

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O), by the Municipality of Amsterdam, by the University of Amsterdam, by the Free University at Amsterdam, and by industries.

Introduction

This report lists the ELAN program MICRO64KL, which constitutes an ALGOL 60 system, that also precompiles and inserts library procedures. Much of this program is taken from MR 84: The ELAN source text of the MC ALGOL 60 system for the EL X8 by F.E.J.Kruseman Aretz and B.J.Mailloux . It is emphasized that the gap between MR 84 and this report is filled by no means with work of the author alone. In fact, the author started from an unpublished program MICRO48K, mainly work of F.E.J.Kruseman Aretz.

In this program a part of the system is allocated above 32K. From the hardware point of view this is no problem. But with the ELAN assembler used it was impossible to assemble instructions in this region. So it became a task of the loader to get the instructions in the right place. The label list given at page II corresponds to the assembler output, and the final address (allocated by the loader) follows the address allocated during assembly time if they differ. To run the system, assemble and execute the ELAN program given here; then load the output produced.

A description of MICRO64KL is given in MR 117/70: An ALGOL 60 library system for the EL X8 (in dutch).

The author should like to thank Mr.P.Beertema for his help in preparation of this manuscript.

II

CRE	'00000'	STA°IST	'01031'	MSIGN	'01174'	HOK	'02035'
RE	'00005'	STA°RAY	'01032'	MM	'01174'	START	'02057'
REST	'00005'	BEG°NLI	'01033'	HEP	'01175'	FOU°ARY	'02060'
TY	'00010'	COR°ION	'01034'	NN	'01175'	FOU°BLE	'02111'
KY	'00011'	BEG°32K	'01035'	XX	'01176'	IOINTP	'02116'
PR	'00013'	LEN°LER	'01036'	ABS	'01200'	INTAB	'02134'
PU	'00022'	TRA°BEG	'01040'	FIXFLO	'01201'	RES°ORE	'02204'
PUST	'00022'	BEG°BLE	'01040'	LAS°ION	'01202'	END°RIT	'02212'
PC	'00023'	INF°END	'01041'	ANEW	'01203'	EXI°VAP	'02217'
CPU	'00024'	LEN°32K	'01042'	LIMIT	'01204'	WAS°IME	'02220'
CRD	'00031'	BEG°RAR	'01043'	CPLTIM	'01205'	DAT°IAL	'02221'
DR	'00033'	END°ACE	'01044'	CPLCHK	'01206'	RUN°IME	'02234'
IBULEN	'00200'	MEM°ETO	'01045'	INI°RPT	'01207'	CTYPE	'02251'
REC°LEN	'00200'	BEG°BLE	'01046'	CRI°CAL	'01210'	TYPE	'02276'
MAX4	'02740'	BEG°BLE	'01047'	KIC°OFF	'01211'	TYPE3	'02301'
BFL	'00052'	NBR°NES	'01050'	NOSWAP	'01212'	TACT	'02310'
ITIM	'00011'	END°GUE	'01051'	ATL°SGN	'01213'	TWAIT	'02315'
PBULEN	'00200'	END°BLE	'01052'	ATL°GN1	'01214'	TOFF	'02331'
MAX2	'00500'	END°BLE	'01053'	ATL°GN2	'01215'	KEYBD	'02334'
BUFLTH	'00400'	END°BLE	'01054'	BEG°ENT	'01216'	XEENS	'02377'
MAX	'01000'	END°ARY	'01055'	DYST	'01216'	STOP	'02401'
INSTCK	'00260'	ERR°OUS	'01056'	D18	'01217'	XEEN	'02403'
STKBEG	'00400'	DAN°OUS	'01057'	D18M1	'01220'	XGET	'02407'
STKEND	'00777'	RUN°BER	'01060'	D23M3	'01221'	NXT°DSL	'02424'
DELTA1	'04000'	VAL°ANT	'01061'	KBUF	'01222'	NKS	'02430'
SUB°VAR	'14745'	LIN°TER	'01063'	NTAB	'01226'	HAND	'02435'
REC°VER	'15503'	LAS°BOL	'01064'	HALF	'01230'	DNAH	'02444'
RUN°YST	'16405'	LAS°IER	'01065'	SGNBIT	'01231'	HANDRD	'02450'
BEGCAT	'36000'	WANTED	'01067'	SIXMM1	'01232'	KYBD	'02460'
END°GUE	'35161'	FLE°ODE	'01070'	TENTH	'01233'	FINIS	'02461'
DELTA	'24551'	PICO	'01071'	SIXMIL	'01235'	FINIS1	'02465'
BEG°32K	'72000'	TAR	'00140'	SCH°TEN	'01240'	KEYA	'02472'
INS°LST	'72574'	KAR	'00144'	DATMES	'01242'	KEYT	'02474'
BEG°TAB	'73115'	CLAR	'00334'	SERMES	'01244'	CLOCK	'02502'
END°32K	'73175'	SERIAL	'01071'	SMES	'01247'	NSTART	'02505'
		DATE	'01072'	SMES1	'01250'	ESTART	'02505'
PRE°NCE	'01000'	TSS	'01073'	TRNMES	'01251'	L	'02506'
COM°DIR	'01001'	SYM	'01076'	RUNMES	'01255'	KEYM	'02510'
DRF°BEG	'01002'	TBUF	'01077'	XMES	'01261'	INOCT	'02513'
DRC°END	'01003'	ASAFE	'01123'	HMES	'01263'	COPY	'02527'
DRL°BEG	'01004'	SSAFE	'01124'	CRNLMS	'01266'	CIAR	'00100'
DRL°END	'01005'	FSAFE	'01125'	CLCKMS	'01267'	CPAR	'00220'
DRT°BEG	'01006'	SAFE8	'01127'	PTR°MES	'01273'	CSE°OND	'02542'
DRL°END	'01007'	BBSAFE	'01130'	CLC°MS1	'01276'	CTHIRD	'02544'
DRBEG8	'01010'	KCASE	'01131'	FIL°MES	'01301'	PASTE0	'02550'
DREND8	'01011'	INSW	'01132'	INF°TOR	'01306'	PASTE1	'02551'
DRBEG4	'01012'	MAYI	'01133'	PASTE	'01324'	SS	'02552'
DREND4	'01013'	XEENW	'01134'	CLP°TE0	'01326'	HEP	'02553'
DRBEG2	'01014'	CBASE	'01135'	CLP°TE1	'01327'	CWD	'02554'
DREND2	'01015'	CASE	'01165'	INTREP	'01330'	DONT	'02555'
DRBEG	'01016'	TEMP	'01166'	TELXTB	'01530'	CRE°DER	'02564'
DREND	'01017'	CNT	'01166'	TENPOW	'01547'	CPCHER	'02572'
DMPBEG	'01020'	BEXP	'01166'	SCALE	'01561'	CPC°ER3	'02575'
DMPEND	'01021'	BSAFE	'01167'	GIANT	'01573'	END	'02610'
BEG°ACK	'01022'	PT	'01167'	DENTST	'01575'	KTAB	'02611'
END°ACK	'01023'	ESIGN	'01167'	NAIGA	'01645'	CW1	'02644'
NBR°NES	'01024'	STOCK	'01170'	ENDRUN	'01724'	CW2	'02650'
MAX°NES	'01025'	HEAD	'01171'	FIXT60	'02002'	KTABR	'02663'
CATEND	'01026'	TAIL	'01172'	LOOP	'02014'	KTAB1	'02665'
PER°END	'01027'	DEXP	'01173'	BRUSH	'02026'	CW4	'02704'
BEG°GUE	'01030'						

III

READ	'02710'	ATTENTION	'03637'	ISKIP	'05401'	GETBUF	'07203'
READ1	'02711'	DRS*ART	'03677'	FRO*URUM	'05405'	GET*UF1	'07217'
LOOP	'02725'	DRS*RT1	'03704'	STA*OUT	'05422'	TODRUM	'07223'
WORK	'02742'	DRUM	'03711'	PRO*SS4	'05425'	STA*IN	'07233'
DECBIN	'02752'	DRNOK	'03721'	DRTOCR	'05425'	PRO*SS2	'07236'
D0	'02771'	TRY*AIN	'03731'	PRO*SS5	'05455'	CRTODR	'07276'
D1	'03002'	REA*OMP	'03743'	CRTODR	'05455'	PRO*SS3	'07267'
P0	'03007'	REA*OMP	'03745'	TODRUM	'05525'	DRTOCR	'07267'
P1	'03020'	PRO*SS6	'03762'	STA*IN	'05541'	FRO*URUM	'07321'
ASEMBL	'03024'	WAI*RPT	'04005'	ISTART	'05544'	STA*OUT	'07332'
ROUND	'03043'	END*CDU	'04024'	IST*RT2	'05546'	PSTART	'07335'
ZERO	'03063'	TRA*ORT	'04030'	IST*RT4	'05550'	PST*RT1	'07343'
SIEVE	'03072'	END*ANS	'04044'	IST*RT12	'05561'	PST*RT2	'07352'
SEPEX	'03114'	PRO*SS9	'04047'	IST*RT13	'05562'	PST*RT3	'07355'
SEP	'03116'	PRO*SS10	'04047'	READER	'05563'	PST*RT4	'07361'
CLEAN	'03120'	PRO*SS11	'04047'	INOK	'05607'	PST*RT5	'07364'
XPONT	'03124'	PRO*SS12	'04047'	CON*ACT	'05625'	PCHER	'07371'
NXTSYM	'03145'	PRO*SS13	'04047'	CON*CT1	'05642'	PCHER1	'07411'
TAPE	'03146'	PRO*SS14	'04047'	CON*CT2	'05674'	PCHER2	'07420'
PARAM	'03147'	PRO*SS15	'04047'	ENS*ART	'05703'	PUNOK	'07431'
STORE	'03164'	HERE	'04050'	HERE	'05713'	PUNOK1	'07453'
SPILL	'03170'	IAR	'00124'	PAR	'00210'	PUNOK2	'07465'
FIX	'03174'	ISS	'04051'	PSS	'05714'	PUNOK3	'07472'
FIX1	'03175'	ICHECK	'04053'	SUM*ECK	'05716'	PUNOK4	'07505'
FLO	'03212'	READY	'04054'	HEPCNT	'05717'	PUNCH	'07510'
FLO1	'03213'	IPOS	'04055'	PPOS	'05720'	FIXP	'07512'
FLO2	'03214'	CRB*OF4	'04056'	SHIFT	'05721'	ABS*IXP	'07516'
TEN	'03242'	CRB*OE4	'04057'	CRB*OF2	'05722'	FLOP	'07520'
LAST	'03253'	QUEU4	'04060'	CRB*OE2	'05723'	PO	'07522'
CONV	'03257'	DRB*OF4	'04061'	QUEU2	'05724'	SMA	'07534'
DIV	'03302'	DRB*OE4	'04062'	DRB*OF2	'05725'	FLEXHP	'07545'
MUL	'03317'	DRR*OM4	'04063'	DRB*OF2	'05726'	COHPTB	'07574'
MULT	'03332'	QUEU5	'04064'	DRR*OM2	'05727'	NLCR	'07575'
AH	'03351'	CRB*OF5	'04065'	QUEU3	'05730'	TAB	'07600'
RU	'03370'	CRB*OE5	'04066'	PBUNOF	'05731'	LC	'07604'
GPONE	'03434'	ITEL	'04067'	PBUNOE	'05732'	UC	'07607'
DIGITS	'03436'	REPOS.	'04070'	PBUSY	'05733'	FLEXIS	'07611'
SHRTCT	'03450'	IBUNOF	'04071'	CHANGE	'05734'	HERE	'07634'
RET	'03456'	IBUNOE	'04072'	PHOK	'05735'	PRAR	'00154'
NDBTP	'03474'	IBUSY	'04073'	DTS	'05736'	EMPTY	'07634'
STSP	'03503'	DONE	'04074'	CCNT	'05737'	VER*CON	'07635'
STZ	'03516'	DFDSGN	'04075'	FLCASE	'05740'	LIN*BER	'07636'
IZERO	'03522'	TRNSGN	'04076'	THEAD	'05741'	PAGCNT	'07637'
IZERO3	'03525'	CRB*F41	'04100'	CRBUF1	'05772'	BUFPOS	'07640'
GEN	'03530'	CRB*F42	'04301'	CRBUF2	'06173'	BUFBEQ	'07641'
FILL	'03545'	CRB*F51	'04502'	PBUF1	'06374'	BUF*OOM	'07642'
HERE	'03550'	CRB*F52	'04703'	PBUF2	'06575'	CRB*NCE	'07643'
DRAR	'00254'	IBUF1	'05104'	PUNOK	'06775'	DRROOM	'07644'
DRSS	'03551'	IBUF2	'05160'	TAPMES	'07001'	QUEU N	'07645'
ATT*ORD	'03555'	IMCS	'05233'	FLEXTB	'07005'	QUE*OUT	'07646'
LAS*URN	'03556'	RENES	'05237'	INITPU	'07024'	DRD*NOF	'07647'
DRBUSY	'03557'	INITRE	'05243'	LOOP1	'07047'	DRD*NCE	'07650'
DROK	'03560'	LOOP1	'05264'	LOOP2	'07060'	PRBUSY	'07651'
DRLF	'03561'	LOOP2	'05276'	SIGNON	'07072'	PRPARF	'07652'
ACTIVE	'03566'	LOOP3	'05311'	SIG*OFF	'07124'	PRB*NOF	'07653'
DRNBK	'03606'	REHEP	'05323'	FLDLNK	'07143'	PRB*NCE	'07654'
DRNES	'03611'	REHEP1	'05342'	AFXP6	'07150'	PRSS	'07655'
WAMES	'03615'	ANA*YZE	'05350'	PUHEP	'07155'	HEAD	'07656'
INTDR	'03622'	INO*IRE	'05356'	PUHEP1	'07172'	LINE1	'07664'
HOKDP	'03632'	WAIT	'05366'	PUHEP2	'07173'	PRPCS	'07664'

IV

LINE2	'07733'	ABS°IXT	'13561'	LVAR	'14750'	SKI°BUF	'16221'
PRPOS2	'07733'	FLOT	'13563'	BASE	'14756'	FRO°RUM	'16225'
LINE3	'10002'	PRO	'13565'	DPO	'14757'	END°VER	'16246'
PRPOS3	'10002'	AMS	'13573'	BEG°RAM	'14760'	PRO°SS7	'16251'
CRBUF1	'10052'	PRNTIS	'13605'	END°RAM	'14761'	DR°OCR	'16251'
CRBUF2	'10454'	DER°ORM	'13622'	REL°RSS	'14762'	NOR°ESH	'16273'
CRBUF3	'11056'	ERRORM	'13623'	BEG°RAY	'14763'	RUNVAR	'16300'
PRBUF1	'11460'	FVOC	'13711'	BEG°AFE	'14764'	STOCK	'16310'
PRBUF2	'12061'	ELOOP0	'13717'	SHI°AFE	'14765'	STOCK1	'16301'
NBKMS	'12461'	ELOOP1	'13722'	WOR°AFE	'14766'	STOCK2	'16302'
PARMS	'12465'	HERE	'13743'	PNTR	'14767'	STOCK3	'16303'
TORNMS	'12472'	NXT°ESL	'13743'	INS°NTR	'14770'	BEG°BJP	'16304'
YOKMS	'12476'	DECIDE	'13754'	COR°TIE	'14771'	END°BJP	'16305'
PRNTMS	'12502'	EXIT	'13757'	USER	'14775'	LOO°ING	'16306'
CONTAB	'12507'	SPE°IAL	'13764'	D24°D22	'14776'	REF	'16307'
PRNTAB	'12707'	PARITY	'14001'	LABEL	'14777'	REF1	'16310'
INI°PR1	'12726'	RCLN	'14007'	SUB°TOR	'15001'	SRT	'16311'
LOOP1	'12744'	S119	'14024'	NPR	'15011'	NBR	'16312'
LOOP2	'12754'	BAT	'14026'	LOOP	'15067'	WS	'16313'
LOOP3	'12771'	UL	'14030'	END	'15111'	WS1	'16314'
INI°PR2	'13005'	LC	'14032'	TRANSF	'15117'	ACTION	'16315'
INI°PR3	'13040'	UC	'14033'	ILR	'15120'	FRE°VO2	'16316'
PRJ°OFF	'13045'	NONFLX	'14035'	NEX°URE	'15160'	SCH°TEN	'16405'
PRHEX	'13051'	TRA°SIT	'14037'	NXTTP	'15215'	HALF	'16407'
PRHEX1	'13055'	TLXREP	'14044'	UPD°ARY	'15224'	ERR°BLE	'16411'
BUF	'13064'	TAB	'14104'	INF°HER	'15233'	END°BLE	'16430'
SECOND	'13102'	TAB3	'14107'	PRIDF	'15235'	ENTRIS	'16436'
SHFTAB	'13111'	SPACE	'14116'	PRINT	'15250'	ENT°IS4	'16442'
NEW°AGE	'13115'	AFXT6	'14123'	PRJ°ORD	'15253'	EXITIS	'16445'
NLCR	'13117'	AFXT	'14124'	ELOOP1	'15257'	ENTRPS	'16450'
CAR°AGE	'13121'	RECODE	'14130'	MAK°ERS	'15265'	ENTRB	'16451'
CAR°GE1	'13127'	CRDAR	'14244'	LOOP	'15273'	DPTR	'16454'
CAR°GE2	'13152'	PCAR	'14214'	ASS°ARY	'15325'	DOUBLE	'16461'
CAR°GE3	'13164'	°° 3342 °0		SYSTEM	'15334'	TEST	'16464'
OFF°NE1	'13177'	CRDSS	'14150'	BCHECK	'15334'	END	'16472'
OFF°INE	'13202'	BUF°RM1	'14151'	CON°DER	'15335'	EXITP	'16474'
LOOP	'13222'	PCSS	'14153'	OBS°RVE	'15342'	CEN	'16507'
HEA°ING	'13234'	BUF°R39	'14155'	ERM	'15346'	FORMAL	'16537'
NXTDEC	'13254'	CASE	'14156'	DANGER	'15350'	MASK0	'16545'
GETBUF	'13266'	POI°TER	'14157'	TRAFO	'15354'	MASK19	'16570'
TODRUM	'13305'	COU°TER	'14160'	BTS°AM9	'15503'	TAR	'16571'
STA°TIN	'13315'	LAST	'14161'	QUEU7	'15504'	TAKE	'16600'
PRO°SS0	'13320'	WRONG	'14162'	CRB°OF7	'15505'	ADD	'16601'
CRTODR	'13320'	INT°BLE	'14163'	CRB°OE7	'15506'	TAST	'16602'
FRO°RUM	'13351'	INSTR	'14203'	DRR°OM7	'15507'	TAA	'16602'
STA°OUT	'13362'	CRD°DER	'14206'	RECPOS	'15510'	TAI	'16605'
PRO°SS1	'13365'	LOOP	'14227'	BUFEND	'15511'	TABO	'16610'
DRTOCR	'13365'	SELECT	'14245'	STR°ORD	'15512'	CLPN	'16613'
STA°TPR	'13417'	RESULT	'14250'	CRBUF1	'15514'	SIMVAR	'16631'
STA°PR1	'13435'	NLCR	'14271'	CRBUF2	'15715'	LIST	'16642'
STA°PR2	'13436'	STA°TPC	'14306'	INI°EAM	'16115'	STAI	'16652'
STA°PR3	'13440'	EIGHT	'14313'	LOOP	'16127'	STABO	'16657'
LOOP	'13450'	NOK	'14323'	JUS°INE	'16135'	STAS	'16663'
PRI°TER	'13466'	I START	'14332'	ASS°NCE	'16142'	STAR	'16667'
PRNOK	'13504'	PUN°HER	'14340'	NEX°ORD	'16147'	MASK	'16672'
LOOP	'13516'	STA°CRD	'14351'	GET°ORD	'16154'	CRV	'16673'
TESTRN	'13534'	FREVO	'14356'	STI°EEN	'16170'	CRV3	'16676'
KEO°FER	'13545'	FIR°MES	'14745'	BIT°H18	'16176'	CIV	'16703'
PRINT	'13553'	HMODE	'14746'	FET°AST	'16207'	CIV5	'16710'
FIXT	'13555'	CMODE	'14747'	BIT°M27	'16213'	CBV	'16715'

CLV	'16722'	JUA	'17372'	PUN°CRL	'17722'	CYCLE	'20202'
IAD	'16727'	REJST	'17405'	PUS°CEL	'17723'	ENTIER	'20207'
BAD	'16732'	TIAV	'17412'	RUN°UTL	'17724'	ENT°ER1	'20210'
RAD	'16735'	TAV	'17414'	REHEPL	'17725'	SQRT	'20222'
JOI°TAD	'16737'	TAV1	'17415'	PUHEPL	'17726'	D16	'20276'
LOOP	'16753'	LOOP	'17433'	HANDL	'17727'	C0	'20277'
RAD35	'17000'	LOOP1	'17471'	XEENL	'17730'	LN	'20303'
LOOP1	'17001'	IDI	'17510'	STOPL	'17731'	CYCLE	'20326'
LOOP2	'17033'	TTP	'17533'	FRO°UML	'17733'	LIST	'20365'
LAD	'17044'	ODD°EST	'17555'	TOD°UML	'17735'	LNx	'20373'
AGAIN	'17047'	LNN°EXP	'17557'	TIMEL	'17737'	LNy	'20375'
IND	'17056'	END	'17563'	SSL	'17741'	C1	'20377'
LOOP	'17074'	MUL	'17565'	MML	'17743'	C3	'20401'
INDB	'17115'	CYCLE	'17567'	TML	'17745'	C5	'20403'
INDU	'17131'	END1	'17602'	MTL	'17747'	LN2	'20405'
SWITCH	'17144'	RUN	'17604'	MVL	'17751'	ONE	'20407'
TFSU	'17152'	TCST	'17624'	TVL	'17753'	GIANT	'20411'
R	'17156'	STST	'17626'	VVL	'17755'	BIN°P40	'20413'
I	'17157'	STSST	'17630'	TNRP	'17757'	EXP	'20414'
BO	'17160'	STSST4	'17630'	TRP	'17764'	EXP1	'20415'
ST	'17161'	CSTV	'17632'	ABSP	'17770'	ENTIRE	'20445'
L	'17162'	STAD	'17634'	SIGNP	'17774'	C0	'20462'
TSL	'17163'	ORAD	'17636'	SQRTP	'20001'	C1	'20464'
TFSL	'17174'	OIAD	'17636'	SINP	'20003'	C2	'20466'
TFSL1	'17175'	OBAD	'17636'	COSP	'20005'	C3	'20470'
TASR	'17205'	OSTAD	'17636'	ARC°ANP	'20007'	C4	'20472'
TASI	'17212'	EXITPC	'17640'	LNP	'20011'	C5	'20474'
TASB	'17217'	FRE°AIN	'17642'	EXPP	'20013'	C6	'20476'
TASB2	'17221'	DEC°ASE	'17644'	ENT°ERP	'20015'	C7	'20500'
TAGST	'17224'	DEC°SE3	'17645'	FIXPP	'20017'	LOGE	'20502'
TASU	'17227'	LIB°ARY	'17647'	ABS°XPP	'20023'	OMEGA	'20504'
STSR	'17235'	FIXTL	'17647'	FLOPP	'20027'	COS	'20506'
STSR4	'17241'	ABS°XTL	'17651'	PUNCHP	'20033'	SIN	'20510'
STSI	'17244'	FLOTL	'17653'	PUNLCR	'20035'	COS°OEF	'20547'
SSTSI	'17252'	PRINTL	'17655'	PUS°CEP	'20037'	SIN°OEF	'20563'
STSB	'17261'	NLCRL	'17657'	PUS°ACE	'20040'	TWO°RPI	'20577'
STSB3	'17264'	TABL	'17661'	RUNOUT	'20046'	ARCTAN	'20601'
STFSU	'17272'	SPACE	'17663'	REHEPG	'20054'	TOG°HER	'20630'
R	'17300'	PRI°XTL	'17665'	PUHEPP	'20057'	C1	'20650'
I	'17301'	CAR°GEL	'17667'	PUHEPG	'20060'	C2	'20652'
BO	'17302'	NEW°GEL	'17671'	HANDP	'20064'	C3	'20654'
ST	'17303'	LIN°ERL	'17673'	XEENP	'20066'	C4	'20656'
L	'17304'	RESYML	'17675'	PUTEXT	'20070'	C5	'20660'
INT°GER	'17305'	PUSYML	'17677'	NEW°ORD	'20105'	C6	'20662'
FAD	'17312'	PRSYML	'17701'	LOOP	'20113'	TG15	'20664'
STASR	'17320'	PUT°XTL	'17703'	CAL°CST	'20124'	TG30	'20666'
STASI	'17323'	ABSL	'17704'	FIXTP	'20125'	PIO°ERG	'20670'
STASB	'17326'	SIGNL	'17705'	ABS°XTP	'20131'	TODRUM	'20672'
STASST	'17331'	SQRTL	'17706'	FLOTP	'20135'	FRO°RUM	'20674'
STASU	'17334'	SINL	'17707'	PRINTP	'20141'	TIME	'20752'
TRSCV	'17337'	COSL	'17710'	SPACEP	'20143'	MATMAT	'20755'
TISCV	'17343'	ARC°ANL	'17711'	PRI°EXT	'20145'	BKJ	'20757'
TSCVU	'17347'	LNL	'17712'	CAR°GFP	'20147'	MAT°AT8	'20765'
R	'17354'	EXPL	'17713'	LIN°ERP	'20151'	MAT°AT9	'20766'
I	'17355'	ENT°ERL	'17714'	RESYII	'20153'	TAMMAT	'20771'
BO	'17356'	READL	'17715'	PUSYM	'20164'	MATTAM	'20774'
ST	'17357'	FIXPL	'17716'	PRSYM	'20171'	BJK	'20776'
L	'17360'	ABS°XPL	'17717'	POL°NFF	'20176'	MATVEC	'21005'
FADCV	'17361'	FLOPL	'17720'	POLINF	'20177'	BK	'21007'
RND	'17365'	PUNCHL	'17721'	START	'20201'	TAMVEC	'21015'

VI

VECVEC	'21020'	WOR°AFC	'22141'	COD°ODY	'22235'	UNGNOP	'23172'
PLI°T22	'21036'	LET°BOL	'22142'	POI°TER	'22236'	RDIDF	'23246'
PLI°222	'21040'	DIG°BCL	'22143'	END°RAY	'22237'	NEXTC	'23261'
PLI°223	'21041'	OWN°YPE	'22144'	NLP	'22240'	ELSEC	'23300'
PLI°T21	'21052'	TYPE	'22145'	BEG°LER	'22241'	ELSE1	'23347'
PLI°T11	'21055'	CHA°TER	'22146'	D16M1	'22241'	RDIDF1	'23311'
PLIST2	'21061'	VAL°HAR	'22147'	D18°IN1	'22242'	NXTPNR	'23316'
PLI°T25	'21066'	ARR°CLS	'22150'	D18	'22243'	LOOKUP	'23325'
PLIST1	'21113'	TPD°CLS	'22151'	D19	'22244'	NEXTC	'23331'
TES°DIM	'21121'	ARR°HCR	'22152'	D20	'22245'	NEXT1	'23335'
IND1	'21130'	REA°BER	'22153'	D21	'22246'	NEXT2	'23346'
IND2	'21140'	SMALL	'22154'	D21°D20	'22247'	NAM°ARY	'23351'
AIK	'21156'	LAS°NLP	'22155'	D22	'22250'	INN°LST	'23362'
AKI	'21166'	WOR°UNT	'22156'	D22°US1	'22251'	NXTIDF	'23407'
INPR	'21176'	INF°IST	'22157'	D24	'22252'	SKPIDF	'23414'
RR	'21215'	INA°DEC	'22160'	D25	'22253'	SKP°DEC	'23421'
RI	'21225'	INT°ELS	'22161'	FRAME	'22254'	SKP°ALI	'23426'
AI	'21235'	INTLAB	'22162'	LOC°BER	'22255'	SKP°PLI	'23435'
IR	'21237'	OLDBCP	'22163'	BASE0	'22256'	DIS°LVL	'23441'
II	'21247'	DSPLVL	'22164'	BASE1	'22261'	TPO°DSP	'23444'
STR°BOL	'21257'	GLO°UNT	'22165'	BASE2	'22264'	LOCSPC	'23447'
CAL°CST	'21317'	ARR°TER	'22166'	MASK	'22267'	PRCLVL	'23452'
DUMP	'21320'	DIM°ION	'22167'	WOR°DEL	'22301'	USECST	'23454'
BTS°RM9	'21320'	SUB°UNT	'22170'	NXTSBL	'22331'	STATUS	'23457'
DRB°OF8	'21321'	FORCNT	'22171'	SKIP0	'22342'	INCDE	'23463'
DRR°OM8	'21322'	NXTBCP	'22172'	ELSE0	'22346'	ENT°BLK	'23466'
QUEU8	'21323'	STATE	'22173'	SKIP1	'22351'	LOCLAB	'23475'
CRB°OF8	'21324'	STACK0	'22174'	ELSE1	'22360'	EXI°BLK	'23502'
CRB°OE8	'21325'	STACK1	'22175'	SKIP2	'22370'	NON°LAB	'23507'
DUPOS	'21326'	STACK2	'22176'	END	'22406'	COR°BCP	'23513'
BUFND	'21327'	MCR	'22177'	TES°DPO	'22424'	SKI°ING	'23530'
STRWRD	'21330'	PAR	'22200'	TES°ERS	'22424'	SKI°TAT	'23540'
CRBUF1	'21332'	NBR	'22201'	LOOP	'22451'	ARUNVA	'23561'
CRBUF2	'21533'	PAR°TER	'22202'	STA°EST	'22460'	RUNVA	'23562'
DUM°OFF	'21733'	NUMBER	'22203'	NON°FT	'22466'	NEX°TRY	'23570'
INI°TRM	'21736'	STCNT	'22204'	NXT°SBL	'22470'	INI°TER	'23576'
LOOP	'21761'	MAX°PTH	'22205'	INSBL	'22503'	INIT	'23601'
INI°POS	'21766'	MAXDI	'22206'	ELSE0	'22521'	PROGRAM	'23624'
STO°ING	'21771'	MAXDL	'22207'	ELSE1	'22527'	PRO°AM0	'23627'
NEX°WRD	'21772'	MAXPRL	'22210'	NEXT0	'22542'	PRO°AM1	'23637'
EMP°ORD	'22003'	RETMD	'22211'	ELSE2	'22560'	PRO°AM2	'23641'
NIN°ORD	'22011'	RETLVL	'22212'	ELSE3	'22565'	PRO°AM3	'23651'
BTS°M18	'22017'	ECNT	'22213'	NEXT1	'22571'	PRO°AM4	'23654'
LAS°ITS	'22025'	IFS°FRB	'22214'	ELSE4	'22606'	PRO°AM5	'23655'
BTS°M27	'22030'	CNT°VAR	'22215'	ELSE5	'22627'	BLOCK	'23661'
GETBUF	'22035'	L0	'22216'	NEXT2	'22643'	BLCK	'23716'
TODRUM	'22043'	L1	'22217'	NEXT3	'22653'	PROC	'23732'
CRTODR	'22064'	L2	'22220'	ELSE6	'22663'	NEXT0	'23744'
PRO°SS8	'22064'	L3	'22221'	UNDSBL	'22673'	SEM°EST	'23762'
FOR°ITS	'22106'	L4	'22222'	OUTSBL	'22677'	NEXT1	'23775'
ASK°END	'22115'	L5	'22223'	AROPLS	'22722'	NEXT2	'24013'
FIN°TRM	'22124'	CMPLTD	'22224'	INVERT	'22725'	SPE°EST	'24016'
END°UMP	'22130'	CMPLST	'22225'	REL°PLS	'22727'	NEXT3	'24020'
COM°VAR	'22133'	SWIDF	'22226'	BOO°PLS	'22732'	TEST	'24050'
STOCK	'22133'	NBR°SWE	'22227'	DECLLS	'22736'	BODY	'24060'
STOCK1	'22134'	SWLST	'22230'	SPECLS	'23035'	NEXT4	'24070'
QUO°TER	'22135'	INS°DEC	'22231'	OPLS	'23130'	ELSE0	'24074'
TEX°TER	'22136'	ADR°CST	'22232'	UNSINT	'23134'	END	'24104'
SHIFT	'22137'	LNC	'22233'	MUL°PLY	'23150'	NEXT5	'24132'
FIR°IFT	'22140'	LAS°LNC	'22234'	TENTH	'23171'	CMPTL	'24161'

VII

DEC°LST	'24167'	INSEPT	'25371'	SCA°ODE	'26515'	AR	'30067'
NEXT1	'24172'	FCTDES	'25372'	ASK°BR	'26531'	UN	'30071'
STR°EST	'24213'	PARLST	'25402'	NEXT0	'26537'	PRE°ARE	'30074'
ELSE0	'24220'	ACTPAR	'25412'	END	'26556'	INASH	'30121'
NEXT2	'24224'	PRC°TAT	'25413'	PRE°AN1	'26604'	REASH	'30165'
NEXT3	'24253'	STA°MNT	'25421'	ARI°EXP	'26633'	BOASN	'30214'
NEXT4	'24306'	ELSE0	'25433'	IFC°USE	'26652'	STASN	'30237'
ELSE1	'24327'	ELSE1	'25442'	FUTURE	'26662'	ARASN	'30261'
NEXT5	'24343'	GTSTAT	'25461'	SAR°EXP	'26673'	UNASN	'30313'
ELSE2	'24360'	CMPTL	'25471'	NXT°ERM	'26704'	FCTDES	'30343'
END	'24375'	IFC°USE	'25503'	TERM	'26715'	PRSTAT	'30351'
STA°MNT	'24405'	FOR°TAT	'25520'	NXT°CTR	'26716'	PRCALL	'30365'
NEXT1	'24412'	NEXT0	'25531'	FACTOR	'26733'	ORDNUM	'30402'
ELSE0	'24422'	ELSE0	'25545'	NXT°RIM	'26734'	PAR°IST	'30421'
ELSE1	'24432'	SWDECL	'25563'	PRI°ARY	'26746'	PAR°ST0	'30435'
ELSE2	'24442'	SWLIST	'25576'	REQ°OSE	'26764'	PAR°ST1	'30472'
ELSE3	'24445'	ARRDEC	'25606'	ARNAME	'26777'	PAR°ST2	'30510'
END	'24454'	NEXT0	'25613'	MCRDOS	'27011'	PAR°ST3	'30514'
BEG°TAT	'24456'	NEXT1	'25636'	MCRRET	'27016'	PAR°ST4	'30522'
LABDEC	'24466'	ELSE0	'25651'	RETURN	'27017'	PAR°ST5	'30543'
ILA°DEC	'24506'	BND°LST	'25655'	SUB°VAR	'27021'	PAR°ST6	'30550'
STN°MCS	'24521'	PRCDEC	'25667'	ADR°SCR	'27034'	PAR°ST7	'30553'
IDF	'24541'	NEXT0	'25676'	EVALON	'27054'	PAR°ST8	'30561'
PRC°IDF	'24542'	ELSE0	'25717'	SBSLST	'27100'	PAR°ST9	'30567'
TRU°ATE	'24552'	END	'25746'	REQBUS	'27112'	PAR°T10	'30573'
ELSE0	'24604'	BLOCK	'25750'	BOO°EXP	'27117'	PAR°T11	'30574'
ENDSBL	'24623'	DEC°LST	'25755'	SBO°EXP	'27136'	PAR°T12	'30606'
ELSE1	'24624'	END	'25767'	NXTIMP	'27137'	PAR°T13	'30614'
END	'24637'	PRO°RAM	'25775'	IMPL	'27151'	ACTPAR	'30634'
CLE°TAB	'24647'	LAB°TAT	'26017'	NXT°TRM	'27152'	ACT°AR0	'30670'
LOOP	'24650'	ILA°TAT	'26022'	BLTRM	'27164'	ACT°AR1	'30674'
PRE°ANO	'24656'	LABDEC	'26025'	NXT°LFC	'27165'	ACT°AR2	'30702'
NORMAL	'24677'	ADD°IDF	'26056'	BLFAC	'27177'	ACT°AR3	'30767'
LOOP	'24751'	FOR°ALS	'26102'	NXT°LSC	'27200'	ACT°AR4	'30772'
ARI°EXP	'24762'	TEST	'26117'	BLSEC	'27212'	ACT°AR5	'31004'
SAR°EXP	'24762'	FOR°ARS	'26145'	BLPRIM	'27221'	ACT°AR6	'31015'
ELSE0	'24774'	LOCALS	'26156'	RELATN	'27263'	ACT°AR7	'31021'
END	'25007'	ELSE0	'26207'	RST°FRL	'27276'	ACT°AR8	'31025'
SUB°VAR	'25012'	END	'26227'	BLP°RST	'27303'	ACT°AR9	'31041'
SBSLST	'25022'	STC°DDR	'26240'	BLNAME	'27314'	ACT°R10	'31045'
BOO°EXP	'25035'	FOR°ARS	'26242'	ARB°EXP	'27326'	ACT°R11	'31064'
SBO°EXP	'25041'	LOCALS	'26254'	SAB°EXP	'27377'	ACT°R12	'31071'
ELSE0	'25056'	ADD°YPE	'26310'	RST°FAE	'27445'	ACT°R13	'31075'
ELSE1	'25071'	ELSE0	'26330'	RST°FBE	'27450'	ACT°R14	'31101'
STREXP	'25102'	END0	'26341'	RST°ARE	'27454'	ACT°R15	'31120'
SST°EXP	'25106'	END1	'26342'	ARB°RST	'27462'	ACT°R16	'31127'
DESEXP	'25127'	BOO°EAN	'26345'	STREXP	'27512'	STA°SR	'31133'
SDE°EXP	'25133'	STRING	'26347'	SST°EXP	'27531'	NUM°SCR	'31141'
EXP	'25153'	ARM°TIC	'26351'	STNAME	'27557'	PRCSOP	'31153'
TYP°EXP	'25170'	ARBOST	'26353'	DESEXP	'27572'	NON°SBL	'31203'
SEXP	'25207'	DES°NAL	'26355'	SDE°EXP	'27611'	LINE	'31211'
ELSE0	'25235'	ASSTO	'26375'	DSNAME	'27635'	LINE1	'31213'
ELSE1	'25246'	SBSVAR	'26411'	ARD°EXP	'27647'	STA°MNT	'31222'
RST°EXP	'25266'	PROC	'26425'	NDSEXP	'27663'	UNC°TAT	'31224'
ASS°TAT	'25302'	FNCTN	'26440'	EXP	'27676'	STAT	'31226'
RMSIDE	'25306'	LSTLTH	'26442'	SEXP	'27712'	GOT°TAT	'31304'
ELSE0	'25333'	SWLTH	'26455'	EXPRST	'30002'	CMPCMB	'31320'
END0	'25356'	ADDRSS	'26460'	ASN°TAT	'30042'	CMPCIN1	'31321'
ELSE1	'25360'	CHKDIM	'26467'	DST°YPE	'30047'	CMPTL3	'31322'
END1	'25366'	IDF	'26475'	LIST	'30057'	CMPTL	'31322'

VIII

IFSTAT	'31336'	STATE0	'32742'	OUTDEC	'33430'	CHAIN	'17622'
FOR*AT	'31360'	END	'32753'	ASS*OFD	'33432'	PSE*VAR	'72167'
STO*EPR	'31423'	STATE1	'32756'	DEC*FED	'33435'	INC*EIC	'72176'
STO*MCR	'31434'	STATE2	'32765'	SUPLOC	'33440'	VALDP1	'72201'
TAK*MCR	'31454'	STACK	'32772'	LOCPOS	'33442'	VAL*NPR	'72205'
FORLST	'31462'	LOAD	'32776'	SET*DEC	'33447'	VAL*DPR	'72211'
FOR*ST0	'31536'	OPT*IZE	'33003'	MRKPQS	'33456'	VAL*ESS	'72215'
FOR*ST1	'31545'	UNLOAD	'33017'	PRADR	'33471'	SUB*R18	'72220'
FOR*ST2	'31576'	PRO*UCE	'33023'	ADDRSS	'33512'	SUB*AR9	'72223'
FOR*ST3	'31613'	NORMAL	'33041'	LSTLTH	'33534'	MCRLST	'72230'
FOR*ST4	'31711'	END	'33052'	TESTFC	'33540'	STACK	'72230'
SWDEC	'31715'	PRC*STP	'33056'	TES*RET	'33544'	NEG	'72231'
SWDEC0	'31733'	REA*T11	'33070'	CHC*DIM	'33546'	EMPTY	'72343'
SWDEC1	'31753'	REA*T15	'33102'	CHC*RET	'33550'	TBC	'72361'
SWDEC2	'31756'	REA*T12	'33111'	CHCKLL	'33562'	TLV	'72363'
SWDEC3	'31770'	REA*T10	'33123'	CHCKTP	'33570'	STST	'72372'
SWDEC4	'32000'	REA*T13	'33126'	NBRLL	'33610'	JU	'72376'
SWDEC5	'32005'	REA*T14	'33150'	NEXTLL	'33613'	SUBJ	'72404'
SWDEC6	'32030'	PRC*PAR	'33161'	NXTFI	'33617'	NIL	'72416'
SWORD	'32044'	KIND2	'33167'	INC*STS	'33627'	LAST	'72417'
SWORD1	'32045'	KIND3	'33200'	IDF	'33631'	TAA	'72423'
SWORD2	'32046'	KIND1	'33203'	SKP*RLI	'33633'	SWP	'72424'
ARRDEC	'32047'	KIND0	'33213'	TRLCD	'33646'	EXITSV	'72427'
BND*LST	'32116'	END*HNO	'33223'	TRLCD1	'33656'	CODE	'72430'
PRDEC	'32137'	LENTY	'33227'	TRLCD1	'33731'	TABEL	'72454'
PRDEC0	'32171'	SAT	'33232'	TRLCD2	'33733'	FUN*LTR	'15354'
PRDEC1	'32204'	OPTOP	'33234'	TRLCD3	'33742'	FUN*DIR	'15355'
PRDEC2	'32207'	OPTNBR	'33236'	UNSNM	'33745'	CVAR	'15356'
PRDEC3	'32211'	BREACT	'33242'	ARCS*	'33765'	END*IST	'15357'
PRDEC4	'32225'	CHA*CTR	'33245'	CST*STR	'33776'	BEG*IST	'15360'
PRDEC5	'32241'	ARM*IC	'33252'	CST*TR0	'34000'	BEG*IST	'15361'
PRDEC6	'32274'	REAL	'33257'	CST*TR1	'34001'	L100P	'15362'
SAV*LNC	'32304'	INT*GER	'33262'	CST*TR2	'34006'	L100P6	'15363'
BLOCK	'32312'	BOO*EAN	'33267'	CST*TR3	'34017'	P1ROD	'15364'
DEC*CM8	'32333'	STRING	'33272'	CRF	'34026'	O18*ODE	'15365'
DEC*ST3	'32343'	DES*NAL	'33275'	LIB*OUT	'34040'	I1M*PAR	'15366'
DECLST	'32343'	NOTYPE	'33275'	FIL*BLE	'34046'	D1YPAR	'15367'
DEC*ST0	'32345'	AROST	'33300'	NP	'34063'	P1U*PAR	'15370'
DEC*ST1	'32364'	UNK*OWN	'33303'	CRSS	'34070'	L1E*TRY	'15371'
DEC*ST2	'32374'	NONAR	'33306'	CYCLE	'34077'	D1E*ECT	'15372'
INS*ECL	'32412'	NONRE	'33315'	END	'34110'	I1M*ORD	'15373'
LOOP	'32461'	NONIN	'33320'	PSE*VAR	'34112'	P1U*ORD	'15374'
LOOP1	'32505'	NONBL	'33323'	LOOP	'34122'	N1O*CRO	'15375'
PRO*END	'32524'	NONSTR	'33327'	END	'34137'	S1U*TIT	'15376'
END0	'32530'	NONDES	'33331'	TRANSL	'34142'	P1S*VAR	'15377'
END1	'32531'	SIMPLE	'33334'	ENDCAT	'34207'	I1N*EIC	'15400'
LABLST	'32571'	SIM*LE1	'33341'	LOOP	'72002'	V1A*DP0	'15401'
PRO*AM3	'32620'	SUBSCR	'33345'	LOOP6	'72010'	V1A*NPR	'15402'
PRO*RAM	'32620'	PROC	'33350'	PROD	'72031'	V1A*DPR	'15403'
PRO*CM8	'32634'	FUNCTN	'33355'	NEXT	'72042'	V1A*ESS	'15404'
PRO*CM2	'32651'	NON*MPL	'33360'	OBJ*ODE	'72055'	S1U*R18	'15405'
LABDEC	'32655'	NON*UBS	'33370'	IMPPAR	'72061'	S1U*AR9	'15406'
SUBSTT	'32670'	NON*ROC	'33373'	DYPAR	'72064'	INS*HBR	'15407'
SUBST2	'32703'	NONFCT	'33402'	PURPAR	'72077'	CLEAR	'15411'
ORDCNT	'32711'	FORMAL	'33405'	LENTY	'72104'	PAR*ART	'15413'
MACRO	'32715'	INVALI	'33411'	DIRECT	'72111'	INS*PRT	'15416'
MACRO2	'32715'	ASSTO	'33414'	IMP*ORD	'72130'	KIND	'15422'
MACRO3	'32717'	DYN*HIC	'33417'	PUR*ORD	'72136'	FRE*WO1	'15425'
MACRO4	'32726'	INLIBR	'33422'	NOT*CRO	'72141'	STIAR	'00124'
STATE3	'32735'	OPLIKE	'33425'	SUB*TIT	'72144'	STPAR	'00210'

IX

STISS	'00000'	BEGCAT	'36000'	'142000'
STPSS	'00000'	END*GUE	'35161'	'141161'
SPBUF	'34207'			
SPAR	'34210'			
READD	'34211'	LOOP	'72002'	'116553'
SKPBLK	'34216'	LOOP6	'72010'	'116561'
SNXTVD	'34225'	PROD	'72031'	'116602'
SCW	'34231'	NEXT	'72042'	'116613'
SREHEP	'34232'	OBJ*ODE	'72055'	'116626'
SREWRD	'34243'	IMPPAR	'72061'	'116632'
ENDRBI	'34257'	DYPAR	'72064'	'116635'
SYSTAP	'34260'	PURPAR	'72077'	'116650'
SIPINT	'34373'	LENTY	'72104'	'116655'
INTINS	'34374'	DIRECT	'72111'	'116662'
WPINS	'34375'	IMP*ORD	'72130'	'116701'
WAINS	'34376'	PUR*ORD	'72136'	'116707'
KEYINS	'34377'	NOT*CRO	'72141'	'116712'
CWP1	'34400'	SUB*TIT	'72144'	'116715'
SIPCW1	'34401'	CHAIN	'116722'	'71151'
SIPCW2	'34402'	PSE*VAR	'72167'	'116740'
SPHBLK	'34403'	INC*EIC	'72176'	'116747'
SPBLOK	'34414'	VALDPO	'72201'	'116752'
SPWORD	'34423'	VAL*NPR	'72205'	'116756'
SPBLNK	'34441'	VAL*DPR	'72211'	'116762'
SPUHEP	'34447'	VAL*ESS	'72215'	'116766'
ENDSYS	'34460'	SUB*R18	'72220'	'116771'
		SUB*AR9	'72223'	'116774'
		MCRLST	'72230'	'117001'
		STACK	'72230'	'117001'
		NEG	'72231'	'117002'
		EMPTY	'72343'	'117114'
		TBC	'72361'	'117132'
		TLV	'72363'	'117134'
		STST	'72372'	'117143'
		JU	'72376'	'117147'
		SUBJ	'72404'	'117155'
		NIL	'72416'	'117167'
		LAST	'72417'	'117170'
		TAA	'72423'	'117174'
		SWP	'72424'	'117175'
		EXITSV	'72427'	'117200'
		CODE	'72430'	'117201'
		TABEL	'72454'	'117225'

"ASSEMBLY PARAMETERS 28/3/70

```

M[0]:          CRE:
M[5]:          RE:          REST:
M[8]:          TY:
M[9]:          KY:
M[11]:         PR:
M[18]:         PU:          PUST:
M[19]:         PC:
M[20]:         CPU:
M[25]:         CRD:
M[27]:         DR:

M[128]:        IBULEN:      RECBULEN:
M[1504]:       MAX4:
M[42]:         BFL:
M[9]:          ITIM:
M[128]:        PBULEN:
M[320]:        MAX2:
M[256]:        BUFLTH:
M[512]:        MAX:

M[176]:        INSTCK:
M[256]:        STK BEG:
M[511]:        STK END:
M['4000']:     DELTA1:

M['14745']:    SUBMOVAR:
SUBMOVAR[350]: RECCVER:
RECCVER[450]: RUNSYST:
M['36000']:    BEGCAT:
BEGCAT[-399]:  END OF PERMANENT CATALOGUE:

M[((( '124551' - '6000' ) - '72000' ))]: DELTA:
M[((( '124551' - '6000' ) -:DELTA))]:     BEG UP 32 K:
BEG UP 32 K[380]:                          INSTR LST:
INSTR LST[209]:                          BEG TRAF0 TAB:
BEG TRAF0 TAB[48]:                       END UP 32 K:

```

" INTERFACE CONSTANTS

```

M[512]:
'BEGIN'
" DRUM DIVIDING CONSTANTS
PRESENCE:          -1
COMP DIR:          '001 000 000'          "D18
DR FICORIMBEG:     '000 000 001'          "DRUM FIXED CORE/COMPILERIMAGE BEGIN
DR CATSPIMEND:     (((:BEGCAT + :DELTA1) -:RUNSYST)+2) "DRUM CATALOGUE SPACE IMAGE END
DR LIBR BEG:       (((:BEGCAT + :DELTA1) -:RUNSYST)+3) "DRUM LIBRARY BEGIN
DR LIBR END:       '000 144 000'          " 50K
DR TRAPA BEG:      '000 144 000'          " DRUMTRANSFORMATOR PART BEGIN
DRLIBRTABEND:      ('000 146 400' + (:BEG TRAF0 TAB -:BEG UP32K)) "DRUM LIBRARY
DR BEG8:           ('000 146 400' + (:BEG TRAF0 TAB -:BEG UP32K))
DR END8:           '000 230 000'          " 76K
DR BEG4:           '000 230 000'          " 76K
DR END4:           '001 020 000'          " 264K
DR BEG2:           '001 020 000'          " 264K
DR END2:           '001 140 000'          " 304K

```

DR BEG:	'001 140 000'	" 304K
DR END:	'001 540 000'	" 432K
DMP BEG:	'001 540 000'	" 432K
DMP END:	'002 000 000'	" 512K

```

"CORE DIVIDING CONSTANTS (COMPILE-TIME)
BEGIN OF STACK:          :STK BEG
END OF STACK:            :STK END
NBR OF PERMANENT ROUTINES: +47
MAX NBR OF ROUTINES:     +256
CAT END:                 :END CAT
PERMANENT CATALOGUE END: (:END OF PERMANENT CATALOGUE +:DELTA1)
BEGIN OF CATALOGUE:      (:BEGCAT + :DELTA1)
START OF CONSTANT LIST:  ((:BEG CAT + :DELTA1) + 2)  "INITIAL VALUE OF DPO
START OF TEXT ARRAY:     ((:BEG CAT + :DELTA1) + 514)
BEGIN OF NL:             ((:BEGUP32K + :DELTA)-2)
CORRECTION:              (((:BEG UP32K + :DELTA)-2)-8191)
BEGIN OF UP32K:          (:BEG UP32K + :DELTA)
LENGTH OF COMPILER:      (((:BEGCAT + :DELTA1) -IRUNSYST) + 1)
                        (((:BEGCAT + :DELTA1) -IRUNSYST) + 1)

```

"CORE-DIVIDING CONSTANTS (COMPILE TIME AND RECOVER TIME)

```

TRAFO TAB BEG:
BEGIN OF TRAFOTABLE:     (:BEG TRAFOTAB + :DELTA)
INFO TAB END:            (((:BEG TRAFO TAB + :DELTA) + (6 * 128)) + 512)
LENGTH OF UP32K:        (((:BEGTRAFO TAB -:BEGUP32K) + (6 * 128)) + 512)
"CORE DIVIDING CONSTANTS (RECOVER AND RUNTIME)

```

```

BEGIN OF PRAR:           :DUMP
END OF COMPILER OUTPUT SPACE: + 32767
"CORE DIVIDING CONSTANTS (RUNTIME)

```

MEMCRY USABLE TO: '126111'

"MEMORY DIVIDING VARIABLES

BEGIN OF CROSSTABLE:	'SKIP' 1	
BEGIN OF INFOTABLE:	'SKIP' 1	
NBR OF ROUTINES:	'SKIP' 1	"
END OF CATALOGUE:	'SKIP' 1	") ORDER OBLIGATE
END OF TRAFOTABLE:	'SKIP' 1	") THIS SET OF VARIABLES
END OF CROSSTABLE:	'SKIP' 1	") SPECIFIES
END OF INFOTABLE:	'SKIP' 1	") STATE OF LIBRARY
END OF DRUMLIBRARY:	'SKIP' 1	")

"INTERFACE VARIABLES

ERRONEOUS:	'SKIP' 1	
DANGEROUS:	'SKIP' 1	
RUNNUMBER:	'SKIP' 1	
VALUE OF CONSTANT:	'SKIP' 2	
LINE COUNTER:	'SKIP' 1	
LAST SYMBOL:	'SKIP' 1	
LAST IDENTIFIER:	'SKIP' 2	
WANTED:	'SKIP' 1	
FLEXCODE:	'SKIP' 1	"TELEX OR FLEX?

PICO:

'BEGIN' KAR,DATE, SYM, TBUF, ASAFE, SSAFE, FSAFE, SAFEB,
 BBSAFE, KCASE, INSW, MAYI, XEENW, CASE, TEMP, CNT, BEXP,
 BSAFE, PT, ESIGN, STOCK, HEAD, TAIL, DEXP, MSIGN, MM,
 HEP, NN, XX, ABS, FIXFLO, LASTACTION, ANEW, LIMIT, CPLTIM,
 CPLCHK, ININTRPT, CRITICAL, KICKOFF, NOSWAP, ATLMSGN,
 ATLMSGN1, ATLMSGN2, DYST, KBUF, NTAB, HALF, SGNBIT,
 SIXMM1, TENTH, SIXMIL, SCHOLTEN, DATMES, SERMES, SMES,
 SMES1, RUNMES, XMES, HMES, CRNLMS, CLCKMS, PASTE,
 CLPASTE0, CLPASTE1, INTREP, TELXTB, TENPOW, SCALE, GIANT,
 NAIGA, FIXT60, BRUSH, INTAB, HOK, RESTORE, ENDCRIT,
 EXITNOSWAP, WASTE TIME, TYPE, TYPE3, TACT, TWAIT,
 TOFF, KEYBD, XEENS, XGET, NXTKYBDSL, NKS, DNAH, HANDRD,
 KYBD, FINIS1, NSTART, ESTART, KEYA, KEYT, L, KEYM,
 INOCT, CLOCK, COPY, DONT, KTAB, CW1, CW2, CW4, SUMCHECK,
 HEPcnt, DTS, INITPU, SIGNON, SIGNOFF, INITPR1,
 INITPR3

M[64+(4*:TY)]:	TAR:
M[64+(4*:KY)]:	KAR:
M[64+(4* 39)]:	CLAR:

PICO[0]:

SERIAL:	'SKIP'1
DATE:	'SKIP'1
TSS:	'SKIP'3
SYM:	'SKIP'1
TBUF:	'SKIP'20
ASAFE:	'SKIP'1
SSAFE:	'SKIP'1
FSAFE:	'SKIP'2
SAFEB:	'SKIP'1
BBSAFE:	'SKIP'1
KCASE:	'SKIP'1

IN SW: 'SKIP'1
MAYI: 'SKIP'1
XEENW: 'SKIP'1
CBASE: 'SKIP'24
CASE: 'SKIP'1
TEMP: CNT: BEXP: 'SKIP'1
BSAFE: PT: ESIGN: 'SKIP'1
STOCK: 'SKIP'1
HEAD: 'SKIP'1
TAIL: 'SKIP'1
DEXP: 'SKIP'1
MSIGN: MM: 'SKIP'1
HEP: NN: 'SKIP'1
XX: 'SKIP'2
ABS: 'SKIP'1
FIXFLO: 'SKIP'1
LAST ACTION: 'SKIP'1
ANew: 'SKIP'1
LIMIT: 'SKIP'1
CPLTIM: 'SKIP'1
CPLCHK: 'SKIP'1
ININTRPT: 'SKIP'1
CRITICAL: 'SKIP'1

KICKOFF: 'SKIP'1
 NOSWAP: 'SKIP'1
 ATLMSGN: 'SKIP'1
 ATLMSGN1: 'SKIP'1
 ATLMSGN2: 'SKIP'1

BEGIN OF PERMANENT:

DYST: SUBC(D18M1)
 D18: '001 000 000'
 D18M1: '000 777 777'
 D23M3: ('040 000 000' - 3)
 KBUF: +2 "CR
 +8 "NL
 'SKIP'2

NTAB: +9
 +99
 HALF: '200 000 000'
 SGNBIT: '400 000 000'
 SIXMM1: +6710885
 TENTH: '031 463 146'
 '146 314 632'
 SIXMIL: +6710887
 + 671089
 + 67109
 SCHOLTEN: + 12288
 + 0

DATMES:	'02 10 44 22 0'	"CR NL BLKLT D
	'30 01 20 77 0'	"A T E END
SERMES:	'02 10 44 24 0'	"CR NL BLKLT S
	'20 12 14 30 0'	"E R I A
	'11 77 00 00 0'	"L END
SMES:	'02 10 40 77 0'	"CR NL BLKDG END
SMES1:	'27 77 00 00 0'	" / END
TRNMES:	'02 10 44 01 0'	"CR NL BLKLT T
	'12 30 06 24 0'	"R A N S
	'11 30 01 14 0'	"L A T I
	'03 06 77 00 0'	"C N END
RUNMES:	'02 10 44 20 0'	"CR NL BLKLT E
	'27 20 16 34 0'	"X E C U
	'01 14 03 06 0'	"T I O N
	'77 00 00 00 0'	"END
XMES:	'02 10 44 27 0'	"CR NL BLKLT X
	'20 20 06 77 0'	"E E N END
HMES:	'02 10 44 05 0'	"NL CR BLKLT H
	'30 06 22 40 0'	"A N D BLKDG
	'77 00 00 00 0'	"END
CRNLMS:	'02 10 77 00 0'	"CR NL END
CLCKMS:	'02 10 44 12 0'	"CR NL BLKLT R
	'34 06 04 01 0'	"U N SP T
	'14 07 20 33 0'	"I M E DIG
	'04 77 00 00 0'	"SP END

PTRNMES:	'02 10 44 11 0'	" CR NL BLKLT L
	'14 23 12 30 0'	" I B R A
	'12 25 77 00 0'	" R Y END
CLCKMS1:	'02 10 44 11 0'	" CR NL BLKLT L
	'03 30 22 33 0'	" O A D DIG
	'04 77 00 00 0'	" SP END
FILLMES:	'04 04 44 31 0'	" SP SP BLKLT W
	'03 12 22 24 0'	" O R D S
	'04 03 06 04 0'	" SP C A SP
	'22 12 34 07 0'	" D R U M
	'04 77 00 00 0'	" SP END

```
INFCRM OPERATOR:      ('660 072 000' + :KY)
                        SUBCD(:CTYPE)
                        :SMES
                        G = DR LIBR END
                        G = END OF DRUMLIBRARY
                        SUBC(:FIXT60)
                        SUBCD(:CTYPE)
                        :FILLMES
                        SUBC(:UPDATE LIBRARY)
                        A = D18
                        COMDIR = A
                        PRESENCE = - A
                        ('760 072 000' + :KY)
                        GOTOR(MC[-1])
```

```

PASTE:      ('000 004 000'+:TY)      "TYPEWRITER
            ('004 004 000'+:KY)      "KEYBOARD
CLPASTED:   ('004 000 000'+ 39)      ") CLOCK
CLPASTE1:   ('000 000 000'+ 39)      ")

INTREP:     -'137 400 002'; + '001 247 401'; + '002 241 002'; - '154 400 000';
            + '004 243 004'; - '152 400 000'; - '151 400 000'; + '007 262 407';
            + '010 261 010'; - '146 400 000'; - '036 400 000'; - '035 400 002';
            - '014 400 000'; - '033 400 001'; - '012 400 005'; - '011 400 000';

            + '147 275 573'; - '027 400 000'; - '006 400 000'; + '121 241 403';
            - '055 400 000'; + '103 255 405'; + '124 262 006'; - '072 400 000';
            - '071 400 000'; + '127 276 411'; - '047 400 007'; - '066 400 000';
            - '115 400 001'; - '134 400 000'; - '113 400 000'; - '132 400 001';

            + '046 250 000'; - '110 400 000'; - '107 400 000'; + '051 234 035';
            - '000 000 000'; + '141 435 037'; + '142 435 440'; - '014 200 000';
            - '033 200 000'; + '165 437 043'; - '011 200 002'; - '030 200 000';
            - '027 200 001'; - '006 200 000'; - '056 200 000'; - '055 200 001';

            - '074 200 000'; + '124 445 110'; + '105 433 434'; - '071 200 000';
            + '127 434 436'; - '047 200 000'; - '066 200 000'; + '062 436 041';
            + '043 436 442'; - '113 200 000'; - '132 200 000'; + '046 474 131';
            - '110 200 000'; - '107 200 001'; - '126 200 006'; - '000 000 000';

            + '160 246 101'; - '077 400 000'; - '175 200 000'; + '023 430 025';
            - '000 000 000'; + '000 031 027'; + '004 431 430'; - '000 000 000';
            - '116 400 000'; + '000 033 033'; - '116 200 001'; - '000 000 000';
            - '077 200 001'; - '000 000 000'; - '000 000 000'; - '176 200 001';

            - '137 200 000'; + '000 027 023'; + '000 027 424'; - '000 000 000';
            + '000 030 426'; - '000 000 000'; - '000 000 000'; + '133 232 031';
            + '153 232 432'; - '104 400 000'; - '024 200 000'; + '025 475 127';
            - '000 000 000'; - '157 600 001'; - '000 000 001'; - '000 000 000';

            - '000 000 000'; + '000 022 412'; + '010 423 013'; - '146 200 000';
            + '026 424 015'; - '170 200 000'; - '000 000 000'; + '000 025 420';
            + '000 026 021'; - '000 000 000'; - '000 000 000'; + '000 076 130';
            - '000 000 000'; - '000 000 001'; - '000 000 001'; - '000 000 000';

            + '000 074 500'; - '000 000 000'; - '000 000 000'; + '000 023 414';
            - '000 000 000'; + '000 024 416'; + '076 025 017'; - '145 600 000';
            - '104 200 000'; + '160 426 422'; - '044 200 004'; - '000 000 000';
            - '000 000 003'; - '000 000 000'; - '161 400 000'; - '161 200 002';

```

TELXTB:

+13
+29
+25
+16
+10
+ 1
+21
+28
+12
+ 3
+17
+24
+ 7
+ 5
+ 4

" 0
" 1
" 2
" 3
" 4
" 5
" 6
" 7
" 8
" 9
" +
" -
" .
" "
" SPACE

TENPOW:

+4194 3040
+5242 8800
+6553 6000
+4096 0000
+5120 0000
+6400 0000
+4000 0000
+5000 0000
+6250 0000
+3906 2500

" 10+1 * 2+(26- 4)
" 10+2 * 2+(26- 7)
" 10+3 * 2+(26-10)
" 10+4 * 2+(26-14)
" 10+5 * 2+(26-17)
" 10+6 * 2+(26-20)
" 10+7 * 2+(26-24)
" 10+8 * 2+(26-27)
" 10+9 * 2+(26-30)
" 10+10* 2+(26-34)

SCALE:

+ 4
+ 7
+10
+14
+17
+20
+24
+27
+30
+34

GIANT:

'377737777'
'377777777'

```

DENTST:      B = BEGIN OF STACK      " INITIALIZE STACK POINTER
              '660 071 046'          " CLEAR INTERRUPT FROM NB
              G = PASTE[0]
              SUBC (:BRUSH)           " RESET TYPE WRITER
              G = PASTE[1]
              SUBC (:BRUSH)           " RESET KEYBOARD
              ININTRPT = - B          " NCT IN INTERRUPT PROGRAM
              CRITICAL = - B          " NC CRITICAL SEGMENT
              KICKOFF = - B           " NC KICKOFFS WAITING
              NOSWAP = - B            " SWAPPING ALLOWED
              SUBCD (:FOUND LIBRARY) " ASSURE LIBRARY TOO
              ('760 070 000' + :KY)   " ACCEPT KEYBOARD SYMBOLS
              '760 060 000'          " SET INTERRUPT PERMIT
              '760 061 000'          " ALLOW WRONG ADDRESS INTERRUPTS

              F = :TSS[1]
              TAR[0] = G              " OK INDICATION
              F = 0
              TAR[1] = G              " INTERRUPT COUNTER = 0
              KCASE = - G             " FIGURE SHIFT
              KAR[0] = - G            " NC SYMBOL IN KEYBOARD YET
              KAR[1] = B              " INTERRUPT COUNTER = 0
              ATLMSGN1 = G            " NO TIME LOCK/COMPILE LINE NUMBERS
              CLAR[2] = G             " NC SPECIAL CLOCK INTERRUPTS

              SUBCD (:CTYPE)          " TYPE "DATE"
              :DATMES
              SUBCD (:DNAH)           " GET DATE FROM KEYBOARD
              DATE = G
              SUBCD (:CTYPE)          " TYPE "SERIAL"
              :SERMES
              SUBCD (:DNAH)           " GET SERIAL NUMBER
              G = 1
              SERIAL = G
              SUBC (:INITDR)          " INITIALIZE DRUM
              SUBC (:INITRE)          " INITIALIZE READER
              SUBC (:INITPU)          " INITIALIZE PUNCH
              SUBC (:INITPR1)         " INITIALIZE LINE PRINTER
              S = 1000
              RUNNUMBER = S           " NC ACTION DONE YET
              LAST ACTION = S         " LAST ACTION DIFFERS FROM EXECUTION
              ANEW = - B, P           " NC NEW STARTS, CONDITION = NO

```

NAIGA:

```

B = BEGIN OF STACK
S = 32767
INSW = S          " PREPARE KEYBD FOR AUTOSTARTS
ERRONEOUS = S      " NC MISTAKES DETECTED YET
DANGEROUS = S      " ALSO, NO DANGEROUS MISTAKES
MAYI = S           " XEEN WORD IS UNDEFINED
STOCK = S          " INITIALIZE READ

S = ANEW, P        "SAME PROGRAM ONCE MORE ?
N, JUMP (2)
S = LAST ACTION, Z " LAST ACTION AN EXECUTION ?
N, GOTO (:ENDRUN)
N, SUBC (:REHEP)    " READ AND DISCARD FIRST HEPTAD
N, RUNNUMBER = B    " SCME ACTION TAKEN
N, A = ATLMMSGN
N, RCA (1)          " ) INDICATE WHETHER LINE NUMBERS
N, WANTED = A        " ) ARE TO BE COMPILED OR NOT
N, A '*' 1          " ISOLATE AT-INDICATOR
N, LUA (14)         " ) A -> = 4384
N, A = 4952
N, LIMIT = A        " VALUE FOR TIME LIMIT
N, SUBC (:READCOMP) " ENSURE PRESENCE OF COMPILER
A = 178
CLAR[0] = A         " INITIALIZE TIME
A = LIMIT
CLAR[1] = A         " PREPARE TIME LOCK
G = CLPASTED
SUBCD (:BRUSH)      " START CLOCK

S = 0
ICHECK = S          " SIGN READER ON
CASE = S            " LCWER CASE
FLEXCODE = B        " NC TELEXCODE

N, SUBC (:SUBMONITOR) " CMPILE THE PROGRAM, IF YOU CAN
Y, SUBC (:DATE AND SERIAL)
SUBCD (:CTYPE)       " TYPE "EXECUTION"
:RUNMES
A = CLAR[0]
CPLTIM = A           " NCTICE COMPILE TIME
A = ICHECK
CPLCHK = A           " NCTICE SUMCHECK OF INPUT
S = 500
RUNNUMBER = S        " NCTICE EXECUTION PHASE
SUBC (:SIGNON)       " SIGN PUNCH ON
F = 0
CASE = G             " LCWER CASE
VALUEOFCONSTANT = - F
SUBC (:INITPR3)      " NEW PAGE, IF NECESSARY
GOTO (:RUN)          " START THE OBJECT PROGRAM

```

```

ENDRUN:      SUBCD(:ASSURE LIBRARY)
              SUBC(:ISKIP)          "SKIP TO END OF INPUT
              SUBC(:ENDTRANS)       "WAIT FOR COMPLETION OF ARRAY TRANSPORTS
              SUBC(:ENDRECDU)       "WAIT FOR RECOVER AND DUMP
              S = RUNNUMBER
U, S = 1000, Z          " NO ACTION DONE AT ALL ?
Y, GOTO (:NAIGA)         " NEXT PROGRAM
              S = 500, Z           " EXECUTION PHASE ?
              LAST ACTION = S
Y, SUBC (:SIGN OFF)      " SIGN PUNCH OFF
              ('660 072 000'+:KY)  " PREVENT KEYBOARD INTERRUPTS
              SUBC (:RUN TIME)
              SUBC (:INITPR3)       " NEW PAGE, IF NECESSARY
              G = CPLTIM
Y, F / 360              " TRANSFORM COMPILE TIME
Y, F + SCHOLTEN
Y, F - SCHOLTEN         " ROUND COMPILE TIME
Y, SUBC (:AFXT6)         " PRINT TRANSLATION TIME
              G = CLAR[0]
              SUBC (:AFXT6)         " PRINT TOTAL TIME
              SUBC (:NLCR)
              G = CPLCHK
Y, SUBC (:AFXT6)         " PRINT PROGRAM CHECK SUM
              G = ICHECK
              SUBC (:AFXT6)         " PRINT TOTAL CHECK SUM
              SUBC (:NLCR)
              G = INSTR CNTR        " INSTR CNTR RELATIVE
N, JUMP(1)              " CR ABSOLUTE?
              G = BEGIN OF PR AR   "
U, S = HMODE, Z
N, SUBC (:AFXT6)         " NO LENGTH FOR LIBR PROG.
              SUBC (:NLCR)
              S = LAST ACTION, Z
N, JUMP(5)              " DCNT LOOK AT HEPCNT
              G = HEPCNT, Z
              G + D18
N, SUBC (:AFXT6)         " PRINT PUNCH COUNT
              G = SUMCHECK
N, SUBC (:AFXT6)         " PRINT PUNCH CHECK SUM
              S = 1000, Z           " CONDITION = NO
              RUNNUMBER = S         " NOT IN ACTION
              SUBCD (:CTYPE)        " NLCR
              :CRNLMS
              SUBC (:PRINT OFF)     " SIGN LINE PRINTER OFF
              ('760 072 000'+:KY)  " ALLOW KEYBOARD INTERRUPTS AGAIN
              GOTO (:NAIGA)         " NEXT PROGRAM

```

```

FIXT60:      'BEGIN'  LOOP

              A = 6           " NN
              S = 0           " MM
              ABS = S         " UNSIGNED VERSION
              SUBC (:FIX1)     " FIX CCVERSION
              '660 060 000'    " PREVENT INTERRUPTS
              SUBC (:TWAIT)    " WAIT TIL TYPEWRITER IS FREE
              A = 8
              COUNT = A       " 8 CHARACTERS
              F = :CBASE
              S = :TBUF
LOOP:         A = MG           " CHARACTER FROM FIX
              A + :TELXTB     " ADDRESS CONVERSION TABLE
              A = MA          " CCNVERT
              MS = A          " STORE
              F + 1
              S + 1
              REPP (:LOOP)     " CCUNT DOWN
              SUBC (:TYPE)
              ('010 000 000'+:TBUF[-1])
              GOTOR (MC[-1])   " EXIT FIXT60

              'END'  FIXT60

BRUSH:        M[216]=G        " RESET INSTR INTO AR[0]
              '760 070 046'    " START RESET APPARATUS
              '760 075 000'    " READ INTERRUPTS INTO S
              LCS(2),P         " NO INTER. FROM RESETTER?
              Y,JUMP(-3)       " WAIT
              '660 071 046'    " REMOVE INTERRUPT
              GOTOR(MC[-1])    " EXIT BRUSH

HOK:          'BEGIN'  START

              S = D18
              MA[1] = S        "ACTION COUNTER = 1
              S = MA[-1]       "ADDRESS OF SS[1]
              S'='D18M1        "CLEAN AWAY HOK SECTION
              G = MS[1]
              MC = G           "PRESERVE OLD CODEWORD
              F = 0
              MS[1] = G        "CODEWORD
              MA[0] = -G       "INTERRUPT COUNTER=-1
              G = A
              RUA(2)           "APPARATUS NUMBER + 16
              A + START        "START INSTR IN A
              DO(A)
              U, S = MG[0] , P  "DONE YET?
              N, JUMP(-2)       "THEN WAIT
              A = MC[-1]
              MS[1] = A         "RESTORE OLD CODEWORD
              GOTOR(MC[-1])     "EXIT HOK
              ('760 070 000' - 16)

START:        'END'  HOK

```

FOUND LIBRARY:

'BEGIN' FOUND CROSSTABLE

S = -3

FIRST 3 TIMES = S

S = PERMANENT CATALOGUE END

END OF CATALOGUE = S

G = BEGIN OF TRAFOTABLE

G + MAX NBR OF ROUTINES

BEGIN OF CROSSTABLE = G

G + MAX NBR OF ROUTINES

BEGIN OF INFOTABLE = G

S = 0

MG(3) = S

") CROSSREFERENCE

") IRRELEVANT FOR

") PERMANENT PART

S = DR LIBR BEG

END OF DRUMLIBRARY = S ") NO PROCEDURE ON

") DRUM YET

G + 4

END OF INFOTABLE = G " 3 ENTRIES

MG = S " FIRST FREE PLACE ON DRUM

S = NBR OF PERMANENT ROUTINES

NBR OF ROUTINES = S

S + 1

S + BEGIN OF TRAFOTABLE

END OF TRAFOTABLE = S

S + MAX NBR OF ROUTINES

END OF CROSSTABLE = S

MS = G " M[END OF CROSSTABLE] :=

" END OF INFOTABLE

G + D23M3 " (('040 000 000' - 3) +:PSEUDO ENTRY)

S - 1 ") ALL PERMANENT ROUTINES

MS = G ") USE PSEUDO ENTRY

U, S = BEGIN OF CROSSTABLE, Z

N, GOTO(:FOUND CROSSTABLE)

GOTO(:UPDATE LIBRARY)

'END' FOUND LIBRARY

FOUND CROSSTABLE:

" ALL INPUT-OUTPUT INTERRUPTS CAUSE A TRANSFER TO I0INTP
 " THROUGH THE INSTRUCTION IN M[24]

I0INTP:	ASAFE = A	
	SSAFE = S	
	SAFE8 = 8	
	FSAFE = F	
	'660 075 000'	"READ INTERRUPT WORDS
	'760 075 001'	"INTO AS
	'660 076 100'	"REMOVE
	'760 076 101'	"UNWANTED INTERRUPT BITS
	NORAS	
	A = :MC	
	A + :INTAB	"SELECT DESTINATION
	B = :INSTCK	"INTERRUPTION STACK
	ININTRPT = B	
	GOTO (MA)	

INTAB:	:CLOCK	"CLOCK	
	:DENTST	"RESET APPARATUS	
	:DYST	"CHARON FAULT HANDLING	
	:DYST	"APPARATUS	36
	:DYST	"	35
	:DYST	"	34
	:DYST	"	33
	:DYST	"	32
	:DYST	"	0
	:DYST	"	1
	:DYST	"	2
	:DYST	"	3
	:DYST	"	4
	:DYST	"	5
	:DYST	"	6
	:DYST	"	7
	:DYST	"	8
	:DYST	"	9
	:DYST	"	10
	:DYST	"	11
	:DYST	"	12
	:DYST	"	13
	:DYST	"	14
	:DYST	"	15
	:DYST	"	16
	:DYST	"	17
	:DYST	"	18
	:DYST	"	19
	:DYST	"	20
	:DYST	"	21
	:DYST	"	22
	:DYST	"	23
	:DYST	"	24
	:DYST	"	25
	:DYST	"	26
	:DYST	"	27
	:DYST	"	28
	:DYST	"	29
	:DYST	"	30
	:DYST	"	31

```

INTAB[8+:KY]:      :KEYBD
INTAB[40]:

RESTORE:           ININTRPT = - B      " END OF INTERRUPT
                   A = ASAFE
                   S = SSAFE
                   B = SAFEB
                   F = FSAFE
                   GOTOR(M[23])        "EXIT INTERRUPT PROGRAMS

END CRIT:          CRITICAL = - B      " END OF CRITICAL SECTION
                   A = KICKOFF, P
N, GOTOR (MC[-1])
                   KICKOFF = - B
                   GOTO (:FINIS1)

EXITNOSWAP:        GOTOR (MC[-1])

WASTE TIME:        GOTOR (MC[-1])

DATE AND SERIAL:   ('660 072 000' +:KY)"PREVENT KEYBOARD INTERRPTS
                   SUBCD (:CTYPE)      "TYPE NLCR BLK DIG
                   :SMES
                   G = DATE
                   SUBC (:FIXT60)      "TYPE DATE
                   SUBCD (:CTYPE)
                   :SMES1
                   G = SERIAL
                   SUBC (:FIXT60)      "TYPE SERIAL
                   ('760 072 000' +: KY)
                   GOTO(:INITPR2)

RUN TIME:          U, S = HMODE, Z
                   N, SUBCD (:CTYPE)
                   :CLCKMS              "TYPE RUN TIME
                   Y, SUBCD (:CTYPE)
                   :CLCKMS1            "TYPE LOADING
                   G = CLPASTE1
                   SUBCD (:BRUSH)
                   G = CLAR[0]
                   F/360
                   F + SCHOLTEN
                   F = SCHOLTEN
                   CLAR[0] = G
                   GOTO(:FIXT60)

