT
         BRU     FLA0S
         N0D     38
         SKA     KM1
         BRU     FLANZ
         CLX
FLANZ    XXA
         SKG     =-257
FLA0F    BRR     FLA0A          SET 0VERFL0W DEVI0USLY
FLA0K    XXA
FLAX     0VT
         BRU     FL0F
FLAX1    LDE
         LDX     SS03
         BRU     XP0P
FLA0A    DATA    10000000B+FLA0K-1
FLAGM    CNA
         SKA     KM100
         LDA     =39
```

```
         LDX     0,6
         XXA
         RSH     0,2
         COPY    8,E
FLAD     SKN     FLAG
         BRU     FLSS
         XMA     SS01
         XAB
         XMA     ZM
         SUB     ZM
         XAB
         SUC     SS01
         BRU     FLAF
FLSS     XAB
         ADD     ZM
         XAB
         ADC     SS01
         BRU     FLAF
FLA8S    RSH     1
         E8R     K4B7
         BRX     FLAX
         XXA
         SKG     =255
         BRU     FLA8K
         BRR     FLA8A          SET 8VERFL8W DEVI8USLY
FL8F     8RX     *+2
         BRU     FL8J
         CLEAR
         LDX     SS03
         BRU     XP8P
FL8J     E8R     K4B7
         RSH     39
         E8R     K4B7
         EAX     255
         BRR     FL8J1
FL8J1    ZR8     FLAX1-1,1
* FSB    FL8ATING SUBTRACT
FSB      P8PD    155B5
$FSBE    STA     SS01
```

```
         LDA      KM1
         BRU      FASE
* FNA FLOATING CNA (BRS 21)
$FNA     LDX      SS03
         BRM      FNAS
         BRU      XPOP
FNAS     ZRO
         STX      FNASX
         SKB      *=1000B
         BRU      FLNA
         CNA
         SKE      ZERO
         SKA      K1S7
         BRR      FNAS
         STE
         SKE      K4B7
         BRU      FLCOM
         RCY      1
         BRX      FLCA
FLCOM    NOP      4
FLCA     XXA
         SKG      =255
         SKG      *=257
         BRR      FNAOFL
         BRU      FLNB1
FNAOFL   ZRO      FNAOF2-1,1
FNAOF2   XXA
         EOR      K4B7
         RSH      39
         EOR      K4B7
         BRX      *+3
         EAX      255
         BRU      FLNB
         CLEAR
         BRU      FLNB
FLNB1    XXA
         ROV
FLNB     LDE
FLMX     LDX      FNASX
```

```
         BRR     FNAS
FLNA     STE
         XAB
         CNA
         XAB
         E0R     KM1
         BRU     FLNB
* ISC    INTERNAL T0 STRING C0NVERSI0N
ISC2     4   24+5000B
ISC      P0PD    140B5
$ISC1    STA     SS01
         LDA     ISC2
         BRU     SC1
* SIC    STRING T0 INTERNAL C0NVERSI0N
SIC2     4   25+5000B
SIC      P0PD    141B5
$SIC1    STA     SS01
         LDA     SIC2
SC1      STA     BSx
         STB     SS02
         STx     SS03
         EAX*    0
         CXA
         ETR     ADMSK
         STA     UBE
         BRU     BS4
* FMP    FL0ATING MULTIPLY
FMP      P0PD    154B5
$FMPE    STX     SS03
         STA     SS01
         STE
         STX     ZE
         BAC
         LDX     SS03
         EAX*    0
         LRSH    2
         MUL     0,6
         STA     ZM
         LDA     1,6
```

```
         CXB
         COPY     AX,A,E
         XXA
         ADM      ZE
         COPY     XA,BX,B
         LRSH     2
         MUL      SS01
         ADD      ZM
         MUL      TW0
         STB      ZM
         XMA      SS01
         MUL      0,6
         XAB
         ADD      ZM
         XAB
         ADC      SS01
         LDX      ZE
         SKA      K3S7
         BRU      FLEND
         SKB      =-1000B
         BRU      FLEND
         SKE      K4B7
         BRU      FLND2
         RCY      1
         BRX      FLEND
FLEND    NOD      4
         ROV
         XXA
         SKG      =255
         SKG      =-257
         BRR      FLA0A
         XXA
         LDE
FLND2    LDX      SS03
         BRU      XP0P

* FDV    FLOATING DIVIDE
FDV      P0PD     15385
$FDVE    STX      SS03
```

```
         STA     SS01
         STE
         STX     ZE
         LDX     SS03
         EAX*    0
         RSH     2
         DIV     0,6
         0VT
         BRU     FDV0
         STA     ZM
         BAC
         RSH     1
         STA     SS01
         LDB     1,6
         CXA
         STE
         XXA
         CNA
         ADD     TW0
         ADM     ZE
         BAC
         RCY     2
         CNA
         MUL     ZM
         ADD     SS01
         DIV     0,6
         MUL     TW0
         ADD     ZM
         LDX     ZE
         SKA     KM1
         BRU     FLEND
         BRU     FLND2
FDV0     LDA     SS01
         E0R     0,6
         BRU     FL0J
* STORAGE
ZE       ZR0
ZM       ZR0
FLAG     ZR0
```

```
        ELSE
* ISC   INTERNAL T0 STRING C0NVERSI0N
ISC2    4  24+5000B
ISC     P0PD    14085
$ISC1   STA     SS01
        LDA     ISC2
        BRU     SC1
* SIC   STRING T0 INTERNAL C0NVERSI0N
SIC2    4  25+5000B
SIC     P0PD    14185
$SIC1   STA     SS01
        LDA     SIC2
SC1     STA     BSX
        STB     SS02
        STX     SS03
        EAX*    0
        CXA
        ETR     ADMSK
        STA     UBE
        BRU     BS4
*
*       HARDWARE FL0ATING P0INT R0UTINES
*
*       M0ST SYSP0PS HAVE TW0 ENTRIES:
*       1. N0RMAL ENTRY
*       2. F0RTRAN II ENTRY WHICH D0UBLES THE INDEX REGISTER
*
*       ALL R0UTINES CL0BBER THE A AND B REGISTERS F0R MAXIMUM SPEED
*       X REGISTER IS PRESERVED
*
*       LDFM - FL0ATING L0AD FR0M MEM0RY  (52765)
$LDFME  C0PY    XB
        IF      FPH
$LDFME1 EAX*    0
        HFLD    (0,6),(1,6)
        C0PY    BX
        BRR     0
        ELSE
```

```
$LDFME1 STB     SS03
        EAX*    0
        LDA     0,6
        LDB     1,6
        LDX     SS03
        BRU     SFPEX1
        ENDF

$LDFMEF COPY    XB,A            (516B5)
        LSH     1
        COPY    XB,BX
        BRU     LDFME1



*       LDFL - LOAD FLOATING FROM AB REGISTERS (BRS 118)
        IF      FPH
$LDFLE  HFLD    SS01,SS02
        BRU     XP0PX
        ELSE
$LDFLE  LDX     SS03
        BRU     SFPEX1
        ENDF


*       STFM - FLOATING STORE INTO MEMORY (530B5)
$STFME  COPY    XB
        IF      FPH
$STFME1 EAX*    0
        HFST    (0,6),(1,6)
        COPY    BX
STFME2  BRR     0
        ELSE
$STFME1 STB     SS03
        LDA     FAC1
        LDB     FAC2
        EAX*    0
        STA     0,6
        STB     1,6
```

```
         BRU     XP0PX
         ENDF
$STFMEF  C0PY    XB,A          (51785)
         LSH     1
         C0PY    XB,BX
         BRU     STFME1


*        STFL - FL0ATING ST0RE INT0 AB REGISTERS (BRS 119)
         IF      FPH
$STFLE   HFST    SS01,SS02
STFLE2   BRU     P0PX
         ELSE
$STFLE   LDA     FAC1
         LDB     FAC2
         BRU     XP0PX
         ENDF


*        FFAD - FL0ATING ADD (55285)
$FFADE   C0PY    XB
         IF      FPH
$FFADE1  EAX*    0
         HFAD    (0,6),(1,6)
         C0PY    BX
         BRR     0
         ELSE
$FFADE1  BRM     SFSTUP
         BRU     FADE
         ENDF
$FFADEF  C0PY    XB,A          (52085)
         LSH     1
         C0PY    BX,XB
         BRU     FFADE1


*        FFMP - FL0ATING MULTIPLY (57285)
$FFMPE   C0PY    XB
```

```
         IF      FPH
$FFMPE1  EAX*    0              (BRU SFFMPE)
         HFMP    (0,6),(1,6)
         COPY    BX
         BRR     0
         ELSE
$FFMPE1  BRM     SFSTUP
         BRU     FMPE
         ENDF
$FFMPEF  COPY    XB,A           (52185)
         LSH     1
         COPY    XB,BX
         BRU     FFMPE1


*        FFDV - FLOATING DIVIDE (54685)
$FFDVE   COPY    XB
         IF      FPH
$FFDVE1  EAX*    0
         HFDV    (0,6),(1,6)
         COPY    BX
         BRR     0
         ELSE
$FFDVE1  BRM     SFSTUP
         BRU     FDVE
         ENDF
$FFDVEF  COPY    XB,A           (52285)
         LSH     1
         COPY    XB,BX
         BRU     FFDVE1


*        FFDI - FLOATING INVERSE DIVIDE (54785)
$FFDIE   COPY    XB
         IF      FPH
$FFDIE1  EAX*    0
         HFDVI   (0,6),(1,6)
```

```
         COPY    BX
         BRR     0
         ELSE
$FFDIE1  BRM     SFSTUP
         XMA     ARG1
         COPY    AB,BA
         XMA     ARG2
         COPY    AB,BA
         BRU     FDVE
         ENDF
$FFDIEF  COPY    XB,A            (51585)
         LSH     1
         COPY    XB,BX
         BRU     FFDIE1


*        FFSB - FLOATING SUBTRACT (54285)
$FFSBE   COPY    XB
         IF      FPH
$FFSBE1  EAX*    0
         HFSB    (0,6),(1,6)
         COPY    BX
         BRR     0
         ELSE
$FFSBE1  BRM     SFSTUP
         BRU     FSBE
         ENDF
$FFSBEF  COPY    XB,A            (52385)
         LSH     1
         COPY    XB,BX
         BRU     FFSBE1


*        FFSI - FLOATING SUBTRACT INVERSE (54385)
$FFSIE   COPY    XB
         IF      FPH
$FFSIE1  EAX*    0
```

```
         HFSBI   (0,6),(1,6)
         COPY    BX
         BRR     C
         ELSE
$FFSIE1  BRM     SFSTUP
         XMA     ARG1
         COPY    AB,BA
         XMA     ARG2
         COPY    AB,BA
         BRU     FSBE
         ENDF

$FFSIEF  COPY    XB,A            (524B5)
         LSH     1
         COPY    XB,BX
         BRU     FFSIE1

*        SKFN - SKIP IF FLOATING AC NEGATIVE (BRS 82)
         IF      FPH
$SKFNE   SHFAN
         BRU     XPOPX
         MIN     0
         BRU     XPOPX
         ELSE
$SKFNE   SKN     FAC1
         BRU     XPOPX
         MIN     0
         BRU     XPOPX
         ENDF

*        SKFZ - SKIP IF FLOATING AC ZERO (BRS 83)
         IF      FPH
$SKFZE   HFST    SS01,SS02
         LDA     SS01
         LDB     SS02
         ELSE
$SKFZE   LDA     FAC1
         LDB     FAC2
```

```
        ENDF
        SKE     ZERO
        BRU     XPOPX
        MIN     0
        BRU     XPOPX

*       SKFUZ - SKIP IF FLOATING AC NON-ZERO (BRS 84)
        IF      FPH
$SKFUZE HFST    SS01,SS02
        LDA     SS01
        LDB     SS02
        ELSE
$SKFUZE LDA     FAC1
        LDB     FAC2
        ENDF
        SKE     ZERO
        MIN     0
        BRU     XPOPX


*       FFNA - NEGATE FLOATING AC (BRS 148)
        IF      FPH
$FFNAE  HFMP    FMIN1,FMIN1+1
        BRU     XPOPX
FMIN1   DATA    487,0
        ELSE
$FFNAE  LDA     FAC1
        LDB     FAC2
        BRM     FNA
        BRU     SFPEX
        ENDF

*       FFIX - FIX FLOATING ACCUMULATOR TO A (BRS 149)
        IF      FPH
$FFIXE  HFST    SS01,SS02
        LDA     SS01
        LDB     SS02
```

```
         BRU     FFIX
         ELSE
$FFIXE   LDA     FAC1
         LDB     FAC2
         BRU     FFIX
         ENDF

*        FFLT - FLOAT A TO FLOATING AC (BRS 150)
         IF      FPH
$FFLTE   LDB     K27
         STB     SS02
         HFLD    SS01,SS02
         BRU     XPUPX
         ELSE
$FFLTE   CLB
         SKE     ZERO
         BRU     *+2
         BRU     *+4
         LDX     K27
         NOD     48
         LDE
         LDX     SS03
         BRU     SFPEX1
         ENDF


         IF      -FPH
*        STORAGE AND EXITS FOR SIMULATED ROUTINES
$SFPFG   DATA    -1              -1 IF SIMULATING NEW SYSPOPS
$ARG1    ZRO     0               ARGUMENT FOR FP SOFTWARE ROUTINES
$ARG2    ZRO     0

$SFPEX   SKN     SFPFG
         BRU     SFPEX2
SFPEX1   STA     FAC1
         STB     FAC2
         BRR     0
SFPEX2   SKR     SFPFG
```

```
        BRU     *+1
        BRR     0

SFSTUP  ZR0     0
        STB     SS03
        EAX*    0
        LDA     0,6
        STA     ARG1
        LDB     1,6
        STB     ARG2
        LDA     FAC1
        LDB     FAC2
        BRR     SFSTUP

IFSTUP  ZR0     0
        MIN     SFPFG
        STX     SS03
        STA     SS01
        EAX*    0
        LDA     0,6
        STA     ARG1
        LDA     1,6
        STA     ARG2
        LDA     SS01
        BRR     IFSTUP
        ENDF

*       ROUTINES FOR PERFORMING STANDARD FLOATING SYSPOPS
*       WITH THE FLOATING POINT HARDWARE
*
        IF      FPH
$HFADE  STX     SS03
        EAX*    0
        SKE     ZERO
        BRU     HFAD1
        LDA     0,6
        LDB     1,6
        BRU     XPOPX
```

```
HFAD1   STA     SS01
        STB     SS02
        HFLD    SS01,SS02       LOAD FA FROM CURRENT AB REGS
        HFAD    (0,6),(1,6)     ADD IN THE OPERAND
HFEX    HFST    SS01,SS02       RETURN RESULTS IN AB REGS
HFOVM1  BRU     POPX

$HFSBE  STA     SS01
        STB     SS02
        STX     SS03
        EAX*    0
        CLA
        SKE     0,6
        BRU     *+2
        BRU     POPX
        HFLD    SS01,SS02       LOAD FA FROM CURRENT AB REGS
        HFSB    (0,6),(1,6)     SUBTRACT THE OPERAND
        BRU     HFEX


$HFMPE  SKE     ZERO
        BRU     *+2
        BRR     0
        STA     SS01
        STB     SS02
        STX     SS03
        EAX*    0
        COPY    A,B
        SKE     0,6
        BRU     *+2
        BRU     XPOPX
        HFLD    SS01,SS02
        HFMP    (0,6),(1,6)     MULTIPLY BY THE OPERAND
        BRU     HFEX


$HFDVE  SKE     ZERO
```

```
         BRU     *+2
         BRR     0
         STA     SS01
         STB     SS02
         STX     SS03
         HFLD    SS01,SS02
         EAX*    0
         HFDV    (0,6),(1,6)    DIVIDE BY THE OPERAND
         BRU     HFEX
         ELSE
$HFADE   BRM     IFSTUP
         BRU     FADE

$HFSBE   BRM     IFSTUP
         BRU     FSBE

$HFMPE   BRM     IFSTUP
         BRU     FMPE

$HFDVE   BRM     IFSTUP
         BRU     FDVE
         ENDF

$HFNA    BRM     FNA
         BRU     XPOPX

         IF      FPH
*
*        HARDWARE FLOATING OVERFLOW ROUTINE
*
$FPOVI   ZRO     0
         STA     FPOV01
         RFPOV
         LDA     KM1
         STA     SFPOVI
         EOM     224000B        ARM THE USER MODE TRAP AND FLAG OVERFLOW
         LDA     FPOV01
         BRI     FPOVI
FPOV01   ZRO
```

```
        ENDF
*
*       FIX AND FLOAT
*
$FFIX SKD NOP24 (DIFFERENCE EXPONENTS AND SKIP IF NUMBER NOT TOO BIG)
        BRU     FIXBIG
        COPY    B,E
        RSH     7776,2 (X WILL ALWAYS BE >0. CONVERT TO AN INTEGER)
        BRU     XPOPX
FIXBIG  COPY    B,E
        LSH     1,2
        SKR     SFPOVI (TELL USER MODE TRAP THIS IS AN OVERFLOW)
NOP24   NOP     240
        EOM     22400B (ARM USER MODE TRAP)
        BRU     XPOPX
        IF -1   (OLD CODE AS OF 6/19/72)
$FFIX   SKA     X4
        BRM     FNA
        SKE     K27
        BRU     FIXBIG
        COPY    B,E
        RSH     0,2
FFIX0   SKN     SS01
        BRU     FFIX1
        COPY    BA,AB
        EOR     KM1
        ADD     ONE
        COPY    AB,BA
        COPY    N
        ADC     KM1
FFIX1   LDX     SS03
        BRU     XPOP
FIXBIG  COPY    B,E
        LSH     0,2
        ETR     XX
        BRU     FFIX0
        ENDF
*
*
```

```
$FFLT   CLB
        SKE     ZER0
        BRU     *+2
        BRU     *+4             GET XREG
        LDX     K27
        N0D     48
        LDE
        LDX     SS03
        BRU     XP0P
*
        IF      -FPH
*       FL0ATING P0INT ARITHMETIC S0FTwARE


*       FL0ATING ADD. w0RKS 0NLY 0N N0RMALIZED NUMBERS. TIME AVERAGES 55
*       CYCLES WITH0UT SIGNIFICANCE CHECK, THREE M0RE WITH IT, PLUS
*       CALL AND RETURN.

