(access)	<nine, bdata, 0276>	EXT	17T1
(accmask)	<nine, bdata, 0277>	EXT	17T2
(accum)	<nine, bdata, 0238>	EXT	17J1A
(acstring)	<nine, bdata, 088>	EXT STRING	15E2
(art)	<nine, bdata, 0371>	EXT	17Z3J
(arte)	<nine, bdata, 0372>	EXT	17Z3K
(astrng)	<nine, bdata, 091>	EXT STRING	15E5
(asub)	<nine, bdata, 0239>	EXT	17J1B
(aulsize)	<nine, bdata, 0359>	EXT	17Z2B
(aulsrt)	<nine, bdata, 0358>	EXT	17Z2A
(autostrt)	<nine, bdata, 0399>	EXT	17AB5
(bfilno)	<nine, bdata, 0386>	EXT	17AA1
(bittable)	<nine, bdata, 0618>	EXT	15D5
(bmarg)	<nine, bdata, 0570>	EXT	54C4
(bndchk)	<nine, bdata, 0233>	EXT	17G
(bugreg)	<nine, bdata, 0553>	EXT	53A
(cactlg)	<nine, bdata, 0252>	EXT	17J3C
(cadflg)	<nine, bdata, 0250>	EXT	17J3A
(calcaux)	<nine, bdata, 0455>	EXT	22K2
(calflg)	<nine, bdata, 0251>	EXT	17J3B
(carpos)	<nine, bdata, 0390>	EXT	17AA5
(castid)	<nine, bdata, 0247>	EXT	17J2C
(cbadent)	<nine, bdata, 0242>	EXT	17J1E
(cc)	<nine, bdata, 0368>	EXT	17Z3G
(cda)	<nine, bdata, 0245>	EXT	17J2A
(cdchct)	<nine, bdata, 0356>	EXT	17Z1K
(cdchrl)	<nine, bdata, 0355>	EXT	17Z1J
(cdpagf)	<nine, bdata, 0382>	EXT	17AQ2
(cdstd1)	<nine, bdata, 0556>	EXT	54A1
(cdstd2)	<nine, bdata, 0557>	EXT	54A2
(cdstop)	<nine, bdata, 0558>	EXT	54A3
(cdstr1)	<nine, bdata, 0540>	EXT STRING	45
(cdstr2)	<nine, bdata, 0541>	EXT STRING	46
(cdtype)	<nine, bdata, 0559>	EXT	54A4
(ckiwheel)	<nine, bdata, 0113>	EXT	15F21
(cm1tmp)	<nine, bdata, 043>	EXT	6A
(cm2mp)	<nine, bdata, 044>	EXT	6B
(cm3tmp)	<nine, bdata, 045>	EXT	6C
(cm4tmp)	<nine, bdata, 046>	EXT	6D
(cm5tmp)	<nine, bdata, 047>	EXT	6E
(cm6tmp)	<nine, bdata, 048>	EXT	6F
(cmarkflg)	<nine, bdata, 0565>	EXT	54B5
(cmntdl)	<nine, bdata, 0160>	EXT STRING	15G5
(coida)	<nine, bdata, 0201>	EXT	15K11
(coidda)	<nine, bdata, 0246>	EXT	17J2B
(colsw)	<nine, bdata, 0202>	EXT	15K12
(comchr)	<nine, bdata, 0470>	EXT	22K17
(commands)	<nine, bdata, 0374>	EXT	17Z3M
(confre)	<nine, bdata, 0478>	EXT	22K25
(conndir)	<nine, bdata, 0505>	EXT	33C2
(conreg)	<nine, bdata, 0542>	EXT STRING	47
(conrng)	<nine, bdata, 0475>	EXT	22K22
(console)	<nine, bdata, 055>	EXT	10A
(continue)	<nine, bdata, 0561>	EXT	54B1
(corpst)	<nine, bdata, 020>	EXT	4C3
(cpauxsw)	<nine, bdata, 0634>	EXT REF	57A6

(cpercount)	<nine, bdata, 0637>	EXT	57A9
(cpfkh)	<nine, bdata, 0629>	EXT	57A1
(cplstflno)	<nine, bdata, 0635>	EXT	57A7
(cpmsgdest)	<nine, bdata, 0632>	EXT	57A4
(cpojfn)	<nine, bdata, 0630>	EXT	57A2
(cpostr)	<nine, bdata, 0644>	EXT STRING	57B2
(cpptlag)	<nine, bdata, 0212>	EXT	15L1
(cppgcount)	<nine, bdata, 0636>	EXT	57A8
(cprcdjfn)	<nine, bdata, 0631>	EXT	57A3
(cpstate)	<nine, bdata, 0638>	EXT	57A11
(cpsw)	<nine, bdata, 0633>	EXT REF	57A5
(cpwdcount)	<nine, bdata, 0674>	EXT	57A10
(cshift)	<nine, bdata, 0491>	EXT	25
(cspcacode)	<nine, bdata, 0435>	EXT	19A6
(cspupdate)	<nine, bdata, 0432>	EXT	19A3
(cspusqcod)	<nine, bdata, 0436>	EXT	19A7
(cspvs)	<nine, bdata, 0437>	EXT	19A8
(cstid)	<nine, bdata, 0198>	EXT	15K8
(ctlstid)	<nine, bdata, 0100>	EXT	15F8
(cuno)	<nine, bdata, 0504>	EXT	33C1
(curindex)	<nine, bdata, 0477>	EXT	22K24
(curmkr)	<nine, bdata, 0430>	EXT TEXT POINTER	19A1
(curstk)	<nine, bdata, 0476>	EXT	22K23
(cversion)	<nine, bdata, 0209>	EXT	15K19
(cwndow)	<nine, bdata, 0590>	EXT	56A
(dabt)	<nine, bdata, 0514>	EXT	35A
(dabtsize)	<nine, bdata, 0517>	EXT	35D
(dacnt)	<nine, bdata, 0527>	EXT	40D
(datesr)	<nine, bdata, 0153>	EXT STRING	15F36D
(dblst)	<nine, bdata, 032>	EXT	4F1A
(dctsptr)	<nine, bdata, 0588>	EXT	55A3
(debugaccess)	<nine, bdata, 0280>	EXT	17T5
(dfoutm)	<nine, bdata, 0253>	EXT	17J3D
(dgbute)	<nine, bdata, 0379>	EXT	17Z3R
(dgstid)	<nine, bdata, 0378>	EXT	17Z3Q
(dgstl)	<nine, bdata, 0376>	EXT	17Z3O
(dgstml)	<nine, bdata, 0377>	EXT	17Z3P
(diststid)	<nine, bdata, 099>	EXT	15F7
(docjtn)	<nine, bdata, 0171>	EXT	15I4
(docstrt)	<nine, bdata, 0173>	EXT	15I6
(docwfn)	<nine, bdata, 0182>	EXT STRING	15I15
(donthelp)	<nine, bdata, 0214>	EXT	16A
(dpcmblk)	<nine, bdata, 0375>	EXT	17Z3N
(dpyarea)	<nine, bdata, 0524>	EXT	40A
(dpyend)	<nine, bdata, 0525>	EXT	40B
(drastic)	<nine, bdata, 0393>	EXT	17AA8
(dsparg)	<nine, bdata, 0610>	EXT	40M
(dspccf)	<nine, bdata, 0667>	EXT	40P
(dspcmd)	<nine, bdata, 0611>	EXT	40N
(dsplslist)	<nine, bdata, 0612>	EXT	40O
(dtable)	<nine, bdata, 0620>	EXT	15D6
(dtbl)	<nine, bdata, 0295>	EXT	17U12
(dtbst)	<nine, bdata, 0298>	EXT	17U15
(dttyda)	<nine, bdata, 0536>	EXT	42B
(dtwndw)	<nine, bdata, 0532>	EXT	40J
(echotg)	<nine, bdata, 0493>	EXT	27

(ecurmkr)	<nine, bdata, 0431>	EXT	19A2
(encsvl)	<nine, bdata, 0617>	EXT	15D4
(encsvacs)	<nine, bdata, 0615>	EXT	15D2
(encsvend)	<nine, bdata, 0616>	EXT	15D3
(entcon)	<nine, bdata, 0481>	EXT	22K28
(ercall)	<nine, bdata, 0387>	EXT	17AA2
(ermark)	<nine, bdata, 0388>	EXT	17AA3
(errwrk)	<nine, bdata, 0447>	EXT STRING	22H
(exp)	<nine, bdata, 0228>	EXT	17B
(exquery)	<nine, bdata, 0564>	EXT	54B4
(fakesrr)	<nine, bdata, 0434>	EXT	19A5
(fastname)	<nine, bdata, 0389>	EXT	17AA4
(fchsct)	<nine, bdata, 0331>	EXT	17Y3D
(fcredt)	<nine, bdata, 0285>	EXT	17U3
(filcnt)	<nine, bdata, 0271>	EXT	17R3
(filehead)	<nine, bdata, 021>	EXT	4C4
(filepart)	<nine, bdata, 022>	EXT	4C5
(filhde)	<nine, bdata, 0303>	EXT	17U19
(filhed)	<nine, bdata, 0283>	EXT	17U2
(flist)	<nine, bdata, 0270>	EXT	17R1
(finit)	<nine, bdata, 0288>	EXT	17U6
(flagjfn)	<nine, bdata, 0101>	EXT	15F9
(fopncnt)	<nine, bdata, 0624>	EXT	17R2
(freemap)	<nine, bdata, 0323>	EXT	17X1
(freesz)	<nine, bdata, 07>	EXT CONSTANT =154B	2B1
(frrcnt)	<nine, bdata, 0438>	EXT	19A9
(frzmax)	<nine, bdata, 0551>	EXT	52D
(ftpem)	<nine, bdata, 0186>	EXT	15J2
(ftperm)	<nine, bdata, 0187>	EXT	15J3
(ftphnd)	<nine, bdata, 0185>	EXT	15J1
(ftpoh)	<nine, bdata, 0188>	EXT	15J4
(ftpoha)	<nine, bdata, 0189>	EXT	15J5
(funo)	<nine, bdata, 0289>	EXT	17U7
(fvstk)	<nine, bdata, 0546>	EXT STACK	51
(fzfree)	<nine, bdata, 0550>	EXT	52C
(fzlste)	<nine, bdata, 0549>	EXT	52B
(fzlsts)	<nine, bdata, 0548>	EXT	52A
(gapbp)	<nine, bdata, 0366>	EXT	17Z3E
(gapcc)	<nine, bdata, 0367>	EXT	17Z3F
(gapcnt)	<nine, bdata, 0365>	EXT	17Z3D
(gapcol)	<nine, bdata, 0364>	EXT	17Z3C
(gprint)	<nine, bdata, 0622>	EXT	15D8
(groupx)	<nine, bdata, 05>	EXT CONSTANT =4	2A1
(gtjfnt)	<nine, bdata, 0342>	EXT	17Y14
(hdebug)	<nine, bdata, 0487>	EXT	22K34
(hdrjfn)	<nine, bdata, 0172>	EXT	15I5
(hdrwfn)	<nine, bdata, 0183>	EXT STRING	15I16
(helplist)	<nine, bdata, 0626>	EXT	41A
(helploc)	<nine, bdata, 0454>	EXT	22K1
(hinc)	<nine, bdata, 0353>	EXT	17Z1H
(hlpfileno)	<nine, bdata, 0458>	EXT	22K5
(holdvs)	<nine, bdata, 0666>	EXT	17A
(hostni)	<nine, bdata, 0229>	EXT	17C
(hostsi)	<nine, bdata, 0230>	EXT	17D
(hseger)	<nine, bdata, 0466>	EXT	22K13
(idcident)	<nine, bdata, 0116>	EXT	15F24

(idcrec)	<nine, bdata, 0114>	EXT	15F22
(idcrtype)	<nine, bdata, 0115>	EXT	15F23
(identstr)	<nine, bdata, 0156>	EXT STRING	15G1
(identwheel)	<nine, bdata, 0163>	EXT	15G8
(idfcnt)	<nine, bdata, 0111>	EXT	15F19
(idfile)	<nine, bdata, 0178>	EXT	15I11
(idfno)	<nine, bdata, 0112>	EXT	15F20
(idirn)	<nine, bdata, 0105>	EXT	15F13
(idmodflag)	<nine, bdata, 0118>	EXT	15F26
(idmodified)	<nine, bdata, 0119>	EXT	15F27
(idnamdel)	<nine, bdata, 0161>	EXT STRING	15G6
(idnfname)	<nine, bdata, 0676>	EXT STRING	58A
(idntdel)	<nine, bdata, 0159>	EXT STRING	15G4
(idwork)	<nine, bdata, 0157>	EXT STRING	15G2
(ijfn)	<nine, bdata, 083>	EXT	15C5
(infacs)	<nine, bdata, 0621>	EXT	15D7
(infopn)	<nine, bdata, 0197>	EXT	15K7
(infork)	<nine, bdata, 085>	EXT	15C7
(inipc)	<nine, bdata, 0614>	EXT	15D1
(inhelp)	<nine, bdata, 0456>	EXT	22K3
(initsr)	<nine, bdata, 0489>	EXT STRING	23A
(inpstp)	<nine, bdata, 028>	EXT	4E3
(inptrf)	<nine, bdata, 026>	EXT	4E1
(interflag)	<nine, bdata, 093>	EXT	15F1
(intmsf)	<nine, bdata, 057>	EXT	11A
(intmsg)	<nine, bdata, 058>	EXT STRING	11B
(ipassw)	<nine, bdata, 0150>	EXT STRING	15F36A
(iswork)	<nine, bdata, 0340>	EXT	17Y11
(jb1)	<nine, bdata, 0147>	EXT TEXT POINTER	15F35C
(jb2)	<nine, bdata, 0148>	EXT TEXT POINTER	15F35D
(jcatstid)	<nine, bdata, 097>	EXT	15F5
(jdebug)	<nine, bdata, 0102>	EXT	15F10
(jdfi)	<nine, bdata, 0110>	EXT	15F18
(jdid)	<nine, bdata, 0109>	EXT	15F17
(jdirn)	<nine, bdata, 0104>	EXT	15F12
(jdno)	<nine, bdata, 0108>	EXT	15F16
(jfnetin)	<nine, bdata, 050>	EXT	7A
(jidshot)	<nine, bdata, 0164>	EXT	15G9
(jidstk)	<nine, bdata, 0162>	EXT STACK	15G7
(jloggfn)	<nine, bdata, 0141>	EXT	15F32
(jnamstr)	<nine, bdata, 0154>	EXT STRING	15F36E
(jniprog)	<nine, bdata, 0398>	EXT	17AB4
(jnisbn)	<nine, bdata, 0522>	EXT	39
(jobtbl)	<nine, bdata, 0232>	EXT	17F
(jokludgeupdtfl)	<nine, bdata, 0142>	EXT	15F33