```

CTYPE:	SUBC(:TWAIT)	"WAIT TIL PREV. LINE IS DONE
	A = 1	"INCREASE LINK BY ONE
	PLUSA(M[B - 1])	"ADDRESS OF FIRST WORD
	G = MA[-1]	"SAVE STACK POINTER
	KBUF[3] = B	"BEGIN ADDRESS OF TYPE BUFFER
	B = :TBUF	"PACKED WORD IN A
	A = MG	"CLEAR S
	S = 0	"SHIFT 6 BITS TO S. DONE?
	LCSA(6) , Z	"INCREMENT FOR NEXT PACKED WORD
Y,	F + 1	"GET NEXT PACKED WORD
Y,	JUMP(-5)	"END MARKER?
U,	S = 63, Z	"STORE IN TYPE BUFFER
N,	M[B] = S	"INCREMENT STORE ADDRESS
N,	B + 1	"RETURN FOR NEXT CHARACTER
N,	JUMP(-8)	
	A = -:TBUF	"NO. OF CHARACTERS IN A
	A + B	"SHIFT TO POSITION
	LUA(18)	"MAKE CODEWORD
	A '+' :TBUF[-1]	"RESTORE STACK POINTER
	B = KBUF[3]	"TYPE CONTENTS OF TBUF
	GOTO(:TYPE3)	
TYPE:	A = 1	"INCREASE LINK BY ONE
	PLUSA(M[B - 1])	"CODEWORD
	A = MA[-1]	"STORE IN START LINK
TYPE3:	TSS[2] = A	"INCREMENT / DECREMENT. MORE?
	A = D18M1, P	
N,	GOTO(:TACT)	"LOOK IN BUFFER
	S = MA	"IS IT CASE DEFINITION?
U,	S = 31, P	"STORE IN KEYBOARD BUFFER
Y,	KBUF[2] = S	"REPEAT
	JUMP(-6)	
TACT:	TAR[1] = -B	"INTERRUPT COUNTER = -1
	S = D18	
	TAR[2] = S	"ACTION COUNTER = 1
	('760 070 000'+:TY)	"START TYPEWRITER
	GOTOR(MC[-1])	"EXIT CTYPE, TYPE, TACT
TWAIT:	S = TAR[1], P	"TYPING COMPLETED?
Y,	GOTO(:TOFF)	" IN INTERRUPT PROGRAM ?
	S = ININTRPT, P	" ELSE ALLOW INTERRUPTS
	'760 360 000'	
	A + 0	" AND PREVENT INTERRUPTS AGAIN
	'660 360 000'	
	S = KAR[0]	"FIGURE SHIFT?
	S = 27, Z	"LETTER SHIFT
N,	S = 3, P	"STORE IN KEYBOARD CASE
Y,	KCASE = S	"ACCEPT SYMBOLS FROM KEYBD
	('760 070 000'+:KY)	
	GOTO(:TWAIT)	
TOFF:	A = TAR[0], P	"OK?
Y,	GOTOR(MC[-1])	"EXIT TWAIT
	SUBC(D18M1)	"MACHINE FAILURE

```

KEYBD:      S = KAR[0], P      "IS THERE ALREADY A KEY PRESSED?
N, JUMP(-2)      "WAIT
      KAR[0] = -S
      SYM = S      "STORE KEYBOARD SYMBOL
      KBUF[3] = S      "STORE FOR PRINTING
      ('660 071 000'+:KY) "CLEAR INTERRUPT BIT
      KAR[1] = B      "INTERRUPT COUNTER = 0
      ('760 070 000'+:KY) "ACCEPT SYMBOLS FROM KEYBD
      A = S
      A = 27, Z      "FIGURE SHIFT?
N, A = 3, P      "LETTER SHIFT?
N, JUMP(4)
      KCASE = A      "STORE IN KEYBOARD CASE
      A = INSW, Z      "IS KEYBD CALLED AS SUBROUTINE?
N, GOTO(:RESTORE)
      GOTO(:KEYBD)      "WAIT FOR NOT SHIFT
      SUBC(:TWAIT)      "WAIT TIL TYPING IS DONE

      S = KCASE
      LUS(2)
      S + 33      "RED COLOUR
U, S = KBUF[2], Z      "COLOUR AND SHIFT OK?
N, KBUF[2] = S      "CHANGE
N, A = CW2      "TWO SYMBOLS TO BE PRINTED
Y, A = CW1      "ONLY ONE SYMBOL
U, A = INSW, Z      "IS KEYBD CALLED AS SUBROUTINE?
N, A = CW4      "FOUR SYMBOLS TO BE PRINTED
      TSS[2] = A      "CODEWORD
      SUBC(:TACT)      "PRINT THE SYMBOL
      A = KCASE
      LUA(5)
      A + SYM      "SYMBOL + SHIFT
      A + :KTAB
      A = MA      "TRANSLATION AND SWITCH ADDRESS
Y, GOTOR(MC[-1])      "EXIT IF CALLED AS SUBROUTINE
      GOTO(A)      "SWITCH

```

```

XEENS:      MAYI=A          " MAKE XEEN WORD UNAVAILABLE
            GOTO(:RESTORE)

STOP:       MAYI=B          " MAKE XEEN WORD UNAVAILABLE
            GOTOR(MC[-1])

XEEN:       SUBC(:RND)      " ROUND PARAMETER
            MC=G            " STACK IT
            '660 060 000'   " CLEAR INTERRUPT PERMIT
            A=MAYI,Z        " IS XEEN WORD AVAILABLE?
XGET:       Y,A=XEENW       " THEN TAKE IT IN A
            Y,A'*MC[-1]     " COLLATE WITH PARAMETER
            G=A             " RESULT IN F
            Y,GOTOR(MC[-1]) " AND EXIT XEEN
            SUBC(:CTYPE)    " TYPE "XEEN"
            :XMES
            A = 0
            INSW = A        " PUT KEYBOARD IN INPUT MODE
            MAYI = A
            SUBC(:INOCT)    " COLLECT OCTAL WORD
            XEENW=A         " STORE COMPLETED RESULT
            INSW=B,P        " RESET KEYBD FOR AUTOSTARTS. COND=YES
            GOTO(:XGET)     " FETCH THE RESULT AND EXIT

NXT KYBD SL: SUBC(:KEYBD)   " GET A KEYBOARD SYMBOL
            RUA(18)         " CLEAN AWAY THE ADDRESS PORTION
            S=A             " COPY IN A
            GOTOR(MC[-1])   " EXIT NXT KYBD SL

NKS:        SUBC(:NXT KYBD SL)
            U,S-255,Z       " SHARP SIGN?
            N,GOTOR(MC[-1]) " EXIT NKS
            B-3             " REMOVE THREE LINKS FROM STACK
            GOTO(:HANDRD)   " BEGIN AGAIN TO READ A NUMBER

HAND:       SUBC(:RND)      " ROUND PARAMETER
            MC=G            " SAVE IT IN STACK
            '660 060 000'   " CLEAR INTERRUPT PERMIT
            SUBC(:CTYPE)    " TYPE "HAND"
            :HMES
            G=MC[-1]        " TAKE PARAMETER FROM STACK
            SUBC(:FIXT60)    " TYPE IT OUT
DNAH:       S=STOCK
            MC=S            " PRESERVE STOCK FROM READ
            S=0
            INSW=S          " PREPARE KEYBOARD FOR INPUT
            S=32767
HANDRD:     STOCK=S         " INITIALIZE STOCK FOR READ
            S=KYBD          " TELL READ TO GET SYMBOLS FROM NKS
            SUBC(:READ1)    " INPUT A NUMBER
            INSW=B          " RESET KEYBOARD FOR AUTOSTARTS
            S=MC[-1]        " RESTORE OLD STOCK TO READ
            STOCK=S
            GOTOR(MC[-1])   " EXIT HAND
KYBD:       GOTO(:NKS)

```

```
FINIS:      A = 999
            S = CRITICAL, P      " TO BE POSTPONED ?
Y, KICKOFF = A
Y, GOTO (:RESTORE)              " THEN POSTPONE
FINIS1:     ININTRPT = - B      " NO LONGER IN INTERRUPT
            '760 060 000'      "SET INTERRUPT PERMIT
            B = BEGIN OF STACK "SET STACK POINTER
            SUBC(:ERRORM)
            GOTO(:ENDRUN)
```

```

KEYA:      A = 0
            JUMP (1)
KEYT:      A = 2
            S = 1
KEYT[2]:   S '*' ATLMSGN1      " KEEP OTHER SIGNAL UNCHANGED
            S + :MA
            ATLMSGN1 = S
            GOTO (:RESTORE)

CLOCK:     ('660 071 000'+ 39) " CLEAR INTERRUPT BIT
            A = 997
            GOTO (:FINIS[1])

NSTART:
ESTART:    GOTO (:ENSTART)

L:         A = 0
            JUMP (1)
KEYM:      A = 1
            S = 2
            GOTO (:KEYT[2])

INOCT:     A = 0
            MC = A
            SUBC(:NXT KYBD SL) " INITIALIZE OCTAL WORD TO ZERO
            A = MC[-1]         " GET SYMBOL FROM KEYBOARD
                                " GET RESULT SO FAR
U, S = 255, Z                 " SHARP SIGN?
Y, GOTO(:INOCT)               " START ALL OVER
U, S = 7, P                   " NOT AN OCTAL DIGIT?
N, LCA(3)                     " MAKE ROOM FOR NEW DIGIT
N, A '+' S                    " PUT IT IN
Y, S = 254, Z                 " IRCN CROSS?
N, GOTO(:INOCT[1])           " GET NEXT SYMBOL
                                " EXIT INOCT
            GOTOR(MC[-1])

```

" COPY SECTION

```

'BEGIN'          CSECOND, CTHIRD, PASTE0, PASTE1, SS, HEP, CWD,
                  CREADER, CPCHER, CPCHER3, END, CIAR, CPAR

COPY:
M[64+(4*:CRE)]:  CIAR:
M[64+(4*:CPU)]:  CPAR:

COPY[0]:          G = PASTE 0
                  SUBC (:BRUSH)          " RESET READER
                  G = PASTE 1
                  SUBC (:BRUSH)          " RESET PUNCH
                  A = :HEP
                  CPAR[0] = A            " PUNCH OK + LABEL
                  A = :CSECOND
                  INTAB[8+:CRE] = A      " SET INTERRUPT ADDRESS FOR READER
                  A = :CPCHER
                  INTAB[8+:CPU] = A      " SET INTERRUPT ADDRESS FOR PUNCH
                  GOTO (:CPCHER 3)      " START READING

CSECOND:          A = :CTHIRD
                  JUMP (1)

CTHIRD:           A = :CREADER
                  INTAB[8+:CRE] = A      " SET INTERRUPT ADDRESS FOR READER
                  ('660 071 000'+:CRE) " REMOVE INTERRUPT BIT
                  GOTO(:CPCHER3)        " START READING

PASTE 0:          ('000 004 000'+:CRE)
PASTE 1:          ('000 004 000'+:CPU)
SS:               'SKIP' 1
HEP:              'SKIP' 1
CWD:              ('001 000 000'+:HEP[-1])

DONT:             ('660 070 000'+:CRE)
                  ('660 070 000'+:CPU)
                  G = PASTE 0
                  SUBC (:BRUSH)          " RESET READER
                  G = PASTE 1
                  SUBC (:BRUSH)          " RESET PUNCH
                  GOTO (:RESTORE)        " EXIT

CREADER:          ('660 071 000'+:CRE) " REMOVE INTERRUPT BIT
                  CPAR[1] = B            " INTERRUPT COUNTER = 0
                  A = CWD
                  CPAR[2] = A            " ACTION COUNTER = 1
                  ('760 070 000'+:CPU) " START PUNCH
                  GOTO (:RESTORE)        " EXIT

```

```
CPCHER:      ('660 071 000'+:CPU) " REMOVE INTERRUPT BIT
              A = CPAR[0], P      " OK ?
N, JUMP (7)
CPCHER 3:    A = :HEP
              CIAR[0] = A          " READER OK + LABEL
              CIAR[1] = A          " INTERRUPT COUNTER = 0
              A = CWD
              CIAR[2] = A          " ACTION COUNTER = 1
              ('760 070 000'+:CRE) " START READER
              GOTO (:RESTORE)      " EXIT
              A = :CPAR[1]
              SUBC (:HOK)          " TCWARDS NOK2
              A = :CPAR[1]
              SUBC (:HOK)          " TCWARDS QK
END:         GOTO (:CREADER[1])    " PUNCH SYMBOL ANEW
'END'
```

KTAB:	('105 000 000' + :RESTORE)	"ASTERISK = ↑
	('005 000 000' + :RESTORE)	"5
	('167 000 000' + :RESTORE)	"CR
	('011 000 000' + :RESTORE)	"9
	('135 000 000' + :RESTORE)	"SPACE
	('131 000 000' + :RESTORE)	"
	('127 000 000' + :RESTORE)	",
	('130 000 000' + :RESTORE)	".
	('167 000 000' + :RESTORE)	"NL
	('175 000 000' + :RESTORE)	")
	('004 000 000' + :RESTORE)	"4
	('145 000 000' + :RESTORE)	"]
	('010 000 000' + :RESTORE)	"8
	('000 000 000' + :RESTORE)	"0
	('174 000 000' + :RESTORE)	":"
	('106 000 000' + :RESTORE)	"#
	('003 000 000' + :RESTORE)	"3
	('100 000 000' + :RESTORE)	"+
	('376 000 000' + :RESTORE)	"IRON CROSS
	('102 000 000' + :RESTORE)	"*
	('170 000 000' + :RESTORE)	"'
	('006 000 000' + :RESTORE)	"6
	('144 000 000' + :RESTORE)	"["
	('103 000 000' + :RESTORE)	"/
	('101 000 000' + :RESTORE)	"_
	('002 000 000' + :RESTORE)	"2
	('133 000 000' + :RESTORE)	";
CW1:	('001 000 000' + :KBUF[2])	
	('007 000 000' + :RESTORE)	"7
	('001 000 000' + :RESTORE)	"1
	('142 000 000' + :RESTORE)	"("
CW2:	('002 000 000' + :KBUF[1])	

	('377 000 000' + :RESTORE)	"SHARP SIGN
	('035 000 000' + :KEYT)	"T
	('167 000 000' + :RESTORE)	"CR
	('030 000 000' + :RESTORE)	"O
	('173 000 000' + :RESTORE)	"SPACE
	('021 000 000' + :RESTORE)	"H
	('027 000 000' + :NSTART)	"N
	('026 000 000' + :KEYM)	"M
	('167 000 000' + :RESTORE)	"NL
	('025 000 000' + :L)	"L
KTABR:	('033 000 000' + :RESTORE)	"R
	('020 000 000' + :RESTORE)	"G
KTABI:	('022 000 000' + :RESTORE)	"I
	('031 000 000' + :RESTORE)	"P
	('014 000 000' + :COPY)	"C
	('037 000 000' + :RESTORE)	"V
	('016 000 000' + :ESTART)	"E
	('043 000 000' + :RESTORE)	"Z
	('015 000 000' + :DONT)	"D
	('013 000 000' + :RESTORE)	"B
	('034 000 000' + :RESTORE)	"S
	('042 000 000' + :RESTORE)	"Y
	('017 000 000' + :FINIS)	"F
	('041 000 000' + :XEENS)	"X
	('012 000 000' + :KEYA)	"A
	('040 000 000' + :RESTORE)	"W
	('023 000 000' + :RESTORE)	"J
CW4:	('004 000 000' + :KBUF(-1))	
	('036 000 000' + :RESTORE)	"U
	('032 000 000' + :RESTORE)	"Q
	('024 000 000' + :RESTORE)	"K

" READ READS THE NEXT NUMBER FROM THE TAPE READER IN MC FLEXOWRITER
 " CODE, CONVERTS IT TO A NORMALIZED FLOATING-POINT NUMBER, AND
 " DELIVERS IT IN THE F-REGISTER.

READ: 'BEGIN' LOOP, WORK, D0, D1, P0, P1,
 ASEMBL, ZERO, SIEVE, SEPEX, SEP, CLEAN,
 XPONT, NXTSYM, TAPE, ROUND

" ASSUMED TO BE DECLARED GLOBALLY ARE: READ1, DECBIN
 " ASSUMED TO BE DECLARED AND DEFINED GLOBALLY ARE:
 " CNT, PT, HEAD, TAIL, MSIGN, STOCK, SIXMIL, DEXP,
 " BSAFE, BEXP, TENPOW, SCALE, GIANT, ESIGN,
 " NXTTAPESL, VALUE OF CONSTANT, SIXMM1
 " STOCK IS SET EXTERNALLY TO 32767 BEFORE FIRST CALL OF READ.
 " A CALL OF SUBC(:DECBIN)
 " MAY BE USED TO CONVERT A DOUBLE-LENGTH POSITIVE INTEGER IN
 " F, ALONG WITH A DECIMAL EXPONENT IN S, INTO
 " A NORMALIZED FLOATING-POINT NUMBER IN F.

READ1: S=TAPE
 NXTSYM=S " INITIALIZE FOR TAPE READER
 F=0
 CNT=F " I.E. CNT:=PT:=0
 HEAD=F " I.E. HEAD:=TAIL:=0
 DEXP=F " I.E. DEXP:=MSIGN:=0
 S=STOCK " +, -, OR SEPARATOR FROM LAST CALL
 JUMP(2) " GO ANALYZE IT
 SUBC(:SIEVE) " GET NEXT SYMBOL
 N,GOTO(:LOOP) " HANDLE FIRST MANTISSA DIGIT
 U,S=32767,Z " IS NON-DIGIT A SEPARATOR?
 N,MSIGN=S " STORE SIGN OF MANTISSA
 JUMP(-5) " AND GET NEXT SYMBOL

LOOP: S=HEAD
 S=SIXMM1,P " CAN MULT BY 10 CAUSE OVERFLOW?
 N,S=TAIL ")
 N,MULAS(10) ") [HEAD, TAIL] :=
 N,TAI=S ") [HEAD, TAIL] * 10
 N,S=HEAD ") + THE NEW DIGIT
 N,MULAS(10) ")
 N,HEAD=S ")
 S=-PT
 Y,S+1
 CNT+S " COUNT DIGITS
 SUBC(:SIEVE) " GET NEXT SYMBOL
 N,GOTO(:LOOP) " IF DIGIT, PROCESS IT

WORK: A=ESIGN
 A=65,Z " NEGATIVE EXPONENT SIGN
 Y,A=-DEXP " THEN TAKE - EXPONENT
 N,A=DEXP " OTHERWISE POSITIVE
 A+CNT " CORRECT EXPONENT WITH CNT
 DEXP=A " STORE DECIMAL EXPONENT
 STOCK=S " STORE THE FINAL SEPARATOR FOR NEXT CALL
 JUMP(3)

```

DECBIN:      HEAD=F
              DEXP=S
              MSIGN=B          " I.E. MSIGN ≠ 65
              BSAFE=B          " PRESERVE THE STACK POINTER
              A=-HEAD
              S=-TAIL          " -MANTISSA IN AS
              NORAS,Z          " NORMALIZE MANTISSA. IS IT ZERO?
              Y,F=0
              Y,GOTO(:ZERO)
              BEXP=-B          " INITIALIZE BINARY EXPONENT
              B=10             " INITIALIZE DECIMAL-BINARY CONVERSION
              HEAD=-A
              U,A=DEXP,P        " IS DEXP PCS?
              Y,GOTO(:P1)       " MULTIPLICATION CYCLE
              GOTO(:D1)         " DIVISION CYCLE

D0:          U,A+TENPOW[B-1],P  " CAN WE DIVIDE WITHOUT OVERFLOW?
              N,RUAS(1)         " IF NOT, SHIFT RIGHT ONE
              DIVAS(TENPOW[B-1]) " DIVIDE BY POWER OF 10
              HEAD=-S           " STORE MORE SIGNIFICANT HALF OF QUOTIENT
              DIVAS(TENPOW[B-1]) " DIVIDE FOR LESS SIGNIFICANT HALF
              A=-SCALE[B-1]     " CORRESPONDING BINARY SCALING FACTOR
              N,A+1             " CORRECT FOR PRESIFT
              BEXP+A            " BRING BEXP UP TO DATE
              A=-HEAD

D1:          U,B+DEXP,P        " IS DEXP > 10(USUALLY)?
              Y,B=-DEXP,Z       " HAS DECIMAL EXPT BEEN REDUCED TO 0 YET?
              N,DEXP+B          " BRING DEXP UP TO DATE
              N,GOTO(:D0)       " DIVIDE
              GOTO(:ASSEMBL)    " BINARIZATION IS NOW COMPLETE

P0:          MULS(TENPOW[B-1])  " MULT TAIL BY POWER OF 10
              S=-HEAD
              MULAS(TENPOW[B-1]) " MULT HEAD BY POWER OF 10
              LCAS(1),P         " LEFT ONE, WAS IT ALREADY NORMALIZED?
              Y,RCAS(1)         " THEN SHIFT IT BACK
              HEAD=-A
              A=SCALE[B-1]     " BINARY SCALING FACTOR
              N,A-1             " CORRECT FOR NORMALIZING SHIFT
              BEXP+A            " BRING BEXP UP TO DATE

P1:          U,B=DEXP,P        " DEXP < 10(USUALLY)?
              Y,B=DEXP,Z       " THEN USE DEXP. IS IT ZERO?
              N,DEXP-B          " BRING DEXP UP TO DATE
              N,GOTO(:P0)       " MULTIPLY

```

```

ASEMBL:      A=HEAD
              S=-S          " PLUS MANTISSA IN AS
              B=BEXP
              B+12          " CORRECT BEXP FOR SHIFT
              U,B+2048,P    " IS EXPONENT UNDERFLOW IMPOSSIBLE?
              Y,GOTO(:ROUND)
              B+2087,P      " APPARENT UNDERFLOW CORRECTABLE BY SHIFT?
              N,B=40        " IF NOT, PREPARE TO SHIFT 40 PLACES
              Y,B-39        " MINUS REQUIRED NUMBER OF SHIFTS
              Y,B=-B
              U,B-31,P      " )
              Y,B-31        " )
              Y,RUAS(31)    " ) SHIFT RIGHT
              RUAS(B)       " )
              B=-2047       " UNDERFLOW EXPONENT
              S+2048,P      " ROUND MANTISSA TAIL, NO CARRY?
ROUND:        N,S=0        " IN CASE OF CARRY, TAIL := 0
              N,A+1,P      " CARRY OVER TO HEAD. NO OVERFLOW?
              N,RCA(1)      " REPAIR OVERFLOW IN HEAD
              N,B+1        " AND CORRECT BINARY EXPONENT
              U,B-2047,P    " EXPONENT OVERFLOW?
              Y,F=GIANT
              Y,GOTO(:ZERO) " THEN PROVIDE LARGEST NUMBER
              RUAS(12)      " SHIFT MANTISSA TO POSITION
              HEAD=A
              A=B,Z        " IS EXPONENT ZERO?
              Y,A=0        " MAKE IT +0 THEN
              A='4095      " REMOVE COPIES OF SIGN BIT
              RCA(12)      " SHIFT EXPONENT INTO PLACE
              A='HEAD      " COLLATE EXPONENT WITH HEAD OF MANTISSA
              F=A          " PUT THE RESULT IN F
ZERC:        B=MSIGN      " SIGN OF THE MANTISSA
              B=65,Z      " IS IT MINUS?
              Y,F=-F       " THEN COMPLEMENT F
              F+0          " NORMALIZE F
              B=BSAFE      " RESTORE THE STACK POINTER
              VALUE OF CONSTANT=F " STORE RESULT FOR ERROR MESS.
              GOTOR(MC[-1]) " EXIT READ

SIEVE:       SUBC(:NXTSYM) " GET A SYMBOL
              A=S          " MAKE A COPY IN A
              U,S=9,P      " IS SYMBOL A NCN-DIGIT?
              N,GOTO(MC[-1]) " RETURN WITH DIGIT, CONDITION = NO
              U,S=65,Z     " IS IT MINUS SIGN?
              N,A=64,Z     " OR PLUS SIGN?
              Y,GOTO(MC[-1]) " RETURN WITH SIGN, CONDITION = YES
              U,S=88,Z     " IS IT DECIMAL POINT?
              Y,A=1
              Y,PT=A       " NOTE OCCURENCE OF POINT
              Y,GOTO(:SIEVE) " AND GET NEXT SYMBOL
              U,S=89,Z     " IS IT " ?
              Y,GOTO(:XPONT) " GO TO PROCESS THE EXPONENT
              U,S=123,Z    " IS IT A SPACE?
              N,GOTO(:SEP) " THEN IT CAN ONLY BE A SEPARATOR
              SUBC(:NXTSYM) " GET SYMBOL WHICH FOLLOWS THE SPACE
              U,S=123,Z    " IS IT ALSO A SPACE?
              N,GOTO(:SIEVE[1]) " THEN GO FIND OUT WHAT IT IS

```

```
SEPEX:      S=32767,P      " SEPARATOR INDICATION IN S, CONDITION = YES
            GOTO(MC[-1])   " RETURN WITH SEPARATOR
SEP:        U,S=120,Z      " IS THE SEPARATOR AN APOSTROPHE?
            N,GOTO(:SEPEX) " IF NOT, AN ORDINARY SEPARATOR
CLEAN:      SUBC(:NXTSYM)  " NEXT SYMBOL
            S=120,Z      " IS IT APOSTROPHE?
            N,GOTO(:CLEAN) " IF NOT, FLUSH IT OUT
            GOTO(:SEPEX)  " END OF COMMENT BETWEEN APOSTROPHES

XPONT:      B=1           " REMOVE THE UNUSED LINK
            S=CNT,Z
            Y,A=1,E      " NO DIGITS IN THE MANTISSA?
            Y,TAI=A      " THEN THE MANTISSA IS 1
            SUBC(:SIEVE) " NEXT SYMBOL
            N,JUMP(3)     " HANDLE FIRST DIGIT OF EXPONENT
            U,S=32767,Z  " IS THE NON-DIGIT A SEPARATOR?
            N,ESIGN=S     " THEN STORE SIGN OF EXPONENT
            JUMP(-5)      " AND GET NEXT SYMBOL
            DEXP=S        " STORE EXPONENT
            SUBC(:SIEVE) " GET NEXT SYMBOL
            Y,GOTO(:WORK) " NON-DIGIT, SO END OF EXPONENT
            S=DEXP
            MULAS(10)     " PROCESS THE NEW DIGIT
            U,S=999,P     " IS THE EXPONENT HUGE?
            Y,S=1000      " THEN 1000 IS BIG ENOUGH
            JUMP(-8)      " REPEAT

NXTSYM:     'SKIP'1
TAPE:       GOTO(:NXTTAPESL)
            'END' READ
```

" OUTPUT CONVERSION ROUTINES

```

'BEGIN' PARAM, STORE, SPILL, FLO2, TEN, LAST, CONV,
      DIV, MUL, MULT, AH, RU, GPONE, DIGITS,
      SHRTCT, RET, NDBTP, STSP, STZ, IZERO,
      IZERO3, FILL

```

```

" ASSUMED TO BE DECLARED GLOBALLY ARE: FIX, FIX1,
"   FLO, FLO1, GEN, PUNCH, FIXP, ABSFIXP, FLOP, PO
" ASSUMED TO BE DECLARED AND DEFINED GLOBALLY ARE:
"   XX, RND, TEMP, ABS, NN, MM, DEXP, SIXMIL,
"   FIXFLO, BSAFE, BEXP, CBASE, HEAD, TAIL, TENPOW,
"   SCALE, HALF, SGNBIT, TENTH, NTAB, CCNT,
"   FLEXHP, FLEXTB

```

```

PARAM:      XX=F          " PRESERVE F
            F=MC[-2]      " TWO LINKS FROM THE STACK
            TEMP=F        " SAVE THEM
            F=MC[-2]      " SECOND PARAMETER
            SUBC(:RND)    " ROUND IT
            S=G          " MM IN S
            F=MC[-2]      " FIRST PARAMETER
            SUBC(:RND)    " ROUND IT
            A=G          " NN IN A
            F=TEMP        " TWO LINKS
            MC=F          " RETURN THEM TO THE STACK
            F=XX          " THIRD PARAMETER, XX, IN F
            GOTOR(LINK)   " EXIT PARAM

STORE:      MG=S          " STORE CHARACTER
            F+1          " INCREMENT STORE LOCATION
            GOTOR(MC[-1]) " EXIT STORE

SPILL:      B=1          " REMOVE UNUSED LINK FROM STACK
            A=13         " NN := 13
            S=3          " MM := 3
            ABS=A        " SIGNED VERSION
            GOTO(:FLO2)  " PERFORM FLOATING CONVERSION

FIX:        SUB(:PARAM)  " GET PARAMETERS IN A, S, AND F
FIX1:      XX=F          " STORE THE NUMBER TO BE CONVERTED
            NN=A         " STORE NUMBER OF DIGITS BEFORE POINT
            MM=S         " STORE NUMBER OF DIGITS AFTER POINT
            A+1,P        " NN ≥ 0?
Y,S+1,P     " MM ≥ 0?
Y,A+S       "
Y,A-2,P     " MM+NN ≠ 0?
N,GOTO(:SPILL)
A-21,P      " MM+NN > 21?
Y,GOTO(:SPILL)
S=1         " FIXFLO INDICATOR
SUBC(:CONV) " CONVERT THE NUMBER
GOTOR(MC[-1]) " EXIT FIX

```

```

FLO:      SUB(:PARAM)
FLO1:     XX=F
FLO2:     COUNT=S
          S=1,Z
          N,S=1,Z
          N,S=1,Z
          N,GOTO(:SPILL)
          MM=A
          U,A=13,P
          A=0,E
          Y,GOTO(:SPILL)
          NN=S
          SUBC(:CONV)
          S=13
          MG[-1]=S
          A=DEXP,P
          Y,S=10
          N,S=11
          N,A=-A,P
          SUBC(:STORE)
          S=A
          A=:SXMIL[-1]
          A+COUNT
          MULS(MA)

TEN:      TENAS
          DEXP=S
          S=A
          REPZ(:LAST)
          Y,S=S,Z
          Y,S=14
          SUBC(:STORE)
          S=DEXP
          GOTO(:TEN)
          SUBC(:STORE)
          S=14
          SUBC(:STORE)
          GOTOR(MC[-1])

LAST:

CONV:     FIXFLO=S
          BSAFE=B
          F=XX,P
          N,F=-F
          A=-F
          S=-A
          RUS(15)
          BEXP=S
          A!*'32767
          A!*'-32767
          S=-G
          F=:CBASE[1]
          NORAS
          B=52
          BEXP=B
          HEAD=A
          A=-0
          DEXP=A
          B=10

" GET PARAMETERS IN A, S, AND F
" STORE THE NUMBER TO BE CONVERTED
" STORE NUMBER OF EXPONENT DIGITS
" WAS IT ONE?
" OR TWO?
" OR THREE?
" IF NONE OF THESE, THEN ERROR
" STORE NUMBER OF FRACTION DIGITS
" MM > 13?
" OR MM < 1 ?
" THEN ERROR
" NUMBER OF DIGITS BEFORE POINT := 0
" CONVERT FRACTION PART OF NUMBER
"
" STORE " OVER SPACE
" IS DECIMAL EXPONENT POSITIVE?
" +
" -
" ABS(DEXP), CONDITION = YES
" STORE SIGN OF EXPONENT

" MULTIPLY DEXP BY .1, .01, OR .001

" MULT BY TEN
" STORE FRACTION
" EXPONENT DIGIT TO S
" COUNT DOWN
" LEADING ZERO?
" THEN REPLACE IT BY A SPACE
" STORE SPACE OR DIGIT
" RESTORE FRACTION TO S
" NEXT DIGIT
" STORE LAST DIGIT
" SPACE
" STORE FINAL SPACE
" EXIT FLO

" STORE FIX/FLO INDICATOR
" PRESERVE STACK POINTER

" ABSOLUTE VALUE OF NUMBER IN F

" BINARY EXPONENT IN S

" MINUS MANTISSA IN AS
" INITIALIZE STORE
" NORMALIZE MANTISSA

" BIN EXP WHEN A,S SEEN AS FRACTION

" INITIALIZE DECIMAL EXPONENT

```

```

DIV:      BEXP=A,P           " UPDATE BEXP, STILL MORE DIVIDING TO DO?
N,A=BEXP
N,GOTO(:MULT)              " ENTER MULTIPLICATION LOOP
DEXP=B                     " UPDATE DEXP
A=HEAD
U,A+TENPOW[B-1],P          " CAN WE DIVIDE WITHOUT OVERFLOW?
N,RUAS(1)                  " PRESIFT FOR DIVISION
DIVAS(TENPOW[B-1])         " DIVIDE BY POWER OF TEN
HEAD=S                     " STORE MORE SIGNIFICANT HALF OF RESULT
DIVA(TENPOW[B-1])          " DIVIDE FOR LESS SIGNIFICANT HALF
A=-SCALE[B-1]              " CORRESPONDING BINARY SCALING FACTOR
N,A+1                      " CORRECT FOR PRESIFT
GOTO(:DIV)                 " REPEAT

MUL:      A+SCALE[B-1]       " UPDATE BEXP WITH BINARY SCALING FACTOR
BEXP=A
DEXP+B                     " UPDATE DEXP
MULS(TENPOW[B-1])          " MULT BY POWER OF TEN
S=HEAD
MULAS(TENPOW[B-1])         " COMPLETE DOUBLE PRECISION MULT
LCAS(1),P                  " NORMALIZE, WAS IT ALREADY NORMALIZED?
Y,RCAS(1)                   " THEN RESTORE IT AS IT WAS
HEAD=A
A=BEXP                     " BINARY EXPONENT
N,A-1                       " CORRECT FOR NORMALIZING SHIFT

MULT:     U,A+SCALE[B-1],P   " DONE WITH MULTIPLYING?
N,GOTO(:MUL)
B-1,P
Y,GOTO(:MULT)
B=-A                       " REMAINS OF BINARY EXPONENT
U,B-3,Z                     " IS IT POSSIBLE TO MULT BY 10/16?
N,GOTO(:AH)
TAIL=S
MULS(TENPOW)                " MULT BY TEN
S=HEAD
MULAS(TENPOW)               " FINISH DOUBLE PRECISION MULT
LCAS(1),E                   " MULT BY TWO, WAS THAT TOO MUCH?
Y,B*1
Y,DEXP+B                     " UPDATE DEXP
N,S=TAIL
N,A=HEAD
N,RUAS(B)                   " NORMALIZE DECIMAL MANTISSA
HEAD=-A                     " STORE POSITIVE
TAIL=-S                     " DECIMAL MANTISSA
S=-0
S=DEXP                      " ZERO IS NOW -0
DEXP=S                      " STORE DEXP WITH CORRECT SIGN
B=FIXFLO,Z                  " FLO?
Y,S=-0
S+MM,P                      " DEXP+MM
M[0]=S                      " STORE IN TWO
M[1]=S                      " COUNTER LOCATIONS
A=HALF                      " AS := .5
S=0                         " FOR ROUNDING
N,JUMP(6)                   " AVOID DIVISIONS

```

```

RU:      RUAS(4)          " )
        DIVAS(TENPOW)    " )
        TEMP=S          " ) .5 * 10 + (-DEXP - MM)
        DIVA(TENPOW)    " )
        A=TEMP          " )
        REPOP(:RU)      " )
        S+TAIL,P        " ROUNDED THE TAIL. NO CARRY?
        N,S'+SGNBIT     " REMOVE THE OVERFLOWN BIT
        N,A+1           " AND CARRY IT OVER TO HEAD
        A+HEAD,P        " ROUNDED HEAD. NO OVERFLOW?
        N,A=1
        N,DEXP+A        " CORRECT DECIMAL EXPONENT FOR OVERFLOW
        N,M[1]+B        " CORRECT COUNTER IF FIX
        N,A=TENTH       " MANTISSA IS
        N,S=TENTH[1]    " .1 IN CASE OF OVERFLOW
        HEAD=A          " STORE ROUNDED
        TAIL=S          " MANTISSA

        S=DEXP
        B=BSAFE         " RESTORE STACK POINTER
        U,S=FIXFLO,Z    " FLO?
        N,JUMP(7)
        A=COUNT        " NUMBER OF EXPONENT DIGITS
        A+:NTAB[-1]
        U,S=MA,P        " EXPONENT OVERFLOW?
        N,S=-S
        N,S=MA,P        " OR EXPONENT UNDERFLOW?
        Y,GOTO(:SPILL[-1]) " HANDLE EXPONENT SPILL
        GOTO(:SHRTCT)    " SHORTCUT FOR FLO
        S=NN,P          " DEXP-NN>0?
        Y,GOTO(:SPILL[-1]) " OVERFLOW IN FIX
        A=DEXP,P
        N,GOTO(:NDBTP)   " NO DIGITS BEFORE THE POINT
        M[0]=-S,Z       " NUMBER OF LEADING SPACES TO COUNTER
        M[1]=A          " NUMBER OF DIGITS BEFORE THE POINT
        Y,JUMP(3)        " IF NO SPACES AT ALL

        S=14,P          " SPACE. CONDITION = YES
        SUBC(:STORE)     " STORE SPACE
        REPOP(:GPONE)
GPONE:
DIGITS:  S=TAIL          " )
        TENAS           " )
        TAIL=S          " ) CALCULATE
        S=HEAD          " ) AND
        MULAS(10)       " ) STORE
        HEAD=S          " ) DIGITS
        MG=A            " )
        F+1             " )
        REP1P(:DIGITS)  " )
        N,GOTO(:RET)    " AFTER DIGITS AFTER THE POINT
SHRTCT:  A=MM,Z
        Y,GOTO(:RET)    " DONE IF NO DIGITS AFTER THE POINT
        M[1]=A          " NUMBER OF DIGITS AFTER POINT TO COUNTER
        S=12
        SUBC(:STORE)    " STORE DECIMAL POINT
        GOTO(:DIGITS)   " FOR DIGITS AFTER THE POINT

```

RET:	S=14	" SPACE
	SUBC(:STORE)	" FILL IN TRAILING SPACE
	CBASE=S	" FILL IN LEADING SPACE
	U,S=ABS,Z	" ABS VERSION?
	Y,GOTOR(MC[-1])	" EXIT CONV
	A=:CBASE	" INITIALIZE SEARCH FOR FIRST NON-SPACE
	U,S=MA[1],Z	" IS CHARACTER A SPACE?
	Y,A+1	" THEN INCREMENT SEARCH POSITION
	Y,JUMP(-3)	" AND REPEAT
	S=XX[1],P	" IS THE NUMBER POSITIVE?
	Y,S=10	" +
	N,S=11	" -
	MA=S	" STORE SIGN OVER LAST LEADING SPACE
	GOTOR(MC[-1])	" EXIT CONV
NDBTP:	M[3]=-A	" NUMBER OF ZEROES AFTER THE POINT
	A=MM,Z	" NUMBER OF PLACES AFTER POINT. NONE?
	Y,GOTO(:IZERO)	" PRODUCE AN INTEGER ZERO
	S=NN,Z	" NUMBER OF SPACES BEFORE THE POINT
	Y,JUMP(5)	" IN CASE NO SPACES
	M[2]=S	" STORE NUMBER OF SPACES IN COUNTER
	S=14	" SPACE
STSP:	SUBC(:STORE)	" STORE SPACE
	REP2P(:STSP)	
	Y,GOTO(:IZERO3)	" WHEN BEING USED FOR INTEGER ZERO
	A=M[1],P	" SIGNIFICANT DIGITS AFTER POINT?
	N,S=0	
	N,HEAD=S	" CREATE AN
	N,TAIL=S	" ARTIFICIAL ZERO
	N,GOTO(:SHRTCT)	" AND CONVERT IT
	S=DEXP,Z	
	Y,GOTO(:SHRTCT)	
STZ:	S=12	" DECIMAL POINT
	SUBC(:STORE)	" STORE DECIMAL POINT OR DIGIT ZERO
	S=0	" DIGIT ZERO
	REP3E(:STZ)	
	GOTO(:DIGITS)	
IZERO:	S=NN	
	S=1,P	" NN-1>0?
	Y,GOTO(:STSP[-2])	" PRODUCE NN-1 SPACES
IZERO3:	S=0	
	SUBC(:STORE)	" STORE SINGLE DIGIT ZERO
	GOTO(:RET)	

```

GEN:      A=F,P
          N,A=-F
          A'=-32767,Z
          A=13
          Y,S=0
          N,S=3
          ABS=A
          N,SUBC(:FLO1)
          N,GOTOR(MC[-1])
          SUBC(:FIX1)
          S=6
          M[0]=S
          S=14
          SUBC(:STORE)
          REPOP(:FILL)
          GOTOR(MC[-1])
          IEND:

          " ABSOLUTE VALUE OF HEAD OF F IN A
          " IS NUMBER INTEGER?
          " NN := 13
          " MM := 0 FOR FIX
          " MM := 3 FOR FLO
          " SIGNED VERSION
          " PERFORM FLOATING CONVERSION
          " AND EXIT GEN
          " PERFORM INTEGER CONVERSION
          " SIX SPACES EXTRA AFTER INTEGER
          " COUNTER LOCATION
          " SPACE
          " STORE SPACE
          " EXIT GEN
          " OUTPUT CONVERSION ROUTINES

FILL:

```

" DRUM SECTION

```

'BEGIN'      HERE, DRAR, DRSS, ATT WORD, LAST TURN, DRBUSY, DROK,
              DRNBK, DRMES, WAMES, HOK DR, DRSTART1, DRUM, DRNOK

"  ASSUMED TO BE DECLARED GLOBALLY ARE:
"  DREF, ACTIVE, INITDR, ATTENTION, DRSTART, READCOMP,
"  ENDTRANS, PROCESS6, WA INTRPT

"  ASSUMED TO BE DECLARED AND DEFINED GLOBALLY ARE:
"  DR, INTAB, BRUSH, SGNBIT, D18, D18M1, RESTORE, HOK,
"  CTYPE, WASTE TIME, PROCESS0, PROCESS1, PROCESS2,
"  PROCESS3, PROCESS4, PROCESS5, PROCESS7, PROCESS8,
"  PROCESS9, PROCESS10, PROCESS11, PROCESS12, PROCESS13,
"  PROCESS14, PROCESS15, PRESENCE, COMPDIR, BEGIN OF COMPILER,
"  LENGTH OF COMPILER

HERE:
INTAB[8+:DR]:      :DRUM
M[64+(4*:DR)]:     DRAR:

HERE[0]:           'SKIP' 1
DRSS:              -      0
                  'SKIP' 3

ATT WORD:          'SKIP' 1
LAST TURN:         'SKIP' 1
DRBUSY:            'SKIP' 1
DROK:              'SKIP' 1
DREF:              'SKIP' 5
ACTIVE:            'SKIP' 16

DRNBK:             '02 10 44 22 0'      " CR NL BLKLT D
                  '12 34 07 04 0'      " R U M SP
                  '06 23 36 77 0'      " N B K END
DRMES:             '02 10 44 22 0'      " CR NL BLKLT D
                  '12 34 07 04 0'      " R U M SP
                  '15 30 12 14 0'      " P A R I
                  '01 25 77 00 0'      " T Y END
WAMES:             '02 10 44 31 0'      " CR NL BLKLT W
                  '12 03 06 13 0'      " R O N G
                  '04 30 22 22 0'      " S P A D D
                  '12 20 24 24 0'      " R E S S
                  '77 00 00 00 0'      " END

```

```

INITDR:      S = 0
              ATT WORD = S          " NO REQUESTS FOR TRANSPORT
              LAST TURN = S
              DRBUSY = - B          " DRUM CHANNEL FREE
              DREF[1] = B          " NO SPECIAL TRANSPORT
              ACTIVE[7] = - B       " NO RECOVER
              ACTIVE[8] = - B       " NO DUMP
              DROK = B              " DRUM CK
HOK DR:      G = ('000 040 000'+:DR)
              SUBCD (:BRUSH)        " RESET DRUM
              S = :DRSS
              DRAR[0] = S           " LABEL
              GOTOR (MC[-1])        " EXIT INITDR

ATTENTION:   S '+' ATT WORD
              ATT WORD = S, Z      " NO ACTIVE PROCESSES ?
N, A = DRBUSY, P                    " OR DRUM CHANNEL OCCUPIED ?
Y, GOTOR (MC[-1])
              DRBUSY = - B          " SAVE STACK POINTER
              B = LAST TURN
              RCS (B+1), P
N, S '+' SGNBIT                      " CLEAR SIGN BIT
NORS                                " )
              S = LAST TURN        " )
              S = :MC[1], Z        " ) FIND NEXT VICTIM
N, S + 0, P                          " )
N, S + 27                            " )
              LAST TURN = S
              B = - DRBUSY          " RESTORE STACK POINTER
              JUMP (S)              " SWITCH OVER PROCESS NUMBER
              GOTO (:PROCESS0)
              GOTO (:PROCESS1)
              GOTO (:PROCESS2)
              GOTO (:PROCESS3)
              GOTO (:PROCESS4)
              GOTO (:PROCESS5)
              GOTO (:PROCESS6)
              GOTO (:PROCESS7)
              GOTO (:PROCESS8)
              GOTO (:PROCESS9)
              GOTO (:PROCESS10)
              GOTO (:PROCESS11)
              GOTO (:PROCESS12)
              GOTO (:PROCESS13)
              GOTO (:PROCESS14)
              GOTO (:PROCESS15)

" IT IS UNDERSTOOD THAT EACH PROCESS, ACTIVATED IN THIS WAY, SHALL
" END WITH THE FOLLOWING INSTRUCTIONS:

"      SUBC (:DRSTART)              " START TRANSPORT
"      GOTOR (MC[-1])              " EXIT PROCESS

```

```

DRSTART:      DRSS[1] = A      " DRUM ADDRESS
              DRSS[2] = S      " CCRE ADDRESS
              DRSS[3] = G      " BUFFER LENGTH
              A = M[B-1]      " LINK
              DRBUSY = A      " DRUM CHANNEL OCCUPIED
DRSTART1:     DRAR[1] = B      " INTERRUPT COUNTER = 0
              A = D18
              DRAR[2] = A      " ACTION COUNTER = 1
              ('760 070 000'+:DR) " START DRUM TRANSPORT
              GOTOR (MC[-1])    " EXIT DRSTART

DRUM:         ('660 071 000'+:DR) " REMOVE INTERRUPT SIGNAL
              S = DRSS[-1], Z   " OK ?
N, GOTO (:DRNOK)
              DROK = B          " DRUM CK
              A = DRBUSY
              DRBUSY = - A      " DRUM CHANNEL FREE AGAIN
              SUBC (:MA[1])     " REPORT COMPLETION OF TRANSPORT
              GOTO (:RESTORE)   " EXIT DRUM

" IT IS UNDERSTOOD THAT EACH REPORT OF COMPLETION SHALL END WITH
" THE FOLLOWING INSTRUCTION:

"          GOTO (:ATTENTION)

DRNCK:        'BEGIN' TRY AGAIN

              S = - S, P      " - COUNTING ERROR ?
N, SUBC (D18M1)          " GIVE UP
U, S '*' D18, Z          " NBK ?
N, JUMP (7)
              S = DROK, P      " FIRST OCCURRENCE ?
Y, SUBC (:CTYPE)        " TYPE "DRUM NBK"
              :DRNBK
              DROK = - B      " NOTE OCCURRENCE
TRY AGAIN:     SUBC (:HOK DR)  " RESET DRUM
              SUBC (:DRSTART1) " START DRUM TRANSPORT ANEW
              GOTO (:RESTORE)  " EXIT DRNOK
U, S '*' 4096, Z        " PARITY ERROR ?
Y, SUBC (:CTYPE)        " TYPE "DRUM PARITY"
              :DRMES
Y, SUBC (D18M1)          " GIVE UP
U, S '*' 8192, Z        " TIMING ERROR ?
N, SUBC (D18M1)          " GIVE UP
              GOTO (:TRY AGAIN)

              'END' DRNOK

```

" DRUM S

```

READ COMP:          S = PRESENCE, P
Y, GOTOR(MC[-1])    A = DRFICORIMBEG      " FIRST WORD ON DRUM
READ NEWCOMP:       S = :RUN SYST        " FIRST WORD IN CORE
                   S + COMPDIR           " CORE TODR ORDR TO CORE
                   G = LENGTH OF COMPILER
                   SUBC(:TRANSPORT)
                   A = DRTRAPABEG
                   S = BEGIN OF UP 32 K
                   S + COMPDIR
                   G = LENGTH OF UP 32 K
                   SUBC(:TRANSPORT)
                   A = 0                  " IF NOT EXPLICIT SAID
                   COMPDIR = A            " NEVER MORE CORE
                   GOTOR(MC[-1])          " TO DRUM

```

```

PROCESS6:          A = 4096              " MAXIMUM TRANSPORT LENGTH
                   DREF[4] = A, P        " ≤ JOB LENGTH ?
N, A + DREF[4]
                   F = :MA               " NUMBER OF WORDS
                   S = DREF[2]           " FIRST WORD IN CORE
                   DREF[2] + A           " INCREMENT
                   PLUSA (DREF[3])       " INCREMENT
                   A = :MG               " FIRST WORD ON DRUM
                   SUBC (:DRSTART)       " START TRANSPORT
                   GOTOR (MC[-1])

```

" AFTER COMPLETION OF THE TRANSPORT, CONTROL IS GIVEN TO
 " THE NEXT INSTRUCTIONS:

```

                   A = DREF[4], P        " TRANSPORT TO BE CONTINUED ?
                   ACTIVE[6] = A
Y, S = 0           " CONTINUE ACTIVITY
N, A = DREF
N, S = MA          " ALLOW
N, MA = - S        " REFERENCE TO THE ARRAY
N, DREF[1] = - S   " FORBID WRONG ADDRESS INTERRUPTS
N, S = 64
                   GOTO (:ATTENTION)

```

WA INTRPT:

```

U, A '+' DREF[1], Z      "INTERRUPT ADMISSABLE
N, B = BEGIN OF STACK
N, SUBC(:CTYPE)          "TYPE WRONG ADDRESS
    :WAMES
    A = M[22]              "LINK
N, A '+' D18M1
N, PRESENCE = - A
    '760 061 000'         "ALLOW WRONG ADDRESS INTERRUPTS
N, GOTO(:FINIS1)         "TRYTO START ANOTHER JOB
    '760 060 000'         "ALLOW 110 INTERRUPTS
    A = 1                 "DECREMENT LINK
    M[22] = A
    SUBC(:WASTE TIME)
    A = DREF[1]
    GOTOR(M[22])          "TRY AGAIN
END RECDU:
    A = ACTIVE[8], P      "DUMP READY
N, A = ACTIVE[7], P      "RECOVER READY
Y, JUMP(-3)
    GOTOR(MC[-1])

```

TRANSPORT:

```

DREF[3] = A              "FIRST WORD ON DRUM
DREF[2] = S              "FIRST WORD IN CORE
DREF[4] = G              "TRANSPORT LENGTH
A = : PRESENCE
DREF[0] = A
PRESENCE = - A
    '660 060 000'         "PREVENT INTERRUPTS
DREF[1] = - A           "ALLOW WRONG ADDRESS INTERRUPTS
ACTIVE[6] = B           "NOTE ACTIVITY OF PROCESS 6
S = 64
SUBC(:ATTENTION)        "ASK FOR TRANSPORT
    '760 060 000'         "ALLOW INTERRUPTS AGAIN
ENDTRANS:
    A = DREF[1], P
N, A = MA
    GOTOR(MC[-1])

```

'END'

100470 -

1

40

PROCESS9:
PROCESS10:
PROCESS11:
PROCESS12:
PROCESS13:
PROCESS14:
PROCESS15:

SUBC (D18M1)

" GIVE UP


```

:CRBUF42
CRBUF41: 'SKIP' :IBULEN
:CRBUF41
CRBUF42: 'SKIP' :IBULEN

:CRBUF52
CRBUF51: 'SKIP' :IBULEN
:CRBUF51
CRBUF52: 'SKIP' :IBULEN

:IBUF2
IBUF1: 'SKIP' (:BFL+1)
:IBUF1
IBUF2: 'SKIP' (:BFL+1)

IMES: '02 10 44 12 0' " CR NL BLKLT R
      '20 30 22 20 0' " E A D E
      '12 04 06 23 0' " R SP N B
      '36 77 00 00 0' " K END
      '02 10 44 14 0' " CR NL BLKLT I
      '06 15 34 01 0' " N P U T
      '04 01 30 15 0' " SP T A P
      '20 77 00 00 0' " E END

```

```

INITRE:      'BEGIN'  LOOP1, LOOP2, LOOP3

              DONE = B           " NO TAPE IN TAPE READER
              IBUSY = - B        " READER INACTIVE
              DFDSEGN = - B      " ) NO
              TRNSEGN = - B      " ) MESSAGES
              A = DRBEG4
              DRBUNOF4 = A        " ) SELECT FIRST
              DRBUNOE4 = A        " ) DRUM PAGE
              A = :MAX4
              DRROOM4 = A        " DRUM EMPTY
              A = 0
              QUEU4 = A          " ) NO QUEUES OF BUFFERS
              QUEU5 = A          " ) WAITING FOR TRANSPORT
              ACTIVE[4] = - B     " ) NO ACTIVE
              ACTIVE[5] = - B     " ) PROCESSES
              S = :CRBUF41
              CRBUNOE4 = S        " ) SELECT FIRST
              CRBUNOF4 = S        " ) CORE BUFFER
LOOP1:        A = MS[-1], P      " ) ASSURE PRESENCE
              N, MS[-1] = - A     " ) OF BUFFER
              SUBC (:FROM DRUM)  " ASK FOR TRANSPORT
              S = CRBUNOE4        " SELECT NEXT BUFFER
              U, S = :CRBUF41, Z  " CYCLE CLOSED ?
              N, GOTO (:LOOP1)
              S = :CRBUF51
              CRBUNOE5 = S        " ) SELECT FIRST
              CRBUNOF5 = S        " ) SECONDARY BUFFER
LOOP2:        REPOS = S         " INITIALIZE REPOS
              MS = B              " NO MESSAGE IN BUFFER
              A = MS[-1], P      " ) ASSURE PRESENCE
              N, MS[-1] = - A     " ) OF BUFFER
              S = MS[-1]         " SELECT NEXT BUFFER
              U, S = CRBUNOE5, Z  " CYCLE CLOSED ?
              N, GOTO (:LOOP2)
              S = (256* :ITIM)
              ITEL = S           " INITIALIZE ITEL
              S = :IBUF1
              IBUNOE = S          " ) SELECT FIRST
              IBUNOF = S          " ) PRIMARY BUFFER
LOOP3:        A = MS[-1], P      " ) ASSURE ABSENCE
              Y, MS[-1] = - A     " ) OF BUFFER
              S = - MS[-1]       " SELECT NEXT BUFFER
              U, S = IBUNOE, Z    " CYCLE CLOSED ?
              N, GOTO (:LOOP3)
              G = ('000 040 000'+:RE)
              SUBCD (:BRUSH)      " RESET TAPE READER
              S = :ISS
              IAR[0] = S          " FIRST LABEL
              GOTO (:ISTART)      " START READER

              'END'  INITRE

```

```

REHEP:          'BEGIN' REHEP1

                S = READY, P
N, SUBC (:ANALYZE)
A = IPOS
RUAS (2)
A = MA[1]
RUS (24), Z
Y, S = 1
N, JUMP (S)
RUA (9)
RUA (9)
'660 060 000'
IPOS + S
U, A '*' 256, Z
A '*' 255
ICHECK + A
REHEP1: Y, S = :MA
Y, GOTOR (MC[-1])
MC = A
SUBCD (:FROM DRUM)
A = MC[-1], P
GOTO (:REHEP1)

                'END' REHEP

ANALYZE:        SUBC (:INQUIRE)
Y, READY = B
Y, GOTOR (MC[-1])
A = 998
SUBC (:ERRORM)
GOTO (:END RUN)

INQUIRE:       S = CRBUNOE4
A = MS[-1], P
N, SUBC (:WAIT)
A = MS, P
Y, ATLMSGN = A
N, LCA (4)
N, ANEW = A
GOTO (MC[-1])

```

```

" ALL OK ?
" ELSE ANALYZE
" POSITION IN BUFFER
" SEPARATE WORD ADDRESS
" TAKE BUFFER WORD
" SHIFT INDICATION
" INCREMENT FOR IPOS

" SHIFT OUT
" SHIFT OUT
" PREVENT INTERRUPTS
" INCREMENT IPOS
" STILL MORE CHAR. IN BUFFER ?
" CLEAN AWAY EXTRANEIOUS BITS
" ADD TO SUMCHECK
" COPY CHARACTER IN S
" EXIT REHEP
" SAVE CHARACTER
" ASK FOR REFILL
" RESTORE CHARACTER. COND YES
" EXIT REHEP

```

```

" INPUT CONTINUED ?
" THEN ALL OK
" REPORT END OF INPUT
" SWITCH OVER TO NEXT PROGRAM

```

```

" SELECT BUFFER
" PRESENT ?
" WAIT FOR ARRIVAL
" INPUT CONTINUED ?
" RECENT VALUE OF A/T AND L/M

" SAVE MESSAGE
" EXIT INQUIRE

```

```

WAIT:          A = - DONE, P          " READER ACTIVE ?
N, A = ACTIVE[5], P          " v TRANSPORT TO DRUM ?
N, A = ACTIVE[4], P          " v TRANSPORT FROM DRUM ?
N, A = MS[-1], P            " v ARRIVED ?
N, SUBCD (:CTYPE)          " TYPE "INPUT TAPE"
      :REMES
      S = CRBUNOE4          " SELECT BUFFER
      A = MS[-1], P          " PRESENT YET ?
Y, GOTOR (MC[-1])          " EXIT WAIT
      SUBC (:WASTE TIME)      " WAIT FOR A WHILE
      JUMP (-5)              " AND TRY AGAIN

ISKIP:         SUBC (:INQUIRE)        " INPUT CONTINUED ?
                SUBCD (:FROM DRUM)     " ASK FOR NEXT BUFFER
N, GOTOR (MC[-1])          " END OF INPUT
                GOTO (:ISKIP)          " CCNTINUE ISKIP

FROM DRUM:     S = CRBUNOE4          " SELECT BUFFER
                A = MS[-1]             " ) BUFFER NOT
                MS[-1] = - A           " ) PRESENT
                CRBUNOE4 = A           " SELECT NEXT BUFFER
                LUA (2)
                IPOS = A              " INITIALIZE IPCS
                READY = - A            " STILL ANALYSIS REQUIRED
                A = 1                  " ) ONE BUFFER MORE
                QUEU4 + A              " ) IN QUEUE
                A = ACTIVE[4], P        " TRANSPORT ACTIVE ?
N, A = DRROOM4
N, A = :MAX4, Z            " v NO BUFFER IN STOCK ?
Y, GOTOR (MC[-1])          " THEN POSTPONE TRANSPORT
START OUT:     ACTIVE[4] = B          " NOTE ACTIVITY
                S = 16
                GOTO (:ATTENTION)      " ACTIVATE TRANSPORT

```

PROCESS4:
DR TO CR:

```

      A = DRBUNOE4          " )
      A + :IBULEN          " ) DETERMINE
U, A = DREND4, Z          " ) FIRST OCCUPIED PLACE
Y, A = DRBEG4             " ) OF DRUM
      DRBUNOE4 = A         " )
      S = CRBUNOF4         " FIRST WORD IN CORE
      F = :IBULEN          " NUMBER OF WORDS
      SUBC (:DRSTART)      " START TRANSPORT
      GOTOR (MC[-1])       " EXIT DR TO CR

```

" AFTER COMPLETION OF THE TRANSPORT, CONTROL IS GIVEN TO
" THE NEXT INSTRUCTIONS:

```

      A = ACTIVE[5], P     " REVERSE TRANSPORT ACTIVE ?
N, A = QUEU5, Z           " OR NO REQUESTS FOR TRANSPORT ?
N, SUBC (:START IN)       " ACTIVATE REVERSE PROCESS
      S = CRBUNOF4         " SELECT BUFFER
      A = MS[-1]          "
      MS[-1] = - A        " BUFFER PRESENT AGAIN
      CRBUNOF4 = - A      " SELECT NEXT BUFFER
      S = 1
      QUEU4 = S, Z        " QUEUE SHORTER. EMPTY NOW ?
      PLUS (DRROOM4)      " MORE FREE SPACE ON DRUM
N, S = :MAX4, Z           " DRUM EMPTY ?
Y, ACTIVE[4] = - B        " NOTE INACTIVITY
Y, S = 16                 " CANCEL ATTENTION
N, S = 0                  " CONTINUE ATTENTION
      GOTO (:ATTENTION)

```

PROCESS5:
CR TO DR:

```

      A = DRBUNOF4      " )
      A + :IBULEN      " ) DETERMINE
U,  A = DREND4, Z      " ) FIRST FREE PLACE
Y,  A = DRBEG4         " ) OF DRUM
      DRBUNOF4 = A      " )
      S = CRBUNOE5      " FIRST WORD IN CORE
      S + D18          " CORE TO DRUM INDICATION
      F = :IBULEN      " NUMBER OF WORDS
      SUBC (:DRSTART)   " START TRANSPORT
      GOTOR (MC[-1])    " EXIT CR TO DR

```

" AFTER COMPLETION OF THE TRANSPORT, CONTROL IS GIVEN TO
" THE NEXT INSTRUCTIONS:

```

      A = ACTIVE[4], P  " REVERSE TRANSPORT ACTIVE ?
N,  A = QUEU4, Z       " OR NO REQUESTS FOR TRANSPORT ?
N,  SUBC (:START OUT)  " ACTIVATE REVERSE PROCESS
      S = CRBUNOE5      " SELECT BUFFER
      A = MS[-1]        "
      MS[-1] = - A      " BUFFER PRESENT AGAIN
      CRBUNOE5 = - A    " SELECT NEXT BUFFER
      S = 1
      QUEU5 = S, Z      " QUEU SHORTER. EMPTY NOW ?
      MINS (DRROOM4)    " LESS FREE SPACE ON DRUM
N,  S = :MS, Z         " DRUM SPACE EXHAUSTED ?
Y,  ACTIVE[5] = - B    " NOTE INACTIVITY
Y,  S = 32             " CANCEL ATTENTION
N,  S = 0              " CONTINUE ATTENTION
      SUBC (:ATTENTION)
      A = TRNSGN, P     " MESSAGE WAITING FOR TRANSPORT ?
      S = CRBUNOF5      " SELECT BUFFER
Y,  MS = - A           " PUT MESSAGE INTO IT
Y,  TRNSGN = - A       " EXTINGUISH SIGNAL
Y,  GOTO (:TO DRUM)    " ASK FOR TRANSPORT
      A = DONE, P      " DONE WITH TAPE ?
Y,  GOTO (:ISTART)     " ASK FOR A NEW ONE
      A = IBUNOE
      A = MA[-1], P    " INPUT WAITING FOR TRANSPORT ?
N,  GOTOR (MC[-1])     " EXIT DRUM INTERRUPT
      SUBC (:CONTRACT)
      A = IBUNOE
      A = MA[-1], P    " MORE INPUT WAITING FOR TRANSPORT ?
Y,  JUMP (-4)
      GOTO (:ISTART)   " ASK FOR MORE INPUT

```

```

TO DRUM:      S = CRBUNOF5      " SELECT SECONDARY BUFFER
               A = MS[-1]
               MS[-1] = - A      " BUFFER NOT PRESENT
               CRBUNOF5 = A      " SELECT NEXT BUFFER
               REPOS = A         " INITIALIZE REPOS
               A = (256*:ITIM)
               ITEL = A          " INITIALIZE ITEL
               A = 1
               QUEU5 + A         " ONE BUFFER MORE IN QUEUE
               A = ACTIVE[5], P  " TRANSPORT ACTIVE ?
N, A = DRROOM4, Z               " OR DRUM SPACE EXHAUSTED ?
Y, GOTOR (MC[-1])               " POSTPCNE TRANSPORT
START IN:     ACTIVE[5] = B      " NCTE ACTIVITY
               S = 32
               GOTO (:ATTENTION) " ASK FOR TRANSPORT

I START:      'BEGIN' I START12, I START13

               A = IBUSY, P      " READER IN ACTION ?
Y, GOTOR (MC[-1])
I START2:     A = DONE, P        " REQUEST FOR A NEW TAPE ?
               Y, A = I START13  " )
I START4:     N, A = I START12   " ) FORM CODE WORD
               A + IBUNOF        " )
               ISS[1] = A        " STORE CODE WORD
               A = D18
               IAR[2] = A        " ACTION COUNTER = 1
               IAR[1] = B        " INTERRUPT COUNTER = 0
               ('760 070 000'+:RE) " START READER
               IBUSY = B         " READER IN ACTION NOW
               GOTOR (MC[-1])    " EXIT I START
I START12:    ('001 000 000'+:BFL)
I START13:    ('001 000 000'+* 2)

               'END' I START

READER:      ('660 071 000'+:RE) " CLEAR INTERRUPT BIT
               S = IBUNOF        " ) SAVE POSITION OF
               G = IAR[3]        " ) LAST CHARACTER READ
               MS = G            " ) IN TOP OF BUFFER
               IBUSY = - B       " READER INACTIVE
               A = IAR[0], P     " OK ?
N, GOTO (:INOK)
               A = DONE, P      " FIRST INTERRUPT AFTER EOT ?
Y, DONE = - B                   " NOTE PRESENCE OF NEW TAPE
Y, A = ATLMMSGN1                " FIX VALUE OF
Y, ATLMMSGN2 = A                " A/T AND L/M SIGNALS
Y, SUBC (:I START2)             " START READER FOR FIRST INFORMATION
Y, GOTO (:RESTORE)             " EXIT READER
               A = - MS[-1]     " ) BUFFER FILLED
               MS[-1] = A       " ) WITH INPUT
               IBUNOF = A       " SELECT NEXT BUFFER
               S = MA[-1], P    " FILLED WITH INPUT ?
N, SUBC (:I START4)             " START READER AGAIN
N, SUBC (:CONTRACT)
               GOTO (:RESTORE)  " EXIT READER

```

```

INOK:      RUA (18), Z      " NBK ?
           Y, SUBC (:CTYPE) " TYPE "READER NBK"
           :IMES
           Y, SUBC (D18M1)  " GIVE UP
           A = DONE, P      " FIRST INTERRUPT AFTER EOT ?
           MS = - G         " MARK LAST BUFFER
           N, A = MS[-1]    " ) BUFFER FILLED
           N, MS[-1] = - A  " ) WITH INPUT
           N, IBUNOF = - A  " SELECT NEXT BUFFER
           A = :IAR[1]
           SUBC (:HOK)      " RESET TAPE READER
           Y, SUBC (:ISTART2) " REJECT
           N, SUBC (:CONTRACT) " ACCEPT
           GOTO (:RESTORE)  " EXIT INOK

CONTRACT:  'BEGIN' CONTRACT1

           S = CRBUNOF5     " SELECT SECONDARY BUFFER
           A = MS[-1], P    " PRESENT ?
           N, GOTOR (MC[-1]) " POSTPONE CONTRACTION
           G = IBUNOE       " SELECT PRIMARY BUFFER
           A = MG, P        " NOT LAST PORTION OF TAPE ?
           N, A = - A
           S = 256
           N, ITEL = - S    " INDICATE SPECIAL CASE
           Y, ITEL = S, P   " MORE PLACE IN SECONDARY BUFFER ?
           N, MA + S        " MARK LAST CHARACTER
           DONE = - B       " SAVE STACK POINTER
           B = REPOS        " POSITION IN SECONDARY BUFFER
           A = :BFL         " NUMBER OF WORDS
           S = MG[1]        " ) STORE
           LUS (9)          " ) THREE WORDS
           S + MG[2]        " ) OF PRIMARY BUFFER
           LCS (9)          " ) IN
           S + MG[3]        " ) ONE WORD
           MC[1] = S        " ) OF SECONDARY BUFFER
           F + 3            " ADVANCE
           A = 3, Z        " COUNT DOWN
           N, GOTO (:CONTRACT1)
           REPOS = B        " NEW POSITION IN SECONDARY BUFFER
           B = - DONE       " RESTORE STACK POINTER
           A = IBUNOE       " SELECT PRIMARY BUFFER
           S = MA[-1]
           MA[-1] = - S     " BUFFER EMPTY AGAIN
           IBUNOE = S       " SELECT NEXT BUFFER
           S = ITEL, P      " NORMAL CASE ?
           Y, GOTOR (MC[-1]) " EXIT CONTRACT
           G = CRBUNOF5     " SELECT SECONDARY BUFFER
           A = ATLMSGN2     " PUT FIXED VALUE OF A/T AND L/M
           MG = A           " IN TOP OF BUFFER
           S + 256, Z       " END OF TAPE ?
           N, GOTO (:TO DRUM) " ASK FOR TRANSPORT AND EXIT
           SUBC (:TO DRUM)  " ASK FOR TRANSPORT
           DONE = B         " DONE WITH TAPE
           A = DFDSGN       " TAKE DEFERRED SIGNAL
           DFDSGN = - B     " DELETE DEFERRED SIGNAL

```

```
CONTRACT2:      TRNSGN = A, P      " MESSAGE WAITING ?
                  G = CRBUNOF5      " SELECT SECONDARY BUFFER
Y, S = MG[-1], P  " PRESENT ?
N, GOTOR (MC[-1]) " PCSTPCNE TRANSMISSION
                  MG = - A          " PUT MESSAGE INTO BUFFER
                  TRNSGN = - A      " EXTINGUISH SIGNAL
                  GOTO (:TO DRUM)   " ASK FOR TRANSPORT AND EXIT

                  'END' CONTRACT

ENSTART:        S = DONE, P        " IN BETWEEN TWO TAPES ?
N, DFDSGN = A    " PUT MESSAGE INTO DEFERRED SIGNAL
N, GOTO (:RESTORE) " EXIT
U, A '+' TRNSGN, Z " IDENTICAL TO PREVIOUS MESSAGE ?
N, JUMP (1)
U, A '*' D18, Z  " ESTART ?
N, SUBC (:CONTRACT2) " ASK FOR TRANSPORT
                  GOTO (:RESTORE) " EXIT

'END'
```

" PUNCH SECTION

' BEGIN '

HERE, PAR, PSS, PPOS, SHIFT, CRBUNOF2, CRBUNOE2,
QUEU2, DRBUNOF2, DRBUNOE2, DRROOM2, QUEU3, PBUNOF,
PBUNOE, PBUSY, CHANGE, PHOK, CCNT, FLCASE, THEAD,
CRBUF1, CRBUF2, PBUF1, PEUF2, PUNBK, TAPMES, FLEXTB,
FLBLNK, AFXP6, GET BUF, GET :BUF1, TO DRUM, START IN,
CR TO DR, DR TO CR, FROM DRUM, START OUT, PSTART,
PSTART3, PSTART4, PSTART5, PCHER, PCHER1, PUNOK

```

" ASSUMED TO BE DECLARED GLOBALLY ARE:
" SUMCHECK, HEPDNT, DTS, INITPU, SIGNON, SIGNOFF,
" PUHEP, PROCESS2, PROCESS3, PUNCH, FIXP, ABSFIXP,
" FLOP, PO, FLEXHP, FLEXIS

" ASSUMED TO BE DECLARED AND DEFINED GLOBALLY ARE:
" INTAB, PU, DRBEG2, DREND2, MAX2, PBULEN, ACTIVE,
" ATTENTION, DRSTART, CRITICAL, END CRIT, DATE,
" SERIAL, D18, CTYPE, HOK, RESTORE, BRUSH, GEN,
" FIX, FLO, FIX60, FIX1, ABS, CBASE, HEP, INTREP

```

HERE:

```
INTAB[8+:PU]:
M[64+(4*:PU)]:
```

: PCHER
PAR:

```
HERE(0):      'SKIP' 1
PSS:          -      0
              'SKIP' 1
```

```

SUMCHECK:                'SK|P'| 1
HEPCNT:                  'SK|P'| 1
PPOS:                    'SK|P'| 1
SHIFT:                   'SK|P'| 1
CRBUNOF2:                'SK|P'| 1
CRBUNOE2:                'SK|P'| 1
QUEUE2:                  'SK|P'| 1
DRBUNOF2:                'SK|P'| 1
DRBUNOE2:                'SK|P'| 1
DRROOM2:                 'SK|P'| 1
QUEUE3:                  'SK|P'| 1
PBUNOF:                  'SK|P'| 1
PBUNOE:                  'SK|P'| 1
PBUSY:                   'SK|P'| 1
CHANGE:                  'SK|P'| 1
PHOK:                   'SK|P'| 1
DTS:                    'SK|P'| 1
CCNT:                   'SK|P'| 1
FLCASE:                  'SK|P'| 1

```

```

THEAD:      'SKIP' 24

:CRBUF2
CRBUF1:     'SKIP' :PBULEN
:CRBUF1
CRBUF2:     'SKIP' :PBULEN

:PBUF2
PBUF1:      'SKIP' :PBULEN
:PBUF1
PBUF2:      'SKIP' :PBULEN

PUNBK:      '02 10 44 15 0'    " CR NL BLKLT P
            '34 06 16 05 0'    " U N C H
            '04 06 23 36 0'    " SP N E K
            '77 00 00 00 0'    " END
TAPMES:     '02 10 44 15 0'    " CR NL BLKLT P
            '34 06 16 05 0'    " U N C H
            '04 20 07 15 0'    " SP E M P
            '01 25 77 00 0'    " T Y END

FLEXTB:     + 32              " 0
            + 1              " 1
            + 2              " 2
            + 19             " 3
            + 4              " 4
            + 21             " 5
            + 22             " 6
            + 7              " 7
            + 8              " 8
            + 25             " 9
            +112             " +
            + 64             " -
            +107            " .
            + 59            " "
            + 16            " SPACE

```

```

INITPU:      'BEGIN'  LOOP1, LOOP2

              DTS = B           " RESET PUNCH SIGNAL
              CHANGE = B        " NEW PROGRAM SIGNAL
              PHOK = - B        " END OF TAPE SIGNAL
              A = DRBEG2        " )
              DRBUNOF2 = A       " ) SELECT FIRST DRUM PAGE
              DRBUNOE2 = A       " )
              A = :MAX2
              DRROOM2 = A        " DRUM EMPTY
              A = 0
              QUEU2 = A          " ) NO QUEUS OF BUFFERS
              QUEU3 = A          " ) WAITING FOR TRANSPORT
              SHIFT = A
              PPOS = A
              ACTIVE[2] = - B    " ) NO ACTIVE
              ACTIVE[3] = - B    " ) PROCESSES
              PBUSY = - B        " PUNCH NOT ACTIVE
              S = :PBUF1
              PBUNOF = S         " SELECT FIRST PUNCH BUFFER
              F = :MS
LOOP1:        A = MG[-1], P      " ) ASSURE PRESENCE
              N, MG[-1] = - A    " ) OF BUFFER
              SUBC (:FROM DRUM) " ASK FOR TRANSPORT
              G = PBUNOE        " SELECT NEXT BUFFER
              U, S = G, Z        " CYCLE CLOSED ?
              N, GOTO (:LOOP1)
              S = :CRBUF1
              CRBUNOE2 = S       " SELECT FIRST CORE BUFFER
              CRBUNOF2 = S
LOOP2:        A = MS[-1], P     " ) ASSURE PRESENCE
              N, MS[-1] = - A    " ) OF BUFFER
              S = MS[-1]        " SELECT NEXT BUFFER
              U, S = CRBUNOE2, Z " CYCLE CLOSED ?
              N, GOTO (:LOOP2)
              G = ('000 040 000'+:PU)
              SUBCD (:BRUSH)     " RESET PUNCH
              S = :PSS
              PAR[0] = S         " LABEL TO START TRIPLE
              GOTOR (MC[-1])     " EXIT INITPU

              'END'  INITPU

```

SIGNON:

```

S = 0
HEPCNT = S
SUMCHECK = S
FLCASE = S      " FLEXHP CASE UNDEFINED
A = (2+1PBULEN[-1])
SUBC (:FLBLNK)   " PUNCH BLANKS
S = 26
SUBC (:FLEXHP)   " PUNCH NLGR
A = 120
SUBC (:FLEXIS)   " PUNCH APOSTROPHE
G = DATE
SUBC (:AFXP6)    " CONVERT AND PUNCH DATE
A = 65
SUBC (:FLEXIS)   " PUNCH -
G = SERIAL
SUBC (:AFXP6)    " CONVERT AND PUNCH SERIAL
A = 120
SUBC (:FLEXIS)   " PUNCH APOSTROPHE
S = 26
FLCASE = S      " FLEXHP CASE UNDEFINED AGAIN
SUBC (:FLEXHP)   " PUNCH NLGR
CCNT = B
A = :PBULEN[-3] " MORE THAN 150 CHARACTERS YET
PPOS = A
S = 127
GOTO (:PUHEP)    " PUNCH ERASE AND EXIT SIGNON

```

SIGNOFF:

```

S = HEPCNT, Z      " NO PUNCHINGS DONE ?
Y, GOTO (:GET BUF1) " CLEAR BUFFER
S = 127
SUBC (:PUHEP)      " PUNCH ERASE
A = 25
SUBC (:FLBLNK)     " PUNCH 25 BLANKS
S = 11
SUBC (:PUHEP)      " PUNCH STOPCODE
A = 150
SUBC (:FLBLNK)     " PUNCH 150 BLANKS
A = PPOS, Z
Y, SUBC (:FLBLNK)  " ANOTHER 150 BLANKS
A = D18
PPOS = A
GOTO (:GET BUF)    " MARK LAST BUFFER
" FORCE PUNCHING AND EXIT SIGNOFF

```

FLBLNK:

```

COUNT = A
S = 0
SUBC (:PUHEP)      " PUNCH BLANK
REPP (:FLBLNK[1]) " COUNT DOWN
GOTOR (MC[-1])     " EXIT FLBLNK

```

AFXP6:

```

A = 6      " NN
S = 0      " MM
ABS = S    " UNSIGNED VERSION
SUBC (:FIX1) " FIX CONVERSION
GOTO (:PO) " PUNCH OUT

```

PUHEP:

'BEGIN' PUHEP1, PUHEP2

```

CRITICAL = B          " PREVENT KICK OFF
S '*' 255             " CLEAN AWAY EXTRANEIOUS BITS
SUMCHECK + S          " ADD TO SUM CHECK
A = 1
PLUSA (PPOS)          " INCREMENT PPOS
U, A = :PBULEN[-1], Z " LAST POSITION OF BUFFER ?
A + CRBUNOP2          " SELECT BUFFER
JUMP (SHIFT)          " SWITCH OVER SHIFT
MA = S                " SHIFT = 0: STORE CHARACTER
GOTO (:PUHEP2)
LUS (9)               " SHIFT = 2: SHIFT TO RIGHT POSITION
GOTO (:PUHEP1)
LUS (18)              " SHIFT = 4: SHIFT TO RIGHT POSITION
MA + S                " ADD CHARACTER TO BUFFER WORD
PUHEP1:
PUHEP2: N, GOTO (:END CRIT) " EXIT PUHEP
S = 2
U, SHIFT = S, P       " SHIFT ALREADY 4 ?
Y, SUBC (:GET BUF)    " GET ANOTHER BUFFER
N, SHIFT + S          " INCREMENT SHIFT
S = 0
PPOS = S              " PPOS = 0
GOTO (:END CRIT)     " EXIT PUHEP

'END' PUHEP

```

```

GET BUF:      A = PPOS
              S = SHIFT, Z
N, A + :PBULEN[-1]
N, S = 2, Z
N, A + :PBULEN[-1]
              HEPNT + A
              G = CRBUNOF2
              MG = A
              SUBCD (:TO DRUM)
              G = CRBUNOF2
              S = MG[-1], P
N, JUMP (-2)
GET BUF1:     S = 0
              SHIFT = S
              PPOS = S
              GOTOR (MC[-1])

TO DRUM:      A = MG[-1]
              MG[-1] = - A
              CRBUNOF2 = A
              A = 1
              QUEU2 + A
              A = ACTIVE[2], P
N, A = DRROOM2, Z
Y, GOTOR (MC[-1])
              ACTIVE[2] = B
START IN:    S = 4
              GOTO (:ATTENTION)

" )
" ) FORM
" ) NUMBER OF CHARACTERS
" ) OF BUFFER IN A
" )
" CCUNT NUMBER OF CHARACTERS
" SELECT BUFFER
" LENGTH OF BUFFER
" ASK FOR TRANSPORT
" SELECT NEXT BUFFER
" PRESENT ?
" WAIT FOR ARRIVAL
" INITIALIZE SHIFT
" PPOS = 0
" EXIT GET BUF
" BUFFER NOT PRESENT
" SELECT NEXT BUFFER
" ONE BUFFER MORE IN QUEU
" TRANSPORT ACTIVE ?
" OR DRUM SPACE EXHAUSTED ?
" POSTPONE TRANSPORT
" NOTE ACTIVITY
" ASK FOR TRANSPORT

```

PROCESS2:
CR TO DR:

A = DRBUNOF2	")
A + :PBULEN	") DETERMINE
U, A = DREND2, Z	") FIRST FREE PLACE
Y, A = DRBEG2	") ON DRUM
DRBUNOF2 = A	")
S = CRBUNOE2	" FIRST WORD IN CORE
S + D18	" INDICATION CORE TO DRUM
F = :PBULEN	" NUMBER OF WORDS
SUBC (:DR START)	" START TRANSPORT
GOTOR (MC[-1])	" EXIT CR TO DR