$FADE   SKE     ZER0
        BRU     FAD9
        LDA     ARG1
        LDB     ARG2
        BRU     FAD7
FAD9    STA     SS01
        STB     SS02
        LDA     ARG1
        SKE     ZER0
        BRU     FAD10
FAD13   LDA     SS01
        LDB     SS02
        BRU     FAD7
FAD10   LDB     ARG2
        SKD     SS02
        BRU     FAD1
*       0PERAND IN MEM0RY (0RIGINALLY IN REGISTERS) HAS BIGGER EXP0NENT
FAD8    RSH     0,2
        XAB
        C0PY    A,E
```

```
        ADD     SS02
*       ROUND SHOULD BE INCLUDED HERE
        XAB
        ADC     SS01
        STE
FAD14   OVT
        BRU     FAD2
        NOD     38
        SKE     ZERO
        BRU     FAD3
*       RESULT IS TRUE ZERO
FAD4    CLAB
        LDX     SS03
        BRU     SFPEX
*       RESULT IS NOT ZERO. TEST FOR UNDERFLOW
FAD3    RES
        OVT
        BRU     FAD4
*       NO UNDERFLOW OR OVERFLOW
FAD5    LDE
FAD7    LDX     SS03
        BRU     SFPEX
*       OVERFLOW ON FLOATING ADD
FAD2    RSH     1
        EOR     K4B7
        BRX     FAD5
*       OVERFLOW AND POSITIVE EXPONENT. POSSIBLE FLOATING OVERFLOW
        RES
        OTS
        BRU     FOVER
        BRU     FAD5
*       OPERAND IN MEMORY (ORIGINALLY IN REGISTERS) HAS SMALLER EXPONENT
FAD1    XMA     SS01
        XAB
        XMA     SS02
        XAB
        BRU     FAD8

*       FLOATING SUBTRACT. AVERAGES THREE MORE CYCLES THAN ADD
```

```
$FSBE   SKE     ZER0
        BRU     FSB2
        SKE     ARG1
        BRU     FSB3
        BRU     FAD7
FSB3    LDB     K400
FSB2    STA     SS01
        STB     SS02
        LDA     ARG1
        SKE     ZER0
        BRU     *+2
        BRU     FAD13
        LDB     ARG2
        SKD     SS02
        BRU     FSB1
        RSH     0,2
        XAB
        COPY    A,E
        XMA     SS02
        COPY    AX,E
        SUB     SS02
        XAB
        XMA     SS01
        SUC     SS01
        COPY    B,E
        BRU     FAD14
FSB1    XMA     SS01
        XAB
        XMA     SS02
        XAB
        RSH     0,2
        XAB
        MRG     K777
        SUB     SS02
        XAB
        SUC     SS01
        LDX     SS02
        COPY    XB,E
```

```
         COPY    X
         COPY    BX,B,E
         BRU     FAD14

*        DEFINITE FLOATING OVERFLOW OR UNDERFLOW
FOVER    BRX     FUNDER
         LCY     1
         LDA     XX
         RCY     1
         EOR     X4
         RSH     38
         EOR     X4
         STA     SS01
         STA     FAC1
         STB     SS02
         STB     FAC2
         SKR     SFPOVI
         BRU     *+1
         EOM     22400B          ARM THE USER MODE TRAP AND FLAG OVERFLOW
         LDX     SS03
         BRU     SFPEX
FUNDER   ROV
         BRU     FAD4

*        FMP  FLOATING MULTIPLY
$FMPE    STA     SS01
         STE
         STX     ZE
         SAC
         LDX     SS03
         LRSH    2
         MUL     ARG1
         STA     ZM
         LDA     ARG2
         CXB
         COPY    AX,A,E
         XXA
```

```
         ADM     ZE
         COPY    XA,BX,B
         LRSH    2
         MUL     SSO1
         ADD     ZM
         MUL     TWO
         STB     ZM
         XMA     SSO1
         MUL     ARG1
         XAB
         ADD     ZM
         XAB
         ADC     SSO1
         LDX     ZE
         SKA     XX
         BRU     FLEND
         SKB     *+1000B
         BRU     FLEND
         SKE     X4
         BRU     FLND2
         RCY     1
         BRX     FLEND
FLEND    NOD     4
         ROV
         REB
         OTB
         BRU     FOVER
         LDE
FLND2    LDX     SSO3
         BRU     SFPEX

*        FDV  FLOATING DIVIDE
$FDVE    STA     SSO1
         STE
         STX     ZE
         RSH     2
         DIV     ARG1
         BTO     0 (SKIP IF NO OVERFLOW BUT DON'T CLEAR IT. 03/02/72)
         BRU     FDVO
```

```
        STA     ZM
        BAC
        RSH     1
        STA     SS01
        LDB     ARG2
        CXA
        STE
        XXA
        CNA
        ADD     TW0
        ADM     ZE
        BAC
        RCY     2
        CNA
        MUL     ZM
        ADD     SS01
        DIV     ARG1
        MUL     TW0
        ADD     ZM
        LDX     ZE
        SKA     KM1
        BRU     FLEND
        BRU     FLND2
FDV0    LDA     SS01
        EOR     ARG1
        BRU     F0VER
*       STORAGE
ZE      ZR0
ZM      ZR0
        ENDF

*       BRS     21              FLOATING NEGATE
$FNA    ZR0     0
        SKB     =-1000B
        BRU     FLNA
        CNA
        SKE     ZER0
        SKA     K1S6
        BRU     FNA1
```

```
         STE
         SKE     X4
         BRU     FNA0V
         RCY     1
         BRX     FNA0V
FNA0V    N0D     4
         R0V
         RE0
         0T0
         BRU     FNA0VR
         LDE
         BRU     FNA1
FNA0VR   BRX     FNAUDR
         LCY     1
         LDA     XX
         RCY     1
         E0R     X4
         RSH     38
         E0R     X4
         SKR     SFP0VI
         BRU     *-1
         E0M     22400B          ARM USER M0DE TRAP T0 FLAG 0VERFL0W
         BRU     FNA1
FNAUDR   C0PY    A,B
         BRU     FNA1
FLNA     STE
         C0PY    AB,BA
         CNA
         C0PY    AB,BA
         E0R     KM1
         LDF
FNA1     LDX     SS03
         BRR     FNA
         ENDF

* PRINT UNSIGNED INTEGER
* BRS 36
* INPUT: A=NUMBER. B=RADIX. X=FILE N0.
*
```

```
$OUTNUM STA     OUTN02
        STB     OUTN03
        STX     OUTN05
        CBA
        SKG     ONE
        BRM TRAP
        CLA
OUTN06  STA     OUTN04
        LDA     OUTN02
OUTN07  STA     OUTN01
        LRSH    23
        DIV     OUTN03
        SKE     OUTN04
        BRU     OUTN07
        CBA
        ADD K27; SKG K40; SUB SEVEN
        SCIO    OUTN05
        LDA     OUTN01
        SKE     OUTN02
        BRU     OUTN06
        LDB     OUTN03
        LDX     OUTN05
        BRU     EPOPX
*
* DEMAND SIGNED INTEGER
* BRS 38
* INPUT: B=RADIX.  X=FILE NO.
* OUTPUT: A=NUMBER
*
$GETNUM STX     GETN01
        BAC
        STA     GETN02
        STB     GETN03
        STB     GETN07
GETNM   SCIO    GETN01
        SKE     K13   PLUS
        BRU     *+2
        BRU     GETN06
        SKE     K15   MINUS
```

```
         BRU     GETNM1
         LDA     KM1
         STA     GETN07
GETN06   SCI0    GETN01
         SUB     K20
         SKG     KM1
         BRU     GETN04
         SKG     GETN02
         BRU     GETN05
GETN04   ADD     K20
         CAB
         LDA     GETN03
         SKN     GETN07
         BRU     EP0PX
         CNA
         BRU     EP0PX
GETN05   SKE     GETN02
         BRU     *+2
         BRU     GETN04
         XMA     GETN03
         MUL     GETN02
         LSH     23
         ADM     GETN03
         BRU     GETN06

GETNM1   SKE     ZER0          IGN0RE LEADING SPACES
         BRU     *+2
         BRU     GETNM
         SKE     =1555         IGN0RE LEADING C.R.
         BRU     GETN06+1
         BRU     GETNM
         IF      -DDIFP
*
* FIX AND FLOAT
*
$FFIX    SKA     X4
         BRM     FNAS
         SKD     K27
         BRU     FIXBIG
```

```
         COPY    B,E
         RSH     0,2
FIXEND   SKN     SS01
         BRU     FEND1
         XAB; EOR KM1; ADD ONE; XAB; CNA; ADC KM1
FEND1    LDX     SS03
         BRU     XPOP
FIXBIG   COPY    B,E
         LSH     0,2
         ETR     XX
         BRU     FIXEND
*
$FFLT    CLB
         SKE     ZERO
         BRU     *+2
         BRU     FFLT9
         LDX     K27
         NOD     48
         LDE
FFLT9    LDX     SS03
         BRU     XPOP
         ENDF
$ENDRUP  BSS 0
         FORGET
         END
```

```
        LISTM

        ENTRY   SMT,SMTE,PMT,PMTM1,PMTE,GO
        ENTRY   PNEXT,PL,RL1,RL2,PPTR,PTEST,PTAB,PIM
        ENTRY   PNXTP1,PL4,PACT1,PPTR2,PPTRU
*
*
* THE FIRST ENTRIES MUST BE SET TO THE EXEC AND COMMON SUBSYSTEMS
* SMT/PMT FORMAT
*   BIT 0: 1=PAGE IS BEING OR HAS BEEN SHARED
*   BIT 1: 1=EXEC PAGE
*   BIT 2: 1=READ ONLY PAGE
*   BITS 3-8: PAGE STATUS. SEE RAD FILE FOR DETAILS.
*   BITS 9-18: RAD ADDRESS
*   BITS 19-23: CORE PAGE
* IF BITS 3-8=12, THE PAGE IS INDIRECT. BITS 10-23 POINT TO
*     THE REAL SMT/PMT.
$SMTPG7 2 7,2  POINTED TO BY RMT+7
SMT     DATA 40B
* MONITOR PAGES
        2 1,2; 2 2,2; 2 3,2; 2 4,2; 2 5,2; 2 6,2; 2 0,2
        2 103,2; 2 128,2
* INITIAL PAGES FOR EXEC AND FIO
        2 153,3; 2 163,3; 2 173,3; 2 203,3; 2 213,2
$NMSMT  EQU *-SMT
        BSS NSMT+SMT-*          SHARED MEMORY TABLE
SMTE    EQU *
        IF EXPMT
$USMT    EQU *-6            (SPECIAL PROPRIETARY SMT PAGES)
        ENDF
PMTM1   EQU *-1
PMT     BSS NUMEM*VJ9B1  PROGRAM MEMORY TABLE. NUMEM=35.
PMTE    BSS 0
        IF EXPMT
$EXSMT1 EQU *-1
$EXSMT  BSS NXSMT   EXPANDED SMT'S
```

```
$EXSMTE BSS 0
$EXSMTD EQU EXSMT-100B
XSMTEM EXT EXSMTE-1
       ENDF


QQ     EQU *
$QTI   DATA    QTI,QTI-PNEXT,QTI
$QTIC  DATA 0
$QIB   DATA    QIB,QIB-PNEXT,QIB
$QIBC  DATA 0
$QSG   DATA    QSG,QSG-PNEXT,QSG
$QSGC  DATA 0
$QQC   DATA    QQC,QQC-PNEXT,QQC
$QQCC  DATA 0
$QQE   DATA    QQE,QQE-PNEXT,QQE
$QQEC  DATA 0
Q1     EQU *

* PACT TABLE
       IF -1
       IDENT   01/12/70        (A4BIN85)/PAC-TABLE/    V10
PAC TABLE FORMAT
PNEXT: NEXT QUEUE OR NEXT FORK IN QUEUE. LT 0=NEXT FORK.
PL:    UUVUUUUUUUPPPPPPPPPPPPPP
RL1:   FIRST PSEUDO-RELABELING REGISTER
RL2:   SECOND PSEUDO-RELABELING REGISTER
PPTR:  DDDDDDDDDDDDKKKKKKKKKKKK
                OR
       UUUUUUUUUUCCCCCCCCCCCCCC
PTEST: UUUAAAAAAQTTTTTTTTTTTTTT
PTAB:  LSWJJJJJJJNNNNNNNNNNNNN
PIM:   MRIQEEEEEEEEEEEEXYFGGGOOH

A=ACTIVATION CONDITION
B=BRS 44 DELAY
C=CHAIN ADDRESS
D=LOWER FORK POINTER
E=PROGRAM INTERRUPTS
```

```
F=EXEC BRS FORK
G=FORK NUMBER
H=TIME OUT INTERRUPT GOING
I=NON-TERMINABLE
J=JOB NUMBER
K=UPPER FORK POINTER
L=LOCAL MEMORY
M=ADD NO MEMORY
N=PANIC TABLE ADDRESS
P=PROGRAM COUNTER
Q=NON-TERMINABLE USER MODE
R=TERMINATION PENDING
S=SUBSYSTEM STATUS
T=ACTIVATION TEST WORD
U=USER MODE
V=OVERFLOW
W=TS BLOCK ASSIGNED
X=EXEC STATUS
Y=SYSTEM STATUS

        ENDF
PACT    BSS NPAC*NPPAR
PACT1   EQU     PACT+NPPAR
PDATA   EQU *
PNEXT   EQU PDATA+0
PL      EQU PDATA+1
RL1     EQU PDATA+2
RL2     EQU PDATA+3
PPTR    EQU PDATA+4
PTEST   EQU PDATA+5
PTAB    EQU PDATA+6
PIM     EQU PDATA+7
*
$QD     EQU     QIO-QTI         NO. OF CELLS BETWEEN QUEUES.
$QTIQ   EQU     QTI-PNEXT
$QIOQ   EQU     QIO-PNEXT
$QSQQ   EQU     QSQ-PNEXT
$QQCQ   EQU     QQC-PNEXT
$QQEQ   EQU     QQE-PNEXT
```

```
PNXTP1  EQU PNEXT+1
$PNX3   EQU     PNEXT+3
PL4     EQU PL+3
PPTR2   EQU PPTR+PACT-PDATA
PPTRU   EQU PPTR-NPPAR
$ENDPMT BSS 0

        FORGET
        END
```

```
        LISTM

        DEFINE CbFO,CSIZE,CE0F,CPT0P,C_,CCkWRD
        DEFINE CFLG,CSw,CSw1,CE0F1,CC,C0V,CP
        DEFINE DbT0P,FBADR,FBWRD,FILE,GFILE,BUFF,LVL,CBIP,CP1,CP2
        DEFINE CRET0,CRET1,CRET2,CRET3,CRET4,CT1,CT2,CT3
        DEFINE CDSA,CDSB,CDSx,DSkMET

*       THE FOLLOWING TABLES MANIPULATE CP BITS DIFERENTLY ACCORDING TO
*       THE CURRENT VALUE OF LVL. SEE ;CP: AND COMMENT IN FILE M.
CMSK    DATA    0,7486,3785,7603,1777B
CSH     LRSH 24; LRSH 20; LRSH 15; LRSH 10; LRSH 0
CSH1    DATA 24,20,15,10,0
CMSK1   DATA    0,7486,3785,7603,0
BFP     DATA 0,1,41B,101B,141B,541B
BFC     DATA 1,40B,40B,40B,400B
BFCN    DATA -1,-40B,-40B,-40B,-400B

*                ********** OPEN AND CLOSE FILES
* INPUT: A=DISC ADDRESS DIVIDED BY 4, B=FILE PRIVELEGE BITS.
$BRS1   LDX =1-NFILE; LDA KM1; SKE ExBP,2; BRX *-1
        STX GFILE     (FIND GLOBAL FILE NUMBER)
        LDA SS02 (PRIVELEGE MASK); LDB KM400    (IF DISC IS FILLING
*       RESTRICT OUTPUT TO DELETES.EXEC SETS SS02=57B FOR DELETE COMMAND.
        SKA SIx; SKb DBITS; LDA =57B  (OK IF INPUT OR LOTS OF DBITS);
        SKB GFILE  (SKIP ON 0 IF XBP TABLE FULL); SKE =57B
        BRU CN8B    (EXEC TYPES DISC FULL MESSAGE FOR THIS ERROR)
        LDB SS01; LSH 26; ETR DMSK; STA SSC1  (MAKE XBLOCK POINTER TO
*       LOGICAL DISC ADRS OR REAL DATA PRODUCT DISC ADRS)
        BRM BGET (ASSIGN HIM A TS BFR); BRU CERR2 (ALL BFRS ARE BUSY);
*       INITIALIZE CURSOR FLAG WORD,GLFG AND,SET UP CFLG TO POINT TO GFLG
        LDA GFILE; ADD =NFILE; STA GFILE; ADD =GFLG; LDX BUFF; STA CFLG,2
        LDB KM1; STB CSw1,2
        LDB J0B; LSH 32; STA* CFLG,2; STB CP,2; STB CSw,2
        LDA FILE; SUB THREE; LSH 15; ADM* CFLG,2
        LDA SS02; ETR K377; MRG =-485; ADM* CFLG,2
        ETR SDBM8; SKA SIX; BRU CNERR-3   (BIT MAP NOT SET);
```

```
         LDA SS01; SKG ZER0; BRU CN3
         LDB* CFLG,2; SKB SIX; MRG X4; LDB KM2; STB T
         LDB DMSK; LDX =-NFILE OPEN FILE ONLY ONCE IF FOR OUTPUT
         SKM EXBP,2; BRU *+3; MIN T; MRG EXBP,2; BRX *-4
         LDX T; BRX CN3; SKA X4; BRU CNERR-1 (FILE BUSY);
CN3      BRM SETCC; LDA SS01; SKA DMSK; BRU CN6
         BRM CCLRX; BRM CG; STA T; CLA; STA LVL
CN4      EQU *   INITIALIZE BUFFERS FOR NEW FILE
         LDX LVL; LDA BUFF; ADD BFP,2; XXA    PUT A DISC ADRS AT THE
         LDB T; MIN T; STB 0,2                BEGINNING OF AN XBLOCK.
         ADD BUFF; CAX; MIN CC+1,2            MARK BFR FOR WRITE.
         LDA LVL; MIN LVL; SKG =MLVL-3; BRU CN4
         LDX BUFF; LDA =25252525B; STA CCKWRD,2
         LDA =257-1; STA CE0F,2
CN5      LDA* BUFF; ETR DMSK; STA SS03
         LDX GFILE; LDB GFLG,2; SKB SIX; MRG X4; STA XBP,2
CN7      LDX BUFF; SKN* CFLG,2; BRM CDMS
         LDA SS01; SKG ZER0; BRU CN8
         LDA =MLVL-1; STA LVL; BRM SETBIP
         LDX BUFF; LDA CCKWRD,2; LDB 0,2; SKE =25252525B; BRU CN8A
         LDB* CFLG,2; SKB =4B4; BRU CNERR-4
         SKB K377; BRM CR (READ DBLOCK IF NOT MAPPING)
CN8      BRM DTS
         BRM SCE0F1; LDA K200; STA C0V,2; LDA FSIZE; STA CQ,2
         DIR
*        CHECKSUM LEVEL 2
         LDA ONE; STA LVL; BRM SETBIP
         LDA CBIP; ADD BFC+1; CAX    (X POINTS TO LEVEL 2 BFR)
         LDA* CBIP
         DIR
         SKA X7; BRM CKSUM; EOR* CBIP; LDB* CBIP
         SKA X7; BRU CN8A
         EIR
         LDA FILE; STA SS01; LDA SS03; LRSH 2; STA SS03
C0K      MIN 0; BRM NP0PX
CN8A     CBA
         IF DISC=3; BRM DP2IBM; ENDF
         BRM IDER; MIN DF; EIR; BRU CNERR-4
```

```
CN8B    LDA TW0; SKA ALARM; BRU *+2; ADM ALARM
        BRU CERR5  SERIOUS, ALMOST OUT OF DBITS OR XBP TABLE FULL.

CN6     CLA; STA LVL     READ  ALL BUT LEVEL MLVL (DATA)
CN6A    BRM SETBIP; LDA SS01; ADD LVL; STA* CBIP
        LDA K286; BRM CRW; MIN LVL
        LDA LVL; SKG =MLVL-2; BRU CN6A; BRU CN5

        MIN CERRC; MIN CERRC; MIN CERRC; MIN CERRC
CNERR   LDA BUFF; BRM BPUT; BRM MONCR; LDA KM1; LDX GFILE; STA XBP,2
        BRU CERR

$BRS20  EQU *   CLOSE A FILE. RESET CIN FOR EXEC IF CLOSING COMMAND FILE.
        LDB ADMSK; SKM UCIN; BRU CCLS; CLB; STB UCIN; BRU CCLS

$BRS17  CLA; STA UCIN; LDB ONE; STB UCOUT; BRU CLSALL  (CLOSE ALL FILES)

$BRS147 LDA UCIN; LDB UCOUT; BRU CLSALL (CLOSE ALL FILES BUT UCIN,UCOUT)
$BRS77  EQU *   REDEFINE THE COUT FILE. CLOSE THE OLD COUT FILE.
        ETR ADMSK; XMA UCOUT; BRU CCLS

$BRS157 EQU *  COPY LVL 3 INDEX BLOCK TO USER'S MEMORY STARTING AT X
        BRM IOI     FILE NUMBER IS IN A
        CXA; ADD BFP+MLVL; ETR ADMSK; MRG =2765; STA B157A
        LDA SS03; ADD BFC+MLVL-1; ETR ADMSK; MRG =6355; STA B157A+1
        LDX BFCN+MLVL-1
B157A   LDA 0,2; STA 0,6; BRX B157A
        BRM NPOPX

$BRS151 EQU *  REDEFINE THE COMMAND FILE. CLOSE THE OLD COMMAND FILE.
        ETR ADMSK; XMA UCIN
$CCLS   EQU *   CLOSE 1 FILE
        ETR ADMSK; SKG TW0; BRM NPOPX
        BRM IOI; BRM CLS
        LDX J0B; SKN TTN0,2; BRU *+2; BRM CDMS
        LDA FILE; BRM CLS5; BRM NPOPX
```

```
CLSALL STA CT1; STB CT2
       LDA THREE; STA CT3
CL1    SKE CT1; BRU *+2; BRU *+3; SKE CT2; BRM CLS (RUNOUT BUFFERS);
       MIN CT3; LDA CT3; SKG =3+NBUF-1; BRU CL1
       LDX J0B; SKN TTN0,2; BRU *+2; BRM CDMS
       LDA THREE; STA CT3
CL2    SKE CT1; BRU *+2; BRU *+3; SKE CT2; BRM CLS5 (CLEAN UP);
       MIN CT3; LDA CT3; SKG =3+NBUF-1; BRU CL2
       BRM NP0PX

$BRS174 EQU *  WRITE ALL BUFFER FOR FILE A ONTO DISC
       BRM I0I; SKG TW0; BRM NP0PX (0,1, OR 2 NOT DISC FILES);
       BRM CWALL (WRITE LVL 1,2, AND 3); BRM CW (WRITE LVL 0); BRM NP0PX