(jpassw)	<nine, bdata, 0151>	EXT STRING	15F36B
(jrnlaccess)	<nine, bdata, 0279>	EXT	17T4
(jrnlstid)	<nine, bdata, 098>	EXT	15F6
(jsavaccess)	<nine, bdata, 0143>	EXT	15F34
(jt0)	<nine, bdata, 0124>	EXT	15F30B
(jt1)	<nine, bdata, 0125>	EXT	15F30C
(jt10)	<nine, bdata, 0134>	EXT	15F30L
(jt11)	<nine, bdata, 0135>	EXT	15F30M
(jt12)	<nine, bdata, 0136>	EXT	15F30N
(jt13)	<nine, bdata, 0137>	EXT	15F30O
(jt14)	<nine, bdata, 0138>	EXT	15F30P

(jt15)	<nine, bdata, 0139>	EXT	15F30Q
(jt2)	<nine, bdata, 0126>	EXT	15F30D
(jt3)	<nine, bdata, 0127>	EXT	15F30E
(jt4)	<nine, bdata, 0128>	EXT	15F30F
(jt5)	<nine, bdata, 0129>	EXT	15F30G
(jt6)	<nine, bdata, 0130>	EXT	15F30H
(jt7)	<nine, bdata, 0131>	EXT	15F30I
(jt8)	<nine, bdata, 0132>	EXT	15F30J
(jt9)	<nine, bdata, 0133>	EXT	15F30K
(jtimfg)	<nine, bdata, 0123>	EXT	15F30A
(jworkstid)	<nine, bdata, 096>	EXT	15F4
(jwpl)	<nine, bdata, 0145>	EXT TEXT POINTER	15F35A
(keypr)	<nine, bdata, 0191>	EXT	15K1
(ladj)	<nine, bdata, 0255>	EXT STRING	17L
(lastch)	<nine, bdata, 0346>	EXT	17Z1A
(lastfnum)	<nine, bdata, 0175>	EXT	15I8
(lavtabad)	<nine, bdata, 0107>	EXT	15F15
(ldrfct)	<nine, bdata, 0328>	EXT	17Y3A
(ldrnct)	<nine, bdata, 0329>	EXT	17Y3B
(ldsdct)	<nine, bdata, 0330>	EXT	17Y3C
(leftover)	<nine, bdata, 0643>	EXT STRING	57B1
(levind)	<nine, bdata, 0321>	EXT	17W1
(libfig)	<nine, bdata, 0396>	EXT	17AB2
(libiod)	<nine, bdata, 0397>	EXT	17AB3
(lit)	<nine, bdata, 0446>	EXT STRING	22F
(lkupreq)	<nine, bdata, 0258>	EXT STRING	17C
(lmarg)	<nine, bdata, 0568>	EXT	54C2
(logstr)	<nine, bdata, 0665>	EXT STRING	23B
(lptjfn)	<nine, bdata, 0169>	EXT	15I2
(lptused)	<nine, bdata, 0170>	EXT	15I3
(lsbitables)	<nine, bdata, 0515>	EXT	35B
(lsbtsize)	<nine, bdata, 0516>	EXT	35C
(lstred)	<nine, bdata, 0235>	EXT	17I
(lwtim)	<nine, bdata, 0291>	EXT	17U8
(mastacprintflag)	<nine, bdata, 0177>	EXT	15I10
(measnt)	<nine, bdata, 0339>	EXT	17Y10
(menchr)	<nine, bdata, 0469>	EXT	22K16
(mentyp)	<nine, bdata, 0471>	EXT	22K18
(merrff)	<nine, bdata, 0195>	EXT	15K5
(mersiz)	<nine, bdata, 0194>	EXT	15K4
(mkrcnt)	<nine, bdata, 0350>	EXT	17Z1E
(mkrend)	<nine, bdata, 0348>	EXT	17Z1C
(mkrtfg)	<nine, bdata, 0351>	EXT	17Z1F
(mkrptr)	<nine, bdata, 0349>	EXT	17Z1D
(mkrtb)	<nine, bdata, 0302>	EXT	17U18
(mkrtbl)	<nine, bdata, 0300>	EXT	17U17
(mkrtxn)	<nine, bdata, 0299>	EXT	17U16
(modeoff)	<nine, bdata, 035>	EXT	4G1
(moremen)	<nine, bdata, 0474>	EXT	22K21
(msaddr)	<nine, bdata, 0333>	EXT	17Y4
(msbufe)	<nine, bdata, 0425>	EXT	18C
(msbuff)	<nine, bdata, 0424>	EXT	18B
(msflag)	<nine, bdata, 064>	EXT	13C
(msist)	<nine, bdata, 0334>	EXT	17Y5
(msiste)	<nine, bdata, 0335>	EXT	17Y6
(msrlsv)	<nine, bdata, 0336>	EXT	17Y7

(msr2sv)	<nine, bdata, 0337>	EXT	17Y8
(msr3sv)	<nine, bdata, 0338>	EXT	17Y9
(msstcount)	<nine, bdata, 062>	EXT	13A
(mstxcount)	<nine, bdata, 063>	EXT	13B
(mswsic)	<nine, bdata, 0326>	EXT	17Y2
(mswsit)	<nine, bdata, 0325>	EXT	17Y1
(multiflg)	<nine, bdata, 0467>	EXT	22K14
(multyp)	<nine, bdata, 0472>	EXT	22K19
(nambuf)	<nine, bdata, 0206>	EXT	15K16
(namdl1)	<nine, bdata, 0292>	EXT	17U9
(namdl2)	<nine, bdata, 0293>	EXT	17U10
(namist)	<nine, bdata, 0205>	EXT	15K15
(namndx)	<nine, bdata, 0204>	EXT	15K14
(namreg)	<nine, bdata, 0545>	EXT STRING	50
(newid)	<nine, bdata, 0158>	EXT STRING	15G3
(newstk)	<nine, bdata, 0479>	EXT	22K26
(ninfil)	<nine, bdata, 0207>	EXT	15K17
(nlcret)	<nine, bdata, 0450>	EXT	22J2
(nlcrist)	<nine, bdata, 0451>	EXT	22J3
(nlcrrt)	<nine, bdata, 0449>	EXT	22J1
(nlmode)	<nine, bdata, 0562>	EXT	54B2
(nlparse)	<nine, bdata, 0563>	EXT	54B3
(nlsdas)	<nine, bdata, 0526>	EXT	40C
(nlsloader)	<nine, bdata, 0587>	EXT	55A2
(nlsnam)	<nine, bdata, 0539>	EXT STRING	44
(nlssbn)	<nine, bdata, 0521>	EXT	38
(nlstyp)	<nine, bdata, 0452>	EXT	22J4
(nlsvwd)	<nine, bdata, 0286>	EXT	17U4
(nodisplay)	<nine, bdata, 0215>	EXT	16B
(noteedbk)	<nine, bdata, 0243>	EXT	17J1F
(normlaccess)	<nine, bdata, 0278>	EXT	17T3
(nortnrings)	<nine, bdata, 0216>	EXT	16C
(nsdcktime)	<nine, bdata, 0106>	EXT	15F14
(num)	<nine, bdata, 0442>	EXT STRING	22B
(numstid)	<nine, bdata, 0166>	EXT	15H1
(nwrecflag)	<nine, bdata, 0117>	EXT	15F25
(nXpg)	<nine, bdata, 019>	EXT	4C2
(ojfn)	<nine, bdata, 082>	EXT	15C4
(ojnestr)	<nine, bdata, 051>	EXT STRING	7B
(ojsqsw)	<nine, bdata, 0140>	EXT	15F31
(oldflg)	<nine, bdata, 0395>	EXT	17AB1
(oldlsrt)	<nine, bdata, 0373>	EXT	17Z3L
(oljmlav)	<nine, bdata, 0120>	EXT	15F28
(oljsbn)	<nine, bdata, 0121>	EXT	15F29
(opbp)	<nine, bdata, 0404>	EXT	17AC2
(opcbp)	<nine, bdata, 0405>	EXT	17AC3
(opccpos)	<nine, bdata, 0406>	EXT	17AC4
(opcmax)	<nine, bdata, 0407>	EXT	17AC5
(oprloat)	<nine, bdata, 0241>	EXT	17J1D
(oplev)	<nine, bdata, 0408>	EXT	17AC6
(opnewst)	<nine, bdata, 0409>	EXT	17AC7
(oprocn)	<nine, bdata, 0179>	EXT STRING	15I12
(oprtty)	<nine, bdata, 0234>	EXT	17H
(oprwrk)	<nine, bdata, 0341>	EXT	17Y12
(opstr)	<nine, bdata, 0181>	EXT STRING	15I14
(opstring)	<nine, bdata, 087>	EXT STRING	15E1

(oqapfg)	<nine, bdata, 0319>	EXT	17V15
(oqnhfg)	<nine, bdata, 0318>	EXT	17V14
(oshift)	<nine, bdata, 0492>	EXT	26
(outcnt)	<nine, bdata, 0199>	EXT	15K9
(outnam)	<nine, bdata, 0210>	EXT STRING	15K20
(overscreen)	<nine, bdata, 0465>	EXT	22K12
(p10)	<nine, bdata, 0268>	EXT TEXT POINTER	17Q8
(p3)	<nine, bdata, 0261>	EXT TEXT POINTER	17Q1
(p4)	<nine, bdata, 0262>	EXT TEXT POINTER	17Q2
(p5)	<nine, bdata, 0263>	EXT TEXT POINTER	17Q3
(p6)	<nine, bdata, 0264>	EXT TEXT POINTER	17Q4
(p7)	<nine, bdata, 0265>	EXT TEXT POINTER	17Q5
(p8)	<nine, bdata, 0266>	EXT TEXT POINTER	17Q6
(p9)	<nine, bdata, 0267>	EXT TEXT POINTER	17Q7
(pageno)	<nine, bdata, 0383>	EXT	17A@3
(patch)	<nine, bdata, 0420>	EXT	17AF
(patche)	<nine, bdata, 0421>	EXT	17AG
(pfilnum)	<nine, bdata, 0174>	EXT	15I7
(pjfn)	<nine, bdata, 084>	EXT	15C6
(pjisw)	<nine, bdata, 0417>	EXT	17AE5
(pjrbab)	<nine, bdata, 0413>	EXT	17AE1
(pjrubout)	<nine, bdata, 0418>	EXT	17AE6
(pjsavf)	<nine, bdata, 0415>	EXT	17AE3
(pjstid)	<nine, bdata, 0419>	EXT TEXT POINTER	17AE7
(pjsw)	<nine, bdata, 0416>	EXT	17AE4
(pjusqc)	<nine, bdata, 0414>	EXT	17AE2
(pmtjfn)	<nine, bdata, 0600>	EXT	17X2
(prbuf)	<nine, bdata, 0259>	EXT STRING	17P
(prmkrf)	<nine, bdata, 0381>	EXT	17A@1
(prntfg)	<nine, bdata, 0176>	EXT	15I9
(proc)	<nine, bdata, 0248>	EXT	17J2D
(prseqwk)	<nine, bdata, 0403>	EXT	17AC1
(prwndw)	<nine, bdata, 0531>	EXT	40H
(psiac1)	<nine, bdata, 0606>	EXT	21A
(psiace)	<nine, bdata, 0608>	EXT	21C
(psiacs)	<nine, bdata, 0607>	EXT	21B
(psiskszs)	<nine, bdata, 0498>	EXT CONSTANT =100	31A
(psistk)	<nine, bdata, 0512>	EXT	34A
(ptstr)	<nine, bdata, 0180>	EXT STRING	15I13
(qagain)	<nine, bdata, 0457>	EXT	22K4
(qcolwidth)	<nine, bdata, 0486>	EXT	22K33
(qda)	<nine, bdata, 0459>	EXT	22K6
(qdavs2)	<nine, bdata, 0464>	EXT	22K11
(qdavspc)	<nine, bdata, 0463>	EXT	22K10
(qmenucnt)	<nine, bdata, 0485>	EXT	22K32
(qmenumax)	<nine, bdata, 0484>	EXT	22K31
(qnewstmt)	<nine, bdata, 0462>	EXT	22K9
(qpspecs)	<nine, bdata, 0483>	EXT	22K30
(qspecs)	<nine, bdata, 0482>	EXT	22K29
(qstorblk)	<nine, bdata, 0461>	EXT	22K8
(qsw)	<nine, bdata, 0460>	EXT	22K7
(rfbs)	<nine, bdata, 0296>	EXT	17U13
(rfcfig)	<nine, bdata, 0103>	EXT	15F11
(rfcnum)	<nine, bdata, 0152>	EXT STRING	15F36C
(rfpmin)	<nine, bdata, 018>	EXT	4C1
(rjumpindex)	<nine, bdata, 053>	EXT	9A

(stac2)	<nine, bdata, 0655>	EXT	20C
(stac3)	<nine, bdata, 0656>	EXT	20D
(stac4)	<nine, bdata, 0657>	EXT	20E
(stac5)	<nine, bdata, 0658>	EXT	20F
(stac6)	<nine, bdata, 0650>	EXT	20G
(stac7)	<nine, bdata, 0651>	EXT	20H
(stac8)	<nine, bdata, 0652>	EXT	20I
(stac9)	<nine, bdata, 0653>	EXT	20J
(stacs)	<nine, bdata, 0595>	EXT	20A
(state)	<nine, bdata, 0441>	EXT	22A
(stn)	<nine, bdata, 0443>	EXT STRING	22C
(stn2)	<nine, bdata, 0444>	EXT STRING	22D
(stno)	<nine, bdata, 0445>	EXT STRING	22E
(strpkey)	<nine, bdata, 0208>	EXT	15K18
(strttm)	<nine, bdata, 0332>	EXT	17Y3E
(subcnt)	<nine, bdata, 0411>	EXT	17AD1
(subdfdir)	<nine, bdata, 0677>	EXT STRING	58B
(subhed)	<nine, bdata, 0274>	EXT	17S2
(subnml)	<nine, bdata, 010>	EXT CONSTANT =120	2C2
(subnum)	<nine, bdata, 09>	EXT CONSTANT =30	2C1
(subrec)	<nine, bdata, 0273>	EXT	17S1
(symloc)	<nine, bdata, 0586>	EXT	55A1
(sysvsp)	<nine, bdata, 060>	EXT	12A
(tabase)	<nine, bdata, 0572>	EXT	54C5A
(tabottom)	<nine, bdata, 0576>	EXT	54C5E
(tacsize)	<nine, bdata, 0577>	EXT	54C5F
(tahinc)	<nine, bdata, 0579>	EXT	54C5H
(taleft)	<nine, bdata, 0573>	EXT	54C5B
(tamind)	<nine, bdata, 0583>	EXT	54C5L
(taright)	<nine, bdata, 0574>	EXT	54C5C
(tatop)	<nine, bdata, 0575>	EXT	54C5D
(tavinc)	<nine, bdata, 0580>	EXT	54C5I