" AFTER COMPLETION OF THE TRANSPORT, CONTROL IS GIVEN TO
" THE NEXT INSTRUCTIONS:

A = ACTIVE[3], P	" REVERSE TRANSPORT ACTIVE ?
N, A = QUEU3, Z	" OR NO REQUEST FOR TRANSPORT ?
N, SUBC (:START OUT)	" ACTIVATE REVERSE PROCESS
S = CRBUNOE2	" SELECT BUFFER
A = MS[-1]	
MS[-1] = - A	" BUFFER PRESENT AGAIN
CRBUNOE2 = - A	" SELECT NEXT BUFFER
S = 1	
QUEU2 = S, Z	" QUEU SHORTER. NOW EMPTY ?
MINS (DRROOM2)	" LESS FREE SPACE ON DRUM
N, S = :MS, Z	" DRUM SPACE EXHAUSTED ?
Y, ACTIVE[2] = - B	" NOTE INACTIVITY
Y, S = 4	" CANCEL ATTENTION
N, S = 0	" CONTINUE ATTENTION
GOTO (:ATTENTION)	

PROCESS3:
DR TO CR:

A = DRBUNOE2	")
A + :PBULEN	") DETERMINE
U, A = DREND2, Z	") FIRST OCCUPIED WORD
Y, A = DRBEG2	") ON DRUM
DRBUNOE2 = A	")
S = PBUNOF	" FIRST WORD IN CORE
F = :PBULEN	" NUMBER OF WORDS
SUBC (:DR START)	" START TRANSPORT
GOTOR (MC[-1])	" EXIT DR TO CR

" AFTER COMPLETION OF THE TRANSPORT, CONTROL IS GIVEN TO
" THE NEXT INSTRUCTIONS:

A = ACTIVE[2], P	" REVERSE TRANSPORT ACTIVE ?
N, A = QUEUE2, Z	" OR NO REQUEST FOR TRANSPORT ?
N, SUBC (:START IN)	" ACTIVATE REVERSE PROCESS
S = PBUNOF	" SELECT BUFFER
A = MS[-1]	
MS[-1] = - A	" BUFFER PRESENT AGAIN
PBUNOF = - A	" SELECT NEXT BUFFER
A = PBUSY, P	" PUNCH ACTIVE ?
N, SUBC (:PSTART)	" START PUNCH
S = 1	
QUEUE3 = S, Z	" QUEUE SHORTER. NOW EMPTY ?
PLUS (DRROOM2)	" MORE FREE SPACE IN DRUM
N, S = :MAX2, Z	" DRUM EMPTY ?
Y, ACTIVE[3] = - B	" NOTE INACTIVITY
Y, S = 8	" CANCEL ATTENTION
N, S = 0	" CONTINUE ATTENTION
GOTO (:ATTENTION)	

```

FROM DRUM:      A = MG[-1]
                 MG[-1] = - A      " BUFFER NOT PRESENT
                 PBUNOE = A        " SELECT NEXT BUFFER
                 A = 1
                 QUEU3 + A        " ONE BUFFER MORE IN QUEU
                 A = ACTIVE[3], P  " TRANSPORT ACTIVE ?
N, A = DRROOM2
N, A = :MAX2, Z   " OR DRUM EMPTY ?
Y, GOTOR (MC[-1]) " POSTPONE TRANSPORT
START OUT:      ACTIVE[3] = B      " NOTE ACTIVITY
                 S = 8
                 GOTO (:ATTENTION) " ASK FOR TRANSPORT

PSTART:         'BEGIN' PSTART1, PSTART2

                 PBUSY = B        " NOTE ACTIVITY
                 G = PBUNOE       " SELECT BUFFER
                 A = CHANGE, P    " FIRST BUFFER OF NEW PROGRAM ?
N, GOTO (:PSTART2)
                 S = :MG[-24]     " )
                 B = :THEAD       " ) TRANSPORT
PSTART1:        F = MS[25]        " ) DATE
                 MC = F           " ) AND SERIAL NUMBER
                 S + 2            " ) TO THEAD
U, S = PBUNOE, Z   " )
N, GOTO (:PSTART1) " )
                 B = PBUSY       " RESTORE STACK POINTER
                 G = :MS         " SELECT BUFFER ANEW
PSTART2:        A = MG
                 CHANGE = - A, P  " LAST BUFFER OF CURRENT PROGRAM ?
Y, A + D18        " REMOVE INDICATION
PSTART3:        A = :PBULEN[-1], P " MORE THAN ONE SHIFT ?
                 MG = A
Y, A = :PBULEN[-1] " COMPLETE SHIFT
N, A + :PBULEN[-1] " INCOMPLETE SHIFT
PSTART4:        LUA (18)
                 A + :MG         " CODE WORD
                 PSS[1] = A
PSTART5:        A = D18
                 PAR[2] = A      " ACTION COUNTER = 1
                 PAR[1] = B      " INTERRUPT COUNTER = 0
                 ('760 070 000'+:PU) " START PUNCH
                 GOTOR (MC[-1])  " EXIT PSTART

                 'END' PSTART

```

```

PCHER:          'BEGIN' PCHER2

                ('660 071 000'+:PU) " REMOVE INTERRUPT BIT
                G = PBUNOE           " SELECT BUFFER
                S = PAR[0], P        " OK ?
N, GOTO (:PUNOK)
                A = MG, P           " BUFFER NOT EMPTY ?
Y, GOTO (:PCHER1)
                SUBC (:FROM DRUM)   " ASK FOR TRANSPORT
                A = CHANGE, P       " LAST BUFFER OF PROGRAM ?
Y, A = DTS, P                       " A RESETTING WANTED ?
Y, G = ('000 040 000'+:PU)
Y, SUBC (:BRUSH)                     " RESET PUNCH
                G = PBUNOE           " SELECT NEXT BUFFER
                S = MG[-1], P        " PRESENT ?
Y, SUBC (:PSTART)
N, PBUSY = - B                       " NOTE INACTIVITY
                GOTO (:RESTORE)      " EXIT PCHER
PCHER1:          A = :PBULEN[-1], P  " STILL MORE THAN ONE SHIFT ?
                MG = A
Y, A = :PBULEN[-1]                   " COMPLETE SHIFT
N, A = :PBULEN[-1]                   " INCOMPLETE SHIFT
                MC = A               " SAVE LENGTH OF SHIFT
                PBUSY = B            " SAVE STACK POINTER
                B = :MG[1]           " )
                S = M[B]             " ) SHIFT
PCHER2:          RUS (9)              " ) BUFFER WORDS
                MC = S               " ) TO THE RIGHT
                A = 1, P            " )
Y, GOTO (:PCHER2)                    "
                B = PBUSY            " RESTORE STACK POINTER
                A = MC[-1]           " RESTORE LENGTH OF SHIFT
                SUBC (:PSTART4)      " START PUNCH
                GOTO (:RESTORE)      " EXIT PCHER

                'END' PCHER

```

```

PUNCK:      'BEGIN' PUNOK1, PUNOK2, PUNCK3, PUNOK4

            RUS (18), Z          " NBK ?
Y, SUBC (:CTYPE)                " TYPE "PUNCH NBK"
            :PUNBK
Y, SUBC (D18M1)                 " GIVE UP
S + 1, Z                        " NCK1 ?
Y, A = :PAR[1]
Y, SUBC (:HOK)                  " GO OVER INTO NOK2
Y, SUBC (:PSTART5)              " REOFFER FOR PUNCHING
Y, GOTO (:RESTORE)              " END NOK1 HANDLING
A = MG, P                       " BUFFER NOT EMPTY ?
Y, GOTO (:PCHER1)               " CONTINUE PUNCHING
S = PHOK, P                     " SECOND TIME NOK2 ?
PHOK = - S                      " RESET INDICATOR
PBUSY = B                       " SAVE STACK POINTER
B = :MG                         " SELECT BUFFER
Y, GOTO (:PUNOK2)
A = :PBULEN                      " )
F = 0                           " )
PUNCK1:    MC = F                " ) FILL BUFFER WITH BLANKS
A = 2, P                         " )
Y, GOTO (:PUNOK1)               " )
B = PBUSY                       " RESTORE STACK POINTER
G = PBUNOE                      " SELECT BUFFER ANEW
A = 127
MG[1] = A                       " ERASE
A = 11
MG[23] = A                      " STOPCCDE
GOTO (:PUNOK4)
PUNCK2:    A = 127
            " ) STOPCODE AS
LUA(18)                          " ) ENDMARKER OF HEPTADS
MG[127] = A                      " ) PUNCHED BY SYSTEM
S = :THEAD                      " )
            A = 12
PUNCK3:    F = MS
            MC[1] = F
            A = 1, P
Y, S + 2                         " ) TRANSPORT
Y, GOTO (:PUNOK3)               " ) DATE
B = PBUSY                       " ) AND
A = :PAR[1]                     " ) SERIAL NUMBER
SUBC (:HOK)                     " ) TO BUFFER
SUBC (:CTYPE)                   " )
            :TAPMES              " RESTORE STACK POINTER
G = PBUNOE                      " GO OVER INTO OK
PUNCK4:    A = (3*:PBULEN[-1])  " TYPE "PUNCH EMPTY"
SUBC (:PSTART3)                 " SELECT BUFFER ANEW
GOTO (:RESTORE)                 " PUNCH BUFFER
                                " EXIT PUNOK

            'END' PUNOK

```

```
PUNCH:      SUBC(:GEN)      " PERFORM GENERAL CONVERSION
             GOTO(:PO)      " GC AND GET IT PUNCHED OUT

FIXP:       S=1             " FOR SIGNED VERSION
             ABS=S          " STORE SIGN INDICATOR
             SUBC(:FIX)     " PERFORM FIXED-POINT CONVERSION
             GOTO(:PO)      " GC AND GET IT PUNCHED OUT

ABSFIXP:    S=0             " SIGNLESS VERSION
             GOTO(:FIXP[1])

FLOP:       ABS=B          " SIGNED VERSION
             SUBC(:FLO)     " PERFORM FLOATING-POINT CONVERSION

PO:         F=:CBASE        " NUMBER OF CHARACTERS IN CBASE
             COUNT=G        " STORE IN COUNTER LOCATION
             G+CCNT         " ADD POSITION ON FLEXOWRITER LINE
             F=150,P        " BEYOND 150 CHARACTERS ON THE LINE?
             Y,S=26         " THEN A CARRIAGE RETURN FIRST
             Y,SUBC(:FLEXHP) " OUTPUT THE NLCR
             S=122          " LOWER CASE
             U,S=FLCASE,Z   " WAS LAST PUNCHED CASE LOWER CASE?
             N,SUBC(:FLEXHP) " IF NOT, OUTPUT A LOWER CASE
             A=:CBASE       " INITIALIZE LOOP
SMA:        S=MA            " CHARACTER IN SPEC. INT. REPRES.
             S+:FLEXTB      " BASE ADDRESS CONVERSION TABLE
             S=MS           " CHARACTER IN FLEXOWRITER CODE
             MC=A           " SAVE A
             SUBC(:FLEXHP)  " OUTPUT CHARACTER
             A=MC[-1]       " RESTORE A
             A+1            " INCREMENT TO NEXT CHARACTER
             REPP(:SMA)     " COUNT AND RETURN
             GOTOR(MC[-1])  " EXIT PUNCH, FIXP, ABSFIXP, FLOP
```

FLEXHP:

'BEGIN' COMPTB, NLCR, TAB, LC, LC

" ASSUMED TO BE DECLARED AND DEFINED GLOBALLY ARE:
 " HEP, PUHEP, INTREP, CCNT, FLCASE

```

HEP=S          " SAVE THE HEPTAD
SUBC(:PUHEP)    " BUFFER IT OUT TO THE PUNCH
S=HEP          " RETRIEVE IT
U,S'-'-127,Z   " NOT MORE THAN SEVEN BITS?
N,S=127        " ELSE NO FLEXOWRITER SYMBOL
S+:INTREP      " BASE ADDRESS CONVERSION TABLE
S=MS,P        " ELEMENT FROM TABLE. NO SPECIAL HANDLING?
Y,A=1         " IN NORMAL CASE,
Y,CCNT+A       " CCNT := CCNT + 1
Y,GOTOR(MC[-1]) " AND EXIT FLEXHP
LUS(10)        " SHIFT OUT PUTTEXT BITS
RUS(10)        " SHIFT BACK
S+:COMPTB      " BASE ADDRESS JUMP TABLE
GOTO(S)        " JUMP TO HANDLE SPECIAL CASE
                GOTOR(MC[-1]) "PUHEB(31)
GOTOR(MC[-1])  "PUHEB(27)
GOTO(:NLCR)
GOTO(:TAB)
GOTOR(MC[-1])  " UNDERLINE OR VERTICAL BAR, EXIT
GOTO(:LC)
GOTO(:UC)
GOTOR(MC[-1])  " ERASE, BLANK, STOPCODE, BACKSPACE
GOTOR(MC[-1])  " NON-FLEXOWRITER CHARACTER
GOTOR(MC[-1])  " EVEN CHARACTER
S=0
CCNT=S        " SET CCNT BACK TO ZERO
GOTOR(MC[-1])  " AND EXIT FLEXHP
S=CCNT
S+9
S'-'-7        " MULTIPLE OF EIGHT
GOTO(:NLCR[1])
S=122        " LOWER CASE
FLCASE=S     " ASSIGN NEW CASE TO FLCASE
GOTOR(MC[-1])  " AND EXIT FLEXHP
S=124        " UPPER CASE
GOTO(:LC[1])
'END' FLEXHP

```

COMPTB:
NLCR:

TAB:

LC:

UC:

```

FLEXIS:      MC = S      "SAVE S-REGISTER
              A '*' 127  "REMOVE EXTRANEIOUS BITS
              A + :INTREP "CONVERSION TABLE
              S = MA     "CONVERSION WORD
              A = 124    "UPPER CASE
              LCS(10), P "LOWER CASE NOT REQUIRED?
N, A = 122      "LOWER CASE
              RCS(1), E  "CASE INDIFFERENT?
              S '*' 127  "ISOLATE CHARACTER
Y, JUMP(6)      "PUNCH CHAR WITHOUT CASE
U, A = FLCASE, Z "REQUIRED CASE SAME AS CURRENT?
Y, JUMP(4)      "PUNCH CHAR WITHOUT CASE
              MC = S     "SAVE CHARACTER
              S = A
              SUBC(:FLEXHP) "PUNCH PRECEDING CASE
              S = MC[-1]    "RESTORE CHARACTER
              SUBC(:FLEXHP) "PUNCH CHARACTER
              S = MC[-1]    "RESTORE S
              GOTOR(MC[-1]) "EXIT FLEXIS

```

100470 -

1

65

'END'

" PUNCH SECTION

" LINE PRINTER SECTION, INCLUDING DRUM TRANSPORT

'BEGIN'

HERE, PRAR, EMPTY, VERTCON, PAGCNT, BUFPOS, BUFBEQ,
 BUFROOM, CRBUNOE, DRROOM, QUEUE IN, QUEUE OUT,
 DRBUNOF, DRBUNOE, PRBUSY, PRPARF, PRBUNOF, PRBUNOE,
 PRSS, HEAD, LINE1, LINE2, LINE3, PRPCS, PRPOS2,
 PRPOS3, CRBUF1, CRBUF2, CRBUF3, PRBUF1, PRBUF2,
 NBKMEs, PARMES, TORNMS, YOKMEs, PRNTMS, CONTAB,
 PRNTAB, PRHEX1, OFFER LINE, OFFER LINE1, HEADING,
 GET BUF, TO DRUM, START IN, CR TO DR, FROM DRUM,
 START OUT, DR TO CR, START PR, START PR3, PRINTER,
 PRNOK, AMS, TAB3

" ASSUMED TO BE DECLARED GLOBALLY ARE:

" LINE NUMBER, INITPR1, INITPR2, INITPR3, PRHEX,
 " NEW PAGE, NLCR, CARRIAGE, PRINT, FIXT, ABSFIXT,
 " FLOT, PRO, PRNTIS, DERRORM, ERRORM, NXTTAPESL,
 " TAB, SPACE, AFXT6, AFXT

" ASSUMED TO BE DECLARED AND DEFINED GLOBALLY ARE:

" INTAB, PR, DR BEG, DR END, MAX, BUFLTH, CRITICAL,
 " END CRIT, NO SWAP, EXIT NO SWAP, DATE, SERIAL,
 " RND, D18, D18M1, CTYPE, HOK, RESTORE, GEN, FIX,
 " FLO, CBASE, DANGEROUS, ERRONEOUS, RUNNUMBER, SMES,
 " FIXT60, WANTED, LINECOUNTER, LASTSYMBOL,
 " VALUE OF CONSTANT, ENDRUN, LAST IDENTIFIER, REHEP,
 " INTREP, CASE, ABS, FIX1, PC, ACTIVE, ATTENTION,
 " DRSTART

HERE:

INTAB[8+:PR]: :PRINTER
 M[64+(4*:PR)]: PRAR:

HERE[0]:

EMPTY:	'SKIP' 1
VERTCON:	'SKIP' 1
LINE NUMBER:	'SKIP' 1
PAGCNT:	'SKIP' 1
BUFPOS:	'SKIP' 1
BUFBEQ:	'SKIP' 1
BUFROOM:	'SKIP' 1
CRBUNOE:	'SKIP' 1
DRROOM:	'SKIP' 1
QUEUE IN:	'SKIP' 1
QUEUE OUT:	'SKIP' 1
DRBUNOF:	'SKIP' 1
DRBUNOE:	'SKIP' 1
PRBUSY:	'SKIP' 1
PRPARF:	'SKIP' 1
PRBUNOF:	'SKIP' 1
PRBUNOE:	'SKIP' 1
PRSS:	'SKIP' 1

```

HEAD:          'SKIP' 6

LINE1: PRPOS:  'SKIP' 39
LINE2: PRPOS2: 'SKIP' 39
LINE3: PRPOS3: 'SKIP' 39

:CRBUF2
CRBUF1:        'SKIP' :BUFLTH
               'SKIP' 1
:CRBUF3
CRBUF2:        'SKIP' :BUFLTH
               'SKIP' 1
:CRBUF1
CRBUF3:        'SKIP' :BUFLTH
               'SKIP' 1

:PRBUF2
PRBUF1:        'SKIP' :BUFLTH
:PRBUF1
PRBUF2:        'SKIP' :BUFLTH

NBKMS:         '02 10 44 15 0'
               '12 14 06 01 0'
               '20 12 04 06 0'
               '23 36 77 00 0'
PARMES:         '02 10 44 15 0'
               '12 14 06 01 0'
               '20 12 04 15 0'
               '30 12 14 01 0'
               '25 77 00 00 0'
TORNMS:        '02 10 44 15 0'
               '30 15 20 12 0'
               '04 23 12 20 0'
               '30 36 77 00 0'
YOKMS:         '02 10 44 25 0'
               '03 36 20 04 0'
               '03 15 20 06 0'
               '77 00 00 00 0'
PRNTMS:        '02 10 44 15 0'
               '12 14 06 01 0'
               '20 12 04 20 0'
               '07 15 01 25 0'
               '77 00 00 00 0'

" CR NL BLKLT P
" R I N T
" E R SP N
" B K END
" CR NL BLKLT P
" R I N T
" E R SP P
" A R I T
" Y END
" CR NL BLKLT P
" A P E R
" SP B R E
" A K END
" CR NL BLKLT Y
" O K E SP
" O P E N
" END
" CR NL BLKLT P
" R I N T
" E R SP E
" M P T Y
" END

```

CONTAB:

+16;	+17;	+18;	+19;
+20;	+21;	+22;	+23;
+24;	+25;	+33;	+34;
+35;	+36;	+37;	+38;
+39;	+40;	+41;	+42;
+43;	+44;	+45;	+46;
+47;	+48;	+49;	+50;
+51;	+52;	+53;	+54;
+55;	+56;	+57;	+58;
-0;	-33;	-34;	-35;
-36;	-37;	-38;	-39;
-40;	-41;	-42;	-43;
-44;	-45;	-46;	-47;
-48;	-49;	-50;	-51;
-52;	-53;	-54;	-55;
-56;	-57;	-58;	-0;
+11;	+13;	+10;	+15;
-0;	-0;	+29;	-0;
+59;	-0;	+60;	-0;
+63;	-0;	-0;	+61;
+62;	-0;	-0;	-0;
-0;	-0;	-0;	+12;
+14;	+6;	+3;	+4;
-0;	+0;	+30;	+1;
+2;	-0;	+8;	+9;
+26;	+27;	-0;	-0;
-0;	-0;	-0;	-0;
-0;	-0;	-0;	-0;
-0;	-0;	-0;	-0;
-0;	-0;	-65;	-66;
+31;	+7;	+5;	+0;
+3;	+9;	-32;	-28;

PRNTAB:

+16	" 0
+17	" 1
+18	" 2
+19	" 3
+20	" 4
+21	" 5
+22	" 6
+23	" 7
+24	" 8
+25	" 9
+11	" +
+13	" -
+14	" ,
+6	" "
-1	" SPACE

```

INITPR1:      'BEGIN'  LOOP1, LOOP2, LOOP3

              F = 0
              PRPARF = - B          " NC PARITY ERRORS YET
              PRBUSY = - B          " LINE PRINTER NOT ACTIVE
              ACTIVE[0] = - B       " TRANSPORT TO DRUM NOT ACTIVE
              ACTIVE[1] = - B       " TRANSPORT TO CORE NOT ACTIVE
              A = DR BEG             " )
              DRBUNOF = A            " ) SELECT FIRST PAGE ON DRUM
              DRBUNOE = A            " )
              A = :MAX
              DRROOM = A             " FREE SPACE = MAX
              QUEU IN = G            " ) NO QUEUS OF BUFFERS
              QUEU OUT = G           " ) WAITING FOR TRANSPORT
              S = :PRBUF1
              PRBUNOF = S             " SELECT FIRST PRINT BUFFER
LOOP1:        A = MS[-1], P          " ) ASSURE PRESENCE
              N, MS[-1] = - A        " ) OF BUFFER
              SUBC (:FROM DRUM)      " ASK FOR TRANSPORT
              S = PRBUNOE            " SELECT NEXT BUFFER
              U, S = PRBUNOF, Z      " CYCLE CLOSED ?
              N, GOTO (:LOOP1)
              S = :CRBUF1
              CRBUNOE = S            " SELECT FIRST CORE BUFFER
LOOP2:        A = MS[-1], P          " ) ASSURE PRESENCE
              N, MS[-1] = - A        " ) OF BUFFER
              S = MS[-1]             " SELECT NEXT BUFFER
              U, S = CRBUNOE, Z      " CYCLE CLOSED ?
              N, GOTO (:LOOP2)
              BUFBEQ = S
              BUFPOS = S
              MS = G                 " NUMBER OF LINES = 0
              A = :BUFLTH            " ROOM = BUFFER LENGTH
              BUFROOM = A            " SELECT FIRST LINE BUFFER
              A = :LINE1             " 3 LINE BUFFERS
              S = 58                 " OF 39 WORDS EACH
              COUNT = S              " STORE 8 BLANKS IN BUFFER
LOOP3:        MA[1] = F              " STEP UP
              A + 2                  " COUNT DOWN
              REPP (:LOOP3)          " )
              S = 1                  " ) POSITION ON LINE
              PRPOS = S              " ) AT BEGIN OF LINE
              PRPOS2 = S             " )
              PRPOS3 = S             " )
              EMPTY = B              " NC SYMBOLS ON LINE YET
              LINE NUMBER = S        " LINE NUMBER = 1
              G = ('000 040 000'+:PR)
              SUBCD (:BRUSH)         " RESET LINE PRINTER
              GOTOR (MC[-1])         " EXIT INITPR1

              'END'  INITPR1

```

```

INITPR2:      S = EMPTY, P      " LINE STILL EMPTY ?
N, F = 0
N, SUBC (:CARRIAGE)      " CARRIAGE (0)
G = DATE
SUBC (:AFXT6)      " CONVERT DATE
S = 13
SUBC (:PRHEX)      " FOLLOWED BY "-"
G = SERIAL
SUBC (:AFXT6)      " CONVERT SERIAL NUMBER
A = :LINE1      " SELECT FIRST LINE BUFFER
F = 70
HEAD = G      " POSITION ON LINE = 70
G = MA[1]      " )
HEAD[1] = G      " ) TRANSPORT
F = MA[2]      " ) RESULT
HEAD[2] = F      " ) OF CONVERSION
F = MA[4]      " ) TO HEAD
HEAD[4] = F      " )
F = 1
PRPOS = G      " POSITION ON LINE AT BEGIN
F = 0      " )
MA[1] = G      " ) CLEAR
MA[2] = F      " ) PRIMARY LINE BUFFER
MA[4] = F      " )
PAGCNT = G      " PAGE NUMBER = 0
EMPTY = B      " LINE EMPTY AGAIN
GOTO (:NEW PAGE)      " START NEW PAGE AND EXIT INITPR2

INITPR3:      S = 1
U, S = LINE NUMBER, Z      " AT TOP OF NEW PAGE ?
Y, S = PRPOS, Z      " ^ BEGINNING OF NEW LINE ?
Y, GOTOR (MC[-1])      " EXIT INITPR3
GOTO (:NEW PAGE)      " START NEW PAGE AND EXIT INITPR3

PRINT OFF:    SUBC (:NLCR)      " EMPTY LINE BUFFER
SUBC (:TAB)      " FORCE HARDWARE
SUBC (:NLCR)      " TAB IN SECONDARY BUFFER
GOTO (:GET BUF)      " FORCE SOFTWARE

```

```

PRHEX:          'BEGIN' BUF, SECOND, SHFTAB

                A = PRPOS          " POSITION ON LINE
U, A = 144, P    " BEYOND THE END ?
Y, SUBC (:NLCR)  " START A NEW LINE
                EMPTY = - 8       " NOTE OCCURRENCE OF CHARACTER
PRHEX1:          F = :LINE1       " SELECT PRIMARY BUFFER
                A = PRPOS          " POSITION ON LINE
                S + 0, P          " ORDINARY CHARACTER ?
Y, GOTO (:BUF)   " BUFFER IT
U, S + 33, P     " SPECIAL CHARACTER ?
                S = - S
Y, GOTO (:SECOND) " HANDLE SPECIAL CASES
BUF:             MC = S           " SAVE CHARACTER
                S = 1
                MG + S            " INCREMENT POSITION ON LINE
                RUAS (2)          " WORD NUMBER IN A
                G + A             " WORD ADDRESS IN F
                RUS (24)
                S + :SHFTAB       " SELECT SHIFT INSTRUCTION
                A = MC[-1]        " RETRIEVE CHARACTER
                DO (MS)           " SHIFT TO RIGHT POSITION
                MG[1] + A         " ADD TO BUFFER WORD
Y, GOTOR (MC[-1]) " EXIT PRHEX
                A = PRPOS
                A = 1             " RESTORE OLD POSITION
                S = 61            " CAPITAL INDICATOR
SECOND:          F = :LINE2       " SELECT SECONDARY BUFFER
U, MG = A, P     " POSITION OCCUPIED ?
Y, F = :LINE3    " SELECT TERTIARY BUFFER
U, MG = A, P     " POSITION OCCUPIED ?
Y, GOTOR (MC[-1]) " THROW CHARACTER AWAY
                MG = A, P         " UPDATE POSITION ON LINE
                GOTO (:BUF)       " BUFFER SPECIAL CHARACTER
SHFTAB:          RCA ( 6)         " SHIFT OVER 21 PLACES
                LUA (14)         " SHIFT OVER 14 PLACES
                LUA ( 7)         " SHIFT OVER 7 PLACES
                A '*' 63         " NO SHIFT AT ALL

                'END' PRHEX

```

```

      'BEGIN'  CARRIAGE1, CARRIAGE2, CARRIAGE3

NEW PAGE:      A = 60
                GOTO (:CARRIAGE1)      " DC CARRIAGE (60)

NLCR:          A = 1
                GOTO (:CARRIAGE1)      " DC CARRIAGE (1)

CARRIAGE:      SUBC (:RND)              " ROUND PARAMETER
                A = G                  "
                U, A + 2, P            " PARAMETER OK ?
                N, A = 1               " HANDLE OUT OF RANGE
                U, A = 1, Z            " NEW PAGE ?
                Y, A = 60              " DC CARRIAGE (60)
CARRIAGE1:     CRITICAL = B            " PREVENT KICK OFF
                MC = S                 " SAVE S
                S = COUNT              "
                MC = S                 " SAVE COUNT
                S = EMPTY, P           " BLANK LINE ?
                Y, GOTO (:CARRIAGE2)
                EMPTY = B              " NEXT LINE STILL BLANK
                MC = A                 " SAVE PARAMETER
                A = VERTCON, Z          " VERTICAL CONTROL = 0 ?
                Y, A = 0               " AVOID = 0
                N, VERTCON = A          " CLEAR VERTICAL CONTROL
                SUBC (:OFFER LINE1)    " PRINT PRIMARY BUFFER
                U, A = PRPOS2, Z        " SECONDARY BUFFER EMPTY ?
                N, S = :LINE2
                N, SUBC (:OFFER LINE)  " PRINT SECONDARY BUFFER
                U, A = PRPOS3, Z        " TERTIARY BUFFER EMPTY ?
                N, S = :LINE3
                N, SUBC (:OFFER LINE)  " PRINT TERTIARY BUFFER
                A = MC[-1]              " RETRIEVE PARAMETER
CARRIAGE2:     S = A
                PLUS (LINE NUMBER)    " INCREMENT LINE NUMBER
                U, S = 60, P            " > 60 ?
                N, GOTO (:CARRIAGE3)
                A = 32
                SUBC (:OFFER LINE1)    " TURN TO NEXT PAGE
                SUBC (:GET BUF)        " START A NEW BUFFER
                SUBC (:HEADING)        " PUT HEADING INTO IT
                LINE NUMBER = A        " LINE NUMBER = 1 AFTER HEADING
                A = 2                  " DOUBLE SPACING AFTER HEADING
CARRIAGE3:     N, A + VERTCON          " VERTICAL CONTROL
                VERTCON = A            "
                RUA (5), Z             " ≤ 31 ?
                Y, S = MC[-1]          "
                Y, COUNT = S           " RESTORE COUNT
                Y, S = MC[-1]          " RESTORE S
                Y, GOTO (:END CRIT)    " EXIT NEW PAGE, NLCR, CARRIAGE
                A = 31
                SUBC (:OFFER LINE1)    " SPACE OVER 31 LINES
                A = - 31
                GOTO (:CARRIAGE3)      " ADJUST VERTCON

      'END'

```

```

      'BEGIN' LOOP
OFFER LINE1:  S = :LINE1          " ) PUT
              RCA (6)           " ) VERTICAL CONTROL INDICATION
              MS(1) + A         " ) INTC BUFFER
OFFER LINE:   A = MS            " POSITION ON LINE
              A + 15
              RUA (2)          " NUMBER OF WORDS REQUIRED
U, BUFROOM = A, P              " PLACE IN BUFFER ?
N, SUBC (:GET BUF)            " ELSE GET A NEW ONE
              BUFROOM = A      " COUNT DOWN
              NO SWAP = B      " PREVENT SWAPPING
              B = BUFPOS
              BUFPOS + A       " NEW LAST OCCUPIED WORD
              A = 3            " NUMBER OF WORDS
              MC(3) = A
              A + 1
              RUA (1)
              COUNT = A       " PREPARE LOOP
              A = 1
              MS = A           " POSITION ON LINE AT BEGIN
LOOP:         F = MS(1)        " ) TRANSPORT
              MC(3) = F        " ) WORDS FROM LINE TO BUFFER
              F = 0           " ) FILL LINE
              MS(1) = F        " ) WITH SPACES
              S + 2           " STEP UP
              REPP (:LOOP)     " COUNT DOWN
              B = NO SWAP      " RESTORE STACK POINTER
              S = BUFBEG
              MS + A           " COUNT LINE
              GOTO (:EXIT NO SWAP) " EXIT OFFER LINE
      'END'

```

HEADING:

'BEGIN' NXT DEC

A = D18
 S = BUFBEQ
 MS = A
 S = :LINE1
 F = HEAD
 MS = F
 F = HEAD[2]
 MS[2] = F
 F = HEAD[4]
 MS[4] = F
 A = 1
 PLUSA (PAGCNT)
 DIVA (10000)
 S + 1
 A = 4
 COUNT = A, P
 TENAS

" MARK BEGIN OF NEW PAGE

")

") TRANSPORT

") DATE AND

") SERIAL NUMBER

") TO PRIMARY BUFFER

")

" INCREMENT PAGE NUMBER

" PREPARE CONVERSION

" TO DECIMAL

" OF 4 DIGITS

NXT DEC:

MC = S
 Y, S = :MA, Z
 N, A + :CONTAB
 N, S = MA
 SUBC (:PRHEX1)
 S = MC[-1]
 REPP (:NXT DEC)
 S = :LINE1
 GOTO (:OFFER LINE)

" LEADING ZERO ?

" ENTRY OF CONVERSION TABLE

" PRINT DIGIT OR SPACE

" NEXT DECIMAL

" PRINT HEADING

'END' HEADING

GET BUF:

MC = A
 MC = S
 S = BUFBEQ
 SUBCD (:TO DRUM)
 S = BUFBEQ
 U, S = MS[-1], P
 N, JUMP (-2)
 BUFPOS = S
 A = 0
 MS = A
 A = :BUFLTH
 BUFROOM = A
 S = MC[-1]
 A = MC[-1]
 GOTOR (MC[-1])

" SAVE A

" SAVE S

" SELECT OLD BUFFER

" TRANSPORT TO DRUM

" SELECT NEXT BUFFER

" PRESENT ?

" WAIT FOR ARRIVAL

" NEW BUFFER STILL EMPTY

" NUMBER OF LINES = 0

" ROOM = BUFFER LENGTH

" RESTORE S

" RESTORE A

" EXIT GET BUF

```

TO DRUM:      A = MS[-1]
               MS[-1] = - A      " BUFFER NOT PRESENT
               BUFBEQ = A        " SELECT NEXT BUFFER
               A = 1
               QUEU IN + A      " ONE BUFFER MORE IN QUEU
               A = ACTIVE[0], P  " TRANSPORT ACTIVE ?
N, A = DRROOM, Z      " OR DRUM SPACE EXHAUSTED ?
Y, GOTOR (MC[-1])     " POSTPCNE TRANSPORT
START IN:      ACTIVE[0] = B    " NOTE ACTIVITY
               S = 1
               GOTO (:ATTENTION) " ASK FOR TRANSPORT

PROCESS0:
CR TO DR:      A = DRBUNOF      " )
               A + :BUFLTH      " ) DETERMINE
U, A = DR END, Z     " ) FIRST FREE PLACE
Y, A = DR BEG        " ) ON DRUM
               DRBUNOF = A      " )
               S = CRBUNOE      " FIRST WORD IN CORE
               S + D18          " INDICATOR CORE TO DRUM
               F = :BUFLTH      " NUMBER OF WORDS
               SUBC (:DRSTART)  " START TRANSPORT
               GOTOR (MC[-1])   " EXIT CR TO DR

" AFTER COMPLETION OF THE TRANSPORT, CONTROL IS GIVEN TO
" THE NEXT INSTRUCTIONS:

               A = ACTIVE[1], P  " REVERSE TRANSPORT ACTIVE ?
N, A = QUEU OUT, Z   " OR NO REQUESTS FOR TRANSPORT ?
N, SUBC (:START OUT) " ELSE ACTIVATE REVERSE PROCESS
               S = CRBUNOE      " SELECT BUFFER
               A = MS[-1]
               MS[-1] = - A      " BUFFER PRESENT AGAIN
               CRBUNOE = - A     " SELECT NEXT BUFFER
               S = 1
               QUEU IN = S, Z    " QUEU SHORTER. EMPTY NOW ?
               MINS (DRROOM)     " LESS FREE SPACE ON DRUM
N, S = :MS, Z        " DRUM SPACE EXHAUSTED ?
Y, ACTIVE[0] = - B   " NOTE INACTIVITY
Y, S = 1             " CANCEL ATTENTION
N, S = 0             " CONTINUE ATTENTION
               GOTO (:ATTENTION)

```

```

FROM DRUM:      A = MS[-1]
                 MS[-1] = - A      " BUFFER NOT PRESENT
                 PRBUNOE = A       " SELECT NEXT BUFFER
                 A = 1
                 QUEU OUT + A      " ONE BUFFER MORE IN QUEU
                 A = ACTIVE[1], P  " TRANSPORT ACTIVE ?
N, A = DRROOM
N, A = :MAX, Z   " OR FREE SPACE = MAX ?
Y, GOTOR (MC[-1]) " PCSTPONE TRANSPORT
START OUT:      ACTIVE[1] = B      " NCTE ACTIVITY
                 S = 2
                 GOTO (:ATTENTION) " ASK FOR TRANSPORT

PROCESS1:
DR TO CR:      A = DRBUNOE        " )
                A + :BUFLTH       " ) DETERMINE
U, A = DR END, Z " ) FIRST OCCUPIED WORD
Y, A = DR BEG    " ) ON DRUM
                DRBUNOE = A       " )
                S = PRBUNOF       " FIRST WORD IN CORE
                F = :BUFLTH       " NUMBER OF WORDS
                SUBC (:DRSTART)   " START TRANSPORT
                GOTOR (MC[-1])    " EXIT DR TO CR

" AFTER COMPLETION OF THE TRANSPORT, CONTROL IS GIVEN TO
" THE NEXT INSTRUCTIONS:

                A = ACTIVE[0], P  " REVERSE TRANSPORT ACTIVE ?
N, A = QUEU IN, Z " OR NO REQUESTS FOR TRANSPORT ?
N, SUBC (:START IN) " ELSE ACTIVATE REVERSE PROCESS
                S = PRBUNOF       " SELECT BUFFER
                A = MS[-1]
                MS[-1] = - A      " BUFFER PRESENT AGAIN
                PRBUNOF = - A     " SELECT NEXT BUFFER
                A = PRBUSY, P     " PRINTER ACTIVE ?
N, SUBC (:START PR) " ELSE START PRINTER
                S = 1
                QUEU OUT - S, Z   " QUEU SHORTER. EMPTY NOW ?
                PLUS (DRROOM)     " MORE FREE SPACE ON DRUM
N, S = :MAX, Z      " DRUM EMPTY ?
Y, ACTIVE[1] = - B  " NCTE INACTIVITY
Y, S = 2            " CANCEL ATTENTION
N, S = 0            " CONTINUE ATTENTION
                GOTO (:ATTENTION)

```

```

START PR:      'BEGIN'  START PR1, START PR2, LOOP

                A = COUNT
                MC = A          " SAVE COUNT
                G = PRBUNOE     " SELECT BUFFER
                A = MG          " NUMBER OF LINES
                                " AT BEGIN OF NEW PAGE ?
U, A = D18, P
N, GOTO (:START PR2)
U, A = PRSS, Z          " - PAPER LOW ?
Y, GOTO (:START PR1)
                SUBC (:CTYPE)  " TYP "PRINTER EMPTY"
                  :PRNTMS
                A = :PRAR[1]
                SUBC (:HOK)     " RESET PRINTER
                G = PRBUNOE     " SELECT BUFFER
                A = MG          " NUMBER OF LINES
START PR1:     A = D18          " REMOVE NEW PAGE INDICATION
START PR2:     COUNT = A, P     " PREPARE LOOP, CONDITION YES
                F + 2
START PR3:     PRAR[0] = G      " LABEL TO FIRST LINE
                A = - :MA[-1]   " )
                LUA (18)        " ) INTERRUPT COUNTER =
                A + 2            " ) - (NUMBER OF LINES - 1)
                PRAR[1] = A      " )
                A = COUNT        " )
                LUA (18)         " ) ACTION COUNTER =
                PRAR[2] = A      " ) NUMBER OF LINES
                                " LENGTH OF LINE
LOOP:          Y, S = MG[1]      " )
                Y, A = :MS       " )
                Y, LUA (18)      " ) CONSTRUCT CODEWORD
                Y, A + :MG[1]     " ) OUT OF LENGTH AND BEGIN ADDRESS
                Y, MG[1] = A      " )
                Y, S + :MG[3]     " FORM LABEL TO NEXT LINE
                Y, MG = S
                Y, F = :MS        " STEP UP
                Y, REPP (:LOOP)   " COUNT DOWN
                A = MC[-1]
                COUNT = A        " RESTORE COUNT
                PRBUSY = B       " PRINTER ACTIVE
                ('760 070 000'+:PR) " START LINE PRINTER
                GOTOR (MC[-1])    " EXIT START PR

                'END'  START PR

```

```

PRINTER:      ('660 071 000'+:PR)  " CLEAR PRINTER INTERRUPT
PRBUSY = - B   " PRINTER NOT BUSY
S = PRBUNOE    " SELECT BUFFER
A = PRAR[0], P " ALL OK ?
N, SUBC (:PRNOK)
N, GOTO (:RESTORE)
A = MA[-1]     " PRESERVE
PRSS = A       " PAPER LOW INDICATION
PRPARF = - B   " SET INDICATOR
SUBC (:FROM DRUM) " ASK FOR REFILL OF BUFFER
S = PRBUNOE    " SELECT NEXT BUFFER
U, S = MS[-1], P " PRESENT ?
Y, SUBC (:START PR) " START PRINTER AGAIN
GOTO (:RESTORE) " EXIT PRINTER

PRNCK:        'BEGIN' LOOP, TESTRN, REOFFER

RUA (18), Z    " NBK ?
Y, SUBC (:CTYPE) " TYPE "PRINTER NBK"
:NBKMES
Y, SUBC (D18M1) " MACHINE FAILURE
A = COUNT     " SAVE COUNT
MC = A        " NUMBER OF LINES
A = MS        " REMOVE NEW PAGE INDICATOR
A = :MA       " PREPARE LOOP
COUNT = A    " LABEL TO FIRST LINE
S + 2         " THIS LINE OK ?
LOOP:         A = MS[-1], P " SELECT NEXT LINE
Y, S = MS     " COUNT DOWN
Y, REPP (:LOOP)
Y, SUBC (D18M1) " MACHINE FAILURE
U, A '*' 2, Z  " - PARITY ERROR ?
Y, GOTO (:TESTRN)
A = PRPARF    " ) PARITY ERROR FOR SECOND TIME
A = :MS, Z    " ) IN SAME LINE ?
Y, SUBC (D18M1) " ) GIVE UP
PRPARF = S    " SET INDICATOR
SUBC (:CTYPE) " TYPE "PRINTER PARITY"
:PARMES
G = PRPARF    " LABEL TO FIRST LINE
GOTO (:REOFFER) " TRY AGAIN
PRPARF = - S  " SET INDICATOR
TESTRN:      U, A '*' 4, Z " - PAPER TORN ?
N, SUBC (:CTYPE) " TYPE "PAPER BREAK"
:TORNMS
Y, SUBC (:CTYPE) " TYPE "YOKE OPEN"
:YOKMES
A = :PRAR[1]
SUBC (:HOK)   " RESET PRINTER
G = - PRPARF " LABEL TO FIRST LINE
A = - 63     " )
RCA (6)      " ) CHANGE
A '*' MG[2]  " ) CARRIAGE CONTROL TO 0
MG[2] = A    " )
A = COUNT, Z " PREPARE RESTART, CONDITION NO
GOTO (:START PR3) " TRY AGAIN

REOFFER:
'END' PRNOK

```

```

PRINT:          SUBC(:GEN)      " PERFORM GENERAL CONVERSION
                GOTO(:PRO)      " PRINT OUT

FIXT:           S = 1           " SIGNED VERSION
                ABS = S         " SIGN INDICATOR
                SUBC(:FIX)      " FIXED-POINT CONVERSION
                GOTO(:PRO)      " PRINT OUT

ABSFIXT:        S = 0           " UNSIGNED VERSION
                GOTO(:FIXT[1])

FLOT:           ABS = B         " SIGNED VERSION
                SUBC(:FLO)      " FLOATING-POINT CONVERSION

PRO:            F = :CBASE      " NUMBER OF CHARACTERS
                COUNT = G
                G + PRPOS       " POSITION ON LINE
                F = 145, P      " BEYOND THE END?
Y, SUBC(:NLCR)    " BEGIN ON NEW LINE
                A = :CBASE
AMS:            S = MA          " CHARACTER
                S + :PRNTAB     " CONVERSION TABLE
                S = MS, P       " CONVERT CODE. - SPACE?
N, PRPOS = S      " PRPOS = PRPOS + 1
Y, MC = A         " SAVE A
Y, SUBC(:PRHEX)   " PRINT CHARACTER
Y, A = MC[-1]     " RESTORE A
                A + 1           " INCREMENT FOR NEXT CHARACTER
                REPP(:AMS)
                GOTOR(MC[-1])   " EXIT PRINT, FIXT, ABSFIXT, FLOT

PRNTIS:         MC = S          " PRESERVE S-REGISTER
                A = 127         " REMOVE EXTRANEIOUS BITS
                A + :CONTAB     " CONVERSION TABLE
                S = MA          " CONVERT TO PRINTER CODE
U, S + 65, P      " - TAB OR NLCR?
Y, SUBC(:PRHEX)   " PRINT
Y, S = MC[-1]     " RESTORE S
Y, GOTOR(MC[-1])  " EXIT PRNTIS
                S + 65, Z       " TAB?
Y, SUBC(:TAB)     " DO TAB
N, SUBC(:NLCR)    " DO NLCR
                S = MC[-1]     " RESTORE S
                GOTOR(MC[-1])  " EXIT PRNTIS

```

'BEGIN' FVOC, ELOOP0, ELOOP1

```

" ASSUMED TO BE DECLARED GLOBALLY ARE: DERRORM, ERRORM
" ASSUMED TO BE DECLARED AND DEFINED GLOBALLY ARE: DANGEROUS
"   RUNNUMBER, ERRONEOUS, PRPOS, CARRIAGE, CTYPE, SMES,
"   FIXT60, PRHEX, AFXT, WANTED, LINE COUNTER, AFXT6,
"   LAST SYMBOL, CONTAB, SPACE, PRNTIS, VALUE OF CONSTANT,
"   PRINT, ENCRUN, LAST IDENTIFIER

DERRORM:  DANGEROUS = -B      " NOTE DANGEROUS ERROR
ERRORM:  U, RUNNUMBER = A, P  " IGNORE ERROR?
        Y, GOTOR(MC[-1])      " EXIT DERRORM AND ERRORM
        MC = S                " SAVE S-REGISTER
        S = 0
        ERRONEOUS = S        " NOTE ERROR
        S = PRPOS            " POSITION ON LINE
        MC = S               " PRESERVE IT IN STACK
        S = EMPTY, P        " BLANK LINE INDICATION
        MC = S              " PRESERVE IT IN STACK
        S = COUNT
        MC = S              " PRESERVE COUNT
        MC = A              " PRESERVE ERROR NUMBER
        Y, F = 1            " ) SINGLE OR
        N, F = 2            " ) DOUBLE
        SUBC(:CARRIAGE)     " ) SPACE VERTICALLY
        ('660 072 000'+:KY) " PREVENT KEYBOARD INTERRUPTS
        SUBCD(:CTYPE)
        :SMES               " TYPE CR NL BLKDIG
        G = M[B - 1]        " ERROR NUMBER
        SUBC(:FIXT60)       " TYPE IT
        ('760 072 000'+:KY) " ALLOW KEYBOARD INTERRUPTS AGAIN

        S = 37
        SUBC(:PRHEX)        " "E"
        S = 50
        SUBC(:PRHEX)        " "R"
        A = 3
        G = MC[-1]          " ERROR NUMBER
        SUBC(:AFXT)         " PRINT IT
        A = RUNNUMBER
        A = 500, Z          " EXECUTION
        N, SUBC(:DUMPING OFF)
        Y, A = -WANTED, P   " ^ - WANTED?
        G = LINECOUNTER     " OBJECT PROG LINE NUMBER
        N, SUBC(:AFXT6)     " PRINT IT

        A = RUNNUMBER
        A = 500, Z          " EXECUTION?
        Y, GOTO(:FVOC)      " AVOID LAST SYMBOL
        A = LAST SYMBOL
        U, A '*' -127, Z    " < 128 ?
        N, JUMP(5)          " ELSE PRINT INT REP
        U, A = 93, P        " ) ≠ 94,
        U, A = 96, E        " ) 95, 96?
        Y, A + :CONTAB      " CONVERSION TABLE
        Y, A = MA, P        " ) NOT COMPOUND?
        Y, A = 0, P         " ) ^ NOT SPACE?
        G = LAST SYMBOL
        N, SUBC(:AFXT6)     " PRINT INTERNAL REPRESENTATION
        N, GOTO(:FVOC)      " DONE

```

```

      F = 6
      SUBC(:SPACE)           " 6 SPACES
      A = LAST SYMBOL
      SUBC(:PRNTIS)         " PRINT THE SYMBOL
      F = 1, Z              " CONDITION = NO
      SUBC(:SPACE)         " 1 SPACE
FVOC:  F = VALUE OF CONSTANT
      SUBC(:PRINT)         " PRINT IT
      Y, GOTO(:ENDRUN)     " TERMINATE EXECUTION

      S = 2                 " 2 WORDS OF IDENTIFIER
      M[6] = S             " COUNTER LOCATION
      S = LAST IDENTIFIER  " FIRST WORD OF IDENTIFIER
ELOOP0: A = 4              " 4 CHARACTERS PER WORD
      COUNT = A            " COUNTER LOCATION
      LCS(2)               " DISCARD UNUSED BITS
ELOOP1: LUAS(6)            " SHIFT TO POSITION
      A = ' 63, Z         " CLEAN CHARACTER, DONE?
      N, A = :MA[-1]       " CORRECT CODE
      N, SUBC(:PRNTIS)     " PRINT CHARACTER
      REPP(:ELOOP1)        " NEXT CHARACTER, IF ANY
      S = LAST IDENTIFIER[1], Z " SECOND WORD, EMPTY?
      N, REP6P(:ELOOP0)    " PRINT SECOND WORD
      F = 2
      SUBC(:CARRIAGE)      " DOUBLE SPACE VERTICALLY
      S = MC[-1]
      COUNT = S            " RESTORE COUNT
      S = MC[-1]          " OLD BLANK LINE INDICATION
      EMPTY = S           " RESTORE IT
      S = MC[-1]          " OLD POSITION ON LINE
      PRPOS = S           " RESTORE IT
      S = MC[-1]          " RESTORE S-REGISTER
      GOTOR(MC[-1])        " EXIT DERRORM AND ERRORM

      'END'                " ERROR MESSAGE

```

'BEGIN' HERE

HERE:

INTREP[27]: -' 066 400 010'

INTREP[31]: -'132 400 011'

HERE[0]:

NXT TAPE SL: 'BEGIN' DECIDE, EXIT, SPECIAL, PARITY, RCLN,
BAT, UL, LC, UC, NONFLX, TRANSIT

" ASSUMED TO BE DECLARED AND DEFINED GLOBALLY ARE:
" REHEP, INTREP, TLXREP, FLEXCODE, CASE,
" RUNNUMBER, PRNTIS, ERRORM, NLCR,
" LINECOUNTER, AFXT, TAB

	SUBC (:REHEP)	" GET HEPTAD FROM BUFFER
	S '*' 127	" REMOVE EXTRANEIOUS BITS
	U, S - 31, P	" HEPTAD > 31 ?
	Y, FLEXCODE = B	" FLEXCODE
	N, A = FLEXCODE, P	" OR CODE KNOWN TO BE FLEXO ?
	Y, S + :INTREP	") BASE ADDRESS
	N, S + :TLXREP	") OF CONVERSION TABLE
	S = MS, P	" TABLE ENTRY IN S
	N, GOTO (:SPECIAL)	" IF NEGATIVE, SPECIAL HANDLING
DECIDE:	U, S = CASE, Z	" LOWER CASE ?
	N, RUS (8)	" SHIFT TO UPPER CASE BITS
	S '*' 255	" MASK OUT EXTRANEIOUS BITS
EXIT:	A = RUNNUMBER	
	A = 100, Z	" PRESCAND ?
	N, GOTOR (MC[-1])	" EXIT NXT TAPE SL
	A = S	
	GOTO (:PRNTIS)	" PRINT CHARACTER AND EXIT NXT TAPE SL
SPECIAL:	LUS (10)	" CLEAN AWAY FLEXIS BITS
	RUS (10)	" SHIFT BACK
	S + :PARITY	" BASE ADDRESS OF JUMP TABLE
	GOTO (S)	" SWITCH TO HANDLE SPECIAL CASES
	GOTO (:TRANSIT)	" - 9 LETTER SHIFT
	GOTO (:TRANSIT)	" - 8 FIGURE SHIFT
	GOTO (:RCLN)	" - 7
	GOTO (:BAT)	" - 6
	GOTO (:UL)	" - 5
	GOTO (:LC)	" - 4
	GOTO (:UC)	" - 3
	GOTO (:NXT TAPE SL)	" - 2 SKIP ERASE, BLANK, STOP, BACKSP
	GOTO (:NONFLX)	" - 1

```

PARITY:      A = 102          " - 0 ERROR NUMBER 102
              S = RUNNUMBER
              S = 500, Z      " EXECUTION ?
Y, A + 413    " THEN INCREMENT ERROR NOS. BY 412
              SUBC (:ERRORM)  " REPORT THE ERROR
              GOTO (:NXT TAPE SL) " AND TRY THE FOLLOWING HEPTAD

RCLN:        A = RUNNUMBER
              A = 100, Z      " PRESCAN ?
N, GOTO (:S119)
              S = COUNT
              MC = S          " SAVE COUNT
              SUBC (:NLCR)    " NEW LINE ON LINE PRINTER
              G = LINECOUNTER
              F + 1
              A = 4
              SUBC (:AFXT)    " PRINT NEW LINE NUMBER IN 4 DIG.
              SUBC (:TAB)     " TAB THE LINE PRINTER
              S = MC[-1]
              COUNT = S       " RESTORE COUNT
S119:        S = 119          " CARRIAGE RETURN
              GOTOR (MC[-1])  " EXIT NXT TAPE SL

BAT:         S = 118          " TAB
              GOTO (:EXIT)

UL:          S = 32638        " UNDERLINE OR VERTICAL BAR
              GOTO (:DECIDE)  " DEPENDING ON CURRENT CASE

LC:          S = 0            " LOWER CASE
UC:          CASE = S         " STORE CASE
              GOTO (:NXT TAPE SL) " AND GET NEXT HEPTAD

NONFLX:      A = 103          " ERROR NUMBER 103
              GOTO (:PARITY[1]) " REPORT ERROR

TRANSIT:     A = RUNNUMBER
              A = 500, Z      " EXECUTION ?
N, GOTO (:MS[8])              " FIND APPROPRIATE ERROR NUMBER
              FLEXCODE = - 8  " CODE = TELEX
              GOTO (:MS[5])    " FIND APPROPRIATE SHIFT

              'END' NXT TAPE SL

```

```

TLXREP:      - '000 000 002' ; + '000 002 435' " BLANK 5/T
              - '000 000 002' ; + '000 004 430' " CR 9/O
              + '000 056 535' ; + '000 054 421' " SPACE 10/H
              + '000 053 427' ; + '000 054 026' " ,/N ,/M

              - '000 000 007' ; + '000 061 425' " NL )/L
              + '000 002 033' ; + '000 062 420' " 4/R ]/G
              + '000 004 022' ; + '000 000 031' " 8/I 0/P
              + '000 055 014' ; + '000 043 037' " :/C =/V

              + '000 001 416' ; + '000 040 043' " 3/E +/Z
              + '000 057 415' ; + '000 041 013' " /D */B
              + '000 074 034' ; + '000 003 042' " ' /S 6/Y
              + '000 062 017' ; + '000 041 441' " [/F //X

              + '000 040 412' ; + '000 001 040' " ~/A 2/W
              + '000 055 423' ; - '000 000 003' " ,./J FIGURES
              + '000 003 436' ; + '000 000 432' " 7/U 1/Q
              + '000 061 024' ; - '000 000 004' " (/K LETTERS

```

'END'

```

TAB:          S = PRPOS " POSITION ON LINE
              S = -7
              S = 9 " NEXT TAB POSITION
TAB3:         U, S = 145, P " LINE OVERFLOW?
              N, PRPOS = S
              N, EMPTY = -B " NOTE OCCURRENCE OF CHARACTERS
              N, GOTOR(MC[-1]) " EXIT TAB AND SPACE
              SUBC(:NLCR)
              S = 144 " START A NEW LINE
              GOTO(:TAB3)

SPACE:        SUBC(:RND) " ROUND PARAMETER
              S = G, P " > 0?
              N, GOTOR(MC[-1]) " ELSE EXIT SPACE
              S = PRPOS " NEW POSITION ON LINE
              GOTO(:TAB3) " TEST AND EXIT

AFXT6:        A = 6 " NN = 6
AFXT:         S = 0 " MM = 0
              ABS = S " UNSIGNED VERSION
              SUBC(:FIX1) " FIXED-POINT CONVERSION
              GOTO(:PRO) " PRINT IT OUT

```

'END'

" LINE PRINTER SECTION

```

'BEGIN'      CRDAR, PCAR, CRDSS, BUFFERM1, PCSS, BUFFER39, CASE,
              POINTER, COUNTER, LAST, WRONG, INTABLE, INSTR,
              CRDREADER, LOOP, SELECT, RESULT, NLCR, START PC,
              EIGHT, NOK, PUNCHER, START CRD

              " ASSUMED TO BE DECLARED GLOBALLY ARE:
              " RECODE, ISTART

              " ASSUMED TO BE DECLARED AND DEFINED GLOBALLY ARE:
              " CRD, PC, BRUSH, INTAB, INTREP, D18, RESTORE, HOK

```

RECODE:

M[64 + (4*:CRD)]: CRDAR:

```

M[64 + (4* : PC)]:          PCAR:

KTAER[0]:                    ('033 000 000' + :RECODE )
KTAB[0]:                      ('022 000 000' + :ISTART )

RECODE[0]:                   F = ('000 040 000'+:CRD)
                              SUBC (:BRUSH)          " RESET CARD READER
                              F = ('000 040 000'+:PC)
                              SUBC (:BRUSH)          " RESET PUNCH
                              A = :CRDSS
                              CRDAR[0] = A          " READER OK + LABEL
                              A = :PCSS
                              PCAR[0] = A          " PUNCH OK + LABEL
                              A = :CRDREADER
                              INTAB[8+:CRD] = A " SET INTERRUPT ADDRESS CARD READER
                              A = :PUNCHER
                              INTAB[8+:PC] = A " SET INTERRUPT ADDRESS PUNCH
                              LAST = - B
                              WRONG = - B
                              GOTO (:START CRD) " START CRD READER

                              'SKIP' 1
CRDSS:                        - 1
BUFFERM1:                     '176777'          " 47 1/2 K - 1

                              'SKIP' 1
PCSS:                          - 1
                              'SKIP' 1

```

BUFFER39: '176 047' " 47 1/2 K + 39
CASE: 'SKIP' 1
POINTER: 'SKIP' 1
COUNTER: 'SKIP' 1
LAST: 'SKIP' 1
WRONG: -0

INTABLE:	- 43 528 934	"	- ? "
	- 10 996 377	"	SEMICOLON + > =
	+ 6 328 088	"	) (- ' +
	- 63 291 536	"	* < / O ~
	- 2 212 991	"	\$, ' ##
	+ 8 486 138	"	[] " :
	- 44 516 902	"	R I Z 9
	- 46 630 567	"	Q F Y 8
	- 48 744 232	"	P G X 7
	- 50 857 897	"	O F W 6
	- 52 971 562	"	N E V 5
	- 55 085 227	"	M D U 4
	- 57 198 892	"	L C T 3
	- 59 312 557	"	K E S 2
	- 61 437 486	"	J A / 1
	+ 36 186 157	"	- ^ 0 SPACE

INSTR: RUA (7)
RUA (21)
RUA (14)

```

CRDREADER:      ('660 071 000'+:CRD) " REMOVE INTERRUPT SIGNAL
                  A = CRDSS[-1], P    " OK ?
N, GOTO (:NOK)
                  G = BUFFER39        " POINTER TO LAST WORD OF BUFFER
                  A = 80              " PREPARE LOOP
                  S = MG, Z           " TWO BLANKS IN BUFFER WORD ?
Y, A = 2, P      " AND STILL MORE COLUMNS ?
Y, F = 1         " INSPECT PREVIOUS COLUMN
Y, JUMP (-4)
                  RUS (12), Z        " ONE BLANK ?
Y, A = 1
                  COUNTER = A, P
                  B = BUFFER39
N, GOTO (:NLCR)
                  CASE = B          " SHIFT UNDEFINED
                  G = BUFFER M1      " POINTER TO FIRST COLUMNS
                  POINTER = G        " PREPARE LOOP
LOOP:            G = POINTER, P
Y, F + 1         " NEXT WORD
                  POINTER = - G
N, F = - F
                  A = MG            " TWO COLUMNS
N, RUA (12)      " SELECT ONE
                  F = :MC           " SAVE POSITION IN BUFFER
                  A '*' 4095        " CLEAN AWAY EXTRANEIOUS BITS
                  S = 0
                  RUAS (3)         " ISOLATE ZONE PUNCHINGS INTO S
U, A '*' 128, Z  " NO PUNCHING IN ROW 8 ?
N, GOTO (:EIGHT)
                  NORA             " NORMALIZE BITS
                  A = INTABLE[B-11] " WORD FROM CONVERSION TABLE
SELECT:          NORS, Z           " ANALYZE ZONE PUNCHINGS
N, DO (INSTR[B]) " SHIFT WORD TO RIGHT POSITION
                  A + 48
RESULT:          A '*' 127         " CLEAN AWAY EXTRANEIOUS BITS
                  B = :MG           " RESTORE POSITION IN BUFFER
                  A + :INTREP        " BASE ADDRESS CONVERSION TABLE
                  S = MA            " SELECT WORD FROM TABLE
                  A = 124           " TRY UPPER CASE
                  LCS (10), P       " GESS CORRECT ?
N, A = 122       " PERHAPS LOWER CASE IS BETTER
                  RCS (1), E        " CASE INDEPENDENT ?
                  S '*' 127        " CLEAN AWAY EXTRANEIOUS BITS
Y, JUMP (3)
U, A = CASE, Z   " CASE CORRECT ?
N, CASE = A      " NEW CASE
N, MC[1] = A     " PUT CASE INTO BUFFER
                  MC[1] = S         " PUT CHARACTER INTO BUFFER
                  A = 1
                  COUNTER = A, Z    " CCUNT DOWN. READY ?
N, GOTO (:LOOP)
                  S = 26
NLCR:            MC[1] = S          " CARRIAGE RETURN
                  F = 0
                  MC[1] = F         " TWO BLANKS
                  MC[1] = F         " AGAIN TWO BLANKS
                  MC[1] = F         " AGAIN TWO BLANKS
                  MC[1] = F         " AGAIN TWO BLANKS
                  MC[1] = F         " AGAIN TWO BLANKS
                  A = :MC

```

```

      A = BUFFER39
      RUA (18)          " ) FORM
      A + BUFFER39      " ) CODE WORD
      PCSS[1] = A        " STORE CODE WORD
START PC:  PCAR[1] = B    " INTERRUPT COUNTER = 0
      A = D18
      PCAR[2] = A        " ACTION COUNTER = 1
      ('760 070 000'+:PC) " START PUNCH
      GOTO (:RESTORE)    " EXIT INTERRUPT CARDREADER

EIGHT:      A '*' 126    " CLEAN AWAY BIT 9, 8, AND 1
      NORA, Z           " NORMALIZE REMAINING BITS
      A = INTABLE[B-19] " SELECT WORD FROM CONVERSION TABLE
      Y, GOTO (:SELECT)
      NORS, Z           " ANALYZE ZONE PUNCHINGS
      N, DO (INSTR[B])  " SHIFT WORD TO RIGHT POSITION
      A + 96
      GOTO (:RESULT)

NOK:        A '*' 15, Z  " NOT END OF FILE ?
      A = :CRDAR[1]
      SUBC (:HOK)        " RESET CARD READER
      Y, LAST = B
      Y, GOTO (:CRDREADER[3])
      WRONG = B
      GOTO (:RESTORE)

I START:    A = WRONG, P
      N, GOTO (:RESTORE)
      WRONG = - B
      A = :CRDAR[1]
      SUBC (:HOK)        " RESET CARD READER ONCE MORE
      GOTO (:START CRD)  " TRY AGAIN

PUNCHER:    ('660 071 000'+:PC) " REMOVE INTERRUPT BIT
      A = PCAR[0], P     " OK ?
      N, A = :PCAR[1]    " )
      N, SUBC (:HOK)      " ) RESET PUNCH
      N, A = :PCAR[1]    " ) TWICE
      N, SUBC (:HOK)      " )
      N, GOTO (:START PC)
      A = LAST, P
      Y, GOTO (:RESTORE)

START CRD:  A = D18
      CRDAR[2] = A        " ACTION COUNTER = 1
      CRDAR[1] = B        " INTERRUPT COUNTER = 1
      ('760 070 000'+:CRD) " START CARD READER
      GOTO (:RESTORE)

'END'
FREEWO:
'END'
" SUBMONITOR VARIABLES 1
" PICO

SUBMOVAR[0]:

'BEGIN'      CMODE, BASE, DPO, BEGIN OF PROGRAM,
              END OF PROGRAM, RELADRSS, BEGIN OF TEXT ARRAY, BEGINSAFE,
              WORDSAFE, SHIFTSafe, PNTR, CORRECTIE, USER,
              LABEL, D24PLUSD22

```

FIRST 3 TIMES:	'SKIP' 1	"INFORMS THE SYSTEM HOW TO SET HMODE
HMODE:	'SKIP' 1	"JOB INTERPRETATION
CMODE:	'SKIP' 1	"COMPILING MODE
LVAR:	'SKIP' 6	
BASE:	'SKIP' 1	
DPO:	'SKIP' 1	") THIS ORDER OBLIGATE
BEGIN OF PROGRAM:	'SKIP' 1	")
END OF PROGRAM:	'SKIP' 1	
RELADRSS:	'SKIP' 1	
BEGIN OF TEXT ARRAY:	'SKIP' 1	
BEGINSAFE:	'SKIP' 1	") THIS ORDER OBLIGATE
SHIFTSAFE:	'SKIP' 1	")
WORDSAFE:	'SKIP' 1	
PNTR:	'SKIP' 1	
INSTR CNTR:	'SKIP' 1	
CORRECTIE:	'SKIP' 1	
	'SKIP' 3	
USER:	+0	" ONLY ONE START 1ORDER
D24PLUSD22:	'120 000 000'	
LABEL:	'SKIP' 2	" 28 VARIABLES

"SUBMONITOR SECTION NR 2.

```

SUBMONITOR:      S = 1              ")UNLESS OTHERWISE SPECIFIED
                  HMODE = S         ")JCB IS INTERPRETED AS NORMAL. PROGRAMS
U, S = FIRST 3 TIMES, Z "-(1ST, 2ND OR 3RD RUN OF THE DAY)
Y, GOTO(:NPR)
                  FIRST 3 TIMES + S
                  S = 0
                  HMODE = S
                  GOTO(:ILR)

NPR:              'BEGIN' LOOP, END
                  SERIAL + S         "INCREASE SERIAL NUMBER
                  SUBC(:DATE AND SERIAL) "INFORM OPERATOR NEXT
                  SUBCD(:CTYPE)      "PROGRAM WILL BE
                  :TRNMEs            "COMPILED
                  A = :BASE 0
                  BASE = A
                  A = 1
                  CMODE = A
                  SUBC(:INIT BITSTRM)
                  SUBC(:PRESCAN0)     "COMPILE PROGRAM
                  A = INSTR CNTR
                  A + RELADRSS
                  USER[-3] = A        "LENGTH OF TOTAL PROGRAM
                  SUBC(:FORCE LAST BITS) "EMPTY STRWRD
                  SUBCD(:ASK DUMP END) "FIRST FREE PLACE
                  A = DRBEG8
                  USER[-1] = A        "START ADDRESS ON DRUM
                  S = A               "NUMBER OF WORDS ON DRUM
                  A = 0
                  DIVAS(:REC BULEN), Z
N, S + 1          "NUMBER OF BUFFERS
                  USER[-2] = S        ")
                  SUBC(:FINISH BITSTRM)
                  SUBC(:END DUMP)

```

"AFTER EXECUTION OF THE INSTRUCTIONS ABOVE ONLY 4 WORDS
 " IN CORE ARE OCCUPIED BY INFORMATION RELEVANT TO THIS
 "USER. THIS ARE THE 4 WORDS DENOTED BY USER, USER[-1], USER[-2],
 " WHICH TOGETHER FORM A START 1 ORDER AND USER[-3] WHICH
 "CONTAINS LENGTH OF THE OBJECT PROGRAM IN CORE.

"SUBMONITOR SECTION NR 3

```

      S = ERRONEOUS, Z          ")NC RECOVERING IN
Y, GOTO(:ENDRUN)              ")CASE OF MISTAKES
      A = USER[-3]
      G = A                    "G MAY BE USED IN ERRORM
      A + BEGIN OF PR AR      "SPACE ENOUGH FOR RECOVERING
      SUBC (:OBSERVE)         "COMPILER WILL BE OVERWRITTEN BY OBJECTPROG
      PRESENCE = - B
      A = :USER
      LABEL = A
      S = MA[-2]              ")NUMBER OF BUFFERS
      LABEL[1] = S           ")TC BE TRANSPORTED
      SUBC (:INIT BITSTREAM)
      S = BEGIN OF PR AR     ")FROM NOW ON INSTR CNTR
      INSTR CNTR = S         ")HAS ABSOLUTE MEANING VERSUS
                              ")RELATIVE MEANING DURING COMPILATION

      SUBC (:TRANSF)
      SUBC (:MAKE START1ORDERS)
      S = INSTR CNTR
      S = MS[-1]              "SUM OF MAXIMA OF HEAD PROGRAM
      MC = S
      F = DPO                 ")DPO AND BEGIN OF PROGRAM
      MC = F                  ")OF HEAD PROGRAM
      SUBC (:INIT BITSTREAM)
LOOP:  S = 2                   ") PNTR IS OUTPUTPARAMETER
      PLUS(PNTR)              ") OF MAKE START1ORDERS
U, S = M[B - 1], Z
Y, GOTO(:END)
      S = 1
      MINS(INSTR CNTR)
      SUBC (:TRANSF)
      A = BEGIN OF PROGRAM
      S = PNTR                ")FILL
      MS = A                  ")PSEUDC
      A + 1                   ")LIBRARY
      MS[1] = A               ")VARIABLE
      A = INSTR CNTR
      A = MA[-1]              ")POSSIBLE REFRESH
U, A = M[B - 3], P           ")SUM OF MAXIMA
Y, M[B - 3] = A              ")
      SUBC (:SKIP REST OF BUF)
      GOTO(:LOOP)

END:  F = MC[-2]              ")RESTORE DPO AND BEGIN OF PROGRAM
      DPO = F                 ")
      S = INSTR CNTR
      A = MC[-1]              " M[END OF TOTAL PROGRAM] = SUM OF MAXIMA
      MS[-1] = A
      GOTOR(MC[-1])

      'END'   NPR

TRANSF:  GOTO(BEGIN OF UP 32 K)

```

" SUBMONITOR 4

```
ILR: BEGIN NEXT PROCEDURE
      SUBCD (:CTYPE)
      :PTRNMES
      SUBC(:INITPR2)
```

A = 8

CMODE = A

" ASK PERMIT TO INSERT ROUTINES

G = :BASE1

BASE = G

SUBC(:INIT BITSTRM)

" NO DUMPING

SUBC(:PRESCAN0)

S = NBR OF ROUTINES

U, MAX NBR OF ROUTINES - S, P " ROOM?

N, A = 807

N, SUBC(:ERRCRM)

G = ERRONEOUS, Z

" IF ANY ERROR THEN

Y, GOTO(:ENDRUN)

" RESTORE STATE

S = LVAR

" NUMBER TO INSERT

END OF TRAFOTABLE + S

END OF CROSSTABLE + S

A = BEGIN OF TEXT ARRAY

BEGINSAFE = A

S = MA

WORDSAFE = S

S = 1

SHIFTSafe = S

F = LVAR

MC = F

WANTED = -B

" VARIABLE TO COUNT PROCEDURES

" VARIABLE TO POINT INTO CATALOGUE

" NO COMPILING OF LINENUMBERS

A = :BASE2

BASE = A

A = 2

CMODE = A

") INDICATE INSERTING

") ROUTINES AUTHORIZED

" SUBMONITOR 5

```

      SUBC(:INIT BITSTRM)
NEXT PROCEDURE: S = END OF INFOTABLE
      MC = S
      SUBC(:PRESCAN0)
      SUBC(:FORCE LAST BITS)
      SUBCD(:ASK DUMP END)
      G = END OF INFOTABLE      ") DELIVERS FIRST FREE PLACE ON DRUM
      MG = S                    ") M[END OF INFOTABLE]:=
      END OF DRUMLIBRARY = S    ") END OF DRUMLIBRARY:=
      A = 1                    ") FIRST FREE PLACE ON DRUM
      PLUSA(M[B-3])            ") NBR OF ROUTINE
      A = BEGIN OF CROSSTABLE  ") SO CALLED "ENTRY" FOR CURRENT ROUTINE
      D = A                    ") D IS INPUTPARAMETER FOR INFORM PROGRAMMER
      MA[1] = G                ") CROSSTABLE[ENTRY+1]:=END OF INFOTABLE
      S = MC[-1]               ") VALUE END OF INFOTABLE BEFORE COMPILATION
      G = INSTR CNTR
      G = 1                    ") M[OLD END OF INFOTABLE]:= LENGTH IN CORE
      MS[1] = G
      S = D24PLUSD22
      MA = S                   ") CROSSTABLE[ENTRY]:=
                               ") ('120 000 000' +:ENTRANCE)
      SUBC(:INFORM PROGRAMMER)
      G = M[B-1]
      SUBC(:NXT PP)            ") LAST PROCEDURE?
N, M[B-1] = A
N, GOTO(:NEXT PROCEDURE)
      SUBC(:FINISH BITSTRM)
      A = LVAR[1]
      G = ERRONEOUS, Z
N, SUBC(:INFORM OPERATOR)    ") AND SYSTEM TOO
      GOTO(:ENDRUN)           ") 60 INSTRUCTIONS

      'END' ILR

NXT PP:      S = MG
      S '*' 32767              ") NEXT PROCEDURE POINTER
      MULS(3)                  ") :=ENTRY IN CATALOGUE
      A = -MS[-1]              ") -(3*FORMAL COUNT +1)
      A = G
      U, A = END OF CATALOGUE, Z ") L-ST PROCEDURE?
      GOTO(MC[-1])             ") 7 INSTRUCTIONS

UPDATE LIBRARY:      F = NBR OF ROUTINES      ")
      LVAR = F                  ") SEE ASSURE LIBRARY
      F = END OF TRAFOTABLE    ")
      LVAR[2] = F              ") SEE ENDRUN TOO
      F = END OF INFOTABLE     ")
      LVAR[4] = F              ")
      GOTOR(MC[-1])            ") 7 INSTRUCTIONS

```

" SUBMONITOR 6

INFORM PROGRAMMER:

'BEGIN' PR IDF, PRINT, PRINT WORD, ELOOP1

```

PR IDF:
    F = 2
    SUBC(:CARRIAGE)
    A = 1
    MINA(M[B-2])
    S = MA[1], P
    N, SUBC(:PRINT WORD)
    N, GOTO(:PR IDF)
    F =:MS
    SUBC(:PRINT)
    G = M0[1]
    SUBC(:PRINT)
    G =-M0
    G + END OF DRUMLIBRARY" LENGTH ON DRUM
    PRINT:
    SUBC(:AFXT6)
    F = 6
    GOTO(:SPACE)
    PRINT WORD:
    S = - S
    A = 4
    COUNT = A
    LCS(2)
    LUAS(6)
    A '*' 63, Z
    ELOOP1:
    N, A =:MA[-1]
    N, SUBC(:PRNTIS)
    REPP(:ELOOP1)
    GOTOR(MC[-1])

```

") DOUBLE SPACE

") VERTICALLY

" NO LONGER IDENTIFIER?

" NUMBER OF PROCEDURE

" LENGTH IN CORE

" LENGTH ON DRUM

" 16 INSTRUCTIONS

" DISCARD UNUSED BITS

" SHIFT TO POSITION

" CLEAN CHARACTER, DONE?

" CORRECT CODE

" PRINT CHARACTER

" NEXT CHARACTER, IF ANY

" 10 INSTRUCTIONS

'END' INFORM PROGRAMMER

"SUBMONITOR SECTION NR 7.

MAKE START1 ORDERS:

```

    'BEGIN'          LOOP

    A = BEGIN OF PROGRAM
    PNTR = A
    A = END OF PROGRAM
    A + RELADRSS      " COREADDRESS LAST WORD OBJECTPROG
    S = 0 , Z         " CONDITION YES
    LABEL[1] = S      " AMOUNT OF BUFFERS TO RECOVER

LOOP: A + 3
    G = A              " )ASK PERMIT TO FILL
    G = CORRECTIE     " )ANOTHER 3 WORDS
    SUBC(:OBSERVE)

Y, LABEL = A          " INITIALIZE LABEL
Y, MA[-2] = S         " )FILLING OF FIRST START1 ORDER
                     " )NOW AUTHORIZED, IF THERE MIGHT
                     " )BE NO PROCEDURES AT ALL THEN NO BUFFERS
                     " )TO RECOVER
                     " )FILL FIRST LABEL WITH ZERO AND
                     " )OTHERWISE POSSIBLE "NEW" LABEL
                     " )WITH PREDECESSOR

    MA = S

    S = 2
    MINS(PNTR)
    S = MS[1] , Z     " )NUMBER OF NEXT ROUTINE, IF ANY
Y, GOTOR(MC[-1])      " )IF NOT FIRST LABEL THEN REFRESH
    LABEL = A

    G = A
    S = BEGIN OF CROSSTABLE
    D = S
    S = -M0
    MG[-1] = -S       " )START ADDRESS ON DRUM INTO
                     " )START1 ORDER
    S = M1             " )LENGTH ON DRUM
    A = 0
    DIVAS(:RECBULEN) , Z " )LENGTH = INTEGER AMOUNT OF BUFFERS?
N, S + 1
    MG[-2] = S        " )AMOUNT OF BUFFERS TO RECOVER
    LABEL[1] + S      " )STORED IN START1 ORDER
    S = LABEL
    A = LABEL         " )32 INSTRUCTIONS
    GOTO(:LOOP)       " )ADDED TO TOTAL AMOUNT
    'END' MAKE START1 ORDERS

ASSURE LIBRARY:      F = LVAR
                     NBR OF ROUTINES = F
                     F = LVAR[2]
                     END OF TRAFOTABLE = F
                     F = LVAR[4]
                     END OF INFOTABLE = F "SEE ENDRUN
                     GOTOR(MC[-1])       "7 INSTRUCTIONS

" SYSTEM 1

'BEGIN' BCHECK, CONSIDER,ERM, DANGER

```

SYSTEM:

```

BCHECK:          'SKIP' 1          " 1 VARIABLE

CONSIDER:        U, B = BCHECK, P      "STACK GROWTH DANGEROUS?
                  N, GOTO(LINK[2])      "CONTINUE EXECUTION
                  B = BEGIN OF STACK
                  A = 609                "END OF EXECUTION
                  SUBC(:ERRORM)         "5 INSTRUCTIONS

OBSERVE:         U, A = END OF COMPILER OUTPUT SPACE, P
                  N, GOTOR(MC[-1])
                  A = 493
                  VALUE OF CONSTANT = P " ) LENGTH OF TOTAL

ERM:             SUBC(:ERRORM)         " ) PROGRAM
                  GOTO(:ENDRUN)        "6 INSTRUCTIONS

DANGER:          S = - DANGEROUS, P
                  Y, A = 490
                  Y, GOTO(:ERM)
                  GOTOR(MC[-1])        "4 INSTRUCTIONS

```

TRAFO:

```

"RECOVER SECTION NR 1.
RECOVER[0]:

```

```

'BEGIN'          QUEU7, CRBUF1, CRBUF2, CRBUNOF7, CRBUNOE7,
                  RECPOS, BUFEND, STRWORD, JUST NINE,
                  NEXT STRWORD, STILL EIGHTEEN, FETCH FAST,
                  FROM DRUM, DRTOCR

```

```

"ASSUMED TO BE DECLARED GLOBALLY ARE: BTSTREAM9, INIT BITSTREAM,
"BITSTREAM18, BITSTREAM27, SKIP REST OF BUF, PROCESS7, END RECOVER

```

```

"ASSUMED TO BE DECLARED AND DEFINED GLOBALLY ARE: ACTIVE, D18MIN1,

```

```

BTSTREAM9:      'SKIP' 1
QUEU7:          'SKIP' 1
CRBUNOF7:       'SKIP' 1
CRBUNOE7:       'SKIP' 1
DRRCOM7:        'SKIP' 1
RECPOS:         'SKIP' 1
BUFEND:         'SKIP' 1
STRWORD:        'SKIP' 1

                  :CRBUF2
CRBUF1:         'SKIP' : RECBULEN
                  :CRBUF1
CRBUF2:         'SKIP' : RECBULEN

```

"RECOVER SECTION NR 2.

```

INIT BITSTREAM:      'BEGIN'          LOOP
                      A = 0            " NO QUEUE OF BUFFERS
                      QUEU7 = A        " WAITING FOR TRANSPORT
                      A = : NEXT STRWORD " )
                      BTSTREAM9 = A    " ) INITIALIZE PSEUDO-SHIFT
                      ACTIVE[7] = -B   " NO ACTIVE PROCESS
                      S = LABEL[1]
                      DRROOM7 = S      " NUMBER OF BUFFER TO RECOVER
                      S = : CRBUF1
                      CRBUNOF7 = S     " SELECT FIRST BUFFER TO FILL
                      CRBUNOE7 = S

LOOP:                A = MS[-1] , P   " ) ASSURE PRESENCE
                      N, MS[-1] = - A  " ) OF BUFFER
                      SUBCD(:FROM DRUM) " ASK FOR TRANSPORT
                      S = CRBUNOE7
                      U, S = :CRBUF1 , Z " CYCLE CLOSED?
                      N, GOTO(:LOOP)

                      'END' INIT BITSTREAM

```

"RECOVER SECTION NR 3.