$C256  EQU *                          SEQENTIAL I/O ROUTINE FOR WI0,CI0
       LDX SS03; LDA* 0; BRM I0I; LDX SS03; STA* 0 (CLEAN UP FILE WORD);
       BRM CE0FF; SKG CP,2; BRU C2 (END OF FILE);
       LDA CP1 CP1 IS WHERE THE FILE NEEDS TO BE POSITIONED.SEE WI0,CI0.
       BRM CP0S  POSITION THE FILE AND MAYBE GET DISMISSED.
       BRM C1
       LDA 0; ETR =40037777B; STA T; LDA* T  (IS HE EXECUTING A CIT)
       ETR =177B5; SKE =13485; BRM NP0PX (NO); MIN 0 (YES); BRM NP0PX
C2     LDA 0; ETR =40037777B; STA T; LDA* T; ETR =77B5
       LDX BUFF; LDB =27657537B; SKE =60B5 (WI0); LDB K137 (NOT A WI0);
       SKN CSW,2; STB SS01
       LDB =40205; SKR C0V,2; BRU FILINT
       LDA K200; STA C0V,2; BRM TRAP  (TOO MANY CI0,S OR WI0,S)

*              ******************** ERASE SYSP0P
$PCEX1 EQU *
       LDA SS01; BRM CPX
       SKN CSW,2; BRU CERR1     MAKE SURE MODE IS OUTPUT
       LDB* CFLG,2; SKB FOUR; BRU *+2; BRU CERR2 (LACKS ERASE PRIVELEGE)
       LDA CP,2; SKG CPX1; BRU *+2; BRU CERR3  (CANT ERASE BACKWARDS);
       LDA CPX1; SKG CPT0P,2; BRU *+2; BRU CERR4 (CANT ERASE PAST CPT0P)
       BRM PE (ERASE)
       BRM SCE0F1; BRM CKERR
       BRU C0K
```

```
*                    ********** SET CURSOR POSITION SYSPOP
$SCPX1 EQU *
       LDX BUFF; LDA =42B; SKN SS01; LDA =41B
       CLB; SKN EXEC1; CAB; SKM* CFLG,2
       BRU CERR1  (SCP PRIVELEGE NOT ENABLED OR WRONG READ/WRITE MODE)
       LDA CEOF,2; SKN SS01; LDA CPTOP,2; ADD ONE; STA T (MAX CURSOR POS)
       LDA SS01; BRM CPX; ADD ONE; SKG T; SKG CBOF,2
       BRU CERR2 (HE ATTEMPTED TO POSITION OUT OF LEGAL RANGE)
       LDA CPX1; BRM CPOS (POSITION AND MAYBE DMS)
       LDA SS01; RSH 23; STA CSW,2
       LDA K200; STA COV,2
       BRM C1; BRM SCEOF1; BRM CKERR
       BRU COK

*                    ********** READ CURSOR POSITION SYSPOP
$RCPX1 EQU *   READ CURSOR POSITION
       BRM CPRIV; LDA CP,2; BRM CPY; BRM NPOPXB

$SSPX1 EQU *   SET CURSOR SIZE. LIMIT THE FILE SIZE.
       BRM CWALL; BRM CW; LDX BUFF; SKN* CFLG,2; BRM CDMS
       LDA SS01; SKA X6; BRU CERR1
       ADD =1400B-1; RSH 23; DIV =1400B; SUB CQ,2
       STA T    HE IS REQUESTING THIS MANY DBITS
       LDA CQ,2; ADD DBITS; RSH 1   (HE MAY HAVE THIS MANY DBITS)
       XMA T; SKG T; BRU SSPX2
       LDA T; SKG CQ,2; LDA CQ,2; MUL =1400B/2; STB SS01; BRU CERR2
SSPX2  ADM CQ,2; BRU COK

$RSPX1 EQU *   READ CURSOR SIZE PARAMETERS
       LDX BUFF; SKN* CFLG,2; BRM CDMS
       LDA CSIZE,2; MUL =1400B/2; STB SS02   PYHSICAL SIZE IN CHARACTERS
       LDA CQ,2; MUL =1400B/2; STB SS03     REMAINING PHYSICAL QUANTUM
       LDA CPTOP,2; BRM CPY; ETR XX; BRM NPOPXB  HIGHEST ADRS WRITTEN


*      BIO      ********** BLOCK I/O ROUTINES

*      BIO PROCEDURE
```

```
*
*:BIOX1:
*       VALIDATE ARGUMENTS, INITIALIZE TS VARIABLES

*:B0:   EXIT IF BIO DONE
*       CALL BWIO IF A PARTIAL TS BUFFER NEEDS TO BE COPIED

*:B0C:   TS BUFFER CAN NOW BE DESTROYED
*       INITIALIZE BT4 WHICH SAYS HOW MANY DISC ADDRESSES ARE AVAILABLE
*       IN CORE MEMORY WITHOUT A CALL TO CPOS TO READ NEW XBLOCKS.
*       SET SS01 TO WORD COUNT FOR TOTAL NUMBER OF FULL DBLOCKS
*       TO BE READ OR WRITTEN.
*       GRAB MEMORY. SAVE RELABELLING FOR PAGE CROSS INTERRUPT ROUTINE.

*:B2:   B2 FIGURES OUT WHAT DISC COMMANDS ARE NEEDED TO DO BIO.
*       B2 CODE IS LESS THAN NEAT. MODIFY WITH CAUTION.
*       B2 CALLS B9 (THE DISMISS ROUTINE) IF
*                A CALL TO CPOS IS NECESSARY TO READ AN XBLOCK
*                OR THE WORD COUNT IS SUFICIENTLY DECREMENTED
*                OR :DLOCK: SAYS TOO MANY PAGES ARE LOCKED FOR THE DISC
*                OR THE DRQ LIST CONTAINS TOO MANY ENTRIES.
*  :B2W:         OUTPUT BIO
*       GRAB DISC SPACE IF DATA BLOCK POINTER IS ZERO
*       RELEASE DISC SPACE IF MEMORY BFR IS ZERO AND WRITE IS UNNECESSARY
*  :B2R:         CLEARS DATA BFR IF THE DISC ADDRESS OF THE DATA = 0.
*  :B2C:         B2C SETS UP FOR PAGECROSS IF PAGECROSSOVER IS IMMINENT.
*  :B2A:         OUTPUT THE PREVIOUS DISC COMMAND IF THE NEXT DISC ADRS
*       IS OUT OF SEQUENCE, OTHERWISE ADD 400B TO THE WORD COUNT THAT THE
*       NEXT COMMAND OUTPUT BY BTC WILL HAVE.

*:B9:   DISMISS ROUTINE
*       OUTPUT ALL DISC COMMANDS BUILT BY B2.
*       PUT ZERO AT THE END OF THE PAGE CROSSOVER USER MEMORY ADRS LIST.
*       DO THE FIRST PAGECROSS COPY LOOP IF IN OUTPUT MODE.
*  :B9A:         POSITION THE FILE AND GET DISMISSED MAYBE.
*       RESTART THE BIO WITH UPDATED VALUES OF SS01,SS03

*:B99: FINAL EXIT. POSITION THE FILE SO THAT ALL LEVELS ARE IN CORE
*       AT THE NEW POSITION. CLEAN UP AND RETURN TO THE USER.
```

```
*:BWIO: BWIO COPIES DATA BETWEEN THE TS BFR AND USER MEMORY.
*:BTC: ALL THIS LITTLE ROUTINE DOES IS MAKE ONE BIG DISC COMMAND
*       WITH THE WORD COUNT IN WR8 WHICH ACCUMULATES A BIG VALUE IF
*       LOTS OF CONSECUTIVE DISC ADDRESSES ARE IN THE XBLOCK THAT
*       HOLDS THE DBLOCK POINTERS.
*:BINC:  INCREMENT A FEW POINTERS
*:BSEZ: SEE IF A DBLOCK IS ALL ZERO
*:BPCT:  PAGE CROSSOVER TEST.SKIP IF PAGE BOUNDARY IS IMMINENT.
*:BVIATS:  B (BIO)   VIA (BY WAY OF)   TS (TS BFR)
*       COPY A TS DATA BFR TO OR FROM USER MEMORY
*:BRMC: DETERMINE THE 2 PAGES INVOLVED IN ANY PAGE CROSS.

*       BIO DICTIONARY
*SS01   WORD COUNT. SLOWLY GETS DECREMENTED TO 0 AS BIO PROCEDES.
*SS03   USER MEMORY ADRS. GOES UP AS BIO PROCEDES.
*WR4    POINTS TO A LIST OF USER MODE CORE ADDRESSES FOR BIO PAGE
*       CROSSOVER BLOCKS. THE LIST START AT WR4+1 AND IS VARIABLE
*       LENGTH ENDING WITH A ZERO.THE ADDRESSES ALL HAVE THE SIGN BIT ON.
*WR6    ADDRESS TO BE USED IN DISC COMMAND IF NOT GOING TO USER CORE.
*WR8    WORD COUNT USED WHEN BTC ASSEMBLES AND OUTPUTS A COMMAND
*       TO THE DISC.
*WR9    LESS THAN 400B. THE NUMBER OF BIO WORDS IN THE LAST DBLOCK.
*WR11,WR12      USER'S REAL RELABELLING AFTER ALL MEMORY IS GRABBED.
*WR13   POINTS TO THE DISC ADDRESS FOR EACH DISC COMMAND. UPDATED BY
*       :BTC:. WR13 GOES UP IN BIG JUMPS FOR WELL ORDERED FILES.


$BIOX1 EQU *
       SKN 0; BRM MBNCR
       LDX BUFF; SKN* CFLG,2; BRM CDMS (WAIT IF DATA IS MOVING);
       SKN SS01; BRU *+2; BRM NPGPX
       LDA SS03; ETR ADMSK; STA SS03; MRG X4; XMA SS03
       ADD SS01; SUB =400000B; COPY N,B; SKA X4; ADM SS01
       LDX =WR4; STX WR4
       STB 1,2; STB WR6; STB WR8; STB WR9; STB BTCA
       BRM CE9FP
       LDA CSW,2; STA WR5; LDA CP,2; STA CP1; SKA THREE; BRM TRAP
*      CHECK IF HIS SS01 IS TOO GREAT AND REDUCE SS01 IF NECESSARY
```

```
        SUB SS01
        SKA X4; BRU *+2; BRU *+4; ADM SS01; LDA =402B5; ADM C0V,2 (E0F)
        LDB SS01; LSH 26; ADD CP,2
        SKG CPT0P,2; BRU *+5; SKN WR5; BRU *+3; STA CPT0P,2; MIN CC+1,2

B0      LDA THREE; STA LVL; BRM SETBIP; STA WR13
B0A     LDA THREE; STA LVL; LDA SS01; SKG ZER0; BRU B99
        LDX BUFF; LDB CP,2; SKA KM400; SKB CMSK+MLVL; BRU BWI0

B0C     EQU *
        BRM CW; LDX BUFF; LDA KM1; STA CC+MLVL,2
        LDA CP1; ETR CMSK+MLVL-1; EXU CSH+MLVL-1
        CNA; ADD BFC+MLVL-1; STA BT4
        LDA SS03; STA BT1; ADD SS01; SUB ONE; STA BT2
        LDA SS01; ETR K377; STA WR9; CNA; ADM SS01
BGRAB   LDA BT2; SKG BT1; BRU BGRABA
        LDB* BT1; SKN WR5; STB* BT1; LDA K4000; ADM BT1; BRU BGRAB
BGRABA  LDB* BT2; SKN WR5; STB* BT2
        LDA RRL1; STA WR11; LDA RRL2; STA WR12
B2      LDA =MLVL-1; STA LVL   MAKE SURE PROPER CC FLAG IS REFERENECED
*               IF ANY LOW LEVEL ROUTINES CHANGE THE XBLOCK
        SKR BT4; BRU *+2; BRU B9
        LDA WR8; RSH 8; ADD WR13; STA BT3
        LDA SS01; SKG WR8; BRU B9
        LDA EIGHT; SKG DLOCK; BRU B9
        IF DISC<3     SEE IF DRQ LIST IS GETTING TOO FULL
        LDA DTXS1; ADD =NDRQ-3; SKG NDCL; BRU B9
        ELSE
        LDA RQ1A; SUB RQ3; SKA X4; ADD =NDRQ*3
        SKG =NDRQ-9; BRU *+2; BRU B9
        ENDF
        CLA; SKN WR5; BRU B2R
B2W     SKE* BT3; BRU *+5; BRM CG; STA* BT3; LDX BUFF; MIN CC+MLVL-1,2
        LDA SS03; ADD WR8; BRM BSEZ; BRU B2C
        LDB BT3; BRM CESW; LDX BUFF; MIN CC+MLVL-1,2
B2W1    BRM BTC; LDA BFC+MLVL; BRM BINC; MIN WR13; BRU B2
B2R     SKE* BT3; BRU B2C
        LDA SS03; ADD WR8; ADD BFC+MLVL; ETR ADMSK; MRG =636B5
```

```
        CLB; LDX BFCN+MLVL; STA *+1; STB 0,6; BRX *-1; BRU B2W1
B2A     LDX BT3; LDA 0,2; SKG ZER0; BRM M0NCR
        IF DISC<3
        SUB F0UR; LDB WR8; SKB KM1; SKE -1,2; BRM BTC
        ELSE
        RSH 23; DIV =4*8*20*200   DISC PACK CR0SS0VER IF B=0
        LDA WR8; SKB KM1; SKG ZER0; BRM BTC
        LDA 0,2; SUB F0UR; SKE -1,2; BRM BTC
        ENDF
B2B     LDA BFC+MLVL; ADM WR8; BRU B2
B2C     LDA BFC+MLVL; BRM BPCT; BRU B2A; BRM BTC
        LDA SS03; SKA K3777; BRU *+2; BRU B2A
        MIN WR4; STA* WR4
        DIR; BRM BRMC; BRM DL0CKI; LDX BRMC1; BRM DL0CKI
        LDX RLTS; BRM DL0CKI; EIR
        LDA BFC+MLVL; STA WR8; LDA BFP+MLVL; ADD BUFF; STA WR6
        MIN BT3; BRM BTC; BRU B2


B9      BRM BTC
        LDA =WR4; XMA WR4; C0PY AX,B; STB 1,2; SKA WR5; SKG =WR4; BRU B9A
        DIR; BRM BVIATS; BRM BRMC; BRM DL0CKD; LDX BRMC1; BRM DL0CKD
        LDX RLTS; BRM DL0CKD; EIR

B9A     EQU *
        CLA; XMA WR9; ADM SS01; LDA CP1; BRM CP0S; BRU BI0X1

B99     LDA CP1; BRM CP0S
        SKN 0; BRM M0NCR
        LDA C0V,2; ETR ADMSK; XMA C0V,2; ETR NADMSK
        MRG* 0; STA* 0; SKN* 0; MIN 0
        SKA WR5; MIN CC+MLVL,2
        BRM C1; LDA SS03; ETR ADMSK; STA SS03; BRM NP0PX8

BWI0    EQU *
        STA T; LSH 22; ETR K377; STA T2; SUB K400
        C0PY B,N, SKG T; STA T
        LSH 2; ADD CP,2; LDB CMSK+MLVL-1; SKM CP,2; MIN WR13
```

```
        SKB WR5; MIN CC+MLVL,2
        LDA T2; ADD BUFF; ADD BFP+MLVL; LDB SS03; SKN WR5; XAB
        ADD T; ETR =40037777B; MRG =23585; STA BWI01+1
        CBA; ADD T; ETR =40037777B; MRG =27685; STA BWI01
        LDA T; BRM BINC; CAX
BWI01   LDA 0,2; STA 0,2; BRX BWI01
        LDA CP1; BRM CP0S; BRU BI0X1


BTC     ZR5;* OUTPUT ONE BIG DISC COMMAND THAT COMBINES MANY DATA BLOCKS
        LDA WR8; SKG ZER0; BRR BTC
*       FIRST MAKE SURE THAT A PAGE BOUNDARY READ IS NOT IMMEDIATELY
*       FOLLOWED BY A TS BFR READ
        LDA WR6; SKG ZER0; BRU BTC1;* PREVIOUS I/O INTO TS BFR
        LDA BTCX; ETR =7486; SKE X2; BRU BTC1;* PAGE BOUNDARY
        XMA BTCA; LDB BTCB;  LDX =467+1; BRM DTC (DO SOME USELESS WORK)
BTC1    CLAB; BRM BPCT; LDB K486; LDA X2; SKA WR5; LDA =6286
        COPY AX,BX (CREATE INTERRUPT NUMBER AND READ/WRITE BIT)
        LDA WR8; RSH 5; COPY AB,XB (PUT IN WORD COUNT); COPY BX,A
        XMA WR6; SKG ZER0; LDA SS03; LDB WR13
        STA BTCA; STB BTCB; STX BTCX; SKB DMSK; BRM CDTC
        CLA; XMA WR8; BRM BINC
        LDA BT3; STA WR13; BRR BTC
BTCA    0;* LAST A REGISTER ARGUMENT FOR CALL TO CDTC
BTCB    0;* LAST B REGISTER ARGUMENT FOR CALL TO CDTC
BTCX    0

BINC    0;* UPDATE SS01 AND SS03 FOR BIO COMMANDS OUTPUTTED AND INCR. CP1.
        ADM SS03; CLB; LSH 2; ADM CP1
        MUL X7 (RSH 2,CNA); ADM SS01; BRR BINC

BSEZ    0;* SKIP IF BLOCK EQUAL ZERO
        ADD K400; ETR =40037777B; ADD BZ1; LDB KM1; LDX KM400
        STA *+1; SKB 0,2; BRR BSEZ; BRX *-2; MIN BSEZ; BRR BSEZ
BZ1     SKB 0,2

BPCT    0
        MIN BPCT; ADD SS03; ADD WR8; SUB ONE; EOR SS03
        SKA K4000; BRR BPCT; BRU* BPCT
```

```
BT1     0
BT2     0
BT3     0
BT4     0

*                   ********** END OF BIO ROUTINES

* READ OR WRITE ON DISC (BRS 124,125)
* INPUT: A=CORE ADDR.  B=DISC ADDR.  X=WORD COUNT.
* FLAGS KEPT BY JOB IN TTNO.
$ARDD   LDX X1; BRM DISCRW      BRS 122. READ  DISC. DONT DISMISS.
        BRU ARWDD
$AWDD   LDX X5; BRM DISCRW      BRS 123. WRITE DISC. DONT DISMISS.
ARWDD   LDA ARMOT; SUB XARMOT; SKG =NDRQ-2      IS DRQ ALMOST FULL
        BRM NPOPX       (NO, LET HIM KEEP RUNNING)
        BRU ARD2 (YES, DISMISS HIM. HE MAYBE DOING LOTS OF BRS 122,123S)
$AWD    LDX X5; BRU ARD1                BRS 125.WRITE DISC AND DISMISS.
$ARD    LDX X1                          BRS 124.READ  DISC AND DISMISS.
ARD1    BRM DISCRW
ARD2    MIN 0; BRM CDMS1        DISMISS UNTIL IO IS COMPLETED
DISCRW  0;* READ OR WRITE ON DISC
        STX T2
        SKA NADMSK; BRM TRAP                    BAD CORE ADRS
        LDA SS03; SKG K4000; SKG ZERO; BRM TRAP     BAD WORD COUNT
        SKA K77; BRM TRAP                       BAD WORD COUNT
        ADD SS01; SUB ONE; EOR SS01; SKA K4000; BRM TRAP  PAGE CROSSOVER
        LDX SS01; LDA 0,6; STA 0,6
        LDA SS01; BRM DTH
        LDA SS03; RSH 5; MRG T2; CAX; LDA T; LDB SS02
        BRM DTC; BRM DTS; BRR DISCRW


*                   ********** BIT MAP ROUTINES
TABLEN EQU TABLEN   THIS SYMBOL USED IN LOAD MAP

GW      EQU 7408                    NUMBER OF MAPPED BLOCKS ON A DISC
GA      EQU *   DATA BLOCK ADDRESSES ON DISC 0 THAT DELIMIT 4 MAP ZONES
        DATA    77B,77B+GW,77B+2*GW,77B+3*GW,77B+4*GW
```

```
GAP     0 GA
GA1P    0 GA+1
GBRAN   0;*     RANDOM. HELPS SMOOTH DISC ALLOCATION. SEE GETBLK.
MAPLBL  DATA RDP*100B+RWP       MAP DISC PAGE IN M6 AND W PAGE IN M7
BITSUM  0;*     BIT MAP CHECKSUM