(tax)	<nine, bdata, 0581>	EXT	54C5J
(tay)	<nine, bdata, 0582>	EXT	54C5K
(tda)	<nine, bdata, 0537>	EXT	42C
(tiw)	<nine, bdata, 079>	EXT	15C1
(tlit)	<nine, bdata, 0678>	EXT STRING	22G
(tmarg)	<nine, bdata, 0569>	EXT	54C3
(tmode)	<nine, bdata, 080>	EXT	15C2
(tmpbug)	<nine, bdata, 0146>	EXT TEXT POINTER	15F35B
(tmpfile)	<nine, bdata, 0604>	EXT STRING	8A
(tmplt)	<nine, bdata, 094>	EXT	15F2
(topcon)	<nine, bdata, 0480>	EXT	22K27
(trmcod)	<nine, bdata, 081>	EXT	15C3
(tskerrcnt)	<nine, bdata, 068>	EXT	15A1
(tstrng)	<nine, bdata, 090>	EXT STRING	15E4
(ttyda)	<nine, bdata, 0535>	EXT	42A
(ttytbl)	<nine, bdata, 0231>	EXT	17E
(typend)	<nine, bdata, 0343>	EXT	17Y15
(udpnwrap)	<nine, bdata, 0664>	EXT	17Z3B
(ukcstr)	<nine, bdata, 0662>	EXT STRING	22I
(untyp)	<nine, bdata, 0473>	EXT	22K20
(uojfn)	<nine, bdata, 0519>	EXT	36A
(upgbend)	<nine, bdata, 072>	EXT	15B3
(upgbsz)	<nine, bdata, 073>	EXT	15B4
(upgbuf)	<nine, bdata, 070>	EXT	15B1

(upgcacnt)	<nine, bdata, 074>	EXT	15B5
(upgfbuf)	<nine, bdata, 071>	EXT	15B2
(upgnms)	<nine, bdata, 077>	EXT STRING	15B8
(upgskix)	<nine, bdata, 075>	EXT	15B6
(upgsksz)	<nine, bdata, 012>	EXT CONSTANT =20	2D1
(upgstk)	<nine, bdata, 076>	EXT	15B7
(usesrr)	<nine, bdata, 0433>	EXT	19A4
(utiitv)	<nine, bdata, 0401>	EXT	17AB7
(vaccum)	<nine, bdata, 0240>	EXT	17J1C
(vcvtp)	<nine, bdata, 0193>	EXT	15K3
(vcvulp)	<nine, bdata, 0192>	EXT	15K2
(vinc)	<nine, bdata, 0354>	EXT	17Z1I
(vrsnno)	<nine, bdata, 0254>	EXT STRING	17K
(vspsav)	<nine, bdata, 0599>	EXT	40L
(vspstr)	<nine, bdata, 0544>	EXT STRING	49
(vswndw)	<nine, bdata, 0598>	EXT	40I
(vtop)	<nine, bdata, 0196>	EXT	15K6
(wacnt)	<nine, bdata, 0530>	EXT	40G
(wndwarea)	<nine, bdata, 0528>	EXT	40E
(wndwend)	<nine, bdata, 0529>	EXT	40F
(wrdreg)	<nine, bdata, 0543>	EXT STRING	48
(wsflag)	<nine, bdata, 0591>	EXT	56B
(wsvalu)	<nine, bdata, 0592>	EXT	56C
(xacs)	<nine, bdata, 0496>	EXT	30
(xsmode)	<nine, bdata, 0508>	EXT	33D1

```
< NINE, BDATA.NLS;16, >, 12-Jul-78 13:19 HGL ;;;;
FILE bdata   % (arcsubsys,XL10,) (arcsubsys,1109,) (RELNINE,bdata.rel,)
%

% stack sizes %
   %call and multiple result stack lengths%
      (groupx) EXTERNAL CONSTANT =  4;     %maximum group number%     2A1
   %Number of pages to be mapped out for processors%
      (freesz) EXTERNAL CONSTANT = 154B;                              2B1
   %substitute%
      (subnum) EXTERNAL CONSTANT = 30;                                2C1
      (subnm1) EXTERNAL CONSTANT = 120; %=lcard*subnum -- see (UTILTY,
      card)%                                                          2C2
   % for stack of user program addresses %
      (upgsksz) EXTERNAL CONSTANT = 20;                               2D1
% *** GLOBAL DATA FOR PAGE 0 *** %
% Special stuff declared in page 0, loaded at 140B %
   %...KEEPTHESE THINGS TOGETHER -- DONT ADD NEW DATA HERE...%
   %...important stuff...%
   %...file block IO...%
      (ritpmin) EXTERNAL = 7;    % Index of lowest file page this
      constant must be changed when the default buffer size is changed
      %                                                               4C1
      (nxpg) EXTERNAL;           %index into corpst for next page%    4C2
      (corpst) EXTERNAL [100B];   %core page status table%            4C3
      (filehead) EXTERNAL [26];   %array of filhdr(fileno) values%    4C4
      (filepart) EXTERNAL [26];   %array of -- TRUE if partial copy for
      file%                                                           4C5
   %...often used work areas...%
      (s2work) EXTERNAL [7]; % second string reading work area %      4D1
   %...seqgen flags...%
      (inptrf) EXTERNAL;  %rubout and ^O flag%                        4E1
      (rubmrk) EXTERNAL = $inptrf;                                    4E2
         %rubmrk is used to indirectly set a location to TRUE on ^O%
      (inpstp) EXTERNAL; % ^S flag %                                  4E3
      (rubabt) EXTERNAL;          %true if rubout causes abort%       4E4
         %rubabt IS NO LONGER NEEDED - IT WAS USED FOR FE FUNCTIONS%
   %...editing global variables...%
      %...for new ring elements or data blocks...%
         (dblst) EXTERNAL;   % index of last used data file block % 4F1A
         (rnglst) EXTERNAL;  % index of last used ring file block % 4F1B
   %...set command and aptstr routine...%
      (modeoff) EXTERNAL=177B;  %  option set  %                      4G1
% sequence generator declarations %
   (sqgwas) EXTERNAL [165];      % 15 sequence generator work areas:
   sqwrkl long %                                                      5A
   (sqgaend) EXTERNAL;                                                5B
   (sqsvws) EXTERNAL [130]; % 2 statement vector work areas; each is
   svmxlev + 1 words long %                                           5C

% TEMPS AVAILABLE TO ANY COMMAND %
   (cm1tmp) EXTERNAL ;                                                6A
   (cm2mp) EXTERNAL ;                                                 6B
   (cm3tmp) EXTERNAL ;                                                6C
   (cm4tmp) EXTERNAL ;                                                6D
   (cm5tmp) EXTERNAL ;                                                6E
```

```
   (cm6tmp) EXTERNAL ;                                                  6F
% jump name external file stid and name string %
   (jfnefin) EXTERNAL = 0;                                              7A
   (ojnestr) EXTERNAL STRING [200];                                     7B
% programmer's template file name string %
   (tmpfile) EXTERNAL STRING [80];                                      8A
% index for stepping through return rings%
   (rjumpindex) EXTERNAL;                                               9A
% terminal stuff %
   (console) EXTERNAL; % octal console number %                         10A
   (simtty) EXTERNAL; %TRUE if tty is simulated %                       10B
% Initialization message %
   (intmsf) EXTERNAL = 0;  %flag for message at initialization time%    11A
   (intmsg) EXTERNAL STRING [500]; %String for entry message%           11B
% Viewspecs %
   (sysvsp) EXTERNAL [2];                                               12A
%%
   (msstcount) EXTERNAL = 0;                                            13A
   (mstxcount) EXTERNAL = 0;                                            13B
   (mstlag) EXTERNAL = 0;  %indicates whether use measurements being
   recorded%                                                            13C
% display core block requirements    %
% Declarations for NLS submodes%
   % NLS utility subsystem %
      (tskerrcnt) EXTERNAL;     % cnt of errors for run tasks %        15A1
   % user program stuff %
      (upgbuf) EXTERNAL = 554000B;   % start of user program buffer %
                                                                        15B1
      (upgffbuf) EXTERNAL = 554000B;  % first free cell in upgbuf %    15B2
      (upgbend) EXTERNAL = 561777B;  % current end of user program
      buffer %                                                          15B3
      (upgbsz) EXTERNAL = $upgbdf;  % current size of user program
      buffer (in pages) %                                               15B4
      (upgcacnt) EXTERNAL ;  % number of Content analyzer patterns
      compiled %                                                        15B5
      (upgskix) EXTERNAL ;   % index into upgstk %                     15B6
      (upgstk) EXTERNAL [upgsksz];   % stack of addresses of user
      programs %                                                        15B7
      (upgnms) EXTERNAL STRING [200];   % string to hold names of user
      programs %                                                        15B8
   % run tenex subsystem stuff %
      (tiw) EXTERNAL = 0;       % saved terminal interrupt word for
      this fork %                                                       15C1
      (tmode) EXTERNAL = 0;     % saved terminal mode word %          15C2
      (trmcod) EXTERNAL = 0;    % termination char terminal code %    15C3
      (ojfn) EXTERNAL = 0;      % -1 or output file jfn %             15C4
      (ijfn) EXTERNAL = 0;      % -1 or input file jfn %              15C5
      (pjfn) EXTERNAL = 0;      % jfn for the program %               15C6
      (infork) EXTERNAL = 0;    % fork handle of inferior tenex
      subsystem %                                                       15C7
   % encapsulator stuff %
      (infpc) EXTERNAL;         % PC of inferior fork when frozen by