"JUST NINE, NEXT STRWORD AND STILL EIGHTEEN FORM TOGETHER
 "A LOGIC BLOCK, THAT PLAYS AN ANALOGOUS ROLE IN THE RECOVER SECTION
 "AS REHEP IN THE READER SECTION. TO FULLFILL THE REQUIREMENTS OF A
 "TRICKY OPTIMIZATION IN BTSTREAM18 THE SEQUENCE IN WHICH THEY
 "ARE IS OBLIGATE

```

JUST NINE:      S = : NEXT STRWORD ")CHANGE PSEUDO-SHIFT
                BTSTREAM9 = S      ")
                S = STRWORD
                S '+' 511          " FETCH BITS
                GOTOR(MC[-1])
        'BEGIN' ASSURE PRESENCE, GET WORD
ASSURE PRESENCE: S = MG[-1], P      " BUFFER PRESENT?
                N, JUMP(-2)
                P + :RECBULEN, Z
                BUFEND = G          " RELEASE BUFFER
                GOTO(:GET WORD)     " COND = NO
NEXT STRWORD:   S = :STILL EIGHTEEN") CHANGE
                BTSTREAM9 = S      ") PSEUDO -SHIFT
                A = 1
                PLUSA(RECP0S)
        GET WORD: U, A = BUFEND, Z  ") NO = WITHIN BUFFER
                S = MA[-1]          ") YES = BEGIN OR END OF BUFFER
                STRWORD = S         " MEANINGLESS FOR BEGIN OF BUFFER
                RUS(18)
                S '+' 511
                N, GOTOR(MC[-1])
                G = CRBUNOE7
                U, A = :MG[1], Z    " BEGIN OF BUFFER?
                Y, GOTO(:ASSURE PRESENCE)
                MC = S              "SAVE BITS
                SUBCD(: FROM DRUM) " ASK FOR REFILL
                S = MC[-1]
                GOTOR(MC[-1])
        'END'
        " NEXT STRWORD
STILL EIGHTEEN: S = : JUST NINE   ")CHANGE PSEUDO-SHIFT
                BTSTREAM9 = S      ")
                S = STRWORD
                RUS(9)              " FETCH BITS
                S '+' 511
                GOTOR(MC[-1])

BITSTREAM18:    S = : NEXT STRWORD") PSEUDO-SHIFT INDICATES
        U, BTSTREAM9 = S , P      ") THAT THERE ARE 18 BITS IN STRINGWORD
        Y, GOTO(:FETCH FAST)
                SUBC(BTSTREAM9)
                LUS(9)
                MC = S
                SUBC(BTSTREAM9)
                S '+' MC[-1]
                GOTOR(MC[-1])

FETCH FAST:     BTSTREAM9 =S
                S = STRWORD        ")BECAUSE THERE ARE STILL
                S '+' D18M1        ")18 BITS IN STRINGWORD
                GOTOR(MC[-1])

```

"RECOVER SECTION NR 4.

```

BITSTREAM27:      SUBC(BTSTREAM9)      ")
                  RCS(9)                ")
                  MC = S                 ")
                  SUBC(:BITSTREAM18)    ")9+18 = 27
                  S '+' MC[-1]
                  GOTOR(MC[-1])

SKIP REST OF BUF: S = RECPOS            ")
                  S = CRBUNOE7, Z       ") COMPLETE NEW BUFFER?
N, SUBCD(:FROM DRUM) " IF NOT ASK FOR REFILL
                  GOTO(:JUST NINE)

FROM DRUM:        G = CRBUNOE7          " SELECT BUFFER
                  A = MG[-1]            ") BUFFER NOT PRESENT
                  MG[-1] = -A          ")
                  CRBUNOE7 = A         " SELECT NEXT BUFFER
                  RECPOS = A           ") INITIALIZE RECPOS
                  A + 1                ") DON'T RELEASE
                  BUFEND = A           ") BUFFER YET
                  A = DRROOM7, Z       " STILL DUTIES ON DRUM
Y, GOTOR(MC[-1]) " IF NOT NEGLECT STIMULI FOR REFILL
                  A = 1
                  QUEU7 + A           " ONE BUFFER MORE IN QUEUE
                  DRROOM7 -A          " ONE DUTY LESS ON DRUM
                  A = ACTIVE[7], P    " TRANSPORT ACTIVE?
Y, GOTOR(MC[-1]) " POSTPONE TRANSPORT
                  ACTIVE[7] = B       " NOTE ACTIVITY
                  S = 128
                  GOTO(:ATTENTION)    " ACTIVATE TRANSPORT

END RECOVER:      S = ACTIVE[7], P
Y, JUMP(-2)
                  GOTOR(MC[-1])

```

"RECOVER SECTION NR 5

'BEGIN' NO REFRESH

PROCESS7:

DR TO CR:

```

S = LABEL          " ) DETERMINE FIRST OCCUPIED
A = MS[-1]         " ) PLACE OF DRUM
S = CRBUNOF7       " FIRST WORD IN CORE
F = : RECBULEN     " NUMBER OF WORDS
SUBC(:DRSTART)     " START TRANSPORT
GOTOR(MC[-1])      " EXIT DR TO CR

```

"AFTER COMPLETION OF THE TRANSPORT, CONTRCL IS GIVEN TO
"THE NEXT INSTRUCTIONS

```

S = CRBUNOF7       " SELECT BUFFER
A = MS[-1]         " ) BUFFER PRESENT
MS[-1] = -A        " ) AGAIN
CRBUNOF7 = -A      " SELECT NEXT BUFFER
S = LABEL
A = : RECBULEN     " )
MS[-1] = A         " ) FIRST NON RECOVERED WORD
A = 1
MS[-2] = A, Z      " ANOTHER BUFFER IN START 1 ORDER?
N, GOTO(:NO REFRESH)
S = MS, Z          " ANOTHER START 1 ORDER?
LABEL = S          " REFRESH LABEL

```

```

NO REFRESH: N, QUEU7 = A, Z      " IE NO OTHER START 1 ORDER STOP TRANSPORT
Y, ACTIVE[7] = -B              " QUEU SHORTER. EMPTY WORD?
Y, S = 128                     " NOTE INACTIVITY
N, S = 0                        " CAUCEL ATTENTION
GOTO(:ATTENTION)              " CONTINUE ATTENTION

```

'END' DRTOCR

'END' RECOVER

" RUNNING SYSTEM

RUN VAR:

```

'BEGIN' STOCK, STOCK1, STOCK2, STOCK3, BEGIN OBJP, END OBJP,
        LOOSE STRING, REF, REF1, SRT, NBR, WS, WS1, ACTION

```

```

STOCK:      'SKIP' 1
STOCK1:     'SKIP' 1
STOCK2:     'SKIP' 1
STOCK3:     'SKIP' 1
BEGIN OBJP: 'SKIP' 1
END OBJP:   'SKIP' 1
LOOSE STRING: 'SKIP' 1
REF:        'SKIP' 1
REF1:       'SKIP' 1
SRT:        'SKIP' 1
NBR:        'SKIP' 1
WS:         'SKIP' 1
WS1:        'SKIP' 1
ACTION:     'SKIP' 1
FREEWO2:

```

" 14 VARIABLES

RUN SYST[0]:

'BEGIN' SCHOLTEN, HALF, ERRORTABLE, ENTRIS, ENTRIS4, EXITIS, ENTRPB,
ENTRB, DPTR, EXITP, CEN, CLPN, CRV, CRV3, CIV, CIV5, CBV, CLV,
IAD, BAD, RAD, JOINT AD, RAD35, LAD, IND, INDB, INDU, TFSU,
TSL, TFSL, TFSL1, TASR, TASI, TASB, TASB2, TASST, TASU, STSR,
STSR4, STSI, SSTSI, STSB, STSB3, STFSU, FAD, STASR, STASI,
STASB, STASST, STASU, TRSCV, TISCV, TSCVU, FADCV, JUA, REJST,
TIAV, TAV, TAV1, IDI, TTP, TCST, STST, STSST, STSST4, CSTV,
STAD, ORAD, OIAD, OBAD, OSTAD, EXITPC, FREE CHAIN, DECREASE,
DECREASE3, PUNLCR, PUSPACE, RUNOUT, REHEPG, PUHEPG, ENTIER,
SQRT, LN, EXP, EXP1, COS, SIN, ARCTAN

SCHOLTEN: + 12288
+ 0
HALF: - 65535
+ 1 " 4 LOCATIONS

ERRORTABLE: 'BEGIN' END OF TABLE

GOTO (:END OF TABLE)
GOTO (:END OF TABLE)
GOTO (:END OF TABLE)
GOTO (:END OF TABLE)
GOTO (:END OF TABLE)

GOTO (:END OF TABLE)
GOTO (:END OF TABLE)
GOTO (:END OF TABLE)
GOTO (:END OF TABLE)
GOTO (:END OF TABLE)

GOTO (:END OF TABLE)
GOTO (:END OF TABLE)
GOTO (:END OF TABLE)
GOTO (:END OF TABLE)
GOTO (:END OF TABLE)

END OF TABLE: A = M[B-1] " LINK
A = MA[-1] " CALL
A '*' 32767
A = :ERRORTABLE " ISOLATE ERROR NUMBER
A + 500
GOTO (:ERRORM) " 21 INSTRUCTIONS

'END' ERRORTABLE

```

ENTRIS:      A = D
              MC = A          " SAVE DP
              S = MS[1]
              D = S          " NEW VALUE OF DP = APIC[1]
ENTRIS4:     U, B = BCHECK, P  " DANGEROUS GROWTH OF STACK ?
              N, GOTO (LINK[2])
              GOTO (:CONSIDER) " 7 INSTRUCTIONS

EXITIS:      S = MC[-1]
              D = S          " RESET DP
              GOTO (MC[-1])  " 3 INSTRUCTIONS

ENTRPS:      D = A          " NEW VALUE OF DP
ENTRS:       MA = B         " WP
              MG = A         " PP
              GOTO (:ENTRIS4) " 4 INSTRUCTIONS

DPTR:        'BEGIN' DOUBLE, TEST, END

              G = D
              MC = F          " DP[CALL] IN LINKDATA
Y, GOTO (:END)              " NC TRANSPORT IN CASE LENGTH = 0
              STOCK = A
              GOTO (:TEST)

DOUBLE:      S = 2
              F = MS
              MC = F          " DOUBLE TRANSPORT

TEST:        A = 2, P
              N, GOTO (:DOUBLE)
              A = 2, Z
              N, A = MS[2]
              N, MC = A       " SINGLE TRANSPORT
              A = STOCK
END:          A = :MC[-2]     " CCNSTRUCT PP IN A
              GOTO (LINK)     " 16 INSTRUCTIONS

              'END' DPTR

EXITP:       B = :MD         " PP
              S = M[B+1]
              D = S          " RESET DP
Y, JUMP (4)      " TAKE CARE OF THE CONDITION
              S = -1, P      " LAST SIGN = NEGATIVE
U, B = DREF[1], E  " DELAY REQUIRED ?
N, GOTO (MC[-1])  " EXIT EXITP WITH CORRECT CONDITION
              JUMP (-3)      " WAIT
U, B = DREF[1], P  " DELAY SUPERFLUOUS ?
Y, GOTO (MC[-1])  " EXIT EXITP WITH CORRECT CONDITION
              JUMP (-3)      " 11 INSTRUCTIONS

```

CEN:

'BEGIN' FORMAL, MASK0, MASK19, TAR, TAI, TABO, TAST,
TAA, TAKE, ADD

```

S = 1
PLUSS (MA[-1]) " INCREASE LINK BY 1
S = MS[-1] " TAKE APD
F = :MS " ISOLATE ADDRESS
S '+' G " ISOLATE CODE
S + :MD " CODE + (CLEARED) DP
MC[1] = S " TOGETHER MAKE APIC[1]
RUS (15)
U, S '*' 8, Z " STATIC ADDRESS ?
RUS (5) " ISOLATE ORDINAL NUMBER
Y, JUMP (4)
G + MASK19
DO (G) " REDUCE DYNAMIC ADDRESS TO STATIC ONE
U, S - 15, Z " PASSED ON FORMAL ?
Y, GOTO (IFORMAL)
MC[1] = G " APIC[2] CONTAINS THE ADDRESS
U, S '*' 48, Z " ACTUAL PARAMETER SIMPLE ?
N, S = - 1 " ELSE SELECT SUBC (:M[0])
S + :MASK0 " ADDRESS ABOVE 32 K ?
F = 32767, P " THEN SELECT INSTRUCTION
Y, DO (TAKE) " ELSE ADD FUNCTION PART TO ADDRESS
N, DO (ADD) " APIC[0]
MC[-2] = G
GOTO (LINK)
S = MG " APIC[0]
M[B-1] = S " COPIED
F = MG[1] " APIC[1] AND APIC[2]
MC = F " COPIED
GOTO (LINK)
SUBC (:M[32767]) " COMPLICATED ACTUAL PARAMETERS
F = M[32767] " REAL VARIABLE
G = M[32767] " INTEGER VARIABLE
S = M[32767], P " BCOLEAN VARIABLE
A = M[32767] " STRING VARIABLE
F = M[32767] " REAL CONSTANT
G = M[32767] " INTEGER CONSTANT
S = 32767, Z " LOGICAL VALUE
F = 32767 " SMALL INTEGER CONSTANT
A = M[32767] " REAL ARRAY IDENTIFIER
A = M[32767] " INTEGER ARRAY IDENTIFIER
A = M[32767] " BOOLEAN ARRAY IDENTIFIER
A = M[32767] " STRING ARRAY IDENTIFIER
A = :M[32767] " SWITCH IDENTIFIER
F = - 32767 " NEGATIVE SMALL INTEGER CONSTANT
A = :M[32767] " LABEL
SUB (:TAR) " REAL VARIABLE ABOVE 32 K
SUB (:TAI) " INTEGER VARIABLE ABOVE 32 K
SUB (:TABO) " BCOLEAN VARIABLE ABOVE 32 K
SUB (:TAST) " STRING VARIABLE ABOVE 32 K
F = :M0[-256]

```

FORMAL:

MASK0:

MASK19:

```

TAR:      G = MS[2]      " ADDRESS OF VARIABLE
          F = MG         " VALUE OF VARIABLE
          GOTO (LINK)
          SUB (:TAA)      " REAL AARRAY IDENTIFIER ABOVE 32 K
          SUB (:TAA)      " INTEGER ARRAY IDENTIFIER ABOVE 32 K
          SUB (:TAA)      " BOOLEAN ARRAY IDENTIFIER ABOVE 32 K
          SUB (:TAA)      " STRING ARRAY IDENTIFIER ABOVE 32 K

TAKE:     G = MS[15]
ADD:      G + MS
TAST: TAA: A = MS[2]     " LABEL ABOVE 32 K / ADDRESS OF IDENT.
          A = MA
          GOTO (LINK)

TAI:      G = MS[2]      " ADDRESS OF VARIABLE
          G = MG         " VALUE OF VARIABLE
          GOTO (LINK)

TABO:     S = MS[2]      " ADDRESS OF VARIABLE
          S = MS, P      " VALUE OF VARIABLE
          GOTO (LINK)   " 68 INSTRUCTIONS

          'END' CEN

CLPN:     'BEGIN' SIMVAR, LIST, STAR, STAI, STABO, STAST, MASK

          SUB (:CEN)
          S = M[B-2]     " APIC[1]
          RUS (20)       " ISOLATE ORDINAL NUMBER
          U, S '1' 12, Z " ACTUAL PARAMETER ASSIGNABLE ?
          N, SUBC (:ERRORTABLE[0])
          U, S = 3, P     " ACTUAL PARAMETER COMPLICATED ?
          N, GOTO (:SIMVAR)
          G = M[B-3]     " APIC[0]
          F + 2          " INCREASED BY 2
          M[B-1] = G     " MAKES NEW APIC[2]
          S + :LIST
          S = MS[-12]    " SELECT PROPER INSTRUCTIONS FROM LIST
          MC = S         " APIC[3]
          GOTO (LINK[1])

SIMVAR:   G = :MS, Z     " REAL VARIABLE ?
          N, JUMP (3)
          S = M[B-1]
          U, S '1' 32767, Z " APIC[2] CONTAINS ADDRESS OF VARIABLE
          Y, S '1' MASK   " ADDRESS BELOW 32 K ?
          N, F + :LIST    " ADD M[0] = F AS FUNCTION PART
          N, S = MG
          MC = S
          GOTO (LINK[1])

LIST:     SUBC (:STAR)
          SUBC (:STAI)
          SUBC (:STABO)
          SUB2 (:STAST)
          SUB3 (:STASR)
          SUB3 (:STASI)

```

```

SUB3 (:STAB)
SUB3 (:STASST)
STAI:      U, S = F, Z      " ALREADY INTEGER ?
N, SUBC (:RND)
S = MS[-1]      " TAKE ADDRESS FROM APIC[2]
MS = G
GOTO (MC[-1])
STABO:     G = T      " SIGN BIT OF T CONTAINS CONDITION
S = MS[-1]      " TAKE ADDRESS FROM APIC[2]
MS = G
GOTO (MC[-1])
STAST:     S = MS[-1]      " TAKE ADDRESS FROM APIC[2]
F = :MS      " COPY IN G
GOTO (:STST)
LIST[20]:  SUB3 (:STASU)
STAR:      S = MS[-1]      " TAKE ADDRESS FROM APIC[2]
MS = F
GOTO (MC[-1])
MASK:      M[0] = F      " 48 INSTRUCTIONS

'END' CLPN

CRV:      MC = A      " SAVE PP
SUB (:CEN)
DOS (M[B-3])      " EXECUTE APIC[0]
CRV3:     B = 1      " REMOVE APIC FROM STACK
A = MC[-3]      " RESET PP IN A
S = MC[-3]      " TAKE LINK IN S
M[B-2] = F      " OVERWRITE APIC BY REAL/LABEL VALUE
GOTO (:MS)      " 8 INSTRUCTIONS

CIV:      MC = A      " SAVE PP
SUB (:CEN)
DOS (M[B-3])      " EXECUTE APIC[0]
S = F, Z      " ALREADY INTEGER ?
N, SUBC (:RND)
CIV5:     B = 2      " REMOVE APIC FROM STACK
A = MC[-2]      " RESET PP IN A
S = MC[-2]      " TAKE LINK IN S
M[B-1] = G      " OVERWRITE IT BY INT/BO VALUE
GOTO (:MS)      " 10 INSTRUCTIONS

CBV:      MC = A      " SAVE PP
SUB (:CEN)
DOS (M[B-3])      " EXECUTE APIC[0]
G = T      " SIGN BIT OF T CONTAINS CONDITION
GOTO (:CIV5)      " 5 INSTRUCTIONS

CLV:      MC = A      " SAVE PP
SUB (:CEN)
DOS (M[B-3])      " EXECUTE APIC[0]
F = MA      " TAKE LABEL VALUE
GOTO (:CRV3)      " 5 INSTRUCTIONS

```

```

IAD:      REF = A           " SAVE ADDRESS OF ARRAY KEY
          A = 5             " 4 * DELTA[0] + 1
          GOTO (:JOINT AD)  " 3 INSTRUCTIONS

BAD:      REF = A           " SAVE ADDRESS OF ARRAY KEY
          A = 6             " 4 * DELTA[0] + 2
          GOTO (:JOINT AD)  " 3 INSTRUCTIONS

RAD:      'BEGIN' LOOP, LOOP1, LOOP2

          REF = A           " SAVE ADDRESS OF ARRAY KEY
          A = 8             " 4 * DELTA[0] + 0
JOINT AD:  SRT = A           "
          NBR = G           " NUMBER OF ARRAYS
          COUNT = S         " DIMENSION
          LUS (2)           "
          B = :MS           " B = 4 * DIMENSION
          RUA (2)           " ISOLATE DELTA[0]
          S = :MA
          A = 2, Z
N, A = 0           " FORM = 0 OR + 0 FOR DELTA[0] = 2 OR 1
          F = - M[B]        " SAVE L[0]
          MC = A
          A = :MC[-1], Z    " CCNDITION NO
LOOP:      Y, F = - MA      " L[1]
          U, S = F, Z       " ALREADY INTEGER ?
          N, SUBC (:RND)
          F + 1
          MC = - G          " L[1] = 1 CONSTRUCTED, AVOIDING - 0
          F = MA[2]         " U[1]
          U, S = F, Z       " ALREADY INTEGER ?
          N, SUBC (:RND)
          U, S = G, Z
          Y, F = 0
          MC = G
          G = M[B-2], P    " AVOID = 0
          N, SUBC (:ERRORTABLE[1]) " U[1] READY
          G * S             " U[1] = L[1] + 1 > 0 ?
          MC = G            " * DELTA[1]
          S = G             " DELTA[1+1] = (U[1]-L[1]+1) * DELTA[1]
          A + 4
          REPP (:LOOP)
          A = :MC[-1]       " RECONSTRUCT DIMENSION
          MC = A            " AR[-2] = DIMENSION
          A = SRT
RAD35:    MC = A
LOOP1:    U, A = 6, Z       " AR[-1] = ARRAY TYPE
          RUA (2)           " BOOLEAN ARRAY ?
          F = :MC[1]        " ISOLATE DELTA[0]
          Y, F = 27         " CONSTRUCT ADDRESS OF FIRST ELEMENT
          G + S             " + DELTA[N]
          G = A             " - DELTA[0] -> ADDRESS OF LAST ELEMENT

```

```

      A = REF
      MA = B           " ARRAY KEY = ADDRESS OF AR[0]
      MC = G           " AR[0] = ADDRESS OF LAST ELEMENT
N, JUMP (3)           " CALCULATE ADDRESS OF FIRST FREE WORD
      A = 0
      DIVAS (27), Z   " FOR BCOLEAN ARRAYS: 27 ELEMENTS/WORD
N, S + 1
      B + S           " ADDRESS OF FIRST FREE WORD READY
U, B = BCHECK, P     " DANGEROUS GROWTH OF STACK ?
Y, SUB2 (:CONSIDER)
      A = 1
      NBR = A, P      " (STILL) MORE ARRAYS ?
N, GOTO (LINK[3])
      PLUSA (REF)
      S = MA[-1]      " ADDRESS OF PREVIOUS AR[0]
      A = MS[-2]      " DIMENSION
      COUNT = A
      S = :MA[3]
      S = :MA
      S = :MA         " ADDRESS OF PREVIOUS AR[-3*DIMENSION-3]
LOOP2: F = MS
      MC = F
      A = MS[2]
      MC = A
      S + 3
      REPE (:LOOP2)
      G = MS[-3]      " PREVIOUS AR[-3] = DELTA[N]
      S = G
      GOTO (:LOOP1)   " 71 INSTRUCTIONS

      'END' RAD

LAD:      'BEGIN' AGAIN

      LUS (18)        " DISPLAY LEVEL * D18
      S + :MD         " (CLEARED) DP ADDED
      G = LINK
AGAIN:    A = MG, P    " TAKE ADDRESS OF LABEL, ANY MORE ?
      MC[1] = S
Y, MC[-1] = A
      F + 1
Y, GOTO (:AGAIN)
      MC[-1] = - A
      GOTO (:MG)      " 10 INSTRUCTIONS

      'END' LAD

```

IND: 'BEGIN' LOOP

```

      S = :MA                      " ADDRESS AR[0]
U, S = F, Z                      " (LAST) INDEX ALREADY INTEGER ?
N, SUBC (:RND)
      A = G
U, A = MS[-5], P
      A = MS[-4], E              " OUTSIDE [LOWERBOUND : UPPERBOUND] ?
Y, SUBC (:ERRORTABLE[2])
      G = MS[-6], P              " TAKE DELTA[N-1]
N, LUA (1)                       " IF NEEDED, TAKE 2 INTO ACCOUNT
      F + 0, Z                  " N = 1 ?
Y, A + MS                         " + AR[0], I.E. + ADDRESS OF LAST ELMNT
Y, GOTO (LINK)
      G * A                      " (INDEX[N-1] - U[N-1]) * DELTA[N-1]
      G + MS                     " + AR[0], I.E. + ADDRESS OF LAST ELMNT
LOOP: WS = G
      S = 3                      " I = I - 1
      F = MC[-2]                 " INDEX[I]
U, S = F, Z                      " ALREADY INTEGER ?
N, SUBC (:RND)
      A = G
U, A = MS[-5], P
      A = MS[-4], E              " OUTSIDE [LOWERBOUND : UPPERBOUND] ?
Y, SUBC (:ERRORTABLE[3])
      G = MS[-6], P              " TAKE DELTA[I]
N, LUA (1)                       " IF NEEDED, TAKE 2 INTO ACCOUNT
      F + 0, Z                  " I = 0 ?
Y, A + WS
Y, GOTO (LINK)
      G * A                      " (INDEX[I] - U[I]) * DELTA[I]
      G + WS
      GOTO (:LOOP)              " 31 INSTRUCTIONS

      'END' IND

```

```

INDB: SUB (:IND)
      S = A
      A = 0
      DIVAS (27)                 " FOR BCOLEAN ARRAYS: 27 ELEMENTS/WORD
      F = :MC                    " SAVE B INTO G
      B = :MA                     " REMAINDER FROM DIVISION INTO B
      A = :MS                     " ADDRESS OF WORD CONTAINING REQUIRED
                                   " BIT INTO A
      S = 1
      LCS (B)                     " SHIFT BIT IN S TO THE RIGHT POSITION
      B = :MG                     " RESTORE B
      F = 2                       " FOR THE SAKE OF INDU
      GOTOR (LINK[1])             " 12 INSTRUCTIONS

```

```

INDU:      'BEGIN' SWITCH
           U, A - ENDOBJP, P
           U, A - BEGINOBJP, E      " A OUTSIDE PROGRAM RANGE ?
           N, GOTO (:SWITCH)
           S = MA[-1]              " AR[-1] CONTAINS THE ARRAY TYPE
           S '*' 3                  " ISOLATE THE TYPE
           U, S - 2, Z              " BOOLEAN ARRAY ?
           Y, GOTO (:INDB)
           WS1 = S                  " SAVE THE ARRAY TYPE
           SUB (:IND)
           G = WS1                  " DELIVER ARRAY TYPE IN F
           GOTO (LINK[1])
SWITCH:    STOCK3 = S              " SAVE THE ADDRESS OF APIC[0] FOR TFSL
           S = LINK[2]              " SAVE LINK OF TFSU BECAUSE OF DANGER OF
           MC = S                   " RECURSION
           S = LINK[1]              " SAVE LINK OF INDU BECAUSE OF DANGER OF
           MC = S                   " RECURSION
           GOTO (:TFSL1)            " 17 INSTRUCTIONS

           'END' INDU

TFSU:      'BEGIN' R, I, BO, ST, L
           SUB1 (:INDU)
           F + :R
           DO (MG)                  " EXECUTE SELECTED INSTRUCTION
           GOTO (LINK[2])
R:          F = MA                  " TSR
I:          G = MA                  " TSI
BO:         S '*' MA, Z             " TSB
ST:         A = MA                  " TSST
L:          GOTO (MC[-1])           " 9 INSTRUCTIONS

           'END' TFSU

TSL:       S = F, Z                " INDEX ALREADY INTEGER ?
           N, SUBC (:RND)
           S = G
           U, S - MA, P             " INDEX > N ?
           N, S - 0, E             " OR < 1 ?
           Y, SUBC (:ERRORTABLE[4])
           A + G
           DO (MA)                  " EXECUTE SELECTED SWORD
           GOTO (MC[-1])           " 9 INSTRUCTIONS

TFSL:      STOCK3 = S              " SAVE THE ADDRESS OF APIC[0]
TFSL1:     S = D
           MC = S                   " SAVE DP
           S = STOCK3
           S = MS[1]
           D = S                    " NEW VALUE OF DP = APIC[1]
           SUBC (:TSL)
           F = 4                    " FOR THE SAKE OF TFSU
           GOTO (:EXITIS)          " 9 INSTRUCTIONS

```

```

TASR:    SUB (:IND)
          F = MC[-2]
          D = G
          F = MA
          GOTO (MC[-1])
          " RESTORE DP SAVED BY ENTRIS
          " TSR
          " 5 INSTRUCTIONS

TAS1:    SUB (:IND)
          F = MC[-2]
          D = G
          G = MA
          GOTO (MC[-1])
          " RESTORE DP SAVED BY ENTRIS
          " TSI
          " 5 INSTRUCTIONS

TASB:    SUB1 (:INDB)
          S '*' MA, Z
          " TSB

TASB2:   F = MC[-2]
          D = G
          GOTO (MC[-1])
          " RESTORE DP SAVED BY ENTRIS
          " 5 INSTRUCTIONS

TASST:   SUB (:IND)
          A = MA
          GOTO (:TASB2)
          " TSST
          " 3 INSTRUCTIONS

TASL:    S = STOCK3
          SUB2 (:TFSU)
          S = MC[-1]
          D = S
          B = 1
          GOTO (MC[-1])
          " STOCK3 GOT ITS VALUE BY THE MACRO SAS
          " RESTORE DP SAVED BY ENTRIS
          " REMOVE EXTRA LINK
          " 6 INSTRUCTIONS

STSR:    STOCK = F
          A = MC[-1]
          F = MC[-2]
          SUB (:IND)
          " ADDRESS OF AR[0]
          " INDEX[N-1]

STSR4:   F = STOCK
          MA = F
          GOTO (LINK[2])
          " STR
          " 7 INSTRUCTIONS

STS1:    S = F, Z
          Y, GOTO (:SSTS1)
          F + SCHOLTEN
          F - SCHOLTEN
          S = F, Z
          " ALREADY INTEGER ?
          " HAS THE ROUNDING BEEN SUCCESSFUL ?
          N, SUBC (:ERRORTABLE[5])

SSTS1:   STOCK1 = G
          A = MC[-1]
          F = MC[-2]
          SUB (:IND)
          G = STOCK1
          MA = G
          GOTO (LINK[2])
          " ADDRESS OF AR[0]
          " INDEX[N-1]
          " SSTI
          " 13 INSTRUCTIONS

```

```

STSE:      A = MC[-1]          " ADDRESS OF AR[0]
           F = MC[-2]          " INDEX[N-1]
           SUB1 (:INDB)        " INDB DOES NOT DISTURB THE CONDITION
STSE3:      STOCK3 = S
           S = - S
           S '*' MA            " CLEAR OLD VALUE
           N, S '+' STOCK3     " SUBSTITUTE NEW VALUE
           MA = S
           GOTO (LINK[2])      " 9 INSTRUCTIONS

STFSU:      'BEGIN' R, I, BO, ST, L, INTEGER

           STOCK = F
           STOCK2 = A
           A = MC[-1]          " ADDRESS OF AR[0]
           F = MC[-2]          " INDEX[N-1]
           SUB1 (:INDU)
           JUMP (G)            " SWITCH OVER THE ARRAY TYPE
R:          GOTO (:STSR4)
I:          GOTO (:INTEGER)
BO:         GOTO (:STSB3)
ST:         GOTO (:STSB4)
L:          SUBC (:ERRORTABLE[6])
INTEGER:    F = STOCK
           S = F, Z            " ALREADY INTEGER ?
           N, SUBC (:RND)
           MA = G              " SSTI
           GOTO (LINK[2])      " 16 INSTRUCTIONS

           'END' STFSU

FAD:        MC = F              " STACK
           U, S = :MA          " STAA
           MC = A
           A = MS[1]
           D = A                " RESTORE DP SAVED BY ENTRIS
           GOTO (MS[-1])        " RETURN OVER LINK OF APIC[2] INSTRUCTN
                                   " 6 INSTRUCTIONS

STASR:      SUB2 (:STSR)
           B = 3                " REMOVE 2 OLD LINKS AND A DP
           GOTO (LINK[3])      " 3 INSTRUCTIONS

STASI:      SUB2 (:STSI)
           B = 3                " REMOVE 2 OLD LINKS AND A DP
           GOTO (LINK[3])      " 3 INSTRUCTIONS

STASB:      SUB2 (:STSB)
           B = 3                " REMOVE 2 OLD LINKS AND A DP
           GOTO (LINK[3])      " 3 INSTRUCTIONS

```

```

STASST:  SUB2 (:STSST)
          B = 3          " REMOVE 2 CLD LINKS AND A DP
          GOTO (LINK[3])  " 3 INSTRUCTIONS

STASU:   SUB2 (:STFSU)
          B = 3          " REMOVE 2 CLD LINKS AND A DP
          GOTO (LINK[3])  " 3 INSTRUCTIONS

TRSCV:   SUB (:IND)
          F = MA         " TSR
          B = 1          " REMOVE EXTRA LINK
          GOTO (MC[-1])   " 4 INSTRUCTIONS

TISCV:   SUB (:IND)
          G = MA         " TSI
          B = 1          " REMOVE EXTRA LINK
          GOTO (MC[-1])   " 4 INSTRUCTIONS

TSCVU:   'BEGIN' R, I, BO, ST, L

          SUB1 (:INDU)
          F + :R
          DO (MG)         " EXECUTE SELECTED INSTRUCTION
          B = 1          " REMOVE EXTRA LINK
          GOTO (MC[-1])

R:        F = MA         " TSR
I:        G = MA         " TSI
BO:       SUBC (:ERRORTABLE[7])
ST:       SUBC (:ERRORTABLE[8])
L:        SUBC (:ERRORTABLE[9]) " 10 INSTRUCTIONS

          'END' TSCVU

FADCV:   MC = F          " STACK
          U, S = :MA      " STAA
          MC = A
          GOTO (MS[-1])   " 4 INSTRUCTIONS

RND:     F + SCHOLTEN
          F - SCHOLTEN
          U, S = F, Z     " HAS THE ROUNDING BEEN SUCCESSFUL ?
          Y, GOTO (MC[-1])
          SUBC (:ERRORTABLE[10]) " 5 INSTRUCTIONS

```

```

JUA:      S = MA[1]
          D = S          " NEW VALUE OF DP
          RUS (18)       " ISOLATE DISPLAY LEVEL
          S + D
          S = MS         " TAKE PP
          B = MS         " B = WP
          U, B + DREF[1], P " DELAY SUPERFLUOUS ?
          N, JUMP (-2)    " WAIT
          A = MA
          MC = A         " CREATE PSEUDO LINK
          GOTO (:DECREASE) " 11 INSTRUCTIONS

REJST:    S = LOOSE STRING, P " DOES A CONTAIN A STRING REFERENCE ?
          N, GOTO (MC[-1])
          LOOSE STRING = - S
          S = MA         " TAKE PORCH[0]
          GOTO (:DECREASE3) " 5 INSTRUCTIONS

TIAV:     F = - 0
          GOTO (:TAV1)    " 2 INSTRUCTIONS

TAV:      'BEGIN' LOOP, LOOP1

          F = 1
TAV1:     REF = A         " SAVE THE ADDRESS OF ARRAY KEY
          DOS (MA)        " EXECUTE APIC[0]
          REF1 = A        " SAVE ADDRESS OF AR[0] OF ACTUAL ARRAY
          S = MA[-1]      " AR[-1] CONTAINS THE ARRAY TYPE
          S 'x' 3         " ISOLATE THE TYPE
          S - G, Z        " SPECIAL ACTIVITY REQUIRED ?
          ACTION = S
          G = MA[-2]
          COUNT = G
          F * 3           " DIMENSION
          A = :MG         " CONSTRUCT ADDRESS OF AR[- 3 * DIMENSION]
          S = MA[-3]      " FORM - 0 OR + 0 FOR DELTA[0] = 2 OR 1
          Y, MC = - S
          N, MC = S       " DELTA[0] ADAPTED
LOOP:     F = MA[-2]
          MC = F         " U[1] AND L[1] - 1 COPIED
          S = MA
          N, JUMP (3)     " TO BE HALVED ?
          G = ACTION, P
          Y, RUS (1)
          N, LUS (1), P  " ELSE TO BE DOUBLED (AND RESET CONDITION)
          MC = S         " DELTA[1] ADAPTED
          A + 3
          REPP (:LOOP)
          G = MA[-2]
          MC = G         " DIMENSION COPIED
          A = MA[-1]
          A 'x' 15
          Y, A 'x' 13    " ARRAY TYPE ADAPTED
          NBR = - A      " JUST FOR 1 ARRAY
          SUB3 (:RAD35)  " RESERVE SPACE IN THE STACK

```

```

      A = REF1
      S = MA[-1]
      S '*' 15
      U, S = 6, Z
      RUS (2)
      S + MA
      S = MA[-3]
      Y, A = 0
      Y, DIVAS (27)
      A = B
      B = REF
      B = M[B]
      B + 1
      LOOP1: G = MS
      U, S = ACTION, Z
      N, JUMP (6)
      U, S = - 0, E
      Y, MC = F
      Y, JUMP (4)
      F = MS
      SUBC (:RND)
      S + 1
      MC = G
      S + 1
      U, B = A, Z
      N, GOTO (:LOOP1)
      MD = B
      GOTO (LINK[4])

      'END' TAV

      IDI: S = MC[-3], Z
      Y, S = F, Z
      N, JUMP (6)
      S = MC[-1], P
      Y, A = 0
      N, A = - 0
      DIVAS (G)
      G = S
      GOTOR (MC[1])
      MC[1] = F, P
      U, S = - F, E
      N, F = MC[-4], P
      U, S = - F, E
      Y, SUBC (:ERRORTABLE[11])
      F / MC[1], P
      N, F = - F
      SUBC (:ENTIER)
      N, F = - F
      GOTOR (M[B+2])

      " AR[-1] OF ACTUAL ARRAY
      " CONTAINS THE ARRAY TYPE
      " BCOLEAN ARRAY ?
      " ISOLATE DELTA[0]
      " CONSTRUCT ADDRESS OF ELMNT AFTER THE LAST
      " CONSTRUCT ADDRESS OF FIRST ELEMENT

      " SAVE E IN A

      " ARRAY KEY CONTAINS ADDRESS AR[0]
      " CONSTRUCT ADDRESS OF FIRST ELEMENT
      " TAKE ELEMENT OF ACTUAL ARRAY
      " SPECIAL ACTIVITY REQUIRED ?
      " ELSE JUST COPY
      " TC BE DOUBLED ?
      " THEN STORE INTO 2 WORDS

      " HALVE; TAKE 2 WORDS
      " RCUND TO INTEGER

      " STORE INTO 1 WORD

      " READY ?

      " NEW VALUE OF WP
      " 60 INSTRUCTIONS

      " DIVIDEND A SMALL INTEGER ?
      " ^ DIVISOR ALSO ?

      " DIVIDEND IN S

      " SIGN CONSISTENT ZERO IN A
      " DIVIDEND & DIVISOR

      " EXIT IDI

      " DIVISOR NOT INTEGER AT ALL ?

      " ~ DIVIDEND NOT INTEGER AT ALL ?

      " TAKE ABSOLUTE VALUE OF QUOTIENT

      " ENTIER WITH CORRECT SIGN
      " 19 INSTRUCTIONS

```

```

TTP:      'BEGIN'  ODD TEST, LN MUL EXP, END, MUL, CYCLE, END1

          F + 0, Z          " EXPONENT = 0 ?
Y, F = 1          " THEN RESULT = 1
Y, GOTO (:END)
          MC = F
          F = M[B-5]        " TAKE BASE
          F + 0, Z          " = 0 ?
Y, S = M[B-1], P      " AND EXPONENT > 0 ?
Y, F = 0          " THEN RESULT = 0
Y, GOTO (:END1)
          S = - M[B-1], P
Y, S = - S
          A = - M[B-2], E    " EXPONENT INTEGER ?
N, GOTO (:LN MUL EXP)
          U, S + 32, P        " ABS (TAIL OF EXPONENT) ≤ 31 ?
Y, A + 0, Z          " ^ HEAD OF EXPONENT = 0 ?
Y, GOTO (:MUL)
          U, S = - G, P
          Y, F = - F
ODD TEST: Y, RUA (15), Z
          Y, RCS (1), P
LN MUL EXP: SUBC (:LN)
          F * MC[-2]
          SUBC (:EXP1)
          Y, F = - F
END:      B = 2
          GOTOR (MC[1])
MUL:      F = 1, P
          M[B-2] = F
CYCLE:    F = M[B-5]
          N, F * F
          N, M[B-5] = F
          U, S '*' 1, Z
          Y, F * M[B-2]
          Y, M[B-2] = F
          RUS (1), Z
          N, GOTO (:CYCLE)
          A + 0, P
          Y, F = 1
          Y, F / M[B-2]
END1:     B = 4
          GOTOR (MC[1])

          " START CYCLE WITH 1 AND CONDITION YES
          " BASE ↑ (2 ↑ 1)
          " BECOMES (EXCEPT FOR THE FIRST TIME)
          " BASE ↑ (2 ↑ (1+1))
          " THIS POWER OF BASE OF INTEREST ?

          " THEN INCORPORATE IT IN THE RESULT
          " READY ?

          " WAS EXPONENT ORIGINALLY NEGATIVE ?

          " THEN INVERT THE RESULT

          " 41 INSTRUCTIONS

          'END' TTP

```

RUN:

```

B = INSTR CNTR
A = MEMORY USABLE TO
A = M[B-1]
A = 20
BCHECK = A
A = DPO
D = A " DP
END OBJP = B
LOOSE STRING = - B " LOOSE STRING = FALSE
A = :MC " TEL (0)
F = :MD[0]
B + 1 " ENTRB (1)
SUB2 (:ENTRB)
A = BEGIN OF PROGRAM
BEGIN OBJP = A
GOTO (A) " 16 INSTRUCTIONS

```

TCST:

```

A = 600
SUBC (:ERRORM)

```

STST:

```

A = 601
SUBC (:ERRORM)

```

STSST: STSST4:

```

A = 602
SUBC (:ERRORM)

```

CSTV:

```

A = 603
SUBC (:ERRORM)

```

STAD:

```

A = 604
SUBC (:ERRORM)

```

ORAD: OIAD:
OBAD: OSTAD:

```

A = 605
SUBC (:ERRORM)

```

EXITPC:

```

A = 606
SUBC (:ERRORM)

```

FREE CHAIN:

```

A = 607
SUBC (:ERRORM)

```

DECREASE:

GOTO (MC[-1])

DECREASE3:

```

A = 608
SUBC (:ERRORM) " 19 INSTRUCTIONS

```

LIBRARY:

" INSTRUCT LIST

INSTR LST[0]:	MC = F	" DIV, STACK
	F = MC[-4]	
	F / MC	
	F = - F	" SUB, NEG
	F + MC[-2]	" ADD
	F * MC[-2]	" MUL
	SUBC (:IDI)	" IDI
	SUBC (:TTP)	" TTP
	F = MC[-2], Z	" EGU, UGU, TEST1
	S = - T, P	" NCN
	F = MC[-2], P	" LES
Y, F = 0, P		
	F = MC[-2], P	" MST, MOR
N, F = F, Z		
	S = - T, P	
	F = MC[-2], Z	" LST
N, S = - 1, E		" STAB
	S = T	
	MC = S	" QVL
	S = T, P	
	S = MC[-1], E	
Y, B = 1		" IMP
N, S = - MC[-1], P		
Y, B = 1		" OR
N, S = MC[-1], P		
N, B = 1		" AND
Y, S = MC[-1], P		
	MC = A	" STAA
	SUB (:IND)	" TSR
	F = MA	
	SUB (:IND)	" TSI
	G = MA	
	SUB1 (:INDB)	" TSB
	S '*' MA, Z	
	SUB (:IND)	" TSST
	A = MA	
	SUB2 (:TFSU)	" TFSU
	SUBC (:TSL)	" TSL
	SUBC (:TFSL)	" TFSL
	SUBC (:TCST)	" TCST
	SUB2 (:STSR)	" STSR
	SUB2 (:STS1)	" STS1
	SUB2 (:SSTS1)	" SSTS1
	SUB2 (:STSB)	" STSB
	SUB2 (:STSST)	" STSST
	SUB2 (:STFSU)	" STFSU
	SUB2 (:ENTRIS)	" ENTRIS
	S = MS[1]	" TFD
	STOCK3 = S	" SAS
	S = 2	" DECS

" INSTRUCT LIST CONTINUED

INSTR LST[50]:	GOTO (:FAD)	" FAD
	GOTO (:TASR)	" TASR
	GOTO (:TASI)	" TASI
	GOTO (:TASB)	" TASB
	GOTO (:TASST)	" TASST
	GOTO (:TASU)	" TASU
	GOTO (:EXITIS)	" EXITIS
	GOTO (:FADCV)	" FADCV
	GOTO (:TRSCV)	" TRSCV
	GOTO (:TISCV)	" TISCV
	GOTO (:TSCVU)	" TSCVU
	GOTOR (MC[-1])	" EXIT
N, F = F, E		" TEST2
N, F = F, Z		
	SUBC (:CRV)	" CRV
	SUBC (:CIV)	" CIV
	SUBC (:CBV)	" CBV
	SUB1 (:CSTV)	" CSTV
	SUBC (:CLV)	" CLV
	SUB (:CEN)	" CEN
	SUB1 (:CLPN)	" CLPN
	SUB4 (:TAV)	" TAV
	SUB4 (:TIAY)	" TIAY
	SUB3 (:RAD)	" RAD
	SUB3 (:IAD)	" IAD
	SUB3 (:BAD)	" BAD
	SUB3 (:STAD)	" STAD
	SUB3 (:ORAD)	" ORAD
	SUB3 (:OIAD)	" OIAD
	SUB3 (:OBAD)	" OBAD
	SUB3 (:OSTAD)	" OSTAD
	LOOSE STRING = 0	" LOS
	GOTO (:EXITP)	" EXITP
	GOTO (:EXITPC)	" EXITPC
	SUBC (:REJST)	" REJST
	GOTO (:JUA)	" JUA
	F = F, P	" ABS
N, F = - F		
	F = F, Z	" SIGN
N, F = 1, E		
N, F = - 1		
	SUBC (:ENTIER)	" ENTIER
	SUBC (:SQRT)	" SQRT
	SUBC (:EXP)	" EXP
	SUBC (:LN)	" LN
	GOTO (:END RUN)	" END
	F = M[0]	" TRV, TRC
	G = M[0]	" TIV, TIC
	F = 0	" TSIC, TNA, STST
	SUB2 (:STST)	

" INSTRUCT LIST CONTINUED

```

INSTR LST[100]:  F = - M[0]
                  G = - M[0]
                  F = 0
                  F + M[0]
                  G + M[0]
                  F + 0
                  F - M[0]
                  G - M[0]
                  F = 0
                  F * M[0]

                  G * M[0]
                  G = 0
                  F / M[0]
                  G / M[0]
                  F / 0
                  F - M[0], Z
                  S = - T, P
                  G - M[0], Z
                  S = - T, P
                  F = 0, Z

                  S = - T, P
                  F - M[0], P
N, F = F, Z
                  S = - T, P
                  G - M[0], P
N, F = F, Z
                  S = - T, P
                  F = 0, P
N, F = F, Z
                  S = - T, P

                  F - M[0], Z
N, S = - 1, E
                  G - M[0], Z
N, S = - 1, E
                  F = 0, Z
N, S = - 1, E
                  F - M[0], P
Y, F = 0, P
                  G - M[0], P
Y, F = 0, P

                  S = M[0], P
                  S = 0, Z
                  A = M[0]
                  A = 0
                  GOTO (:MA[1])
                  M[0] = F
                  S = F, Z
N, SUBC (:RND)
                  M[0] = G
                  S = T

                  " TNRV, TNRC
                  " TNIV, TNIC
                  " TNSIC
                  " ADDR, ADDR
                  " ADDIV, ADDIC
                  " ADDSIC
                  " SUBRV, SUBRC
                  " SUBIV, SUBIC
                  " SUBSIC
                  " MULRV, MULRC

                  " MULIV, MULIC
                  " MULSIC
                  " DIVRV, DIVRC
                  " DIVIV, DIVIC
                  " DIVSIC
                  " EQUVR, EQUVC, UQURV, UQURC
                  " EQUIV, EQUIC, UQUIV, UQUIC
                  " EQUVIC, UQUSIC

                  " LESRV, LESRC, LSTRV, LSTRC

                  " LESIV, LESIC, LSTIV, LSTIC

                  " LESSIC, LSTSIC, MORVIC

                  " MSTRV, MSTRC

                  " MSTIV, MSTIC

                  " MSTSIC

                  " MORRV, MORRC

                  " MORIV, MORIC

                  " TBV
                  " TBC
                  " TSTV, TAK
                  " TLV, TSWE, TAA, JU1

                  " STR
                  " STI

                  " SSTI
                  " STB

```

" INSTRUCT LIST CONTINUED

```

INSTR LST[150]:  M[0] = S
                  DOS (M[0])          " DCS
                  DOS (M[2])          " DCS2
                  DOS (M[3])          " DCS3
                  GOTO (:M[0])        " JU
                  GOTO (M[0])         " IJU
                  GOTO (M[1])         " IJU1
N, GOTO (:M[0])    " COJU
Y, GOTO (:M[0])    " YCOJU
                  SUBC (:M[0])        " SUBJ

                  SUBC (M[0])         " ISUBJ
                  B = 0               " DECB
                  DO (M[0])           " DO
                  A = :MC             " TBL
                  F = :MD[0]
                  B + 0               " INCRB, ENTRB
                  SUB2 (:ENTRB)
                  S = D               " DPTR
                  '0327 77776'
                  SUB (:DPTR)

                  F = :MA[0]          " TDL
                  B + 0               " ENTRPB
                  SUB2 (:ENTRPB)
                  A + M[0]            " NIL
                  A = M[0]            " LAST
                  S = 0               " TDA, LAD
                  SUB (:LAD)
M0[0] = B          " SWP
B = MD[0]          " EXITB, EXITC
SUBC (:FREE CHAIN)

S = :MC[0]         " EXITSV
GOTO (MS)
SUBC (:SIN)        " SIN
SUBC (:COS)        " COS
SUBC (:ARCTAN)     " ARCTAN
SUBC (:READ)       " READ
SUBC (:FIXP)       " FIXP
SUBC (:ABSFIXP)    " ABSFIXP
SUBC (:FLOP)       " FLOP
SUBC (:PUNCH)      " PUNCH

SUBC (:PUNLCR)     " PUNLCR
SUBC (:PUSPACE)    " PUSPACE
SUBC (:RUNOUT)     " RUNOUT
SUBC (:REHEPG)     " REHEP
SUBC (:PUHEPG)     " PUHEP
SUBC (:HAND)       " HAND
SUBC (:XEEN)       " XEEN
SUBC (:STOP)       " STOP
S = LINE COUNTER   " SLNC
M[2] = S

```

" INSTRUCT LIST CONTINUED

```
INSTR LST(200):      S = M[2]           " RLNC
                     LINE COUNTER = S
                     S = 0              " LNC
                     LINE COUNTER = S
                     B = MD[0]          " EXITB
U, B + DREF[1], P
N, JUMP (-2)
U, S = :MA            " STAA
MC = A
```

" LIBRARY NR. 1

```
'BEGIN'  FIXT L, ABSFIXT L, FLOT L, PRINT L, NLCR L, TAB L,
        SPACE L, PRINTTEXT L, CARRIAGE L, NEW PAGE L,
        LINE NUMBER L, RESYM L, PUSYM L, PRSYM L, PUTEXT L,
        ABS L, SIGN L, SQRT L, SIN L, COS L, ARCTAN L, LN L,
        EXP L, ENTIER L, READ L, FIXP L, ABSFIXP L, FLOP L,
        PUNCH L, PUNLCR L, PUSPACE L, RUNOUT L, REHEP L, PUHEP L,
        HAND L, XEEN L, STOP L, FROM DRUM L, TO DRUM L, TIME L,
        SS L, MM L, TM L, MT L, MV L, TV L, VV L,
        TO DRUM, FROM DRUM, TIME, MATMAT, BKJ, MATMAT8,
        MATMAT9, TAMMAT, MATTAM, BJK, MATVEC,
        BK, TAMVEC, VECVEC, PLIST22, PLIST222, PLIST 223,
        PLIST 21, PLIST11, PLIST2, PLIST25, PLIST1, TEST DIM,
        IND1, IND2, AIK, AKI, INPR, RR, RI, AI, IR, II,
        STRING SYMBOL, TNRP, TRP, ABS P, SIGN P, SQRT P,
        SIN P, COS P, ARCTAN P, LN P, EXP P, ENTIER P, FIXP P,
        ABSFIXP P, FLOP P, PUNCH P, PUSPACE P, PUHEP P, HAND P,
        XEEN P, PUTEXT, FIXT P, ABSFIXT P, FLOT P, PRINT P,
        SPACE P, PRINTTEXT, CARRIAGE P, LINE NUMBER P, RESYM,
        PUSYM, PRSYM, POL IN FF, POL IN F, ENTIER1
```

LIBRARY[0]:

```
FIXT L:      :FIXT P
              :FIXT P
ABSFIXT L:   :ABSFIXT P
              :ABSFIXT P
FLOT L:      :FLOT P
              :FLOT P
PRINT L:     :PRINT P
              :PRINT P
NLCR L:      :NLCR
              :NLCR
TAB L:       :TAB
              :TAB
SPACE L:     :SPACE P
              :SPACE P
PRINTTEXT L: :PRINTTEXT
              :PRINTTEXT
CARRIAGE L:  :CARRIAGE P
              :CARRIAGE P
NEW PAGE L:  :NEW PAGE
              :NEW PAGE
LINE NUMBER L: :LINE NUMBER P
              :LINE NUMBER P
RESYM L:     :RESYM
              :RESYM
PUSYM L:     :PUSYM
              :PUSYM
PRSYM L:     :PRSYM
              :PRSYM
PUTEXT L:    :PUTEXT
ABS L:       :PUTEXT
SIGN L:      :ABS P
SQRT L:      :SIGN P
SIN L:       :SQRT P
COS L:       :SIN P
ARCTAN L:    :COS P
LN L:        :ARCTAN P
```

EXP L:	:LN P
ENTIER L:	:EXP P
READ L:	:ENTIER P
FIXP L:	:READ
ABSFIXP L:	:FIXP P
FLOP L:	:ABSFIXP P

" LIBRARY NR. 2

PUNCH L:	:FLOP P
PUNLCR L:	:PUNCH P
PUSPACE L:	:PUNLCR
RUNOUT L:	:PUSPACE P
REHEP L:	:RUNOUT
PUHEP L:	:REHEP G
HAND L:	:PUHEP P
XEEN L:	:HAND P
STOP L:	:XEEN P
	:STOP
FROM DRUM L:	:FROM DRUM
	:FROM DRUM
TO DRUM L:	:TO DRUM
	:TO DRUM
TIME L:	:TIME
	:TIME
SS L:	:STRING SYMBOL
	:STRING SYMBOL
MM L:	:MATMAT
	:MATMAT
TM L:	:TAMMAT
	:TAMMAT
MT L:	:MATTAM
	:MATTAM
MV L:	:MATVEC
	:MATVEC
TV L:	:TAMVEC
	:TAMVEC
VV L:	:VECVEC
	:VECVEC

"72 WORDS

" LIBRARY ROUTINES NR. 1

TNRP:	A = MC[-1] S = MC[-1] MC = F MC = S MC = A	" LINK TNRP " LINK LIBRARY ROUTINE " LAST PARAMETER " LINK LIBRARY ROUTINE " LINK TNRP
TRP:	A = :MC[-1] SUBC (:CRV) B = 2 GOTOR (MC[-1])	" 9 INSTRUCTIONS
ABS P:	SUBC (:TRP) F = F, P N, F = - F GOTOR (MC[-1])	" 4 INSTRUCTIONS
SIGN P:	SUBC (:TRP) F = F, Z N, F = 1, E N, F = - 1 GOTOR (MC[-1])	" 5 INSTRUCTIONS
SQRT P:	SUBC (:TRP) GOTO (:SQRT)	" 2 INSTRUCTIONS
SIN P:	SUBC (:TRP) GOTO (:SIN)	" 2 INSTRUCTIONS
COS P:	SUBC (:TRP) GOTO (:COS)	" 2 INSTRUCTIONS
ARCTAN P:	SUBC (:TRP) GOTO (:ARCTAN)	" 2 INSTRUCTIONS
LN P:	SUBC (:TRP) GOTO (:LN)	" 2 INSTRUCTIONS
EXP P:	SUBC (:TRP) GOTO (:EXP)	" 2 INSTRUCTIONS
ENTIER P:	SUBC (:TRP) GOTO (:ENTIER)	" 2 INSTRUCTIONS
FIXP P:	SUBC (:TRP) SUBC (:TNRP) SUBC (:TNRP) GOTO (:FIXP)	" 4 INSTRUCTIONS

" LIBRARY ROUTINES NR. 2

```

ABSFIXP P:      SUBC (:TRP)
                 SUBC (:TNRP)
                 SUBC (:TNRP)
                 GOTO (:ABSFIXP)      " 4 INSTRUCTIONS

FLOP P:         SUBC (:TRP)
                 SUBC (:TNRP)
                 SUBC (:TNRP)
                 GOTO (:FLOP)        " 4 INSTRUCTIONS

PUNCH P:        SUBC (:TRP)
                 GOTO (:PUNCH)       " 2 INSTRUCTIONS

PUNLCR:         S = 26
                 GOTO (:FLEXHP)      " 2 INSTRUCTIONS

PUSPACE P:      SUBC (:TRP)
PUSPACE:        SUBC (:RND)
                 COUNT = G, P        " ROUNDO TO INTEGER
                                     " > 0 ?
PUSPACE[2]:     S = 16
Y, SUBC (:FLEXHP)
Y, REPP (:PUSPACE[2])
                 GOTOR (MC[-1])      " 7 INSTRUCTIONS

RUNCUT:         F = 80
                 COUNT = G
RUNCUT[2]:      S = 0
                 SUBC (:FLEXHP)
                 REPP (:RUNCUT[2])
                 GOTOR (MC[-1])      " 6 INSTRUCTIONS

REHEP G:        SUBC (:REHEP)
                 G = S
                 GOTOR (MC[-1])      " 3 INSTRUCTIONS

PUHEP P:        SUBC (:TRP)
PUHEP G:        SUBC (:RND)
                 S = G, Z            " ROUNDO TO INTEGER
Y, S = 0
                 GOTO (:FLEXHP)      " AVOID = 0
                                     " 5 INSTRUCTIONS

HAND P:         SUBC (:TRP)
                 GOTO (:HAND)        " 2 INSTRUCTIONS

XEEN P:         SUBC (:TRP)
                 GOTO (:XEEN)        " 2 INSTRUCTIONS

```

" LIBRARY ROUTINES NR. 3

```

PUTEXT:      'BEGIN' NEW WORD, LOOP, CALL OF TCST

              S = :FLEXIS
              MC = S              " OUTPUT CODE = FLEXO
              A = :MC[-1]
              SUB (:CEN)
              S = MC[-2]          " APIC[1]
              RUS (20)            " ISOLATE COPY OF APD CODE
              S = 23, Z          " UNASSIGNABLE STRING EXPRESSION ?
              S = MC[-2]          " APIC[0]
              S '*' 32767         " ISOLATE BEGIN ADDRESS OF ISR
              A = MS[1]           " TAKE SECOND INSTRUCTION OF ISR
              Y, A = CALL OF TCST, Z " IS IT A CALL OF TCST ?
              N, A = 611
              N, SUBC (:ERRORM)   " ELSE THE EXECUTION IS ENDED
NEW WORD:    S + 1
              M[B-1] = S         " SAVE ADDRESS OF WORD IN ISR
              A = 3
              COUNT = A          " 3 SYMBOLS PER WORD
              S = MS[2]          " WCRD
              LCS (2)            " SHIFT OUT IRRELEVANT BITS
LOOP:        LUAS (8)            " SHIFT NEXT SYMBOL INTO A
              A '*' 255
              U, A = 255, Z      " END MARKER ?
              N, SUBC (M[B-2])   " DO THE CONVERSION AND PUT OUT
              N, REPP (:LOOP)
              N, S = M[B-1]      " ADDRESS OF WORD IN ISR
              N, GOTO (:NEW WORD) " HANDLE NEXT STRING WORD
              B = 2
              GOTOR (MC[-1])
CALL OF TCST: SUBC (:TCST)       " 29 INSTRUCTIONS

              'END' PUTEXT

FIXT P:      SUBC (:TRP)
              SUBC (:TNRP)
              SUBC (:TNRP)
              GOTO (:FIXT)      " 4 INSTRUCTIONS

ABSFIXT P:   SUBC (:TRP)
              SUBC (:TNRP)
              SUBC (:TNRP)
              GOTO (:ABSFIXT)   " 4 INSTRUCTIONS

FLOT P:      SUBC (:TRP)
              SUBC (:TNRP)
              SUBC (:TNRP)
              GOTO (:FLOT)      " 4 INSTRUCTIONS

PRINT P:     SUBC (:TRP)
              GOTO (:PRINT)     " 2 INSTRUCTIONS

SPACE P:     SUBC (:TRP)
              GOTO (:SPACE)     " 2 INSTRUCTIONS

```

" LIBRARY ROUTINES NR. 4

```

PRINTTEXT:      S = :PRNTIS
                  GOTO (:PUTEXT[1])      " 2 INSTRUCTIONS

CARRIAGE P:      SUBC (:TRP)
                  GOTO (:CARRIAGE)      " 2 INSTRUCTIONS

LINE NUMBER P:   G = LINE NUMBER
                  GOTOR (MC[-1])      " 2 INSTRUCTIONS

RESYM:           SUBC (:NXT TAPE SL)
                  U, S = 123, Z      " SPACE ?
                  Y, S = 93
                  U, S = 124, Z      " OR COLON ?
                  Y, S = 90
                  U, S = 125, Z      " OR CLCSE ?
                  Y, S = 99
                  G = S
                  GOTOR (MC[-1])      " 9 INSTRUCTIONS

PUSYM:           SUBC (:TRP)
                  SUBC (:RND)      " RCUND TO INTEGER
                  A = G, Z
                  Y, A = 0      " AVOID = 0
                  GOTO (:FLEXIS)    " 5 INSTRUCTIONS

PRSYM:           SUBC (:TRP)
                  SUBC (:RND)      " RCUND TO INTEGER
                  A = G, Z
                  Y, A = 0      " AVOID = 0
                  GOTO (:PRNTIS)    " 5 INSTRUCTIONS

POL IN FF:       'BEGIN' START, CYCLE

POL IN F:        F * F
                  M[B] = F
                  U, GOTO (:START)    " CLEAR OF
                  F = MA      " TAKE LAST COEFFICIENT
START:           F * M[B]      " * ARGUMENT
CYCLE:           F + MA[-2]    " + COEFFICIENT
                  U, GOTOR (MC[-1])    " IF OF THEN EXIT
                  A = 2
                  GOTO (:CYCLE)

                  'END' POL IN FF    " 9 INSTRUCTIONS

ENTIER:          F + 0, P
ENTIER1:         U, S = F, E      " ARGUMENT ALREADY INTEGER ?
                  Y, GOTOR (MC[-1])
                  F = HALF, E      " +0 < F < .5 ?
                  Y, JUMP (4)      " THEN ENTIER = + 0
                  U, S = F, E      " F = .5 INTEGER ?
                  Y, GOTOR (MC[-1])

```

" LIBRARY ROUTINES NR. 5

```

      F + SCHOLTEN
      F = SCHOLTEN, Z
Y, F = 0
      GOTOR (MC[-1]) " 11 INSTRUCTIONS

SQRT:      'BEGIN' D16, C0

      F = 0, P      " ARGUMENT ≤ 0 ?
N, F = 0      " THEN RESULT = + 0
N, GOTOR (MC[-1])
      MC = F      " PRESERVE X
      A = F
      RUA (15)      " ISOLATE
      MC = A      " AND PRESERVE BINARY EXPONENT
      A = F
      S = G
      F = :MC      " PRESERVE STACK POINTER
      A '*' 32767      " ISOLATE MANTISSA IN AS
      NORAS
      B = - :MC
      B = 26
      S = B      " BINARY EXPONENT + 2 IN S
      B = :MG      " RESTORE STACK POINTER
      S + MC[-1], P      " FORM COMPLETE BINARY EXPONENT + 2
Y, S + 1
U, S '*' 1, Z      " WAS IT ODD ?
      S '*' - 1
      MC = S
Y, RUA (1)
      MC = A
      RUA (3)      " ESTIMATE SQRT (A):
      MC = A
      RUA (2)
      A + C0
      M[B-1] + A      " X0 = .3657 + (5/8) * A
      A = M[B-2]
      RUA (4)
      DIVA (M[B-1])      " FIRST NEWTON STEP:
      M[B-1] + S      " X1 = (A/X0 + X0) / 2
      A = MC[-2]
      RUA (2)
      DIVA (MC)      " SECOND NEWTON STEP:
      S + M[B+1]      " X2 = (A/X1 + X1) / 2
      A = MC[-1]      " TAKE BINARY EXPONENT
      LUAS (14), P      " TRANSFORM 2 * X2 INTO FLOATING POINT
      F = MC[-2]      " RESTORE X
      F / A
      A = D16, E
N, A = 32767      " AS = X2/2
      F + A
      GOTOR (MC[-1])
      '000 200 000'
      + 61 35102      " .09142

D16:
C0:
      'END' SQRT      " 46 LOCATIONS

```

" LIBRARY ROUTINES NR. 6

```

LN:      'BEGIN'  CYCLE, LIST, LN X, LN Y, C1, C3, C5,
          LN2, ONE, GIANT, BIN EXP 40

          F = 0, P      " ARGUMENT > 0 ?
N, F = - GIANT
N, GOTOR (MC[-1]) " ELSE DELIVER - INFINITY
          A = F
          RUA (15)      " ISOLATE
          MC = A        " AND PRESERVE BINARY EXPONENT
          A = F
          A '*' 32767    " REPLACE BINARY EXPONENT
          A + BIN EXP 40 " BY 40
          S = G
          F = A
          F + 0          " STANDARDIZE
          A = F          " AND ISOLATE ANEW
          RUA (15)      " BINARY EXPONENT
          M[B-1] + A    " ADD IT TO FORMER BINARY EXPONENT
          A = F          " REPLACE
          A '*' 32767    " BINARY EXPONENT BY 0
          S = G
          F = A          " AND CONSIDER 40 BITS OF F AS FRACTION
          LUAS (12)     " ANALYSE FIRST 27 BITS OF FRACTION
          S = 0
          M[B] = A
CYCLE:   RUA (3)        " IN ORDER TO
          PLUSA (M[B]), P
Y, S = 1          " BRING FRACTION
Y, GOTO (:CYCLE)
          A = :LIST     " IN RANGE SQRT (8/9) < F < SQRT (9/8)
          A = S
          G * MA        " MULTIPLY FRACTION BY APPROPRIATE POWER OF 9/8
          MC = F        " AND PRESERVE RESULTING FRACTION
          F + ONE
          MC = F        " COMPUTE
          F = ONE       " F1 =
          F - MC[-4]    " (1 - F) /
          F / MC        " (1 + F)
          MC = F        " PRESERVE F1
          F * F
          MC = F
          F * C5        " COMPUTE
          F + C3        " POLYNOMIAL
          F * MC[-2]    " IN
          F + C1        " F1 SQUARE
          F * MC[-2]    " RESULT * F1
          MC = F
          G = S, Z
N, F * LN Y          " APPROPRIATE MULTIPLE OF LN (8/9)
          F + MC[-2]    " ADDED
          F + LN X     " AND (1/2) * LN (9/8) ADDED
          MC = - F
          G = MC[-3], Z " TAKE BINARY EXPONENT
N, F * LN2           " * LN (2)
          F + MC[-1]
          GOTOR (MC[-1])
LIST:    + 32768      " 2+15
          + 36864     " (9/8) * 2+15

```

100470 -

1

131

+

41472

" (9/8)42 * 2415

" LIBRARY ROUTINES NR. 7

```

+      46656      " (9/8)↑3 * 2↑15
+      52488      " (9/8)↑4 * 2↑15
+      59049      " (9/8)↑5 * 2↑15
LN X:      - 14 59121      " LN (9/8) / 2 = +.588915178282...-1
+      38 95639
LN Y:      + 14 26353      " LN (8/9) = -.117783035656
+      38 95640
C1:        - 12 69759      " +.2000000000002...+1
+      6
C3:        - 13 32565      " +.6666666478939
+      445 32834
C5:        - 13 63134      " +.400433275889
+      266 81432
LN2:       - 13 32131      " LN (2) = +.693147180560
+      351 25202
ONE:       + 5 06966      " SQRT (8/9) * 2↑55 = +.339682755868...+17
+      659 90648
GIANT:     '3777 37777'    " +.177664619751...+629
'3777 77777'
BIN EXP 40: + 13 10720    " 40 * D15
'END' LN      " 76 LOCATIONS

EXP:       'BEGIN' ENTIRE, C0, C1, C2, C3, C4, C5, C6, C7, LOG E,
           OMEGA

EXP1:      F + 0
N, M[B] = F, P      " ABS (X)
F = - F            " > 1447 ?
F = 1447, P
N, F = M[B]
Y, F = 1447
Y, S = - M[B+1], P
Y, F = - 1447
F * LOG E
MC = F, P
SUBC (:ENTIER1)
A = G
F = MC[-2], Z
Y, F = 1
Y, GOTO (:ENTIRE)
MC = A
F = - F
F = HALF, P
F = HALF
N, F * HALF
A = :C7
SUBC (:POL IN F)
N, F * F
A = MC[-1]
A + 1
ENTIRE:    U, A + 2048, P
U, A = 2047, E
N, JUMP (5)
A = A, P
N, F / OMEGA
N, GOTOR (MC[-1])
F * OMEGA
" THEN REPLACE
" X BY 1447
" WITH CORRECT SIGN
" * LOG (E) -> COMPUTATION OF 2↑X
" INTEGER PART OF X
" FRACTIONAL PART OF X = ZERO ?
" COMPUTE LOG (FRACTIONAL PART)
" BRING ARGUMENT IN RANGE -1/2 < X < 0
" AND COMPUTE POLYNOMIAL
" SQUARE, IF ARGUMENT WAS HALVED
" INTEGER PART
" CORRECTED FOR SUBTRACTION OF 2 * HALF
" OUTSIDE RANGE -2047, +2047 ?
" > 2047 ?

```

" LIBRARY ROUTINES NR. 8

```

      A = 2047
      LUA (15)
      A '*' = 32767
      S = 1
      F * A
      GOTOR (MC[-1])
C0:   - 13 27104
      - 0
      " +.999999999999999
C1:   - 13 32131
      + 351 25162
      " +.693147180524
C2:   - 13 93280
      + 324 98472
      " +.240226505508
C3:   - 14 60009
      + 42 24865
      " +.555040853706e-1
C4:   - 15 30010
      + 98 20595
      " +.961794504001e-2
C5:   - 16 27221
      + 234 73550
      " +.133256313600e-2
C6:   - 17 26494
      + 299 64123
      " +.152132608000e-3
C7:   - 17 35842
      + 348 22787
      " +.128376319999e-4
LOG E: - 12 98901
      + 374 31644
      " LOG (E) = +.144269504089e+1
OMEGA: + 670 76096
      + 1
      " 2 + 2047 = +.161585030357e+617

      'END' EXP
      " 58 LOCATIONS

COS:   'BEGIN' COSCOEF, SINCOEF, TWO OVER PI

      A = 1
      JUMP (1)
      " CCS (X) = SIN (X + PI/2)
SIN:   A = 0
      F * TWO OVER PI
      " REPLACE X BY X + 2/PI
      MC = F, P
      " X INTEGER ?
      U, S = F, E
      " ELSE ROUND X TO INTEGER:
      N, F + SCHOLTEN
      " N = X
      N, F - SCHOLTEN, P
      " IN CASE OF COS: N = N + 1
      A + G, E
      " TAKE CARE OF OVERFLOW IN A
      N, A '*' 3, P
      " N EVEN ?
      RCA (1), E
      " THEN USE POLYNOMIAL FOR SIN
      Y, S = :SINCOEF
      " ELSE USE POLYNOMIAL FOR COS, WITH
      N, S = :COSCOEF
      " Y = X - N AS ARGUMENT
      F - MC[-2]
      " POLYNOMIAL FOR SIN STARTS WITH Y TERM
      Y, MC = - F
      F * F
      " EVALUATE
      M[B] = F
      F * MS[10]
      " POLYNOMIAL
      F + MS[8]
      F * M[B]
      F + MS[6]
      F * M[B]
      F + MS[4]
      F * M[B]
      F + MS[2]
      F * M[B]
      F + MS
      " IN Y SQUARE

```

" LIBRARY ROUTINES NR. 9

```

Y, F * MC[-2]
LCA (1)
Y, A + 1
A '2', Z
N, F = - F
GOTOR (MC[-1])
COSCOEF:
+ 0
+ 1
+ 12 42657
- 415 22043
- 13 08641
+ 40 67027
+ 14 96389
- 312 98891
- 16 01776
+ 173 93881
+ 12 12415
- 17 08115
- 12 97852
SINCOEF:
+ 646 60001
+ 13 32904
- 319 28607
- 14 31346
+ 316 68041
+ 14 08717
- 552 53273
- 16 98552
+ 75 84785
+ 11 79647
- 1 22197
TWO OVER PI:
- 13 33057
+ 253 90670
'END' COS

" MULTIPLY BY Y IN THE SIN CASE
" NO EXTRA MINUS SIGN ?
" ELSE REVERSE SIGN
" C0 = 1
" C2 = -.12337 00550 12 *+1
" C4 = +.25366 95072 72 *+0
" C6 = -.20863 46891 37 *-1
" C8 = +.91916 54179 15 *-3
" C10 = -.24856 34468 03 *-4
" C1 = +.15707 96326 79 *+1
" C3 = -.64596 40974 93 *+0
" C5 = +.79692 62611 83 *-1
" C7 = -.46817 52592 89 *-2
" C9 = +.16042 92697 34 *-3
" C11 = -.35564 00770 32 *-5
" 2/PI = +.63661 97723 67 *+0
" 59 LOCATIONS

ARCTAN: 'BEGIN' TOGETHER, C1, C2, C3, C4, C5, C6, TG15, TG30,
PI OVER 6

F + 0
MC = G, P
N, F = - F
MC = F
S = 3
F = 1, P
Y, F = 1
Y, F / MC[-2]
Y, MC = F
N, F = M[B-2]
N, S = 2
F = TG15, E
Y, S = 1
U, A = 1, E
F = M[B-2]
N, GOTO (:TOGETHER)
F * TG30
F + 1
MC = F
F = M[B-4]

" PRESERVE ORIGINAL SIGN OF X
" AND REPLACE X BY ABSX = ABS (X)
" PRESERVE ABSX
" AND START ANALYSIS:
" ABSX > 1 ?
" THEN REPLACE ABSX BY 1/ABSX
" ELSE RESTORE ABSX IN F
" AND CHANGE INDICATION APPROPRIATELY
" ABS (X) > 1 = ABSX > TG (PI/12) ?
" THEN CHANGE INDICATION APPROPRIATELY
" ABSX > TG (PI/12) ?
" THEN
" REPLACE
" ABSX
" BY

```

" LIBRARY ROUTINES NR. 10

```

      F = TG30                                " (ABSX = TG (PI/6)) /
      F / MC[-2]                              " (1 + ABSX * TG (PI/6))
      M[B-2] = F
TOGETHER:  MC = S                                " PRESERVE INDICATION
      A = :C6
      SUBC (:POL IN FF)                        " COMPUTE POLYNOMIAL IN F SQUARE
      F * M[B+1]                              " * ABSX SQUARE
      F + 1
      F * MC[-3]                              " * ABSX
      S = MC[1]                                " INDICATION
      U, S = 1, P                              " > 1 ?
      Y, MC = - F
      N, MC = F
      G = S, Z                                " INDICATION = 0 ?
      N, F * PI OVER 6
      F + MC[-2]
      S = MC[-1], P                          " ORIGINAL SIGN OF X > 0 ?
      N, F = - F
      GOTOR (MC[-1])
C1:        + 13 65333
           + 223 69812 " -.333333333246
C2:        - 13 69702
           + 402 22387 " +.199999980477
C3:        + 13 99661
           - 119 33742 " -.142855496622
C4:        - 14 27237
           + 571 72235 " +.111044707738
C5:        + 14 03157
           - 595 65289 " -.895216002193e-1
C6:        - 14 58249
           + 433 90644 " +.622201788749e-1
TG15:      - 13 08524
           + 26 69885 " TG (PI/12) = +.267949192430
TG30:      - 13 05990
           + 438 49281 " TG (PI/6) = 1/SQRT(3) = +.577350269190
PI OVER 6: - 13 34909
           + 431 06668 " PI/6 = +.523598775599

      'END' ARCTAN " 57 LOCATIONS

```

```

TO DRUM:      S = D18
              JUMP (1)
FROM DRUM:    S = 0
              A = DREF[1], P      " ) GARANTY COMPLETION OF
N, A = MA      " ) PREVIOUS TRANSPORT
              DREF[2] = S        " INDICATOR DIRECTION OF TRANSPORT
              A = :MC
              SUB (:CEN)         " CONSTRUCT APIC FOR FIRST PARAMETER
              SUBC (:CIV)        " EVALUATE SECOND PARAMETER
              S = M[B-3]         " APIC[1] OF FIRST PARAMETER
              RUS (20)           " ISOLATE CODE
U, S = 7, P      " ) NOT THE CODE OF
U, S = 11, E     " ) AN ARRAY IDENTIFIER ?
Y, A = 520
Y, SUBC (:ERRORM) " REPORT ERROR
              S = M[B-2]         " APIC[2] OF ARRAY IDENTIFIER
              DREF = S           " CCNTAINS ADDRESS OF REF VARIABLE
              G = MS             " ADDRESS OF AR[0]
              MS = - G           " PREVENT ANY REFERENCE TO ARRAY
              S = MG[-1]         " AR[-1] OF ARRAY
              S '15' 15         " CCNTAINS THE ARRAY TYPE
U, S = 6, Z      " BOOLEAN ARRAY ?
              RUS (2)           " ISOLATE DELTA[0]
              S + MG            " CCNSTRUCT ADDRESS OF ELMNT AFTER THE LAST
              S = MG[-3]         " CCNSTRUCT ADDRESS OF FIRST ELEMENT
Y, A = 0         " ) FOR BOOLEAN ARRAYS
Y, DIVAS (27)    " ) 27 ELEMENTS/WORD
              DREF[2] + S       " FIRST CORE ADDRESS + DIRECTION
              S = MG[-3]         " AR[-3] CONTAINS DELTA[N]
N, JUMP (3)      " ) FOR BOOLEAN ARRAYS
              A = 0             " )
              DIVAS (27), Z     " ) 27 ELEMENTS/WORD
N, S + 1         " NUMBER OF WORDS
              DREF[4] = S       " DRUM ADDRESS
              A = MC[-1]        " ADD BASE
              A + DMP BEG       " FIRST DRUM WORD
              DREF[3] = A       " OUTSIDE LOWER BOUND ?
U, DMP BEG = A, P " OR OUTSIDE UPPER BOUND ?
              A + :MS
N, A = DMP END, P
Y, A = 521
Y, SUBC (:ERRORM) " REPORT ERROR
              B = 3            " REMOVE APIC FROM STACK
              '660 060 000'    " PREVENT INTERRUPTS
              DREF[1] = - G     " ALLOW WRONG ADDRESS INTERRUPTS
              ACTIVE[6] = B     " NOTE ACTIVITY
              S = 64
              GOTO (:ATTENTION) " ASK FOR TRANSPORT
                              " 48 INSTRUCTIONS

TIME:         G = M[64 + (4*39)]
              F / 100
              GOTOR (MC[-1])

```

```

MATMAT:      SUBC (:PLIST22)
BKJ:         SUB1 (:AIK)      " ADDRESS A[I,L] AND A[I,U]
              A = M[B-5]      " B
              S = M[B-12]     " L
              G = M[B-9]      " J
              SUB (:IND2)     " ADDRESS B[L,J]
              S = M[B-12]     " U
              G = M[B-10]     " J
MATMAT8:     SUB (:IND2)      " ADDRESS B[U,J]
MATMAT9:     SUB (:INPR)      " INNER PRODUCT
              B = 16
              GOTOR (MC[-1])  " 12 INSTRUCTIONS

TAMMAT:      SUBC (:PLIST22)
              SUB1 (:AKI)     " ADDRESS A[L,I] AND A[U,I]
              GOTO (:BKJ)     " 3 INSTRUCTIONS

MATMAT:      SUBC (:PLIST22)
BJK:         SUB1 (:AIK)      " ADDRESS A[I,L] AND A[I,U]
              A = M[B-5]      " B
              S = M[B-9]      " J
              G = M[B-12]     " L
              SUB (:IND2)     " ADDRESS B[J,L]
              S = M[B-10]     " J
              G = M[B-12]     " U
              GOTO (:MATMAT8)  " 9 INSTRUCTIONS

MATVEC:      SUBC (:PLIST21)
BK:          SUB1 (:AIK)      " ADDRESS A[I,L] AND A[I,U]
              A = M[B-5]      " B
              S = M[B-12]     " L
              SUB (:IND1)     " ADDRESS B[L]
              S = M[B-12]     " U
              SUB (:IND1)     " ADDRESS B[U]
              GOTO (:MATMAT9)  " 8 INSTRUCTIONS

TAMVEC:      SUBC (:PLIST21)
              SUB1 (:AKI)     " ADDRESS A[L,I] AND A[U,I]
              GOTO (:BK)      " 3 INSTRUCTIONS

```

VECVEC:

SUBC (:PLIST11)

A = M[B-6]

S = M[B-10]

SUB (:IND1)

S = M[B-10]

SUB (:IND1)

A = M[B-5]

S = M[B-12]

S + M[B-10]

SUB (:IND1)

S = M[B-12]

S + M[B-11]

SUB (:IND1)

GOTO (:MATMAT9)

" A

" L

" ADDRESS A[L]

" U

" ADDRESS A[U]

" B

" L

" SHIFT

" ADDRESS B[L+SHIFT]

" U

" SHIFT

" ADDRESS B[L+SHIFT]

" 14 INSTRUCTIONS

PLIST22:

SUBC (:PLIST2)

S = 2

F = 2

SUB (:TEST DIM)

G = M[B-9]

G = M[B-10]

COUNT = G

F + 1, P

Y, GOTOR (M[B-12])

F = 0

B = 12

GOTOR (MC[-1])

" DIMENSION OF B

" DIMENSION OF A

" U

" L

" U - L + 1 > 0 ?