        IF DISC=1

$GETBLK 0;*     PUT ADDRESS OF AVAILABLE DISC BLOCK IN A
        LDA GB6A; STA *-1
        LDA REAL; RSH 23; DIV =TABLEN; STB GBRAN   SET GBRAN ONCE PER DAY
GB6     SKR DBITS; BRU *+4; LDA FIVE; ADM DBITS; BRM TRAP
        LDA MAPLBL; BRM RLABEL
        LDA DGET; ADD GBRAN     FIND GOOD PLACE TO START LOOKING FOR A FREE BIT
        MRG =4B4
GB1     AXC
GB3     SKE BITMP,2; BRU *+2; BRX GB3
        CXA; ETR ADMSK; SKG =TABLEN-1; BRU GB2
        LDA =4B4; BRU GB1    WRAP AROUND BIT MAP
GB2     EQU *   BIT FOUND.CONVERT WORD POSITION INTO BIT POSITION.
        MUL K30; LSH 23; STA GB9
        LDA BITMP,2; LRSH 1; STX GB8; CLX; NOD 23; CXA; CNA; ADM GB9; CAX
        LDA X4; RSH 0,2; LDX GB8; ADM BITMP,2; ADM BITSUM; LDA GB9
*       CONVERT BIT POSITION INTO BLOCK ADDRESS
        RSH 23; DIV =GW;* DISC IN A. SECTOR DISPLACEMENT IN B.
        STB GB8; MUL K4000; LSH 23; ADD* GAP; ADD ONE  BEGINNING OF DISC
        ADD GB8; RSH 22   DISC ADRS IN B
        LDA T4; BRM RLABEL; CBA
        BRR GETBLK
GB8     0
GB9     0

GB6A    BRU GB6

LOCBIT  0;* CONVERT DISC ADDRESS IN A TO BITMAP WORD ADRS IN LB
*       AND BIT IN A.
*       NO SKIP IF GARBAGE. 1 SKIP IF UNMAPPED. 2 SKIPS NORMALLY.
        SKA =77000003B; BRR LOCBIT (GARBAGE);
        MIN LOCBIT; RSH 2; STA LB9
```

```
        ETR K3777    GET ARM POSITION
        SKG* GA1P; SKG* GAP; BRR LOCBIT (PUT IN HOLE TABLE)
        LDA MAPLBL; BRM RLABEL
        LDA LB9; RSH 23; DIV K4000     DISC NUMBER
        MUL =GW; LSH 23; STA LB8 (BIT NUMBER OF BEGINNING OF DISC
        LDA LB9; ETR K3777; SUB* GAP; SUB ONE; ADD LB8   BIT POS. IN MAP
        RSH 23; DIV K30; ADD =BITMP; STA LB    WORD POSITION
        LDA X4; CBX; LRSH 0,2; MIN LOCBIT; BRR LOCBIT
LB9     0;*      INPUT TO LOCBIT
LB8     0
LB      0;*      POINTS TO WORD IN BIT MAP.
        ENDF DISC=1

        IF DISC<3

* ENTER COMMAND IN LIST AND LOCK MEMORY BLOCK AND DECREMENT TTNO.
* A= ABS CORE ADDR
* B= DISC ADDR
* X= WORD COUNT
$DTC    ZRO
        SKE DTCA; BRU DTC2;* READ IS REQUESTED IN SAME PAGE AS PREVIOUS.
        STA DTCA; CXA; XMA DTCX; STB DTCB; ETR =74B6; SKE X2; BRU DTC8
        MIN DTCSW;* LAST CMND = BIO PAGECROSS.STALL FOR TIME. SEE ;IDBPR:
        LDA DTCA; LDB =740210B (GARBAGE DUMP AREA); LDX =4B7+4; BRU DTC3
DTC2    STA DTCA; STB DTCB; STX DTCX
DTC3    STB* EDCL; XAB; ETR DMSK; SKG DSCTOP; BRU *+2; BRM TRAP
        MIN ARMBT; MIN DTXS1
        CXA; LDX EDCL; STX DTXS2
        STB 1,2 (CORE ADRS); STA 2,2 (WORD COUNT,ETC); LCY 13; ETR K37
        XXA; ADD THREE; SKE =DRQU; BRU *+2; LDA =DRQ; STA EDCL
        DIR; BRM DLOCKI; EIR
        LDB JOB; LSH 37; LDX DTXS2; ADM 2,2
        LDX JOB; LDA =-285; ADM TTNO,2
        LDA =NDRV-2; SUB DTXS1; SKG NDCL; BRM MONCR
        IF ST
        DIR; SKN STSW; BRU DTC9
        LDA =RDP*1000+RSP; BRM RLABEL
        BRM SSS
        2 =485+5
```

```
         11B DTxS2
         1 REAL
DTC9     EIR
         ENDF
         SKN DTCSw; SKR DTCSw; BRR DTC
DTC8     LDA DTCA; LDB DTCB; LDX DTCX; BRU DTC3
DTCSw    DATA -1 SWITCH TO HELP DTC CALL ITSELF
DTCA     0;* ARGUMENT FOR NORMAL CALL TO DTC
DTCB     0
DTCx     0

$DTS     ZRO;* START DISC IF IT IS NOT ALREADY EXECUTING A COMMAND CHAIN.
         LDA KM1; XMA DTxS1; SKG DTxS1; BRR DTS
         ADD ONE; DIR; ADM NDCL; SKG NDCL; BRU DTS1
         SKN IDCL1; BRU DTS1; MIN DSw; LDA UNIT; SKG ZERO; BRM IDMGO
DTS1     EIR; BRR DTS

         ENDF DISC=3

         IF DISC>1

$GETBLK  0;*     PUT ADDRESS OF AVAILABLE DISC BLOCK IN A
         LDA GB6A; STA *-1
         LDA REAL; RSH 23; DIV =TABLEN; STB GBRAN   SET GBRAN ONCE PER DAY
GB6      SKR DBITS; BRU *+4; LDA FIVE; ADM DBITS; BRM TRAP
         LDA MAPLBL; BRM RLABEL
         LDA DGET; ADD GBRAN     FIND GOOD PLACE TO START LOOKING FOR A FREE BIT
         MRG =484
GB1      AXC
GB2      SKE BITMP,2; BRU *+2; BRX GB2
         CXA; ETR ADMSK; SKG =TABLEN-1; BRU GB3
         LDA =484; BRU GB1
GB3      MUL K14; STB GB9
         LDA BITMP,2; LRSH 1; STX GB8; CLX; NOD 23; CXA; CNA; ADM GB9
         CAX; LDA X4; RSH 0,2; LDX GB8; ADM BITMP,2; ADM BITSUM
         LDA T4; BRM RLABEL
         LDA MAPNUM; MUL =8*20*200/2; BAC; ADD GB9; LSH 2
         BRR GETBLK
GB6A     BRU GB6
```

```
L0CBIT 0;* C0NVERT DATA PR0DUCTS SECT0R ADRS INT0 BIT MAP P0INTER
       SKA =74000003B; BRR L0CBIT; MIN L0CBIT
       ABC; DIV =32000*2;* A =DISC NUMBER
       SKE MAPNUM; BRR L0CBIT; MIN L0CBIT
       LDA MAPL0L; BRM RLABEL
       CLA; DIV K30; ADD =BITMP; STA LB
       CBX; LDA X4; LRSH 0,2
       BRR L0CBIT
LB     0

       ENDF DISC>1

       IF DISC=3

$DTC   0
       MIN ARN0T; STB DTC0SV; STB* RQ1A
       XAQ; SKA =3777774B; BRU *+4; SKE X7; BRM TRAP; CLA
       ETR DMSK; SKG DSCT0P; BRU *+2; BRM TRAP
       CXA; LDX RQ1A; STB 1,2; STA 2,2
       SKA SEVEN; BRU *+4; LDB DTC0SV; SKB THREE; BRM TRAP ; BRU DTC1
       SKA =170B; BRM TRAP  (N0 DATA CHAINING WITH LARGE WC);
       RSH 1; ETR THREE; STA T; LDA 0,2; ETR THREE; ADD T; SKG F0UR; BRU DTC1
       BRM TRAP (A SMALL WC REQUEST 0VERLAPPED A 2314 PHYSICAL REC0RD)
DTC1   LDB J0B; LSH 37; ADM 2,2
       LDA 0,2; BRM DP2IBM; LDX RQ1A; STA 0,2
*      PUT 4 BIT F0LDED CHECKSUM INT0 DRQ S0 620 CAN CHECK MIC IN P0LLD
       E0R 1,2; EOR 2,2; STA T; LRSH 8; E0R T; STA T; RSH 16
       E0R T; STA T; RSH 4; E0R T; LSH 16; ETR =36B5
       ADD 1,2; XMA 1,2
       RSH 11; CAX; BRM DL0CKI
       LDX J0B; LDA =-285; ADM TTN0,2; SKN TTN0,2; BRM M0NCR
       EAX RQ1A; DIR; BRM INCRQ; EIR
       BRR DTC
DTCBSV 0;* C0PY 0F UNTRANSLATED DISC ADRS. G00D F0R DEBUGGING PURP0SES.

DP2IBM 0;* C0NVERT DP ADRS IN A T0 IBM ADRS IN A
```

```
*       THERE ARE 32 100 WORD RECORDS PER TRACK
*       THERE ARE 20 HEADS PER CYLINDER
*       THERE ARE 200 CYLINDERS PER DISC
*       FIRST WORD OF 3 WORD DRQ ENTRY SHOULD BE:
*       BITS 0-2  CHECKSUM OR 0, BITS 3-5   DISC NUMBER
*       BITS 6-13  CYL#, BITS 14-18  HEAD, BITS 19-23 FOR 100 WORD RECORD
        STA T; ETR X7 (CHECKSUM); XMA T; ETR DMSK
        RSH 23; DIV =32*20*200   PUT DISC PACK IN A
        COPY BX,B; LSH 18; ADM T   (ADM PACK NUMBER ONLY)
        CXA; RSH 23; DIV =32*20   (CYLINDER IN A)
        LSH 10; ADM T   (CYL)
        LSH 14; ADD T   ADD TO HEAD AND RECORD
        BRR DP2ILM

$DTS    0
        LDB RQ1A; STB RQ1; BRR DTS

        ENDF

$BRS66  EQU *   DELETE A FILE
        SKA SDBM8; BRM NPOPX
        LDB ONE; BRM CDEL1; BRM PE   DELETE ALL BLOCKS
        LDA* BUFF; ETR DMSK; STA B66A; LDB =B66A; BRM CE (REL. TOP XBLOCK
        LDA KM1; LDX BUFF; STA CSW1,2
        BRU CBK
B66A    0

$GRABB1 EQU *  BRS 54. GRAB A BIT FROM THE BIT MAP AND RETURN DISC ADRS
*   RETURN ALWAYS SKIPS
        SKN SDBM8; BRU *+2; BRM NPOPX (ERROR);
        BRM GETBLK; MIN 0; BRM NPOPXB

$GRABB2 EQU *  BRS 25. GRAB A PARTICULAR BIT IN MAP.INPUT IS DISC ADRS.
        SKA SDBM8; BRM CGRB1; BRM NPOPX; BRU CBK

$GIVEB EQU *   BRS 30.GIVE BIT TO BIT MAP. DISC ADDRESS IN A.
        SKN SDBM8; BRM CDEL3; BRM NPOPX; BRU CBK

$LOCBLK EQU *  BRS 41. RETURN DISC ADRS OF CURRENT DATA BLOCK
```

```
        LDA THREE; STA LVL; BRM SETBIP; LDA* CBIP; BRM NP0PXB

$BRS92  EQU *   SET FSIZE, THE DISC SPACE QUANTUM PARAMETER FOR FILES.
        SKG =287-1; SKG =1400B*5-1; BRM TRAP
        ADD =1400B-1; RSH 23; DIV =1400B; STA FSIZE; BRM NP0PX

$SDBM   EQU *
        BRU *,2; BRU SDBMA; BRU SDBMB; BRU SDBMC

SDBMBD  LDA KM1; STA SDBM8; BRU C0K

SDBMA   EQU *   SET MAP NUMBER. DETERMINE BIT MAP ZONE.
        SKA =-10B; BRM TRAP
        STA MAPNUM  STORE THE NUMBER OF THE CURRENT MAP ZONE.
        IF DISC=1; CLB; ELSE; MUL =32000/2; ENDF
        STB T  INITIAL 2314 RECORD TO BE MAPPED
        LDA MAPLBL; BRM RLABEL
        LDA =-25; STA DBITS  MAKE DBITS RUN OUT EARLY.SEE GETBLK.
        LDA X4; LDX MAPNUM; LRSH 24+3,2; STB MAPBIT
        IF DISC=1; EAX GA,2; STX GAP; EAX 1,2; STX GA1P; ENDF
        LDX =-TABLEN; STA EBITMP,2; BRX *-1   CLEAR BITMAP
        LDA =TABLEC/8-1; STA T2; LDB =776B5   X CONTAINS 0
SDBMA1  LDA T; SKA =-255; BRU SDBMA3  SKIP IF POSSIBLY IN EXEC AREA
        IF DISC>1; ETR K3777; SKG =3677B; SKG K77; BRU SDBMA2; ENDF
SDBMA3  LDA DSCTOP; SKG T; BRU SDBMA4 (NO MORE PHYSICAL DISC SPACE);
        LDA EIGHT; ADM DBITS; CBA; ADM BITMP,2; ADM BITSUM      FREE 8 BITS
SDBMA2  CBA; RCY 8; SKB X4; EAX 1,2; LDA EIGHT; ADM T; SKR T2; BRU SDBMA1
SDBMA4  BRU C0K

SDBMB   EQU *   MAP ALL THE DISC SPACE FOR A FILE.
        CLB; BRM CDEL1   SET UP SIMILIAR TO BRS 66
        BRM PE                    CALL DELETE LOGIC
        BRM SETCC; LDA KM1; STA CSW1,2
        LDA 0,2; BRM CGRB1 (MAP TOP xBLOCK); BRM MP0PX; BRU C0K

SDBMC   EQU *   SET ACCOUNTING POINTER. MIN SDBM8 TO INDICATE MAP DONE.
        MIN SDBM8; BRU C0K
```

```
*       END OF BIT MAP ROUTINES  **********
*                ********** LOW LEVEL DISC SUBROUTINES
*
* SETS FILE, BUFF, GFILE, CPTOP
* 10I DISMISSES IF FILE BUSY AND ACTS ON POSSIBLE EXCEPTIONAL CONDITIONS
* INPUT: A=LOCAL FILE NUMBER
* OUTPUT: A=LOCAL FILE NUMBER. X=TS BLOCK BFR ADRS.
$10I    0;* VALIDATE LOCAL FILE NUM. DMS IF BFR BUSY. TRAP ON I/O ERRORS
        ETR ADMSK; SKG TWO; BRR 10I; SKG =2+NBUF; BRU *+2; BRM TRAP
        STA FILE; CAX; LDX BUF3,2; STX BUFF
        LDA CFLG,2; SKG ZERO; BRM TRAP  (FILE NOT OPENED)
        SUB =GFLG; STA GFILE    (INDEX IN GLOBAL GFLG TABLE)
        LDA =4B4; SKN GFILE; SKA* CFLG,2; BRU 10I2
        SKN CSW,2; BRU 10I1
        LDA CP,2; SKG CPTOP,2; BRU *+3; STA CPTOP,2; MIN CC+1,2
10I1    SKN* CFLG,2; BRM CDMS1; LDA FILE; SKN CQ,2; BRR 10I
        LDA =MLVL; STA CQ,2; LDB =410B5 (QUANTUM EMPTY FLAG); BRU FILINT
10I2    EQU *   FILE NOT OPENED OR UNRECOVERABLE ERROR
        SKN FILE; BRM CKERR; BRM TRAP

CKERR   0;* GENERATE SOFTWARE INT OR TRAP IF FILE I-O ERROR HAS OCCURED.
        LDA =4B4; SKA* CFLG,2; BRU *+2; BRR CKERR (NO ERROR, DO NOTHING);
        COPY N,B; ADM* CFLG,2 (CLEAR ERROR FLAG)
        STB CSW,2   ENTER READ MODE
        LDA =4045; BRU FILINT

CDMS    0;* CALL POPDMS. ACTIVATE AT :CDMS2: WHEN DISC IS DONE.
        STA CDSA; STB CDSB; STX CDSX; BRM DTS
        LDA CDMS; STA CRET1 (:CDMS2: WILL DO A BRR CRET1 AFTER AWHILE)
        LDA =CDMS2; XMA 0; STA CRET0  (RESTART THIS USER AT CDMS2)
        SKA X4, STA SBRSRT    (SET UP SBRSRT IF NOT ALREADY SET UP)
        BRM CDMS1

CPX     ZRO;* CONVERT USERS CP TO MONITOR CP.MONITOR CP HAS 4 CHARS/WORD.
        RSH 23; STA CPX2; CLA; DIV THREE
        STB CPX1; LSH 2; ADD CPX1; STA CPX1; BRR CPX
CPX1    0
CPX2    0
```

```
CPY     0;* CONVERT MONITOR CP (4 CHARS PER WORD) TO 3 CHARS PER WORD CP.
        STA CPY1; ETR THREE; XMA CPY1; RSH 2; MUL THREE; LSH 23
        ADD CPY1; LDX BUFF; SKN CSW,2; BRU *+2; MRG X4; STA CPY1; BRR CPY
CPY1    ZRO

CERR5   MIN CERRC
CERR4   MIN CERRC
CERR3   MIN CERRC
CERR2   MIN CERRC
CERR1   MIN CERRC
CERR    CLA; XMA CERRC; STA SS03; BRM NPOPX
CERRC   0 0

CPRIV   0;* PUT PRIVELEGE MASK IN HIS X REGISTOR
        LDX BUFF; LDB* CFLG,2; LSH 39; ETR =377B5
        XMA SS03; ETR =40077777B; ADM SS03; BRR CPRIV

* COMPUTES WORD COUNT, CORE AND DISC ADDRESSES FOR CDTC.
* INPUT: A=INT. ROUTINE NO.  NEG. FOR WRITE.
CRW     0;* FILE READ WRITE ROUTINE.GET DISC ADRS AND CALL LOW LVL DRIVER
        STA CRW9
        LDX BUFF; LDB* CFLG,2; SKA X4; SKB TWO; BRU *+2; BRR CRW
        LDX LVL; LDA BFC+1,2; RSH 5; ADM CRW9
        BRM SETBIP; LDB* CBIP; SKB KM1; BRU *+2; BRM MONCR; LDB CBIP
        LDX LVL; LDA BUFF; ADD BFP+1,2; LDX CRW9; BRM CDTC
        BRR CRW
CRW9    ZRO

CW      0;* FILE WRITE.WONT WRITE A ZERO BFR. GETS AND GIVES BIT MAP BITS
        BRM SETBIP; LDA LVL; ADD BUFF; ADD =CC+1; STA CW8
        LDA KM1; XMA* CW8; SKG* CW8; BRR CW
        LDA* CBIP; SKA THREE; BRU CW3 (NON-DELETABLE DISC BLOCK)
        LDX LVL; LDA BUFF; ADD BFP+2,2; ADD CW9; STA CW1
        LDX BFCW+1,2; CLA
CW1     SKE 0,2; BRU CW2; BRX CW1
        LDB CBIP; SKE* CBIP; BRM CESW; BRR CW
CW2     SKE* CBIP; BRU CW3
        LDX BUFF; SKN CSW1,2; BRR CW
        BRM CG; STA* CBIP; SKR CW8; MIN* CW8
```

```
CW3     LDX BUFF; LDA =44B6; SKA CSW1,2; BRM CRW; BRR CW
CW9     SKE 0,2
CW8     0

CWALL   0;* START BUFFER LEVEL 1, 2, AND 3 WRITING. NO DISMISS NECESSARY.
        LDA =MLVL-1; STA LVL
        BRM CW; SKR LVL; LDA LVL; SKE ZERO; BRU *-4; BRR CWALL

CR      0;* FILE READ.CLEAR BFR IF DISC POINTER=0.
        LDB CR; STB CRET2
        BRM CW; LDA* CBIP; SKA DMSK; BRU CR1
        LDX BUFF; SKN* CFLG,2; BRM CDMS (DONT CLR YET.MAY BE WRITING);
        BRM CCLR; BRR CRET2
CR1     LDA <256; BRM CRW; BRR CRET2

CPOS    0;* DO EVERYTHING NEEDED TO POSITION FILE TO CP SPECIFIED IN A
        STA CT1 (POSITION TO CT1)  THIS ROUTINE MAY DISMISS ITSELF.
        LDX CPOS; STX CRET3; LDX =NLVL-2; STX LVL
CPS1    LDX LVL; LDX CSH1,2; LDB KM1; LSH 0,2
        LDX BUFF; LDA CP,2; SKM CT1; BRM CW; SKR LVL; BRU CPS1
CPS2    LDX BUFF; LDA CP,2; LDX =-NLVL
        LDB CMSK+NLVL,2; SKM CT1; BRU CPS4; BRX *-3
        BRM DTS; BRM SCEOF1; BRR CRET3
CPS4    LDA CT1; EOR CMSK1+NLVL+1,2; COPY AB,XA; LDX BUFF; STB CP,2
        ADD =NLVL; STA LVL; SKG =MLVL-1; BRM CR
        LDX BUFF; SKN* CFLG,2; BRM CDMS; BRU CPS2