      jsys trap%                                                        15D1
      (encsvacs) EXTERNAL [16]; % saving registers of this fork %      15D2
      (encsvend) EXTERNAL;      % CAUTION: must follow definition of
      encsvacs %                                                        15D3
```

```
   (encsv1) EXTERNAL;          % saving ac 1 %                          15D4
   (bittable) EXTERNAL _       % for indicating which jsys's to trap %
                                                                       15D5
      (0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 );
   (dtable) EXTERNAL [256];  % dispatch table for jsys handlers %
                                                                       15D6
   (infacs) EXTERNAL [16];    % ACS for inferior fork %                15D7
   (gprint) EXTERNAL;         %for encapsulator debugging%           15D8
% CALCULATOR strings %
   (opstring) EXTERNAL STRING [100];   %char represetation; current
   operand%                                                            15E1
   (acstring) EXTERNAL STRING [100];   %char representation; current
   accum%                                                              15E2
   (signstr) EXTERNAL STRING [5];   %number of accum used as
   operand%                                                            15E3
   (tstrng) EXTERNAL STRING [5];    %number cfaccum used as operand%
                                                                       15E4
   (astrng) EXTERNAL STRING [5];    %number of the accumulator being
   used%                                                               15E5
%Journal%
   (interflag) EXTERNAL ; %interrogate mode flag%                      15F1
   (tmplt) EXTERNAL ; %template submission flag%                       15F2
   (sabtfg) EXTERNAL ; %submit abort flag%                             15F3
   (jworkstid) EXTERNAL ; %stid of jwork file%                         15F4
   (jcatstid) EXTERNAL ; %stid of jcat%                                15F5
   (jrnlstid) EXTERNAL ; %Journal message file%                        15F6
   (diststid) EXTERNAL ; %distributionn file%                          15F7
   (ctlstid) EXTERNAL ; %stid for on-line distribution control
   files%                                                              15F8
   (flagjfn) EXTERNAL = 0 ; % jfn of nls flag file
   <netsys>nlsflags.flags %                                            15F9
   (jdebug) EXTERNAL  = 1; %true for debugginh...files go under
   duvall%                                                             15F10
   (rfcflg) EXTERNAL ; %True if RFC number has been assignnd%          15F11
   (jdirn) EXTERNAL ; %for saving number of connected directory
   (journal)%                                                          15F12
   (idirn) EXTERNAL ; %for saving number of connected directory
   (ident)%                                                            15F13
   (nsdcktime) EXTERNAL ;  %next (gtad) time to check if system
   going down%                                                         15F14
   (lavtabad) EXTERNAL ; %getab table address for load average%       15F15
   (jdno) EXTERNAL ; % address of string with list of numbers to be
   delivered; zero if none %                                           15F16
   (jdid) EXTERNAL ; % address of string with list of idents whose
   mail is to be processed-- zero if none %                            15F17
   (jdfi) EXTERNAL ; % address of string with list of files to be
   processed-- zero if none %                                          15F18
   (idfcnt) EXTERNAL ; %counter for opening and closing ident file%
                                                                       15F19
   (idfno) EXTERNAL ; %file number of ident-file or zero%              15F20
   (ckiwheel) EXTERNAL = 1 ; %true if should check for identwheels
   when open identfile%                                                15F21
   (idcrec) EXTERNAL ; %address of current record string%              15F22
   (idcrtype) EXTERNAL ; %current record type (indtyp) grptyp or
   orgtyp%                                                             15F23
   (idcident) EXTERNAL ; %address of current record ident string%
```

```
                                                                    15F24
   (nwrecflag) EXTERNAL = 0 ; %true if user is defining a new record
   for the ident-file%                                              15F25
   (idmodflag) EXTERNAL ; %true if user is allowed to modify the
   current record%                                                  15F26
   (idmodified) EXTERNAL ; %true if user has modified the current
   ident record%                                                    15F27
   (oljmlav) EXTERNAL = 202688 ; %max load average (in floating
   point) for running on line del%                                  15F28
   (oljsbn) EXTERNAL ; %six bit journal system name%                15F29
   %Now timing variables%
      (jtimfg) EXTERNAL ; %True if timing%                       15F30A
      (jt0) EXTERNAL ;                                           15F30B
      (jt1) EXTERNAL ;                                           15F30C
      (jt2) EXTERNAL ;                                           15F30D
      (jt3) EXTERNAL ;                                           15F30E
      (jt4) EXTERNAL ;                                           15F30F
      (jt5) EXTERNAL ;                                           15F30G
      (jt6) EXTERNAL ;                                           15F30H
      (jt7) EXTERNAL ;                                           15F30I
      (jt8) EXTERNAL ;                                           15F30J
      (jt9) EXTERNAL ;                                           15F30K
      (jt10) EXTERNAL ;                                          15F30L
      (jt11) EXTERNAL ;                                          15F30M
      (jt12) EXTERNAL ;                                          15F30N
      (jt13) EXTERNAL ;                                          15F30O
      (jt14) EXTERNAL ;                                          15F30P
      (jt15) EXTERNAL ; %for individual times..see jtime for
      meanings%                                                  15F30Q
   (ojsqsw) EXTERNAL ; %seq work area for net jounal delivery%    15F31
   (jlogjfn) EXTERNAL =0 ; %jfn of log file (<journal>jlog.txt)
   while primary output is being diverted there%                  15F32
   (jckludgeupdtfl) EXTERNAL =0 ; %flag for updtfl to use ":"
   iistead of ";" in origin statement link for oldnls journal
   delivery at office-1%                                          15F33
   (jsavaccess) EXTERNAL ; %for restoring file access after Journal
   has run%                                                       15F34
   % TEXT POINTERS %
      (jwp1) EXTERNAL TEXT POINTER ; %pointer used to locate header
      in jwork%                                                  15F35A
      (tmpbug) EXTERNAL TEXT POINTER ; %pointer to user journal
      template%                                                  15F35B
      (jb1) EXTERNAL TEXT POINTER ;                              15F35C
      (jb2) EXTERNAL TEXT POINTER ; %misc pointers needed by
      Journal%                                                   15F35D
   % EXTERNAL STRINGS %
      (ipassw) EXTERNAL STRING [40]; %for saving password of
      connected directory (ident)%                               15F36A
      (jpassw) EXTERNAL STRING [40]; %for saving password of
      connected directory (journal)%                             15F36B
      (rfcnum) EXTERNAL STRING [5]; %for saving off RFC number if
      one is assigned%                                           15F36C
      (datesr) EXTERNAL STRING [20]; %for date/time of entry%    15F36D
      (jnamstr) EXTERNAL STRING [50]; %for building file names used
      by Journal%                                                15F36E
%Identification system%
```

```
   (identstr) EXTERNAL STRING [200]; %used for building new entries%
                                                                    15G1
   (idwork) EXTERNAL STRING [50]; %work area%                       15G2
   (newid) EXTERNAL STRING [50]; %used in collection of new id's%
                                                                    15G3
   (idntdel) EXTERNAL STRING = "?