" SUM NOT EMPTY

" EMPTY SUM

" RETURN OVER PROGRAM LINK

" 12 INSTRUCTIONS

PLIST21:

SUBC (:PLIST1)

S = 1

GOTO (:PLIST22)

" DIMENSION OF B

" 3 INSTRUCTIONS

PLIST11:

SUBC (:PLIST1)

S = 1

F = 1

GOTO (:PLIST223)

" DIMENSION OF B

" DIMENSION OF A

" 4 INSTRUCTIONS

```

PLIST2:      A = :MC[-2]
              SUBC (:CIV)
              SUBC (:CIV)
              SUBC (:CIV)
              SUBC (:CIV)
PLIST25:     SUB (:CEN)
              SUB (:CEN)
              S = M[B-5]
              RUS (20)
              S = 9, P
              M[B-5] = S
              S + 2, E
N, DOS (M[B-6])
N, S = MA
N, M[B-6] = A
              S = M[B-2]
              RUS (20)
N, S = 9, P
              M[B-2] = S
N, S + 2, E
N, DOS (M[B-3])
N, S = MA
N, M[B-3] = A
N, GOTOR (M[B-11])
              A = 522
              SUBC (:ERRORM)

PLIST1:      A = :MC[-2]
              SUBC (:CIV)
              SUBC (:CIV)
              SUBC (:CIV)
              B + 1
              GOTO (:PLIST25)

TEST DIM:    A = M[B-6]
              G = MA[-2], Z
              A = M[B-3]
Y, S = MA[-2], Z
Y, GOTOR (LINK)
              A = 523
              SUBC (:ERRORM)

IND1:        U, S = MA[-5], P
              S = MA[-4], E
Y, SUBC (:ERRORTABLE[2])
U, S = MA[-6], P
N, LUS (1)
              S + MA
              MC = S
              GOTOR (LINK)

```

" L
" U
" I
" J
" A
" B
" APIC1 OF A

" REAL = -1, INTEGER = -0
" NOT REAL OR INTEGER ARRAY ?
" EXECUTE APIC0 OF A
" WAIT FOR COMPLETION OF DRUM TRANSPORT
" ADDRESS OF ARO OF A
" APIC1 OF B

" REAL = -1, INTEGER = -0
" NOT REAL OR INTEGER ARRAY ?
" EXECUTE APIC0 OF B
" WAIT FOR COMPLETION OF DRUM TRANSPORT
" ADDRESS OF ARO OF B
" EXIT PLIST2
" REPORT ERROR
" 26 INSTRUCTIONS

" L
" U
" I OR SHIFT

" 6 INSTRUCTIONS

" A
" DIMENSION OF A CORRECT ?
" B
" DIMENSION OF B CORRECT ?
" EXIT TEST DIM
" REPORT ERROR
" 7 INSTRUCTIONS

" INDEX OUTSIDE BOUNDS ?

" INTEGER ARRAY ?
" FACTOR 2 FOR REAL ARRAY
" + ADDRESS OF LAST ELEMENT

" 8 INSTRUCTIONS

```

IND2:      G = MA[-5], P
           G + MA[-5]
           G = MA[-4], E      " SECOND INDEX OUTSIDE BOUNDS ?
Y, SUBC (:ERRORTABLE[2])    " (I1 - U1) * DELTA1
           G * MA[-6]
U, S = MA[-8], P
           S = MA[-7], E      " FIRST INDEX OUTSIDE BOUNDS ?
Y, SUBC (:ERRORTABLE[3])    " INTEGER ARRAY ?
U, S = MA[-9], P            " FACTOR 2 FOR REAL ARRAY
N, LUS (1)                  " ADDRESS OF LAST ELEMENT
           G + MA
           S + :MG
           MC = S
           GOTOR (LINK)      " 14 INSTRUCTIONS

AIK:      A = M[B-6]         " A
           S = M[B-8]         " I
           G = M[B-10]        " L
           SUB (:IND2)        " ADDRESS A[I,L]
           S = M[B-9]         " I
           G = M[B-10]        " U
           SUB (:IND2)        " ADDRESS A[I,U]
           GOTOR (LINK[1])    " 8 INSTRUCTIONS

AKI:      A = M[B-6]         " A
           S = M[B-10]        " L
           G = M[B-8]         " I
           SUB (:IND2)        " ADDRESS A[L,I]
           S = M[B-10]        " U
           G = M[B-9]         " I
           SUB (:IND2)        " ADDRESS A[U,I]
           GOTOR (LINK[1])    " 8 INSTRUCTIONS

```

```

INPR:      G = M[B-3]
           G = M[B-4]
           G / COUNT
           M[B] = G
           G = M[B-1]
           G = M[B-2]
           G / COUNT
           M[B+1] = G
           A = M[B-4]
           S = M[B-2]
           F = 0
U, A = M[B-9], Z
Y, GOTO (:A1)
U, A = M[B-6], Z
Y, GOTO (:R1)
RR:        M[B+2] = F
           F = MA
           F * MS
           F + M[B+2]
           A + M[B]
           S + M[B+1]
           REPE (:RR)
           GOTOR (LINK)
R1:        M[B+2] = F
           F = MA
           G * MS
           F + M[B+2]
           A + M[B]
           S + M[B+1]
           REPE (:R1)
           GOTOR (LINK)
A1:        U, A = M[B-6], Z
Y, GOTO (:11)
IR:        M[B+2] = F
           G = MA
           F * MS
           F + M[B+2]
           A + M[B]
           S + M[B+1]
           REPE (:IR)
           GOTOR (LINK)
11:        M[B+2] = F
           G = MA
           G * MS
           F + M[B+2]
           A + M[B]
           S + M[B+1]
           REPE (:11)
           GOTOR (LINK)

" STEP CF A
" STEP CF B
" FIRST ELEMENT OF A
" FIRST ELEMENT OF B
" A AN INTEGER ARRAY ?
" B AN INTEGER ARRAY ?
" SAVE SUM
" ELEMENT * ELEMENT
" + SUM
" INCREMENT ARRAY A
" INCREMENT ARRAY B
" COUNT DOWN
" SAVE SUM
" ELEMENT * ELEMENT
" + SUM
" INCREMENT ARRAY A
" INCREMENT ARRAY B
" COUNT DOWN
" B AN INTEGER ARRAY ?
" SAVE SUM
" ELEMENT * ELEMENT
" + SUM
" INCREMENT ARRAY A
" INCREMENT ARRAY B
" COUNT DOWN
" SAVE SUM
" ELEMENT * ELEMENT
" + SUM
" INCREMENT ARRAY A
" INCREMENT ARRAY B
" COUNT DOWN
" 49 INSTRUCTIONS

```

STRING SYMBOL: 'BEGIN' CALL OF TCST

A = :MC	" K
SUBC (:CIV)	" TEXT
SUB (:CEN)	" APIC1 OF TEXT
S = MC[-2]	
RUS (20)	
S = 23, Z	" UNASSIGNABLE STRING EXPRESSION ?
S = MC[-2]	" APIC0 OF TEXT
Y, S '*' 32767	" ISOLATE BEGIN ADDRESS OF ISR
Y, A = MS[1]	" SECOND INSTRUCTION OF ISR
Y, A = CALL OF TCST, Z	" IS IT THE MACRO TCST ?
N, A = 611	
N, SUBC (:ERRORM)	" REPORT ERROR
A = MS[2]	" THIRD INSTRUCTION OF ISR
A '*' 32767	" ISOLATE END ADDRESS OF ISR
G = S	" SAVE BEGIN ADDRESS
M[B-1] = A	" SAVE END ADDRESS
S = MC[-2], Z	" K = 0 ?
Y, S = 0, P	" THEN K = + 0
A = 0, E	" K ≥ 0 ?
Y, DIVAS (3)	
S = :MG[4]	" ADDRESS OF WORD[1]
U, S = :MG[3], P	
U, S = MC, E	" OUTSIDE TEXT ?
Y, S = - 0	
N, S = MS[-1]	" WCRD
U, A = 2, Z	") SHIFT
N, RUS (8)	") TC
N, A = 1, Z	") RIGHT
N, RUS (8)	") POSITION
S '*' 255	" CLEAN AWAY EXTRANEIOUS BITS
G = S	
GOTOR (MC[-1])	" EXIT
CALL OF TCST: SUBC (:TCST)	" 33 INSTRUCTIONS
DUMP:	

'END'

"DUMP SECTION NR 1.

DUMP[0]:

'BEGIN' DRBUNOF8, DRROOM8, QUEU8, CRBUF1, CRBUF2, CRBUNOF8,
CRBUNOE8, DUPOS, BUFND, STRWRD, NEXT STRWRD, EMPTYWORD,
NINE IN WORD, STOP DUMPING, LAST BITS, GET BUF,
TO DRUM, CRTODR

"ASSUMED TO BE DECLARED GLOBALLY ARE: INIT BITSTRM, BTSTRM9,
"BTSTRM18, BTSTRM27, FINISH BITSTRM, END DUMP, PROCESS8

"ASSUMED TO BE DECLARED AND DEFINED GLOBALLY ARE:
"CMODE, END OF DRUMLIBRARY, DRBEG8, DRLIBREND, DRENDS,
"ACTIVE, RECBULEN, ATTENTION, ERRORM, D18.

BTSTRM9:	'SKIP'	1	
DRBUNOF8:	'SKIP'	1	
DRROOM8:	'SKIP'	1	
QUEU8:	'SKIP'	1	
CRBUNOF8:	'SKIP'	1	
CRBUNOE8:	'SKIP'	1	
DUPCS:	'SKIP'	1	" DUMP POSITION
BUFND:	'SKIP'	1	
STRWRD:	'SKIP'	1	
	:CRBUF2		
CRBUF1:	'SKIP':	RECBULEN	
	:CRBUF1		
CRBUF2:	'SKIP':	RECBULEN	

" DUMP SECTION NR 2.

DUMPING OFF:

S = : STOP DUMPING
 BTSTRM9 = S
 GOTOR(MC[-1])

'BEGIN' LOOP

```

INIT BITSTRM:      S = CMODE
                   U, S -5 , P           " NO TRANSPORT WANTED?

                   Y, GOTO(:DUMPING OFF)
                   U, S -1 , Z           " YES = NORMAL PROGRAM
                                           " NO = INSERT IN LIBRARY
                                           "
                                           " )SELECT FIRST DRUMPAGE
                                           "
                   N, A = END OF DRUMLIBRARY
                   Y, A = DRBEG8
                   DRBUNOF8 = A
                   N, S = DRLIBREND
                   Y, S = DREND8
                   S = A
                   DRROOM8 = S           " NUMBER OF WORDS FREE ON DRUM
                   A = 0                 " )NO QUEUE OF BUFFERS WAITING
                   QUEUE8 = A           " )FOR TRANSPORT
                   S = : EMPTY WORD
                   BTSTRM9 = S           " PSEUDO-SHIFT
                   ACTIVE(8) = -B        " NO ACTIVE PROCESS
                   S = : CRBUF1
                   CRBUNOF8 = S
                   CRBUNOE8 = S
                                           " FIRST BUFFER TO FILL
                                           " FIRST BUFFER TO DRUM

LOOP:              A = MS[-1] , P
                   N, MS[-1] = -A
                   S = MS[-1]
                   U, S = CRBUNOE8 , Z
                   N, GOTO(:LOOP)
                                           " )ASSURE PRESENCE
                                           " )OF BUFFERS
                                           " SELECT NEXT BUFFER
                                           " CYCLE CLOSED

INITDUPOS:        DUPOS = S
                   S + : RECBULEN
                   BUFND = S
                                           " END OF BUFFER

STOP DUMPING:     GOTOR(MC[-1])
                   'END'      INIT BITSTRM

```

"DUMP SECTION NR 3,

"NEXT STRWORD, EMPTYWORD, NINE IN WORD FORM TOGETHER A LOGIC BLOCK, THAT
 "PLAYS AN ANALOGOUS AS PUHEP IN THE PUNCH SECTIONH THEIR SEQUENCE AS
 "WRITTEN IS OBLIGATE AS IS STOPDUMPING.

```

NEXT STRWORD:      A = : EMPTYWORD
                   S '+' STRWORD           " FILL S WITH BITS
                   BTSTRM9 = A             " CHANGE PSEUDO-SHIFT
                   A = 1
                   PLUSA(DUPOS)            " )ITS INTO BUFFER
                   MA[-1] = S              " )
                   U, A = BUFND , Z        "BUFFER FULL
                   N, GOTOR(MC[-1])
                   GOTO(:GET BUF)          " IF BUFFER FULL GET A NEW ONE

EMPTYWORD:         A = : NINE IN WORD
                   BTSTRM9 = A             " CHANGE PSEUDO SHIFT
                   LUS(17)                 " )OCCUPY FIRST
                   LCS(1)                  " )9 BITS IN STRINGWORD.
                   STRWORD = S             " )
                   GOTOR(MC[-1])

NINE IN WORD:      A = : NEXT STRWORD
                   BTSTRM9 = A             " PSEUDO SHIFT
                   LUS(9)
                   S '+' STRWORD
                   STRWORD = S
                   GOTOR(MC[-1])

BTSTRM18:          A = : EMPTY WORD
                   U, BTSTRM9 = A, P       " )PSEUDO SHIFT INDICATES
                   Y, GOTO(:NEXT STRWORD[1]) " )OPTIMIZABLE PROCESSING?
                   MC = S
                   RUS(9)
                   SUBC(BTSTRM9)

LASTBITS:          S = MC[-1]
                   S '*' 511
                   GOTO(BTSTRM9)

BTSTRM27:          MC = S                  " 27 = 9 + 18
                   RUS(9)
                   S '*' D18M1
                   SUBC(:BTSTRM18)
                   GOTO(:LAST BITS)

```

"DUMP SECTION NR 4,

```

GET BUF:      G = CRBUNOF8      " SELECT BUFFER
              SUBCD(:TO DRUM)   " ASK FOR TRANSPORT
              S = CRBUNOF8      " SELECT NEXT BUFFER
              A = MS[-1] , P    " PRESENT?
N, JUMP(-2)    " INITIALIZE DUPOS8 BUFND
              GOTO(:INIT DUPOS)

TO DRUM:      A = MG[-1]
              MG[-1] = -A      " BUFFER NOT PRESENT
              CRBUNOF8 = A     " SELECT BUFFER
              A = : RECBULEN   " SPACE ON DRUM?
MINA(DRROOM8) , P
N, JUMP(7)    " IF SO ADD ONE BUFFER TO
              A = 1           " QUEUE
              QUEU8 = A
              A = ACTIVE[8] , P
Y, GOTOR(MC[-1]) " POSTPONE TRANSPORT
              ACTIVE[8] = B   " NOTE ACTIVITY
              S = 256
              GOTO(:ATTENTION) " ASK FOR TRANSPORT
              S = : STOP DUMPING " )
              BTSTRM9 = S      " ) NO FURTHER TRANSPORT
              A = 495          " ) OF BITS TO DRUM
              GOTO(:ERRORM)    " )

CRTODR:
PROCESS8:     A = DRBUNOF8      " ) PLACE ON DRUM
              S = CRBUNOE8     " FIRST WORD IN CORE
              S + D18          " INDICATION CORE TO DRUM
              F = : RECBULEN   " NUMBER OF WORDS
              SUBC(:DRSTART)   " START TRANSPORT
              GOTOR(MC[-1])    " EXIT CR TO DR

"AFTER COMPLETION OF THE TRANSPORT, CONTROL IS GIVEN TO
"THE NEXT INSTRUCTIONS

              S = : RECBULEN   " FIRST FREE PLACE
              DRBUNOF8 + S     " SELECT BUFFER
              S = CRBUNOE8
              A = MS[-1]
              MS[-1] = -A      " BUFFER PRESENT AGAIN
              CRBUNOE8 = -A    " SELECT NEXT BUFFER
              S = 1
              QUEU8 = S , Z    " QUEUE SHORTER, NOW EMPTY?
Y, ACTIVE[8] = -B             " NOTE INACTIVITY
Y, S = 256                    " CAUCEL ATTENTION
N, S = 0                       " CONTINUE ATTENTION
              GOTO(:ATTENTION)

```

" DUMP SECTION NR 5

FORCE LAST BITS:

```

S = 0
A = BTSTRM9
U, A = :NINE IN WORD, Z
Y, GOTO(:BTSTRM18)
A = :NEXT STRWRD, Z
Y, GOTO(BTSTRM9)
GOTOR(MC[-1])

```

ASK DUMP END:

```

G = QUEU8
F = :RECBULEN
S = DUPOS
S = CRBUNOF8
S = DRBUNOF8
S = G
GOTOR(MC[-1])

```

" PLACE FREE ON DRUM
" FOR NEXT PROC

FINISH BITSTRM:

```

A = DUPOS
A = CRBUNOF8, Z
N, GOTO(:GET BUF)
GOTOR(MC[-1])

```

" PART OF BUFFER TO DRUM

END DUMP:

```

A = ACTIVE[8], P
Y, JUMP(-2)
GOTOR(MC[-1])

```

" WAIT

COMPVAR:

'END'

" DUMP SECTION

BEG TRAFO TAB[0]:

```

'SKIP' 1

```

" PUT IN CORE AFTER INSTR
" LST AND TRANSF

```

:PUTEXT L
:STOP L
:XEEN L
:HAND L
:PUHEP L
:REHEP L
:RUNOUT L
:PUSPACE L
:PUNLCR L
:PUNCH L

```

```

:FLOP L
:ABSFIXP L
:FIXP L
:READ L
:ENTIER L
:EXP L
:LN L
:ARCTAN L
:COS L
:SIN L

```

```

:SQRT L
:SIGN L
:ABS L
:PRSYM L
:PUSYM L

```

:RESYM L
:LINE NUMBER L
:NEW PAGE L
:CARRIAGE L
:PRINTTEXT L

:SPACE L
:TAB L
:NLCR L
:PRINT L
:FLOT L
:ABSFIXT L
:FIXT L
:FROM DRUM L
:TO DRUM L
:TIME L

:VV L
:TV L
:MV L
:MT L
:TM L
:MM L
:SS L

" 48 ENTRIES

'END' LIBRARY
'END' RUN SYST
'END' RUN VAR

" CATALOGUE NR 1

END OF PERMANENT CATALOGUE[1]:

+	0	
+	0	
+ 51 904 512		" 99 * D19
-	1	
+	0	
+ 34 078 720		" 65 * D19
-	1	
('232 000 000' + 3)		
(8 912 896 + 47)		" 17 * D19 + NUMBER
- ' 0 62 47 64 61'		
- ' 0 63 54 70 76'		
- ' 0 70 71 67 56'		" STRING SYMBOL
+	0	
+	3	
+ 56 623 104		" 108 * D19
-	1	
+	3	
+ 56 623 104		" 108 * D19
-	1	
+	0	
+ 34 078 720		" 65 * D19
-	1	
+	0	
+ 34 078 720		" 65 * D19
-	1	
+	0	
+ 34 078 720		" 65 * D19
-	1	
+	0	
+ 34 078 720		" 65 * D19
-	1	
('232 000 000' + 7)		
(8 388 608 + 46)		" 16 * D19 + NUMBER
- ' 0 00 00 13 36'		
- ' 0 27 13 36 27'		" MATMAT
+	0	
+	3	
+ 56 623 104		" 108 * D19
-	1	
+	3	
+ 56 623 104		" 108 * D19
-	1	
+	0	
+ 34 078 720		" 65 * D19
-	1	
+	0	
+ 34 078 720		" 65 * D19
-	1	
+	0	
+ 34 078 720		" 65 * D19
-	1	
+	0	
+ 34 078 720		" 65 * D19
-	1	
('232 000 000' + 7)		
(8 388 608 + 45)		" 16 * D19 + NUMBER

100470 -

1

150

-' 0 00 00 13 36'

-' 0 36 13 27 27'

" TAMMAT

" CATALOGUE NR 2

+	0	
+	3	
+ 56 623 104		" 108 * D19
-	1	
+	3	
+ 56 623 104		" 108 * D19
-	1	
+	0	
+ 34 078 720		" 65 * D19
-	1	
+	0	
+ 34 078 720		" 65 * D19
-	1	
+	0	
+ 34 078 720		" 65 * D19
-	1	
+	0	
+ 34 078 720		" 65 * D19
-	1	
('232 000 000' + 7)		
(8 388 608 + 44)		" 16 * D19 + NUMBER
- ' 0 00 00 13 27 '		
- ' 0 27 13 36 36 '		" MATTAN
+	0	
+	2	
+ 56 623 104		" 108 * D19
-	1	
+	3	
+ 56 623 104		" 108 * D19
-	1	
+	0	
+ 34 078 720		" 65 * D19
-	1	
+	0	
+ 34 078 720		" 65 * D19
-	1	
+	0	
+ 34 078 720		" 65 * D19
-	1	
('232 000 000' + 6)		
(8 388 608 + 43)		" 16 * D19 + NUMBER
- ' 0 00 00 17 15 '		
- ' 0 27 13 36 40 '		" MATVEC

" CATALOGUE NR 3

+	0	
+	2	
+ 56 623 104		" 108 * D19
-	1	
+	3	
+ 56 623 104		" 108 * D19
-	1	
+	0	
+ 34 078 720		" 65 * D19
-	1	
+	0	
+ 34 078 720		" 65 * D19
-	1	
+	0	
+ 34 078 720		" 65 * D19
-	1	
(' 232 000 000' + 6)		
(8 388 608 + 42)		" 16 * D19 + NUMBER
- ' 0 00 00 17 15'		
- ' 0 36 13 27 40'		" TAMVEC
+	0	
+	2	
+ 56 623 104		" 108 * D19
-	1	
+	2	
+ 56 623 104		" 108 * D19
-	1	
+	0	
+ 34 078 720		" 65 * D19
-	1	
+	0	
+ 34 078 720		" 65 * D19
-	1	
+	0	
+ 34 078 720		" 65 * D19
-	1	
(' 232 000 000' + 6)		
(8 388 608 + 41)		" 16 * D19 + NUMBER
- ' 0 00 00 17 15'		
- ' 0 40 17 15 40'		" VECVEC

" CATALOGUR NR 4

+	0	
(' 232 000 000' + 1)		" 16 * D19 + NUMBER
(8 388 608 + 40)		" TIME
- '036 23 27 17'		

+	0	
+	0	
+ 34 078 720		" 65 * D19
-	1	
+	0	
+ 40 370 176		" 77 * D19
-	1	
('222 000 000' + 3)		
(12 582 912 + 39)		" 24 * D19 + NUMBER
- ' 0 00 00 72 62'		
- ' 0 71 64 51 67'		" TO DRUM

+	0	
+	0	
+ 34 078 720		" 65 * D19
-	1	
+	0	
+ 40 370 176		" 77 * D19
-	1	
(' 222 000 000' + 3)		
(12 582 912 + 38)		" 24 * D19 + NUMBER
- ' 0 51 67 72 62'		
- ' 0 53 67 64 62'		" FROM DRUM

" CATALOGUE NR 5

+	0	
+	0	
+ 33 554 432	1	" 64 * D19
-	0	
+	0	
+ 34 078 720	1	" 65 * D19
-	0	
+	0	
+ 34 078 720	1	" 65 * D19
-	1	
('222 000 000' + 4)		" 010 010 010
(12 582 912 + 37)		" 24 * D19 + NUMBER
- '0 53 56 75 71'		" FIXT
+	0	
+	0	
+ 33 554 432	1	" 64 * D19
-	0	
+	0	
+ 34 078 720	1	" 65 * D19
-	0	
+	0	
+ 34 078 720	1	" 65 * D19
-	1	
('222 000 000' + 4)		" 24 * D19 NUMBER
(12 582 912 + 36)		
- '0 00 56 75 71'		" ABSFIXT
- '0 46 47 70 53'		
+	0	
+	0	
+ 33 554 432	1	" 64 * D19
-	0	
+	0	
+ 34 078 720	1	" 65 * D19
-	0	
+	0	
+ 34 078 720	1	" 65 * D19
-	1	
('222 000 000' + 4)		" 24 * D19 + NUMBER
(12 582 912 + 35)		" FLOT
- '0 53 61 64 71'		

" CATALOGUE NR.6

+	0	
+	0	
+	33 554 432	" 64 * D19
-	1	
(	'222 000 000' + 2)	
(	12 582 912 + 34)	" 24 * D19 + NUMBER
-	'0 00 00 00 71'	
-	'0 65 67 56 63'	" PRINT
+	0	
+	0	
+	33 554 432	" 64 * D19
-	1	
(	'222 000 000' + 2)	
(	12 582 912 + 34)	" 24 * D19 + NUMBER
-	'0 00 00 00 36'	
-	'0 32 34 23 30'	" PRINT
+	0	
(	'222 000 000' + 1)	
(	12 582 912 + 33)	" 24 * D19 + NUMBER
-	'0 63 61 50 67'	" NLCR
+	0	
(	'222 000 000' + 1)	
(	12 582 912 + 32)	" 24 * D19 + NUMBER
-	'0 00 71 46 47'	" TAB
+	0	
+	0	
+	34 078 720	" 65 * D19
-	1	
(	'222 000 000' + 2)	
(	12 582 912 + 31)	" 24 * D19 + NUMBER
-	'0 00 00 00 52'	
-	'0 70 65 46 50'	" SPACE
+	0	
+	0	
+	51 904 512	" 99 * D19
-	1	
(	'222 000 000' + 2)	
(	12 582 912 + 30)	" 24 * D19 + NUMBER
-	'0 00 00 00 71'	
-	'0 71 71 52 75'	
-	'0 65 67 56 63'	" PRINTTEXT
+	0	
+	0	
+	34 078 720	" 65 * D19
-	1	
(	'222 000 000' + 2)	
(	12 582 912 + 29)	" 24 * D19 + NUMBER
-	'0 56 46 54 52'	
-	'0 50 46 67 67'	" CARRIAGE

" CATALOGUE NR.7

+	0	
	(222 000 000' + 1)	" 24 * D19 + NUMBER
	(582 912 + 28)	
	0 00 46 54 52'	" NEW PAGE
	1 63 52 74 65'	
	0	
	(232 000 000' + 1)	" 010 011 010'
	(8 912 896 + 27)	" 17 * D19 + NUMBER
-	'0 00 00 52 67'	
-	'0 63 72 62 47'	" LINE NUMBER
-	'0 61 56 63 52'	
	0	
	(232 000 000' + 1)	" 17 * D19 + NUMBER
	(8 912 896 + 26)	
-	'0 00 00 00 62'	" RESYM
-	'0 67 52 70 76'	
	0	
	0	
+	34 078 720	" 65 * D19
-	1	
	(222 000 000' + 2)	" 24 * D19 + NUMBER
	(12 582 912 + 25)	
-	'0 00 00 00 62'	" PUSYM
-	'0 65 72 70 76'	
	0	
	0	
+	34 078 720	" 65 * D19
-	1	
	(222 000 000' + 2)	" 24 * D19 + NUMBER
	(12 582 912 + 24)	
-	'0 00 00 00 62'	" PRSYM
-	'0 65 67 70 76'	
	76	
	(272 000 000' + 2)	" 010 111 010
	(8 388 608 + 23)	" 16 * D19 + NUMBER
-	'0 00 13 14 35'	" ABS
	77	
	(272 000 000' + 2)	" 17 * D19 + NUMBER
	(8 912 896 + 22)	" SIGN
-	'0 35 23 21 30'	
	79	
	(272 000 000' + 2)	" 16 * D19 + NUMBER
	(8 388 608 + 21)	" SQRT
-	'0 35 33 34 36'	
	129	
	(272 000 000' + 2)	" 16 * D19
	(8 388 608 + 20)	" SIN
-	'0 00 35 23 30'	

" CATALOGUE NR. 8

+	130	
('272 000 000' + 2)		" 16 * D19 + NUMBER
(8 388 608 + 19)		" COS
- '0 00 15 31 35'		
+	131	
('272 000 000' + 2)		" 16 * D19 + NUMBER
(8 388 608 + 18)		" ARCTAN
- '0 00 00 13 30'		
- '0 13 34 15 36'		
+	81	
('272 000 000' + 2)		" 16 * D19 + NUMBER
(8 388 608 + 17)		" LN
- '0 00 00 26 30'		
+	80	
('272 000 000' + 2)		" 16 * D19 + NUMBER
(8 388 608 + 16)		" EXP
- '0 00 17 42 32'		
+	78	
('272 000 000' + 2)		" 17 * D19 + NUMBER
(8 912 896 + 15)		" ENTIER
- '0 00 00 17 34'		
- '0 17 30 36 23'		
+	132	
('272 000 000' + 1)		" 16 * D19 + NUMBER
(8 388 608 + 14)		" READ
- '0 34 17 13 16'		
+	132	
('272 000 000' + 1)		" 16 * D19 + NUMBER
(8 388 608 + 14)		" READ
- '0 67 52 46 51'		
+	133	
('262 000 000' + 4)		" 010 110 010
(12 582 912 + 13)		" 24 * D19 + NUMBER
- '0 53 56 75 65'		" FIXP
+	134	
('262 000 000' + 4)		" 24 * D19 + NUMBER
(12 582 912 + 12)		" ABSFIXP
- '0 00 56 75 65'		
- '0 46 47 70 53'		
+	135	
('262 000 000' + 4)		" 24 * D19 + NUMBER
(12 582 912 + 11)		" FLOP
- '0 53 61 64 65'		
+	136	
('262 000 000' + 2)		" 24 * D19 + NUMBER
(12 582 912 + 10)		" PUNCH
- '0 00 00 00 55'		
- '0 65 72 63 50'		

" CATALOGUE NR. 9

```

+          137
('262 000 000' + 1)
(12 582 912 + 9 )
- '0 00 00 50 67'
- '0 65 72 63 61'
" 24 * D19 + NUMBER
" PUNLCR

+          138
('262 000 000' + 2)
(12 582 912 + 8)
- '0 00 46 50 52'
- '0 65 72 70 65'
" 24 * D19 + NUMBER
" PUSPACE

+          139
('262 000 000' + 1)
(12 582 912 + 7)
- '0 00 00 72 71'
- '0 67 72 63 64'
" 24 * D19 + NUMBER
" RUNOUT

+          140
('272 000 000' + 1)
(8 912 896 + 6)
- '0 00 00 00 65'
- '0 67 52 55 52'
" 17 * D19 + NUMBER
" REHEP

+          141
('262 000 000' + 2)
(12 582 912 + 5)
- '0 00 00 00 65'
- '0 65 72 55 52'
" 24 * D19 + NUMBER
" PUHEP

+          142
('272 000 000' + 2)
(8 388 608 + 4)
- '0 55 46 63 51'
" 16 * D19 + NUMBER
" HAND

+          143
('272 000 000' + 2)
(8 912 896 + 3)
- '0 75 52 52 63'
" 17 * D19 + NUMBER
" XEEN

+          144
('262 000 000' + 1)
(12 582 912 + 2)
- '0 70 71 64 65'
" 24 * D19 + NUMBER
" STOP

+          0
+          0
+ 51 904 512
-          1
('222 000 000' + 2)
(12 582 912 + 1)
- '0 00 00 75 71'
- '0 65 72 71 52'
" 99 * D19
" 010 010 010
" 24 * D19 + NUMBER
" PUTEXT

```

" COMPILER VARIABLES

COMP VAR[0]:

'BEGIN' STOCK, STOCK1, QUOTE COUNTER, TEXT ARRAY POINTER, SHIFT,
 FIRST SHIFT, WORD1SAFE, LETTER LAST SYMBOL,
 DIGIT LAST SYMBOL, OWN TYPE, TYPE, CHARACTER, VAL CHAR,
 ARR DEC LS, TP DEC LS, ARR DEC MCR, REAL NUMBER, SMALL,
 LAST NLP, WORD COUNT, IN FORMAL LIST, IN AR DEC, INT LABELS,
 INT LAB, OLD BCP, DSP LVL, GLOBAL COUNT, ARR POINTER,
 DIMENSION, SUBCOUNT, FOR CNT, NXT BCP, STATE, STACK0, STACK1,
 STACK2, MCR, PAR, NBR, PARAMETER, NUMBER,
 ST CNT, MAX DPTH, MAX DI, MAX DL, MAX PRL, RET MD,
 RET LVL, ECNT, IFSTAT PRB, CNTLD VAR, L0, L1, L2, L3, L4, L5,
 CMPLTD, CMPL ST, SW IDF, NBR OF SW E, SW LST, IN SW DEC,
 ADRS OF CST, LNC, LAST LNC, CCDE BODY, POINTER, END OF TEXT ARRAY,
 NLP

STOCK:	'SKIP' 1	
STOCK1:	'SKIP' 1	")THIS ORDER OBLIGATE
QUOTE COUNTER:	'SKIP' 1	")
TEXT ARRAY POINTER:	'SKIP' 1	") THIS ORDER OBLIGATE
SHIFT:	'SKIP' 1	")
FIRST SHIFT:	'SKIP' 1	
WORD1SAFE:	'SKIP' 1	
LETTER LAST SYMBOL:	'SKIP' 1	
DIGIT LAST SYMBOL:	'SKIP' 1	
OWN TYPE:	'SKIP' 1	
TYPE:	'SKIP' 1	
CHARACTER:	'SKIP' 1	
VAL CHAR:	'SKIP' 1	
ARR DEC LS:	'SKIP' 1	
TP DEC LS:	'SKIP' 1	
ARR DEC MCR:	'SKIP' 1	
REAL NUMBER:	'SKIP' 1	
SMALL:	'SKIP' 1	
LAST NLP:	'SKIP' 1	
WORD COUNT:	'SKIP' 1	
IN FORMAL LIST:	'SKIP' 1	
IN AR DEC:	'SKIP' 1	
INT LABELS:	'SKIP' 1	
INT LAB:	'SKIP' 1	
OLD BCP:	'SKIP' 1	
DSP LVL:	'SKIP' 1	
GLOBAL COUNT:	'SKIP' 1	
ARR POINTER:	'SKIP' 1	
DIMENSION:	'SKIP' 1	
SUBCOUNT:	'SKIP' 1	
FOR CNT:	'SKIP' 1	
NXT BCP:	'SKIP' 1	
STATE:	'SKIP' 1	
STACK0:	'SKIP' 1	
STACK1:	'SKIP' 1	") THIS ORDER OBLIGATE
STACK2:	'SKIP' 1	")
MCR:	'SKIP' 1	
PAR:	'SKIP' 1	")THIS ORDER OBLIGATE
NBR:	'SKIP' 1	")

PARAMETER: 'SKIP' 1

") THIS ORDER OBLIGATE

" COMPILER VARIABLES NR. 2

NUMBER: 'SKIP' 1
 ST CNT: 'SKIP' 1
 MAX DPTH: 'SKIP' 1
 MAX DI: 'SKIP' 1
 MAX DL: 'SKIP' 1

")

MAX PRL: 'SKIP' 1
 RET MD: 'SKIP' 1
 RET LVL: 'SKIP' 1
 ECNT: 'SKIP' 1
 IFSTAT FRB: 'SKIP' 1

CNTLD VAR: 'SKIP' 1
 L0: 'SKIP' 1
 L1: 'SKIP' 1
 L2: 'SKIP' 1
 L3: 'SKIP' 1

L4: 'SKIP' 1
 L5: 'SKIP' 1
 CMPLTD: 'SKIP' 1
 CMPL ST: 'SKIP' 1
 SW IDF: 'SKIP' 1

NBR OF SW E: 'SKIP' 1
 SW LST: 'SKIP' 1
 IN SW DEC: 'SKIP' 1
 ADRS OF CST: 'SKIP' 1
 LNC: 'SKIP' 1

LAST LNC: 'SKIP' 1
 CODE BODY: 'SKIP' 1
 POINTER: 'SKIP' 1
 END OF TEXT ARRAY: 'SKIP' 1
 NLP: 'SKIP' 1

" 70 VARIABLES

" COMPILER CONSTANTS

BEGIN OF COMPILER:

'BEGIN' D18 MIN 1, D18, D19, D20, D21, D21 PLUS D20, D22,
 D22 PLUS 1, D24, D25, FRAME, LOCAL NUMBER, MASK, WORD DEL

D16 M1: '177 777'

D18 MIN 1: '777 777'
 D18: '001 000 000'
 D19: '002 000 000'
 D20: '004 000 000'
 D21: '010 000 000'
 D21 PLUS D20: '14 000 000'
 D22: '020 000 000'
 D22 PLUS 1: '020 000 001'
 D24: '100 000 000'
 D25: '200 000 000'
 FRAME: '000 177 400'

```

LOCAL NUMBER:      '222 000 001'
BASE0:             :PROGRAM3
                   :CMP TL3
                   :DEC LST3
BASE1:             :PROGRAM3CM8
                   :CMPTL3CM8
                   :DEC LST3CM8      " 22 LOCATIONS
BASE2:             :PROGRAM 3 CM2
                   :CMP TL 3
                   :DEC LST3

MASK:              (20 * '004 000 000') " RE
                   (21 * '004 000 000') " IN
                   (22 * '004 000 000') " BO
                   (23 * '004 000 000') " ST
                   (20 * '004 000 000') " AR
                   (31 * '004 000 000') " NONDES
                   (28 * '004 000 000') " DES
                   (31 * '004 000 000') " UN
                   (31 * '004 000 000') " ARBO
                   (29 * '004 000 000') " INTLAB

WORD DEL:          +296926             " IF
                   +477407             " THEN
                   +232160             " ELSE
                   +182120             " BEGIN
                   +232425             " END
                   +265297             " GOTO
                   +248914             " FOR
                   +462574             " STEP = STRING
                   +494548             " UNTIL
                   +526549             " WHILE
                   +216150             " DO
                   +199777             " COMMENT
                   +397418             " OWN
                   +444267             " REAL
                   +297964             " INTEGER
                   +625773             " BOOLEAN
                   +183405             " BOOLEAN
                   +167407             " ARRAY
                   +413168             " PROCEDURE
                   +462961             " SWITCH
                   +345458             " LABEL
                   +509299             " VALUE
                   +478708             " TRUE
                   +24/157             " FALSE

```

" GENERAL PURPOSE PROCEDURES NR. 40

```

'BEGIN' NXT SBL, TEST DP0, TEST POINTERS, NXT BSC SBL,
      IN SBL, UND SBL, OUT SBL, AR OP LS,
      INVERT, REL OP LS, BOOL OP LS, DECL LS, SPEC LS, OP LS,
      UNS INT, MULTIPLY, UNS NBR, RD IDF, RD IDF1, NXT PNR, LOOK UP,
      NAME IN LIBRARY, IN NM LST, NXT IDF, SKP IDF, SKP TP DEC,
      SKP VA LI, SKP SP LI, DISP LVL, TP OF DSP, LOC SPC, PRC LVL,
      USE CST, STATUS, IN CODE, ENTR BLK, EXIT BLK, LOC LAB,
      NONFRM LAB, CORSP BCP, SKIP STRING, SKIP RST OF STAT, ARUNVA,
      INIT POINTER, INIT, PRESCAN1, TRANSL, INSTR NBR,
      CLEAR, PAR PART, INSTR PRT, RUNVA, NEXT ENTRY

```

NXT SBL: 'BEGIN' SKIP0, SKIP1, SKIP2, ELSE0, ELSE1, END

```

      S = STOCK1, P      " SYMBOL IN STOCK ?
Y, STOCK1 = - S
N, SUBC (:NXT BSC SBL)
U, S = 97, Z      " NEXT BASIC SYMBOL = COMMENT ?
N, GOTO (:ELSE0)
  A = LAST SYMBOL
U, A = 91, Z      " LAST SYMBOL = SEMICOLON ?
N, A = 104, Z      " ~ LAST SYMBOL = BEGIN ?
N, GOTO (:ELSE0)
SKIP0: SUBC (:NXT BSC SBL)
U, S = 91, Z      " NEXT BASIC SYMBOL = SEMICOLON ?
N, GOTO (:SKIP0)
  GOTO (:NXT SBL)
ELSE0: A = LAST SYMBOL
      A = 105, Z      " LAST SYMBOL = END ?
N, GOTO (:ELSE1)
SKIP1: A = S      " LAST BASIC SYMBOL
      A = 105, Z      " = END ?
N, A = 91, Z      " = SEMICOLON ?
N, A = 5, Z      " = ELSE, ?
Y, GOTO (:END)      " THEN ACCEPT
  SUBC (:NXT BSC SBL)
  GOTO (:SKIP1)
ELSE1: U, S = 125, Z      " LAST BASIC SYMBOL = CLOSE ?
N, GOTO (:END)
  SUBC (:NXT BSC SBL)
U, S = 9, P      " NEXT BASIC SYMBOL
U, S = 63, E      " NOT A LETTER ?
Y, STOCK1 = S
Y, S = 99      " INTERNAL REPRESENTATION OF CLOSE
Y, GOTO (:END)
SKIP2: SUBC (:NXT BSC SBL)
U, S = 9, P      " NEXT BASIC SYMBOL
U, S = 63, E      " NOT A LETTER ?
N, GOTO (:SKIP2)      " ELSE SKIP
U, S = 90, Z      " LAST BASIC SYMBOL = COLON ?
Y, SUBC (:NXT BSC SBL)
  STOCK1 = S
N, A = 100
N, SUBC (:ERRORM)
U, S = 98, Z      " LAST BASIC SYMBOL = OPEN ?
Y, STOCK1 = - S
N, A = 101
N, SUBC (:ERRORM)
  S = 87      " INTERNAL REPRESENTATION OF COMMA
END: LAST SYMBOL = S
      A = S
U, A = 87, P      " (LAST SYMBOL ≠ PERIOD

```

" GENERAL PURPOSE PROCEDURES NR. 40 CONTINUED

```

U, A = 89, E           " ^ LAST SYMBOL ≠ TEN)
Y, A = 9, P            " IMPLIES LAST SYMBOL > 9 ?
N, S = 0
    DIGIT LAST SYMBOL = S
Y, S '*' = 63, Z      " = DIGIT LAST SYMBOL ^ LAST SYMBOL
N, S = 1              " < 64 ?
    LETTER LAST SYMBOL = S
    S = LAST SYMBOL
    A = RUN NUMBER
    SUBC (:OUT SBL)
    GOTO (:TEST POINTERS)" 59 INSTRUCTIONS

'END' NXT SBL

```

TEST DPO:

TEST POINTERS: 'BEGIN' LOOP, STACK TEST, NON SHIFT

```

A = BEGIN OF TEXT ARRAY
M[6] = - A           " - LOWER BOUND
S = - END OF TEXT ARRAY
M[6] = S             " + UPPER BOUND
A = DPO
S + NLP
U, A = 20, P         " ENOUGH SPAAE FOR CONSTANTS?
N, JUMP(1)
U, S = 20, P         " ENOUGH SPACE FOR IDENTIFIERS
Y, GOTO(:STACK TEST)
    A + S
U, A = 60, P         " DOES SHIFT MAKE ANY SENSE?
N, GOTO(:NON SHIFT)
    RUA(1)
    S = A             " AMOUNT OF SHIFT
    M[B] = S, P
    A = END OF TEXT ARRAY
    END OF TEXT ARRAY + S
    TEXT ARRAY POINTER + S
    BEGIN OF TEXT ARRAY + S
LOOP:
N, A = M[6]
    F =:MA
    S = MG
    G + M[B]
    MG = S
Y, A = 1
N, A+ 1
    REP6E(:LOCP)
STACK TEST:
    S = END OF STACK
    S = :MC[20], P
Y, S = LAST SYMBOL
Y, GOTOR(MC[-1])
    A = 491
    GOTO(:ERM)
NON SHIFT:
    A = 492
    GOTO(:ERM)           " 36 INSTRUCTIONS

'END' TEST POINTERS

NXT BSC SBL:
    A = RUN NUMBER
    SUBC (:IN SBL)

```

```

U, S = 119, Z          " SYMBOL = NEW LINE ?
N, GOTOR (MC[-1])
  A = 1
  LINE COUNTER + A
  A = QUOTE COUNTER, Z
N, GOTOR (MC[-1])
  A = RUN NUMBER
  SUBC (:OUT SBL)
  GOTO (:NXT BSC SBL) " 11 INSTRUCTIONS

IN SBL:  'BEGIN' ELSE0, ELSE1, ELSE2, ELSE3, ELSE4, ELSE5, ELSE6,
        NEXT0, NEXT1, NEXT2, NEXT3

U, A = 100, Z          " RUNNUMBER = 100?
  A = CMODE            " ^ (CMODE = 1~8~16)?
Y, A '*' -25, Z
Y, GOTO(:ELSE1)
  A = TEXT ARRAY POINTER
  S = MA              " TEXT[TEXT ARRAY POINTER]
  A = SHIFT
U, A = 256, P          " SHIFT > 256 ?
N, GOTO (:ELSE0)
  RUS (16)
  A = 1
  TEXT ARRAY POINTER + A
  SHIFT = A
  GOTOR (MC[-1])
ELSE0:  U, A = 1, Z      " SHIFT = 1 ?
  N, RUS (8)
  S '*' 255
  LUA (8)             " SHIFT := 256 * SHIFT
  SHIFT = A
  GOTOR (MC[-1])
ELSE1:  S = STOCK, P    " SYMBOL IN STOCK ?
  Y, STOCK = - S
  N, SUBC (:NXT TAPE SL)
  U, S = 101, P        " NEXT TAPE SYMBOL > BUS ?
  N, GOTOR (MC[-1])
  U, S = 123, Z
  Y, S = 93           " INTERNAL REPRESENTATION OF SPACE
  A = QUOTE COUNTER, Z

```

" GENERAL PURPOSE PROCEDURES NR. 40/41

```

      Y, GOTO (:ELSE3)
      U, S = 127, Z      " LAST TAPE SYMBOL = BAR ?
      N, GOTO (:ELSE2)
NEXT0: SUBC (:INXT TAPE SL)
      U, S = 127, Z      " NEXT TAPE SYMBOL = BAR ?
      Y, GOTO (:NEXT0)
      STOCK = S
      A = QUOTE COUNTER
      U, S = 72, Z      " LES AFTER BAR ?
      Y, A + 2          " THEN QUOTE COUNTER = QUOTE COUNTER + 1
      N, S = 74, Z      " MOR AFTER BAR ?
      S = 127          " INTERNAL REPRESENTATION OF BAR
      N, JUMP (4)
      A = 1, P          " NEW VALUE OF QUOTE COUNTER > 0 ?
      Y, QUOTE COUNTER = A
      N, S = 103        " INTERNAL REPRESENTATION OF UNQUOTE
      N, STOCK = - S
ELSE2: U, S = 124, Z    " LAST TAPE SYMBOL = COLON ?
      Y, S = 90         " INTERNAL REPRESENTATION OF COLON
      U, S = 125, Z    " LAST TAPE SYMBOL = CLOSE ?
      Y, S = 99         " INTERNAL REPRESENTATION OF CLOSE
      GOTOR (MC[-1])
ELSE3: U, S = 118, P    " LAST TAPE SYMBOL ≥ NEW LINE ?
      N, GOTO (:ELSE1)  " ELSE SKIP
      U, S = 127, Z    " LAST TAPE SYMBOL = BAR ?
      N, GOTO (:ELSE4)
NEXT1: SUBC (:INXT TAPE SL)
      U, S = 127, Z    " NEXT TAPE SYMBOL = BAR ?
      Y, GOTO (:NEXT1)
      U, S = 80, Z      " AND AFTER BAR ?
      Y, S = - 2
      N, S = 70, Z      " EQU AFTER BAR ?
      Y, S = 31
      N, S = 2, Z       " LES AFTER BAR ?
      Y, S = 1
      N, S = 2, Z       " MCR AFTER BAR ?
      Y, S + 103
      N, S = 160        " ELSE INADMISSABLE
      GOTOR (MC[-1])
ELSE4: U, S = 126, Z    " LAST TAPE SYMBOL = UNDERLINING ?
      N, GOTO (:ELSE6)
      SUBC (:UND SBL)
      U, S = 63, P      " AFTER UNDERLINING NO LETTER OR DIGIT ?
      N, GOTO (:ELSE5)
      U, S = 70, Z      " EQU AFTER UNDERLINING ?
      Y, S = 4
      N, S = 72, Z      " LES AFTER UNDERLINING ?
      Y, S = 2
      N, S = 2, Z       " MCR AFTER UNDERLINING ?
      Y, S = 3
      N, S = 2, Z       " NCN AFTER UNDERLINING ?
      Y, S + 10
      N, S = 48, Z      " COLON AFTER UNDERLINING ?
      Y, S + 68
      N, S = 161        " ELSE INADMISSABLE
      GOTOR (MC[-1])
ELSE5: LUS (7)
      STOCK = S

```

SUBC (:NXT TAPE SL)

" GENERAL PURPOSE PROCEDURES NR. 41 CONTINUED

```
      U, S = 126, Z          " NEXT TAPE SYMBOL = UNDERLINING ?
      N, STOCK = S
      N, GOTO (:ELSE5[-2])    " ELSE INADMISSABLE
      SUBC (:UND SBL)
      S = STOCK
      LUS (7)
      A = 23
      COUNT = A
      F = :WORD DEL
NEXT2:  A = MG                " CYCLE: LOOK UP WORD DELIMITER
      A = S
      U, A '*' - 127, Z      " FOUND ?
      N, F + 1
      N, REPE (:NEXT2)
      Y, S = A
      N, S = 162             " ELSE INADMISSABLE COMBINATION
      STOCK = S
NEXT3:  SUBC (:NXT TAPE SL)   " CYCLE: SKIP REST OF WORD DELIMITER
      U, S = 126, Z          " NEXT TAPE SYMBOL = UNDERLINING ?
      Y, SUBC (:UND SBL)
      Y, GOTO (:NEXT3)
      A = STOCK
      STOCK = S
      S = A
      GOTOR (MC[-1])
ELSE6:  U, S = 124, Z         " LAST TAPE SYMBOL = COLON ?
      N, GOTOR (MC[-1])
      SUBC (:NXT TAPE SL)
      U, S = 70, Z           " ECU AFTER COLON ?
      Y, S = 92              " INTERNAL REPRESENTATION OF COLONEQUAL
      N, STOCK = S
      N, S = 90              " INTERNAL REPRESENTATION OF COLON
      GOTOR (MC[-1])         " 120 INSTRUCTIONS

      'END' IN SBL

UND SBL: SUBC (:NXT TAPE SL)
      U, S = 126, Z          " NEXT TAPE SYMBOL = UNDERLINING ?
      N, GOTOR (MC[-1])
      GOTO (:UND SBL)        " 4 INSTRUCTIONS
```

" GENERAL PURPOSE PROCEDURES NR. 42

```

OUT SBL:      A = 100, Z           " DESTINATION = 100 ?
              Y, A = CMODE
              Y, A = '6', Z         " ^ ( CMCDE#2^CMODE#4 ) ?
              N, GOTOR (MC[-1])
              A = SHIFT
              U, A = 256, Z         " PREVIOUS SHIFT = 256 ?
              Y, LUS (8)
              U, A = 256, P         " PREVIOUS SHIFT > 256 ?
              N, LCSA (8)
              Y, A = 1
              SHIFT = A            " UPDATE SHIFT
              A = TEXT ARRAY POINTER
              N, MA[-1] + S         " TEXT[TEXT ARRAY POINTER] =
              N, GOTOR (MC[-1])    " TEXT[TEXT ARRAY POINTER]+SHIFT*SOURCE
              A + 1                " TEXT ARRAY POINTER =
              TEXT ARRAY POINTER = A " TEXT ARRAY POINTER + 1
              END OF TEXT ARRAY = A
              MA[-1] = S           " TEXT[TEXT ARRAY POINTER] = SOURCE
              GOTOR (MC[-1])       " 19 INSTRUCTIONS

AR CP LS:     S = LAST SYMBOL
              U, S = 63, P
              U, S = 69, E

INVERT:       U, S = - T, P
              GOTO (MC[-1])        " 5 INSTRUCTIONS

REL OP LS:    S = LAST SYMBOL
              U, S = 75, P
              GOTO (:AR OP LS[2])  " 3 INSTRUCTIONS

BOOL OP LS:   S = LAST SYMBOL
              U, S = 76, P
              U, S = 80, E
              GOTO (:INVERT)       " 4 INSTRUCTIONS

DECL LS:      S = LAST SYMBOL
              S = 106, Z           " LAST SYMBOL = OWN ?
              OWN TYPE = S
              Y, SUBC (:NXT SBL)
              Y, S = 106, Z
              U, S = 4, E          " LAST SYMBOL NO <TYPE> ?
              Y, A = 1000
              N, A = :MS[-1]
              TYPE = A
              N, SUBC (:NXT SBL)
              N, JUMP (6)
              S = 5, Z            " LAST SYMBOL = ARRAY ?
              Y, TYPE = S
              A = OWN TYPE, Z
              Y, A = 104
              Y, SUBC (:ERRORM)
              S = LAST SYMBOL
              S = 111, Z          " LAST SYMBOL = ARRAY ?
              ARR DEC LS = S
              N, JUMP (6)
              S = 8               " C-ARA = 8

```

" GENERAL PURPOSE PROCEDURES NR. 42 CONTINUED

```

      A = OWN TYPE, Z
N, A = - 4
  A + TYPE
  ARR DEC MCR = A
  JUMP (9)
U, S = 1, Z          " LAST SYMBOL = PROCED ?
N, JUMP (4)
  S = TYPE
U, S = 3, P          " TYPE > 3 ?
Y, S = 10
N, S = 2, P
N, S = 2, Z          " LAST SYMBOL = SWITCH ?
Y, S + 14
N, S = TYPE
  CHARACTER = S
U, S = 8, P          " CHARA > 8 ?
Y, A = OWN TYPE, Z  " ^ OWN TYPE ?
Y, A = 105
Y, SUBC (:ERRORM)
U, S = 14, Z          " LAST SYMBOL = SWITCH ?
Y, A = - TYPE
Y, A + 4, P          " ^ TYPE < 4 ?
Y, A = 106
Y, SUBC (:ERRORM)
  S = - CHARACTER
U, S + 25, P          " CHARA < 25 ?
N, GOTO (MC[-1])
U, S + 4, P          " CHARA < 4 ?
Y, A = 0
N, A = 1
  TP DEC LS = A
  A = TYPE
Y, JUMP (3)
U, A = 3, P          " TYPE > 3 ?
Y, A = - S          " THEN CHARA
N, A = S            " ELSE TYPE + CHARA
U, A = OWN TYPE, Z
Y, A + 32
  LUA (19), P
Y, CHARACTER = A
N, CHARACTER = - A, P
  GOTO (MC[-1])      " 63 INSTRUCTIONS

```

" GENERAL PURPOSE PROCEDURES NR. 43

SPEC LS:

```

      S = LAST SYMBOL
U, S = 111, Z      " LAST SYMBOL = ARRAY ?
Y, S = 112         " THEN TYPE = 5
U, S = 106, E      " LAST SYMBOL NOT <TYPE> OR ARRAY ?
Y, A = 1000        " THEN TYPE = 1000
N, A = :MS[-107]
  TYPE = A
U, A = 3, P        " TYPE > 3 ?
N, SUBC (:NXT SBL)
U, S = 114, Z      " LAST SYMBOL = LABEL ?
Y, S = - 8
N, S = 113, Z      " LAST SYMBOL = SWITCH ?
Y, S + 14
N, S = 1000
  CHARACTER = S
  S + TYPE
U, S = 999, P      " TYPE + CHARA ≥ 1000 ?
N, A = 107
N, SUBC (:ERRORM)
  S = LAST SYMBOL
U, S = 112, Z      " LAST SYMBOL = PROCED ?
N, JUMP (4)
  S = TYPE
U, S = 3, P        " TYPE > 3 ?
Y, S = 16
N, S = 8, P
N, S = 111, Z      " LAST SYMBOL = ARRAY ?
Y, S + 8
Y, CHARACTER = S
  S = - CHARACTER
U, S + 25, P       " CHARA < 25 ?
Y, SUBC (:NXT SBL)
Y, S = - CHARACTER
  S = TYPE
  S + 2000, P      " TYPE + CHARA < 2000 ?
N, GOTO (MC[-1])
  S = CHARACTER
  A = TYPE
U, S = 8, P        " CHARA > 8 ?
Y, JUMP (5)
U, S = 6, Z        " CHARA = 6 ?
Y, A = 0
U, A = 5, Z        " TYPE = 5 ?
Y, A = 8
N, A + S
  A + 64
  VAL CHAR = A
  A = TYPE
U, A = 5, P        " TYPE > 5 ?
Y, JUMP (5)
U, A = 1, P        " TYPE > 1 ?
N, A = 4
U, S = 1000, Z     " CHARA = 1000 ?
Y, S = 0
  S + A
  S + 96
  LUS (19)
  CHARACTER = S, P

```

100470 -

1

171

GOTO (MC[-1])

" 99 INSTRUCTIONS

" GENERAL PURPOSE PROCEDURES NR. 43 CONTINUED

```

OP LS:      SUBC (:AR OP LS)
            N, SUBC (:REL OP LS[1])
            Y, GOTO (MC[-1])
            GOTO (:BOOL OP LS[1]) " 4 INSTRUCTIONS

UNS INT:    F = 0
            MC = F " HEAD = TAIL = 0
            SUBC (:MULTIPLY)
            U, S = 9, P " LAST SYMBOL > 9 ?
            N, JUMP (-3)
UNS INT[5]: G = MC[-1] " TAIL
            U, S = MC[-1], Z " HEAD = U ?
            N, F = 32767 " ELSE REPLACE TAIL BY 32767
            VALUE OF CONSTANT = F
            F = 32767, P " > 32767 ?
            Y, SMALL = S " THEN SMALL = FALSE
            GOTOR (MC[-1]) " 12 INSTRUCTIONS

MULTIPLY:   'BEGIN' TENTH

            A = LAST SYMBOL
            U, A = 9, P " LAST SYMBOL > 9 ?
            Y, A = 109
            Y, SUBC (:ERRORM)
            Y, S = LAST SYMBOL
            Y, JUMP (9)
            S = M[B-3]
            S = TENTH, P " HEAD > 6710885 ?
            N, S = M[B-2]
            N, MULAS (10) " TAIL = 10 * TAIL + LAST SYMBOL
            N, M[B-2] = S
            N, S = M[B-3]
            N, MULAS (10) " HEAD = 10 * HEAD + CARRY FROM TAIL
            N, M[B-3] = S
            SUBC (:NXT SBL)
            A = 1
            GOTO (MC[-1])
TENTH:      + 67 10885 " 18 INSTRUCTIONS

            'END' MULTIPLY

```

" GENERAL PURPOSE PROCEDURES NR. 44

```

UNS NBR:  F = 0          " MANTISSA = 0
          REAL NUMBER = G " REAL NUMBER = TRUE
          SMALL = G      " SMALL = TRUE, DECIMAL EXPONENT = 0
          S = LAST SYMBOL
U, S = 89, Z          " LAST SYMBOL = TEN ?
Y, F = 1              " THEN MANTISSA = 1
          MC = F        " HEAD AND TAIL OF MANTISSA
U, S = 9, P           " LAST SYMBOL > 9 ?
Y, JUMP (3)
          SUBC (:MULTIPLY)
Y, SMALL + A          " COUNT IN CASE OF OVERFLOW
          JUMP (-5)
          A = DIGIT LAST SYMBOL, Z " LAST SYMBOL = PERIOD OR TEN ?
N, A = M[B-2]
N, A = 0, P           " ~ HEAD = 0 ?
N, REAL NUMBER = B    " ELSE REAL NUMBER = FALSE
N, GOTO (:UNS INT[5]) " AND DELIVER SINGLE LENGTH INTEGER
U, S = 88, Z          " LAST SYMBOL = PERIOD ?
N, JUMP (5)
          SUBC (:NXT SBL)
          SUBC (:MULTIPLY) " HANDLE DECIMAL FRACTION
N, SMALL = A          " COUNT EXCEPT WHEN OVERFLOW
U, S = 9, P           " LAST SYMBOL > 9 ?
N, JUMP (-4)
U, S = 89, Z          " LAST SYMBOL = TEN ?
N, S = SMALL
N, JUMP (10)
          SUBC (:NXT SBL)
U, S = 64, Z          " LAST SYMBOL = PLUS ?
N, S = 65, Z          " ~ LAST SYMBOL = MINUS ?
Y, MC = S
Y, SUBC (:NXT SBL)
          SUBC (:UNS INT)
          S = M[B+2]      " TAIL OF DECIMAL EXPONENT
Y, A = MC[-1], Z      " WAS TEN FOLLOWED BY MINUS ?
Y, S = -S
          S + SMALL      " DECIMAL EXPONENT NOW COMPLETED
          A = S
          RUA (10), Z    " ABS (DECIMAL EXPONENT) < 1024 ?
N, S = 1024, E        " ELSE DECIMAL EXPONENT = 1024 *
N, S = -1024          " * SIGN (DECIMAL EXPONENT)
          F = MC[-2]     " HEAD AND TAIL OF MANTISSA
          SMALL = B      " SMALL = FALSE
          GOTO (:DEC BIN) " CONVERT TO BINARY NUMBER
                          " 44 INSTRUCTIONS

```

```

RD IDF:  'BEGIN' NEXT0, ELSE0, ELSE1

```

```

          S = NLP
          WORD COUNT = S " WORD COUNT = 0
U, S = LETTER LAST SYMBOL, Z
Y, A = 0
N, A = 1
          MS = -A        " NAME LIST[NLP] = -1, WORD = 0
N, A = 110
N, SUBC (:ERRORM)
N, GOTO (:ELSE1)
          COUNT = A      " COUNT = 0

```

" GENERAL PURPOSE PROCEDURES NR. 44 CONTINUED

```

      S = LAST SYMBOL
NEXT0:  U, S '*' - 63, Z      " LAST SYMBOL < 64 ?
      N, GOTO (:ELSE0)
      S = COUNT
      S + 4, Z      " COUNT = 4 ?
      Y, COUNT = S      " THEN COUNT = 0
      A = WORD COUNT
      Y, A = 1      " AND WORD COUNT = WORD COUNT + 1
      Y, WORD COUNT = A
      N, S = MA
      N, LUS (6)
      S = LAST SYMBOL
      S = 1
      MA = S
      SUBC (:NXT SBL)
      REP (:NEXT0)
ELSE0:  A = NLP
      S = - MA
      LAST IDENTIFIER = S  " LAST IDENTIFIER = NAME LIST[NLP]
      U, A = WORD COUNT, Z  " WORD COUNT = 0 ?
      Y, S = 0
      N, S = - MA[-1]      " ELSE LAST IDENTIFIER[1] =
      LAST IDENTIFIER[1] = S  " NAME LIST[NLP + 1]
ELSE1:  A = WORD COUNT
      S = 127
RD IDF1: LUS (19)
      MA[-1] = S      " NAME LIST[NLP + WORD COUNT + 1] =
      A = NLP      " 127 * D19
      WORD COUNT = - A
      GOTOR (MC[-1])  " 40 INSTRUCTIONS
      'END' RD IDF
NXT PNR: A = - MS, P      " WORD = - NAME LIST[POINTER]
      N, S = 1      " IF WORD ≤ 0 THEN POINTER = POINTER + 1
      N, GOTO (:NXT PNR)
      U, A = D25, P      " WORD > 33554432 ?
      N, GOTOR (MC[-1])
      S = :MA
      GOTO (:NXT PNR)      " 7 INSTRUCTIONS
LOOK UP: 'BEGIN' NEXT0, NEXT1, NEXT2
      S = IN FORMAL LIST, Z
      N, S = IN AR DEC, Z
      S = :MD[-4]
      Y, S = 1      " BLOCK CELL POINTER + (4 OR 5)
NEXT0:  SUBC (:NXT PNR)
      A = WORD COUNT
      COUNT = A
      G = LAST NLP
NEXT1:  A = -MS, P      " NAME LIST[POINTER + COUNT] =
      Y, A + MG, Z      " NAME LIST[LAST NLP + COUNT] ?
      N, GOTO (:NEXT2)
      S = 1      " THEN COUNT = COUNT + 1
      F = 1
      REPE (:NEXT1)

```

" GENERAL PURPOSE PROCEDURES NR. 44 CONTINUED

```

      A = MS, P          " FOUND ?
Y, GOTOR (MC[-1])
  S = 1
NEXT2:  A = MS, P
      Y, GOTO (:NEXT0)
      GOTO (:NEXT2[-1])  " 20 INSTRUCTIONS

      'END' LOOK UP
NAME IN LIBRARY: S = NLP
                S + D 25
                A = END OF CATALOGUE      " NAME LIST[END OF CATALOGUE] =
                MA = -S                    " -33554432 -NLP
                S = BEGIN OF CATALOGUE
                SUBC(:LOOK UP[4])
U, S = BEGIN OF CATALOGUE, P
U, S = END OF CATALOGUE, E  " OUTSIDE CATALOGUE?
                GOTO(MC[-1])              " 9 INSTRUCTIONS

IN NM LST:      S = INT LABELS, Z
N, GOTO (MC[-1])
                S = REAL NUMBER, Z
Y, GOTO (:INVERT)
                S = VALUE OF CONSTANT[1]
                RUS (18)                  " HEAD = VALUE OF CONSTANT : D18
                S + 4096
                A = NLP
                LAST NLP = A              " LAST NLP = NLP
                MA = - S                  " NAME LIST[NLP] = - 4096 - HEAD
                A = 1
                S = 4097
                LUS (18)                  " (HEAD = 1) * D18
                S = VALUE OF CONSTANT[1]
                MA = S
                S = 6
                SUBC (:RD IDF1)
                SUBC (:LOOK UP)
                INT LAB = S
                S = NLP, P                " INT LAB < NLP ?
                GOTO (MC[-1])            " 21 INSTRUCTIONS

```

" GENERAL PURPOSE PROCEDURES NR. 45

```

NXT IDF:      SUBC (:NXT PNR)
              S = 1
              A = MS, P
              Y, GOTOR (MC[-1])
              GOTO (:NXT IDF[1]) " 5 INSTRUCTIONS

SKP IDF:      S = LAST SYMBOL
              U, S '*' = 63, Z      " LAST SYMBOL < 64 ?
              Y, SUBC (:NXT SBL)
              Y, GOTO (:SKP IDF[1])
              GOTOR (MC[-1])      " 5 INSTRUCTIONS

SKP TP DEC:   SUBC (:SKP IDF)
              U, S = 87, Z      " LAST SYMBOL = COMMA ?
SKP TP DEC[2]: Y, SUBC (:NXT SBL)
              Y, GOTO (:SKP TP DEC)
              GOTOR (MC[-1])      " 5 INSTRUCTIONS

SKP VA LI:    S = LAST SYMBOL
              U, S = 115, Z      " LAST SYMBOL = VALUE ?
              N, GOTOR (MC[-1])
SKP VA LI[3]: SUBC (:SKP TP DEC[2])
              U, S = 91, Z      " LAST SYMBOL = SEMICOLON ?
              Y, GOTO (:NXT SBL)
              GOTOR (MC[-1])      " 7 INSTRUCTIONS

SKP SP LI:    SUBC (:SPEC LS)
              N, GOTOR (MC[-1])
              SUBC (:SKP VA LI[3])
              GOTO (:SKP SP LI) " 4 INSTRUCTIONS

DISP LVL:     S = MD[-1]      " NAME LIST[BLOCK CELL POINTER + 1]
              S '*' 63
              GOTOR (MC[-1])      " 3 INSTRUCTIONS

TP OF DSP:    S = MD[-1]      " NAME LIST[BLOCK CELL POINTER + 1]
              RUS (6)
              GOTO (:DISP LVL[1]) " 3 INSTRUCTIONS

LOC SPC:      S = MD[-1]      " NAME LIST[BLOCK CELL POINTER + 1]
              RUS (13)
              GOTOR (MC[-1])      " 3 INSTRUCTIONS

PRC LVL:      S = MD[-2]      " NAME LIST[BLOCK CELL POINTER + 2]
              GOTO (:DISP LVL[1]) " 2 INSTRUCTIONS

USE CST:      S = - MD[-2]    " D6 OF NAME LIST[BLOCK CELL POINTER+2]
              S '*' 64, Z      " = 1 ?
              GOTO (MC[-1])      " 3 INSTRUCTIONS

```

" GENERAL PURPOSE PROCEDURES NR. 45 CONTINUED

STATUS:	S = MD[-2]	" NAME LIST[BLOCK CELL POINTER + 2]
	RUS (13)	
	S + CORRECTION	
	GOTOR (MC[-1])	" 4 INSTRUCTIONS
IN CODE:	A = - MS[-1]	" D24 OF NAME LIST[N + 1]
	LCA (2), P	" = 1 ?
	GOTO (MC[-1])	" 3 INSTRUCTIONS

" GENERAL PURPOSE PROCEDURES NR. 46

```

ENTR BLK:      S = NXT BCP
                D = S
                S = MS
                S = 8191
                S + CORRECTION
                NXT BCP = S
                GOTOR (MC[-1])      " 7 INSTRUCTIONS

LOC LAB:      SUBC (:NONFRM LAB)
N, GOTO (MC[-1])
                SUBC (:CORSP BCP)
                G = D, Z
                GOTO (MC[-1])      " 5 INSTRUCTIONS

EXIT BLK:     S = MD
                RUS (13)
                S + CORRECTION
                D = S
                GOTOR (MC[-1])      " 5 INSTRUCTIONS

NONFRM LAB:   A = MS
                RUA (19)
                A = 6, Z
                GOTO (MC[-1])      " CODE BITS (N) = 6 ?
                " 4 INSTRUCTIONS

CORSP BCP:    G = D
                " P = BLOCK CELL POINTER
                U, S = G, P
                " N < P ?
N, A = MG[-2]
N, RUA (13)
N, A + CORRECTION
N, A = S, P
N, GOTOR (MC[-1])
                " N > NAME LIST[P + 2] & 8192 ?
                A = MG
                RUA (13), Z
Y, GOTOR (MC[-1])
                A + CORRECTION
                G = A
                GOTO (:CORSP BCP[1]) " 13 INSTRUCTIONS

SKIP STRING:  S = 1
                QUOTE COUNTER = S      " QUOTE COUNTER = 1
                SUBC (:NXT SBL)
U, S = 103, Z      " NEXT SYMBOL = UNQUOTE ?
N, JUMP (-3)      " ELSE SKIP
                A = 0
                QUOTE COUNTER = A      " QUOTE COUNTER = 0
                GOTOR (MC[-1])      " 8 INSTRUCTIONS

```

" GENERAL PURPOSE PROCEDURES NR. 46 CONTINUED

```

SKIP RST OF STAT: MC = A      " PR
                          S = LAST SYMBOL      " LAST SYMBOL
                          U, S = 86, Z        " = DC ?
                          Y, SUBC (:NXT SBL)
                          N, S = 81, Z        " = GOTC ?
                          N, S = 1, Z         " = FOR ?
                          N, S = 22, Z        " = BEGIN ?
                          Y, SUBC (M[B-1])    " THEN PR
                          S = LAST SYMBOL
                          U, S = 102, Z      " LAST SYMBOL = QUOTE ?
                          Y, SUBC (:SKIP STRING)
                          U, S = 91, Z       " LAST SYMBOL = SEMICOLON ?
                          N, S = 103, Z      " ~ LAST SYMBOL = END ?
                          N, SUBC (:NXT SBL)
                          N, GOTO (:SKIP RST OF STAT[2])
                          B = 1
                          GOTOR (MC[-1])     " 17 INSTRUCTIONS
ARUNVA:                  A = 262
RUNVA:                   LUA(17)
                          LCA(1)
                          S = S, Z          " ) AVOID -0
                          Y, S = 0          " )
                          S '+' A
                          GOTO (:BTSTRM27)
NEXT ENTRY:              S = 1              " ) POINTER := POINTER + 1
                          PLUSS(POINTER)    " )
                          A = MS             " CONTENTS OF CROSSTABLEWORD
                          U, S = END OF CROSSTABLE, Z
                          S = BEGIN OF CROSSTABLE " NUMBER OF ENTRY IN S
                          GOTO(MC[-1])      " 6 INSTRUCTIONS

INIT POINTER:            S = BEGIN OF CROSSTABLE
                          POINTER = S
                          GOTOR(MC[-1])     " 3 INSTRUCTIONS

INIT:                    RUNNUMBER = S
                          S = FIRST SHIFT
                          SHIFT = S        " INITIALIZE SHIFT THE SAME
                          F = 0             " AS IN PRESCAN0
                          S = 1
                          STOCK = -S       " STOCK = -1
                          STOCK1 = -S      " STOCK1 = -1
                          LAST SYMBOL = -S " LAST SYMBOL = -1
                          WORD COUNT = -S " WORD COUNT = -1
                          A = BEGIN OF TEXT ARRAY
                          TEXT ARRAY POINTER = A
                          VALUE OF CONSTANT = -F
                          LAST IDENTIFIER = F " LAST IDENTIFIER = EMPTY
                          LINE COUNTER = G " LINE COUNTER = 0
                          QUOTE COUNTER = G " QUOTE COUNTER = 0
                          FOR CNT = S      " FOR COUNT = 1
                          IN FORMAL LIST = S " IN FORMAL LIST = FALSE
                          IN AR DEC = S    " IN ARRAY DECLARATION = FALSE
                          GOTOR(MC[-1])    " 19 INSTRUCTIONS

```

" PRESCAN0 PROCEDURES NR. 50

'BEGIN'PROGRAM, BLOCK, CMP TL, DECL LST, STATMNT, BEGIN STAT, LAB DEC,

ILAB DEC, ST NUM CS, IDF, PRCS IDF, TRUNCATE, CLEAR CRSTAB

PROGRAM: 'BEGIN' PROGRAM0, PROGRAM1, PROGRAM2, PROGRAM3, PROGRAM4,
PROGRAM5

```

      S = 6
      LUS (19)
      CHARACTER = S          " CHARACTER = 6 * D19
PROGRAM0: S = LETTER LAST SYMBOL, Z
      N, GOTO (:PROGRAM2)
      SUBC (:RD IDF)
      A = LAST SYMBOL
      A = 90, Z              " LAST SYMBOL = COLON ?
      Y, SUBC (:PRCS IDF)
      Y, SUBC (:LAB DEC)
      A = 111
PROGRAM1:Y, GOTO (:PROGRAM0)
      GOTO (:PROGRAM4)
PROGRAM2: S = DIGIT LAST SYMBOL, Z
      N, GOTO (:PROGRAM3)
      SUBC (:UNS NBR)
      A = LAST SYMBOL
      A = 90, Z              " LAST SYMBOL = COLON ?
      Y, SUBC (:ILAB DEC)
      A = 112
      GOTO (:PROGRAM1)
PROGRAM3: S = LAST SYMBOL
      U, S = 104, Z          " LAST SYMBOL = BEGIN ?
      A = 113
PROGRAM4:N, SUBC (:ERRORM)
PROGRAM5:N, SUBC (:NXT SBL)
      U, S = 104, Z          " LAST SYMBOL = BEGIN ?
      Y, GOTO (:BEGIN STAT)
      GOTO (:PROGRAM5)      " 29 INSTRUCTIONS

```

'END' PROGRAM

BLOCK: 'BEGIN' BLCK, PROC, SEM TEST, SPEC TEST, TEST, BODY, END,
NEXT0, NEXT1, NEXT2, NEXT3, NEXT4, NEXT5, ELSE0

```

S = NLP
MS = B          " NAME LIST[NLP] = ADDRESS OF DUMP
MC = A, Z       " DUMP0 = PROC IDENTIFIER
A = D
MC = A          " DUMP1 = BLOCK CELL POINTER
F = 0
MC = F          " LOCAL FOR COUNT = MAX FOR COUNT = 0
MC = F          " LOCAL COUNT = LABEL COUNT = 0
MC = F          " INTERNAL BLOCK DEPTH =
A = M1[8]       " STRING OCCURRENCE = 0
MC = A          " DUMP8 = PRC LEVEL
S = 1
D = S           " BLOCK CELL POINTER = NLP + 1
A = OLD BCP     " NAME LIST[OLD BLOCK CELL POINTER] =
S = CORRECTION " NAME LIST[OLD BLOCK CELL POINTER] +
MA + S         " + BLOCK CELL POINTER

```

" PRESCANO PROCEDURES NR. 50 CONTINUED

```

      S + CORRECTION
      OLD BCP = S      " OLD BLOCK CELL POINTER =
      A = M1[1]        " BLOCK CELL POINTER
      A = CORRECTION
      LUA (13)
      MS = A           " NAME LIST[BLOCK CELL POINTER] =
      A = 1            " 8192 * DUMP1
      PLUSA (DSP LVL) " DISPL LEVEL = DISPL LEVEL + 1
      MS[-1] = A       " NAME LIST[BLOCK CELL POINTER + 1] =
      MS[-3] = G       " DISPL LEVEL
      S = 5            " NAME LIST[BLOCK CELL POINTER + 3] = 0
      NLP = S          " NLP = NLP + 6
      N, GOTO (:PROC)  " IF PRCC IDENTIFIER ≠ 0 THEN GOTO PROC
BLCK:  A = D25
      A = 4
      A + M1[1]
      MS = - A        " NAME LIST[NLP] = - 33554436 - DUMP1
      S = 1
      NLP = S         " NLP = NLP + 1
      S + D25         " NAME LIST[BLOCK CELL POINTER + 4] =
      MD[-4] = - S    " - 33554432 - NLP
      N, GOTOR (MC[-1]) " RETURN TO BODY IF CALLED FROM IT
      SUBC (:DECL LST)
      SUBC (:CMP TL)
      GOTO (:END)
PROC:  M1[8] = A      " PRC LEVEL = DISPL LEVEL
      MC = G         " FORMAL COUNT = 0
      S + D25        " NAME LIST[BLOCK CELL POINTER + 4] =
      MD[-4] = - S    " - 33554432 - NLP
      S = LAST SYMBOL
      U, S = 98, Z    " LAST SYMBOL = OPEN ?
      N, GOTO (:SEM TEST)
      S = 127
      LUS (19)
      CHARACTER = S   " CHARACTER = 127 * D19
NEXTU: SUBC (:NXT SBL)
      SUBC (:IDF)
      S = 1
      M[B-1] + S      " FORMAL COUNT = FORMAL COUNT + 1
      MINS (NLP)      " NLP = NLP + 1
      A = 0
      MS[1] = A       " NAME LIST[NLP - 1] = 0
      S = LAST SYMBOL
      U, S = 87, Z    " LAST SYMBOL = COMMA ?
      Y, GOTO (:NEXTU)
      U, S = 99, Z    " LAST SYMBOL = CLOSE ?
      Y, SUBC (:NXT SBL)
      N, A = 114
      N, SUBC (:ERRORM)
SEM TEST: U, S = 91, Z " LAST SYMBOL = SEMICOLON ?
      Y, SUBC (:NXT SBL)
      N, A = 115
      N, SUBC (:ERRORM)
      A = MC[-1]
      A + D22 PLUS 1
      S = M1
      MS[-1] = A      " NAME LIST[PROC IDENTIFIER] =
      S = LAST SYMBOL " D22 + FORMAL COUNT + 1

```

100470 -

1

182

U, S - 115, Z

" LAST SYMBCL = VALUE ?