CLS     0;*   CLEAN UP FILE I/O SO THAT FILE PARAMETERS CAN BE CLEARED.
        ETR ADMSK; SKG =NBUF+2; SKG TWO; BRM TRAP
        CAX; LDX BUF3,2; LDB CFLG,2; SKB KM1; BRU *+2; BRR CLS
        BRM IOI; LDA =MLVL-1; STA LVL; BRM CW; SKR LVL; BRU *-2
        BRR CLS

CLS5    0;*    CLEAN UP LOCAL AND GLOBAL FILE PARAMETERS
        CAX; LDX BUF3,2; LDB CFLG,2; SKB KM1; BRU *+2; BRR CLS5
        BRM IOI; LDA BUFF; BRM BPUT; BRM MONCR
        LDX GFILE; LDA KM1; STA XBP,2
        BRR CLS5
```

```
$EXBGET EQU *  BRS 144•LEND VERNE A BUFFER FOR AWHILE.                    D          PAGE  21
        BRM BGET; BRM NPOPX (NO FREE BFRS);
        LDA BUFF; SUB =34000B; STA SSO1; BRU COK

$EXBPUT EQU *  BRS 145• FREE THE BFR THAT WAS BORROWED BY BRS 144•
        ADD =34000B; BRM BPUT; BRM MONCR; BRM NPOPX

BGET    0;*   FIND A FREE BUFFER
        LDX =-NBUF; LDA X4
        RSH 1; SKA FBWRD; BRU BGET1; BRX *-3; BRR BGET
BGET1   ADM FBWRD; EAX NBUF+3,2; CXA; ETR ADMSK; STA FILE
        LDX BUF3,2; STX BUFF
        MIN BGET; BRR BGET

BPUT    0;*   FREE A TS BFR• BFR ADRS IN A•
        LDX =-NBUF
        SKE BUFN,2; BRX *-1; SKE BUFN,2; BRR BPUT
        LDA X4; LRSH NBUF+1,2; SKA FBWRD; BRR BPUT
        ADM FBWRD; LDX BUFN,2; CLA; STA CFLG,2
        MIN BPUT; BRR BPUT

BSET    0;* CHANGE TS BLOCK VIRTUAL ADRS TO 16 BIT ABSOLUTE ADRS.
        RSH 11; LDA RLTS; LSH 11; STA T; BRR BSET

CEOFP   0;*  SET CEOF POINTER• CPTOP FOR INPUT, CEOF FOR OUTPUT
        LDX BUFF  POINT TO CEOF UNLESS (INPUT)AND(CPTOP LESS THAN CEOF).
CEOFP2  LDA CEOF,2; LDB CEOFP3; SKN CSW,2
CEOFP3  SKG CPTOP,2; LDB CEOFP2; STB CEOFP1; LDA* CEOFP1; BRR CEOFP
CEOFP1  0

SETCC   0;* SET CURSOR CHANGED FLAGS TO INHIBIT WRITES
        LDA BUFF; ADD SETCC1; LDX =-NLVL
        STA *+2; LDA KM1; STA BUFF+NLVL,2; BRX *-1; LDX BUFF; BRR SETCC
SETCC1  STA CC+NLVL,2

SCEOF1  0;*  SET CFOF1 SO THAT CIO,WIO LOGIC WILL KNOW WHEN TO CALL C256.
        BRM CEOFP; LDA CP,2; MRG CMSK+MLVL
        SKG* CEOFP1; BRU *+2; LDA* CEOFP1; STA CEOF1,2; BRR SCEOF1
```

```
C1      0;*  SET DATA BL0CK (BFR LEVEL=MLVL) CHANGED FLAG IF 0UTPUT M0DE.
        LDX BUFF; SKN CSW,2; BRR C1; MIN CC+MLVL,2; BRR C1

SETBIP  0;* P0INT T0 DISC ADRS THAT IS IN PARENT BFR 0F BUFFER T0 BE
*       READ 0R WRITTEN
        LDX BUFF; LDA CP,2; LDX LVL; ETR CMSK,2; EXU CSH,2
        ADD BFP,2; ADD BUFF; STA CBIP; BRR SETBIP

CCLR    0;* CLEAR THE BFR SPECIFIED BY LVL.
        LDX LVL; LDA BFP+1,2; ADD BFC+1,2; ADD BUFF; ADD =235B5; STA CCLR1
        LDX BFCN+1,2; CLA
CCLR1   STA 0,2; BRX CCLR1; BRR CCLR

CCLRX   0;*   CLEAR ALL THE BFRS F0R A GIVEN FILE.
        CLA
CX1     STA LVL; BRM CCLR; LDA LVL, ADD 0NE
        SKG =MLVL-1; BRU Cx1; BRR CCLRX


* SETS UP REGISTERS F0R DTC
* INPUT: A=C0RE ADDRESS, B=P0INTER T0 DISC ADDRESS
*          X=INT. N0. AND W0RD C0UNT
CDTC    0;* CURS0R DTC. F0R FILES 0NLY
        STA CDTCA; STB CDTCB; STX CDTCX
        SKN CDTCX; BRU CDTC1;* CHECK F0R SMALL W0RD C0UNT IF 0UTPUT
        SKA SDBM8; BRR CDTC
        XXA; SKA =176B; BRU CDTC1 (W0RD C0UNT GREATER THAN 40B);
        LDA* CDTCB; DIR; BRM CKSUM; EIR
*       PUT CHECKSUM INT0 DISC ADDRESS AND MARK PARENT BUFFER CHANGED
        XMA* CDTCB; CAB; ETR DMSK; ADM* CDTCB
        LDA LVL; ADD BUFF; C0PY AX,BA; SKE* CDTCB; MIN CC,2
CDTC1   LDX BUFF; LDA TW0; SKA CSW,2; SKA* CFLG,2; BRU *+2; BRR CDTC
        LDB GFILE; LSH 31; SKG ZER0; BRM M0NCR; ADM CDTCX
        LDA CDTCA; SKA X4; BRU *+3; BRM BSET; BRU *+2; BRM DTH
        LDB* CDTCB; LDX CDTCX; BRM DTC
        LDX BUFF; LDA K4B5; ADM* CFLG,2; BRR CDTC
CDTCA   0
```

```
CDTCB   0
CDTCX   0
CDTC9   EØR 0,2

PE      0;* MØVE TØ CURSØR PØSITIØN IN A. BACK UP CPTØP IF NECESSARY.
        STA CRET4; BRM CWALL; LDA PE; xMA CRET4
        LDX BUFF; SKE CPTØP,2; BRU PE6
        LDA CP,2; xMA CPTØP,2; MIN CC+1,2  SPECIAL CASE. DECREASE CPTØP.
PE6     STA CP2    CP2 IS THE FINAL GØAL ØF THE ERASE
        LDA CP,2   START AT THE CURRENT PØSITIØN
PE7     STA CP1  CP1 WØRKS ITS WAY UP TØ CP2.
PE1     BRM CWALL; LDA CP1; BRM CPØS
        LDX =MLVL-1; STX LVL; BRM SETBIP
        LDA CP2; SKG CP1; BRU PE9  (ERASE CØMPLETED);
        LDA CP1; SKA CMSK+MLVL; BRU PE3 (CLEAR A FRACTIØN ØF DATA);
        EØR CP2; SKG CMSK+MLVL; BRU PE4 (NØ MØRE FULL DATA BLØCKS);
        LDA CP1; ETR CMSK+MLVL-1; EXU CSH+MLVL-1
        EØR KM1 (CMA, SUB =1)  CØMPUTE HØW MANY FULL DATA BLØCKS....
        ADD BFC+MLVL-1; STA CT2    .... CAN BE DELETED WITHØUT I/Ø.
        LDA CP2; SUB CP1; EXU CSH+MLVL-1; SUB ØNE; SKG CT2; STA CT2
        SKR CBIP
PE2     MIN CBIP; LDB BFC+MLVL; LSH 26; ADM CP1  INCR. CP1 1 DATA BLØCK.
        LDB CBIP, CLA; SKE* CBIP; BRM CESW   DELETE A DATA BLØCK
        SKR CT2; BRU PE2; BRU PE1
PE3     MRG CMSK+MLVL; ADD ØNE; SKG CP2; BRU PE5 (CLR BEG. ØF DATA BLØCK);
PE4     LDA CP2    (CLR END ØF DATA BLØCK)
PE5     STA CP1; BRM CLRPRT; BRU PE7
PE9     LDX BUFF; LDA CP,2; SKE CPTØP,2; SKG CPTØP,2; BRR CRET4
        LDA CP,2; MRG CMSK+MLVL; ADD ØNE; BRM CLRPRT CLEAR GARBAGE PART ØF
*       LAST DATA BLØCK SØ IT WILL BE RELEASED IF USED PART WAS ALL ZERØ.
        LDX BUFF; SKN CSW1,2; BRU PE9A
        LDA CPTØP,2; STA CP2; BRU PE7;* BACKUP TØ THE NEW CPTØP.
PE9A    BRM CWALL; BRR CRET4;* DØNT BØTHER TØ REPØSITIØN .

CLRPRT  0;* CLEAR SØME ØF THE CHARACTERS IN A DATA BLØCK.
        STA CLP9   CLEAR UP TØ CLP9-1. START AT CURRENT CP.
        LDX BUFF; LDA CP,2; STA CLP8   CLP8 WØRKS ITS WAY UP TØ CLP9
CLP1    LDA CLP9; SKG CLP8; BRR CLRPRT
        LDX BUFF; MIN CC+MLVL,2; LDA CLP8; ETR CMSK+MLVL; RSH 2
```

```
        ADD BUFF; ADD BFP+MLVL; STA CLP7
        CLA; LSH 2; CAx; SKA THREE; BRU CLP3
        LDA CLP9; SUB CLP8; SKA KM4; BRU CLP2
CLP3    LDA CLP6,2; ADM CLP8; LDA CLP5,2
        ETR* CLP7; STA* CLP7; BRU CLP1
CLP2    LDA CLP9; SUB CLP8; ETR KM4; ADM CLP8
        MUL x7 (RSH 2,CNA); CBPY AX,N,B; ADD CLP7; ADD CLP4
        STA *+1; STB 0,2; BRx *-1; BRU CLP1
CLP9    0
CLP8    0
CLP7    0
CLP6    DATA    1,1,2
CLP4    STB 0,2
CLP5    DATA    177777B,77600377B,77777400B

CDEL1   0;*  SET UP TO DELETE OR MAP A FILE
        LDX SSG1; LDX BUF3,2; LDA* CFLG,2; SKA =4B4; BRM NPOPX (I/O ERR);
        STB CSW1,2; LDA SSG1; BRM IOI
        LDA* CFLG,2; MRG K377; STA* CFLG,2; LDA KM1; STA CSW,2
        LDA CPTOP,2; BRR CDEL1

CDEL3   0;*    PUT A BIT INTO THE BIT MAP. FREE A BLOCK ON THE DISC.
        STA GB8   HANDY DEBUGGING INFO FOR CE CRASHES
        SKN SDBM8; BRM LOCBIT; BRR CDEL3 (ERR); BRU CDEL3B  (OK,UNMAPPED)
        SKA* LB; BRU CDEL3A  (ERR, BIT ALREADY MAPPED);
        ADM* LB; ADM BITSUM; MIN DBITS
        MIN CDEL3
CDEL3A  BRM MPDSC; BRR CDEL3
CDEL3B  MIN CDEL3; BRR CDEL3

CGRB1   0;*   GRAB A PARTICULAR BIT FROM THE BIT MAP.
        BRM LOCBIT; BRR CGRB1 (ERR); BRU CGRB1B (LEGAL ADRS NOT MAPPED)
        SKA* LB; BRU *+2; BRU CGRB1A (ERR, BIT ALREADY GRABBED);
        CNA; ADM* LB; ADM BITSUM; SKR DBITS
        MIN CGRB1
CGRB1A  BRM MPDSC; BRR CGRB1
CGRB1B  MIN CGRB1; BRR CGRB1

CG      0;*     GRAB A DISC BLOCK FOR A FILE
```

```
        LDA BUFF; ADD LVL; CAX; MIN CC,2; BRM GETBLK
        LDX BUFF; MIN CSIZE,2; MIN CC+1,2; SKR CQ,2; NOP 0
        BRR CG   EXIT WITH DISC ADRS IN A AND BUFF IN X

* INPUT: B=BLOCK LOCATION INDIRECT
CE      0;*     ERASE A BLOCK
        STB CE9; SKB SDBM8; BRM MONCR
        LDA* CE9; SKA DMSK; SKA THREE; BRR CE; CLA; XMA* CE9
        ETR DMSK; BRM CDEL3; BRM TRAP
        LDA BUFF; ADD LVL; CAX; MIN CC,2
        LDX BUFF; MIN CQ,2; MIN CC+1,2; SKR CSIZE,2; BRR CE
*       BAD FILE WITH GOOD CHECKSUMS. CAUSE IT TO BE DESTROYED
        MIN CCKWRD,2    GARBAGE UP THE CHECKWORD SO FILE IS INACCESSABLE
        BRM SETCC; MIN CC+1,2   (WRITE TOP LEVEL XBLOCK ONLY); BRM TRAP
CE9     0;* POINTS TO DISC ADDRESS BEING DELETED

CESW    0;* CALL CE NORMALLY. CALL CE1 IF MAPPING A FILE.
        LDX BUFF; LDX CSW1,2; EXU CESW1+1,2; BRR CESW
CESW1   BRM CE (CSW1 =-1,NORMAL); BRM CE1 (BIT MAPPING); BRM CE (BRS 66)

CE1     0;* MAP A BIT
        SKN SDBM8; BRM MONCR
        STB CE9; LDA* CE9; SKA DMSK; SKA THREE; BRR CE1
        CLA; XMA* CE9; ETR DMSK
        BRM CQRB1; BRM MPOPX
        LDA BUFF; ADD LVL; CAX; MIN CC,2; BRR CE1

*               ********************* END OF LOW LEVEL ROUTINES

        FORGET
$ENDD   EQU *
        END
```

```
      LISTM
* ENTRY POINTS
      ENTRY   CTLI,TBSFI,TE0FI,TFSFI,PRDI,TRINT,TWI
      ENTRY   CTRL,ETTW,TREAD,TWRT,PRD

      TDT     TRT,SKS,10410   SKIP IF TAPE NOT READY
      TDT     BTT,SKS,12010   SKIP IF NOT BEGINNING OF TAPE
      TDT     ETT,SKS,11010   SKIP IF NOT END OF TAPE
      TDT     TFT,SKS,13610   SKIP IF NOT END OF FILE
      TDT     FPT,SKS,14010   SKIP IF TAPE NOT FILE PROTECTED
      TDT     D8T,SKS,17210   SKIP IF NOT 800 DENSITY
      TDT     D2T,SKS,16210   SKIP IF NOT 200 DENSITY
      TDT     D5T,SKS,16610   SKIP IF NOT 556 DENSITY
      TDT     RTB,E0M*,3610   READ TAPE BINARY
      TDT     WTB,E0M*,3650   WRITE TAPE BINARY
      TDT     WFM,E0M*,2050   WRITE FILE MARK
      TDT     SFB,E0M*,3630    SCAN FORWARD IN BINARY
      TDT     SRB,E0M*,7630    SCAN REVERSE IN BINARY
      TDT     ETF,E0M*,3670   ERASE TAPE FORWARD
      TDT     ETR,E0M*,7670   ERASE TAPE IN REVERSE
      TDT     REW,E0M,14010   REWIND

SETXX MACR0; LDA XX; STA TREAL; ENDM
SETRL MACR0; LDA REAL; ADD =180; STA TREAL; ENDM

* NEW TAPE LOGIC

* UNIT ASSIGNMENT
*   1=TRINT, 2=TWI, 3=CTLI, 4=TE0FI, 5=TFSFI, 6=TBSFI, 7=PRDI
* BRS 102   READ TAPE
* INPUT: A=CORE ADDR., B=WORD COUNT, X=NO. OF RECORDS (EQ OR LT 64)
* OUTPUT: 1 WORD AT BEGINNING OF EACH RECORD.
* OP CODE FLAGS:
*       00=TAPE RECORD WAS READ. IT HAD N WORDS WHICH FOLLOW.
*       01=TAPE RECORD WAS READ. IT HAD N WORDS WHICH FOLLOW, BUT
*          AN ERROR WAS MADE. NO FURTHER RECORDS WERE READ.
```

```
*       02=END OF FILE. NO MORE READING.
*       03=LAST TAPE RECORD DUE TO LACK OF CORE SPACE.
*          NO MORE RECORDS. RECORD MAY HAVE BEEN TRUNCATED.
*       04=TAPE RECORD WAS TRUNCATED DUE TO OVERRUN OF 3 SEC.
*          TIME LIMIT. (PERHAPS RUNAWAY)
*       05=NO MORE RECORDS DUE TO REACHING SPECIFIED RECORD COUNT.
*       06=END OF TAPE. NO MORE READING.
*       07=TAPE NOT READY. NO MORE READING.
*       10=PAGE BOUNDARY ERROR ON WRITE.
*       11=TAPE WRITE PROTECTED.
*       12=BEGINNING OF TAPE. NO WRITE.
*       13=DEVICE NO LONGER ASSIGNED.
*       14=RUBOUT
TREAD   SKN TNUM; BRM TRAP
        SKN WBIU; BRM TRAP
        SKN UNIT; BRU BLKBSY
        LDA UTTY; SKE DEVCH; BRM TRAP; LDX DEVICE
        CXA; SKG ONE; SKG KM1; BRM TRAP
        STX TNUM; LDA SS03; SUB ONE; STA TNR; SKN TNR; BRU *+2
        BRM TTRAP; SKG K100; BRU *+2; BRM TTRAP
        STB TWCNT; SKR TWCNT; BRU *+2; BRM TTRAP; LDA SS01
        ETR ADMSK; STA TCA; ETR PGMSK; ADD TWCNT; SKA =77774B3;
        BRM TTRAP; LDX TCA; BRM GRBMW; CXA; BRM DTH
        STA TCA
        LDX TNUM; LDB RTBW,2
        BRM TPSET; BRU TRD4; LDA TRA; STA TMCA
        CLA; LDX SS01; STA 0,6; LDA ONE; STA UNIT
TRD5    LDA =-285; LDX JOB; ADM TTNO,2; STX WJOB
        LDA TCA; BRM WB
        LDA TNUM; CAX; ADD TPDMS2; MIN TPDMS,2
        CAB; LDX =GTI; BRM TFIN;  MIN 0; BRU NPOPDS
TRD4    BRU TWR4+2
TPDMS2  33B TPDMS

* INPUT: A=TCA, B=RTB OR WTB
* SETS TRA=(TCA) IN P7.
* SETS UP COMMANDS AT (TCA)+1
TPSET   ZRO; XAB; ETR TPAR; XAB
        STB TRD2; RCY 9; ETR =140B; MRG =216000B; STA IORD; LDA TWCNT
```

```
        SKG ZER0; BRU TPSET2
        LCY 14; ETR =7774B4; STA TD2; LDA 0NE; SKB 0NE; ADM I0RD
        LDA TCA; ETR ADMSK; ADD 0NE; ADM TD2; SUB 0NE
        ETR <3777; ADD =34000B
        STA TRA; BRM TRD; BRR TPSET; MIN WBIU
        MIN TPSET; BRR TPSET
TPSET2  LDA =3B5; BRR TPSET

* TAPE READ INTERRUPT
TRINT   LDX TNUM; ASCW; PIN TMAR
        SETXX
        LDA TMAR; CAB; SUB TCA; STB TCA; ETR ADMSK
        SUB 0NE
        SKN DEVICE; BRU *+2; BRU TRI9 (LOST DEVICE)
        SKN DEVCH; BRU *+2; BRU TRI10 (RUBOUT)
        EXU TFTW,2; BRU TRI2 (E0F)
        BETW; BRU TRI1 (ERR0R)
        IF DISC<3; SKN DSW; BRM IDMW; ENDF
        SKE TWCNT; BRU TRI8; MRG =3B5; BRU TRI6
TRI8    SKR TNR; BRU TRI5; MRG =5B5; BRU TRI6 (RECORD COUNT ZER0)
TRI5    EXU ETTW,2; BRU TRI3 (END 0F TAPE)
        STA* TRA; ADD 0NE; CNA; ADM TWCNT; CBA; LDB RTBW,2
        BRM TPSET; BRU TRI6; LDA TRA; STA TMCA; CLA; STA* TMCA
        LDA 0NE; STA UNIT; SETRL; BRU WRET
TRI6    STA* TRA
        LDX TNUM; SKR TP0MG,2; N0P; LDA KM1; STA TNUM
        STA WBIU
        BRU NINT2
TRI1    MRG K1B5 (ERR0R); BRU TRI6
TRI2    MRG K2B5 (E0F); BRU TRI6
TRI3    MRG =6B5 (END 0F TAPE); BRU TRI6
TRI9    MRG =13B5 (L0ST DEVICE); BRU TRI6
TRI10   MRG =14B5 (RUB0UT); BRU TRI6

* BRS 103   WRITE TAPE.
* INPUT: A=C0RE ADDRESS.
*   (A) C0MMUNICATI0N FR0M SYSTEM T0 USER.
*   (A)+1 1ST REC0RD W0RD C0UNT.
*   (A)+2 THRU (A)+N+1 REC0RD DATA.
```

```
*    (A)+N+2 NEXT RECORD WORD COUNT.
*    ETC. COUNT=0 AFTER LAST RECORD.
* OUTPUT: A=ADDRESS OF LAST RECORD.