   ,()/\+!#$%_=@;*:<>)[] ";                                         15G4
   (cmntdl) EXTERNAL STRING = ")"; %terminators for comments in
   identlists%                                                      15G5
   (idnamdel) EXTERNAL STRING [5];  %contains delimiters for reading
   names %                                                          15G6
   (jidstk) EXTERNAL STACK[20, 2];                                  15G7
   (identwheel) EXTERNAL;  %true if user is a priviledged Ident
   system user%                                                     15G8
   (jidsbot) EXTERNAL;                                              15G9
%Number system%
   (numstid) EXTERNAL; %stid of number file%                        15H1
%Hard copy Journal distribution%
   % hcdiststid, Sometimes used for Global stid into hcdistfile%
   (lptjfn) EXTERNAL ; % Line ptr JFN--used globally by printing
   machinery%                                                       15I2
   (lptused) EXTERNAL ; %Used as a count of number of documents
   which have been printed since the line printer was opened%       15I3
   (docjfn) EXTERNAL ; %JFN of Document portion of document being
   printed%                                                         15I4
   (hdrjfn) EXTERNAL ; %JFN of header portion of document being
   printed%                                                         15I5
   (docstrt) EXTERNAL ; %Character displacement into docjfn file to
   bypass header%                                                   15I6
   (pfilnum) EXTERNAL ; %file number used to tell printing
   machiinery when to change the document file%                     15I7
   (lastfnum) EXTERNAL ; %Used in conjunction winth lastfnum%       15I8
   (prntfg) EXTERNAL ; %True if anything has been priinted this
   running of hard copy distribution%                               15I9
   (mastacprintflag) EXTERNAL ; %print master and access copies on
   this machine%                                                    15I10
   (idfile) EXTERNAL ; %Occaisionally used globally for file number
   of ident file%                                                   15I11
   (oprocn) EXTERNAL STRING [50]; %Contains name of Output
   Processor%                                                       15I12
   (ptstr) EXTERNAL STRING [50]; %Name of printer file--LPT or
   other%                                                           15I13
   (opstr) EXTERNAL STRING [50]; %Contains ident of operator%       15I14
   (docwfn) EXTERNAL STRING = "<journal>DPNTWRK.txt;0
   "; %Name of document print work file%                            15I15
   (hdrwfn) EXTERNAL STRING = "<journal>HPNTWRK.txt;0
   "; %Name of header print work file%                              15I16
%FTP routines%
   (ftphnd) EXTERNAL = 0 ;                                          15J1
   (ftpem) EXTERNAL ;                                               15J2
   (ftperm) EXTERNAL ;                                              15J3
   (ftpoh) EXTERNAL ;                                               15J4
   (ftpoha) EXTERNAL ;                                              15J5
%collector sorter%
   (keypr) EXTERNAL ;                                               15K1
   (vcvulp) EXTERNAL ;                                              15K2
```

```
    (vcvtp) EXTERNAL ;                                              15K3
    (mersiz) EXTERNAL ;                                             15K4
    (merrff) EXTERNAL ;                                             15K5
    (vtop) EXTERNAL ; %vector index%                                15K6
    (infopn) EXTERNAL ; %flag--true if input file open%             15K7
    (cstid) EXTERNAL ; %stid of output file%                        15K8
    (outcnt) EXTERNAL ; %count of statements in current output file%
                                                                    15K9
    (smtmax) EXTERNAL = 10000; %maximum numbr of statements in one
    output file%                                                   15K10
    (colda) EXTERNAL ; %address of display area%                   15K11
    (colsw) EXTERNAL ; %addess of sequence area%                   15K12
    (sortfg) EXTERNAL ; %true if we are to sort%                   15K13
    (namndx) EXTERNAL ; %index into input file name list%          15K14
    (namlst) EXTERNAL [50]; %pointers to text for input file names
    (indexed by naamndx%                                           15K15
    (nambuf) EXTERNAL [100]; %text area for input file names%      15K16
    (ninfil) EXTERNAL ; %number of input files%                    15K17
    (strpkey) EXTERNAL ; %true if delete keys%                     15K18
    (cversion) EXTERNAL ; %Number in sequence of output file%      15K19
    (outnam) EXTERNAL STRING[50]; %contains root name for output
    files%                                                         15K20
%Catalog system%
    (cppflag) EXTERNAL = FALSE; % Catalog Production Processor in
    Operation Flag %                                                15L1
% Flags for branching around NLS specific code that probably would not
be wanted by other 'TOOLS' %
    (donthelp) EXTERNAL = TRUE;    %no help returns allowed %       16A
    (nodisplay) EXTERNAL = TRUE;  % edon't do dpsets or other display
    stuff %                                                         16B
    (nortnrings) EXTERNAL = FALSE;  % don't bother with file return
    rings. %                                                        16C
% DECLARE's %
    (holdvs) EXTERNAL = 0;  %global to keep cmdfinish from doing too
    much too soon to the various viewspecs values%                  17A
    (exp) EXTERNAL = 0;     %experimental system flag%              17B
    (hostni) EXTERNAL = -1; % LH is HOSTN table #; RH is # entries in
    HOSTN %                                                         17C
    (hostsi) EXTERNAL = -1;  % LH is HSTNAM table #; RH is # entries in
    HSTNAM %                                                        17D
    (ttytbl) EXTERNAL ;   % system JOBTTY table number %           17E
    (jobtbl) EXTERNAL ;   % system JOBDIR table number %           17F
    (bndchk) EXTERNAL ;  % TRUE => perform boundary checks at tt (gt) %
                                                                    17G
    (oprtty) EXTERNAL ;    %operator tty (gets journal errors, archive
    ret requests)%                                                  17H
    (istred) EXTERNAL ;  % last read date of initial file for user
    programs %                                                      17I
    %....CALCULATOR.....%
        %accumulators %
            (accum) EXTERNAL [20];                                 17J1A
            (asub) EXTERNAL ;     %subscript of current accumulator% 17J1B
            (vaccum) EXTERNAL [2];  %scratch accum for EVALUATE%   17J1C
            (opfloat) EXTERNAL [2]; %double floating current operand% 17J1D
            (cbadent) EXTERNAL  = 0; %TRUE if user enters calc from the
            calc %                                                 17J1E
```

```
      (nofeedbk) EXTERNAL ;   % true if TNLS user set terse feedback
      %                                                          17J1F
   % misc %
      (cda) EXTERNAL ;       %display area address. may be actual or
      pseudo%                                                    17J2A
      (coldda) EXTERNAL ;  %original display area address - for
      CALCUATOR%                                                 17J2B
      (castid) EXTERNAL ;   %current location in CALC-IDENT file%
                                                                 17J2C
      (proc) EXTERNAL ;    %addr of arithmetic execution routine%
                                                                 17J2D
   % format variables %
      (cadflg) EXTERNAL = 0;                                     17J3A
      (calflg) EXTERNAL = 0;                                     17J3B
      (cacflg) EXTERNAL = 0;                                     17J3C
      (dfoutm) EXTERNAL = 0640141202008; % mask for dfout JSYS %
                                                                 17J3D
(vrsnno) EXTERNAL STRING [5];  %save area for version number%     17K
(ladj) EXTERNAL STRING [40];   %string forlevel adjust%           17L
(sfhead) EXTERNAL STRING [80];                                    17M
(sfstng) EXTERNAL STRING [100];                                   17N
(lkupreg) EXTERNAL STRING [100]; %<JUMP; lookup> string save area%
                                                                  17O
(prbuf) EXTERNAL STRING [300];  %typewriter output buffer%        17P
% EXTERNAL TEXT POINTERS %
   (p3) EXTERNAL TEXT POINTER ;                                   17Q1
   (p4) EXTERNAL TEXT POINTER ;                                   17Q2
   (p5) EXTERNAL TEXT POINTER ;                                   17Q3
   (p6) EXTERNAL TEXT POINTER ;                                   17Q4
   (p7) EXTERNAL TEXT POINTER ;                                   17Q5
   (p8) EXTERNAL TEXT POINTER ;                                   17Q6
   (p9) EXTERNAL TEXT POINTER ;                                   17Q7
   (p10) EXTERNAL TEXT POINTER ;                                  17Q8
%...file status table...%
   (filst) EXTERNAL [100] ;  %1 entry per file, 25 (filmax) files,
   4 (filstl) words per entry%                                    17R1
   (fopncnt) EXTERNAL [26] ;  % 25 (filmax) files,  + 1 so we may
   index by fileno.  The number of times a file status table entry
   has been used.  For use by, for example, the compiler
   encapsulator. %                                                17R2
   (filcnt) EXTERNAL = 0;                                         17R3
%...substitute work area...%
   (subrec) EXTERNAL [subnm1];                                    17S1
   (subhed) EXTERNAL [4]; %=lshed -- see (UTILTY, shed)%          17S2
%...file access stuff...%
   (access) EXTERNAL = 1; %normal access%                         17T1
   (accmask) EXTERNAL = (1, 2, 4, 10B, 20B, 40B, 100B, 200B);     17T2
   (normlaccess) EXTERNAL = 0;                                    17T3
   (jrnlaccess) EXTERNAL = 1;                                     17T4
   (debugaccess) EXTERNAL = 1;                                    17T5
%...file header...%
   % DONT CHANGE THE ITEMS IN THE HEADER %
   (filhed) EXTERNAL [5] ;                                        17U2
      % these extra words may be taken for additions to header%
   (fcredt) EXTERNAL ;   % file creation date %                   17U3
   (nlsvwd) EXTERNAL = 1 ;   % nls version word (keyword) %       17U4
```

```
   (sidcnt) EXTERNAL ;   %count for generating SID's%                    17U5
   (finit) EXTERNAL ;    % initials at last write %                      17U6
   (funo) EXTERNAL ;     % user number (file owner) %                    17U7
      %if <0, FH is pointer to string in fileheader%
   (lwtim) EXTERNAL ;    % last write time %                              17U8
   (namdl1) EXTERNAL ;   % left name delimiter default character %
                                                                        17U9
   (namdl2) EXTERNAL ;   % right name delimiter default character %
                                                                       17U10
   (rngl) EXTERNAL ;     % upper bound on data ring file blocks %
                                                                       17U11
   (dtbl) EXTERNAL ;     % upper bound on data file blocks %            17U12
   (rfbs) EXTERNAL [6] ;    % start of random file block status
   tables %                                                            17U13
   (rngst) EXTERNAL [95] ;  % ring block status table %                 17U14
   (dtbst) EXTERNAL [370] ; % data block status table %                 17U15
   (mkrtxn) EXTERNAL = 20 ;   % marker table maximum length %           17U16
   (mkrtbl) EXTERNAL ;     % number of markers in marker table %
                                                                       17U17
      %each marker takes MKRL words%
   (mkrtb) EXTERNAL [20] ;  % marker table %                            17U18
   (filhde) EXTERNAL ;   %end of the file header%                       17U19
%...output sequential/quickprint...%
   (sfpmr1) EXTERNAL ;  %r1 for pmaps to output file%                    17V1
   (sfpmr2) EXTERNAL ;  %r2 for pmaps to output file%                    17V2
   (sfpmr3) EXTERNAL ;  %r3 for pmaps to output file%                    17V3
   (sfbyte) EXTERNAL ;  %number of bytes in file%                        17V4
   (sfbufl) EXTERNAL ;  %length of output buffer ( in bytes)%            17V5
   (sfucf)  EXTERNAL ;  %if true, convert to upper case%                 17V6
   (sfpgno) EXTERNAL ;  %if true, paginate; contains current page
   number%                                                             17V7
   (sflnel) EXTERNAL ;  %number columns (characters) per line%          17V8
   (sfndlv) EXTERNAL ;  %number characters per level to indent%         17V9
   (sfmxnd) EXTERNAL ;  %maximum number characters to indent%           17V10
   (sfndbf) EXTERNAL ;  %bytptr to end of op buffer%                    17V11
   (sfbptr) EXTERNAL ;  %byte pointer to current poosition in
   buffer%                                                             17V12
   (sflnpg) EXTERNAL ;  %number of line, this page%                     17V13
   (oqnhfg) EXTERNAL ;  %put only page # on output quickprint if
   true%                                                               17V14