" PRESCANO PROCEDURES NR. 50/51

```

      N, GOTO (:SPEC TEST)
NEXT1:  SUBC (:NXT SBL)
        SUBC (:IDF)
        U, S = LAST NLP, P " N < LAST NLP ?
        Y, A = 95
        Y, LUA (19)
        Y, MS = A " NAME LIST[N] = 95 * D19
        N, A = 116
        N, SUBC (:ERRORM)
          S = LAST NLP
          NLP = S " NLP = LAST NLP
          S = LAST SYMBOL
        U, S = 87, Z " LAST SYMBOL = COMMA ?
        Y, GOTO (:NEXT1)
          A = 117
NEXT2:  U, S = 91, Z " LAST SYMBOL = SEMICOLON ?
        Y, SUBC (:NXT SBL)
        N, SUBC (:ERRORM)
SPEC TEST: SUBC (:SPEC LS) " SPECIFIER LAST SYMBOL ?
        N, GOTO (:BODY)
NEXT3:  SUBC (:IDF)
        U, S = LAST NLP, P " N < LAST NLP ?
        N, A = 118
        N, SUBC (:ERRORM)
        N, GOTO (:TEST)
          A = MS
          RUA (19)
        U, A = 127, Z " NAME LIST[N] = 127 * D19 ?
        Y, A = CHARACTER
        Y, MS = A " NAME LIST[N] = CHARACTER
        Y, GOTO (:TEST)
        U, A = 95, Z " NAME LIST[N] = 95 * D19 ?
        N, A = 119
        N, GOTO (:NEXT3(3))
          A = VAL CHAR
        U, A = 75, P " VALUE CHARACTER > 75 ?
        Y, A = 120
        Y, SUBC (:ERRORM)
        N, LUA (19)
        N, MS = A " NAME LIST[N] = VALUE CHARACTER * D19
          A = TYPE
          A = 3, Z " TYPE = 3 ?
        Y, A = 64
        Y, M1[7] = A " THEN STRINGOCCURRENCE = 64
TEST:   S = LAST NLP
        NLP = S " NLP = LAST NLP
        S = LAST SYMBOL
        U, S = 87, Z " LAST SYMBOL = COMMA ?
        Y, SUBC (:NXT SBL)
        Y, GOTO (:NEXT3)
          A = 121
          GOTO (:NEXT2)
BODY:   S = NLP, Z
        SUBC (:BLCK)
        S = LAST SYMBOL
        U, S = 102, Z " LAST SYMBOL = QUOTE ?
        N, GOTO (:ELSE0)
          A = M1

```

100470 -

1

184

S = D24

" NAME LIST[PROC IDENTIFIER + 1] =

" PRESCAN0 PROCEDURES NR. 51 CONTINUED

```

NEXT4:      MA[-1] + S      " NAME LIST[PROC IDENTIFIER + 1] + D24
            SUBC (:NXT SBL)
            S = 103, Z      " NEXT SYMBOL = UNQUOTE ?
            N, GOTO (:NEXT4)
            JUMP (7)
ELSEU:      U, S = 104, Z    " LAST SYMBOL = BEGIN ?
            N, SUBC (:STATMNT)
            N, GOTO (:END)
            SUBC (:NXT SBL)
            SUBC (:DECL LS) " DECLARATOR LAST SYMBOL ?
            Y, SUBC (:DECL LST)
            SUBC (:CMP TL)
            SUBC (:NXT SBL)
END:         S = NLP
            S = CORRECTION
            LUS (13)        " 8192 + NLP (AS STATUS)
            S + MC[-1]      " + PRC LEVEL
            S + MC[-1]      " + STRING OCCURRENCE
            MD[-2] = S
            S = MC[-1]      " INTERNAL BLOCK DEPTH
            S + 1
            LUS (6)
            MD[-1] + S
            S = M[B+2], Z    " PRC LEVEL = 0 ?
            S = MC[-2]      " LOCAL COUNT
            N, B = 1
            Y, S + MC        " + LABEL COUNT
            S + MC[-1]      " + MAX FOR COUNT
            Y, GLOBAL COUNT + S
            N, LUS (13)
            N, MD[-1] + S
            S = MC, Z        " MAX FOR COUNT = 0 ?
            N, M[7] = S
            S = NLP
            N, A = D19
NEXT5:      N, MS = A        " NAME LIST[NLP] = D19
            N, S = 1         " NLP = NLP + 1
            N, REPP (:NEXT5)
            A = D25
            A = 5
            A + D            " NAME LIST[NLP] =
            MS = - A         " = 33554437 - BLOCK CELL POINTER
            S = 1
            NLP = S          " NLP = NLP + 1
            S + D25          " NAME LIST[BLOCK CELL POINTER - 1] =
            MD[1] = - S      " = 33554432 - NLP
            S = CMODE
            S '*' -2, Z      " INSERTING LIBRARY ROUTINES?
            S = MC[-1]      " DUMP1
            A = MS[-1]
            DSP LVL = A      " DISPL LEVEL = NAME LIST[DUMP1 + 1]
            A '*' 63
            Y, A = 1, Z      "(INSERTING)^(DISP LVL = 1) ?
            D = S           " BLOCK CELL POINTER = DUMP1
            A = MC[5]        " INTERNAL BLOCK DEPTH
            A + 1
            Y, SUBC (:TRUNCATE)
            GOTOR (MC[-1])   " 192 INSTRUCTIONS

```

100470 -

1

186

'END' BLOCK

" PRESCANO PROCEDURES NR. 52

```

CMP TL:          SUBC (:STATMNT)
                  S = LAST SYMBOL
U, S = 91, Z      " LAST SYMBOL = SEMICOLON ?
N, GOTOR (MC[-1])
                  SUBC (:NXT SBL)
                  GOTO (:CMP TL)          " 6 INSTRUCTIONS

DECL LST:        'BEGIN' STRINGTEST, END, ELSE0, ELSE1, ELSE2, NEXT1,
                  NEXT2, NEXT3, NEXT4, NEXT5

                  S = TP DEC LS, Z        " TYPE DECLARATOR LAST SYMBOL ?
N, GOTO (:ELSE0)
MC = S          " COUNT = 0
NEXT1:          S = 1
                  M[B-1] + S              " COUNT = COUNT + 1
                  SUBC (:IDF)
                  S = LAST NLP, P        " N < LAST NLP ?
Y, A = 122
Y, SUBC (:DERRORM)
S = LAST SYMBOL
U, S = 87, Z      " LAST SYMBOL = COMMA ?
Y, SUBC (:NXT SBL)
Y, GOTO (:NEXT1)
S = MC[-1]
A = TYPE, Z      " TYPE = 0 ?
N, A = 3, Z      " v TYPE = 3 ?
Y, LUS (1)       " THEN COUNT = 2 * COUNT
A = OWN TYPE, Z
Y, GLOBAL COUNT + S
N, M1[4] + S      " LOCAL COUNT = LOCAL COUNT + COUNT
STRINGTEST:     A = TYPE
A = 3, Z         " TYPE = 3 ?
Y, A = 64        " THEN STRINGOCCURRENCE = 64
Y, M1[7] = A
GOTO (:END)
ELSE0:          S = ARR DEC LS, Z        " ARR DECLARATOR LAST SYMBOL ?
N, GOTO (:ELSE1)
MC = S          " COUNT = 0
ARR POINTER = - S " ARRAY POINTER = 0
NEXT2:          S = 1
                  M[B-1] + S              " COUNT = COUNT + 1
                  SUBC (:NXT SBL)
                  SUBC (:IDF)
                  S = LAST NLP, P        " N < LAST NLP ?
Y, A = 123
Y, SUBC (:DERRORM)
S = NLP
A = ARR POINTER
MS = A          " NAME LIST[NLP] = ARRAY POINTER
ARR POINTER = S " ARRAY POINTER = NLP
S = 1
NLP = S         " NLP = NLP + 1
S = LAST SYMBOL
U, S = 87, Z      " LAST SYMBOL = COMMA ?
Y, GOTO (:NEXT2)
A = 0
DIMENSION = A   " DIMENSION = 0
U, S = 100, Z    " LAST SYMBOL = SUB ?

```

" PRESCAN0 PROCEDURES NR. 52 CONTINUED

```

N, A = 125
N, SUBC (:ERRORM)
N, GOTO (:NEXT4)
      SUBCOUNT = A      " SUBCOUNT = 0
NEXT3:  SUBC (:NXT SBL)
U, S = LETTER LAST SYMBOL, Z
Y, SUBC (:SKP IDF)
Y, JUMP (3)
U, S = DIGIT LAST SYMBOL, Z
Y, SUBC (:UNS NBR)
Y, SUBC (:ST NUM CS)
Y, S = LAST SYMBOL
U, S = 102, Z      " LAST SYMBOL = QUOTE ?
Y, SUBC (:SKIP STRING)
      A = 1
U, S = 90, Z      " LAST SYMBOL = COLON ?
Y, DIMENSION + A      " THEN DIMENSION = DIMENSION + 1
Y, GOTO (:NEXT3)
U, S = 100, Z      " LAST SYMBOL = SUB ?
Y, SUBCOUNT + A      " THEN SUBCOUNT = SUBCOUNT + 1
Y, GOTO (:NEXT3)
U, S = 101, Z      " LAST SYMBOL = BUS ?
N, GOTO (:NEXT3)
      S = SUBCOUNT, Z      " SUBCOUNT = 0 ?
N, SUBCOUNT = A      " ELSE SUBCOUNT = SUBCOUNT - 1
N, GOTO (:NEXT3)
      S = DIMENSION, Z      " DIMENSION = 0 ?
Y, A = 124
Y, SUBC (:ERRORM)
N, DIMENSION + A      " ELSE DIMENSION = DIMENSION + 1
      SUBC (:NXT SBL)
NEXT4:  G = DIMENSION
      S = ARR POINTER
      A = MS
      MS = G
      S = A, Z
      " N = NAME LIST[ARRAY POINTER]
      " NAME LIST[ARRAY POINTER] = DIMENSION
      " ARRAY POINTER = N
N, GOTO (:NEXT4[2])
      S = MC[-1]
      " CCUNT
U, S = OWN TYPE, Z
Y, F = 3
Y, F + 3
Y, MULS (G)
Y, GLOBAL COUNT + S
N, M1[4] + S
      S = LAST SYMBOL
      S = 87, Z      " LAST SYMBOL = COMMA ?
Y, GOTO (:NEXT2[-2])
      GOTO (:STRINGTEST)
ELSE1:  S = LAST SYMBOL
      " LAST SYMBOL = SWITCH ?
U, S = 113, Z
      SUBC (:NXT SBL)
      SUBC (:IDF)
N, GOTO (:ELSE2)
      S = LAST NLP, P      " N < LAST NLP ?
Y, A = 126
Y, SUBC (:DERRORM)
      A = 0
      S = 1

```

100470 -

1

189

MINS (NLP)

" NLP = NLP + 1

" PRESCANO PROCEDURES NR. 52 CONTINUED

```

      MS[1] = A           " NAME LIST[NLP - 1] = 0
NEXT5:  SUBC (:NXT SBL)
      U, S = LETTER LAST SYMBOL, Z
      Y, SUBC (:SKP IDF)
      Y, JUMP (3)
      U, S = DIGIT LAST SYMBOL, Z
      Y, SUBC (:UNS NBR)
      Y, SUBC (:ST NUM CS)
      Y, S = LAST SYMBOL
      U, S = 102, Z       " LAST SYMBOL = QUOTE ?
      Y, SUBC (:SKIP STRING)
      U, S = 91, Z        " LAST SYMBOL = SEMICOLON ?
      N, GOTO (:NEXT5)
      GOTO (:END)
ELSE2:  S = LAST NLP, P   " N < LAST NLP ?
      Y, A = 127
      Y, SUBC (:DERRORM)
      A = NLP
      S = TYPE
      U, S = 3, P         " TYPE > 3 ?
      N, LUS (19)
      N, MA[-1] = S       " ELSE NAME LIST[NLP + 1] = TYPE * D19
      S = :MA[-1]
      N, S = 1
      NLP = S             " NLP = NLP + (1 OR 2)
      A + 1
      SUBC (:BLOCK)
END:    S = LAST SYMBOL   " LAST SYMBOL = SEMICOLON ?
      U, S = 91, Z
      Y, SUBC (:NXT SBL)
      N, A = 128
      N, SUBC (:ERRORM)
      SUBC (:DECL LST)
      Y, GOTO (:DECL LST) " 142 INSTRUCTIONS
      GOTOR (MC[-1])
      'END' DECL LST

```

" PRESCANO PROCEDURES NR. 53

```

STATMNT:  'BEGIN' END, ELSE0, ELSE1, ELSE2, ELSE3, NEXT1

          A = M1[2]
          MC = A          " LFC = LOCAL FOR COUNT
          A = 6
          LUA (19)
          CHARACTER = A    " CHARACTER = 6 * D19
NEXT1:    S = LETTER LAST SYMBOL, Z
          N, GOTO (:ELSE0)
          SUBC (:RD IDF)
          A = LAST SYMBOL
          U, A = 90, Z      " LAST SYMBOL = COLON ?
          Y, SUBC (:PRCS IDF)
          Y, SUBC (:LAB DEC)
          Y, GOTO (:NEXT1)
ELSE0:    S = DIGIT LAST SYMBOL, Z
          N, GOTO (:ELSE1)
          SUBC (:UNS NBR)
          A = LAST SYMBOL
          U, A = 90, Z      " LAST SYMBOL = COLON ?
          Y, SUBC (:LAB DEC)
          Y, GOTO (:NEXT1)
          SUBC (:ST NUM CS)
ELSE1:    S = LAST SYMBOL
          U, S = 82, Z      " LAST SYMBOL = FOR ?
          N, GOTO (:ELSE2)
          S = 1
          PLUSS (M1[2])    " LOCAL FOR COUNT = LOCAL FOR COUNT + 1
          U, S = M1[3], P  " LOCAL FOR COUNT > MAX FOR COUNT ?
          Y, M1[3] = S     " THEN MAX FOR COUNT = LOCAL FOR COUNT
          GOTO (:END)
ELSE2:    U, S = 104, Z     " LAST SYMBOL = BEGIN ?
          Y, SUBC (:BEGIN STAT)
          Y, GOTO (:END)
ELSE3:    U, S = 102, Z     " LAST SYMBOL = QUOTE ?
          Y, SUBC (:SKIP STRING)
          U, S = 91, Z      " LAST SYMBOL = SEMICOLON ?
          N, S = 105, Z     " ~ LAST SYMBOL = END ?
          Y, S = MC[-1]
          Y, M1[2] = S      " LOCAL FOR COUNT = LFC
          Y, GOTOR (MC[-1])
END:      SUBC (:INXT SBL)
          GOTO (:NEXT1)    " 41 INSTRUCTIONS

          'END' STATMNT

BEGIN STAT: SUBC (:INXT SBL)
            SUBC (:DECL LS)    " DECLARATOR LAST SYMBOL ?
            N, GOTO (:CMP TL)
            A = 0
            SUBC (:BLOCK)
            U, A = M1[6], P     " N = BLOCK (0)
            Y, M1[6] = A       " N > INTERNAL BLOCK DEPTH ?
            GOTOR (MC[-1])     " THEN INTERNAL BLOCK DEPTH = N
                                " 8 INSTRUCTIONS

```

" PRESCANQ PROCEDURES NR. 53/54

```

LAB DEC:      S = LAST NLP, P      " N < LAST NLP ?
Y, A = 129
Y, SUBC (:DERRORM)
      A = M1[5], Z      " LABEL COUNT = 0 ?
      A + 2
      M1[5] = A      " LABEL COUNT = LABEL COUNT + 2
      S = NLP
Y, A = :MS[1]
Y, A = CORRECTION
Y, LUA (13)      " NAME LIST[BLOCK CELL POINTER + 3] =
Y, MD[-3] = A      " 8192 * (NLP - 1)
      A = D18
      MS = A      " NAME LIST[NLP] = D18
      S = 1
      NLP = S      " NLP = NLP + 1
      GOTO (:NXT SBL)      " 16 INSTRUCTIONS

ILAB DEC:     S = REAL NUMBER, Z
Y, A = 130
Y, SUBC (:ERRORM)
Y, GOTO (:NXT SBL)
      S = 0
      INT LABELS = S      " INT LABELS = TRUE
      SUBC (:IN NM LST)
      S = 3
      NLP = S      " NLP = NLP + 3
      S = INT LAB
      GOTO (:LAB DEC)      " 11 INSTRUCTIONS

ST NUM CS:    S = SMALL, Z
Y, GOTOR(MC[-1])
      F = VALUE OF CONSTANT
      S = 2
      INSTR CNTR + S
      PLUS(DP0)
      MS[-2] = F
      S = 257
      SUBC(BTSTRM9)
      S = VALUE OF CONSTANT
      SUBC(:BTSTRM27)
      S = 257
      SUBC(BTSTRM9)
      S = VALUE OF CONSTANT[1]
      SUBC(:BTSTRM27)
      GOTO(:TEST DP0)      " 16 INSTRUCTIONS

IDF:          SUBC (:RD IDF)
PRCS IDF:     S = NLP
      LAST NLP = S      " LAST NLP = NLP
      S = WORD COUNT
      S = 2
      NLP = S      " NLP = NLP + WORD COUNT + 2
      A = CHARACTER
      MS[1] = A      " NAME LIST[NLP - 1] = CHARACTER
      GOTO (:LOOK UP)      " 9 INSTRUCTIONS

```

```

TRUNCATE:      'BEGIN' ELSE0, ELSE1, END

                MC = A
                S = LAST SYMBOL          ")
                MC = S                  ") RELEVANT INTERNAL
                F = STOCK1              ") STATE FOR NXT SBL
                MC = F                  ")
                F = TEXT ARRAY POINTER  ")
                MC = F                  ")
                SUBC(: NXT SBL)
U, S = 105,Z
Y, GOTO(:END)

                S = SHIFT
                A = TEXT ARRAY POINTER
U, S = 256,Z
N, GOTO(:ELSE0)
                LUS(8)
                SHIFTSafe = S
                A = 1
                BEGINSafe = A
                S = 104                  ")SIMULATE TEXT ARRAY BEGINNING WITH
                LUS(16)                  ")BEGIN SYMBOL FOR NEXT PROCEDURE
                WORDSafe = S            ")
                S = MA[1]                ") SAVE SECOND WORD OF TEXT FOR
                WORD1SAFE = S           ") NEXT PROCEDURE
                S = 105
                MA[1] = S                " TRUNCATE TEXT BY INSERTING OF END SYMBOL
                GOTO(:END)

ELSE0:          U, S = 1, Z
N, GOTO(:ELSE1)
                S = 256
                SHIFTSafe = S
                A = 1
                BEGINSafe = A
                S = MA                  " TEXT[BEGINSafe]
                S '*' -FRAME            " MAKE PLACE FOR BEGIN SYMBOL
                S + 26624               " INSERT BEGIN SUMBOL
                WORDSafe = S
                S = MA                  " TEXT[BEGINSafe]
                S '*' D16 M1
                S + ENDSBL
                MA = S
                GOTO(:END)
ENDSBL:         + 6881280

```

```

ELSE1:      BEGINSAFE = A
            S = 1
            SHIFTSAFE = S
            S = MA
            S '*' = 255
            S + 104
            WORDSAFE = S
            S = MA
            S '*'-FRAME
            S + 26880
            " TEXT[BEGINSAFE]
            " MAKE PLACE FOR BEGIN SYMBOL
            " INSERT BEGIN SYMBOL

            " TEXT[BEGINSAFE]
            " MAKE PLACE FOR END_SYMBOL
            " INSERT END SYMBOL

            MA = S
            " TRUNCATE TEXT
END:        F = MC[-2]
            TEXT ARRAY POINTER = F
            F = MC[-2]
            STOCK1 = F
            S = MC[-1]
            LAST SYMBOL = S
            A = MC[-1]
            GOTOR(MC[-1])
            " 69 INSTRUCTIONS

            'END' TRUNCATE

CLEAR CRSTAB: 'BEGIN' LOOP

            SUBC(:INIT POINTER)
LOOP:      SUBC(:NEXT ENTRY)
            Y, GOTOR(MC[-1])
            A '*'-D21 PLUS D20
            S + BEGIN OF CROSSTABLE
            MS = A
            GOTO(:LOOP)
            " 7 INSTRUCTIONS

            'END' CLEAR CRSTAB

```

" MAIN PROGRAM OF PRESCAN0

```

      'BEGIN' NORMAL, LOOP

PRESCAN0:  SUBC(:CLEAR CRSTAB)      " CLEAR CROSSTABLE
          A = START OF CONSTANT LIST
          DPO = A                  " ) CONSTANT LIST POINTER
          A = CMODE
          A = 6, Z                  " ) IF CMODE = 2 ~ CMODE = 4
          N, A = BEGINSAFE          " ) THEN REPAIR
          Y, A = START OF TEXT ARRAY
          BEGIN OF TEXT ARRAY = A  " ) ELSE INITIALIZE
          N, S = SHIFTSAFE          " ) BEGIN OF TEXT ARRAY
          Y, S = 1                  " SAME FOR FIRST
          FIRST SHIFT = S
          Y, GOTO(:NORMAL)
          S = 256, P
          Y, S = WORD1SAFE
          Y, MA[1] = S
          S = - WORDSAFE, P
          MA = - S
          S = 100
NORMAL:    SUBC(:INIT)
          DSP LVL = G
          A = BEGIN OF NLI
          MA[1] = B
          MC = F
          MC = F
          MC = F
          MC = G
          MC = F
          GLOBAL COUNT = G
          INT LABELS = S
          D = A
          OLD BCP = A
          MA[-1] = F
          MA[-3] = G
          S = :MA[-6]
          NLP = S
          S + D25
          MA[-4] = - S
          Y, SHIFT = A
          SUBC(:NXT SBL)
          S = 1
          INSTR CNTR = S
          SUBC(:PROGRAM)
          S = GLOBAL COUNT
          S + M1[3]
          S + M1[5]
          S + M1[6]
          LUS (7)
          S + M1[6]
          LUS (6)
          S + 8256
          MD[-1] = S
          S = NLP
          S = CORRECTION
          LUS (13)
          MD[-2] = S
          " DISPL LEVEL = 0
          " NAME LIST(BEGIN OF NAME LIST - 1) =
          " ADDRESS OF DUMP0
          " DUMP0 = DUMP1 = 0
          " LOCAL FOR COUNT = MAX FOR COUNT = 0
          " LOCAL COUNT = LABEL COUNT = 0
          " INTERNAL BLOCK DEPTH = 0
          " STRING OCCURRENCE = PRC LEVEL = 0
          " GLOBAL COUNT = 0
          " INT LABELS = FALSE
          " BLOCK CELL POINTER = NLP
          " OLD BLOCK CELL POINTER = NLP
          " NAME LIST[NLP] = NAME LIST[NLP+1] = 0
          " NAME LIST[NLP + 3] = 0
          " NLP = NLP + 6
          " NAME LIST(BLOCK CELL POINTER + 4) =
          " - 33554432 - NLP
          " MAKE SHIFT > 256 IF CMODE = 1~8~16
          " INITIALIZE INSTRUCT COUNTER
          " INTERNAL BLOCK DEPTH
          " * 128
          " INTERNAL BLOCK DEPTH
          " * 64 (THEREFORE * 8192 IN TOTAL)
          " + 8256
          " NAME LIST(BLOCK CELL POINTER + 2) =
          " 8192 * NLP

```

```
      S = M1[3], Z      " MAX FOR COUNT
N, COUNT = S
      S = NLP
N, A = D19
LOOP:  N, MS = A        " NAME LIST[NLP] = D19
      N, S = 1
      N, REPP (:LOOP)
      N, NLP = S        " NLP = NLP + MAX FOR COUNT
      S = D25          " NAME LIST[BLOCK CELL POINTER + 5] =
      MD[-5] = - S      " - 33554432 - NLP
      B = 9
      SUBC (:DANGER)
      GOTO (:PRESCAN 1) " 67 INSTRUCTIONS

'END'
```

'END'

" PRESCAN1 PROCEDURES NR. 60

'BEGIN' ARITHEXP, S ARITHEXP, SUBS VAR, SBS LST, BOOLEXP, S BOOLEXP,
STREXP, S STREXP, DESEXP, S DESEXP, EXP, TYPE EXP, S EXP,
RST OF EXP, ASS STAT, RH SIDE, INSERT, FCT DES, PAR LST,
ACT PAR, PRC STAT, STATMNT, GT STAT, CMP TL, IFCLAUSE,
FOR STAT, SW DECL, SW LIST, ARR DEC, BND PR LST, PRC DEC,
BLOCK, DECL LST, PROGRAM, LAB STAT, ILAB STAT, LAB DEC,
ADDR BL IDF, STC ADDR, ADD TYPE, BOOLEAN, STRING, ARMETIC,
ARPOST, DESINAL, ASS TO, SBS VAR, PROC, FNCTN, LST LTH,
SW LTH, ADDRSS, CHK DIM, IDF, SCAN CODE, ASK LIBR

ARITHEXP:

S ARITHEXP: 'BEGIN' ELSE0, END

 SUBC (:AR OP LS) " ARITHOPERATOR LAST SYMBOL ?
Y, SUBC (:NXT SBL)
U, S = 98, Z " LAST SYMBOL = OPEN ?
N, GOTO (:ELSE0)
 SUBC (:NXT SBL)
 SUBC (:ARITHEXP)
 S = LAST SYMBOL
U, S = 99, Z " LAST SYMBOL = CLOSE ?
Y, SUBC (:NXT SBL)
 GOTO (:END)
ELSE0: U, S = 94, Z " LAST SYMBOL = IF ?
Y, A = :ARITHEXP
Y, GOTO (:IFCLAUSE)
 A = DIGIT LAST SYMBOL, Z
Y, SUBC (:UNS NBR)
Y, GOTO (:END)
 A = LETTER LAST SYMBOL, Z
Y, SUBC (:IDF)
Y, SUBC (:ARMETIC)
Y, SUBC (:SUBS VAR)
Y, SUBC (:FCT DES)
END: SUBC (:AR OP LS)
Y, GOTO (:ARITHEXP[1])
 GOTOR (MC[-1]) " 24 INSTRUCTIONS
 'END' S ARITHEXP

SUBS VAR: A = LAST SYMBOL
 A = 100, Z " LAST SYMBOL = SUB ?
N, GOTOR (MC[-1])
 MC = S " N
 SUBC (:SBS VAR)
 SUBC (:SBS LST)
 S = MC[-1] " N
 GOTO (:LST LTH) " 8 INSTRUCTIONS

SBS LST: SUBC (:NXT SBL)
 SUBC (:ARITHEXP)
 S = LAST SYMBOL
U, S = 87, Z " LAST SYMBOL = COMMA ?
Y, SUBC (:SBS LST)
SBS LST[5]: Y, A + 1
Y, GOTOR (MC[-1])

" PRESCAN1 PROCEDURES NR. 60/61

```

      U, S = 101, Z          " LAST SYMBOL = BUS ?
SBS LST[8]: Y, SUBC (:NXT SBL)
      A = 1
      GOTOR (MC[-1])        " 11 INSTRUCTIONS

BOOLEXP: 'BEGIN' ELSE0, ELSE1

      S = LAST SYMBOL
      U, S = 94, Z          " LAST SYMBOL = IF ?
      Y, A = :BOOLEXP
      Y, GOTO (:IFCLAUSE)
S BOOLEXP: S = LAST SYMBOL
      U, S = 76, Z          " LAST SYMBOL = NON ?
      Y, SUBC (:NXT SBL)
      U, S = 98, Z          " LAST SYMBOL = OPEN ?
      N, GOTO (:ELSE0)
S BOOLEXP[5]: SUBC (:NXT SBL)
      SUBC (:EXP)
      S = LAST SYMBOL
      U, S = 99, Z          " LAST SYMBOL = CLOSE ?
      Y, MC = A             " TYPE
      Y, SUBC (:NXT SBL)
      Y, A = MC[-1]        " TYPE
      GOTO (:RST OF EXP)
ELSE0: A = LETTER LAST SYMBOL, Z
      N, GOTO (:ELSE1)
      SUBC (:IDF)
      SUBC (:SUBS VAR)
      SUBC (:FCT DES)
      A = LAST SYMBOL
      U, A = 63, P          " LAST SYMBOL NOT AN
      U, A = 75, E          " ARITHOPERATOR OR A RELATOPERATOR ?
      N, SUBC (:ARMETIC)
      Y, SUBC (:BOOLEAN)
      GOTO (:RST OF EXP)
ELSE1: U, S = 116, Z        " LAST SYMBOL = TRUE ?
      N, S = 117, Z        " ~ LAST SYMBOL = FALSE ?
      Y, SUBC (:NXT SBL)
      Y, GOTO (:RST OF EXP)
      S + 53, Z            " LAST SYMBOL = PLUS ?
      N, S = 1, Z          " ~ LAST SYMBOL = MINUS ?
      N, S = DIGIT LAST SYMBOL, Z " ~ DIGIT LAST SYMBOL ?
      Y, SUBC (:S ARITHEXP)
      GOTO (:RST OF EXP)  " 37 INSTRUCTIONS

      'END' BOOLEXP

STREXP: S = LAST SYMBOL
      U, S = 94, Z          " LAST SYMBOL = IF ?
      Y, A = :STREXP
      Y, GOTO (:IFCLAUSE)
S STREXP: S = LAST SYMBOL
      U, S = 98, Z          " LAST SYMBOL = OPEN ?
      N, JUMP (6)
      SUBC (:NXT SBL)
      SUBC (:STR EXP)
S STREXP[5]: S = LAST SYMBOL

```

" PRESCAN1 PROCEDURES NR. 61 CONTINUED

```

      U, S = 99, Z           " LAST SYMBOL = CLOSE ?
S STREXP[7]: Y, GOTO (:NXT SBL)
              GOTOR (MC[-1])
              A = LETTER LAST SYMBOL, Z
      Y, SUBC (:IDF)
      Y, SUBC (:STRING)
      Y, SUBC (:SUBS VAR)
      Y, GOTO (:FCT DES)
      U, S = 102, Z          " LAST SYMBOL = QUOTE ?
      Y, SUBC (:SKIP STRING)
      GOTO (:S STREXP[7])   " 21 INSTRUCTIONS

DESEXP:      S = LAST SYMBOL
      U, S = 94, Z          " LAST SYMBOL = IF ?
      Y, A = :DESEXP
      Y, GOTO (:IFCLAUSE)
S DESEXP:    S = LAST SYMBOL
      U, S = 98, Z          " LAST SYMBOL = OPEN ?
      Y, SUBC (:NXT SBL)
      Y, SUBC (:DESEXP)
      Y, GOTO (:S STREXP[5])
      A = LETTER LAST SYMBOL, Z
      Y, SUBC (:IDF)
      Y, SUBC (:DESINAL)
      Y, GOTO (:SUBS VAR)
      A = DIGIT LAST SYMBOL, Z
      N, GOTOR (MC[-1])
      SUBC (:UNS NBR)
      SUBC (:IN NM LST)
      N, GOTOR (MC[-1])
      S = INT LAB
      GOTO (:DESINAL)      " 20 INSTRUCTIONS

EXP:         S = LAST SYMBOL
      U, S = 94, Z          " LAST SYMBOL = IF ?
      N, GOTO (:S EXP)
      SUBC (:NXT SBL)
      SUBC (:BOOLEXP)
      SUBC (:NXT SBL)
      SUBC (:S EXP)
      S = LAST SYMBOL
      U, S = 96, Z          " LAST SYMBOL = ELSE ?
      N, GOTOR (MC[-1])
      MC = A               " TYPE
      SUBC (:NXT SBL)
      A = MC[-1]           " TYPE
      S = - A
      S '*' 5, Z           " TYPE = 5 ~ TYPE = 7 ?
      N, MC = A            " TYPE
      A + :TYPE EXP[7]
      SUBC (:MA)
      N, A = MC[-1]        " RESTORE TYPE
      GOTOR (MC[-1])
      GOTO (:ARITHEXP)
      GOTO (:ARITHEXP)
      GOTO (:BOOLEXP)
      GOTO (:STREXP)

```

TYPE EXP[7]:

" PRESCAN1 PROCEDURES NR. 61/62

```

        GOTO (:ARITHEXP)
        GOTO (:EXP)
        GOTO (:DESEXP)
        GOTO (:EXP)          " 28 INSTRUCTIONS

S EXP:   'BEGIN' ELSE0, ELSE1

        S = LAST SYMBOL
U, S = 98, Z          " LAST SYMBOL = OPEN ?
Y, GOTO (:S BOOLEXP(5))
A = LETTER LAST SYMBOL, Z
N, GOTO (:ELSE0)
    SUBC (:IDF)
    SUBC (:SUBS VAR)
    SUBC (:FCT DES)
    A = LAST SYMBOL
U, A = 63, P          " LAST SYMBOL NOT AN
U, A = 75, E          " ARITHCOPERATOR OR A RELATOPERATOR ?
N, SUBC (:ARMETIC)
N, GOTO (:RST OF EXP)
U, A = 80, E          " AND NOT A BOOLOPERATOR ?
N, SUBC (:BOOLEAN)
N, GOTO (:RST OF EXP)
    SUBC (:NON FRM LAB)
Y, SUBC (:DESINAL)
    A = MS          " NAME LIST(N)
    RUA (19)
    A '*' 7          " TYPE BITS (N)
    GOTOR (MC[-1])
ELSE0:   A = DIGIT LAST SYMBOL, Z
N, GOTO (:ELSE1)
    SUBC (:UNS NBR)
    SUBC (:IN NM LST)
Y, S = INT LAB
Y, SUBC (:DESINAL)
Y, A = 7          " TYPE = UN
N, A = 4          " TYPE = AR
    GOTO (:RST OF EXP)
ELSE1:   S = LAST SYMBOL
U, S = 63, P          " LAST SYMBOL NOT
U, S = 65, E          " A PLUS OR MINUS SIGN ?
N, SUBC (:S ARITHEXP)
N, GOTO (:ELSE1[-2])
U, S = 102, Z          " LAST SYMBOL = QUOTE ?
Y, SUBC (:S STREXP)
Y, A = 3          " TYPE = ST
Y, GOTOR (MC[-1])
U, S = 76, Z          " LAST SYMBOL = NON ?
N, S = 116, Z          " ~ LAST SYMBOL = TRUE ?
N, S = 1, Z          " ~ LAST SYMBOL = FALSE ?
Y, SUBC (:S BOOLEXP)
Y, A = 2          " TYPE = BO
N, A = 7          " TYPE = UN
    GOTO (:RST OF EXP)    " 47 INSTRUCTIONS

        'END' S EXP

```

" PRESCAN1 PROCEDURES NR. 62 CONTINUED

```

RST OF EXP:  SUBC (:AR OP LS)
              Y, SUBC (:S ARITHEXP[1])
              Y, A = 4 " TYPE = AR
              SUBC (:REL OP LS)
              Y, SUBC (:NXT SBL)
              Y, SUBC (:S ARITHEXP)
              Y, A = 2 " TYPE = BO
              SUBC (:BOOL OP LS)
              Y, SUBC (:NXT SBL)
              Y, SUBC (:S BOOLEXP)
              Y, A = 2 " TYPE = BO
              GOTOR (MC[-1]) " 12 INSTRUCTIONS

ASS STAT:    'BEGIN' ELSE0, ELSE1, END0, END1

              SUBC (:SUBS VAR)
              A = LAST SYMBOL
              U, A = 92, Z " LAST SYMBOL = COLONEQUAL ?
              N, GOTOR (MC[-1])
              MC = S " N
              SUBC (:ASS TO)
              A = MS
              RUA (19)
              A '*' 7
              MC = A " TYPE N = TYPE BITS (N)
              SUBC (:NXT SBL)
              A = LETTER LAST SYMBOL, Z
              N, GOTO (:ELSE1)
              SUBC (:IDF)
              SUBC (:SUBS VAR)
              A = LAST SYMBOL
              U, A = 92, Z " LAST SYMBOL = COLONEQUAL ?
              N, GOTO (:ELSE0)
              A = M[B-1] " TYPE N
              SUBC (:INSERT)
              SUBC (:RH SIDE)
              A = MS
              RUA (19)
              A '*' 7 " TYPE = TYPE BITS (M)
              GOTO (:END1)
ELSE0:       SUBC (:FCT DES)
              A = LAST SYMBOL
              U, A = 63, P " LAST SYMBOL NOT AN
              U, A = 75, E " ARITHOPERATOR OR A RELATOPERATOR ?
              N, SUBC (:ARMETIC)
              N, GOTO (:END0)
              U, A = 80, E " AND NOT A BOOLOPERATOR ?
              N, SUBC (:BOOLEAN)
              N, GOTO (:END0)
              SUBC (:ARBQST)
              A = M[B-1] " TYPE = TYPE N
              U, A = 1, P " TYPE ≠ RE ^ TYPE ≠ IN ?
              N, A = 4 " ELSE TYPE = AR
              SUBC (:INSERT)
              A = MS
              RUA (19)
              A '*' 7 " TYPE = TYPE BITS (M)
              U, A = 1, P " TYPE ≠ RE ^ TYPE ≠ IN ?

```

" PRESCAN1 PROCEDURES NR. 62/63

```

      N, A = 4                                " ELSE TYPE = AR
      SUBC (:RST OF EXP)
      GOTO (:END1)
ELSE1:  A = M[B-1]                            " TYPE N
      U, A = 5, Z                            " = NONDES ?
      SUBC (:TYPE EXP)
      N, A = M[B-1]                            " ELSE LEAVE TYPE N UNCHANGED
      U, A = 1, P                            " TYPE ≠ RE ^ TYPE ≠ IN ?
      N, A = 4                                " ELSE TYPE = AR
ENDJ:   B = 1
      S = MC[-1]                             " N
      GOTO (:INSERT)                         " 55 INSTRUCTIONS

      'END' ASS STAT

INSERT: GOTO (:ADD TYPE)                     " 1 INSTRUCTION

FCT DES: A = LAST SYMBOL
      U, A = 98, Z                            " LAST SYMBCL = OPEN ?
      N, GOTOR (MC[-1])
      SUBC (:FNCTN)
FCT DES[4]: MC = S                          " N
      SUBC (:PAR LST)
      S = MC[-1]                             " N
      GOTO (:LST LTH)                        " 8 INSTRUCTIONS

PAR LST: SUBC (:NXT SBL)
      SUBC (:ACT PAR)
      S = LAST SYMBOL
      U, S = 87, Z                            " LAST SYMBCL = COMMA ?
      Y, SUBC (:PAR LST)
      Y, GOTO (:SBS LST[5])
      U, S = 99, Z                            " LAST SYMBCL = CLOSE ?
      GOTO (:SBS LST[8])                     " 8 INSTRUCTIONS

ACT PAR: GOTO (:EXP)                        " 1 INSTRUCTION

PRC STAT: SUBC (:PROC)
      A = LAST SYMBOL
      U, A = 98, Z                            " LAST SYMBCL = OPEN ?
      Y, GOTO (:FCT DES[4])
      A = 0
      GOTO (:LST LTH)                        " 6 INSTRUCTIONS

STATMNT: 'BEGIN' ELSE0, ELSE1

      A = LETTER LAST SYMBOL, Z
      N, GOTO (:ELSE0)
      SUBC (:IDF)
      A = LAST SYMBOL
      U, A = 90, Z                            " LAST SYMBCL = COLON ?
      Y, GOTO (:LAB STAT)
      U, A = 100, Z                          " LAST SYMBCL = SUB ?

```

" PRESCAN1 PROCEDURES NR. 63/64

```

      N, A = 92, Z           " ~ LAST SYMBOL = COLONEQUAL ?
      Y, GOTO (:ASS STAT)
      GOTO (:PRC STAT)
ELSE0:  A = DIGIT LAST SYMBOL, Z
      N, GOTO (:ELSE1)
      SUBC (:UNS NBR)
      A = LAST SYMBOL
      U, A = 90, Z           " LAST SYMBOL = COLON ?
      Y, GOTO (:LAB STAT)
      GOTOR (MC[-1])
ELSE1:  S = LAST SYMBOL
      U, S = 81, Z           " LAST SYMBOL = GOTO ?
      Y, GOTO (:GT STAT)
      U, S = 94, Z           " LAST SYMBOL = IF ?
      Y, A = :STATMNT
      Y, GOTO (:IFCLAUSE)
      U, S = 82, Z           " LAST SYMBOL = FOR ?
      Y, GOTO (:FOR STAT)
      U, S = 104, Z          " LAST SYMBOL = BEGIN ?
      N, GOTOR (MC[-1])
      SUBC (:NXT SBL)
      SUBC (:DECL LS)        " DECLARATOR LAST SYMBOL ?
      Y, SUBC (:BLOCK)
      N, SUBC (:CMP TL)
      GOTO (:NXT SBL)        " 32 INSTRUCTIONS

      'END' STATMNT

GT STAT:  SUBC (:NXT SBL)
          A = LETTER LAST SYMBOL, Z
      N, GOTO (:DESEXP)
          SUBC (:IDF)
          SUBC (:LOC LAB)
      Y, GOTOR (MC[-1])
          SUBC (:DESINAL)
          GOTO (:SUBS VAR)    " 8 INSTRUCTIONS

CMP TL:  SUBC (:STATMNT)
          S = LAST SYMBOL
      U, S = 91, Z           " LAST SYMBOL = SEMICOLON ?
      Y, SUBC (:NXT SBL)
      Y, GOTO (:CMP TL)
      U, S = 105, Z          " LAST SYMBOL = END ?
      Y, GOTOR (MC[-1])
          A = :STATMNT
          SUBC (:SKIP RST OF STAT)
          GOTO (:CMP TL[1])  " 10 INSTRUCTIONS

IFCLAUSE: MC = A             " PR
          SUBC (:NXT SBL)
          SUBC (:BOOLEXP)
          S = LAST SYMBOL
      U, S = 95, Z           " LAST SYMBOL = THEN ?
      Y, SUBC (:NXT SBL)
          SUBC (M[B-1])
          S = LAST SYMBOL

```

" PRESCAN1 PROCEDURES NR. 64 CONTINUED

```

      U, S = 96, Z           " LAST SYMBOL = ELSE ?
      Y, SUBC (:NXT SBL)
      Y, GOTO (MC[-1])       " THEN PR ONCE MORE
      B = 1
      GOTOR (MC[-1])        " 13 INSTRUCTIONS

FOR STAT:      'BEGIN' NEXT0, ELSE0

      SUBC (:NXT SBL)
      A = LETTER LAST SYMBOL, Z
      N, GOTOR (MC[-1])
      SUBC (:IDF)
      SUBC (:ARMETIC)
      SUBC (:SUBS VAR)
      SUBC (:ASS TO)
      S = LAST SYMBOL
NEXT0:      U, S = 92, Z           " LAST SYMBOL = COLONEQUAL ?
      Y, SUBC (:NXT SBL)
      SUBC (:ARITHEXP)
      S = LAST SYMBOL
      U, S = 110, Z          " LAST SYMBOL = STEP ?
      N, GOTO (:ELSE0)
      SUBC (:NXT SBL)
      SUBC (:ARITHEXP)
      S = LAST SYMBOL
      U, S = 84, Z           " LAST SYMBOL = UNTIL ?
      Y, SUBC (:NXT SBL)
      Y, SUBC (:ARITHEXP)
      JUMP (3)
ELSE0:      U, S = 85, Z           " LAST SYMBOL = WHILE ?
      Y, SUBC (:NXT SBL)
      Y, SUBC (:BOOLEXP)
      S = LAST SYMBOL
      U, S = 87, Z           " LAST SYMBOL = COMMA ?
      Y, GOTO (:NEXT0)
      U, S = 86, Z           " LAST SYMBOL = DO ?
      Y, SUBC (:NXT SBL)
      A = 1
      FORCNT + A             " FORCOUNT = FORCOUNT + 1
      SUBC (:STATMNT)
      A = 1
      FORCNT - A            " FORCOUNT = FORCOUNT - 1
      GOTOR (MC[-1])        " 35 INSTRUCTIONS

      'END' FOR STAT

SW DECL:      SUBC (:NXT SBL)
      A = LETTER LAST SYMBOL, Z
      N, GOTOR (MC[-1])
      SUBC (:IDF)
      A = LAST SYMBOL
      U, A = 92, Z           " LAST SYMBOL = COLONEQUAL ?
      N, GOTOR (MC[-1])
      MC = S                " N
      SUBC (:SW LIST)
      S = MC[-1]            " N
      GOTO (:SW LTH)        " 11 INSTRUCTIONS

```

" PRESCAN1 PROCEDURES NR. 65

```

SW LIST:      SUBC (:NXT SBL)
               SUBC (:DESEXP)
               S = LAST SYMBOL
               U, S = 87, Z           " LAST SYMBOL = COMMA ?
               Y, SUBC (:SW LIST)
               Y, A = 1
               N, A = 1
               GOTOR (MC[-1])        " 8 INSTRUCTIONS

ARR DEC:      'BEGIN' NEXT0, NEXT1, ELSE0

               SUBC (:NXT SBL)
               SUBC (:IDF)
               A = 1
               MC = A                " CCUNT = 1
               MC = S                " N
NEXT0:        S = LAST SYMBOL
               U, S = 87, Z           " LAST SYMBOL = COMMA ?
               Y, SUBC (:NXT SBL)
               Y, SUBC (:SKP IDF)
               Y, A = 1
               Y, M[B-2] = A         " CCUNT = COUNT + 1
               Y, GOTO (:NEXT0)
               S = 100, Z           " LAST SYMBOL = SUB ?
               Y, IN AR DEC = S      " IN ARRAY DECLARATION = TRUE
               Y, SUBC (:BND PR LST)
               Y, IN AR DEC = A      " IN ARRAY DECLARATION = FALSE
               N, A = 0
               DIMENSION = A
               S = MC[-1]           " N
               SUBC (:CHK DIM)
               A = OWN TYPE, Z
               A = MC[-1]           " CCUNT
N, GOTO (:ELSE0)
COUNT = A
NEXT1:        A = INSTR CNTR
               SUBC (:ADDRSS)
               MC = S                " N
               S = DIMENSION
               S = :MS[6]
               S = DIMENSION
               INSTR CNTR = S        " 3 * DIMENSION + 6
               SUBC (:ARUNVA)        " ADDED TO INSTRUCT COUNTER
               S = MC[-1]           " N
               SUBC (:NXT IDF)
               REPP (:NEXT1)
ELSE0:        S = LAST SYMBOL
               U, S = 87, Z           " LAST SYMBOL = COMMA ?
               Y, GOTO (:ARR DEC)
               GOTOR (MC[-1])        " 39 INSTRUCTIONS

               'END' ARR DEC

BND PR LST:   SUBC (:NXT SBL)
               SUBC (:ARITHEXP)
               S = LAST SYMBOL
               U, S = 90, Z           " LAST SYMBOL = COLON ?
               Y, SUBC (:NXT SBL)

```

Y, SUBC (:ARITHEXP)
S = LAST SYMBOL

" PRESCAN1 PROCEDURES NR. 65 CONTINUED

```

      U, S = 87, Z           " LAST SYMBOL = COMMA ?
      Y, SUBC (:BND PR LST)
      GOTO (:SBS LST[5])    " 10 INSTRUCTIONS

PRC DEC:  'BEGIN' NEXT0, ELSE0, END

      SUBC (:NXT SBL)
      SUBC (:IDF)
      MC = S                " N
      SUBC (:ENTR BLK)
      S = LAST SYMBOL
      U, S = 98, Z         " LAST SYMBOL = OPEN ?
      N, GOTO (:ELSE0)
NEXT0:    A = 0
          IN FORMAL LIST = A    " IN FORMAL LIST = TRUE
          SUBC (:NXT SBL)
          SUBC (:IDF)
          A = MS
          RUA (19)
          A = 95, Z         " NAME LIST[M] = 95 * D19 ?
      Y, A = D24            " 32 * D19
      Y, MS + A            " NAME LIST[M] = 127 * D19
          A = 201
          IN FORMAL LIST = A    " IN FORMAL LIST = FALSE
      Y, SUBC (:ERRORM)
      S = LAST SYMBOL
      U, S = 87, Z         " LAST SYMBOL = COMMA ?
      Y, GOTO (:NEXT0)
      U, S = 99, Z         " LAST SYMBOL = CLOSE ?
      Y, SUBC (:NXT SBL)
ELSE0:    U, S = 91, Z         " LAST SYMBOL = SEMICOLON ?
      Y, SUBC (:NXT SBL)
          SUBC (:SKP VA LI)
          SUBC (:SKP SP LI)
          S = M[B-1]        " N
          SUBC (:IN CODE)
      Y, B = 1
      Y, GOTO (:SCAN CODE)
          A = 64
      U, A 'A' MD[-2], Z    " - USE OF COUNTERSTACK ?
      Y, S = MS
      Y, RUS (19)
      Y, S = 19, Z         " ^ NAME LIST[N] & D19 = 19 ?
      Y, MD[-2] + A        " THEN ADD 64 TO
          S = LAST SYMBOL    " NAME LIST[BLOCK CELL POINTER+2]
      U, S = 104, Z        " LAST SYMBOL = BEGIN ?
      N, SUBC (:STATMNT)
      N, GOTO (:END)
          SUBC (:NXT SBL)
          SUBC (:DECL LS)    " DECLARATOR LAST SYMBOL ?
      Y, SUBC (:DECL LST)
          SUBC (:CMP TL)
          SUBC (:NXT SBL)
END:      S = MC[-1]        " N
          GOTO (:ADDR BL IDF) " 49 INSTRUCTIONS

      'END' PRC DEC

```

" PRESCAN1 PROCEDURES NR. 66

```

BLOCK:          SUBC (:ENTR BLK)
                SUBC (:DECL LST)
                SUBC (:CMP TL)
                S = 0
                GOTO (:ADDR BL IDF) " 5 INSTRUCTIONS

DECL LST:       'BEGIN' END

                A = TP DEC LS, Z      " TYPE DECLARATOR LAST SYMBOL ?
Y, SUBC (:SKP TP DEC)
Y, GOTO (:END)
                A = ARR DEC LS, Z     " ARR DECLARATOR LAST SYMBOL ?
Y, SUBC (:ARR DEC)
Y, GOTO (:END)
                S = LAST SYMBOL
U, S = 113, Z      " LAST SYMBOL = SWITCH ?
Y, SUBC (:SW DECL)
N, SUBC (:PRC DEC)
                S = LAST SYMBOL
END:            U, S = 91, Z          " LAST SYMBOL = SEMICOLON ?
Y, SUBC (:INXT SBL)
                SUBC (:DECL LS)      " DECLARATOR LAST SYMBOL ?
Y, GOTO (:DECL LST)
                GOTOR (MC[-1])      " 16 INSTRUCTIONS

                'END' DECL LST

PROGRAM:        A = LETTER LAST SYMBOL, Z
Y, SUBC (:IDF)
PROGRAM[2]:     Y, A = LAST SYMBOL
Y, A = 90, Z      " LAST SYMBOL = COLON ?
Y, SUBC (:LAB DEC)
Y, GOTO (:PROGRAM)
                A = DIGIT LAST SYMBOL, Z
Y, SUBC (:UNS NBR)
Y, SUBC (:IN NM LST) " IN NAME LIST ?
Y, S = INT LAB
Y, GOTO (:PROGRAM[2])
                S = LAST SYMBOL
U, S = 104, Z     " LAST SYMBOL = BEGIN ?
                SUBC (:INXT SBL)
N, JUMP (-3)      " SKIP UNTIL BEGIN
                SUBC (:DECL LS)      " DECLARATOR LAST SYMBOL ?
Y, GOTO (:BLOCK)
                GOTO (:CMP TL)      " 18 INSTRUCTIONS

LAB STAT:      SUBC (:NON FRM LAB)
Y, SUBC (:LAB DEC)
                GOTO (:STATMNT)      " 3 INSTRUCTIONS

ILAB STAT:     SUBC (:IN NM LST)
Y, S = INT LAB
                GOTO (:LAB STAT[1]) " 3 INSTRUCTIONS

```

" PRESCAN1 PROCEDURES NR. 66/67

```

LAB DEC:      A = MD[-2]
              A '*' 63, Z
              A = FORCNT
              LUA (20)
Y, A + INSTR CNTR
              MS[-1] + A
N, GOTO (:NXT SBL)
              SUBC (:DESINAL)
              A = INSTR CNTR
              SUBC (:ADDRSS)
              A = 2
              INSTR CNTR + A
              S = 257
              SUBC (BTSTRM9)
              S = 0
              SUBC (:BTSTRM27)
              S = 258
              SUBC (BTSTRM9)
              SUBC (:DISP LVL)
              LUS (18)
              S + DP0
              S = START OF CONSTANT LIST
              S + 1
              SUBC (:BTSTRM27)
              GOTO (:NXT SBL)
              " 25 INSTRUCTIONS

```

ADDR BL IDF: 'BEGIN' FORMALS, FOR VARS, LCCALS, TEST, ELSE0, END

```

              MC = S, Z
Y, S = 8192
Y, MD[-1] + S
              A = MD[-2]
              A '*' 63, Z
Y, B = 1
Y, GOTO (:STC ADDR)
              A = M[B-1], Z
              A = 257
Y, A + D18
N, SUBC (:TP OF DSP)
N, A + :MS[-1]
              SUBC (:DISP LVL)
N, A + S
              LUS (9)
              S + A
              MC = S
Y, GOTO (:FOR VARS)
              S = :MD[-5]
              GOTO (:TEST)
FORMALS:      A = M[B-1]
              SUBC (:ADDRSS)
              RUA (18)
U, A = 141, P
N, JUMP (3)
              A '*' 1
              A + 3
              JUMP (4)
U, A = 128, P
U, A = 132, E

```

```

" PROC LEEL = 0 ?
" D20 * FORCOUNT
" + (POSSIBLY) INSTRUCT COUNTER
" ADDED TO NAME LIST[N + 1]
") INSTRUCT CNTR:= INSTRUCT CNTR +2
" N = 0 ?
" THEN ADD 8192 TO
" NAME LIST[BLOCK CELL POINTER + 1]
" PROC LEVEL = 0 ?
" N = 0 ?
" 256 + 1 + D18 IN A
" OR TOP OF DISPLAY +
" + DISPLAY LEVEL + 256 IN A
" 512 * DISPLAY LEVEL IN S
" CCOUNTER
" F = BLOCK CELL POINTER + 5
" CCOUNTER
" CODE1
" CODE > /0 ?
" CODE1 -2 * CODE ?
" CODE #
" 65 AND # 66 ?

```

100470 -

1

210

TEST:

N, A = 1
Y, A = 2
M[B-1] + A
SUBC (:NXT IDF)
U, S = D, P

" INCREASE COUNTER APPROPRIATELY
" F = NEXT IDENTIFIER (F)
" F < BLOCK CELL POINTER ?

" PRESCAN1 PROCEDURES NR. 67 CONTINUED

```

N, GOTO (:FORMALS)
  A = D18
  M[B-1] + A
  S = M[B-2]
  A = MS
  RUA (19)
U, A = 24, Z
Y, GOTO (:FOR VARS)
U, A = 16, Z
N, A = 19, Z
  A = M[B-1]
  S = 2
  SUBC (:ADDRSS)
Y, A = 2
N, A = 1
U, A = WANTED, P
Y, A = 3
  M[B-1] + A
  LUA (13)
  MD[-1] + A
FOR VARS: SUBC (:STATUS)
  A = M[B-1]
U, A = MS, P
Y, MS + A
Y, A + 1
Y, S = 1
Y, GOTO (:FOR VARS[2])
  M[B-1] = A
  S = :MD[-4]
LOCALS: SUBC (:NXT IDF)
U, S = D, P
N, A = MD[-2]
N, RUA (13)
N, A + CORRECTION
N, A = S, P
N, A = MS
N, RUA (19)
N, A = 63, P
Y, GOTO (:END)
  A + 39, P
N, GOTO (:ELSE0)
  A = 11, P
Y, GOTO (:LOCALS)
  A = 0, Z
N, A + 3, Z
  A = INSTR CNTR
  SUBC (:ADDRSS)
  MC = S
Y, S = 2
N, S = 1
  INSTR CNTR + S
  SUBC (:ARUNVA)
  S = MC[-1]
  GOTO (:LOCALS)
ELSE0: A + 11, P
Y, GOTO (:LOCALS)
  G = A
  F + 7, Z

```

```

" CCOUNTER = COUNTER + D18
" N

```

```

" CCDE = NAME LIST[N] : D19

```

```

" CCDE = 16 ?
" ~ CODE = 19 ?
" CCOUNTER

```

```

" WANTED ?

```

```

" INCREASE COUNTER APPROPRIATELY
" INCREASE APPROPRIATELY
" NAME LIST[BLOCK CELL POINTER + 1]
" F = STATUS
" CCOUNTER
" NAME LIST[F] > 0 ?
" ADDRESS (F, COUNTER)
" CCOUNTER = COUNTER + 1
" F = F + 1

```

```

" CCOUNTER
" F = BLOCK CELL POINTER + 4
" F = NEXT IDENTIFIER (F)
" F < BLOCK CELL POINTER ?

```

```

" ~ F > STATUS ?

```

```

" ~ NAME LIST[F] : D19 > 63 ?

```

```

" CCDE > 24 ?

```

```

" CODE > 35 ?

```

```

" CODE = 35 ?
" ~ CODE = 32 ?

```

```

" CCDE > 13 ?

```

```

" CCDE = 6 ?

```

100470 -

1

212

Y, A = - MS(-1)
Y, A '*' D18, Z
Y, GOTO (:LOCALS)
A = M[B-1]
SUBC (:ADDRSS)

" ^ D18 OF NAME LIST[F + 1] = 1 ?

" CCOUNTER

" PRESCAN1 PROCEDURES NR. 67/68

```

      A = G, Z          " CCDE = 6 ?
N, A + 3, Z           " ~ CODE = 3 ?
N, A + 3, Z           " ~ CODE = 0 ?
Y, A = 2
N, A = 1
      M[B-1] + A        " INCREASE COUNTER APPROPRIATELY
      GOTO (:LOCALS)
      SUBC (:DISP LVL)
      S + 1
      LUS (9)           " 512 * (DISPLAY LEVEL + 1)
      S + D18           " + D18
      S = MC[-1], P     " > COUNTER ?
N, A = 202
N, SUBC (:ERRORM)
      B = 1
      GOTO (:EXIT BLK)  " 114 INSTRUCTIONS

      'END' ADDR BL IDF

STC ADDR:  'BEGIN' FOR VARS, LOCALS

      SUBC (:STATUS)    " F = STATUS
      A = INSTR CNTR
FOR VARS: U, A = MS, P   " NAME LIST[F] > 0 ?
Y, MS + A              " ADDRESS (F, INSTRUCT COUNTER)
Y, A + 1               " INSTRUCT COUNTER = INSTRUCT COUNTER+1
Y, S = 1               " F = F + 1
Y, GOTO (:FOR VARS)
      S = - INSTR CNTR
      INSTR CNTR = A
      S + A, Z          " ) INCREASE INSTR CNTR DURING
N, SUBC (:ARUNVA)      " ) LOADING WHEN APPROPRIATE
      S = :MD[-4]       " F = BLOCK CELL POINTER + 4
LOCALS:  SUBC (:NXT IDF) " F = NEXT IDENTIFIER (F)
U, S = D, P           " F < BLOCK CELL POINTER ?
N, A = MD[-2]
N, RUA (13)
N, A + CORRECTION
N, A = S, P           " ~ F > STATUS ?
Y, GOTO (:EXIT BLK)
      A = MS
      RUA (19)          " CODE = NAME LIST[F] & D19
U, A = 35, P          " CODE > 35 ?
Y, GOTO (:LOCALS)
U, A = 24, P          " CODE > 24 ?
Y, JUMP (4)
U, A = 13, P          " CODE > 13 ?
Y, GOTO (:LOCALS)
U, A = 6, Z           " CODE = 6 ?
Y, GOTO (:LOCALS)
      A = 31, Z         " CCDE = 0 (MOD 32) ?
N, A = 3, Z           " ~ CODE = 3 (MOD 32) ?
      A = INSTR CNTR
      SUBC (:ADDRSS)
      MC = S           " N
Y, S = 2
N, S = 1
      INSTR CNTR + S    " INCREASE INSTRUCT COUNTER APPROPRIATELY

```

100470 -

1

214

SUBC(:ARUNVA)
S = MC[-1]
GOTO (:LOCALS)

" N
" 40 INSTRUCTIONS

'END' STC ADDR

" PRESCAN1 PROCEDURES NR. 68/69

ADD TYPE: 'BEGIN' ELSE0, END0, END1

```

        M[B] = A          " T
        A = MS
        RUA (19)          " CODE = NAME LIST[N] + D19
        U, A = 95, P      " CODE > 95 ?
        N, GOTOR (MC[-1])
        U, A = 127, Z      " CODE = 127 ?
        Y, A = 31          " THEN (CODE - NEW CODE) = 31
        Y, JUMP (6)
        U, A = 120, Z      " CODE = 120 ?
        N, GOTO (:ELSE0)
        A = M[B]          " T
        A = 5, P          " T > 5 ?
        Y, GOTOR (MC[-1]) " THEN NO GAIN OF INFORMATION
        A = 8, P          " ELSE (CODE - NEW CODE) = 8 (AND YES)
        A = M[B]          " = T
        GOTO (:END1)
ELSE0:   A '*' 7          " TYPE = CODE - CODE + 8 * 8
        U, A = 7, Z      " TYPE = 7 ?
        Y, GOTO (:END0)
        U, A = 5, Z      " TYPE = 5 ?
        Y, GOTO (:END0)
        U, A = 4, Z      " TYPE = 4 ?
        Y, A = 2
        Y, A = M[B], P    " ^ T < 2 ?
        Y, A = 4
        END0: Y, A = M[B], P " TYPE > T ?
        END1: Y, LUA (19)
        Y, MS = A
        GOTOR (MC[-1])    " 29 INSTRUCTIONS

        'END' ADD TYPE

BOOLEAN: A = 2           " BC
        GOTO (:ADD TYPE) " 2 INSTRUCTIONS

STRING:  A = 3           " ST
        GOTO (:ADD TYPE) " 2 INSTRUCTIONS

ARMETIC: A = 4           " AR
        GOTO (:ADD TYPE) " 2 INSTRUCTIONS

ARBOST:  A = 5           " NCNDES
        GOTO (:ADD TYPE) " 2 INSTRUCTIONS

DESINAL: SUBC (:NONPRM LAB) " NCNFORMAL LABEL ?
        N, A = 6         " DES
        N, GOTO (:ADD TYPE)
        A = MS[-1]
        U, A '*' D18, Z   " D18 OF NAME LIST[N + 1] = 0 ?
        Y, GOTOR (MC[-1]) " THEN LABEL KNOWN TO BE COMPLICATED
        A '*' = D18
        MS[-1] = A       " D18 OF NAME LIST[N + 1] = 0

```

" PRESCAN1 PROCEDURES NR. 69 CONTINUED

```

      SUBC (:CORSP BCP) " P = CORRESPONDING BLOCK CELL POINTER
      A = MG[-2]
      A '*' 63, Z      " CORRESPONDING PROC LEVEL = 0 ?
N, A = 1
N, MG[-3] + A          " ELSE ADD 1 TO NAME LIST[P + 3]
N, A = 16384
N, MG[-1] + A          " AND ADD 2 TO CORRESPONDING LOCAL SPACE
      GOTOR (MC[-1])    " 16 INSTRUCTIONS

ASS TO:      A = MS
      RUA (19)          " CODE = NAME LIST[N] ± D19
U, A = 95, P          " CODE > 95 ?
N, GOTOR (MC[-1])
U, A = 127, Z          " CODE = 127 ?
Y, A = 101             " THEN CODE = 101
U, A = 101, P          " CODE > 101 ?
Y, GOTO (:ARBOST)
      LUA (19)          " CODE = D19
      A + D18           " + D18
      MS = A            " AS NEW VALUE OF NAME LIST[N]
      GOTOR (MC[-1])    " 12 INSTRUCTIONS

SBS VAR:      A = MS
      RUA (19)          " CODE = NAME LIST[N] ± D19
U, A = 95, P          " CODE > 95 ?
N, GOTOR (MC[-1])
U, A = 127, Z          " CODE = 127 ?
Y, A = - 16           " THEN (NEW CODE - CODE) = - 16
Y, JUMP (2)
U, A '*' 24, Z          " CODE < 104 ?
Y, A = 8              " THEN (NEW CODE - CODE) = 8
SBS VAR[9]: Y, LUA (19)
Y, MS + A
      GOTOR (MC[-1])    " 12 INSTRUCTIONS

PROC:      A = MS
      RUA (19)          " CODE = NAME LIST[N] ± D19
U, A = 95, P          " CODE > 95 ?
N, GOTOR (MC[-1])
U, A = 127, Z          " CODE = 127 ?
Y, A = - 7            " THEN (NEW CODE - CODE) = - 7
Y, GOTO (:SBS VAR[9])
U, A = 101, P          " CODE > 101 ?
Y, GOTOR (MC[-1])
      A = 16, P
      GOTO (:SBS VAR[9])" 11 INSTRUCTIONS

FNCTN:      SUBC (:ARBOST)
      GOTO (:PROC)      " 2 INSTRUCTIONS

LST LTH:      M[B] = A      " DIMENSION
      A = MS
      RUA (19)          " CODE = NAME LIST[N] ± D19
U, A = 95, P          " CODE > 95 ?

```

" PRESCAN1 PROCEDURES NR. 69/70

```

      N, GOTOR (MC[-1])
      A = MS[-1]
      U, A = :MA, Z      " DIMENSION NOT YET KNOWN ?
      Y, A + M[B]
      Y, A + 1
      Y, MS[-1] = A      " THEN ADD DIMENSION + 1
      GOTOR (MC[-1])    " 11 INSTRUCTIONS

SW LTH:  A + 1
         MS[-1] + A
         GOTOR (MC[-1]) " 3 INSTRUCTIONS

ADDRSS:  M[B] = A      " M
         A = MS        " NAME LIST[N]
         RUA (18)
         LUA (18)
         A + M[B]
         MS = A
         GOTOR (MC[-1]) " 7 INSTRUCTIONS

CHK DIM:  A + 1
         A = MS[-1], Z  " NAME LIST[N + 1] = DIMENSION + 1 ?
      Y, GOTOR (MC[-1])
         MS[-1] + A
         A = 203
         GOTO (:ERRORM) " 6 INSTRUCTIONS

IDF:     S = NLP
         LAST NLP = S    " LAST NLP = NLP
         SUBC (:RD IDF)
         SUBC (:LOOK UP)
      U, S = NLP, P      " N < NLP ?
      Y, GOTOR (MC[-1])
         SUBC (:ASK LIBR)
      U, S = NLP, P      " N < NLP ?
      Y, GOTOR (MC[-1])
         A = WORD COUNT
         A + 3
         MINA (NLP)      " NLP = NLP + WORD COUNT + 3
         F = 0
         MA[1] = G       " NAME LIST[NLP - 1] = 0
         A = 204
         GOTO (:ERRORM)  " 16 INSTRUCTIONS

SCAN CODE: SUBC (:EXIT BLK)
           SUBC (:NXT SBL)
      U, S = 65, Z      " LAST SYMBOL = MINUS ?
      Y, SUBC (:NXT SBL)
         A = LETTER LAST SYMBOL, Z
      Y, SUBC (:IDF)
      N, SUBC (:UNS INT)
         S = LAST SYMBOL
      U, S = 87, Z      " LAST SYMBOL = COMMA ?
      Y, GOTO (:SCAN CODE[1])

```

" PRESCAN1 PROCEDURES NR. 70 CONTINUED

```

      U, S = 103, Z          " LAST SYMBOL = UNQUOTE ?
      GOTO (:S STREXP[7])    " 12 INSTRUCTIONS

ASK LIBR:  'BEGIN' NEXT0, END

      MC = S                  " N
      SUBC(:NAME IN LIBRARY)
Y, GOTO(:END)
      A = NLP
      A = WORD COUNT
      A = 1                  " NLP = NLP + WORD COUNT + 1
NEXT0:    F = MS[-1]          " NAME LIST[NLP] = NAME LIST[M]
          MA[-1] = F          " NAME LIST[NLP + 1] = NAME LIST[M + 1]
          G = MS[-2], P      " NAME LIST[M + 2] ≥ 0 ?
          MA[-2] = G         " NAME LIST[NLP + 2] = NAME LIST[M + 2]
          A = 3              " NLP = NLP + 3
          N, S = 3           " M = M + 3
          N, GOTO (:NEXT0)
          NLP = A
          S = M[B-1]         " N
          S = MS              " M[N]
          S '*' 32767         " ISOLATE NUMBER IN LIBRARY
          S + BEGIN OF CROSSTABLE " CORRESPONDING ENTRY OF CROSSTABLE
          A = MS              " ) INDICATE
          A '+' D20           " ) PRIMARY EEEDED
          MS = A
END:      S = MC[-1]
          GOTOR (MC[-1])     " 22 INSTRUCTIONS

          'END' ASK LIBR
          'SKIP' 20          "RESERVE SPACE IN COMPILER PART
                              "OF CORE.