TWRT    SKN TNUM; BRM TRAP
        SKN WBIU; BRM TRAP
        SKN UNIT; BRU BLKBSY
        LDA UTTY; SKE DEVCH; BRM TRAP; LDX DEVICE
        CXA; SKG BNE; SKG KM1; BRM TRAP
        STX TNUM; LDA SS01; ETR ADMSK
        CAX; BRM GRBMW
        ETR =34000B; STA TURL; CXA; ETR ADMSK; ADD BNE; STA 0,6; CXA
        ETR K3777; ADD =34000B; STA TMCA; EAX 1,2
        ADD TW0; ADD 0,6; SKA =7774B4; BRM TTRAP; CXA; ETR ADMSK; STA TCA
        LDA 0,6; STA TWCNT; LDA TCA
        BRM DTH; STA TCA
        LDX TNUM; LDA =11B5; EXU FPTW,2; BRU TWR4
        LDA =12B5; EXU BTTW,2; BRU TWR4
        LDA TCA; LDB WTBW,2
        BRM TPSET; BRU TWR4; LDA TW0; STA UNIT
        BRU TRD5
TWR4    ADD SS01; ADD BNE; LDX SS01; STA 0,6
TWR5    LDA KM1; STA TNUM; BRM NP0PX
* UNIT  BUSY
BLKBSY  LDB TPDMSB; LDX =QTI; BRU NP0PDS

* TAPE WRITE INTERRUPT.
TWI     BSS 0
        SETXX
        SKN DEVICE; BRU *+2; BRU TWI7 (LOST DEVICE)
        SKN DEVCH; BRU *+2; BRU TWI10 (RUBOUT)
        BETW; BRU TWI1 (ERROR)
        IF DISC<3; SKN DSW; BRM IDMW; ENDF; LDX TNUM
        EXU ETTW,2; BRU TWI3 (END OF TAPE)
        CLA; STA TPE,2
        LDA* TRA; ETR ADMSK; ADD BNE; ADM TCA; ADM TRA
        LDA* TRA; SKE ZERO; BRU *+2; BRU TWI6; STA TWCNT; LDA TCA
        ETR PGMSK; ADD TWCNT; SKA =77774B3; BRU TWI2
        LDA TCA; LDB WTBW,2; BRM TPSET; BRU TWI6
```

```
        LDA TCA; ETR K3777; ADD TURL; STA* TMCA
        LDA TW0; STA UNIT; SETRL; BRU WRET
TWI1    LDA <165 (ERROR)
TWI6    ADM* TMCA; LDX TNUM; SKR TPDMS,2; NOP
        LDA <M1; STA TNUM; STA WBIU; BRU NINT2
TWI3    LDA =685 (END OF TAPE); BRU TWI6
TWI2    LDA =1085 (PAGE OVERRUN); BRU TWI6
TWI7    LDA =1385 (LOST DEVICE); BRU TWI6
TWI10   LDA =1485 (RUBOUT); BRU TWI6
*
* BRS 105   NEW TAPE CONTROLS
* INPUT: A=1=NOP, 2=BACKSPACE RECORD, 3=FORWARD SPACE FILE
*          4=BACKSPACE FILE, 5=WRITE 3 INCHES BLANK TAPE
*          6=REWIND, 7=WRITE END OF FILE
CTRL    SKN DEVICE; BRU *+2; BRM TRAP
        LDA UTTY; SKE DEVCH; BRM TRAP; LDX DEVICE
        SKN WBIU; BRM TRAP; CXA; SKG ONE; SKG KM1; BRM TRAP
        SKN UNIT; BRU BLKBSY; LDA =CTMCA; STA TMCA
        STX TNUM; MIN WBIU
        EXU TRTN,2; BRU CTL10 (READY)
        BRU TCL2
CTL10   LDA PACPTR; STA WPTR
        LDA SS01; SKG SEVEN; SKG ZERO; BRM TTRAP
        CAX; BRU* *,2
CRL6    DATA TREW3,TBSR,TFSF,TBSF,TERS,TREW,TEOF

* REWIND
TREW    LDX TNUM
TREW2   EXU REWW,2
TREW3   BRM CTLX3

* BACKSPACE RECORD
TBSR    LDX TNUM
        EXU BTTW,2; BRM CTLX3
        LDA SRBW,2; ETR TPAR; STA *+1
        EXU SRBW,2; EOM 16000B; POT X0
*  CONTROL EXIT
CTLX    LDA THREE
CTLX2    STA UNIT; CXA; ADD TPDMS2; MIN TPDMS,2; CAB
```

```
        LDA =-2B5; LDX J0B; STX WJ0B; ADM TTN0,2
        LDA X4; STA WPAGE
        BRM TFIN; LDX =QTI; MIN 0; BRU NP0PDS
CTLX3   ZR0; LDX KM1; STX TNUM; STX WBIU; BRM NP0PX
*  CONTROL INTERRUPT ROUTINE.
$CTLI   BSS 0
CTLI2   LDX TNUM; SKR TPDMS,2; NOP; LDA KM1
        STA TNUM; STA WBIU; LDA XX; STA TREAL
        IF DISC<3; SKN DSW; BRM IDMW; ENDF
        LDX WJ0B; LDA K2B5; ADM TTN0,2; BRU WRET

* ERASE 3 INCH GAP
TERS    LDX TNUM
        LDA =11B5; EXU FPTW,2; BRU TERS3 (FILE PROTECTED)
        LDA =6B5; EXU ETTW,2; BRU TERS3 (END OF TAPE)
        EXU ETFW,2; EOM 16000B; POT X4; BRU CTLX
TERS3   STA SS01; BRM CTLX3

* WRITE END OF FILE
TEOF    LDX TNUM
        LDA =11B5; EXU FPTW,2; BRU TERS3 (FILE PROTECTED)
        EXU ETTW,2; BRU TE0F3 (END OF TAPE)
        CLA; STA TPE,2
TE0F4   EXU ETFW,2; EOM 16000B; POT X4; LDA FOUR; BRU CTLX2
TE0F3   MIN TPE,2; LDA TPE,2; SKG FOUR; BRU TE0F4; BRM TTRAP
TE0FI   LDX TNUM; EXU TRTW,2; BRU *+2; BRU *-2
        EXU WFMW,2; EOM 16000B; POT FMK
        LDA THREE; STA UNIT; BRU WRET

* FORWARD SPACE FILE
TFSF    LDX TNUM; LDA =6B5; EXU ETTW,2; BRU TFSF2 (EOT)
        LDA SFBW,2; ETR TPAR; STA TFSFI3; STA *+1
        EXU SFBW,2; EOM 16000B; POT XO; LDA FIVE; BRU CTLX2
TFSF2   STA SS01; BRM CTLX3
TFSFI   SKN DEVICE; BRU *+2; BRU CTLI2 (LOST DEVICE)
        SKN DEVCH; BRU *+2; BRU CTLI2 (RUBOUT)
        LDX TNUM; EXU TRTW,2; BRU *+2; BRU CTLI2
        EXU TFTW,2; BRU CTLI2 (EOF)
        EXU ETTW,2; BRU CTLI2 (EOT)
```

```
        SETXX
        IF DISC<3; SKN DSW; BRM IDMW; ENDF
        LDA FIVE; STA UNIT
TFSFI3  EXU SFBW,2; EOM 16000B; POT XO
        SETRL; BRU WRET

* BACKSPACE FILE
TBSF    LDX TNUM; EXU BTTW,2; BRM CTLX3 (EOT)
        LDA SRBW,2; ETR TPAR; STA TBSFI3; STA *+1
        EXU SRBW,2; EOM 16000B; POT XO; LDA SIX; BRU CTLX2
TBSFI   SKN DEVICE; BRU *+2; BRU CTLI2 (LOST DEVICE)
        SKN DEVCH; BRU *+2; BRU CTLI2 (RUBOUT)
        LDX TNUM; EXU TRTW,2; BRU *+2; BRU CTLI2
        EXU TFTW,2; BRU CTLI2 (EOF)
        EXU BTTW,2; BRU CTLI2 (BOT)
        SETXX
        IF DISC<3; SKN DSW; BRM IDMW; ENDF
TBSFI3  EXU SRBW,2; EOM 16000B; POT XO; LDA SIX; STA UNIT
        SETRL; BRU WRET

* BRS 108   TEST DENSITY
* OUTPUT: A=0 FOR 200, A=1 FOR 556, A=2 FOR 800.
$TSTD2  LDA DEVICE; SKG ONE; SKG KM1; BRM TRAP; CAX; CLA
        EXU D2TW,2; BRU TSTD3; LDA ONE; EXU D5TW,2; BRU TSTD3
        LDA TWO; EXU D8TW,2; BRU TSTD3; BRM TRAP
TSTD3   LDX SSD3; BRU NXPOP

TTRAP   ZRO; LDA KM1; STA TNUM; STA WBIU; BRM TRAP
*
* BRS 110   TEST TAPE READY
$TSTRDY LDA DEVICE; ETR ONE; CAX; EXU TRTW,2; MIN 0 (READY)
        BRM NPOPX (NOT READY)
*
*
* BRS 106   PRINTER DRIVER
* INPUT: A=CORE ADDRESS, X=WORD COUNT
*       (A) COMMUNICATION WORD FROM SYSTEM TO USER.
*       (A+1) PAPER CONTROL,  NEG FOR SKIP, POS FOR UPSPACE.
*       (A+2)-(A+34) ONE LINE OF DATA.
```

```
PRD     BSS 0
        LDA UTTY
        SKN WBIU; BRM TRAP; SKE DEVCH; BRM TRAP; LDA TW0
        SKE DEVICE; BRM TRAP; SKN UNIT; BRU BLKBSY
        LDA SS01; ETR ADMSK; CAX; BRM GRBMW; ETR =34000B
        STA TURL; EAX 1,2; LDA SS01; ETR K3777; ADD =34000B
        STA TMCA; COPY XA,B; ETR ADMSK; STA TCA
        ETR PGMSK; ADD SS03; ETR NPGMSK; SKE ZERO; BRM TRAP
        LDA TCA; BRM DTH; STA TCA
        LDA TCA; ETR K3777
        ADD =34000B; STA TRA; LDA SS03; SUB TW0; STA PRWC
        LDA SS03; SUB ONE; RSH 23; DIV =34; SKE KM1; BRM TRAP
        SKS 14000B (CAT); BRM MONCR
        LDA =705; SKS 12060B (PRT); BRU PRD2 (NOT READY)
        LDA TCA; BRM WB; LDA WPAGE; MRG RLWB; XMA RRL3
        BRM CHR3; BRM PRSET; BRM TFIN
        BRM MPWB; CLA; STA WBIU
        LDX J0B; LDA =-205; ADM TTN0,2
        MIN PRDMS1; LDB PRDMS; LDX =GTI; MIN 0; BRU NP0PDS
PRDMS   33B PRDMS1
PRD2    LDX SS01; STA 0,6; BRM NP0PX
PRSET   ZRO; LDX TRA; LDA 0,2; ETR SEVEN; CLB; LSH 9; MRG PRSK
        SKN 0,2; MRG K200; STA P0SL
P0SL    EOM 10460B; LDA TCA; RCY 9; ETR =140B; MRG =216000B
        STA PRD4; MIN WBIU; LDA TCA; ETR ADMSK; ADD ONE
        MRG =33*4B4; STA PRD6; EOM* 2660B
PRD4    EOM 16000B; POT PRD6; LDA SEVEN; STA UNIT
        BRR PRSET
PRSK    EOM 10460B
* PRINTER INTERRUPT ROUTINE
PRDI    LDA =34; ADM TCA; ADM TRA; COPY N,B; ADM PRWC
        SKN PRWC; BRU PRDI2
PRDI3   LDA KM1; STA WBIU
        SKR PRDMS1; NOP; LDA XX; STA TREAL; BRU NINT2
PRDI2   LDA =1385; SKN DEVICE; BRU *+2; BRU PRDI4
        LDA =1485; SKN DEVCH; BRU *+2; BRU PRDI4 (RUBOUT)
        LDA =705; SKS 12060B; BRU PRDI4 (NOT READY)
```

```
        SETXX
        IF DISC<3; SKN DSW; BRM IDMW; ENDF
        BRM PRSET; SETRL; BRU WRET
PRDI4   STA* TMCA; BRU PRDI3
PRWC    ZRB 0

*    BRS 97, 98, 99
* THESE BRS'S RESET, INCREMENT AND READ ONE OF 32
* COUNTERS SPECIFIED BY A
SBCT    ZRB; LDA SS01; CAX; SKA =-32; BRM TRAP; BRR SBCT
* BRS 97
$WRSB   BRM SBCT; STB CTTAB,2; BRM NPBPX
* BRS 98
$WINB   BRM SBCT; CBA; ADM CTTAB,2; BRM NPBPX
* BRS 99
$WRDB   BRM SBCT; LDA CTTAB,2; STA SS01; BRM NPBPX
CTTAB   BSS 32
*
$ENDW    LDA =44445555B
        FORGET
        END
```

```
NET     IDENT                                                   6/21/72  15:21     PAGE   1

        LISTM


TLOG    EQU 1   LOG WORDS SENT TO TBO AND WORDS RECEIVED FROM TBI

$TGC    0
TGC1    CLA; SKE TC,2; BRU *+2; BRU TGC3
        LDA THIRD; DIR
        MUL TP,2; AXC; LDX 0,2
        LCY 2; COPY AX,XA,XB; EXU TGC5,2
        ETR K377; LDX TPORT
        SKG SEVEN; BRU TGCE
        MIN TGC
TGC3    EIR; BRR TGC
TGC5    LCY 8; RCY 8; NOP 0; BRM MONCR

TGCE    ADD =2300003B+*; STA *+1; EXU *
        BRU TGCEO; BRM MONCR; BRM MONCR; BRM MONCR
        BRM MONCR; BRM MONCR; BRM MONCR; BRU TGCE7
TGCEO LDA TINC3X; STA TGCINC SET UP POSSIBLE TP,TC UPDATE
        LDA KM1; XMA TGC; ADD ONE; STA TINC1
        LDA ONE; ADD TP,2; MUL THIRD; AXC; LDX 0,2
        LCY 2; COPY AX,XA,XB; EXU TGC5,2; ETR K377
        LDX TPORT; EIR; BRR TINC1

TGCE7   EQU *   ENTER PROGRAM ECHO MODE
        MIN TP,2; RST =7IY,TF1
        SKR TC,2, BRU TGC1; BRM MONCR

$TGCINC 0 TINC2 POINT AT TP,TC UPDATE SUBROUTINE
TINC1 0;* USED BY TGCEO
TINC2 0;* REMOVE 1 CHAR FROM BUFFER
 DIR; SKR TC,2; BRU *+3; MIN TC,2; BRU *+2; MIN TP,2; EIR
TINC2X BRR TINC2
TINC3 0;* RESET TGCINC TO POINT AT TINC2. MAYBE REMOVE TWO CHARS
* FROM BUFFER
 STA TGCINC; LDA TINC2X; XMA TGCINC; BRM TINC2; SKN TGC
```

```
$TWC    0
        STA TWA; STX TWX; MIN TRATE,2
TWC1    LDA TP+1,2; SUB TC,2; SKG TP,2; BRU TG1
        LDA TC,2; SKG K377; BRU TWC2
        ABC; STA TT,2; LDA =-1-RCY; ETR TF2,2; STA TF2,2
        CBA; SKG =700B; BRU TWC2
        LDA TF1,2; SKA =XY (IS TERMINAL IN X-OFF MODE )
        SKA =X0FFY (HAS AN X-OFF BEEN SENT )
        BRU TWC3 (DON'T SEND X-OFF)
        MRG =X0FFY  (SEND X-OFF AND SET SAID FLAG)
        STA TF1,2; COPY XA,B; LCY 8; MRG =200223B (X0FF); BRM TR0
TWC3    LDA TC,2; LDB TF1,2; SKA =-1000B; SKB =IIHY; BRU TWC2
        SKB =AUXY+8IY; BRU *+5; LDB TF2,2; SKB =B113Y+B161Y; BRU *+2
        BRU TWC2 (HIS PROGRAM CAN BE INTERRUPTED FROM THE TTY WITH AN ESCAPE CHR);
        LDA =IIHY; ADM TF1,2; COPY XA,B; LCY 8; MRG =7*2B5; BRM TR0; LDA TC,2
TWC2    ADD TP,2; MUL THIRD; XMA TWA; ETR K377; AXC
        LCY 2; COPY AX,XA,XB
        EXU TWC5,2
        XMA* TWA; ETR TWMASK,2
        ADM* TWA
        LDX TWX; MIN TC,2; BRR TWC
TWA     0;* INPUT CHR. CHANGED TO BFR POINTER.
$TWX    0;* PORT
TWC5    RCY 8; LCY 8; NOP 0; BRM MONCR
TWMASK  DATA 177777B,77600377B,-400B

        IF -1
TGCK    0;* REMOVE WHEN NOT SHOWING BUGS
        CLA; LDX =-NPORT
TGCK0   LDB KM1
TGCK1   SKB TC+NPORT,2; BRU TGCK3
TGCK2   BRX TGCK1; BRR TGCK
TGCK3   STA TGSUM; LDA TP+NPORT,2; MUL THIRD; XMA TGSUM; ADD* TGSUM
        BRX TGCK0; BRR TGCK
TGSUM   0
TGSUM1  0
        ENDF -1
```

```
TG1     EQU *   GARBAGE COLLECTOR
        MIN TG1C
TG1A    LDA TP,2; SUB TP-1,2; SUB TC-1,2; SKG K17; BRU TG3
        LDA TP,2; MUL THIRD; STA TGBRX1
        LDA TP-1,2, ADD TC-1,2; ADD EIGHT; MUL THIRD; STA TGBRX1+1
        SUB TGBRX1; MUL THREE; LSH 23
        ADM TP,2; COPY AB,XA; EAX 1,2; EOR TWX; XAB; SKB ADMSK; BRU *-6
        LDA TP,2; ADD FIVE; MUL THIRD; SUB TGBRX1
        CNA; COPY AX,N
        ADD =27605; ADM TGBRX1; ADD =23505-27605; ADM TGBRX1+1
        STX TBRX1X
TGBRX1  LDA 0,2; STA 0,2; BRX TGBRX1
TG2     EQU *   GARBAGE COLLECTION FINISHED. TRY TO STORE CHARACTER.
        LDX TWX; BRU TWC1
TG3     CXB; EAX -1,2; SKB ADMSK; BRU TG1A
        EAX NPORT
        MIN TG5C
TG5     EAX -1,2; LDA TP+1,2; SUB TP,2; SUB TC,2; SKG K17; BRU TG6
        LDA TP,2; ADD TC,2; MUL THIRD; STA TGBRX2
        LDA TP+1,2; MUL THIRD; SUB THREE; STA TGBRX2+1
        SUB TGBRX2; MUL THREE; LSH 23
        ADM TP,2; EAX -1,2; COPY AB,XA; EOR TWX; XAB; SKB ADMSK; BRU *-6
        LDA TP,2; ADD TC,2; MUL THIRD
        CNA; ADD TGBRX2; ADD ONE; COPY AX,N
        ADD =27605; ADM TGBRX2; ADD =23505-27605; ADM TGBRX2+1
        STX TBRX2X
TGBRX2  LDA 0,2; STA 0,2; EAX -2,2; BRX TG2; BRU TGBRX2
TG6     COPY XA,B; EOR TWX; SKA ADMSK; BRU TG5
        MIN BZ  BUFFER ZAP COUNT
      • LDX =-NPORT     FIND THE BIGGEST BUFFER
        LDA TC+NPORT,2; STX TG6X; SKG TC+NPORT+1,2; BRX *-3; BRX *-2
        STA BZSIZE  PORT AND SIZE OF BUFFER ZAPPED.
        LDX TG6X; STB TC+NPORT,2    ZAP IT
        SET =GOBY,TF1+NPORT     INDICATE THAT HIS BUFFER WAS ZAPPED
        LDA KM1; XMA TY1EOF; EOR KM1; CAB (RESET FLAG, COMPLIMENT TO B)
        LDA =NPORT; ADD TG6X; STA TG6X (ZAPPED PORT)
*       (CHECK IF THIS CHARACTER IS THE SECOND CHARACTER OF
*       A DOUBLE CHARACTER PAIR, AND ITS BUFFER JUST GOT ZAPPED)
```