   (oqapfg) EXTERNAL ;  %try open append for @ Q if true%               17V15
%...insert sequential work area...%
   (levind) EXTERNAL = (0,0,0,0,0,0,0,0,0,0,0,0);                       17W1
%...High segment page map table -- see (seqfil, processor)%
   (freemap) EXTERNAL [freesz];                                         17X1
   (pmfjfn) EXTERNAL = 0; %jfn of "pmf" file for private pages of
   high seg when mapped out%                                           17X2
%...other work areas...%
   (mswsit) EXTERNAL ; %cell to accumulate time for wsi%                17Y1
   (mswsic) EXTERNAL ; %cell to accumulate count for wsi%               17Y2
   %wsd-meas cells%
      (ldrfct) EXTERNAL ; %Number of calls on lodrfb%                   17Y3A
      (ldrnct) EXTERNAL ; %number of calls on lodrng%                   17Y3B
      (ldsdct) EXTERNAL ; %Number of calls on lodsdb%                   17Y3C
      (fchsct) EXTERNAL ; %numbr of calls on fchsdb%                    17Y3D
      (strttm) EXTERNAL ; %JOBTM at start of NLS (intmeas)%             17Y3E
```

```
   (msaddr) EXTERNAL [15] ;  %for measurement -- 3 * number of
   entries%                                                    17Y4
   (mslst) EXTERNAL [39] ;  %for measurement -- msentl * number of
   entries%                                                    17Y5
   (mslste) EXTERNAL ;  %end of measurement list%              17Y6
   (msr1sv) EXTERNAL ;  %save area for r1 during measurement%  17Y7
   (msr2sv) EXTERNAL ;  %save area for r2 during measurement%  17Y8
   (msr3sv) EXTERNAL ;  %save area for r3 during measurement%  17Y9
   (measnt) EXTERNAL ;  %current item being measured%         17Y10
   (iswork) EXTERNAL [4] ;   %insert sequential work area%    17Y11
   (oprwrk) EXTERNAL [20] ;  %output processor work area%     17Y12
   (comexflag) EXTERNAL CONSTANT = 0; %KLUDGE until output COM
   syntax fixed-- 0 is comp80, 1 is singer%
   (gtjfnt) EXTERNAL [10] ;  %for long getjfn calls%          17Y14
   (typend) EXTERNAL ;                                        17Y15
%...global display/print parameters...%
   %...display global support...%
      (lastch) EXTERNAL ;                                     17Z1A
      % (mkrbuf) EXTERNAL ;        pointer to marker display buffer%
      (mkrend) EXTERNAL ;        %ditto%                      17Z1C
      (mkrptr) EXTERNAL ;        %pointer into marker display
      buffer%                                                 17Z1D
      (mkrcnt) EXTERNAL ;        %char count of last marker found%
                                                              17Z1E
      (mkrflg) EXTERNAL ;        %true if found a marker in a
      statement%                                              17Z1F
      % (cdctrn) EXTERNAL ; %
      (hinc) EXTERNAL=1 ;                                     17Z1H
      (vinc) EXTERNAL=1 ;                                     17Z1I
      (cdchrl) EXTERNAL ;                                     17Z1J
      (cdchct) EXTERNAL ;                                     17Z1K
   %...fast create display support...%
      (aulsrt) EXTERNAL ;  %address, current auxiliary display
      table%                                                  17Z2A
      (aulsize) EXTERNAL ;  %size, in lsrtl entries, of auxiliary
      table%                                                  17Z2B
      (rttop) EXTERNAL ;   %starting address, current table being
      formatted%                                              17Z2C
      (rtsize) EXTERNAL ;    %size, current table being formatted%
                                                              17Z2D
   %...display support...%
      % (aplint) EXTERNAL ; %
      (udpnwrap) EXTERNAL ;                                   17Z3B
      (gapcol) EXTERNAL ;                                     17Z3C
      (gapcnt) EXTERNAL ;                                     17Z3D
      (gapbp) EXTERNAL ;                                      17Z3E
      (gapcc) EXTERNAL ;                                      17Z3F
      (cc) EXTERNAL ;                                         17Z3G
      (rt) EXTERNAL ;                                         17Z3H
      (rte) EXTERNAL ;                                        17Z3I
      (art) EXTERNAL ;                                        17Z3J
      (arte) EXTERNAL ;                                       17Z3K
      (oldlsrt) EXTERNAL ;  %address of old Line Segment Reference
      Table%                                                  17Z3L
      (commands) EXTERNAL [65] ; %first block of display
      commands--others to be dynamically allocated%           17Z3M
```

```
      (dpcmblk) EXTERNAL ; %address of commands block currently
      being filled%                                              17Z3N
      (dgstl) EXTERNAL ;    %current character count for statement
      formatting%                                                17Z3O
      (dgstml) EXTERNAL ;   %max character count%                17Z3P
      (dgstid) EXTERNAL ;    %stid of statement being formatted%
                                                                 17Z3Q
      (dgbufe) EXTERNAL ;                                        17Z3R
%...typewriter print variables...%
   (prmkrf) EXTERNAL = 0 ;    %typewriter print marker flag%     17A@1
   (cdpagf) EXTERNAL = 1 ;    %page flag%                        17A@2
   (pageno) EXTERNAL ;    %page number for print group%          17A@3
   (slshfg) EXTERNAL ;    %slash command flag%                   17A@4
%...general global variables...%
   (bfilno) EXTERNAL ; %Contains file number on bad file SIGNAL%
                                                                 17AA1
   (ercall) EXTERNAL ; %Contains the address of last call to error%
                                                                 17AA2
   (ermark) EXTERNAL ; %Contains value of mark at last call to err%
                                                                 17AA3
   (fastname) EXTERNAL = 1 ; %Flag for which jump name algorithm to
   use on normal jumps%                                          17AA4
   (carpos) EXTERNAL ;        %position of carraige (0=left margin)%
                                                                 17AA5
   (slashflg) EXTERNAL = 0 ;  %show selections flag%             17AA6
   (srchtype) EXTERNAL ;     %Type of item to search for (name/word)
   contents%                                                     17AA7
   (drastic) EXTERNAL ; %set by wrpi to indicate a bad situation;
   checked by wpiabt%                                            17AA8
%...Library  stuff...%
   (oldflg) EXTERNAL = 0; %sticky state of libflg%               17AB1
   (libflg) EXTERNAL = 0; %non-zero means routine running using NLS
   as library (so don't expect input)...number indicates which% 17AB2
   (liblod) EXTERNAL = 0; % TRUE if load average cut-off set by hand
   %                                                             17AB3
   (jnlprog) EXTERNAL = 0; % TRUE if journal userprog in; FALSE if
   not. %                                                        17AB4
   (autostrt) EXTERNAL = 0; %True if the library started
   automatically at system startup%                              17AB5
   (slnkrv) EXTERNAL = 1;                                        17AB6
   (utiltv) EXTERNAL = 4;                                        17AB7
%...processor variables...%
   (prseqwk) EXTERNAL ;       %address of  processor's seqgen work
   area%                                                         17AC1
   (opbp) EXTERNAL ; % byte pointer to start of current string %
                                                                 17AC2
   (opcbp) EXTERNAL ;         % byte pointer to current char position
   %                                                             17AC3
   (opccpos) EXTERNAL ;       % current char postion %          17AC4
   (opcmax) EXTERNAL ;        % max chars in current string %   17AC5
   (opiev) EXTERNAL ;         % current statement level %       17AC6
   (opnewst) EXTERNAL ;       % flag) EXTERNAL ; used to make level
   find efficient %                                              17AC7
%...substitute variables...%
   (subcnt) EXTERNAL ;     %number of substitutions made%        17AD1
%...Print Journal...%
```

```
      (pjrbab) EXTERNAL ; % save for contents of rubabt %                17AE1
      (pjusqc) EXTERNAL ; % save for pntr to user seq gen %              17AE2
      (pjsavf) EXTERNAL ; % save for user seq gen viewspec %             17AE3
      (pjsw) EXTERNAL ;                                                  17AE4
      (pjlsw) EXTERNAL ; % pntrs to seq workareas %                      17AE5
      (pjrubout) EXTERNAL ; % set TRUE on RUBOUT %                       17AE6
      (pjstid) EXTERNAL TEXT POINTER;                                    17AE7
   (patch) EXTERNAL [403];                                               17AF
   (patche) EXTERNAL ; % patch space %                                   17AG
% page aligned output buffers %
   PAGE 140B; %get aligned on page boundary (assume load starts 140B)%
   (msbuff) EXTERNAL [5113]; %measurement file buffer%                   18B
   (msbufe) EXTERNAL ;  %end of measurement buffer%                      18C
   (sfbuff) EXTERNAL [5113]; %Sequential file buffer%                    18D
   (sfbufe) EXTERNAL ;  %end of Sequential file buffer%                  18E
%declarations moved from pdata - are now backend functions%
   % CURRENT MARKER %
      (curmkr) EXTERNAL TEXT POINTER ;  % current marker %               19A1
      (ecurmkr) EXTERNAL ;       % current marker for insert statement
      mode %                                                             19A2
      (cspupdate) EXTERNAL ;     %flag to set up csp from curmkr in dnls
      -- contains da address or zero%                                    19A3
      (usesrr) EXTERNAL ;        %address of statement return ring to
      use for jump file return or zero%                                  19A4
      (fakesrr) EXTERNAL ;       %fake usesrr to get around limits on
      number of returns%                                                 19A5
      (cspcacode) EXTERNAL ; %content analyzer code address -- see
      CMDFINISH%                                                         19A6
      (cspusqcod) EXTERNAL ; %user sequence generator program address
      -- see CMDFINISH%                                                  19A7
         % the above two variables are provided so that dumb users can
         set content analyzers and seq generators without messing with
         da's.  CMDFINISH simply copies these into the da (if non
         zero). %
      (cspvs) EXTERNAL [2];      %viewspecs to be used with curmkr if
      changecsp is set%                                                  19A8
         % the above two variables are provided so that dumb users can
         set viewspecs without messing with da's.  CMDFINISH copies
         these into the da. %
      (frrcnt) EXTERNAL ;  %used for jump file return%                   19A9
      (srrcnt) EXTERNAL ;  %used for jump return%                        19A10
% start up ac's: registers 3-15 are saved away from pre-save runtime
initialization, then restored at startup time.  registers 0-2 are set
up by the FE at startup and must be saved away.  at the entry vector to
the BE (cf. nswnls) the image and real registers are made to be alike
with 0-2 as per the FE, 3-15 as per runtime presave %
   (stacs) EXTERNAL;  %ac 0%                                             20A
   (stac1) EXTERNAL;  %ac 1%                                             20B
   (stac2) EXTERNAL;  %ac 2%                                             20C
   (stac3) EXTERNAL;  %ac 3%                                             20D
   (stac4) EXTERNAL;  %ac 4%                                             20E
   (stac5) EXTERNAL;  %ac 5%                                             20F
   (stac6) EXTERNAL;  %ac 6%                                             20G
   (stac7) EXTERNAL;  %ac 7%                                             20H
   (stac8) EXTERNAL;  %ac 8%                                             20I
   (stac9) EXTERNAL;  %ac 9%                                             20J
```

```
   (stac10) EXTERNAL;  %ac 10%                                          20K
   (stac11) EXTERNAL;  %ac 11%                                          20L
   (stac12) EXTERNAL;  %ac 12%                                          20M
   (stac13) EXTERNAL;  %ac 13%                                          20N
   (stac14) EXTERNAL;  %ac 14%                                          20O
   (stac15) EXTERNAL;  %ac 15%                                          20P
%buffers for saving ac's in psi routine for state change interrupt%
   (psiac1) EXTERNAL;                                                   21A
   (psiacs) EXTERNAL[15];                                               21B
   (psiace) EXTERNAL;                                                   21C
% These are from nls,undata %
   (state) EXTERNAL [4];  %previous state%                              22A
   (num) EXTERNAL STRING [40] ;                                         22B
   (stn) EXTERNAL STRING [40] ;                                         22C
   (stn2) EXTERNAL STRING [40] ;                                        22D
   (stno) EXTERNAL STRING [40] ;                                        22E
   (lit) EXTERNAL STRING [2000] ;                                       22F
   (tlit) EXTERNAL STRING [2000] ; % literal string for use in cprint.