```

" MAIN PROGRAM OF PRESCAN1

```

PRESCAN1: S = 200
  SUBC (:INIT)
  A = NLP
  MA = - S
  MA[-1] = G
  A = BEGIN OF NLI
  D = A
  NXT BCP = A
  S = INSTR CNTR
  A = 263
  SUBC (:RUNVA)
  S = CMODE
  S ' * ' -2, Z
Y, JUMP(3)
  SUBC (:TP OF DSP)
  INSTR CNTR + S
  SUBC (:ARUNVA)
  SUBC (:NXT SBL)
  SUBC (:ENTR BLK)
  SUBC (:PROGRAM)
  SUBC (:STC ADDR)
  SUBC (:DANGER)
  GOTO (:TRANSL)

```

" NAMelist[NLP] = -1
 " NAMelist[NLP + 1] = 0
 " BLOCKCELL POINTER = BEGIN OF NAME LIST
 " NEXT BLOCKCELL POINTER =
 " BEGIN OF NAME LIST
 " SEND VALUE OF DPO TO DRUM
 " IN ORDER TO INITIALIZE RUN CORRECTLY
 " INSTRUCT COUNT = INSTRUCT COUNTER +
 " TOP OF DISPLAY
 " INDICATE INCREMENT OF INSTRUCT COUNTER
 " 21 INSTRUCTIONS

'END'

" TRANSLATOR PROCEDURES NR. 1

```

'BEGIN'
  ARITHEXP, IFCLAUSE, FUTURE, S ARITHEXP, NXT TERM,
  TERM, NXT FCTR,
  FACTOR, NXT PRIM, PRIMARY, REQ CLOSE, AR NAME, MCR DOS, MCR RET,
  RETURN, SUBS VAR, ADR DSCR, EVALON, SBS LST, REQ BUS, BOOLEXP,
  S BOOLEXP, NXT IMP, IMPL, NXT BL TRM, BL TRM, NXT BL FC, BL FAC,
  NXT BL SC, BL SEC, BL PRIM, RELATN, RST OF RL, BL PR RST,
  BL NAME, AR BL EXP, S A BL EXP, RST OF AE, RST OF BE, RST OF ABE,
  AR BL RST, STR EXP, S STR EXP, ST NAME, DES EXP, S DES EXP,
  DS NAME, AR DS EXP, N DS EXP, EXP, S EXP, EXP RST, ASN STAT,
  DST TYPE, PREPARE, IN ASN, RE ASN, BO ASN, ST ASN, AR ASN,
  UN ASN, FCT DES, PR STAT, PR CALL, ORD NUM, PAR LIST, ACT PAR,
  START ISR, NUM DSCR, PRCS OP, NON ASBL, LINE, LINE1, STATMNT,
  UNC STAT, STAT, GOTO STAT, CMPTL3MIN1, CMP TL, IF STAT, FOR STAT,
  STORE MCR, TAKE MCR, FOR LST, SW DEC, ARR DEC, BND PR LST,
  PR DEC, SAVE LNC, BLOCK, DEC LST, INSPECT DECL, LAB LST, PROGRAM,
  SUBSTT, SUBST2, ORD CNT, MACRO, MACRO2, MACRO4, MACRO3, LOAD,
  PRODUCE, PRCS STP, PRCS PAR, SAT, OPT OP, OPT NBR, LAB DEC,
  B REACT, CHARCTR, ARMETIC,
  REAL, INTEGER, BOOLEAN, STRING, DESINAL, NO TYPE, ARBOST,
  UNKNOWN, NON AR, NON RE, NON IN, NON BL, NON STR, NON DES,
  SIMPLE, SIMPLE1, SUBSCR, PRCC, FUNCTN, NON SMPL, NON SUBS,
  NON PROC, NON FCT, FORMAL, IN VA LI, ASS TO, DYNAMIC, IN LIBR,
  OP LIKE, OUT DEC, ASS TO FD, DECLARED, SUP LOC, LOC POS,
  SET IN DEC, MRK POS, PR ADR, ADDRSS, LST LTH, TEST FC, TEST RET,
  CHCK DIM, CHCK RET, CHCK LL, CHCK TP, NBR LL, NEXT LL, NXT F I,
  INCR STS, IDF, SKP PR LI, TRL CD, UNS NUM, AR CST, CST STR,
  CRF
  ARITHEXP:
  S = LAST SYMBOL

```

```

U, S = 94, Z          " LAST SYMBOL = IF ?
N, GOTO (:S ARITHEXP)
  SUBC (:IFCLAUSE)
  MC = S              " FUTURE1
Y, SUBC (:NXT SBL)
N, A = 300
N, SUBC (:ERRORM)
  SUBC (:S ARITHEXP)
  SUBC (:FUTURE)
Y, SUBC (:ARITHEXP)
  A = MC[-1]          " FUTURE2
Y, GOTO (:SUBSTT)
  A = 301
  GOTO (:ERRORM)      " 15 INSTRUCTIONS

IFCLAUSE:             SUBC (:NXT SBL)
                      SUBC (:BOOLEXP)
                      A = 106
                      S = 0          " FUTURE1 = 0
                      SUBC (:MACRO2) " MACRO2 (COJU, FUTURE1)
                      A = LAST SYMBOL
U, A = 95, Z          " LAST SYMBOL = THEN ?
  GOTO (MC[-1])       " 8 INSTRUCTIONS

FUTURE:              S = LAST SYMBOL
                      S = 96, Z      " LAST SYMBOL = ELSE ?
Y, A = 102            " THEN FUTURE2 = 0

```

" TRANSLATOR PROCEDURES NR. 1/2

```

      Y, SUBC (:MACRO2) " AND MACRO2 (JU, FUTURE2)
      Y, A = M[B-2]    " FUTURE1
      Y, M[B-2] = S    " FUTURE2
      Y, SUBC (:SUBSTT)
      Y, SUBC (:NXT SBL)
      GOTO (MC[-1])    " 9 INSTRUCTIONS

S ARITHEXP:      S = LAST SYMBOL
      U, S = 64, Z    " LAST SYMBOL = PLUS ?
      N, S = 65, Z    " v LAST SYMBOL = MINUS ?
      Y, S = 1
      Y, MC = - S     " (- 63 IF PLUS, + 1 IF MINUS)
S ARITHEXP[5]:  Y, SUBC (:NXT SBL)
      SUBC (:TERM)
      Y, A = MC[-1], P " MACRO = NEG/ADD/SUB ?
      Y, SUBC (:MACRO) " 9 INSTRUCTIONS

NXT TERM:      S = LAST SYMBOL
      U, S = 63, P
      U, S = 65, E    " LAST SYMBOL NOT A PLUS OR MINUS SIGN ?
      Y, GOTOR (MC[-1]) " THEN EXPRESSION COMPLETED
      S = 62          " FORM MACRO ADD/SUB
      MC = S
      A = 0, Z
      SUBC (:MACRO)    " MACRO (STACK)
      GOTO (:S ARITHEXP[5]) " 9 INSTRUCTIONS

TERM:          SUBC (:FACTOR)    " 1 INSTRUCTION

NXT FCTR:      S = LAST SYMBOL
      U, S = 65, P
      U, S = 68, E    " LAST SYMBOL = MUL OR DIV OR IDI ?
      Y, GOTOR (MC[-1]) " THEN TERM COMPLETED
      S = 62          " FORM MACRO MUL/DIV/IDI
      MC = S
      A = 0
      SUBC (:MACRO)    " MACRO (STACK)
      SUBC (:NXT SBL)
      SUBC (:FACTOR)
      A = MC[-1]
      SUBC (:MACRO)    " MACRO (MUL/DIV/IDI)
      GOTO (:NXT FCTR) " 13 INSTRUCTIONS

FACTOR:        SUBC (:PRIMARY) " 1 INSTRUCTION

NXT PRIM:      S = LAST SYMBOL
      U, S = 69, Z    " LAST SYMBOL = TTP ?
      N, GOTOR (MC[-1]) " ELSE FACTOR COMPLETED
      A = 0
      SUBC (:MACRO)    " MACRO (STACK)
      SUBC (:NXT SBL)
      SUBC (:PRIMARY)
      A = 7

```

" TRANSLATOR PROCEDURES NR. 2 CONTINUED

```

SUBC (:MACRO)          " MACRO (TTP)
GOTO (:NXT PRIM)      " 10 INSTRUCTIONS

PRIMARY:              S = DIGIT LAST SYMBOL, Z
Y, SUBC (:UNS NUM)
Y, GOTO (:AR CST)
S = LETTER LAST SYMBOL, Z
Y, SUBC (:IDF)
Y, SUBC (:SUBS VAR)
Y, SUBC (:FCT DES)
Y, GOTO (:AR NAME)
S = LAST SYMBOL
U, S = 98, Z          " LAST SYMBOL = OPEN ?
N, JUMP (7)
SUBC (:NXT SBL)
SUBC (:ARITHEXP)
A = 302
REQ CLOSE:           S = LAST SYMBOL
U, S = 99, Z          " LAST SYMBOL = CLOSE ?
Y, GOTO (:NXT SBL)
GOTO (:ERRORM)
A = 303
SUBC (:ERRORM)
U, S = 94, Z          " LAST SYMBOL = IF ?
N, S = 64, Z          " v LAST SYMBOL = PLUS ?
N, S = 1, Z           " v LAST SYMBOL = MINUS ?
Y, GOTO (:ARITHEXP)
GOTOR (MC[-1])        " 25 INSTRUCTIONS

AR NAME:              SUBC (:NON AR)          " NONARITHMETIC ?
Y, A = 304
Y, SUBC (:ERRORM)
SUBC (:FORMAL)         " FORMAL ?
N, SUBC (:FUNCTION)    " v FUNCTION ?
Y, CMPLTD = S          " THEN COMPLICATED = TRUE
SUBC (:SIMPLE)         " SIMPLE ?
N, GOTOR (MC[-1])
MC = S                 " N
SUBC (:FORMAL)         " FORMAL ?
MCR DOS:              Y, A = 99              " MACRO2 (DOS, N)
Y, GOTO (:MCR RET)
SUBC (:INTEGER)        " INTEGER ?
Y, A = 84              " MACRO2 (TIV, N)
N, A = 83              " MACRO2 (TRV, N)
MCR RET:              SUBC (:MACRO2)
RETURN:               S = MC[-1]
GOTOR (MC[-1])        " 18 INSTRUCTIONS

SUBS VAR:              SUBC (:SUBSCR)        " SUBSCRIPTED ?
N, GOTOR (MC[-1])
SUBC (:ADR DSCR)
A = LAST SYMBOL
U, A = 92, Z          " LAST SYMBOL = COLONEQUAL ?
N, GOTO (:EVALON)
MC = S                 " N
A = 0

```

" TRANSLATOR PROCEDURES NR. 2/3

	SUBC (:MACRO)	" MACRO (STACK)
	A = 20	" MACRO (STAA)
	GOTO (:MCR RET)	" 11 INSTRUCTIONS
ADR DSCR:	A = LAST SYMBOL	
	U, A = 100, Z	" LAST SYMBOL = SUB ?
	N, A = 305	
	N, GOTO (:ERRORM)	" N
	MC = S	
	SUBC (:NXT SBL)	" N
	SUBC (:SBS LST)	
	DIMENSION = A	
	S = M[B-1]	
	SUBC (:CHCK DIM)	" FORMAL ?
	SUBC (:FORMAL)	" MACRO2 (DOS, N)
Y, GOTO (:MCR DOS)		" DESIGNATIONAL ?
	SUBC (:DESINAL)	" MACRO2 (TSWE, N)
Y, A = 93		" MACRO2 (TAK, N)
N, A = 92		" 16 INSTRUCTIONS
	GOTO (:MCR RET)	
EVALON:	MC = S	" N
	SUBC (:DESINAL)	" DESIGNATIONAL ?
N, JUMP (4)		
	SUBC (:FORMAL)	" FORMAL ?
Y, A = 27		" MACRO (TFSL)
N, A = 26		" MACRO (TSL)
	GOTO (:MCR RET)	
	SUBC (:BOOLEAN)	" BOOLEAN ?
Y, A = 23		
Y, GOTO (:MCR RET)		" MACRO (TSB)
	SUBC (:STRING)	" STRING ?
Y, A = 24		
Y, GOTO (:MCR RET)		" MACRO (TSST)
	SUBC (:FORMAL)	" FORMAL ?
Y, A = 25		
Y, GOTO (:MCR RET)		" MACRO (TFSU)
	SUBC (:INTEGER)	" INTEGER ?
Y, A = 22		" MACRO (TSI)
N, A = 21		" MACRO (TSR)
	GOTO (:MCR RET)	" 20 INSTRUCTIONS
SBS LST:	SUBC (:ARITHEXP)	
	S = LAST SYMBOL	
U, S = 87, Z		" LAST SYMBOL = COMMA ?
Y, A = 0		" MACRO (STACK)
Y, SUBC (:MACRO)		
Y, SUBC (:NXT SBL)		
Y, SUBC (:SBS LST)		
Y, A + 1		
Y, GOTOR (MC[-1])		
	A = 306	
REQ BUS:	U, S = 101, Z	" LAST SYMBOL = BUS ?
	Y, SUBC (:NXT SBL)	
	N, SUBC (:ERRORM)	
	A = 1	

" TRANSLATOR PROCEDURES NR. 3 CONTINUED

```

                                GOTOR (MC[-1])          " 15 INSTRUCTIONS

BOOLEXP:                      S = LAST SYMBOL
                                U, S = 94, Z            " LAST SYMBOL = IF ?
                                N, GOTO (:S BOOLEXP)
                                SUBC (:IFCLAUSE)
                                MC = S                  " FUTURE1
                                Y, SUBC (:NXT SBL)
                                N, A = 307
                                N, SUBC (:ERRORM)
                                SUBC (:S BOOLEXP)
                                SUBC (:FUTURE)
                                Y, SUBC (:BOOLEXP)
                                A = MC[-1]              " FUTURE2
                                Y, GOTO (:SUBSTT)
                                A = 308
                                GOTO (:ERRORM)          " 15 INSTRUCTIONS

S BCOLEXP:                     SUBC (:IMPL)            " 1 INSTRUCTION

NXT IMP:                       S = LAST SYMBOL
                                U, S = 77, Z            " LAST SYMBOL = QVL ?
                                N, GOTOR (MC[-1])        " ELSE EXPRESSION COMPLETED
                                A = 14
                                SUBC (:MACRO)            " MACRO (STAB)
                                SUBC (:NXT SBL)
                                SUBC (:IMPL)
                                A = 16
                                SUBC (:MACRO)            " MACRO (QVL)
                                GOTO (:NXT IMP)          " 10 INSTRUCTIONS

IMPL:                          SUBC (:BL TRM)          " 1 INSTRUCTION

NXT BL TRM:                    S = LAST SYMBOL
                                U, S = 78, Z            " LAST SYMBOL = IMP ?
                                N, GOTOR (MC[-1])        " ELSE IMPLICATION COMPLETED
                                A = 14
                                SUBC (:MACRO)            " MACRO (STAB)
                                SUBC (:NXT SBL)
                                SUBC (:BL TRM)
                                A = 17
                                SUBC (:MACRO)            " MACRO (IMP)
                                GOTO (:NXT BL TRM)       " 10 INSTRUCTIONS

BL TRM:                        SUBC (:BL FAC)          " 1 INSTRUCTION

NXT BL FC:                     S = LAST SYMBOL
                                U, S = 79, Z            " LAST SYMBOL = OR ?
                                N, GOTOR (MC[-1])        " ELSE BOOLEAN TERM COMPLETED
                                A = 14
                                SUBC (:MACRO)            " MACRO (STAB)
                                SUBC (:NXT SBL)

```

" TRANSLATOR PROCEDURES NR. 3/4

```

      SUBC (:BL FAC)
      A = 18
      SUBC (:MACRO)           " MACRO (OR)
      GOTO (:NXT BL FC)      " 10 INSTRUCTIONS

BL FAC:      SUBC (:BL SEC)   " 1 INSTRUCTION

NXT BL SC:   S = LAST SYMBOL
U, S = 80, Z           " LAST SYMBOL = AND ?
N, GOTOR (MC[-1])      " ELSE BOOLEAN FACTOR COMPLETED
      A = 14
      SUBC (:MACRO)           " MACRO (STAB)
      SUBC (:NXT SBL)
      SUBC (:BL SEC)
      A = 19
      SUBC (:MACRO)           " MACRO (AND)
      GOTO (:NXT BL SC)      " 10 INSTRUCTIONS

BL SEC:      S = LAST SYMBOL
U, S = 76, Z           " LAST SYMBOL = NON ?
N, GOTO (:BL PRIM)
      SUBC (:NXT SBL)
      SUBC (:BL PRIM)
      A = 15
      GOTO (:MACRO)           " 7 INSTRUCTIONS

BL PRIM:     S = LETTER LAST SYMBOL, Z
Y, SUBC (:IDF)
Y, SUBC (:SUBS VAR)
Y, GOTO (:BL PR RST)
      S = LAST SYMBOL
U, S = 98, Z           " LAST SYMBOL = OPEN ?
N, JUMP (14)
      SUBC (:NXT SBL)
      SUBC (:AR BL EXP)
      MC = A
      A = 309
      SUBC (:REQ CLOSE)      " REQUIRE CLOSE AS LAST SYMBOL
      A = MC[-1]
U, A = 4, Z           " TYPE = AR ?
Y, GOTO (:RST OF RL)
U, A = 8, Z           " TYPE = ARBO ?
N, GOTOR (MC[-1])
      SUBC (:AR OP LS)       " ARITHCOPERATOR LAST SYMBOL ?
Y, GOTO (:RST OF RL)
      SUBC (:RELATN)         " RELATION ?
      GOTOR (MC[-1])
U, S = 64, Z           " LAST SYMBOL = PLUS ?
N, S = 65, Z           " ~ LAST SYMBOL = MINUS ?
N, A = DIGIT LAST SYMBOL, Z " ~ DIGIT LAST SYMBOL ?
Y, SUBC (:S ARITHEXP)
Y, GOTO (:RST OF RL)
      S = 51, Z           " LAST SYMBOL = TRUE ?
N, S = 1, Z           " ~ LAST SYMBOL = FALSE ?
Y, S = LAST SYMBOL

```

" TRANSLATOR PROCEDURES NR. 4/5

	Y, A = 89	
	Y, SUBC (:MACRO2)	" MACRO2 (TBC, LAST SYMBOL)
	Y, GOTO (:NXT SBL)	
	A = 310	
	GOTO (:ERRORM)	" 34 INSTRUCTIONS
RELATN:	SUBC (:REL OP LS)	" RELATOPERATOR LAST SYMBOL ?
	N, GOTO (MC[-1])	
	S = 62	
	MC = S	" RELMACRO
	A = 0	
	SUBC (:MACRO)	" MACRO (STACK)
	SUBC (:NXT SBL)	
	SUBC (:S ARITHEXP)	
	A = MC[-1]	
	SUBC (:MACRO)	" MACRO (RELMACRO)
	GOTO (MC[-1])	" 11 INSTRUCTIONS
RST OF RL:	SUBC (:RST OF AE)	
	SUBC (:RELATN)	" RELATION ?
	Y, GOTOR (MC[-1])	
	A = 311	
	GOTO (:ERRORM)	" 5 INSTRUCTIONS
BL PR RST:	MC = S	" N
	SUBC (:FCT DES)	
	SUBC (:ARMETIC)	" ARITHMETIC ?
	N, SUBC (:AR OP LS)	" ~ ARITHOPERATOR LAST SYMBOL ?
	N, SUBC (:REL OP LS)	" ~ RELATOPERATOR LAST SYMBOL ?
	S = MC[-1]	" N
	N, GOTO (:BL NAME)	
	SUBC (:AR NAME)	
	GOTO (:RST OF RL)	" 9 INSTRUCTIONS
BL NAME:	SUBC (:NON BL)	" NONBOOLEAN ?
	Y, A = 312	
	Y, SUBC (:ERRORM)	
	SUBC (:SIMPLE)	" SIMPLE ?
	N, GOTOR (MC[-1])	
	MC = S	" N
	SUBC (:FORMAL)	" FORMAL ?
	Y, A = 99	" MACRO2 (DOS, N)
	N, A = 88	" MACRO2 (TBV, N)
	GOTO (:MCR-RET)	" 11 INSTRUCTIONS
AR BL EXP:	S = LAST SYMBOL	
	U, S = 94, Z	" LAST SYMBOL = IF ?
	N, GOTO (:S A BL EXP)	
	SUBC (:IFCLAUSE)	
	MC = S	" FUTURE1
	Y, SUBC (:NXT SBL)	
	N, A = 313	
	N, SUBC (:ERRORM)	
	SUBC (:S A BL EXP)	

" TRANSLATOR PROCEDURES NR. 5 CONTINUED

```

MC = A          " TYPE
A = 314
AR BL EXP[11]: S = LAST SYMBOL
S = 96, Z      " LAST SYMBOL = ELSE ?
N, SUBC (:ERRORM)
N, JUMP (15)
A = 102        " FUTURE2 = 0
SUBC (:MACRO2) " MACRO2 (JU, FUTURE2)
A = M[B-2]     " FUTURE1
M[B-2] = S     " FUTURE2
SUBC (:SUBSTT)
SUBC (:NXT SBL)
A = M[B-1]
U, A = 6, P    " TYPE=UN v TYPE=ARBO v TYPE=INTLAB ?
Y, JUMP (1)
U, A = 5, Z    " v TYPE = NONDES ?
A = :MT[5]    " BASE ADDRESS OF LIST
DO (MA)       " EXECUTE SELECTED INSTRUCTION
Y, M[B-1] = A  " REPLACE TYPE BY NEW TYPE
A = M[B-2]    " FUTURE2
SUBC (:SUBSTT)
A = MC[-1]    " TYPE
B = 1
GOTOR (MC[-1])
SUBC (:BOOLEXP)
SUBC (:STR EXP)
SUBC (:ARITHEXP)
SUBC (:N DS EXP)
SUBC (:DES EXP)
SUBC (:EXP)
SUBC (:AR BL EXP)
SUBC (:AR DS EXP) " 41 INSTRUCTIONS

S A BL EXP: S = LETTER LAST SYMBOL, Z
Y, SUBC (:IDF)
Y, SUBC (:SUBS VAR)
Y, GOTO (:AR BL RST)
S = LAST SYMBOL
U, S = 98, Z   " LAST SYMBOL = OPEN ?
N, JUMP (16)
SUBC (:NXT SBL)
SUBC (:AR BL EXP)
MC = A        " TYPE
A = 315
SUBC (:REQ CLOSE) " REQUIRE CLOSE AS LAST SYMBOL
A = MC[-1]
U, A = 8, Z    " TYPE = ARBO ?
Y, SUBC (:BOOL OP LS) " ^ BOOLOPERATOR LAST SYMBOL ?
N, A = 2, Z    " v TYPE = BO ?
Y, SUBC (:RST OF BE)
Y, JUMP (14)   " TYPE = BO
A = 2, Z      " TYPE = AR ?
N, SUBC (:AR OP LS) " v ARITHOPERATOR LAST SYMBOL ?
N, SUBC (:REL OP LS) " v RELATOPERATOR LAST SYMBOL ?
Y, GOTO (:RST OF ABE)
JUMP (13)     " TYPE = ARBO
U, S = 64, Z  " LAST SYMBOL = PLUS ?
N, S = 65, Z  " v LAST SYMBOL = MINUS ?

```

" TRANSLATOR PROCEDURES NR. 5/6

```

N, A = DIGIT LAST SYMBOL, Z " ~ DIGIT LAST SYMBOL ?
Y, SUBC (:S ARITHEXP)
Y, GOTO (:RST OF ABE)
U, S = 11, Z " LAST SYMBOL = NON ?
N, S = 51, Z " ~ LAST SYMBOL = TRUE ?
N, S = 1, Z " ~ LAST SYMBOL = FALSE ?
Y, SUBC (:S BOOLEXP)
Y, A = 2 " TYPE = BO
Y, GOTOR (MC[-1])
A = 316
SUBC (:ERRORM)
A = 8 " TYPE = ARBO
GOTOR (MC[-1]) " 38 INSTRUCTIONS

RST OF AE: SUBC (:NXT PRIM)
SUBC (:NXT FCTR)
GOTO (:NXT TERM) " 3 INSTRUCTIONS

RST OF BE: SUBC (:NXT BL SC)
SUBC (:NXT BL FC)
SUBC (:NXT BL TRM)
GOTO (:NXT IMP) " 4 INSTRUCTIONS

RST OF ABE: SUBC (:RST OF AE)
SUBC (:RELATN) " RELATION ?
Y, SUBC (:RST OF BE)
Y, A = 2 " TYPE = BO
N, A = 4 " TYPE = AR
GOTOR (MC[-1]) " 6 INSTRUCTIONS

AR BL RST: MC = S " N
SUBC (:FCT DES)
SUBC (:BOOLEAN) " BCOLEAN ?
N, SUBC (:BOOL OP LS) " ~ BCOLOPERATOR LAST SYMBOL ?
S = M[B-1] " N
Y, SUBC (:BL NAME)
Y, SUBC (:RST OF BE)
Y, A = 2 " TYPE = BO
Y, GOTO (:RETURN)
SUBC (:ARMETIC) " ARITHMETIC ?
N, SUBC (:AR OP LS) " ~ ARITHOPERATOR LAST SYMBOL ?
N, SUBC (:REL OP LS) " ~ RELATOPERATOR LAST SYMBOL ?
S = M[B-1] " N
Y, SUBC (:AR NAME)
Y, SUBC (:RST OF ABE)
Y, GOTO (:RETURN)
SUBC (:STRING) " STRING ?
N, SUBC (:DESINAL) " ~ DESIGNATIONAL ?
Y, A = 317
Y, SUBC (:ERRORM)
A = 99
SUBC (:MACRO2) " MACRO2 (DOS, N)
A = 8 " TYPE = ARBO
GOTO (:RETURN) " 24 INSTRUCTIONS

```

" TRANSLATOR PROCEDURES NR. 6 CONTINUED

```

STR EXP:      S = LAST SYMBOL
              U, S = 94, Z      " LAST SYMBOL = IF ?
              N, GOTO (:S STR EXP)
              SUBC (:IFCLAUSE)
              MC = S          " FUTURE1
              Y, SUBC (:NXT SBL)
              N, A = 318
              N, SUBC (:ERRORM)
              SUBC (:S STR EXP)
              SUBC (:FUTURE)
              Y, SUBC (:STR EXP)
              A = MC[-1]      " FUTURE2
              Y, GOTO (:SUBSTT)
              A = 319
              GOTO (:ERRORM)  " 15 INSTRUCTIONS

S STR EXP:    S = LETTER LAST SYMBOL, Z
              Y, SUBC (:IDF)
              Y, SUBC (:SUBS VAR)
              Y, GOTO (:ST NAME)
              S = LAST SYMBOL
              U, S = 98, Z      " LAST SYMBOL = OPEN ?
              Y, SUBC (:NXT SBL)
              Y, SUBC (:STR EXP)
              Y, A = 320
              Y, GOTO (:REQ CLOSE) " REQUIRE CLOSE AS LAST SYMBOL
              U, S = 102, Z      " LAST SYMBOL = QUOTE ?
              N, A = 321
              N, GOTO (:ERRORM)
              A = 28
              SUBC (:MACRO)      " MACRO (TCST)
              S = 0              " FUTURE = 0
              A = 102
              SUBC (:MACRO2)      " MACRO2 (JU, FUTURE)
              MC = S              " FUTURE
              SUBC (:CST STR)
              A = MC[-1]          " FUTURE
              GOTO (:SUBSTT)      " 22 INSTRUCTIONS

ST NAME:      SUBC (:NON STR)    " NONSTRING ?
              Y, A = 322
              Y, SUBC (:ERRORM)
              SUBC (:FCT DES)
              SUBC (:SIMPLE)      " SIMPLE ?
              N, GOTOR (MC[-1])    " N
              MC = S              " FORMAL ?
              SUBC (:FORMAL)
              Y, A = 99            " MACRO2 (DOS, N)
              N, A = 90            " MACRO2 (TSTV, N)
              GOTO (:MCR RET)      " 11 INSTRUCTIONS

DES EXP:      S = LAST SYMBOL
              U, S = 94, Z      " LAST SYMBOL = IF ?
              N, GOTO (:S DES EXP)
              SUBC (:IFCLAUSE)
              MC = S          " FUTURE1

```

" TRANSLATOR PROCEDURES NR. 6/7

```

      Y, SUBC (:NXT SBL)
      N, A = 323
      N, SUBC (:ERRORM)
      SUBC (:S DES EXP)
      SUBC (:FUTURE)
      Y, SUBC (:DES EXP)
      A = MC[-1]           " FUTURE2
      Y, GOTO (:SUBSTT)
      A = 324
      GOTO (:ERRORM)       " 15 INSTRUCTIONS

S DES EXP:      S = LETTER LAST SYMBOL, Z
      Y, SUBC (:IDF)
      Y, SUBC (:SUBS VAR)
      Y, GOTO (:DS NAME)
      S = LAST SYMBOL
      U, S = 98, Z         " LAST SYMBOL = OPEN ?
      Y, SUBC (:NXT SBL)
      Y, SUBC (:DES EXP)
      Y, A = 325
      Y, GOTO (:REQ CLOSE) " REQUIRE CLOSE AS LAST SYMBOL
      S = DIGIT LAST SYMBOL, Z
      N, A = 327
      N, GOTO (:ERRORM)
      SUBC (:UNS NUM)
      SUBC (:IN NM LST)   " IN NAME LIST ?
      Y, S = INT LAB
      Y, A = 91
      Y, GOTO (:MACRO2)   " MACRO2 (TLV, INTEGER LABEL)
      A = 326
      GOTO (:ERRORM)     " 20 INSTRUCTIONS

DS NAME:      SUBC (:NON DES) " NONDESIGNATIONAL ?
      Y, A = 328
      Y, SUBC (:ERRORM)
      SUBC (:SIMPLE)      " SIMPLE ?
      N, GOTOR (MC[-1])
      MC = S              " N
      SUBC (:FORMAL)      " FORMAL ?
      Y, A = 99           " MACRO2 (DOS, N)
      N, A = 91          " MACRO2 (TLV, N)
      GOTO (:MCR RET)     " 10 INSTRUCTIONS

AR DS EXP:    SUBC (:EXP)
      MC = A              " TYPE
      U, A = 2, Z         " TYPE = BO ?
      N, A = 3, Z        " ~ TYPE = ST ?
      Y, A = 329
      Y, SUBC (:ERRORM)
      A = MC[-1]
      U, A = 7, Z        " TYPE = UN ?
      Y, A = 9           " THEN NEW TYPE = INTLAB
      U, A = 5, Z        " TYPE = NONDES ?
      Y, A = 4           " THEN NEW TYPE = AR
      GOTOR (MC[-1])     " 12 INSTRUCTIONS

```

" TRANSLATOR PROCEDURES NR. 7/8

```

N DS EXP:      SUBC (:EXP)
                U, A = 6, Z          " TYPE = DES ?
                Y, MC = A
                Y, A = 330
                Y, SUBC (:ERRORM)
                Y, A = MC[-1]
                U, A = 7, Z          " TYPE = UN ?
                Y, A = 5             " THEN NEW TYPE = NONDES
                U, A = 9, Z          " TYPE = INTLAB ?
                Y, A = 4             " THEN NEW TYPE = AR
                GOTOR (MC[-1])       " 11 INSTRUCTIONS

EXP:            S = LAST SYMBOL
                U, S = 94, Z          " LAST SYMBOL = IF ?
                N, GOTO (:S EXP)
                SUBC (:IFCLAUSE)
                MC = S               " FUTURE1
                Y, SUBC (:NXT SBL)
                N, A = 331
                N, SUBC (:ERRORM)
                SUBC (:S EXP)
                MC = A               " TYPE
                A = 332
                GOTO (:AR BL EXP[11]) " 12 INSTRUCTIONS

S EXP:          S = LETTER LAST SYMBOL, Z
                Y, SUBC (:IDF)
                Y, SUBC (:SUBS VAR)
                Y, GOTO (:EXP RST)
                S = LAST SYMBOL
                U, S = 98, Z          " LAST SYMBOL = OPEN ?
                N, JUMP (20)
                SUBC (:NXT SBL)
                SUBC (:EXP)
                MC = A               " TYPE
                A = 333
                SUBC (:REQ CLOSE)    " REQUIRE CLOSE AS LAST SYMBOL
                A = - M[B-1]
                U, A = 5, Z          " TYPE = NONDES v TYPE = UN ?
                Y, SUBC (:BOOL OP LS) " ^ BOOLOPERATOR LAST SYMBOL ?
                N, A = 2, Z          " v TYPE = .BO ?
                Y, SUBC (:RST OF BE)
                Y, A = 2             " TYPE = BO
                Y, GOTO (:RETURN)
                SUBC (:OP LS)        " OPERATOR LAST SYMBOL ?
                A = MC[-1]           " TYPE
                N, GOTOR (MC[-1])
                U, A = 3, Z          " TYPE = ST ?
                Y, GOTOR (MC[-1])
                U, A = 6, Z          " TYPE = DES ?
                Y, GOTOR (MC[-1])
                GOTO (:RST OF ABE)
                U, S = DIGIT LAST SYMBOL, Z
                N, JUMP (11)
                SUBC (:UNS NUM)
                SUBC (:AR CBT)
                SUBC (:IN NM LST)    " IN NAME LIST ?

```

S EXP[30]:

" TRANSLATOR PROCEDURES NR. 8 CONTINUED

```

N, GOTO (:RST OF ABE)
SUBC (:OP LS)           " OPERATOR LAST SYMBOL ?
Y, GOTO (:RST OF ABE)
S = INT LAB
A = 91
SUBC (:MACRO2)          " MACRO2 (TLV, INTEGER LABEL)
A = 9                   " TYPE = INTLAB
GOTOR (MC[-1])
U, S = 64, Z           " LAST SYMBOL = PLUS ?
N, S = 65, Z           " ~ LAST SYMBOL = MINUS ?
Y, GOTO (:S A BL EXP)
U, S = 11, Z           " LAST SYMBOL = NON ?
N, S = 51, Z           " ~ LAST SYMBOL = TRUE ?
N, S = 1, Z            " ~ LAST SYMBOL = FALSE ?
Y, SUBC (:S BOOLEXP)
Y, A = 2               " TYPE = BO
Y, GOTOR (MC[-1])
U, S = 15, Z           " LAST SYMBOL = QUOTE ?
Y, SUBC (:S STR EXP)
Y, A = 3               " TYPE = ST
N, A = 334
N, SUBC (:ERRORM)
N, A = 7               " TYPE = UN
GOTOR (MC[-1])        " 56 INSTRUCTIONS

EXP RST: SUBC (:DESINAL) " DESIGNATIONAL ?
Y, SUBC (:DS NAME)
Y, A = 6               " TYPE = DES
Y, GOTOR (MC[-1])
SUBC (:STRING)         " STRING ?
Y, SUBC (:ST NAME)
Y, A = 3               " TYPE = ST
Y, GOTOR (MC[-1])
SUBC (:FCT DES)
MC = S                 " N
SUBC (:BOOLEAN)        " BOOLEAN ?
N, SUBC (:BOOL OP LS) " ~ BOOLOPERATOR LAST SYMBOL ?
S = M[B-1]             " N
Y, SUBC (:BL NAME)
Y, SUBC (:RST OF BE)
Y, A = 2               " TYPE = BO
Y, GOTO (:RETURN)
SUBC (:ARMETIC)        " ARITHMETIC ?
N, SUBC (:AR OP LS)    " ~ ARITHOPERATOR LAST SYMBOL ?
N, SUBC (:REL OP LS)   " ~ RELATOPERATOR LAST SYMBOL ?
S = M[B-1]             " N
Y, SUBC (:AR NAME)
Y, SUBC (:RST OF ABE)
Y, GOTO (:RETURN)
SUBC (:SIMPLE)         " SIMPLE ?
Y, A = 99
Y, SUBC (:MACRO2)      " MACRO2 (DOS, N)
S = MC[-1]             " N
SUBC (:UNKNOWN)        " UNKNOWN ?
Y, A = 7               " TYPE = UN
N, A = 5               " TYPE = NONDES
GOTOR (MC[-1])        " 32 INSTRUCTIONS

```

" TRANSLATOR PROCEDURES NR. 9

```

ASN STAT:          SUBC (:SUBS VAR)
                   A = LAST SYMBOL
U, A = 92, Z       " LAST SYMBOL = COLONEQUAL ?
N, A = 335
N, GOTO (:ERRORM)  " 5 INSTRUCTIONS

DST TYPE:          'BEGIN' LIST, AR, UN

                   SUBC (:REAL)          " REAL ?
                   A 'A' 7
                   MC = A                " TYPE
Y, SUBC (:RE ASN)
N, A + :LIST
N, DO (MA)          " EXECUTE SELECTED INSTRUCTION
                   A = MC[-1]           " TYPE
LIST:              GOTOR (MC[-1])
                   GOTO (:UN)
                   SUBC (:IN ASN)
                   SUBC (:BO ASN)
                   SUBC (:ST ASN)
                   GOTO (:AR)
                   GOTO (:UN)
                   GOTO (:UN)
AR:                GOTO (:UN)
                   SUBC (:AR ASN)
UN:                JUMP (1)
                   SUBC (:UN ASN)
                   B = 1
                   GOTOR (MC[-1])       " 21 INSTRUCTIONS

                   'END' DST TYPE

PREPARE:           SUBC (:FUNCTN)        " FUNCTION ?
Y, JUMP (5)
                   SUBC (:SIMPLE)        " SIMPLE ?
Y, SUBC (:FORMAL)  " ^ FORMAL ?
Y, A = 100
Y, SUBC (:MACRO2)   " MACRO2 (DOS2, N)
                   JUMP (8)
                   SUBC (:FORMAL)        " FORMAL ?
Y, A = 336
Y, JUMP (2)
                   SUBC (:OUT DEC)        " OUTSIDE DECLARATION ?
Y, A = 337
Y, SUBC (:ERRORM)
N, SUBC (:LOC POS)
N, M[B-2] = S      " N
                   SUBC (:NXT SBL)
                   S = LETTER LAST SYMBOL, Z
Y, SUBC (:IDF)
Y, SUBC (:SUBS VAR)
Y, A = LAST SYMBOL
                   GOTO (MC[-1])         " 21 INSTRUCTIONS

IN ASN:            MC = S                " ROUNDED = FALSE
                   MC = S                " N

```

" TRANSLATOR PROCEDURES NR. 9 CONTINUED

SUBC (:NON IN)	" NONINTEGER ?
Y, A = 338	
Y, SUBC (:ERRORM)	
SUBC (:PREPARE)	" PREPARE, IDENTIFIER READ ?
N, SUBC (:ARITHEXP)	
N, JUMP (6)	
A = 92, Z	" LAST SYMBOL = COLONEQUAL ?
N, SUBC (:FCT DES)	
N, SUBC (:AR NAME)	
N, SUBC (:RST OF AE)	
Y, SUBC (:IN ASN)	
Y, M[B-2] = A	" RCUNDED
S = M[B-1]	" N
SUBC (:SUBSCR)	" SUBSCRIPTED ?
N, JUMP (7)	
SUBC (:FORMAL)	" FORMAL ?
Y, A = 34	" MACRO (STFSU)
Y, JUMP (10)	
A = M[B-2], Z	" RCUNDED ?
Y, A = 31	" MACRO (SSTS1)
N, A = 30	" MACRO (STS1)
JUMP (6)	
SUBC (:FORMAL)	" FORMAL ?
Y, A = 101	" MACRO2 (DOS3, N)
Y, JUMP (3)	
A = M[B-2], Z	" RCUNDED ?
Y, A = 96	" MACRO2 (SST1, N)
N, A = 95	" MACRO2 (ST1, N)
SUBC (:MACRO2)	
S = MC[-1]	" N
SUBC (:FORMAL)	" FORMAL ?
A = MC[-1]	" RCUNDED
N, A = 0	
GOTOR (MC[-1])	" 36 INSTRUCTIONS
MC = S	" N
SUBC (:NON RE)	" NONREAL ?
Y, A = 339	
Y, SUBC (:ERRORM)	
SUBC (:PREPARE)	" PREPARE, IDENTIFIER READ ?
N, SUBC (:ARITHEXP)	
N, JUMP (5)	
A = 92, Z	" LAST SYMBOL = COLONEQUAL ?
Y, SUBC (:RE ASN)	
N, SUBC (:FCT DES)	
N, SUBC (:AR NAME)	
N, SUBC (:RST OF AE)	
S = M[B-1]	" N
SUBC (:SUBSCR)	" SUBSCRIPTED ?
N, JUMP (4)	
SUBC (:FORMAL)	" FORMAL ?
Y, A = 34	" MACRO (STFSU)
N, A = 29	" MACRO (STR)
GOTO (:MCR RET)	
SUBC (:FORMAL)	" FORMAL ?
Y, A = 101	" MACRO2 (DCS3, N)
N, A = 94	" MACRO2 (STR, N)
GOTO (:MCR RET)	" 23 INSTRUCTIONS

RE ASN:

" TRANSLATOR PROCEDURES NR. 10

BO ASN:	MC = S	" N
	SUBC (:NON BL)	" NONBOCLEAN ?
Y, A = 340		
Y, SUBC (:ERRORM)		
SUBC (:PREPARE)		" PREPARE, IDENTIFIER READ ?
N, SUBC (:BOOLEXP)		
N, JUMP (4)		
A = 92, Z		" LAST SYMBOL = COLONEQUAL ?
Y, SUBC (:BO ASN)		
N, SUBC (:BL PR RST)		
N, SUBC (:RST OF BE)		
S = M[B-1]		" N
SUBC (:SUBSCR)		" SUBSCRIPTED ?
Y, A = 32		
Y, GOTO (:MCR RET)		" MACRO (STSB)
SUBC (:FORMAL)		" FORMAL ?
Y, A = 101		" MACRO2 (DOS3, N)
N, A = 97		" MACRO2 (STB, N)
GOTO (:MCR RET)		" 19 INSTRUCTIONS
ST ASN:	MC = S	" N
	SUBC (:NON STR)	" NONSTRING ?
Y, A = 341		
Y, SUBC (:ERRORM)		
SUBC (:PREPARE)		" PREPARE, IDENTIFIER READ ?
N, SUBC (:STREXP)		
N, JUMP (3)		
A = 92, Z		" LAST SYMBOL = COLONEQUAL ?
Y, SUBC (:ST ASN)		
N, SUBC (:ST NAME)		
S = M[B-1]		" N
SUBC (:SUBSCR)		" SUBSCRIPTED ?
Y, A = 33		
Y, GOTO (:MCR RET)		" MACRO (STST)
SUBC (:FORMAL)		" FORMAL ?
Y, A = 101		" MACRO2 (DOS3, N)
N, A = 98		" MACRO2 (STST, N)
GOTO (:MCR RET)		" 18 INSTRUCTIONS
AR ASN:	MC = S	" N
	SUBC (:NON AR)	" NONARITHMETIC ?
Y, A = 342		
Y, SUBC (:ERRORM)		
SUBC (:PREPARE)		" PREPARE, IDENTIFIER READ ?
F = 4		
MC = G		" TYPE = AR
N, SUBC (:ARITHEXP)		
N, JUMP (10)		
A = 92, Z		" LAST SYMBOL = COLONEQUAL ?
N, SUBC (:FCT DES)		
N, SUBC (:AR NAME)		
N, SUBC (:RST OF AE)		
N, JUMP (5)		
SUBC (:NON AR)		" NONARITHMETIC ?
Y, A = 343		
Y, SUBC (:ERRORM)		
SUBC (:DST TYPE)		

" TRANSLATOR PROCEDURES NR. 10/11

	M[B-1] = A	" TYPE
	S = M[B-2]	" N
	SUBC (:SUBSCR)	" SUBSCRIPTED ?
AR ASN[22]:	Y, A = 34	" MACRO (STFSU)
AR ASN[23]:	N, A = 101	" MACRO2 (DCS3, N)
	SUBC (:MACRO2)	" TYPE
	A = MC[-1]	" 26 INSTRUCTIONS
	GOTO (:RETURN)	
UN ASN:	MC = S	" N
	SUBC (:NO TYPE)	" NCNTYPE ?
	Y, A = 344	
	Y, SUBC (:ERRORM)	" PREPARE, IDENTIFIER READ ?
	SUBC (:PREPARE)	
	N, SUBC (:N DS EXP)	" NCNTYPE ?
	N, JUMP (7)	
	SUBC (:NO TYPE)	
	Y, A = 345	" LAST SYMBOL = COLONEQUAL ?
	Y, SUBC (:ERRORM)	
	A = LAST SYMBOL	
	A = 92, Z	
	Y, SUBC (:DST TYPE)	" TYPE
	N, SUBC (:EXP RST)	" N
	MC = A	" SUBSCRIPTED ?
	S = M[B-2]	" MACRO2 (DCS3, N)
	SUBC (:SUBSCR)	" TYPE
N, GOTO (:AR ASN[22])	A = M[B-1]	" TYPE ≠ 80
U, A = 1, P		" ^ TYPE ≠ ST ?
U, A = 3, E		" MACRO (STFSU)
Y, A = 34		" MACRO (STSB/STST)
N, A + 30		" 24 INSTRUCTIONS
	GOTO (:AR ASN[23])	
FCT DES:	SUBC (:PROC)	" PROC ?
N, GOTOR (MC[-1])	SUBC (:NON FCT)	" NONFUNCTION ?
Y, A = 346		
Y, SUBC (:ERRORM)		" 6 INSTRUCTIONS
	GOTO (:PR CALL)	
PR STAT:	SUBC (:PROC)	" PROC ?
N, A = 347		
N, GOTO (:ERRORM)		
	SUBC (:PR CALL)	" IN LIBRARY ?
	SUBC (:IN LIBR)	
N, SUBC (:LINE1)		" FUNCTION ?
	SUBC (:FUNCTN)	" ^ STRING ?
Y, SUBC (:STRING)		" v FORMAL ?
N, SUBC (:FORMAL)		
N, GOTOR (MC[-1])		" MACRO (REJST)
A = 73		" 12 INSTRUCTIONS
	GOTO (:MACRO)	

" TRANSLATOR PROCEDURES NR. 11 CONTINUED

```

PR CALL:          SUBC (:OP LIKE)          " OPERATORLIKE ?
                  Y, GOTO (:PRCS OP)
                  SUBC (:LST LTH)
                  A = 0, P                  " NUMBER OF PARAMETERS > 0 ?
                  Y, GOTO (:PAR LIST)
PR CALL[5]:       MC = S                  " N
                  SUBC (:FORMAL)           " FORMAL ?
                  Y, A = 99
                  Y, GOTO (:MCR RET)        " MACRO2 (DCS, N)
                  SUBC (:IN LIBR)          " IN LIBRARY ?
                  Y, A = 109               " MACRO2 (ISUBJ, N)
                  N, A = 108               " MACRO2 (SUBJ, N)
                  GOTO (:MCR RET)          " 13 INSTRUCTIONS

ORD NUM:          SUBC (:FORMAL)           " FORMAL ?
                  Y, A = 15
                  Y, GOTOR (MC[-1])
                  A '*' 31                 " CHARACTER
                  U, A = 15, P             " PROC ?
                  N, JUMP (4)
                  U, A '*' 8, Z            " FUNCTION ?
                  Y, A + 8                 " THEN 24/25/26/27
                  N, A = 30                 " ELSE 30
                  GOTOR (MC[-1])
                  U, A = 14, Z             " SWITCH ?
                  Y, A = 12                 " THEN 12
                  U, A = 6, Z              " LABEL ?
                  Y, A = 14                 " THEN 14
                  GOTOR (MC[-1])           " 15 INSTRUCTIONS

```

" TRANSLATOR PROCEDURES NR. 12

PAR LIST: 'BEGIN' PAR LIST0, PAR LIST1, PAR LIST2, PAR LIST3,
PAR LIST4, PAR LIST5, PAR LIST6, PAR LIST7,
PAR LIST8, PAR LIST9, PAR LIST10, PAR LIST11,
PAR LIST12, PAR LIST13

PAR LIST0:

```

A + B
MA = B           " ADDRESS FIRST APD
MA[1] = B        " ADDRESS CURRENT APD
B = :MA[2]
MC = S           " N
MC = S           " F = N
F = 0
MC = F           " TYPE, FUTURE = 0
S = LAST SYMBOL
U, S = 98, Z     " LAST SYMBOL = OPEN ?
N, A = 362       " PL15
N, GOTO (:PAR LIST12)
SUBC (:NXT SBL)
SUBC (:ACT PAR)
M[B-2] = S       " TYPE
S = M[B-5]       " ADDRESS CURRENT APD
U, S = :MC[-6], Z " > ADDRESS LAST APD ?
Y, A = 359       " PL12
Y, GOTO (:PAR LIST10)
MS = A          " APD
S + 1           " ADDRESS NEXT APD
M[B-5] = S       " N
S = M[B-4]       " FORMAL ?
SUBC (:FORMAL)
Y, GOTO (:PAR LIST11)
S = M[B-3]       " F
SUBC (:NXT F I)
M[B-3] = S       " F = NEXT FORMAL IDENTIFIER
F + 0, Z        " SIMPLE IDENTIFIER ?
N, GOTO (:PAR LIST4)
SUBC (:SUBSCR)
" F SUBSCRIPTED ?
N, GOTO (:PAR LIST1)
S = M[B-2]       " TYPE
SUBC (:NON SUBS) " NCNSUBSCRIPTED ?
Y, A = 348       " PL1
Y, SUBC (:ERRORM)
A = M[B-3]       " F
SUBC (:CHCK TP)
A = M[B-3]       " F
SUBC (:CHCK LL)
GOTO (:PAR LIST11)
SUBC (:PROC)     " F PROC ?
N, GOTO (:PAR LIST3)
S = M[B-2]       " TYPE
SUBC (:NON PROC) " NCNPRCC ?
Y, A = 349       " PL2
Y, SUBC (:ERRORM)
A = M[B-3]       " F
SUBC (:CHCK LL)
S = M[B-3]       " F
SUBC (:FUNCTN)   " FUNCTION ?
N, GOTO (:PAR LIST11)
S = M[B-2]       " TYPE

```

PAR LIST1:

100470 -

1

239

SUBC (:NON FCT)

" NONFUNCTION ?

" TRANSLATOR PROCEDURES NR. 12 CONTINUED

	Y, A = 350	" PL3
PAR LIST2:	Y, SUBC (:ERRORM) A = M[B-3]	" F
	SUBC (:CHCK TP) GOTO (:PAR LIST11)	
PAR LIST3:	SUBC (:SIMPLE1) N, GOTO (:PAR LIST11)	" F SIMPLE1 ?
	S = M[B-2]	" TYPE
	SUBC (:NON SMPL)	" NCNSIMPLE ?
	Y, A = 351 GOTO (:PAR LIST2)	" PL4
PAR LIST4:	SUBC (:SUBSCR)	" F SUBSCRIPTED ?
	N, SUBC (:PROC)	" ~ PROC ?
	Y, A = 352 Y, SUBC (:ERRORM)	" PL5
	SUBC (:ASS TO)	" ASSIGNED TO F ?
	Y, A = M[B-5]	" ADDRESS CURRENT APD
	Y, A = MA[-1]	" APD
	Y, SUBC (:NON ASBL)	" NCNASSIGNABLE ?
	Y, A = 353 Y, SUBC (:ERRORM)	" PL6
	SUBC (:ARMETIC)	" F ARITHMETIC ?
	N, GOTO (:PAR LIST5)	
	A = 354	" PL7
	S = M[B-2]	" TYPE
	U, S '*' = 3, Z	" BC ~ ST ?
	N, S = 6, Z	" ~ DES ?
PAR LIST5:	GOTO (:PAR LIST10)	
	SUBC (:BOOLEAN)	" F BOOLEAN ?
	N, GOTO (:PAR LIST7)	
	A = 355	" PL8
	S = - M[B-2]	" TYPE
	U, S + 2, Z	" BC ?
PAR LIST6:	N, S '*' 5, Z	" ~ NONDES ~ UN ?
	N, SUBC (:ERRORM)	
	GOTO (:PAR LIST11)	
PAR LIST7:	SUBC (:STRING)	" F STRING ?
	N, GOTO (:PAR LIST8)	
	A = 356	" PL9
	S = - M[B-2]	" TYPE
	U, S + 3, Z	" ST ?
	GOTO (:PAR LIST6)	
PAR LIST8:	SUBC (:DESINAL)	" F DESIGNATIONAL ?
	N, GOTO (:PAR LIST9)	
	A = 357	" PL10
	S = - M[B-2]	" TYPE
	U, S + 6, P	" BC ~ ST ~ AR ~ NONDES ?
	GOTO (:PAR LIST10)	
PAR LIST9:	SUBC (:ARBOST)	" F ARBCST ?
	Y, S = M[B-2]	" TYPE
	Y, S = 6, Z	" DES ?
	Y, A = 358	" PL11
PAR LIST10:	Y, SUBC (:ERRORM)	
PAR LIST11:	S = LAST SYMBOL	
	U, S = 87, Z	" LAST SYMBOL = COMMA ?
	Y, GOTO (:PAR LIST0)	
	U, S = 99, Z	" LAST SYMBOL = CLOSE ?
	N, A = 361	" PL14

100470 -

1

241

N, GOTO (:PAR LIST12)

" TRANSLATOR PROCEDURES NR. 12/13

```

      SUBC (:NXT SBL)
      S = M[B-5]          " ADDRESS CURRENT APD =
      U, S = :MC[-6], Z   " = ADDRESS LAST APD + 1 ?
      N, A = 360          " PL13
PAR LIST12: N, SUBC (:ERRORM)
      A = M[B-1], Z      " FUTURE = 0 ?
      N, SUBC (:SUBSTT)
      S = M[B-4]         " N
      SUBC (:PR CALL[5]) " MACRO2 (DOS/ISUBJ/SUBJ, N)
      S = M[B-6]         " ADDRESS FIRST APD
PAR LIST13: M[B-2] = S   " ADDRESS NEXT APD
      U, S = M[B-5], Z   " = ADDRESS CURRENT APD ?
      Y, S = M[B-4]      " N
      Y, B = M[B-6]
      Y, GOTOR (MC[-1])
      A = -MS
      S = -A             " PARAMETER INTO S
      U, A'=' D18, Z
      N, RUA(20)
      N, A + 6, Z
      N, A + 1, Z
      N, A + 6, Z
      SUBC (:MACRO3)
      S = M[B-2]         " ADDRESS NEXT APD
      S + 1
      GOTO (:PAR LIST13) " 139 INSTRUCTIONS

'END' PAR LIST

ACT PAR: 'BEGIN' ACT PAR0, ACT PAR1, ACT PAR2, ACT PAR3, ACT PAR4,
          ACT PAR5, ACT PAR6, ACT PAR7, ACT PAR8, ACT PAR9,
          ACT PAR10, ACT PAR11, ACT PAR12, ACT PAR13,
          ACT PAR14, ACT PAR15, ACT PAR16

      SUBC (:ORD CNT)
      A = M[B-2], Z      " FUTURE = 0 ?
      Y, S + 1
      MC = S             " BEGIN ADDRESS
      S = LETTER LAST SYMBOL, Z
      N, GOTO (:ACT PAR5)
      SUBC (:IDF)
      MC = S             " N
      A = LAST SYMBOL
      U, A = 87, Z       " LAST SYMBOL = COMMA ?
      N, A = 99, Z       " ~ LAST SYMBOL = CLOSE ?
      N, GOTO (:ACT PAR2)
      SUBC (:PROC)
      GOTO (:ACT PAR0)   " PROC ?
      SUBC (:FORMAL)
      GOTO (:ACT PAR0)   " FORMAL ?
      Y, GOTO (:ACT PAR0)
      S = M[B-4], Z      " FUTURE = 0 ?
      Y, A = 102
      Y, SUBC (:MACRO2)   " MACRO2 (JU, FUTURE)
      Y, M[B-4] = S       " FUTURE
      A = 36
      SUBC (:MACRO)
      S = M[B-1]          " MACRO (TFD)
      SUBC (:IN LIBR)     " N
                          " IN LIBRARY ?

```

100470 -

1

243

```
Y, A = 105          " MACRO2 (IJU1, N)
N, A = 103          " MACRO2 (JU1, N)
  SUBC (:MACRO2)
  GOTO (:ACT PAR1)
ACT PAR0:          SUBC (:DYNAMIC)    " DYNAMIC ?
```

" TRANSLATOR PROCEDURES NR. 13 CONTINUED

```

      SUBC (:ADDRSS)
Y, A + D18
      M[B-2] = A
ACT PAR1: S = MC[-1]           " N
      SUBC (:ORD NUM)
      LUA (20)
      A + MC[-1]           " APD
      F = 0               " SIMPLE IDENTIFIER = TRUE
ACT PAR2: GOTOR (MC[-1])
      SUBC (:START ISR)
      S = M[B-1]           " N
      SUBC (:SUBSCR)       " SUBSCRIPTED ?
Y, SUBC (:ADR DSCR)
      A = LAST SYMBOL
U, A = 87, Z             " LAST SYMBOL = COMMA ?
N, A = 99, Z             " ~ LAST SYMBOL = CLOSE ?
N, GOTO (:ACT PAR3)
      SUBC (:DESINAL)       " DESIGNATIONAL ?
Y, GOTO (:ACT PAR3)
      SUBC (:UNKNOWN)       " UNKNOWN ?
Y, A = 37
Y, SUBC (:MACRO)         " MACRO (SAS)
      A = 127
      S = - DIMENSION
      LUS (1)
      SUBC (:MACRO2)       " MACRO2 (EXITSV, - 2 * DIMENSION)
      SUBC (:ORD CNT)
      MC = S
      S = M[B-2]           " N
      SUBC (:BOOLEAN)       " BCOLEAN ?
N, SUBC (:STRING)        " ~ STRING ?
Y, JUMP (3)
      SUBC (:FORMAL)       " FORMAL ?
Y, A = 16
N, A = '*' 3             " TYPE
      A + 16
      LUA (20)
      M[B-1] + A           " APD
      SUBC (:ARMETIC)       " ARITHMETIC ?
Y, A = 4                 " AR
N, SUBC (:INTEGER)       " ISOLATE TYPE BITS
      M[B-2] = A           " TYPE
      A = 108
      S = - M[B-3]         " BEGIN ADDRESS
      SUBC (:MACRO2)       " MACRO2 (SUBJ, - BEGIN ADDRESS)
      A = M[B-1]           " APD
      RUA (20)
U, A = 32, Z
Y, A = 44
N, A + 24
      SUBC (:MACRO)         " MACRO (TASR/TASI/TASB/TASST/TASU)
      A = 38
      SUBC (:MACRO)       " MACRO (DECS)
      A = 108
      S = - M[B-3]         " BEGIN ADDRESS
      SUBC (:MACRO2)       " MACRO2 (SUBJ, - BEGIN ADDRESS)
      A = 39
      SUBC (:MACRO)       " MACRO (FAD)

```

100470 -

1

245

A = MC[-1]

PD

" TRANSLATOR PROCEDURES NR. 13/14

```

      S = MC[-1]           " TYPE
      G = MC[-1]           " SIMPLE IDENTIFIER = FALSE
      GOTOR (MC[-1])
ACT PAR3:  SUBC (:SUBSCR)    " SUBSCRIPTED ?
      Y, SUBC (:EVALON)
      SUBC (:EXP RST)
ACT PAR4:  M[B-1] = A       " TYPE
      A = 45
      SUBC (:MACRO)        " MACRO (EXITIS)
      S = MC[-1]          " TYPE
      A = S
      A + :MASK
      A = MA
      A + MC[-1]          " BEGIN ADDRESS
      F = 1               " SIMPLE IDENTIFIER = FALSE
      GOTOR (MC[-1])
ACT PAR5:  S = DIGIT LAST SYMBOL, Z
      N, GOTO (:ACT PAR8)
      SUBC (:UNS NUM)
      S = LAST SYMBOL
      U, S = 87, Z        " LAST SYMBOL = COMMA ?
      N, S = 99, Z        " ~ LAST SYMBOL = CLOSE ?
      N, GOTO (:ACT PAR7)
      SUBC (:IN NM LST)    " IN NAME LIST ?
      Y, GOTO (:ACT PAR7)
ACT PAR6:  SUBC (:NUM DSCR)
      S = 4               " TYPE = AR
      G = MC[-1]          " SIMPLE IDENTIFIER = FALSE
      GOTOR (MC[-1])
ACT PAR7:  B + 1           " RESERVE WORD FOR TYPE
      SUBC (:START ISR)
      SUBC (:S EXP[30])
      GOTO (:ACT PAR4)
ACT PAR8:  S = LAST SYMBOL
      U, S = 64, Z        " LAST SYMBOL = PLUS ?
      N, GOTO (:ACT PAR10)
      SUBC (:NXT SBL)
      S = DIGIT LAST SYMBOL, Z
      N, GOTO (:ACT PAR9)
      SUBC (:UNS NUM)
      S = LAST SYMBOL
      U, S = 87, Z        " LAST SYMBOL = COMMA ?
      N, S = 99, Z        " ~ LAST SYMBOL = CLOSE ?
      Y, GOTO (:ACT PAR6)
      GOTO (:ACT PAR7)
ACT PAR9:  B + 1           " RESERVE WORD FOR TYPE
      SUBC (:START ISR)
      SUBC (:AR BL EXP)
      GOTO (:ACT PAR4)
ACT PAR10: U, S = 65, Z    " LAST SYMBOL = MINUS ?
      N, GOTO (:ACT PAR14)
      SUBC (:NXT SBL)
      S = DIGIT LAST SYMBOL, Z
      N, GOTO (:ACT PAR13)
      SUBC (:UNS NUM)
      S = LAST SYMBOL
      U, S = 87, Z        " LAST SYMBOL = COMMA ?
      N, S = 99, Z        " ~ LAST SYMBOL = CLOSE ?

```

100470 -

1

247

Y, S = SMALL, Z

" ^ SMALL ?

" TRANSLATOR PROCEDURES NR. 14 CONTINUED

```

      N, GOTO (:ACT PAR11)
      A = 13
      LUA (20)
      A + VALUE OF CONSTANT[1]
      GOTO (:ACT PAR6[1])
ACT PAR11:  B + 1          " RESERVE WORD FOR TYPE
            SUBC (:START ISR)
            SUBC (:AR CST)
            SUBC (:NXT PRIM)
            SUBC (:NXT FCTR)
ACT PAR12:  A = 1
            SUBC (:MACRO)          " MACRO (NEG)
            SUBC (:RST OF ABE)
            GOTO (:ACT PAR4)
ACT PAR13:  B + 1          " RESERVE WORD FOR TYPE
            SUBC (:START ISR)
            SUBC (:TERM)
            GOTO (:ACT PAR12)
ACT PAR14:  U, S = 115, P      " LAST SYMBOL ≠ TRUE
            U, S = 117, E      " ^ LAST SYMBOL ≠ FALSE ?
            Y, GOTO (:ACT PAR16)
            MC = S             " N = LAST SYMBOL
            SUBC (:NXT SBL)
            U, S = 87, Z        " LAST SYMBOL = COMMA ?
            N, S = 99, Z        " ~ LAST SYMBOL = CLOSE ?
            N, GOTO (:ACT PAR15)
            A = 6
            LUA (20)
            A + MC[-1]          " N
            A = 116
            S = 2               " TYPE = 80
            G = MC[-1]          " SIMPLE IDENTIFIER = FALSE
            GOTOR (MC[-1])
ACT PAR15:  SUBC (:START ISR)
            A = 89
            S = M[B-1]
            SUBC (:MACRO2)       " MACRO2 (TBC, N)
            SUBC (:RST OF BE)    " TYPE = 80
            A = 2
            GOTO (:ACT PAR4)
ACT PAR16:  B + 1          " RESERVE WORD FOR TYPE
            SUBC (:START ISR)
            SUBC (:EXP)
            GOTO (:ACT PAR4)      " 191 INSTRUCTIONS

      'END' ACT PAR

START ISR:  S = M[B-5], Z      " FUTURE = 0 ?
            Y, A = 102
            Y, SUBC (:MACRO2)    " MACRO2 (JU, FUTURE)
            Y, M[B-5] = S        " FUTURE
            A = 35               " MACRO (ENTRIS)
            GOTO (:MACRO)        " 6 INSTRUCTIONS

NUM DSCR:   S = SMALL, Z      " SMALL ?
            Y, A = 7
            Y, JUMP (3)

```

" TRANSLATOR PROCEDURES NR. 14/15

```

      S = REAL NUMBER, Z      " REAL NUMBER ?
N, A = 5
Y, A = 4, Z                  " (CONDITION NO)
  LUA (20)
Y, A + VALUE OF CONSTANT[1]
N, A + ADRS OF CST
  GOTOR (MC[-1])              " 10 INSTRUCTIONS

PRCS OP:      MC = S          " N
              A = 0
              MC = A          " COUNT = 0
              S = LAST SYMBOL
U, S = 98, Z          " LAST SYMBOL = OPEN ?
N, JUMP (11)
PRCS OP[6]:   SUBC (:NXT SBL)
              SUBC (:ARITHEXP)
              A = 1
              M[B-1] + A      " COUNT = COUNT + 1
              S = LAST SYMBOL
U, S = 87, Z          " LAST SYMBOL = COMMA ?
Y, A = 0
Y, SUBC (:MACRO)        " MACRO (STACK)
Y, GOTO (:PRCS OP[6])
              A = 361
              SUBC (:REQ CLOSE[1]) " REQUIRE CLOSE AS LAST SYMBOL
              S = M[B-2]        " N
              SUBC (:LST LTH)
              A = MC[-1], Z      " LIST LENGTH = COUNT ?
N, A = 363
N, SUBC (:ERRORM)
              A = MS[-2]        " MACRO (OPERATOR MACRO (N))
              GOTO (:MCR RET)    " 24 INSTRUCTIONS

NON ASBL:     RUA (20)          " RANK
U, A = 14, P
U, A = 15, E          " RANK + 15 ?
Y, A = '15
Y, A = 3, P          " ^ RANK = RANK + 16 * 16 > 3 ?
              GOTO (MC[-1])      " 6 INSTRUCTIONS

LINE:         A = LNC
              A = LAST LNC, Z    " LNC = LAST LNC ?
LINE1:        N, A = - WANTED, P  " v = WANTED ?
Y, GOTOR (MC[-1])
              MC = S
              S = LNC
              LAST LNC = S      " LAST LNC = LNC
              A = 147           " MACRO2 (LNC, LNC)
              GOTO (:MCR RET)    " 9 INSTRUCTIONS

STATMNT:      S = 1            " IF STATEMENT FORBIDDEN = FALSE
              JUMP (1)          " 2 INSTRUCTIONS

```

" TRANSLATOR PROCEDURES NR. 15 CONTINUED

UNC STAT: S = 0 " IFSTATEMENT FORBIDDEN = TRUE
 IFSTAT FRB = S " 2 INSTRUCTIONS

STAT: S = LINE COUNTER
 LNC = S " LNC = LINE COUNTER
 S = LETTER LAST SYMBOL, Z
 N, JUMP (10)
 SUBC (:IDF)
 SUBC (:DESINAL) " DESIGNATIONAL ?
 STAT[6]: Y, SUBC (:LAB DEC)
 Y, GOTO (:STAT)
 SUBC (:LINE)
 SUBC (:SUBSCR) " SUBSCRIPTED ?
 N, A = LAST SYMBOL
 N, A = 92, Z " ~ LAST SYMBOL = COLONEQUAL ?
 Y, GOTO (:ASN STAT)
 GOTO (:PR STAT)
 S = DIGIT LAST SYMBOL, Z
 N, JUMP (6)
 SUBC (:UNS NUM)
 SUBC (:IN NM LST) " IN NAME LIST ?
 Y, S = INT LAB
 Y, GOTO (:STAT[6])
 A = 364
 GOTO (:ERRORM)
 SUBC (:LINE)
 S = LAST SYMBOL
 U, S = 81, Z " LAST SYMBOL = GOTO ?
 Y, GOTO (:GOTO STAT)
 U, S = 104, Z " LAST SYMBOL = BEGIN ?
 N, JUMP (5)
 SUBC (:NXT SBL)
 SUBC (:DECL LS) " DECLARATOR LAST SYMBOL ?
 Y, SUBC (:BLOCK)
 N, SUBC (:CMP TL)
 GOTO (:NXT SBL)
 U, S = 94, Z " LAST SYMBOL = IF ?
 N, JUMP (4)
 S = IFSTAT FRB, Z " IFSTATEMENT FORBIDDEN ?
 Y, A = 365
 Y, SUBC (:ERRORM)
 GOTO (:IF STAT)
 U, S = 82, Z " LAST SYMBOL = FOR ?
 Y, SUBC (:FOR STAT)
 Y, S = LAST SYMBOL
 Y, S = 96, Z " LAST SYMBOL = ELSE ?
 N, GOTOR (MC[-1])
 A = 366
 GOTO (:ERRORM) " 46 INSTRUCTIONS

GOTC STAT: SUBC (:NXT SBL)
 S = LETTER LAST SYMBOL, Z
 N, SUBC (:DES EXP)
 N, JUMP (6)
 SUBC (:IDF)
 SUBC (:SUBS VAR)
 SUBC (:LOC LAB) " LOCAL LABEL ?

" TRANSLATOR PROCEDURES NR. 15/16

```

                                Y, SUBC (:TEST FC)
                                Y, A = 102                                " MACRO2 (JU, N)
                                N, SUBC (:DS NAME)
                                N, A = 74                                " MACRO (JUA)
                                GOTO (:MACRO)                            " 12 INSTRUCTIONS
                                A = 802
                                SUBC (:ERRORM)
CMP TL3CM8:
                                SUBC (:STATMNT)
CMP TL3MIN1:                   S = LAST SYMBOL
CMP TL3:                       U, S = 91, Z                                " LAST SYMBOL = SEMICOLON ?
CMP TL:                         Y, SUBC (:NXT SBL)
                                Y, GOTO (:CMP TL)
                                U, S = 105, Z                                " LAST SYMBOL = END ?
                                Y, GOTOR (MC[-1])
                                A = 367
                                SUBC (:ERRORM)
                                A = :STATMNT
                                SUBC (:SKIP RST OF STAT)
                                GOTO (:CMP TL[1])                            " 14 INSTRUCTIONS

IF STAT:                       SUBC (:IFCLAUSE)
                                MC = S                                " FUTURE1
                                Y, SUBC (:NXT SBL)
                                N, A = 368
                                N, SUBC (:ERRORM)
                                S = LINE COUNTER
                                MC = S                                " SAVE LINE COUNTER
                                SUBC (:UNC STAT)
                                S = MC[-1]                                " LAST LNC = SAVED LINE COUNTER
                                SUBC (:FUTURE)
                                Y, S = LINE COUNTER
                                Y, MC = S                                " SAVE LINE COUNTER
                                Y, SUBC (:STATMNT)
                                Y, S = MC[-1]
                                Y, LAST LNC = S                            " LAST LNC = SAVED LINE COUNTER
                                A = MC[-1]                                " FUTURE1 OR FUTURE2
                                GOTO (:SUBSTT)                            " 18 INSTRUCTIONS

FOR STAT:                      G = LINE COUNTER
                                MC = F                                " SAVE LINE COUNTER
                                S = 0
                                LO = S                                " LO = 0
                                SUBC (:NXT SBL)
                                SUBC (:FOR LST)
                                A = 102
                                S = 0
                                SUBC (:MACRO2)                            " FUTURE = 0
                                M[B-2] = S                                " MACRO2 (JU, FUTURE)
                                A = LO, Z                                " FUTURE
                                N, SUBC (:SUBSTT)
                                S = LAST SYMBOL
                                U, S = 86, Z                                " LAST SYMBOL = DO ?
                                Y, SUBC (:NXT SBL)
                                N, A = 369
                                N, SUBC (:ERRORM)
                                A = 8192

```

" TRANSLATOR PROCEDURES NR. 16 CONTINUED

```

      SUBC (:INCR STS)
      A = 1
      FOR CNT + A      " FORCOUNT = FORCOUNT + 1
      SUBC (:STATMNT)
      A = - 8192
      SUBC (:INCR STS)
      A = 1
      FOR CNT - A      " FORCOUNT = FORCOUNT - 1
      S = MC[-1]
      U, S = LNC, Z      " LNC = SAVED LINE COUNTER ?
      N, LNC = S      " ELSE LNC = SAVED LINE COUNTER
      N, SUBC (:LINE1)
      SUBC (:STATUS)
      A = 104
      SUBC (:MACRO2)      " MACRO2 (IJU, STATUS)
      A = MC[-1]      " FUTURE
      GOTO (:SUBSTT)      " 35 INSTRUCTIONS

STORE PR:      S = CNTLD VAR      " CONTROLLED VARIABLE
      SUBC (:SUBSCR)      " SUBSCRIPTED ?
      Y, A = 108
      Y, S = - L2      " MACRO2 (SUBJ, - L2)
      Y, GOTO (:MACRO2)
      SUBC (:FORMAL)      " FORMAL ?
      N, GOTOR (MC[-1])
      A = 100      " MACRO2 (DOS2, CONTROLLED VARIABLE)
      GOTO (:MACRO2)      " 9 INSTRUCTIONS

STORE MCR:      S = CNTLD VAR      " CONTROLLED VARIABLE
      SUBC (:SUBSCR)      " SUBSCRIPTED ?
      N, JUMP (8)
      SUBC (:FORMAL)      " FORMAL ?
      Y, A = 34      " MACRO (STFSU)
      N, A '*' 1
      N, A + 29      " MACRO (STSR/STSI)
      SUBC (:MACRO)
      A = 110
      S = 2
      GOTO (:MACRO2)      " MACRO2 (DECB, 2)
      SUBC (:FORMAL)      " FORMAL ?
      Y, A = 101      " MACRO2 (DOS3, CONTROLLED VARIABLE)
      N, A '*' 1
      N, A + 94      " MACRO2 (STR/STI, CONTROLLED VARIABLE)
      GOTO (:MACRO2)      " 16 INSTRUCTIONS

TAKE MCR:      S = CNTLD VAR      " CONTROLLED VARIABLE
      SUBC (:SUBSCR)      " SUBSCRIPTED ?
      N, GOTO (:AR NAME)
      A = 108
      S = - L1      " MACRO2 (SUBJ, - L1)
      GOTO (:MACRO2)      " 6 INSTRUCTIONS

```

" TRANSLATOR PROCEDURES NR. 17

FOR LST: 'BEGIN' FOR LST0, FOR LST1, FOR LST2, FOR LST3, FOR LST4

```

S = LETTER LAST SYMBOL, Z
N, A = 375
N, GOTO (:ERRORM)
SUBC (:IDF)
CNTLD VAR = S " CONTROLLED VARIABLE
SUBC (:NON AR) " NONARITHMETIC ?
Y, A = 370
Y, SUBC (:ERRORM)
SUBC (:SUBSCR) " CONTROLLED VARIABLE SUBSCRIPTED ?
N, GOTO (:FOR LST0)
A = 102
S = 0 " L3 = 0
SUBC (:MACRO2) " MACRO2 (JU, L3)
L3 = S " L3
SUBC (:ORD CNT)
L4 = S " L4 = ORDER COUNTER
S = CNTLD VAR " CONTROLLED VARIABLE
SUBC (:ADR DSCR)
A = 127
S = - DIMENSION
LUS (1)
S + 1
SUBC (:MACRO2) " MACRO2 (EXITSV, 1 - 2 * DIMENSION)
SUBC (:ORD CNT)
L1 = S " L1 = ORDER COUNTER
A = 108
S = - L4
SUBC (:MACRO2) " MACRO2 (SUBJ, - L4)
S = CNTLD VAR " CONTROLLED VARIABLE
SUBC (:FORMAL) " FORMAL ?
Y, A = 49 " MACRO (TSCVU)
N, A '1' 1
N, A + 47 " MACRO (TRSCV/TISCV)
SUBC (:MACRO)
SUBC (:ORD CNT)
L2 = S " L2 = ORDER COUNTER
A = 108
S = - L4
SUBC (:MACRO2) " MACRO2 (SUBJ, - L4)
A = 46
SUBC (:MACRO) " MACRO (FADCV)
A = L3
SUBC (:SUBSTT)
JUMP (3)
SUBC (:FUNCTN) " FUNCTION ?
Y, A = 371
Y, SUBC (:ERRORM)
S = LAST SYMBOL
U, S = 92, Z " LAST SYMBOL = COLONEQUAL ?
N, A = 372
N, SUBC (:ERRORM)
FOR LST1: SUBC (:ORD CNT)
L3 = S " L3 = ORDER COUNTER
A = 87
S = 0
SUBC (:MACRO2) " MACRO2 (TSIC, 0)

```

100470 - 1

254

SUBC (:STATUS)

" TRANSLATOR PROCEDURES NR. 17 CONTINUED

```

      A = 96
      SUBC (:MACRO2)           " MACRO2 (SST1, STATUS)
      SUBC (:ORD CNT)
      L4 = S                   " L4 = ORDER COUNTER
      SUBC (:STORE PR)
      SUBC (:NXT SBL)
      SUBC (:ARITHEXP)
      S = LAST SYMBOL
U, S = 87, Z                   " LAST SYMBOL = COMMA ?
N, S = 86, Z                   " ~ LAST SYMBOL = DO ?
N, GOTO (:FOR LST2)
      SUBC (:STORE MCR)
      A = 102
      S = L0
      SUBC (:MACRO2)           " MACRO2 (JU, L0)
      L0 = S                   " L0
      A = L3
      SUBC (:SUBSTT)
      GOTO (:FOR LST4)
FOR LST2:
N, GOTO (:FOR LST3)           " LAST SYMBOL = WHILE ?
      SUBC (:STORE MCR)
      SUBC (:NXT SBL)
      SUBC (:BOOLEXP)
      A = 107
      S = L0
      SUBC (:MACRO2)           " MACRO2 (YCOJU, L0)
      L0 = S                   " L0
      A = L3
      S = L4
      SUBC (:SUBST2)
      GOTO (:FOR LST4)
FOR LST3:
      S = 25, Z               " LAST SYMBOL = STEP ?
N, A = 374
N, SUBC (:ERRORM)
N, GOTO (:FOR LST4)
      A = 102
      SUBC (:MACRO2)           " MACRO2 (JU, L5)
      L5 = S                   " L5
      SUBC (:ORD CNT)
      L4 = S                   " L4 = ORDER COUNTER
      CMPLTD = - S             " COMPLICATED = FALSE
      SUBC (:NXT SBL)
      SUBC (:ARITHEXP)
      SUBC (:ORD CNT)
      S = L4
      S = 1, P
N, S = CMPLTD, P              " ORDER COUNTER > L4 + 1 ?
      CMPL ST = S              " ~ COMPLICATED ?
      Y, A = 50                " COMPLEX STEP ELEMENT
Y, SUBC (:MACRO)              " MACRO (EXIT)
      A = L3
      SUBC (:SUBSTT)
      SUBC (:STORE PR)
      SUBC (:TAKE MCR)
      A = 0
      SUBC (:MACRO)            " MACRO (STACK)
      S = CMPL ST, P           " COMPLEX STEP ELEMENT ?

```

100470 -

1

256

Y, A = 108

" TRANSLATOR PROCEDURES NR. 17/18

```

Y, S = - L4           " MACRO2 (SUBJ, - L4)
N, A = 111
N, S = L4             " MACRO2 (DO, L4)
  SUBC (:MACRO2)
  A = 2
  SUBC (:MACRO)        " MACRO (ADD)
  A = L5
  SUBC (:SUBSTT)
  SUBC (:STORE MCR)
  S = CNTLD VAR       " CONTROLLED VARIABLE
  SUBC (:SUBSCR)      " SUBSCRIPTED ?
N, SUBC (:FORMAL)     " ~ FORMAL ?
Y, SUBC (:TAKE MCR)
  A = 0
  SUBC (:MACRO)        " MACRO (STACK)
  S = LAST SYMBOL
  U, S = 84, Z        " LAST SYMBOL = UNTIL ?
Y, SUBC (:NXT SBL)
Y, SUBC (:ARITHEXP)
N, A = 373
N, SUBC (:ERRORM)
  A = 51
  SUBC (:MACRO)        " MACRO (TEST1)
  S = CMPL ST, P      " COMPLEX STEP ELEMENT ?
Y, A = 108
Y, S = - L4           " MACRO2 (SUBJ, - L4)
N, A = 111
N, S = L4             " MACRO2 (DO, L4)
  SUBC (:MACRO2)
  A = 52
  SUBC (:MACRO)        " MACRO (TEST2)
  A = 107
  S = L0
  SUBC (:MACRO2)       " MACRO2 (YCOJU, L0)
  L0 = S              " L0
FOR LST4:
  S = LAST SYMBOL
  U, S = 87, Z        " LAST SYMBOL = COMMA ?
Y, GOTO (:FOR LST1)
  GOTOR (MC[-1])      " 155 INSTRUCTIONS

'END' FOR LST

SW DEC:
  'BEGIN' SW DEC0, SW DEC1, SW DEC2, SW DEC3, SW DEC4,
          SW DEC5, SW DEC6, SW DEC0, SW DEC1, SW DEC2

  SUBC (:NXT SBL)
  S = LETTER LAST SYMBOL, Z
N, A = 381
N, GOTO (:ERRORM)
  SUBC (:IDF)
  SW IDF = S          " SWITCH IDENTIFIER
  SUBC (:LST LTH)
  NBR OF SW E = A     " NUMBER OF SWITCH ELEMENTS
  S = LAST SYMBOL
  S = 92, Z          " LAST SYMBOL = COLONEQUAL ?
N, A = 380
N, GOTO (:ERRORM)
  SW LST = B         " SWORD LIST

```

" TRANSLATOR PROCEDURES NR. 18 CONTINUED

```

      IN SW DEC = S      " IN SWITCH DECLARATION = TRUE
SW DEC0:  SUBC (:NXT SBL)
          S = LETTER LAST SYMBOL, Z
          N, GOTO (:SW DEC3)
          SUBC (:IDF)
          SUBC (:NON DES)      " NONDESIGNATIONAL ?
          Y, A = 376
          Y, SUBC (:ERRORM)
          SUBC (:SUBSCR)      " SUBSCRIPTED ?
          N, GOTO (:SW DEC2)
          MC = S      " M
          SUBC (:ORD CNT)
          S + SWORD0
          A = MC[-1]      " M
          MC = S      " SWORD
          S = A
          SUBC (:SUBS VAR)
SW DEC1:  A = 50
          SUBC (:MACRO)      " MACRO (EXIT)
          GOTO (:SW DEC5)
SW DEC2:  SUBC (:FORMAL)      " FORMAL ?
          Y, A = SWORD1
          Y, JUMP (3)
SW DEC2[3]: SUBC (:DYNAMIC)      " DYNAMIC ?
          A = SWORD2
          Y, A + FUNC DIT
          MC = A
          SUBC (:ADDRSS)
          M[B-1] + A      " SWORD
          GOTO (:SW DEC5)
SW DEC3:  S = DIGIT LAST SYMBOL, Z
          N, GOTO (:SW DEC4)
          SUBC (:UNS NUM)
          SUBC (:IN NM LST)      " IN NAME LIST ?
          N, A = 377
          N, SUBC (:ERRORM)
          S = INT LAB      " INTEGER LABEL
          GOTO (:SW DEC2[3])
SW DEC4:  SUBC (:ORD CNT)
          S + SWORD0
          MC = S      " SWORD
          SUBC (:DES EXP)
          GOTO (:SW DEC1)
SW DEC5:  S = SW LST
          S + NBR OF SW E
          U, B = S, P      " SWITCH LIST COUNT >
          Y, A = 378      "     NUMBER OF SWITCH ELEMENTS ?
          Y, SUBC (:ERRORM)
          S = LAST SYMBOL
          U, S = 87, Z      " LAST SYMBOL = COMMA ?
          Y, GOTO (:SW DEC0)
          S = SW LST
          IN SW DEC = S      " IN SWITCH DECLARATION = FALSE
          S + NBR OF SW E
          U, S = B, P      " SWITCH LIST COUNT <
          Y, A = 379      "     NUMBER OF SWITCH ELEMENTS ?
          Y, SUBC (:ERRORM)
          S = SW IDF      " SWITCH IDENTIFIER

```

100470 -

1

259

SUBC (:MRK POS)

" TRANSLATOR PROCEDURES NR. 18 CONTINUED

```

      A = 128
      S = NBR OF SW E
      SUBC (:MACRO2)      " MACRO2 (CODE,
SW DEC6:      S = 1      "          NUMBER OF SWITCH ELEMENTS)
      PLUS (IN SW DEC)    " M = M + 1
      U, S = B, P        " M > SWITCH LIST COUNT ?
      Y, B = SW LST
      Y, GOTOR (MC[-1])
      A = MS[-1]
      S = A              " PARAMETER INTO S
      RUA(21)
      U, A + 16, Z
      N, A = 6, Z
      SUBC (:MACRO3)
      GOTO (:SW DEC6)
      GOTO (:M[0])
      DOS (M0[-256])
      A = :M[0]          " 90 INSTRUCTIONS

      'END' SW DEC

ARR DEC:      S = LINE COUNTER
      LNC = S            " LNC = LINE COUNTER
      SUBC (:LINE)
      SUBC (:NXT SBL)
      SUBC (:IDF)
      MC = S            " N
      S = 1
      MC = S            " CCUNT = 1

ARR DEC[8]:   S = LAST SYMBOL
      U, S = 87, Z      " LAST SYMBOL = COMMA ?
      Y, SUBC (:NXT SBL)
      Y, SUBC (:IDF)
      Y, S = 1
      Y, M[B-1] + S      " COUNT = COUNT + 1
      Y, GOTO (:ARR DEC[8])
      S = 100, Z        " LAST SYMBOL = SUB ?
      Y, IN AR DEC = S   " IN ARRAY DECLARATION = TRUE
      Y, SUBC (:BND PR LST)
      A = 382
      Y, IN AR DEC = A   " IN ARRAY DECLARATION = FALSE
      N, SUBC (:ERRORM)
      A = 122
      S = MC[-1]        " CCUNT
      SUBC (:MACRO2)     " MACRO2 (TNA, COUNT)
      S = M[B-1]        " N
      SUBC (:LST LTH)
      S = A
      A = 121
      SUBC (:MACRO2)     " MACRO2 (TDA, DIMENSION)
      A = 123
      S = MC[-1]        " N
      SUBC (:MACRO2)     " MACRO2 (TAA, N)
      A = ARR DEC MCR
      A + 66
      SUBC (:MACRO)      " MACRO (ARR DECLA MACRO)
      S = LAST SYMBOL
      U, S = 87, Z      " LAST SYMBOL = COMMA ?