```
        SKB KM1; SKE TWX; BRU TG2 (NO, JUST TRY AND STORE CHAR)
        LDA TWA; STA TWAO; LDA TWC; STA TWCO (SAVE CHAR AND RETURN LOC)
        CLA; LDX TWX; BRM TWC   (RESTORE ZERO CHARACTER IN ZAPPED BUFFER)
        LDA TWCO; STA TWC; LDA TWAO; LDX TWX; BRU TWC+1 (STORE 2ND CHAR)
TWAO    0;*     (2ND CHAR OF DOUBLE PAIRS WHEN ZERO CHAR GETS ZAPPED)
TWCO    0;*     (ORIGANAL CALLER OF TWC ROUTINE)
TG1C    0;*     GARBAGE COLLECTOR STATISTIC
TG5C    0;*     GARBAGE COLLECTOR STATISTIC
TBRX1X  0
TBRX2X  0
TG6X    0
BZSIZE  ZRO

$TRI    0
        LDX TIX
        LDA ETBI+1,2; SKE TFREE; BRU *+3; MIN TRIC1; BRU TRI1
        MIN TIICTR; EOR ETBI,2; SKE KM1; BRM MONCR
        LDA TFREEC; STA ETBI,2
        LDA TFREE; XMA ETBI+1,2
        IF TLOG
        MIN TLOGP; LDB KM1; SKB* TLOGP; BRU *+3; LDB =TLOGP+1; STB TLOGP
        EOR X4; STA* TLOGP; EOR X4
        ENDF TLOG
        BRX *+1; BRX *+2; LDX =-NTBI; STX TIX
TRI1    COPY AB,AX,A; LCY 8; SKG K14; BRU *+2; BRM MONCR; STA TQTYPE
        CLA; LCY 8; SKG =NPORT-1; BRU *+2; BRM MONCR; STA TQPORT
        XXA; ETR K377; STA TQCHR
        BRR TRI
$TIX    0;* NEGATIVE INDEX INTO TBI
TRIC1   0

$TROPAIR 0;* OUTPUT 2 WORDS TO THE RING. THE SECOND WORD FIRST.
        STX TPAIRX; STA TPAIR1
        LDX TOX; EAX* TRO1    POINTER TO WHERE THE FIRST COMPLEMENT GOES
        LDA TFREE; BRM TRO  ADVANCE THE POINTERS
        LDA TPAIR2; BRM TRO   STORE THE SECOND WORD AND ITS COMPLEMENT
        LDA TPAIR1; EOR KM1; STA 0,2; LDA TPAIR1; STA 1,2  1ST WORD
        IF TLOG
        MIN TLOGP; LDB KM1; SKB* TLOGP; BRU *+3; LDB =TLOGP+1; STB TLOGP
```

```
        STA* TL0GP
        ENDF TL0GP
        LDX TPAIRX
        BRR TR0PAIR
TPAIR1  0
$TPAIR2 0
TPAIRX  0

$TR0TY1 0
        DIR
        RCY 8; LDA K77; ETR TF2,2; XXA; MIN CCT,2; CAX; LCY 8; MRG K2B5
        SKA =3700
        BRU *+5; STA TPAIR2; ETR KM400; BRM TR0PAIR; BRU *+2; BRM TR0
        EIR; BRR TR0TY1

$TR0M   0;* CALL TR0 IMMEDIATE. DONT SET ESCAPE LIKE TR0TY1 DOES.
        DIR; RCY 8; CXA; LCY 8; MRG K2B5; BRM TR0; EIR
        BRR TR0M

$TR0    0
        MIN T0ICTR
        STX TR0X; LDX T0X; EOR KM1
TR01    STA TB0+32,2
        BRX *+2; BRM M0NCR; EOR KM1
        IF TL0G
        MIN TL0GP; LDB KM1; SKB* TL0GP; BRU *+3; LDB =TL0GP+1; STB TL0GP
        STA* TL0GP
        ENDF TL0G
        XMA* TR01; SKE TFREE; BRM M0NCR
        BRX TR02
        LDA TR01; ADD =32-NTB0; SKE TR09; ADD =NTB0
        STA TR01      MOVE TO NEXT GROUP OF 32 OR WRAP AROUND
        ADD =16-2385-NTB0; SKG =TB0; ADD =NTB0  (LOOK AHEAD TO MAKE SURE
        CAX; LDA 0,2      THERE IS PLENTY OF ROOM IN THE RING)
        SKE TFREEC; BRU *+2; BRU TR02A
        CXA; SUB =NTB0/2; SKG =TB0-1; ADD =NTB0
        STA TR0SW     LET RING HALF EMPTY BEFORE USING IT AGAIN
        LDA CLINTV; STA CLINT1
TR02A   LDX =-32
```

```
TR02    STX T0X; LDX TR0X
        BRR TR0
TR0X    0;* SAVED X REGISTER
T0X     DATA -32  INDEX INTO 32 WORD GROUP IN OUTPUT RING
TR09    STA TB0+32,2

        IF TL0G      IF TL0G IS SET FALSE, B300 MUST BE MOVED SOMEWHERE.
TL0GP   0 *      POINTER INTO LOG BUFFER (WHICH FOLLOWS)
        IF DISC<3
        RPT 400B; 0 *,1  KEY THE LOG AREA WITH NON-ZERO CELLS); ENDR
        0 0      END OF LOG BUFFER PLUS 1 MUST CONTAIN A ZERO
        ELSF DISC=3; RPT 100B; 0 *,1; ENDR; DATA 0
$B300   BSS 300B        2314 SCRATCH BFR. SEE DRMSET AND DIM.
        ENDF DISC
        ENDF

        END
```

```
TS84A   IDENT   6-21-72                                  6/21/72  15:22     PAGE   1
        DELSYM
10RG    EQU     *
L0C0RG  MACR0 D;20RG EQU *-10RG; BSS D(1)-20RG; ENDM

$X      EQU     34000B

        L0C0RG  2

        IF      -EXEC

$SBRSRT ZR0
$WR4    BSS     30
$0UTN01 ZR0
$0UTN02 ZR0
$0UTN03 ZR0
$0UTN04 ZR0
$0UTN05 ZR0
GSIN1   EXT     0UTN01
GSIN2   EXT     0UTN02
GSIN3   EXT     0UTN03
STR01   EXT     0UTN01
STR02   EXT     0UTN02
STR021  EXT     0UTN03
STR03   EXT     0UTN04
GETN01  EXT     0UTN01
GETN02  EXT     0UTN02
GETN03  EXT     0UTN03
GETN07  EXT     0UTN05
$WR1    ZR0
$WR2    ZR0
$WR3    ZR0
$WR31   ZR0
$WR5    ZR0
$WR6    ZR0
$WR7    ZR0
$WR8    ZR0
$WR9    ZR0
$WR10   ZR0
```

```
$WR11   ZRB
$WR12   ZRB
$WR13   ZRB
        BSS     2               USED BY EXEC
$UPL    ZRB
$UBA    ZRB
$UBB    ZRB
$UBX    ZRB
$UBRL1  ZRB
$UBRL2  ZRB
$UBE    ZRB
$UCIN   ZRB
$UCBUT  ZRB
$UUNB   ZRB
$UEXFLG ZRB

        LBCBRG  105B

$EXRL1  ZRB
$EXRL2  ZRB
$UPRRL1 ZRB
$UPRRL2 ZRB
$SSRL1  ZRB
$SSRL2  ZRB
$UREAL  ZRB
$EXECL  BSS     6
$UEXL6  ZRB
$EXECLM EQU     EXECL-34000B
$ERCBDE ZRB

        ELSE    1

        ZRB
$XWR4   BSS     30
XWR41   EXT     XWR4+1
XWR42   EXT     XWR4+2
XWR43   EXT     XWR4+3
XWR44   EXT     XWR4+4
XWR45   EXT     XWR4+5
```

```
XWR46   EXT     XWR4+6
XWR47   EXT     XWR4+7
XWR4E   EXT     XWR47+1
XWR4E8  EXT     XWR4E+8         FOR 8 RESOURCES
        BSS     5
$XWR1   ZR0
$XWR2   ZR0
$XWR3   ZR0
$XWR31  ZR0
T1      ZR0
T2      ZR0
T3      ZR0
T4      ZR0
T5      ZR0
T6      ZR0
T7      ZR0
T8      ZR0
T9      ZR0
T10     ZR0
T11     ZR0
$UBRSRT ZR0     0               REFERENCE FOR XT1-11 SEE AF
$BASE   BSS 0
$UA     ZR0
$UB     ZR0
$UX     ZR0
$CMRL1  ZR0
$CMRL2  ZR0
$UE     ZR0
$CIN    ZR0
$COUT   ZR0
$UNO    ZR0
$EXFLAG ZR0

        LOCORG  1058

$EXRL1  ZR0
$EXRL2  ZR0
$PRGRL1 ZR0
$PRGRL2 ZR0
```

```
$SSRL1  ZRØ
$SSRL2  ZRØ
$UREAL  ZRØ
$EXECL  ZRØ
$EXECA  ZRØ
$EXECB  ZRØ
$EXECx  ZRØ
$EXECR  BSS     2
$EXECL6 ZRØ
$ERCØDE ZRØ


EXRL    EXT     EXRL1
PRGRL   EXT     PRGRL1
SSRL    EXT     SSRL1
$REMAC  ZRØ
$ANØ    ZRØ
$FDLUDL ZRØ
$LUDL   ZRØ     Ø               LUD LØCATER FØR THIS USER (SAME WERIS)
$FDUN   ZRØ
$GRU    ZRØ
$NGRP   ZRØ
$CR3    ZRØ
$UNPTR  BSS     2
UNPTR1  EXT     UNPTR+1
$MSZ2   ZRØ     Ø           USED AS Sw FØR TYPE ØF LØGØN FØR ACCTNG.
$LDTEMP ZRØ     Ø               STØRAGE FØR DISC. BUF. LØCATIØN
$SW6Ø   ZRØ     Ø               BRS 48/60 SW., NEG. IF 48/60
$SWPRP  ZRØ     Ø               PRØPRIETARY FILE SW.
$BRKTBL ZRØ     Ø               BRKTBL FØR ExEC BRS'S
$FDAN   ZRØ     Ø               FILE DIR. ACCT. NØ.
$SwINIT ZRØ     Ø               INITIAL CØMMANDS-FRØM FILE SW.
$NAST   ZRØ     Ø               NExT ACCØUNTING STØRAGE (FØR DISC ERR.)
$UNM    BSS     2
$UNM2   BSS     2
$UNMP   ZRØ
$UNMP1  ZRØ
$UNMPCF EQU     3*UNM-3*XWR1
$AFSVS  ZRØ
```

```
$NMCHX  ZRB
$ERRSW  ZRB
$SWBFF  ZRB
$SWTM   ZRB
$PJ     BSS     2
$PJ2    BSS     2
$EPJ    EQU     *
$FLPBS  ZRB
$FLX    ZRB
$GRCT   ZRB
$CMND   ZRB
$GRP    ZRB
$GAN    ZRB
$EBASE  EQU     *
T12     ZRB
T13     ZRB
T14     ZRB
$TLVO   ZRB     0               ( MAX. ADR. 177B)

        LOCBRG  2108

$DBF    BSS     128
$EDBF   EQU     *
$BVFP   EQU     DBF+124
$LFADR  EQU     BVFP-2
$BVFP1  EQU     BVFP+1
$BVFP2  EQU     BVFP+2
$MTIME  EQU     BVFP+3

$FDCBNT ZRB
$SSTIME ZRB
$SSID   ZRB
$INFIL  ZRB
ECRFIL  EXT     INFIL
$BUTFIL ZRB
ECDFIL  EXT     BUTFIL
$PVTSW  ZRB
$BPNCN  ZRB
$LETP   ZRB
```

```
$LGRS   ZR0
$RSREAL ZR0     0               FIRST RESOURCE STORAGE
$RSEND  EQU     7


* PANIC TABLES FOR EXEC FORKS

$EXSL   ZR0     0
$EXSA   ZR0     0
$EXSB   ZR0     0
$EXSX   ZR0     0
$EXSR1  ZR0     0
$EXSR2  ZR0     0
$EXSM   ZR0     0
$SSL    ZR0     0
$SSA    ZR0     0
$SSB    ZR0     0
$SSX    ZR0     0
$SSR1   ZR0     0
$SSR2   ZR0     0
$SSM    ZR0     0
$GETSTL ZR0
$SYSTL  ZR0
$EFORKD EQU     *

*EXEC TEMP. STORAGE
$TLV1   ZR0
$TLV11  ZR0
$TLV2   ZR0
$TLV21  ZR0
$TLV22  ZR0
$TLV23  ZR0
$TLV3   ZR0


* WORKING SPACE FOR EXEC ORS'S
```

```
GFN1    EXT     T10
G0FN1   EXT     T11
G0FN2   EXT     T12
G0FN3   EXT     T13
$G0FNFT ZR0
FN1     EXT     T2
FN2     EXT     T3
$FNPTR  BSS     2
FNPTR1  EXT     FNPTR+1
I0PTR   EXT     FNPTR
I0PTR1  EXT     FNPTR+1
LFDC1   EXT     T1
DF1     EXT     T1
DF2     EXT     T2
$GETPL  ZR0     0
DEL1    EXT     T1
DEL2    EXT     T2
DEL3    EXT     T3
FDEL2   EXT     T10
FDEL3   EXT     T11
FDEL4   EXT     T12
FDEL41  EXT     T13
PRMA1   EXT     T1
CSYS2   EXT     T1
CDEF4   EXT     T2
CDEF5   EXT     T5
PTR2    EXT     T2
PTSTAT  EXT     J14
PTR3    EXT     T3
PTR5    EXT     T5
$UNS1   ZR0     0
DUMP1   EXT     T8
ECDMP6  EXT     T12
ECRV6   EXT     T12
ERB2    EXT     T1
ERB3    EXT     T2
DRSW9   EXT     T3
$SYSRRL ZR0     0
RSYS2   EXT     T4
```

```
$GBI0PT ZR0     0
$GBI0FN ZR0     0
$GBI0SW ZR0     0
$GBI02  ZR0     0
$GBI0CR ZR0     0
$GBI0C2 ZR0     0
$GBI04  ZR0     0
$GET1   ZR0
I0SP1   EXT     GET1
$GET2   ZR0
I0SP2   EXT     GET2
$GET3   ZR0
I0SP3   EXT     GET3
$GET4   ZR0
I0SP4   EXT     GET4
ZYEAR   EXT     GET4
$GET5   ZR0
I0SP5   EXT     GET5
ZM0NTH  EXT     GET5
$GET6   ZR0
ZDAY    EXT     GET6
$GETFN  ZR0
I0FILE  EXT     GETFN
ZH0UR   EXT     GETFN
SUSR1   EXT     XWR4+31
$SUSR2  ZR0
ZMIN    EXT     SUSR2
$SUSR3  ZR0
ZSEC    EXT     SUSR3
SRSU1   EXT     XWR4+32
$SRSU20 ZR0
ZMS     EXT     SRSU20
$SRSU21 ZR0
$SRSU3  ZR0
$SRSU4  ZR0
$SAVELL ZR0     0
$SAVEFL ZR0     0
$SAVE0R ZR0     0
$SAVESL ZR0     0
```

```
*       PLUS,MINUS,PERI0D,EEE
PLUS    EXT     SAVELL
MINUS   EXT     SAVEFL
PERI0D  EXT     SAVE0R
EEE     EXT     SAVESL
C0PY1   EXT     SAVELL
C0PY2   EXT     SAVEFL
C0PY7   EXT     SAVE0R
C0PYT   EXT     SAVESL
CHFD    EXT     T1
MFD2    EXT     T1
MFD21   EXT     T2
CFI1    EXT     T4
CFI2    EXT     T5
TIME1   EXT     T1
TIME2   EXT     T2
TIME3   EXT     T3
TFILE   EXT     T9
TWMSG   EXT     T10
DGTCNT  EXT     XWR4+33
SPACES  EXT     XWR4+34
0VFL    EXT     XWR4+35
I0W     EXT     T4
I0FMT   EXT     T5
DEXP    EXT     T6
I0DEXP  EXT     T7
SIGN    EXT     T8
NC0NV   EXT     T9
RLI01   EXT     TLV1
RLI02   EXT     TLV2
RLI04   EXT     TLV21
RLI05   EXT     TLV22
RLI06   EXT     TLV23
RLI07   EXT     TLV3
RNDFLG  EXT     GET6
FFL     EXT     GBI0FN
FFA     EXT     GBI0SW
FFX     EXT     GBI02
```

```
OUTSX   EXT     GBIOCR
RNDX    EXT     GBIOC2
RLITEX  EXT     GBI04
A1      EXT     T13
A2      EXT     T14
A3      EXT     T11
IOLINK  EXT     T12
$SW     ZRO     0               WORKING ON EXEC BRS IF NEG.
$PJCSW  ZRO     0               PJC SW IF NOT ZERO HAS LUDL FOR PJC
*FOLLOWING 5 USED BY FIO ONLY (CAN COMBINE IF SPACE NEEDED)
$FIOW   ZRO
$ERRFLG ZRO
$9VDGTS ZRO
$IOD    ZRO
$ERRNUM ZRO

*       PUSHDOWN STACK
$PDP    ZRO
$PDC    ZRO
$PDL    BSS     9


$SVSSL  BSS     9
$ESAVD  EQU     *
$SSYSTL EQU     *-1
$RLV4   ZRO
$TLV4   ZRO
EXPFLG  EXT     RLV4
DFLG    EXT     TLV4
$DSTRCT ZRO 0                   DISTRICT NO. FROM LUD ON NEW FILE
$OPDSTR ZRO 0                     FROM OPERATOR PSWD FILE
$AFSV4  ZRO
*FOLLOWING 2 LOCS. NOT NECESSARY IF LUD IS READ FOR PROJ. COM.
$AFSV6  ZRO
$AFSV7  ZRO
$SXWR31 ZRO
$G9REMA ZRO
$RPTM   ZRO     0               ROYALTY PROG. TIME OF ID CHANGE
$OPTM   ZRO     0               TIME CONTROL OF OPERATOR STATUS
```

```
$TUZN   ZRB     0               USED BY CDX IN AF FOR TEMP STOR UZONE
$TCZN   ZRB

XT1     EXT     T1
XT2     EXT     T2
XT3     EXT     T3
XT4     EXT     T4
XT5     EXT     T5
XT6     EXT     T6
XT7     EXT     T7
XT8     EXT     T8
XT9     EXT     T9
XT10    EXT     T10
XT11    EXT     T11
XT12    EXT     T12
XT13    EXT     T13
XT14    EXT     T14
$XTE    ZRB     0

        ENDF

        LOCORG  6000B
$LMCPU  ZRB
$SVST   ZRB     0               SUPERVISOR START IF NEG.