   %                                                                    22G
      %eventually replace with allocated string %
   (errwrk) EXTERNAL STRING [100]; % work string for errors. %          22H
   (ukcstr) EXTERNAL STRING _ "U";  %used to display unknown character%
                                                                        22I
   %...measurement variables...%
      (nlcrrt) EXTERNAL = 0;    %realtime at beginning of last command%
                                                                        22J1
      (nlcret) EXTERNAL = 0;    %cpu time at beginning of last command%
                                                                        22J2
      (nlcrlst) EXTERNAL = 0;   %time (real) last NLSCR JSYS issued%
                                                                        22J3
      (nlstyp) EXTERNAL = -1;   %type of NLS system being run%          22J4
   %..... HELP Declarations..... %
      (helploc) EXTERNAL [4]; % fake subsystem dispatch record %        22K1
      (calcaux) EXTERNAL ;                                              22K2
      (inhelp) EXTERNAL =0; % TRUE if in help command, FALSE otherwise.
      %                                                                 22K3
      (qagain) EXTERNAL =0; % TRUE if in cntl-q typed while in help
      command, FALSE otherwise. %                                       22K4
      (hipfileno) EXTERNAL ;                                            22K5
      (qda) EXTERNAL ;                                                  22K6
      (qsw) EXTERNAL ;                                                  22K7
      (qstorblk) EXTERNAL ;                                             22K8
      (qnewstmt) EXTERNAL ;                                             22K9
      (qdavspc) EXTERNAL ;                                             22K10
      (qdavs2) EXTERNAL ;                                              22K11
      (overscreen) EXTERNAL ; % TRUE if screen overflows. %            22K12
      (hseqer) EXTERNAL = 0 ; % TRUE if trouble in help sequence
      generator %                                                       22K13
      (multiflg) EXTERNAL = 0 ;                                         22K14
      (sflg) EXTERNAL =1; %search type%                                 22K15
      (menchr) EXTERNAL = "];                                           22K16
      (comchr) EXTERNAL = "%; % Format characters %                     22K17
      (mentyp) EXTERNAL =1;                                             22K18
      (multyp) EXTERNAL =2;                                             22K19
      (untyp) EXTERNAL =0; % Format type of statement %                 22K20
      (moremen) EXTERNAL ; % TRUE if more menu to be shown. %           22K21
```

```
      (conrng) EXTERNAL ; % The address of the context ring block. %
                                                                    22K22
      (curstk) EXTERNAL ;                                           22K23
      (curindex) EXTERNAL ; % The address of the current context ring
      element %                                                     22K24
      (confre) EXTERNAL ; % The next free context index. %          22K25
      (newstk) EXTERNAL = 1 ; % flag for qmenu.  calls pushent for new
      context stack only if TRUE %                                  22K26
      (topcon) EXTERNAL [2];                                        22K27
      (entcon) EXTERNAL [2]; % Contexts for entry and top. %        22K28
      (qspecs) EXTERNAL ;     % current query viewspecs in force %  22K29
      (qpspecs) EXTERNAL ;     % previous query viewspecs in force %
                                                                    22K30
      (qmenumax) EXTERNAL = 15;                                     22K31
      (qmenucnt) EXTERNAL = 0;   % current value for NEXT menu item %
                                                                    22K32
      (qcolwidth) EXTERNAL = 24 ;                                   22K33
      (hdebug) EXTERNAL = 1 ; %true for debugging...loads xhelp%    22K34
% These are from nls,fdata %
   (initsr) EXTERNAL STRING [15];  %user's ident%                    23A
   (logstr) EXTERNAL STRING [40];  %user's login directory%          23B
% Moved here from fdata %
(cshift) EXTERNAL ;  %character which causes lower to upper case shift
in TNLS for the following character -- see <inpfbk,tinptc>%          25
(oshift) EXTERNAL ;  %typewriter output shift mode%                 26
(echofg) EXTERNAL ;  %typewriter echo flag for output%              27
(savedda) EXTERNAL ;  %address of DNLS display area "Taken" to TNLS%  28
(savetda) EXTERNAL ;  %address of TNLS display area "Taken" to DNLS%  29
(xacs) EXTERNAL  [16];  % ACs at entry to NLS %                     30
%private stack for illegal intruction pseudo-interrupt%
   (psiskszs) EXTERNAL CONSTANT = 100;                              31A
% *** GLOBAL DATA FOR PAGE 0 *** %
% Special stuff declared in page 0, loaded at 140B %
   %...KEEP THESE THINGS TOGETHER -- DONT ADD NEW DATA HERE...%
   %...important stuff...%
   %...user related data...%
      (cuno) EXTERNAL ;    % current user number (login dir) %      33C1
      (conndir) EXTERNAL ;  %current connected dir%                 33C2
   %...set command and aptstr routine...%
      (xsmode) EXTERNAL =40B;    %  case set    %                  33D1
   % save stack for viewspec input backup%
      (savevspec) EXTERNAL STACK [20, 2];                           33E1
% stack for illegal instruction pseudo-interrupt %
   (psistk) EXTERNAL [100];                                         34A
%Bit tables for LSID and DAID assignment%
   (dabt) EXTERNAL [3];   %daid bit table%                          35A
   (lsbitables) EXTERNAL [56]; %7 wrds, 8 das%  %lsid bit tables for
   file text display areas, pointed to by dalsidb field in da records%
                                                                    35B
   (lsbtsize) EXTERNAL = 7 ; %size (number of words) of a lsbitable% 35C
   (dabtsize) EXTERNAL = 3 ; %size (number of words) of the dabt%   35D
% useroptions %
   (uojfn) EXTERNAL ;    % jfn for the user profile file %          36A
(savnldevice) EXTERNAL ;   %used to save nldevice during screen
sharing%                                                            37
(nlssbn) EXTERNAL ;    %contains six-bit name for version of nls being
```

```
used%                                                                    38
(jnlsbn) EXTERNAL =0;  %contains six-bit name for journal%               39
%...display area stuff.....%
   (dpyarea) EXTERNAL [160]; %dal*damax%                                 40A
   (dpyend) EXTERNAL ;                                                   40B
   (nlsdas) EXTERNAL [40];  %da's for default text window%               40C
   (dacnt) EXTERNAL = 0 ;    %0 => no da's active%                       40D
   (wndwarea) EXTERNAL [60];  %wal*wamax%                                40E
   (wndwend) EXTERNAL ;                                                  40F
   (wacnt) EXTERNAL = 0 ;    %0 => no wa's active%                       40G
   (prwndw) EXTERNAL ;  %address of primary window window area%         40H
   (vswndw) EXTERNAL = 0 ;  %address of viewspec window window area%    40I
   (dtwndw) EXTERNAL ;  %address of default text window window area%    40J
   (rtfree) EXTERNAL = 0 ;   %free list pointer--physical entry count%
                                                                         40K
   (vspsav) EXTERNAL = (0, 0);  %viewspec displayed in viewspec window%
                                                                         40L
   (dsparg) EXTERNAL _ 0;  %addr. of allocated list for dpy arguments
   to FE%                                                                40M
   (dspcmd) EXTERNAL _ 0;  %addr. of allocated list of commands to FE%
                                                                         40N
   (dSplslist) EXTERNAL _ 0;  %addr. of line segment list for
   'buildcom'%                                                           40O
   (dspccf) EXTERNAL _ 1;  %switch for clearing command feedback
   window%                                                               40P
      % TRUE if cf window should be cleared by 'dafrmt' %
% List for arguments to FE HELP procedure %
    (helplist) EXTERNAL _ 0;  %addr. of allocated list for args%         41A
%...system handles for display areas...%
   (ttyda) EXTERNAL ;                                                    42A
   (dttyda) EXTERNAL ;  %dafault tty display area%                       42B
   (tda) EXTERNAL ;     %current display area address%                   42C
(spacestr) EXTERNAL STRING   =  "
                                ";                                       43
(nlsnam) EXTERNAL STRING [10]; %contains name of nls system being used%
                                                                         44
(cdstr1) EXTERNAL STRING [2000];  %used by create display for
intra-statement edits%                                                   45
(cdstr2) EXTERNAL STRING [2000];  %used by create display for
intra-statement edits%                                                   46
(conreg) EXTERNAL STRING [100];  %jump string save area%                 47
(wrdreg) EXTERNAL STRING [40];  %jump string save area%                  48
(vspstr) EXTERNAL STRING [30];  %viewspec string%                        49
(namreg) EXTERNAL STRING [40];  %name default register%                  50
(fvstk) EXTERNAL STACK [65 %svmxlev+1%];                                 51
%...frozen statement list...%
   (fzlsts) EXTERNAL [80];                                               52A
   (fzlste) EXTERNAL ; %end of buffer%                                   52B
   (fzfree) EXTERNAL ; %free list pointer for fzlsts%                    52C
   (frzmax) EXTERNAL = 20; %length of freeze list%                       52D
%...input stuff...%