```

100470 -

1

261

Y, GOTO (:ARR DEC)
GOTOR (MC[-1])

" 39 INSTRUCTIONS

" TRANSLATOR PROCEDURES NR. 19

```

BND PR LST:      SUBC (:NXT SBL)
                  SUBC (:ARITHEXP)
                  A = 0
                  SUBC (:MACRO)           " MACRO (STACK)
                  S = LAST SYMBOL
U, S = 90, Z      " LAST SYMBOL = COLON ?
Y, SUBC (:NXT SBL)
Y, SUBC (:ARITHEXP)
Y, A = 0
Y, SUBC (:MACRO)   " MACRO (STACK)
N, A = 383
N, SUBC (:ERRORM)
  S = LAST SYMBOL
U, S = 87, Z      " LAST SYMBOL = COMMA ?
Y, GOTO (:BND PR LST)
  A = 384         " REQUIRE BUS AS LAST SYMBOL
  GOTO (:REQ BUS) " 17 INSTRUCTIONS

PR DEC:          'BEGIN' PR DEC0, PR DEC1, PR DEC2, PR DEC3,
                  PR DEC4, PR DEC5, PR DEC6

                  SUBC (:NXT SBL)
                  SUBC (:IDF)
                  MC = S           " N
                  SUBC (:SKP PR LI)
                  SUBC (:SKP VA LI)
                  SUBC (:SKP SP LI)
                  S = M[B-1]       " N
                  SUBC (:IN LIBR)   " IN LIBRARY ?
N, SUBC (:MRK POS)
  S = M[B-1]                     " N
  SUBC (:IN CODE)                " IN CODE ?
Y, B = 1
Y, GOTO (:TRL CD)
  SUBC (:FUNCTN)                  " FUNCTION ?
Y, SUBC (:SET IN DEC)
  SUBC (:ENTR BLK)
  SUBC (:DISP LVL)
  A = 114
  SUBC (:MACRO2)                  " MACRO2 (DPTR, DISPLAY LEVEL)
  SUBC (:TP OF DSP)
  A = 115
  SUBC (:MACRO2)                  " MACRO2 (INCRB, TOP OF DISPLAY)
  S = M[B-1]                     " N
  SUBC (:LST LTH)
  COUNT = A
  GOTO (:PR DEC3)
PR DEC0:          SUBC (:NXT F I)
                  MC = S           " F = NEXT FORMAL IDENTIFIER
                  SUBC (:IN VA LI) " IN VALUE LIST ?
N, GOTO (:PR DEC1)
  SUBC (:SUBSCR)                  " SUBSCRIPTED ?
Y, A = 58                        " MACRO (GEN)
Y, GOTO (:PR DEC2)
  SUBC (:DESINAL)                 " DESIGNATIONAL ?
Y, A = 57                        " MACRO (CLV)
N, A + 53                        " MACRO (GRV/CIV/CBV/CSTV)
  GOTO (:PR DEC2)

```

" TRANSLATOR PROCEDURES NR. 19 CONTINUED

```

PR DEC1:          SUBC (:ASS TO)          " ASSIGNED TO F ?
                  Y, A = 59                " MACRO (CLPN)
                  N, A = 58                " MACRO (CEN)

PR DEC2:          SUBC (:MACRO)
                  S = MC[-1]              " F

PR DEC3:          REPE (:PR DEC0)
                  SUBC (:DISP LVL)
                  A = 116
                  SUBC (:MACRO2)          " MACRO2 (TDL, DISPLAY LEVEL)
                  SUBC (:LOC SPC)
                  A = 117
                  SUBC (:MACRO2)          " MACRO2 (ENTRPS, LOCAL SPACE)
                  SUBC (:LAB LST)
                  S = M[B-1]              " N
                  SUBC (:LST LTH)
                  COUNT = A
                  GOTO (:PR DEC5)

PR DEC4:          SUBC (:NXT F I)          " F = NEXT FORMAL IDENTIFIER
                  SUBC (:IN VA LI)        " IN VALUE LIST ?
                  Y, SUBC (:SUBSCR)        " ^ SUBSCRIPTED ?
                  N, GOTO (:PR DEC5)
                  MC = S
                  SUBC (:INTEGER)         " F
                  A = 123                 " INTEGER ?
                  SUBC (:MACRO2)          " MACRO2 (TAA, F)
                  Y, A = 61                " MACRO (TIAV)
                  N, A = 60                " MACRO (TAV)
                  SUBC (:MACRO)
                  S = MC[-1]              " F

PR DEC5:          REPE (:PR DEC4)
                  LAST LNC = - S          " MAKE LAST LNC = LINE COUNTER
                  S = M[B-1]              " N
                  F = 145
                  SUBC (:SAVE LNC)        " MACRO2 (SLNC, N)
                  S = LAST SYMBOL
                  U, S = 104, Z            " LAST SYMBOL = BEGIN ?
                  N, SUBC (:STATMNT)
                  N, JUMP (5)
                  SUBC (:NXT SBL)
                  SUBC (:DECL LS)
                  Y, SUBC (:DEC LST)
                  SUBC (:CMP TL)
                  SUBC (:NXT SBL)
                  S = M[B-1]              " N
                  SUBC (:NON FCT)         " NONFUNCTION ?
                  Y, GOTO (:PR DEC6)
                  SUBC (:SET IN DEC)
                  SUBC (:LOC POS)
                  SUBC (:ARMETIC)         " ARITHMETIC ?
                  Y, SUBC (:AR NAME)
                  Y, GOTO (:PR DEC6)
                  SUBC (:BOOLEAN)        " BOOLEAN ?
                  Y, SUBC (:BL NAME)
                  N, SUBC (:ST NAME)
                  N, A = 70
                  N, SUBC (:MACRO)        " MACRO (LOS)
                  S = MC[-1]              " N
                  F = 146

```

PR DEC6:

100470 -

1

264

SUBC (:SAVE LNC)

" MACRO2 (RLNC, N)

" TRANSLATOR PROCEDURES NR 19/20

```

      SUBC(:USE CST)           " USE OF COUNTER STACK ?
Y, A = 72                     " MACRO (EXITPC)
N, A = 71                     " MACRO (EXITP)
      SUBC (:MACRO)
      GOTO (:EXIT BLK)        " 101 INSTRUCTIONS

      'END' PR DEC

SAVE LNC:  A = WANTED, P      " WANTED ?
           Y, SUBC (:FUNCTN)  " ^ FUNCTION ?
           N, GOTOR (MC[-1])
           A = G
           S = 2              " LOCAL POSITION
           GOTO (:MACRO2)     " 6 INSTRUCTIONS
BLOCK:     SUBC (:ENTR BLK)
           SUBC (:DISP LVL)
           A = 112
           SUBC (:MACRO2)     " MACRO2 (TBL, DISPLAY LEVEL)
           SUBC (:LOC SPC)
           A = 113
           SUBC (:MACRO2)     " MACRO2 (ENTR8, LOCAL SPACE)
           SUBC (:LAB LST)
           S = BASE          " SUBC(:DEC LST3) OR
           SUBC(MS[2])       " SUBC(:DEC LST3CM8)
           SUBC (:CMP TL)
           SUBC (:USE CST)   " USE OF COUNTER STACK ?
           SUBC (:DISP LVL)
           Y, A = 126        " MACRO2 (EXITC, DISPLAY LEVEL)
           N, A = 125        " MACRO2 (EXITB, DISPLAY LEVEL)
           SUBC (:MACRO2)
           GOTO (:EXIT BLK)  " 21 INSTRUCTIONS

DEC LST3CM8: S = :BASE0      " THIS PART TO BE CARRIED OUT
           BASE = S          " ONLY CNCE FOR THE WHOLE PROGRAM
           SUBC(:DEC LST)
           S = LAST SYMBOL   " ) LAST SYMBOL = END?
           U, S = 105, Z     " )
           N, A = 803
           N, SUBC(:ERRORM)
           GOTOR(MC[-1])    " 8 INSTRUCTIONS

DEC LST3:
DEC LST:   'BEGIN' DEC LST0, DEC LST1, DEC LST2

           F = 0
           MC = F           " FUTURE = ARR DEC = 0
DEC LST0:  S = CMODE
           S '*'- 8, Z      " ) IF TESTING OF LIBRARY PROCEDURES
           Y, SUBC(:INSPECT DECL) " )VERIFY PROCEDURE DECLARATOR
           S = TP DEC LS, Z " TYPE DECLARATOR LAST SYMBOL ?
           Y, SUBC (:SKP TP DEC)
           Y, GOTO (:DEC LST2)
           S = ARR DEC LS, Z " ARR DECLARATOR LAST SYMBOL ?
           N, GOTO (:DEC LST1)
           A = M[B-1], Z    " FUTURE = 0 ?
           N, M[B-1] = S    " FUTURE = 0
           N, SUBC (:SUBSTT)
           A = 1

```

```
DEC LST1:      M[B-2] = A          " ARR DEC = 1
                SUBC (:ARR DEC)
                GOTO (:DEC LST2)
                S = M[B-1], Z      " FUTURE = 0 ?
Y, A = 102
Y, SUBC (:MACRO2)                " MACRO2 (JU, FUTURE)
Y, M[B-1] = S                    " FUTURE
                S = LAST SYMBOL
U, S = 113, Z                    " LAST SYMBOL = SWITCH ?
Y, SUBC (:SW DEC)
N, SUBC (:PR DEC)
```

" TRANSLATOR PROCEDURES NR. 20 CONTINUED

```
DEC LST2:      S = LAST SYMBOL
               U, S = 91, Z      " LAST SYMBOL = SEMICOLON ?
               Y, SUBC (:NXT SBL)
               N, A = 385
               N, SUBC (:ERRORM)
               SUBC (:DECL LS)   " DECLARATOR LAST SYMBOL ?
               Y, GOTO (:DEC LST0)
               A = MC[-1], Z     " FUTURE = 0 ?
               N, SUBC (:SUBSTT)
               A = MC[-1], Z     " ARR DEC = 0 ?
               Y, GOTOR (MC[-1])
               SUBC (:DISP LVL)
               A = 124           " MACRO2 (SWP, DISPLAY LEVEL)
               GOTO (:MACRO2)    " 39 INSTRUCTIONS

               'END' DEC LST
```

INSPECT DECL: 'BEGIN' LOOP, LOOP1, PROC END, END0, END1

```

      SUBC(:DISP LVL)          " IF DISPLAY LEVEL>1 THEN
      S = 1, Z                " INSPECT DECL NOT
N, GOTOR(MC[-1])              " APPROPRIATE
      S = CHARACTER
      RUS(19)
U, S = 25, P                  " PROCEDURE DECLARATION?
N, S = 15, E
Y, A = 804                    " ) IF NO PROCEDURE REPORT THIS FACT
Y, GOTO(:ERRORM)              " ) AND LEAVE INSPECT DECL
      S = LAST SYMBOL
      MC = S                  " )
      F = STOCK1              " )
      MC = F                  " ) RELEVANT INTERNAL STATE
      F = TEXT ARRAY POINTER " ) FOR NXT SBL
      MC = F                  " )

      SUBC(:NXT SBL)
      S = LETTER LAST SYMBOL, Z
N, GOTO(:END1)
      SUBC(:IDF)
      MC = S                  " N
      SUBC(:NAME IN LIBRARY)
N, A = 805                    " REPORT NAME ALREADY USED
N, SUBC(:ERRORM)              " IN LIBRARY
      S = M[B-1]              " N
      S = MS[-1]
      S '+' 32767              " FORMAL COUNT + 1
      MULS(4)                  " 4* FORMAL COUNT + 4 + WORD COUNT
      S + WORD COUNT
      S + CAT END
      S = END OF CATALOGUE, P " SPACE ENOUGH?
Y, A = 809
Y, SUBC(:ERRORM)
Y, GOTO(:END0)
      G = END OF CATALOGUE
      A = 1
      NBR OF ROUTINES + A
      A + WORD COUNT
      M[B] = A
      A = NLP
LOOP:  S = MA                  " )
      MG = S                  " ) JUST READ
      A = 1                  " ) PROCEDURE IDENTIFIER
      G = 1                  " ) INTO
      S = 1                  " ) CATALOGUE
      M[B] = S, Z            " ) BY CCOPYING FROM
N, GOTO(:LOOP)               " ) BOTTOM OF NAMELIST
      A = M[B-1]             " N OF PROC IDENTIFIER
      S = MA                 " )
      S '+' -32767           " ) CHAR#D19 OUT OF NAMELIST
      S '+' NBR OF ROUTINES " ) + NUMBER OF ROUTINES
      MG = S                 " ) INTO CATALOGUE
      S = MA[-1]
      S '+' 32767
      S = 1, Z
      M[B] = S
      S + LOCAL NUMBER      " ) '222 000 001' + FORMAL COUNT
      MG[-1] = S            " ) INTO CATALOGU

```

```

      A = 7
Y, GOTO(:PROC END)
LOOP1:  S = -1
      G = 1
      MG[-1] = S
      S = MA[-2], P
      A = 1
N, JUMP(-3)
      S '*' - D18MIN1
      MG[-2] = S
      S = 0
      MG[-3] = S
      G = 2
      A = 1
      S = 1
      M[B] = S, Z
PROC END: N, GOTO(:LOOP1)
      S = 0
      MG[-2] = S
      G = 3
      END OF CATALOGUE = G
END0:    B = 1
END1:    F = MC[-2]
      TEXT ARRAY POINTER = F
      F = MC[-2]
      STOCK1 = P
      S = MC[-1]
      LAST SYMBOL = S
      GOTOR(MC[-1])
      'END' INSPECT DECL
      'SKIP' 25
      " ) SIMULATE LETTERS
      " )
      " ) SKIP LETTERS
      " )
      " CHARACTER OF FORMAL
      "
      " FILL ZERO IN AT END OF PROCEDURE
      " THROW AWAY N OF PROC IDENTIFIER
      " 86 INSTRUCTIONS
      "RESERVE SAME SPACE IN COMPILER
      "PART

```

```

LAB LST:      SUBC(:NBR LL)
               U, S = 0, Z      " NUMBER OF LOCAL LABELS = 0 ?
               Y, GOTOR (MC[-1])
               MC = S          " COUNT
               LUS (1)
               A = 110
               SUBC (:MACRO2)   " MACRO2 (DECB, 2 * COUNT)
               SUBC (:DISP LVL)
               A = 120
               SUBC (:MACRO2)   " MACRO2 (LAD, DISPLAY LEVEL)
               S = 0           " N = 0
LAB LST[11]:  SUBC (:NEXT LL)   " N = NEXT LOCAL LABEL
               SUBC (:SUP LOC)  " SUPER LOCAL ?
               Y, GOTO (:LAB LST[11])
               A = 1
               M[B-1] = A, Z    " COUNT = 1 ?
               Y, A = 119       " MACRO2 (LAST, N)
               N, A = 118       " MACRO2 (NIL, N)
               N, MC = S        " N
               SUBC (:MACRO2)   " N
               S = MC[-1]
               N, GOTO (:LAB LST[11])
               GOTOR (MC[-1])   " 23 INSTRUCTIONS

```

PROGRAM3:

PROGRAM:

PROGRAM[2]:

```

               A = LETTER LAST SYMBOL, Z
               Y, SUBC (:IDF)
               Y, A = LAST SYMBOL
               Y, A = 90, Z      " LAST SYMBOL = COLON ?
               Y, SUBC (:LAB DEC)
               Y, GOTO (:PROGRAM)
               A = DIGIT LAST SYMBOL, Z
               Y, SUBC (:UNS NUM)
               Y, SUBC (:IN NM LST) " IN NAME LIST ?
               Y, S = INT LAB
               Y, GOTO (:PROGRAM[2])
               JUMP(3)

```

PROGRAM3CM8:

```

               U, S = 104, Z
               N, A = 801
               N, SUBC (:ERRORM)
               S = LAST SYMBOL
               U, S = 104, Z      " LAST SYMBOL = BEGIN ?
               SUBC (:NXT SBL)
               N, JUMP (-3)
               SUBC (:DECL LS)   " DECLARATOR LAST SYMBOL ?

```

" TRANSLATOR PROCEDURES NR. 20 CONTINUED

```

Y, SUBC (:BLOCK)
S =BASE
N, SUBC(MS[1])
A = 82
GOTO(:MACRO)
PROGRAM3CM2: SUBC(:NXT SBL)
SUBC(:DECL LS)
SUBC(:ENTR BLK)
GOTO(:PRDEC)

LAB DEC: LAST LNC = - S
SUBC (:SUBSCR)
Y, A = 388
Y, SUBC (:ERRORM)
Y, SUBC (:SUBS VAR)
N, SUBC (:MRK POS)
S = LAST SYMBOL
U, S = 90, Z
Y, GOTO (:NXT SBL)
A = 389
GOTO (:ERRORM)

```

" GO ACCORDING TO BASE TO
 ") CMP TLJ OR CMP TL3CM8
 " 25 INSTRUCTIONS
 ") THE SYSTEM KNOWS
 ") THAT THE PROGRAM
 ") IS CORRECT AND IS A PROCEDURE
 ") DECLARATION EMBRACED BY
 ") BEGIN END END
 " MAKE LAST LNC + LINE COUNTER
 " SUBSCRIPTED ?
 " LAST SYMBOL = COLON ?
 " 11 INSTRUCTIONS

" MACRO PROCESSOR PROCEDURES NR. 21

```

SUBSTT:          S = A, P          " ) PAR ≥ 0?
                N, S = - A          " )
                MC = S              "STACK ABS(PAR)
                S = 511, P
                SUBC(:ORDCNT)
                Y, S = 267          "PAR IN 18 BITS
                N, S = 268          "PAR IN 9 BITS
                SUBC(BTSTRM9)
                S = MC[-1]
                Y, GOTO(:BTSTRM18)  "PAR
                GOTO(BTSTRM9)      "PAR

SUBST2:          MC = A, P
                N, M[8-1] = - A    " ASSURE PARAMETER ≥ 0 ~ - (-0)
                A = 260
                SUBC(:RUNVA)
                S = MC[-1]
                GOTO(:BTSTRM18)    " 6 INSTRUCTIONS

ORD CNT:        A = 75
                SUBC (:MACRO)      " MACRO (EMPTY)
                S = INSTR CNTR
                GOTOR (MC[-1])     " 4 INSTRUCTIONS

MACRO:
MACRO2:         'BEGIN' STATE0, STATE1, STATE2, STATE3, END, STACK

                NBR = A            " MACRONUMBER
                JUMP(4)

MACRO3:        Y, A = 257          " DETERMINE MACRONUMBER
                N, A = 258          " ACCORDING TO CONDITION
                NBR = A
                A = 128            " MACROWORD MUST BECOME CODE

                A + BEGIN OF MCR LIST
                A = MA
                MCR = A
                PAR = S
                G = NBR
                S = STATE
                JUMP (S)            " SWITCH ACCORDING TO STATE
                GOTO (:STATE0)
                GOTO (:STATE1)
                GOTO (:STATE2)

STATE3:        G = 1, Z            " MACRO = NEG ?
                Y, GOTO (:OPTIMIZE)
                SUBC (:UNLOAD)
                A = MCR
                G = NBR
                G = 0, Z            " MACRO = STACK ?

STATE0:        Y, A = 1
                Y, STATE = A        " STATE = 1
                Y, GOTOR (MC[-1])
                SUBC (:SAT)         " SIMPLE ARITHMETIC TAKE MACRO ?
                Y, S = 3
                Y, GOTO (:LOAD)
                F = PAR             " F = (PAR, NBR)

```

100470 -

1

273

END: SUBC (:PRODUCE)
 S = - INSTR CNTR
 S + 1
STATE1: GOTOR (MC[-1])
 S = 2

 SUBC (:LOAD)
 SUBC (:SAT)

" SIMPLE ARITHMETIC TAKE MACRO ?

" MACRO PROCESSOR PROCEDURES NR. 21/22

```

        Y, GOTOR (MC[-1])
        SUBC(:STACK)
        SUBC (:UNLOAD)
        GOTO (:END)
STATE2:  SUBC (:OPT OP)          " OPTIMIZABLE OPERATOR ?
        Y, GOTO (:OPTIMIZE)
        SUBC(:STACK)
        G = NBR
        GOTO(:STATE3)
STACK:   F = 0                  " F = (0,0)
        A = BEGIN OF MCR LIST
        A = MA
        GOTO(:PRODUCE)        " 49 INSTRUCTIONS

'END' MACRO

LOAD:    STATE = S              " STATE = STATE 1
        STACK0 = A             " STACK0 = MACRO
        F = PAR                " STACK1 = PARAMETER
        STACK1 = F             " STACK2 = MACRONUMBER
        GOTOR (MC[-1])        " 5 INSTRUCTIONS

OPTIMIZE: A = MCR
        SUBC(:OPT NBR)
        MULS(5)
        MC = S
        A = STACK0
        SUBC (:OPT NBR)
        S + MC[-1]
        S + 143                " MACRONUMBER COUNTS FROM OFF :MCR LST
        STACK2 = S
        S + BEGIN OF MCR LIST
        A = MS
        STACK0 = A             " 12 INSTRUCTIONS

UNLCAD:  S = 0
        STATE = S              " STATE = 0
        A = STACK0
        F = STACK1            " 4 INSTRUCTIONS

PRODUCE: 'BEGIN' NORMAL, END

        PARAMETER = F          " ) INITIALIZE PARAMETER AND
                                " ) NUMBER(INPUT-PARAMETERS OF
                                " ) PRCS PAR ^ PRCS STP

        S = NUMBER
        U, S = 75, Z           " MACRO = EMPTY?
        Y, GOTOR(MC[-1])
        MC = A                 " MACROWORD
        U, S '*' 256, Z        " MACRONUMBER = 257 ~ 258(CODE)
                                " ~ 259 (NOT LISTED MACRO)
        SUBC(BTSTRM9)          " DIRECTIVE OR MACRONUMBER INTO STRING
        Y, GOTO(:NORMAL)
        S = NUMBER             " MACROWORD
        S = 259, Z             " MACROWORD NOT LISTED?
        N, S = PARAMETER       " ) IF MACRO = CODE THEN PARAMETER INTO STRING
        Y, S = M[B-1] " )      ELSE MACROWORD INTO STRING
        SUBC(:BTSTRM27)

```

```

N, GOTO(:END)                " IF MACRO = CODE THEN GOTO END
NORMAL:  A = M[B-1]           " MACROWORD
        SUBC(:PRCS STP)
        SUBC(:PAR PART)
        U, S = 0, Z           " PARAMETER?
        Y, GOTO(:END)         " INDICATE NO PARAMETER
        SUBC(:PRCS PAR)      " GET PARAMETER AND DELIVER
                                " ) INDICATOR IN S

N, SUBC(BTSTRM9)
  S = PARAMETER
  SUBC(:BTSTRM18)            " PARAMETER INTO STRING
END:     A = MC[-1]
        SUBC(:INSTR NBR)
        INSTR CNTR + S
        GOTOR(MC[-1])        " 27 INSTRUCTIONS

'END' PRODUCE

PRCS STP: 'BEGIN' REACT10, REACT11, REACT12,
          REACT13, REACT14, REACT15

        S = CODE BODY, P
        Y, GOTOR (MC[-1])
        SUBC (:B REACT)
        U, S = 8, P           " REACTION 2 9 ?
        Y, JUMP (S)
        S = 4
        PLUS (ST CNT)         " B
        U, S = MAX DPTH, P    " B > MAX DEPTH ?
        Y, MAX DPTH = S
        REACT11:             GOTOR (MC[-1])
        S = DIMENSION
        LUS (1)
        S = 2
        ST CNT = S           " B = B - 2 * (DIMENSION - 1)
        GOTOR (MC[-1])
        GOTO (:REACT10)
        GOTO (:REACT11)
        GOTO (:REACT12)
        GOTO (:REACT13)
        GOTO (:REACT14)
        REACT15:             S = ST CNT
        ST CNT = S           " B
        U, S = MAX PRL, P    " B > MAX PROC LEVEL ?
        Y, MAX PRL = S
        REACT15[4]:          S = RET MD
        MAX DPTH = S         " MAX DEPTH = RET MAX DEPTH
        GOTOR (MC[-1])
        REACT12:             S = ECNT, Z
        S + 1                 " ECOUNT = 0 ?
        ECNT = S
        N, GOTOR (MC[-1])
        S = ST CNT
        RET LVL = S          " RET LEVEL = B
        S = MAX DPTH
        RET MD = S           " RET MAX DEPTH = MAX DEPTH
        S = MAX DI
        MAX DPTH = S         " MAX DEPTH = MAX DEPTH ISR
        REACT10:            S = 0
        ST CNT = S           " B = 0

```

100470 -

1

276

GOTOR (MC[-1])

" MACRO PROCESSOR PROCEDURES NR. 22/23

```

REACT13:      S = NUMBER
              S = 127, Z
              N, JUMP (4)
              S = ST CNT
              U, S = MAX DI, P
              Y, MAX DI = S
              SUBC (:REACT11)
              S = ECNT, Z
              Y, GOTOR (MC[-1])
              S = 1, Z
              ECNT = S
              N, GOTOR (MC[-1])
              S = MAX DPTH
              U, S = MAX DI, P
              Y, MAX DI = S
              S = RET LVL
              ST CNT = S
              GOTO (:REACT15[4])
REACT14:      SUBC (:DISP LVL)
              ST CNT = S
              SUBC (:TP OF DSP)
              PLUS (ST CNT)
              U, S = MAX DL, P
              Y, MAX DL = S
              S = MAX DPTH
              RET MD = S
              GOTOR (MC[-1])

              'END' PRCS STP

PRCS PAR:      'BEGIN' KIND2, KIND3, KIND1, KIND0, END WITH NO, LENTRY

              SUBC (:KIND)
              JUMP (S)
              GOTO (:KIND0)
              GOTO (:KIND1)
              GOTO (:KIND2)
              GOTO (:KIND3)

KIND2:        S = NUMBER
              U, S = 89 , Z
              N, S = 124 , Z
              N, JUMP (5)

              S = S , Z
              A = PARAMETER
              Y, LUA (9)
              N, A = 116
              PARAMETER = A

              " MACRO = TBC?
              " ~ MACRO = SWP?

              " ) IF MACRO = SWP THEN
              " ) PAR := PAR*512
              " ) IF MACRO = TBC THEN
              " ) PAR := IF PAR = TRUE SYMBOL
              " ) THEN 0 ELSE 1

KIND3:        A = PARAMETER, P
              N, PARAMETER = -A, P
              GOTO (MC[-1])

              " PAR := ABS(PAR)
              " AND AVOID -0
              " CONDITION = YES

KIND1:        A = PARAMETER, Z
              Y, JUMP (3)
              S = A, P
              Y, SUBC (:PR ADR)

              " ) IF PAR = 0 THEN PAR IS PURE
              " ) IF PAR > 0 THEN GET PAR OUT
              " ) OF NAMELIST ELSE PAR ALREADY IN A

```

```

      A = A, Z
Y, S = 2
N, S = 0
      GOTO(:END WITH NO)

KIND0:      S = PARAMETER
            SUBC(:PROC)          " ) CALL OF LIBRAR PROC?
Y, SUBC(:IN LIBR)              " )
Y, GOTO(:LENTRY)
            SUBC(:DYNAMIC)
            SUBC(:ADDRSS)
N, S = 0
Y, S = 1

END WITH NO:      PARAMETER = A, P          " PAR := ABS(PAR)
N, PARAMETER = -A          " AND AVOID -0
U, S = 1, Z
      GOTO(MC[-1])          " CONDITION = NO

LENTY:      SUBC(:ADDRSS)
            S = 3          " LIBRARY INDICATION
            GOTO(:END WITH NO)      " 41 INSTRUCTIONS

'END' PRCS PAR

SAT:      U, A '*' 1, Z
            GOTO (:INVERT)          " 2 INSTRUCTIONS

OPT OP:      U, A '*' 2, Z
            GOTO (:INVERT)          " 2 INSTRUCTIONS

OPT NBR:      S = A
            RUS (4)
            S '*' 15
            GOTOR (MC[-1])          " 4 INSTRUCTIONS

B REACT:      S = A
            RUS (21)
            GOTOR (MC[-1])          " 3 INSTRUCTIONS

```

" NAME LIST PROCEDURES NR. 24

CHARCTR:	A = MS RUA (19) A '*' 31 U, A = 24, Z GOTO (:INVERT)	" CHARACTER " = 24 ? " 5 INSTRUCTIONS
ARMETIC:	SUBC (:CHARCTR) N, GOTO (MC[-1]) U, A '*' 6, Z N, A '*' 3, Z GOTO (MC[-1])	" CHARACTER ≠ 24 ? " TYPE BITS = 0, 1 ? " √ TYPE BITS = 4 ? " 5 INSTRUCTIONS
REAL:	SUBC (:CHARCTR) Y, A '*' 7, Z GOTO (MC[-1])	" CHARACTER ≠ 24 ? " ^ TYPE BITS = 0 ? " 3 INSTRUCTIONS
INTEGER:	A = MS RUA (19) A '*' 7 U, A = 1, Z GOTO (MC[-1])	" TYPE BITS " = 1 ? " 5 INSTRUCTIONS
BOOLEAN:	SUBC (:INTEGER) U, A = 2, Z GOTO (MC[-1])	" TYPE BITS = 2 ? " 3 INSTRUCTIONS
STRING:	SUBC (:INTEGER) U, A = 3, Z GOTO (MC[-1])	" TYPE BITS = 3 ? " 3 INSTRUCTIONS
DESINAL: NO TYPE:	SUBC (:INTEGER) U, A = 6, Z GOTO (MC[-1])	" TYPE BITS = 6 ? " 3 INSTRUCTIONS
ARBCST:	SUBC (:CHARCTR) N, GOTO (MC[-1]) GOTO (:NON DES)	" CHARACTER ≠ 24 ? " 3 INSTRUCTIONS
UNKNOWN:	SUBC (:INTEGER) U, A = 7, Z GOTO (MC[-1])	" TYPE BITS = 7 ? " 3 INSTRUCTIONS
NON AR:	SUBC (:CHARCTR) N, GOTO (:INVERT) A '*' 7 U, A = 2, Z N, A = 3, Z N, A = 3, Z GOTO (MC[-1])	" CHARACTER ≠ 24 ? " TYPE BITS " = 2 ? " √ = 3 ? " √ = 6 ? " 7 INSTRUCTIONS

" NAME LIST PROCEDURES NR. 24/25

NON RE:	SUBC (:NON AR)	" NONARITHMETIC ?
	N, A + 5, Z	" ✓ TYPE BITS = 1 ?
	GOTO (MC[-1])	" 3 INSTRUCTIONS
NON IN:	SUBC (:NON AR)	" NCNARITHMETIC ?
	N, A + 6, Z	" ✓ TYPE BITS = 0 ?
	GOTO (MC[-1])	" 3 INSTRUCTIONS
NON BL:	SUBC (:BOOLEAN)	" BOOLEAN ?
	N, A - 5, Z	" ✓ TYPE BITS = 5 ?
	N, A - 2, Z	" ✓ TYPE BITS = 7 ?
	GOTO (:INVERT)	" 4 INSTRUCTIONS
NON STR:	SUBC (:STRING)	" STRING ?
	GOTO (:NON BL[1])	" 2 INSTRUCTIONS
NON DES:	SUBC (:INTEGER)	
	U, A - 5, P	" TYPE BITS > 5 ?
	GOTO (:INVERT)	" 3 INSTRUCTIONS
SIMPLE:	A = MS	
	RUA (19)	" CODE BITS
	U, A - 127, Z	" = 127 ?
	N, A '*' 24, Z	" ✓ CHARACTER 1 8 = 0 ?
	GOTO (MC[-1])	" 5 INSTRUCTIONS
SIMPLE1:	A = MS	
	RUA (22)	
	A '*' 3, Z	" CHARACTER 1 8 = 0 ?
	GOTO (MC[-1])	" 4 INSTRUCTIONS
SUBSCR:	SUBC (:SIMPLE1)	
	A - 1, Z	" CHARACTER 1 8 = 1 ?
	GOTO (MC[-1])	" 3 INSTRUCTIONS
PROC:	A = MS	
	RUA (19)	
	U, A - 127, Z	" CODE BITS = 127 ?
	N, A '*' 16, Z	" ✓ CHARACTER 1 8 < 2 ?
	GOTO (:INVERT)	" 5 INSTRUCTIONS
FUNCTN:	SUBC (:SIMPLE1)	
	A - 2, Z	" CHARACTER 1 8 = 2 ?
	GOTO (MC[-1])	" 3 INSTRUCTIONS
NON SMPL:	SUBC (:SIMPLE)	" SIMPLE ?
	Y, GOTO (:INVERT)	
	A - 16, Z	" FUNCTION ?
	N, SUBC (:FORMAL)	" ✓ FORMAL ?

" NAME LIST PROCEDURES NR. 25 CONTINUED

	Y, SUBC (:PROC)	" ^ PROC ?
	Y, A = MS[-1]	
	Y, A '*' 32766, Z	" ^ LIST LENGTH < 1 ?
	GOTO (:INVERT)	" 8 INSTRUCTIONS
NON SUBS:	SUBC (:SIMPLE1)	" SIMPLE1 ?
	Y, GOTO (MC[-1])	
	GOTO (:PROC)	" 3 INSTRUCTIONS
NON PROC:	A = MS	
	RUA (18)	
	U, A '*' 32, Z	" CHARACTER : 8 < 2 ?
	N, GOTO (MC[-1])	
	U, A = 191, P	" FORMAL ?
	Y, A '*' 49, Z	" ^ SIMPLE1 ^ ~ ASSIGNED TO ?
	GOTO (:INVERT)	" 7 INSTRUCTIONS
NON FCT:	SUBC (:FUNCTN)	" FUNCTION ?
	N, SUBC (:FORMAL)	" v FORMAL ?
	GOTO (:INVERT)	" 3 INSTRUCTIONS
FORMAL:	A = MS	
	RUA (19)	
	U, A = 95, P	" CCDE BITS > 95 ?
	GOTO (MC[-1])	" 4 INSTRUCTIONS
IN VALI:	SUBC (:FORMAL)	" FORMAL ?
	U, A = 63, E	" v CCDE BITS < 64 ?
	GOTO (:INVERT)	" 3 INSTRUCTIONS
ASS TO:	A = - MS	
	LCA (8), P	" D18 OF NAME LIST[N] = 1 ?
	GOTO (MC[-1])	" 3 INSTRUCTIONS
DYNAMIC:	SUBC (:ASS TO)	" ASSIGNED TO ?
	N, A '*' 64, Z	" v CODE BITS > 63 ?
	GOTO (MC[-1])	" 3 INSTRUCTIONS
IN LIBR:	A = - MS[-1]	
	LCA (1), P	" D25 OF NAME LIST[N + 1] = 1 ?
	GOTO (MC[-1])	" 3 INSTRUCTIONS
OP LIKE:	A = - MS[-1]	
	LCA (3), P	" D23 OF NAME LIST[N + 1] = 1 ?
	GOTO (MC[-1])	" 3 INSTRUCTIONS
OUT DEC:	SUBC (:OP LIKE)	
	GOTO (:IN LIBR[1])	" 2 INSTRUCTIONS

" NAME LIST PROCEDURES NR. 25/26

ASS TO FD:	A = - MS[-1] LCA (5), P GOTO (MC[-1])	" D21 OF NAME LIST[N + 1] = 1 ? " 3 INSTRUCTIONS
DECLARED:	A = - MS[-1] LCA (7), P GOTO (MC[-1])	" D19 OF NAME LIST[N + 1] = 1 ? " 3 INSTRUCTIONS
SUP LOC:	A = - MS[-1] GOTO (:ASS TO[1])	" 2 INSTRUCTIONS
LOC POS:	SUBC (:ASS TO FD) N, A = D21 N, MS[-1] + A S = 2 GOTOR (MC[-1])	" ASS TO FUNCTION DESIGNATOR ? " CHANGE (D21) " 5 INSTRUCTIONS
SET IN DEC:	A = - MS[-1] A '+' D22 MS[-1] = - A N, LCA (5), P Y, GOTOR (MC[-1]) A = 390 GOTO (:ERRORM)	" CHANGE (D22) " BCOL < ASS TO FUNCTION DESIGNATOR ? " 7 INSTRUCTIONS
MRK POS:	SUBC (:DECLARED) Y, A = 391 Y, GOTO (:ERRORM) MC = S SUBC (:PR ADR) A + 0, Z N, SUBC (:SUBSTT) S = MC[-1] A = D19 MS[-1] + A GOTOR (MC[-1])	" DECLARED ? " N " PROGRAM ADDRESS " = 0 ? " N " CHANGE (D19) " 11 INSTRUCTIONS
PR ADR:	SUBC (:NONFRM LAB) A = MS[-1] Y, S = 1 LCA (7), P A = MS N, JUMP (9) MC = S MC = A SUBC (:ORD CNT) A = MC[-1] A '*' = 32767 A + S S = MC[-1] MS = A A = M[B+1] A '*' 32767	" CCDE BITS = 6 ? " M " = DECLARED ? " NAME LIST[M] " M " WCRD " WCRD " HEAD " M " NAME LIST[M] = HEAD + ORDER COUNTER " WCRD

NAME LIST PROCEDURES NR. 26/27

```

GOTOR (MC[-1]) " 17 INSTRUCTIONS

ADDRSS:      A = MS
              RUA (19) " CCDE BITS
U, A = 13, P
U, A = 24, E " < 13 ~ ≥ 25 ?
N, GOTO (:PR ADR)
  SUBC (:DYNAMIC) " DYNAMIC ?
  A = MS
  A '*' 32767
N, GOTOR (MC[-1])
U, A = IN SW DEC, Z
Y, GOTOR (MC[-1])
  SUBC (:PRC LVL)
  LUS (9)
  S '+' A
  S '*' 32256, Z " LEVEL = PROC LEVEL ?
Y, A '*' 511
Y, A + 32256
  GOTOR (MC[-1]) " 18 INSTRUCTIONS

LST LTH:    A = MS[-1]
            A '*' 32767
            A = 1
            GOTOR (MC[-1]) " 4 INSTRUCTIONS

TEST FC:    A = MS[-1]
            RUA (20)
            A = FOR CNT, P " NAME LIST[N + 1] & D20 > FOR COUNT ?
            A = 392

TEST RET:   N, GOTOR (MC[-1])
            GOTO (:ERRORM) " 6 INSTRUCTIONS

CHKC DIM:   A = 393
            G = DIMENSION
CHKC RET:   MC = A " ERROR NUMBER
            SUBC (:FORMAL) " CCDE BITS
            A = 14, Z " NCN FORMAL SWITCH ?
Y, A = 1
N, SUBC (:LST LTH)
U, A + 1, Z " = 1 ?
N, A = G, Z " CORRECT ?
            A = MC[-1] " ERROR NUMBER
Y, GOTOR (MC[-1])
  GOTO (:ERRORM) " 12 INSTRUCTIONS

CHKC LL:    A = MA[-1]
            A '*' 32767, Z " LIST LENGTH (F) = - 1 ?
Y, GOTOR (MC[-1])
  F = :MA[-1]
  A = 394
  GOTO (:CHKC RET) " 6 INSTRUCTIONS

```

" NAME LIST PROCEDURES NR. 27 CONTINUED

```

CHK TP:      A = MA
              RUA (19)
              A '*' 7      " TYPE EITS (F)
              A + :CHK TP[8]
              DO (MA)      " TYPE ERRONEOUS ?
Y, SUBC (:CHARCTR)      " ^ CHARACTER # 24 ?
              A = 395
              GOTO (:TEST RET)
CHK TP[8]:   SUBC (:NON AR)
              SUBC (:NON AR)
              SUBC (:NON BL)
              SUBC (:NON STR)
              SUBC (:NON AR)
              SUBC (:NO TYPE)
              SUBC (:NON DES)
              GOTOR (MC[-1])      " 16 INSTRUCTIONS

NBR LL:      S = MD[-3]      " NAME LIST[BLOCK CELL POINTER + 3]
              S '*' 8191
              GOTOR (MC[-1])      " 3 INSTRUCTIONS

NEXT LL:     S + 0, Z      " N = 0 ?
N, GOTO (:NXT IDF)
              S = MD[-3]
              GOTO (:STATUS[1])      " 4 INSTRUCTIONS

NXT F I:     SUBC (:FORMAL)      " FORMAL ?
N, SUBC (:IN VA LI)      " ~ IN VALUE LIST ?
N, SUBC (:IN LIBR)      " ~ IN LIBRARY ?
Y, JUMP (2)
              SUBC (:FUNCTN)      " FUNCTION ?
              S = 8
Y, S = 1
              GOTO (:NXT IDF)      " 8 INSTRUCTIONS

INCR STS:    MD[-2] = A
              GOTOR (MC[-1])      " 2 INSTRUCTIONS

IDF:         SUBC (:RD IDF)
              GOTO (:LOOK UP)      " 2 INSTRUCTIONS

```

" TRANSLATOR PROCEDURES NR. 28

```

SKP PR LI:      S = LAST SYMBOL
                U, S = 98, Z          " LAST SYMBOL = OPEN ?
                N, JUMP (5)
                SUBC (:NXT SBL)
                SUBC (:SKP TP DEC)
                S = LAST SYMBOL
                U, S = 99, Z          " LAST SYMBOL = CLOSE ?
                Y, SUBC (:NXT SBL)
                U, S = 91, Z          " LAST SYMBOL = SEMICOLON ?
                Y, GOTO (:NXT SBL)
                GOTOR (MC[-1])        " 11 INSTRUCTIONS

TRL CD:         'BEGIN' TRL CD0, TRL CD1, TRL CD2, TRL CD3

                S = LAST SYMBOL
                U, S = 102, Z          " LAST SYMBOL = QUOTE ?
                A = 401
                N, GOTO (:TRL CD3)
                CODE BODY = A        " CODE BODY = TRUE
                U, A = HMODE, Z
                N, A = 402
                N, SUBC (:ERRORM)
                SUBC (:NXT SBL)
                S = DIGIT LAST SYMBOL, Z
                N, A = 399
                N, SUBC (:ERRORM)
                N, GOTO (:TRL CD2)
                SUBC (:UNS INT)
                A = VALUE OF CONSTANT[1], Z
                Y, A = 0              " AVOID -U
                U, A = 511, P         " MACRO ≥ 511 ?
                Y, S = 259
                Y, NBR = S
                Y, JUMP(5)

                NBR = A
                U, A = 256, P          ")
                Y, A = 128             ") SELECT CODE AS MACROWORD
                A = BEGIN OF MCR LIST
                A = MA

                MCR = A               " MACRO
                SUBC (:PAR PART)
                S = 0, P              " PAR PART > 0 ?
                N, GOTO (:TRL CD1)
                S = LAST SYMBOL
                U, S = 87, Z           " LAST SYMBOL = COMMA ?
                Y, SUBC (:NXT SBL)
                N, A = 396
                N, SUBC (:ERRORM)
                S = LETTER LAST SYMBOL, Z
                Y, SUBC (:IDF)
                Y, GOTO (:TRL CD1)
                S = DIGIT LAST SYMBOL, Z
                Y, SUBC (:UNS INT)
                Y, S = VALUE OF CONSTANT[1]
                Y, GOTO (:TRL CD1)
                S = LAST SYMBOL

```

```
U, S = 65, Z          " LAST SYMBOL = MINUS ?
N, A = 398
N, JUMP (3)
    SUBC (:NXT SBL)
    S = DIGIT LAST SYMBOL, Z
N, A = 397
N, SUBC (:ERRORM)
Y, SUBC (:UNS INT)
Y, S = - VALUE OF CONSTANT[1]
    A = MCR
    SUBC (:MACRO4)
    S = LAST SYMBOL

TRL CD1:
TRL CD2:
```

" TRANSLATOR PROCEDURES NR. 28/

```

      U, S = 87, Z          " LAST SYMBOL = COMMA ?
      Y, GOTO (:TRL CD0)
      U, S = 103, Z        " LAST SYMBOL = UNQUOTE ?
      Y, SUBC (:NXT SBL)
      A = 400
      CODE BODY = - A      " CODE BODY = FALSE
TRL CD3:  N, SUBC (:ERRORM)
          SUBC (:ENTR BLK)
          GOTO (:EXIT BLK)  " 60 INSTRUCTIONS

          'END' TRL CD

UNS NUM:  SUBC (:UNS NBR)
          S = SMALL, Z
          Y, GOTOR (MC[-1])
          S = START OF CONSTANT LIST
UNS NUM[4]: U, S = DPO, Z
          Y, JUMP (6)
          F = VALUE OF CONSTANT
          F = MS, Z
          N, S + 2
          N, GOTO (:UNS NUM[4])
          U, S = REAL NUMBER, Z
          N, S + 1
          S = START OF CONSTANT LIST
          S + 1            " CORRECTIE = B+GIN OF PR AR-1
          ADRS OF CST = S
          GOTOR (MC[-1])   " 16 INSTRUCTIONS

AR CST:   S = SMALL, Z
          Y, A = 87
          Y, S = VALUE OF CONSTANT[1]
          Y, GOTO (:MACRO2) " MACRO2 (TSIC, VALUE OF CONSTANT)
          S = REAL NUMBER, Z
          Y, A = 85         " MACRO2 (TRC, ADDRESS OF CONSTANT)
          N, A = 86         " MACRO2 (TIC, ADDRESS OF CONSTANT)
          S = ADRS OF CST
          GOTO (:MACRO2)    " 9 INSTRUCTIONS

CST STR:  'BEGIN' CST STR0, CST STR1, CST STR2, CST STR3

          S = 1
          QUOTE COUNTER = S " QUOTE COUNTER = 1
CST STR0: F = 3            " WCRD = 0, COUNT = 3
CST STR1: MC = F
          SUBC (:NXT SBL)
          U, S = 103, Z      " LAST SYMBOL = UNQUOTE ?
          S = MC[-1]        " CCUNT
          A = MC[-1]        " WORD
          LUA (8)           " * 256
CST STR2: Y, GOTO (:CST STR3)
          A + LAST SYMBOL   " + LAST SYMBOL
          S = 1, Z         " CCUNT = COUNT - 1
          N, F = A
          N, GOTO (:CST STR1)
          S = A             " PARAMETER INTO S
          SUBC (:MACRO3)    " CCNDITION = YES

```

" TRANSLATOR PROCEDURES NR. 29 CONTINUED

```

      GOTO (:CST STR0)
CST STR3:  A = 255
           S = 1, P           " CCUNT = CCUNT - 1
Y, GOTO (:CST STR2)
           QUOTE COUNTER = - S " QUOTE COUNTER = 0
           S = A, P           " PARAMETER INTO S AND MAKE
           SUBC (:MACRO3)      " CONDITION = YES
           GOTO (:NXT SBL)     " 24 INSTRUCTIONS

      'END' CST STR

CRF:       'BEGIN' LIBRARY ROUT, FILL INFOTABLE, NP, CRSS,
           PSEUDO LVAR

           SUBC (:INIT POINTER)
           S = CMODE
U, S '*' 8, Z
N, GOTOR(MC[-1])           " IF TESTING OF PROCEDURES EXIT CRF
S '*' 2, Z                 " - INSERTING ROUTINES?
Y, GOTO(:NP)

           S = 3
           PLUSA(END OF INFOTABLE)
           A = 0
           MS[-1] = A       " 10 INSTRUCTIONS

LIBRARY ROUT:  SUBC (:NEXT ENTRY)
Y, GOTOR(MC[-1])           " EXIT CRF
U, A '*' D20, Z           " IF PROCEDURE IS USED THEN CONDITION
N, A '*' D24, Z           " A CN DRUM
N, SUBC (:FILL INFOTABLE) " BECOMES NO
GOTO (:LIBRARY ROUT)      " 5 INSTRUCTIONS

FILL INFOTABLE:  A = END OF INFOTABLE
U, A = MA[-1], Z
N, A = 1               " IF ALREADY A NUMBER IN WORD
Y, JUMP(1)
PLUSA(END OF INFOTABLE) " THEN INSERT CURRENT NUMBER
N, LUS(13)             " AFTER SHIFT AND MAKE NEXT
N, MA[-2] + S          " ENTRY ZERO
N, S = 0               "
MA[-1] = S             " ELSE INSERT NUMBER
A = INFO TAB END, P
Y, A = 806
Y, GOTO(:ERM)          " END OF COMPILATION
GOTOR(MC[-1])          " 13 INSTRUCTIONS

NP:           SUBC (:NEXT ENTRY)
Y, GOTO (:PSEUDO LVAR)
U, A '*' D20, Z         " PROCEDURE CALLED?
N, SUBC (:CRSS)         " WORK OUT CROSSREFERENCE
GOTO (:NP)             " 5 INSTRUCTIONS

CRSS:       'BEGIN' END, CYCLE
           " CRSS EXPECTS IN S AN ENTRY
           " OF THE CROSSTABLE AND BESIDES
           " THAT IF CALLED FROM OUTSIDE IN
           " A AN ENTRY OF THE INFOTABLE

```

```

      S + BEGIN OF CROSSTABLE
      MC = A
      A = MS
      U, A '*' D21, Z           " ALREADY NEEDED?
      N, GOTO(:END)            " THEN EXIT CRSS
      A '+' D 21
      MS = A                    " NOTICE PROCEDURE NEEDED
CYCLE:  S = MA[2], Z            " ANY OTHER PROCEDURE NEEDED?
      Y, GOTO(:END)            " IF NOT EXIT CRSS
      S '*' 8191
      SUBC(:CRSS)
      S = MA[2]
      RUS(13), Z                " ANY OTHER PROCEDURE NEEDED?
      N, SUBC(:CRSS)
      N, A + 1                  " NEXT ENTRY OF INFOTABLE
      N, GOTO(:CYCLE)
END:    A = MC[-1]              " RESTORE ENTRY OF INFOTABLE
      GOTOR(MC[-1])            " 18 INSTRUCTIONS

'END' CRSS

PSEUDO LVAR: 'BEGIN' LOOP, END

      S = 257                    ")
      SUBC(BTSTRM9)              ") LEADING ZERO OF
      S = 0                      ") PSEUDO
      SUBC(:BTSTRM27)           ") LIBRARY VARIABLES
      S = 1
      INSTR CNTR + S             " INSTRUCT COUNTER:=INSTRUCT COUNTER+1
      RELADRSS = S               " INITIALIZE RELADDRESS
      SUBC(:INIT POINTER)
      SUBC(:NEXT ENTRY)
LOOP:   Y, GOTO(:END)
      G = MA[1]
      U, A '*' D21, Z            ") PROCEDURE NEEDED
      N, A '*' D24, Z            ") ^ ON DRUM?
      Y, GOTO(:LOOP)
      A = G
      RELADRSS + A
      A = 261                    ") IN S ENTRY OF CROSSTABLE
      SUBC(:RUNVA)
      S = 2
      INSTR CNTR + S
      GOTO(:LOOP)
END:    S = RELADRSS
      A = 266                    ") INDICATE END ADDRESS
      GOTO(:RUNVA)              ") OF PROCEDURES TO BITSTRING
                                   " EXIT CRF 23 INSTRUCTIONS

'END' PSEUDO LVAR
'END' CRF
" MAIN PROGRAM OF TRANSLATOR

TRANSL: S = 300
      SUBC(:INIT)
      STATE = G                  " STATE = 0
      ST CNT = G                 " B = 0
      MAX DPTH = G               " MAX DEPTH = 0
      MAX DI = G                 " MAX DEPTH ISR = 0
      MAX DL = G                 " MAX DISPLAY LEVEL = 0

```

```

MAX PRL = G      " MAX PROC LEVEL = 0
ECNT = G         " ECOUNT = 0
IN SW DEC = S    " IN SWITCH DECLARATION = FALSE
CODE BODY = - S  " CODE BODY = FALSE
LAST LNC = - S   " MAKE LAST LNC ≠ LINE COUNTER
SUBC(:CRF)       " CROSSREFERENCE
S = INSTR CNTR
A = 264          " ) VALUE OF BEGIN OF PROGRAM
SUBC(:RUNVA)     " ) INTO STRING
A = BEGIN OF NLI " NEXT BLOCK CELL POINTER =
NXT BCP = A      " = BEGIN OF NAME LIST
A = NLP
LAST NLP = A     " LAST NLP = NLP
SUBC (:ENTR BLK)
SUBC (:NXT SBL)
A = BASE         " GO ACCORDING TO BASE TO PROGRAM3
SUBC(MA)         " OR PROGRAM3CM8
S = MAX DPTH
S + MAX DI
S + MAX DL
S + MAX PRL
A = 128          " MACRO2 (CODE, SUM OF MAXIMA)
SUBC(:MACRO2)
S = INSTR CNTR
S - 1
A = 265         " VALUE OF END OF PROGRAM INTO STRING
SUBC(:RUNVA)
S = 256         " INDICATE END OF BITSTRING
GOTO(BTSTRM9)   " 36 INSTRUCTIONS
'SKIP' 1

```

END CAT:
'END'

"LOADER SECTION NR 1.

```

'BEGIN' FUNC LTR, C VAR, L1COP, L1OCP6, P1ROD,
OBJECTCODE, I1MPPAR, D1YPAR, P1URPAR, L1ENTRY,
D1IRECT, I1MPWORD, P1URWORD, N1CT LISTED MACRO,
S1UBSTIT, P1SEUDO LVAR, I1NCREASE IC, V1ALDP0,
V1AL BEGIN PR, V1AL END PR, V1AL RELADRESS, LOOP,
LOOP6, PROD, OBJECTCODE, IMPPAR, DYPAR, PURPAR, LENTRY,
DIRECT, IMPWORD, PURWORD, NOT LISTED MACRO, SUBSTIT,
PSEUDO LVAR, INCREASE IC, VAL DP0, VAL BEGIN PR,
VAL END PR, VAL RELADRESS

```

BEG UP32K[0]:

```

S = 1
CORRECTIE = S
LOOP: SUBC(BTSTREAM9)
      U, S '*' 256, Z
      N, GOTO(D1IRECT)
      MC = S
      S + BEGIN OF MCR LIST
      S = MS
      MC = S
      A = S
      SUBC(:PAR PART)
      U, S = 0, Z
      Y, GOTO(P1ROD)
      SUBC(:KIND)

```

```

        JUMP(S)
        JUMP(3)
        JUMP(2)
        GOTO(P1URPAR)
        GOTO(I1MPPAR)
        SUBC(BTSTREAM9)
        JUMP(S)
        GOTO(I1MPPAR)
        GOTO(D1YPAR)
        GOTO(P1URPAR)
        GOTO(L1ENTRY)

PROD:      'BEGIN' NEXT

        A = MC[-1]
        MC = S
        SUBC(:INSTR NBR)
        MC = S
        SUBC(:PAR PART)
        MC = S
        SUBC(:INSTR PRT)
        S + BEGIN OF INSTR LIST
        MC = S
        A = 1
        M[B-2]-A, Z
        PLUSA(M[B-1])
        S = MA[-1]
        Y, S + M[B-4]
        SUBC(OBJECTCODE)
        A = 1
        M[B-3] = A, P
        Y, JUMP(-9)
        B = 5
        GOTO(L100P)
        'END' PROD

OBJECTCODE:
        A = 1
        PLUSA(INSTR CNTR)
        MA[-1] = S
        GOTOR(MC[-1])

IMPPAR:    SUBC(:BITSTREAM18)
        S + CORRECTIE
        GOTO(P1ROD)

DYPAR:     SUBC(:BITSTREAM18)
        A = M[B-2]
        MC = S
        U, A = 98, Z
        V, S = FUNC LTR
        N, S = C VAR
        U, A = 91, Z
        N, A = 123, Z
        Y, S = FUNC DIT
        S + MC[-1]
        GOTO(P1ROD)

" GET INDICATOR
" GET INDICATOR
" MACROWORD
" STACK PARAMETER IF ANY
" NUMBER
" PAR NUMBER
" ENTRY
" PAR NUMBER = COUNT?
" ENTRY:=ENTRY +1
" ADD PARAMETER
" GOTO(:NEXT)
" 20 INSTRUCTIONS
" 4 INSTRUCTIONS
" 3 INSTRUCTIONS
" ENTRY IN MACROTABLE
" STACK PARAMETER
" MACRO = STST?
" MACRO = TLV
" MACRO = TAA
" ADD VARIANT BITS TO PARAMETER
" 11 INSTRUCTIONS

```

" TRANSFORMATOR ROUTINES NR 4

PUR FAR: SUBC(:BITSTREAM18)

A = M[B-2]

A = 127, Z

" MACRO =EXITSV?

Y, S = - S

GOTO(P1ROD)

" 5 INSTRUCTIONS

LENTY: SUBC(:BITSTREAM18)

S + BEGIN OF TRAFOTABLE

S = MS

" ADDRESS OF PROCEDURE

S '*' 32767

" CLEAR WORD

GOTO(P1ROD)

" 5 INSTRUCTIONS

DIRECT: S '*' - 256

JUMP(S)

GOTOR(MC[-1])

GOTO(P1URWORD)

GOTO(I1MPWORD)

GOTO(N1OT LISTED MACRO)

GOTO(S1UBSTIT)

GOTO(P1SEUDO LVAR)

GOTO(I1NCREASE IC)

GOTO(V1AL DP0)

GOTO(V1AL BEGIN PR)

GOTO(V1AL END PR)

GOTO(V1AL RELADDRESS)

GOTO(S1UBST ONE PAR18)

GOTO(S1UBST ONE PAR 9)" 15 INSTRUCTIONS

IMPWORD: SUBC(:BITSTREAM27)

A = S

S '*' 32767, P

" ISOLATE ADDRESS PART

" AND MAKE CONDITION = YES

S + CORRECTIE

" CORRECT ADDRESS

A '*' - 32767

" FADE OUT OLD ADDRESS

S '*' A

" AND INSERT NEW ADDRESS

PUR WORD: N, SUBC(:BITSTREAM27)

" RECOVER ONLY WHEN

" COMING STRAIGHT OUT OF DIRECTIVE

SUBC(O1BJECTCODE)

GOTO(L1OOP)

" 9 INSTRUCTIONS

" TRANSFORMATOR ROUTINES NR 5

NOT LISTED MACRO: SUBC(:BITSTREAM27)

B + 1
GOTO(L100P6) " 3 INSTRUCTIONS

SUBSTIT: 'BEGIN' CHAIN

SUBC(:BITSTREAM18)
S + CORRECTIE
MC = S
SUBC(:BITSTREAM18)
S + CORRECTIE

CHAIN:
G = S " TAKE ADDRESS TO BE FILLED
A = MS " TAKE CONTENTS OF IT
S = A " COPY IN S
A '*' -32767 " HEAD
A + M[B-1] " HEAD + VALUE OF NEW ADDRESS
MG = A " FILL ADDRESS
S '*' 32767, Z " ANY OTHER ADDRESS TO BE FILLED?
N, JUMP(-8) " GOTO(:CHAIN)
B = 1
S = A
RUA (15)
U, A = END OF LIST, Z
Y, MG = - S
GOTO(L100P) " 19 INSTRUCTIONS

'END' SUBSTIT

PSEUDO LVAR: SUBC(:BITSTREAM18)
SUBC(OBJECTCODE) " NUMBER OF PROCEDURE
SUBC(OBJECTCODE) " CHEAPEST WAY OF ADDING 1
" TO INSTRUCT COUNTER HERE
S + BEGIN OF TRAFOTABLE " FILL TRAFOTABLE
A - 2 " WITH ADDRESS OF
MS = A " PSEUDO LVAR
GOTO(L100P) " 7 INSTRUCTIONS

INCREASE IC: SUBC(:BITSTREAM18)
INSTR CNTR + S
GOTO(L100P) " 3 INSTRUCTIONS

VAL DP0: SUBC(:BITSTREAM18)
S + CORRECTIE
DP0 = S
GOTO(L100P) " 4 INSTRUCTIONS

VAL BEGIN PR: SUBC(:BITSTREAM18)
S + CORRECTIE
BEGIN OF PROGRAM = S
GOTO(L100P) " 4 INSTRUCTIONS

VAL END PR: SUBC(:BITSTREAM18)
S + CORRECTIE
END OF PROGRAM = S
GOTO(L100P) " 4 INSTRUCTIONS

VAL RELADDRESS: SUBC(:BITSTREAM18)
RELADRSS = S

```
          GOTO(L100P)
SUBST ONE PAR18: S = INSTR CNTR
                  A = S1UBSTIT
                  GOTO(:MA[2])

SUBST ONE PAR9:  S = INSTR CNTR
                  MC= S
                  SUBC(BTSTREAM9)
                  A = S1UBSTIT
                  GOTO(:MA[4])
```

```
" 3 INSTRUCTIONS
"FIRST PAR FOR SUBSTIT
```

```
"FIRST PAR FOR SUBSTIT
"SECOND PAR FOR SUBSTIT
```

" MACRO LIST

MCR LST:STACK:	+ 12583168 ;	NEG:	+ 8401168 ;
	+ 4210978 ;		+ 4207154 ;
	+ 4215106 ;		+ 4195154 ;
	+ 4219136 ;		+ 4223232 ;
	+ 4227426 ;		+ 4227698 ;
	+ 4235906 ;		+ 4244114 ;
	+ 4244386 ;		+ 4256434 ;
	+ 10555904 ;		+ 8425728 ;
	+ 6369792 ;		+ 6377984 ;
	+ 6386176 ;		+ 6394368 ;
	+ 11334144 ;		+ 23183872 ;
	+ 23192064 ;		+ 23200256 ;
	+ 23208448 ;		+ 23216384 ;
	+ 8540416 ;		+ 8544512 ;
	+ 8548608 ;		+ 21135616 ;
	+ 21139712 ;		+ 21143808 ;
	+ 21147904 ;		+ 21152000 ;
	+ 21156096 ;		+ 25354496 ;
	+ 8581376 ;		+ 8585472 ;
	+ 8589568 ;		+ 8593664 ;
	+ 8597760 ;		+ 8601856 ;
	+ 8605952 ;		+ 8610048 ;
	+ 8614144 ;		+ 27492608 ;
	+ 8622336 ;		+ 8626432 ;
	+ 8630528 ;		+ 8634624 ;
	+ 8638720 ;		+ 4227328 ;
	+ 8643072 ;		+ 12845312 ;
	+ 10752256 ;		+ 10756352 ;
	+ 12857600 ;		+ 12861696 ;
	+ 12865792 ;		+ 17064192 ;
	+ 8679680 ;		+ 8683776 ;
	+ 21270784 ;		+ 21274880 ;
	+ 21278976 ;		+ 21283072 ;
	+ 21287168 ;		+ 21291264 ;
	+ 21295360 ;		+ 21299456 ;
	+ 8720640 ;		+ 8724736 ;
	+ 8728832 ;		+ 8732928 ;
	+ 8737024 ;	EMPTY:	+ 8388608 ;
	+ 8741376 ;		+ 8749824 ;
	+ 8761600 ;		+ 8765696 ;
	+ 8769792 ;		+ 8773888 ;
	+ 8777984 ;		+ 8783105 ;
	+ 8787217 ;		+ 8783149 ;
	+ 8787261 ;		+ 8791369 ;
	+ 8963328 ;	TEC:	+ 8967432 ;

" MACRO LIST CONTINUED

MCR LST[90]:	+ 8971520 ;	TLV:	+ 8975616 ;
	+ 8971520 ;		+ 8975620 ;
	+ 8983808 ;		+ 8990464 ;
	+ 8996096 ;		+ 9001472 ;
STST:	+ 8791552 ;		+ 9008384 ;
	+ 9012480 ;		+ 9016576 ;
JU:	+ 9020676 ;		+ 8975876 ;
	+ 9024768 ;		+ 9028864 ;
	'42352410' ;		'42362404' ;
SUBS:	+ 9041156 ;		'42402400' ;
	+ 9049352 ;		+ 9053452 ;
	+ 9058824 ;		+ 9065992 ;
	+ 30046984 ;		+ 9065736 ;
	+ 9086216 ;		+ 32159240 ;
NIL:	+ 9098500 ;	LAST:	+ 9102596 ;
	+ 9106952 ;		+ 9106696 ;
	+ 8791304 ;	TAA:	+ 8975616 ;
SWP:	+ 9114888 ;		+ 9225992 ;
	+ 9119240 ;	EXITSV:	+ 28001800 ;
CODE:	+ 9098504 ;		+ 9134336 ;
	+ 9138432 ;		+ 9142528 ;
	+ 9146624 ;		+ 762112 ;
	+ 766208 ;		+ 770304 ;
	+ 9163008 ;		+ 9167104 ;
	+ 9171200 ;		+ 9175296 ;
	+ 9179392 ;		+ 9183488 ;
	+ 9187584 ;		+ 9191680 ;
	+ 9195776 ;		+ 9202176 ;
	+ 9209344 ;		+ 9217544 ;

" TABLE OF OPTIMIZED MACROS

TABEL:	+ 8799488 ;	+ 8803584 ;
	+ 8799500 ;	+ 8803596 ;
	+ 8807688 ;	+ 8811776 ;
	+ 8815872 ;	+ 8811788 ;
	+ 8815884 ;	+ 8819976 ;
	+ 8824064 ;	+ 8828160 ;
	+ 8824076 ;	+ 8828172 ;
	+ 8832264 ;	+ 8836352 ;
	+ 8840448 ;	+ 8836364 ;
	+ 8840460 ;	+ 8844552 ;
	+ 8848640 ;	+ 8852736 ;
	+ 8848652 ;	+ 8852748 ;
	+ 8856840 ;	+ 8860928 ;
	+ 8869120 ;	+ 8860940 ;
	+ 8869132 ;	+ 8877320 ;
	+ 8861184 ;	+ 8869376 ;
	+ 8861196 ;	+ 8869388 ;
	+ 8877576 ;	+ 8886016 ;
	+ 8898304 ;	+ 8886028 ;
	+ 8898316 ;	+ 8910600 ;
	+ 8922624 ;	+ 8930816 ;
	+ 8922636 ;	+ 8930828 ;
	+ 8939016 ;	+ 8947200 ;
	+ 8955392 ;	+ 8946956 ;
	+ 8955148 ;	+ 8910088 ;
	+ 8885760 ;	+ 8898048 ;
	+ 8885772 ;	+ 8898060 ;
	+ 8910344 ;	

TRAFO[0]:

```

FUNC LTR:          '100 000 000'  " TRANSFORMS F = : STAT INTO F = : DYN
FUNC DIT:          '040 000 000'  " TRANSFORMS A = : STAT INTO A = : DYN
CVAR:              '006 000 000'  " TRANSFORMS A = STAT INTO A = DYN
END OF LIST:       + 64
BEGIN OF MCR LIST: (:MCR LST + : DELTA)  ")
BEGIN OF INSTR L ST: (:INSTR LST + : DELTA)  ")
L1OCP:             (:LOOP + : DELTA)  ")
L1OCP6:            (:LOOP6 + : DELTA)  ")
P1RCD:             (:PROD + : DELTA)  ")
O1BJECTCODE:       (:OBJECTCODE + : DELTA)  ")
I1MPPAR:           (:IMPPAR + : DELTA)  ")
D1YPAR:            (:DYPAR + : DELTA)  ")CCMPUTE REFERENCE
P1URPAR:           (:PURPAR + : DELTA)  ")DURING RUNTIME OF
L1ENTRY:           (:LENTY + : DELTA)  ")THIS SYSTEM
D1IRECT:           (:DIRECT + : DELTA)  ")
I1MPWORD:          (:IMPWORD + : DELTA)  ")
P1URWORD:          (:PURWORD + : DELTA)  ")
N1OT LISTED MACRO: (:NOTLISTEDMACRO+:DELTA)")
S1UBSTIT:          (:SUBSTIT + : DELTA)  ")
P1SEUDO LVAR:      (:PSEUDO LVAR + : DELTA)")
I1NCREASE IC:      (:INCREASE IC + : DELTA)")
V1AL DPO:          (:VAL DPO + : DELTA)  ")
V1AL BEGIN PR:     (:VAL BEGIN PR + :DELTA)")
V1AL END PR:       (:VAL END PR + : DELTA)  ")
V1AL RELADDRESS:   (:VAL RELADDRESS +:DELTA)
S1UBST ONE PAR18:  (:SUBST ONE PAR18 + : DELTA)")
S1UBST ONE PAR 9:  (:SUBST ONE PAR 9 + : DELTA)  ") 27 LOCATIONS

```

"LOADER SECTION NR LAST.

```

INSTR NBR:          S = A
                   RUS(8)

CLEAR:             S '*' 3
                   GOTOR(MC[-1])          " 4 INSTRUCTIONS

PAR PART:          S = A
                   RUS(10)
                   GOTO(:CLEAR)          " 3 INSTRUCTIONS

INSTR PRT:         S = A
                   RUS(12)
                   S '*' 511
                   GOTOR(MC[-1])          " 4 INSTRUCTIONS

KIND:              S = A
                   RUS(2)
                   GOTO(:CLEAR)          "3 INSTRUCTIONS

FREEW01:

```

'END' TRAF0

A = A

'END'

'END'

A = A

'END' COMPVAR

A = A

'END' SYSTEM

A = A

'END' SUBMOVAR

" THIS SECTION PUNCHES THE SYSTEM AND COMPILER TAPES AT THE END
" OF ASSEMBLY

'BEGIN' STIAR, STPAR, STISS, STPSS, SPBUF, SPAR, READBI, SKPBLK,
SNXTWD, SCW, SREHEP, SREWDR, SIPINT, INTINS,
WPINS, WAINS, KEYINS, CWP1, SIPCW1, SIPCW2, SPBBLK,
SPBLOK, SPWORD, SPBLNK, SPUHEP, ENDSYS

" ASSUMED TO BE DECLARED GLOBALLY IS: SYSTAP

" ASSUMED TO BE DECLARED AND DEFINED GLOBALLY ARE:

" REST, PUST, PRESENCE, DENTST, BEGIN OF STACK,

" END OF STACK, BEGIN OF PERMANENT, END OF PERMANENT,

" BEGIN OF COMPILER, END OF COMPILER, IOINTP, FINIS,

" WAITRPT

```

M[64+(4*:REST)]:  STIAR:
M[64+(4*:PUST)]:  STPAR:

```

```

M[0]:                      STISS: STPSS:

END CAT[0]:

SPBUF:  'SKIP' 1
SPAR:   'SKIP' 1

READBI: ('660 071 000' + :REST) " CLEAR INTERRUPT FROM IP-READER
        S=:STISS[1]
        STIAR[0]=S                " INITIALIZE INPUT APPARATUS REGISTER
        S=SCW
        STISS[2]=S                " CODE WORD TO READ ONE HEPTAD
SKPELK: SUB(:SREHEP)
        Y,JUMP(-2)                " SKIP BLANK TAPE
        SUB1(:SREWRD)             " READ WORD FROM TAPE IN A
        B=A,Z                    " BEGIN ADDRESS IN B, ALL DONE?
        Y,GOTO(:DENTST)           " INITIALIZE SYSTEM
        SUB1(:SREWRD)             " WORD FROM TAPE
        M[6]=A                   " NUMBER OF WORDS TO BE READ
SNXTWD: SUB1(:SREWRD)
        MC=A                      " STORE WORD
        REP6P(:SNXTWD)            " GET NEXT WORD
        GOTO(:SKPBLK)             " NEXT BLOCK OF WORDS
SCW:    ('001 000 000' + :STISS[0])

SREHEP: G=-0
        STIAR[1]=G                " INTERRUPT COUNTER = -1
        G = SCW
        STIAR[2]=G                " ACTION COUNTER = 1
        ('760 070 000' + :REST) " START TAPE READER
        G=STIAR[1],P              " IS THE HEPTAD PRESENT YET?
        N,JUMP(-2)                " WAIT FOR IT
        G=STISS[1],Z              " HEPTAD IN G, IS IT BLANK?
        GOTO(LINK)                " EXIT SREHEP

```

```

SREWRD:      S=4
              COUNT=S
              SUB(:SREHEP)
              LUSA(7)
              A='G
              REPP(:SREWRD[2])
              M[0]=A
              A = M[0]
              CLP
              Y,S'+1
              S'+1,Z
              N,GOTOR(LINK[1])
              S=MS[-2]
              " FOUR HEPTADS PER WORD
              " READ HEPTAD IN G
              " MAKE ROOM FOR NEW HEPTAD
              " ADD IT IN
              " GET NEXT HEPTAD
              " SET LAST PARITY INDICATOR
              " BY READING THE WORD FROM MEMORY
              " COPY LAST PARITY INTO CONDITION
              " PARITY ERROR?
              " EXIT SREWRD
              " ERROR STOP
ENDRBI:

```

" CONTROL IS TRANSFERRED TO SYSTAP AT THE END OF ASSEMBLY

```

SYSTAP:      S=INTINS
              M[24]=S
              S=WPINS
              M[25]=S
              S=WAINS
              M[26]=S
              S=KEVINS
              M[28]=S
              S=:STPSS[1]
              STPAR[0]=S
              S=CWP1
              STPSS[2]=S
              " I/O INTERRUPT ENTRY
              " WRONG PARITY ENTRY
              " WRONG ADDRESS INTERRUPT ENTRY
              " OPERATOR'S INTERRUPT ENTRY
              " PCINTER DURING SYSTEM PUNCHING
              " CCDE WORD TO PUNCH ONE WORD
              " INSERT IN CHAIN

              S=1
              SPAR=S
              SUB1(:SPBLNK)
              SUB1(:SPBLNK)
              A=SPCW1
              SUB1(:SPWORD)
              A=SPINT
              SUB1(:SPWORD)
              A=SPCW2
              SUB1(:SPWORD)
              B=:READBI
              G=:ENDRBI
              SUB2(:SPBLOK)
              SUB1(:SPBLNK)
              A=0
              SPAR=A
              SUB1(:SPWORD)
              " PARITY BIT FOR IP SECTION
              " MUST ALWAYS BE A 1
              " 100 BLANKS
              " ANOTHER 100
              " ONE WORD INTO LOCATION 24
              " PUNCH THE CODE WORD
              " THE ENTRY FOR M[24]
              " PUNCH IT
              " CCDE WORD TO LOAD BINARY LOADER
              " PUNCH IT
              " BEGIN ADDRESS OF BINARY READER
              " END ADDRESS OF BINARY LOADER
              " PUNCH BLOCK BETWEEN TWO ADDRESSES
              " 100 BLANKS

              B=:M[24]
              G=:M[28]
              SUB2(:SPHBLK)
              " PUNCH INTERRUPT ENTRY INSTRUCTIONS

              B = 512
              G = :ENDSYS
              SUB2(:SPHBLK)
              " PUNCH THE SYSTEM
              " ) ECCEPT FOR CATALOGUE
              " ) AND PART ABOVE 32 K

              SUB1(:SPBLNK)
              S = 127
              SUB(:SPUHEP)
              " 100 BLANKS
              " PUNCH BEGINMARKER

              A = PERMANENT CATALOGUE END

```

```

SUB1(:SPWORD)          " PLACE WANTED AT RUNTIME
A = BEGIN OF CATALOGUE
A = PERMANENT CATALOGUE END
A + 1                  " ) NUMBER OF WORDS
SUB1(:SPWORD)
B = :END OF PERMANENT CATALOGUE
G = :BEG CAT
SUB2(:SPBLOK)

B = :END OF PERMANENT CATALOGUE
G = :BEG CAT
SUB2(:SPHBLK)          " SHADOW

SUB1(:SPBLNK)          " 100 BLANKS
S = 127
SUB(:SPUHEP)           " PUNCH BEGINMARKER

A = BEGIN OF UP32K
SUB1(:SPWORD)

A = :END UP 32K
A = :BEG UP 32K
A + 1
SUB1(:SPWORD)          " PUNCH NUMBER OF WORDS

B = :BEG UP 32K
G = :END UP 32K
SUB2(:SPBLOK)          " PUNCH PART OF TRAFO ABOVE 32K

B = :BEGUP 32K
G = :ENDUP 32K
SUB2(:SPHBLK)          "PUNCH SHADOW OF TRAFO

SUB1(:SPBLNK)          " 100 BLANKS
S=127
SUB(:SPUHEP)           " PUNCH BEGIN MARKER
A = 0
SUB1(:SPWORD)          " END MARKER OF SYSTEM TAPE
SUB1(:SPBLNK)          " 100 BLANKS

SUB1(:SPBLNK)          " ANOTHER 100
A=-1
S=MA
GOTO(:DENTST)          " FULL STOP
                        " INITIALIZE SYSTEM

SIPINT:                GOTO(:READBI)
INTINS:                SUB15(:IOINTP)
WPINS:                DO(MT[-1])
WAINS:                SUB14(:WAINTRPT)
KEYINS:                GOTO(:FINIS)
CWP1:                 ('001 000 000'+:SPBUF[-1])
SIPCW1:                ('001 000 000'+:M(23))
SIPCW2:                (((:ENDRBI-:READBI[-1])*'001 000 000')+ :READBI[-1])

SPHELK:                SUB1(:SPBLNK)          " 100 BLANKS
                        S=127
                        SUB(:SPUHEP)          " PUNCH BEGIN MARKER
                        A=B

```

```
SPBLOK:      SUB1(:SPWORD)      " PUNCH BEGIN ADDRESS
              A=G
              A=B
              A+1
              SUB1(:SPWORD)      " PUNCH NUMBER OF WORDS
              G=B
              COUNT=G            " NUMBER OF WORDS -1
              A=M[B]            " WORD TO BE PUNCHED
              SUB1(:SPWORD)      " PUNCH IT
              B+1               " INCREMENT
              REPE(:SPBLOK[2])   " GET NEXT WORD
              GOTOR(LINK[2])     " EXIT SPBLOK AND SPBLOK
```

```

SPWORD:      M[0]=A          " SET LAST PARITY INDICATOR
              A=M[J]        " BY READING THE WORD FROM MEMORY
              CLP           " COPY LAST PARITY INTO CONDITION
              N,S=1
              Y,S=SPAR      " PARITY BIT IN S
              LCSA(6)       " FIRST SEVEN BITS IN S
              SUB(:SPUHEP)  " PUNCH THEM
              LCSA(7)       " SECCND SEVEN BITS
              SUB(:SPUHEP)  " THIRD SEVEN BITS
              LCSA(7)       " LAST SEVEN BITS
              SUB(:SPUHEP)  " EXIT SPWORD
              GOTOR(LINK[1])

SPBLNK:      S=100          " 100 TIMES
              COUNT=S       " BLANK
              S=0           " PUNCH IT
              SUB(:SPUHEP)  " RETURN FOR NEXT ONE
              REPP(:SPBLNK[2])
              GOTOR(LINK[1])

SPUHEP:      S, '*', 127    " HEPTAD INTO BUFFER
              SPBUF=S
              S=-0
              STPAR[1]=S    " INTERRUPT COUNTER = -1
              S=CWP1        " D18 + ...
              STPAR[2]=S    " ACTION COUNTER = 1
              ('760 070 000'+:PUST) " START PUNCH
              S=STPAR[1],P  " PUNCHING COMPLETED?
              N,JUMP(-2)    " WAIT FOR IT
              GOTOR(LINK)  " EXIT SPUHEP

ENDSYS:      'END'         " PUNCHING OF SYSTEM TAPES
              'END'         " ALGOL SYSTEM
              'START' :SYSTAP

```