$RPCHG  ZRB
$RPMAX  ZRB
$RPN    ZRB
$UZONE  ZRB
$ENDTS  ZRB

        IF      -EXEC

        FORGET
        ENDF

        END
```

```
SET   IDENT                                                    6/21/72  15:23     PAGE   1

      LISTM
* MONITOR MODULE 0
DR    EQU -1

$RAERT EQU *-4000B     REFERENCED THRU PAGE 6 BY RAD ROUTINES
       BSS 512 RAD ADDRESSING ERROR TABLE

* 'DRMSET'
* THIS ROUTINE INITIALIZES THE DISC I/O.
*
DRMSET ZRO
       LDA KM1; STA UNIT
       LDX =-NFILE; STA EXBP,2; BRX *-1
       IF DISC<3
       LDA =DRQ; STA EDCL; STA IDCL
       LDA KM1; STA NDCL; STA IDCL1; STA DTXS1; STA WSW
       LDA =02B5-20B5  (NOP-EOM) SEE PPOFF IN PAC
       CLX; IF DISC=1; BRU DSET1; ENDF
       SKS 10027B  (DISC READY TEST); BRU DSET1; LDX ONE; ADM PPOFF,2
       SKS 10022B; BRU DSET1; LDX TWO; ADM PPOFF,2
       SKS 10023B; BRU DSET1; LDX THREE; ADM PPOFF,2
DSET1  STX DPC     COUNT OF DATA PRODUCTS DISCS
       CXB; LSH 24+18; ADD KS6; STA DSCTOP  HIGHEST DP SECTOR
       ELSE DISC=3
       LDA B300A; ADD =40003*3; STA 25B    REAL ADRS OF 2314 SCRATCH BFR
       ENDF
       BRR DRMSET

PROTEC ZRO;*   SET DISC WRITE PROTECT SWITCH IF DISC IS WRITE PROTECTED.
       LDA KM1; STA RWD; CLA; STA T
       IF DISC<3
       LDA =20002B; LDX KM40       CHECK DISC
RW1    EOM 10026B; POT T; SKS 12026B; BRU *-1; SKS 13026B; MIN RWD
       ADM T; BRX RW1
       SKS 10027B; BRR PROTEC; LDX K166; STX T; LDX KM40
RW2    EOM 10027B; POT T; SKS 12027B; BRU *-1 (WAIT FOR STATE 1 READY)
       SKS 13027B; MIN RWD (WRITE PROTECTED); ADM T; BRX RW2
```

```
        SKS 10022B; BRR PROTEC; LDX K2B6; STX T; LDX KM40
RW3     E0M 10022B; P0T T; SKS 12022B; BRU *-1
        SKS 13022B; MIN RWD; ADM T; BRX RW3
        SKS 10023B; BRR PROTEC; LDX =3B6; STX T; LDX KM40
RW4     E0M 10023B; P0T T; SKS 12023B; BRU *-1
        SKS 13023B; MIN RWD; ADM T; BRX RW4
        ENDF DISC<3
        BRR PROTEC
*
CK205   ZR0;*   TELL EXEC WHETHER OR NOT 205 IS WIRED HIGH
        LDA *+1; XMA 205B; ARMI ARM205; MIN CK205; ARMI DIS205
        SKE CK205+2; MIN SW205; BRI CK205

CKXMA   ZR0;* TEST XMA ON READ ONLY PAGE
        LDB =RSR+40B; LDX RRL3; BRM LABEL
        LDA CKXMA2; XMA 43B; CAB; XMA *,4
CKXMA1  STB 43B; SKE 43B; MIN SWXMA; BRR CKXMA
CKXMA2  BRU* =CKXMA1  MINIMUM OF 2 CYCLES REQUIRED FOR INTRPT INSTRUCTION

CKRD    ZR0; CLB;ZILCH MACRO D;J EQU D(1)
        E0D 10226B+2000B*J+D(2); PIN T
        LDA T; E0D 10226B+2000B*J+D(2); PIN T; SKE T; BRU *+2; BRU *-4
        LDA =1+20000B*J; ADM T; E0D 10026B+D(2); P0T T
        E0D* 10000B+D(2); E0D 14200B; P0T =CKRDBF+100B*4B4
        E0D 2226B+D(2); LDA =1500000/1750/4; STA TS
        SKR TS; BRU *-1; SKS* 10026B+D(2); BRU *+2
        BRU *+6; LDA ONE; LSH J; ADM RAT; SKS* 10026B+D(2); BRU *-1
        ENDM
        ZILCH 0,0; IF DR; ZILCH 0,100B; ELSE; ZILCH 1,0; ENDF
        IF DR; LDX =-512; CLA; STX CKRDBF+1; STA CKRDBF; BRM HRELB
         LDA CKRDBF; LDX CKRDBF+1; ADD K40; BRX *-6; ENDF; BRR CKRD
CKRDBF  BSS 100B                BUFFER FOR CKRD

CKMEM   0;* SCAN MEMORY FOR BAD PARITY
        LDA MP7; XMA 56B; STA T1    SAVE MEMORY TRAP LOCATION
        LDA RRL3; STA T; ETR K37; STA RRL3
MP1     LDA RRL3; ADD K100; BRM RLABEL; LDX KM4B3
MP2     LDA 340000B,2    CHECK A CELL AND POSSIBLY TRAP TO MP7+1
MP2A    BRX MP2
```

```
        LDA RRL3; SKN M48K; SUB K1000; SKG =2700B; BRU MP1
        LDA MP9; LDB MP8; LDX =100076B; SKB KM1; BRM STOP
        LDA T; BRM RLABEL; LDA T1; STA 56B
        BRR CKMEM

MP9     0;* NUMBER OF CELLS IN MEMORY HAVING BAD PARITY
MP8     0;* ADDRESS OF LAST CELL IN MEMORY HAVING BAD PARITY
MP7     BRM *+1; 0;* MEMORY PARITY TRAP ROUTINE FOR CKMEM
        STA 34000B,2   FIX UP BAD WORD
        LDA RRL3; LSH 5; ETR =37B*40000B; STA MP8   PAGE OF ERROR
        CXA; ADD K4000; ETR K3777; ADM MP8  16 BIT ADRS
        MIN MP9; BRI =MP2A

RADTYM  0;* SYSTEM IS NOW FULLY SET UP. IF RAD IS WRITE PROTECTED SET RWR
*       PLUS FOR EXEC AND AS A SEPERATE FUNCTION WHICH CAN BE OVERLAPPED
*       KEEP SETTING THE TYMNET KEY FOR 1/2 SEC SO BASE KNOWS 940 IS UP.
        LDA <M1; STA RWR; CLA; STA T
        LDA =40000B/32+1; LDX =-32
RM1     EOD 10026B; POT T; SKS* 13026B; MIN RWR; ADM T
        LDB FG940A; STB UPFL; LDB FG940B; STB UPFL+1
        BRX RM1
        BRR RADTYM

*       SYSTEM INITIALIZE ROUTINE
*
$SETSA  EQU *   BEGINNING OF SYSTEM SET UP CODE. REACHED FROM SETSET.
        IF DISC<3; EOM 140106     REWIND THE DSWAP TAPE; ENDF
        LDA =RSR; STA RRL3; ETR K37; STA RLTS
        CLA; BPT1; LDA KM1; SKN M48K; STA M48K
        IF DISC=3; LDA 2; ELSE; LDA 22B; RSH 13; ENDF
        ETR K37; STA DISCN
SET9    BRM SSET; BRM CKRD
        BRM DRMSET; BRM PROTEC; BRM CKXMA; BRM CKMEM; BRM RADTYM
$SET7   EQU *
        IF DISC<3
        SKS 14026B; BRM MONCR (DISC HEADER SWITCH ON)
        SKS 10027B; BRU *+3; SKS 14027B; BRM MONCR
        SKS 10022B; BRU *+3; SKS 14022B; BRM MONCR
        SKS 10023B; BRU *+3; SKS 14023B; BRM MONCR
```

```
        ENDF

        CKN; EIR
        BRU PACG01      START SCHEDULAR

*
$SSET   ZRB;* RESET SYSTEM
        ARMI AIRMD
        CLA; STA REAL
        LDX =-77B; LDA CLINTC; STA 100B,2; BRX *-1
        STA SETSET
        LDX =-NPAC*NPPAR; CLB
        LDA =SMT; STA ADRSMT
SET1    LDA =7000000B; STA PTEST,2
        CXA; ADD =PPTR; ADD =NPPAR; STA PPTR,2
        EAX NPPAR-1,6; BRX SET1
        LDA =PPTR2; STA FPLST; STB PPTRU
        STB PUCTR
        LDX =-NPUC*4
$SET3   CXA; ADD =EPUCT3; STA EPUCT,2
        EAX 3,6; BRX SET3; CLB; STB EPUCM3
        LDA =PUCT; STA FPULST; LDA =PUBPTR; STA PUBPTR; STA PUEPTR
        LDA BNE; STA CMPST
$SET6   LDX =-NPORT; LDA KM1; STA EWERIS,2; BRX *-1
        LDX =-NJOB; CLA; STA EPMTP,2; BRX *-1
        STA NBACT; LDA K13; BRM SETFRE; LDA K14; BRM SETFRE
        LDA K22; STA TS; SUB K15; SKN M48K; ADD EIGHT; STA TSA
FRLP    LDA TS; BRM SETFRE; MIN TS; SKR TSA; BRU FRLP
        IF      DDIFP
SET4    LDX     KM100
        LDA     EPBPTB,2
        STA     200B,2
        BRX     *-2
        ELSE
SET4    LDB BSTU; CLA; LDX KM100
        SKE 200B,2; BRU *+2; STB 200B,2; BRX *-3
        ENDF
        LDX =WDTB-WDTE
        LDA WDTE,2; STA* WDTE+1,2; BRX *+1; BRX *-3
```

```
        LDA =ACST; STA PACST
* MEMORY MAPPING
*   PAGE 7=LOG AND SET, PAGE 10=DISC, 11=W, 12=NET CODE AND TTY BUFFERS

        IF DISC=3
*   EXEC IS LOADED AT 12-16, NET AT 17. MOVE EXEC TO 15-21 AND NET TO 12
        LDA =17221222B
SET2    STA RRL1; LRR1; POT RRL1    SET UP TO MOVE A PAGE OF CODE
        LDX KM4B3; LDB 4000B,6; STB 10000B,6; BRX *-2
        SUB =0101B4; SKE =11141222B; BRU SET2
*       EXEC IS AT 15-21 AND NET IS AT 22. MOVE NET TO 12
        LDX KM4B3; LDB 20000B,6; STB 14000B,6; BRX *-2

        ELSE DISC<3

*       READ SYSNO FROM DISC AND PUT IN LOCATION 200B
        LDA K200; LDB =10B*4000B; BRM GDC
        LDA =10B6; STA RRL1; LRR1; POT RRL1; LDA 75B,4; LRSH 12; STA 200B

        LDA =DSCTBL; STA T4; BRU SET2
DSCTBL  DATA 0,10B*4000B      DISC CODE
        DATA 40B,11B*4000B    W BFR CODE
        DATA 340B,12B*4000B   NET CODE AND TYMNET BUFFERS
        DATA 100B,15B*4000B   EXEC PAGE 12
        DATA 140B,16B*4000B   EXEC PAGE 13
        DATA 200B,17B*4000B   EXEC PAGE 14
        DATA 240B,20B*4000B   EXEC PAGE 15
        DATA 300B,21B*4000B   FIO
        DATA -1
SET2    LDB DISCN; LSH 24+13    (MUL BY 20000)
        ADD =20000B-400B+20000B  (20000-400 IS ARM POS 62)
*               20000 MOVES UP 1 DISC TO SECOND HALF OF SYSTEM
        ADD* T4; MIN T4; LDB* T4; MIN T4
        BRM GDC     GET DISC TO CORE
        SKN* T4; BRU SET2
        ARMI AIRWD

        ENDF
```

```
* SET SMT'S AS GRABBED PAGES
        LDX =NSSMT-NSMT; LDA =1B7+1; STA SMTE,2; BRX *-1
        IF EXPMT; LDX =-NXSMT; STA EXSMTE,2; BRX *-1; ENDF
* SET MONITOR AND EXEC RMT'S AND RMC'S
        LDB <27; LDA =SMT; ADD ONE; LDX ONE; STB RMC,2; STA RMT,2
        ADD ONE; EAX 1,2; SKR MPGC; BRU *-5
        STB RMC; STA RMT; EAX 1,2
        ADD ONE; STB RMC,2; STA RMT,2      DISC CODE
        ADD ONE; EAX 2,2    SKIP OVER w CODE. IT IS NOT AN SMT.
        STB RMC,2; STA RMT,2      TYM BUFFERS
        CLB; EAX 2,2
        ADD ONE; EAX 1,2; STB RMC,2; STA RMT,2; SKR EPGC; BRU *-5
        LDA =SMT; ADD TEN; STA T; LDX KM5; STX T2
        BRM REL; MIN T; LDA T; LDX T2; BRX *-5
        LDX =20200007B; STX SMTPG7; LDA =SMTPG7; STA RMT,2
        BRM TYMSET
        BRR SSET
SMTAD   DATA    SMT
EXP     DATA    3               NO. OF EXEC PAGES -1.
MPGC    DATA    5               NO. OF MONITOR PAGES-1
EPGC    DATA    4               NO. OF EXEC PAGES-1+FIO
*
*
SETFRE  ZRB; STA TS; CAX; LDA =32; BRM RAINS; LDX TS; LDA KM1
        STA RMT,2; CLA; STA RMC,2; BRR SETFRE
TS      ZRB
TSA     ZRB
*
*


WDTB    EQU *

WDPIT   BRM TRAPI; O 40B
WDIMT   BRM TRAPM; O 41B
WDRMT   BRM TRAPR; O 43B
WDUMT   BRM TRAPT; O 44B
WD31    BRM INT31; O 31B
```

```
WD33    BRM INT31; 0 33B                                                SET     PAGE    7
        IF DISC=3
        BRU 1; 0 1      KEEP FROM MONCRASHING ON TOO FAST BRING UP
        0 DRQ; 0 26B
        0 DRQU; 0 27B
        0 DRQ; 0 RQ1A
        0 DRQ; 0 RQ1
        0 DRQ; 0 RQ2
        0 DRQ; 0 RQ3
        ENDF
WDPWNI  BRM PWNI; 0 36B
WDPWFI  BRM PWFI; 0 37B
WDCP    BRM CPUP; 0 56B
WDIOP   BRM IOP; 0 57B
        BRM RAI1; 0 64B
WDIDM   BRM RADI2; 0 65B
        IF FPH; BRM FPOVI; 066B; ENDF
CLOCK2  BRM CLINT; 0 74B
        SKR CLOCK3; 0 75B
        BRU 201B; 0 201B
        BRU 202B; 0 202B
        IF DISC=3; BRM IDM; 0 203B; ELSE; BRU 203B; 0 203B; ENDF
        BRM TQ; 0 204B
        BRM RCL; 0 205B
        BRU 206B; 0 206B
        BRM MONCR; 0 215B

WDTE    EQU *
*
*       SYSPOP TRANSFER VECTOR
*
POPTB   BRM     TRAP            500B5
        BRM     TRAP
        BRM     TRAP
        BRM     TRAP
        BRM     TRAP            504B5
        BRM     TRAP
        BRM     TRAP
        BRM     TRAP
```

```
BRM     TRAP
BRM     TRAP
BRM     TRAP
BRM     TRAP
BRM     TRAP            514B5
BRU     FFDIEF
BRU     LDFMEF
BRU     STFMEF
BRU     FFADEF          520B5
BRU     FFMPEF
BRU     FFDVEF
BRU     FFSBEF
BRU     FFSIEF            524B
BRU     RSPX
BRU     SSPX
BRU     LDFME
BRU     STFME             530B
BRU     RCPX
BRU     SCPX
BRU     PCEX
BRU     CITS            534B5
BRU     WCDP
BRU     SNEX
BRU     GCDP
BRU     ISC1            540B5
BRU     SIC1
BRU     FFSBE
BRU     FFSIE
BRU     SKPX              544B5
BRU     SKLX
BRU     FFDVE
BRU     FFDIE
BRM     TRAP            550B5
BRM     TRAP
BRU     FFADE
BRU     HFDVE
BRU     HFMPE           554B5
BRU     HFSBE
BRU     HFADE
```

```
         BRU     WCIP
         BRU     WIE             560B5
         BRU     CIE
         BRU     SKSGP
         BRU     SKSEP
         BRU     WCMP            564B5
         BRU     GCIP
         BRU     LDPP
         BRU     STPP
         BRU     SBRME           570B5
         BRU     GRB
         BRU     FFMPE
         BRU     BS
         BRU     TCIX            574B5
         BRU     TCBX
         BRU     BIBX
         BRM     TRAP
EP0PTB   EQU     *

*        ***************     STATISTICS
         IF      ST
$SSS     ZRB;* STORE STATISTICS
         XMA SSS; ETR ADMSK; ADD =40001B; COPY AX,B
SSL00P   EQU * ENTER A STATISTIC INTO TABLE
         LDA 0,2; LCY 9; LDA* 0,2; XXB; EXU *,2
         BRU BPSS1                 STORE A WORD AND RETURN
         BRU BPSS2                 STORE A WORD AND CONTINUE
         BRU BPSS3
         BRU BPSS4
         BRU BPSS5
         BRU BPSS6
         BRU BPSS7
         BRU BPSS10
         BRU BPSS11

SS9      EQU *   SETUP STATISTICS POINTER IF NEAR BUFFER BOUNDARY,EXIT
         LDA SP; ETR K377; SKG =340B; BRU SS9A
         EOR SP; ADD =400B+40B
         SKG =ELDATA; BRU *+3; LDA =LDATA; ADD K40; SUB K40
```

```
        SKE SP1; BRU SS9B (NO DATA OVERRUN);
        LDA =7500001B; STA* SP (DATA OVERRUN); MIN STSW; BRU SS9A
SS9B    XMA SP; CAX; LDA =7600002B; STA 0,2; LDA REAL; STA 1,2
SS9A    CBX; LDA SSS (CALLER'S RELABELLING); BRU SSRET

OPSS1   STA* SP; MIN SP; BRU SS9

OPSS2   STA* SP; MIN SP; COPY BX,B; BRX SSLOOP

OPSS3   CBX; RSH 6          JOB
        LDA PACQUE; RSH 6       PACQUE
        LDA PG44; RSH 6         PG44
        LDA PACJOB; RSH 3       PACJOB
        LDA ACTR; RSH 3         ACTR
        STB* SP; CLB; MIN SP; BRX SSLOOP

OPSS4   CAX; STA* SP    UTTY
*       LDA TTYBRK,2; COPY BX,B TTYBRK (OLD STATISTIC)
        LSH 6; ADM* SP; MIN SP; BRX SSLOOP

OPSS5   CBX; RSH 8
        LDA RPAGEH; RSH 8
        LDA RPAGEL; RSH 8; STB* SP
        CLB; MIN SP; BRX SSLOOP

OPSS6   ADD ONE; BRU OPSS11      STORE A P.U. ENTRY

OPSS7   COPY BX,B
        RSH 12; LDA TIME1; LSH 12; STA* SP; MIN SP; BRX SSLOOP

OPSS10  CAX; LDA RL1,2; STA* SP; MIN SP
        LDA RL2,2; STA* SP; MIN SP
        LDA PIM,2; STA* SP; MIN SP
        COPY BX,B; BRX SSLOOP

OPSS11  EQU *
        CAX; LDA 0,2; STA* SP; MIN SP
        LDA 1,2; STA* SP; MIN SP
        LDA 2,2; STA* SP; MIN SP
```

```
$STG01 BSS 0       BRS 27. TURN ON STATISTICS
       BRM TRAP    REMOVE THIS TRAP WHEN STATISTICS ARE CHECKED OUT
       LDA X4; SKA SDBM8; BRM TRAP; SKA STSW; BRU SG1
       LDA =7402008; STA STDISC
       IF ALOG
       LDA LOG9; STA LOG2
       ENDF
       LDA =LDATA; STA SP; STA SP1
       CLA; STA STC
SG1    LDA STC; STA SS01
       LDA XM1; STA STSW; BRM NPOPX

$STSTP1 BSS 0        BRS 28. TURN OFF STATISTICS
       LDA XX; XMA STSW
       SKA X4; BRU *+2; BRU STP2
       LDA RRL3; BRM SSS
       2 =77B5+2                FINAL TERMINATOR
       1 REAL
STP2   LDA SP; ETR KM400; ADD K400
       ADD ONE; SKG =ELDATA; BRU *+3; LDA =LDATA; ADD ONE; SUB ONE
       STA SP; BRM NPOPX

$SDUMP EQU *   DUMP 400B WORDS OF STATISTICS DATA
       CLB; LDA DLITS; SKG K400; BRU SD2
       DIR; LDA* SP; SKE =7500001B; BRU SD1
       LDA SP; ETR KM400; ADD =400B+40B
       SKG =ELDATA; BRU *+3; LDA =LDATA; ADD K40; SUB K40
       STA SP
SD1    EIR
       BRM GETBLK; LDX SP1; STA 377B,2
       MIN STC; LDB STC; STB 376B,2     BLOCK NUMBER
       XMA STDISC; COPY AB,XA
       ADD K1000; SKG =ELDATA; BRU *+3; LDA =LDATA; ADD K400; SUB K400
       XMA SP1; LDX =4B7+10B; BRM DTC; BRM DTS
       LDA XM1; SKA STSW; BRU *+2; STA STSW
       BRU PUST1
SD2    EQU *   RUNNING OUT OF DISC SPACE. TURN OFF STATISTICS.
```

```
        LDA SP1; STA SP; LDA XX; STA STSW; BRU PUST1

        ENDF
*       A BUFFER SHARED BY LOG AND STATISTICS IS DEFINED AT THE END
*       OF SET BY THE LOAD MAP

        FORGET
        END
```