   (bugreg) EXTERNAL [2];      %bug mark area%                           53A
%...global display/print parameters...%
   %...parameters to dpset for selective recreate display%
      (cdstd1) EXTERNAL ;  %stid of stmt to be reformatted; if cdtype =
      dsprfmt  or dsprfst; else passed as file indicators so da's may
```

```
    be selectively updated%                                              54A1
    (cdstd2) EXTERNAL ;  %second stid or file indicator%                 54A2
    (cdstop) EXTERNAL ;  %after encountering this stid, daupdate
    knows no      further structural or editing changes took place on
    the old lsrt entries.%                                               54A3
    (cdtype) EXTERNAL ;  %option to inform daupdate of general nature
    of operation performaed by command issued.  Options are defined
    in data.%                                                            54A4
  %...general global variables...%
    (continue) EXTERNAL = 0; %  continue flag--set by terminate  %
                                                                         54B1
    (nlmode) EXTERNAL ;     % typewriter or NOT typewriter%             54B2
    (niparse) EXTERNAL = 1; %TRUE for non-query entry; FALSE for
    query%                                                               54B3
    (exquery) EXTERNAL ;                % no ident or initial file
    needed %                                                             54B4
    (cmarkflg) EXTERNAL =  0;  %show control marker flag%                54B5
  %...initial values for display areas...%
    (rmarg) EXTERNAL ; %right margin%                                    54C1
    (lmarg) EXTERNAL ; %left margin%                                     54C2
    (tmarg) EXTERNAL ; %top margin%                                      54C3
    (bmarg) EXTERNAL ; %bottom margin%                                   54C4
    %text area%
       (tabase) EXTERNAL = $taleft;                                      54C5A
       (taleft) EXTERNAL = 0; %lmarg%                                    54C5B
       (taright) EXTERNAL = 0; %rmarg%                                   54C5C
       (tatop) EXTERNAL = 0; %tmarg + 9*tavinc%                          54C5D
       (tabottom) EXTERNAL = 0; %bmarg%                                  54C5E
       (tacsize) EXTERNAL = 1; %character size%                          54C5F
       %( tafont) EXTERNAL = 0; font is normal%
       (tahinc) EXTERNAL = 0; %14*256%                                   54C5H
       (tavinc) EXTERNAL = 0; %30*256%                                   54C5I
       (tax) EXTERNAL = 0;                                               54C5J
       (tay) EXTERNAL = 0;                                               54C5K
       (tamind) EXTERNAL = 15;                                           54C5L
% Symbol table stuff %
  % variables used in PSEDT1 and BSYMMNP %
    (symloc) EXTERNAL = 116B;                                            55A1
    (nisloader) EXTERNAL = 117B;                                         55A2
    (ddtsptr) EXTERNAL = 770001B; % ptr to ddt's alt i -1 %              55A3
% current windowid variables%
  (cwndow) EXTERNAL = 1; % current window id- last window referenced
  by user%                                                               56A
  (wsflag) EXTERNAL ;  %window selection flag for argument conversion%
                                                                         56B
  (wsvalu) EXTERNAL ;  %window selectin value for argument conversion%
                                                                         56C
% data for the encapsulated compiler %
  % global support variables %
    (cpfkh) EXTERNAL;           % compiler fork handle %                 57A1
    (cpojfn) EXTERNAL;          % jfn for output rel file %              57A2
    (cprcdjfn) EXTERNAL;        % jfn for sequential recording %         57A3
    (cpmsgdest) EXTERNAL;       % diagnostics dump destination %         57A4
    (cpsw) EXTERNAL REF;        % compiler source sequence wa %          57A5
    (cpauxsw) EXTERNAL REF;     % compiler auxiliary sequence work area
    %                                                                    57A6
```

```
      (cpistflno) EXTERNAL;      % file number of last formatted
      statement %                                                    57A7
      (cppgcount) EXTERNAL;      % source file dummy page count %    57A8
      (cpercount) EXTERNAL;      % number of errors detected by compiler
      %                                                              57A9
      (cpwdcount) EXTERNAL;      % number of words created by compiler %
                                                                     57A10
      (cpstate) EXTERNAL;        % describes the state of compile % 57A11
   % globals strings %
      (leftover) EXTERNAL STRING [20];  % leftover text for page
      formatting %                                                   57B1
      (cpostr) EXTERNAL STRING [100];   % output string from compiler %
                                                                     57B2
   % text pointers %
      (fmtstid) EXTERNAL TEXT POINTER;  % t.p. into the next statement
      to format %
% regarding "subsystems"  (from BCONST) %
   (idnfname) EXTERNAL STRING _ "<IDENTS>IDENTS.NLS"; %changed by
   (bintnls,saveit) for some hosts%                                  58A
   (subdfdir)   EXTERNAL STRING [40];  % default directory for programs
   (subsystems too) %                                                58B
FINISH of bdata
```

(af)	<nine, beignore, 0142>	EXT 3E3
(allocilist)	<nine, beignore, 050>	EXT 3L1
(allsubs)	<nine, beignore, 0140>	EXT 3C2
(altinp)	<nine, beignore, 0128>	EXT 3I7
(an)	<nine, beignore, 0143>	EXT 3E4
(arowon)	<nine, beignore, 073>	EXT 3E9
(auxmd2)	<nine, beignore, 0119>	EXT 3G5
(auxmod)	<nine, beignore, 0118>	EXT 3G4
(auxsav)	<nine, beignore, 0115>	EXT 3G1
(auxtp1)	<nine, beignore, 0116>	EXT 3G2
(auxtp2)	<nine, beignore, 0117>	EXT 3G3
(auxwrk)	<nine, beignore, 0120>	EXT 3G6
(basestateflag)	<nine, beignore, 0122>	EXT 3G9
(bmcnt)	<nine, beignore, 068>	EXT 3E6
(bmoff)	<nine, beignore, 0152>	EXT 3E22H
(cfldsp)	<nine, beignore, 0144>	EXT 3E5
(cntrlgetchar)	<nine, beignore, 0162>	EXT 3I4
(curchr)	<nine, beignore, 0131>	EXT 3I10
(current)	<nine, beignore, 0133>	EXT 3E1
(detttysim)	<nine, beignore, 096>	EXT 3E22S
(delsubsys)	<nine, beignore, 0138>	EXT 3C1
(dev33)	<nine, beignore, 0164>	EXT 3E22AE
(dev35)	<nine, beignore, 0165>	EXT 3E22AF
(dev37)	<nine, beignore, 0166>	EXT 3E22AG
(devlproc)	<nine, beignore, 0101>	EXT 3E22Y
(devtiex)	<nine, beignore, 0109>	EXT 3E22AH
(dn)	<nine, beignore, 0148>	EXT 3E22D
(dspjfn)	<nine, beignore, 0106>	EXT 3E22AB
(dsubsys)	<nine, beignore, 0147>	EXT 3E22C
(echda)	<nine, beignore, 098>	EXT 3E22U
(gcords)	<nine, beignore, 0135>	EXT 3E22A
(getchar)	<nine, beignore, 0159>	EXT 3I1
(imlac0)	<nine, beignore, 099>	EXT 3E22W
(imlac1)	<nine, beignore, 0100>	EXT 3E22X
(initbt)	<nine, beignore, 0155>	EXT 3E23A
(initch)	<nine, beignore, 0156>	EXT 3E23B
(initdis)	<nine, beignore, 0151>	EXT 3E22G
(inittimer)	<nine, beignore, 094>	EXT 3E22Q
(inpjfn)	<nine, beignore, 0129>	EXT 3I8
(inpwatchjfn)	<nine, beignore, 0127>	EXT 3I6
(ldspjfn)	<nine, beignore, 088>	EXT 3E22K
(linkcnsl)	<nine, beignore, 095>	EXT 3E22R
(litapflag)	<nine, beignore, 0107>	EXT 3E22AC
(litdahandle)	<nine, beignore, 0103>	EXT 3E22A@
(litline)	<nine, beignore, 087>	EXT 3E22J
(litreset)	<nine, beignore, 097>	EXT 3E22T
(lpaltgetchar)	<nine, beignore, 0161>	EXT 3I3
(lpbaud)	<nine, beignore, 076>	EXT 3E12
(lpdlpad)	<nine, beignore, 077>	EXT 3E13
(lpgetchar)	<nine, beignore, 0160>	EXT 3I2
(lpilpad)	<nine, beignore, 078>	EXT 3E14
(lplitline)	<nine, beignore, 089>	EXT 3E22L
(lplitreset)	<nine, beignore, 090>	EXT 3E22M
(lppch)	<nine, beignore, 081>	EXT 3F17
(lppcol)	<nine, beignore, 082>	EXT 3E18
(lppjfn)	<nine, beignore, 084>	EXT 3E20

(ipplin)	<nine, beignore, 083>	EXT 3E19
(iptab)	<nine, beignore, 079>	EXT 3E15
(iptype)	<nine, beignore, 093>	EXT 3E22P
(ipxmax)	<nine, beignore, 091>	EXT 3E22N
(ipymax)	<nine, beignore, 092>	EXT 3E22O
(itvsda)	<nine, beignore, 069>	EXT 3E7
(mdemand)	<nine, beignore, 0121>	EXT 3G7
(msgreset)	<nine, beignore, 074>	EXT 3E10
(namereset)	<nine, beignore, 075>	EXT 3E11
(nettty)	<nine, beignore, 0167>	EXT 3E22AI
(nldevice)	<nine, beignore, 0170>	EXT 3E22V
(nlsdae)	<nine, beignore, 0104>	EXT 3E22AA
(nWheelt)	<nine, beignore, 0171>	EXT 3A1
(offline)	<nine, beignore, 0102>	EXT 3E22Z
(padstr)	<nine, beignore, 085>	EXT 3E21
(pbug)	<nine, beignore, 0149>	EXT 3E22E
(rawchr)	<nine, beignore, 0130>	EXT 3I9
(recrunfil)	<nine, beignore, 0126>	EXT 3I5
(rstiit)	<nine, beignore, 0153>	EXT 3E22I
(savctl)	<nine, beignore, 071>	EXT 3E8
(setdev)	<nine, beignore, 0157>	EXT 3E23C
(ssysname)	<nine, beignore, 0146>	EXT 3E22B
(strdev)	<nine, beignore, 0110>	EXT 3E22AJ
(tracking)	<nine, beignore, 0134>	EXT 3E2
(tracksend)	<nine, beignore, 080>	EXT 3E16
(ttysim)	<nine, beignore, 0108>	EXT 3E22AD
(typech)	<nine, beignore, 0163>	EXT 3G8
(xsimdev)	<nine, beignore, 0150>	EXT 3E22F

```
< NINE, BEIGNORE.NLS;3, >, 16-Apr-78 15:16 KIRK ;;;;
FILE beignore  % <ARCSUBSYS>XL10 <RELNINE>beignore %  %
(arcsubsys,xl10,)  (RELNINE,beignore.rel,) %

%This file contains as globals all the undefined symbols that we don't
have to deal with yet.%

   %Journal stuff%
      (nwheelf) EXTERNAL;                                          3A1

   %Programs stuff (subsystems)%
      (delsubsys) EXTERNAL;                                        3C1
      (allsubs) EXTERNAL;                                          3C2

   %Display stuff% %used only by old DSPGEN%
      (current) EXTERNAL;                                          3E1
      (tracking) EXTERNAL;                                         3E2
      (af) EXTERNAL;                                               3E3
      (an) EXTERNAL;                                               3E4
      (cfldsp) EXTERNAL;                                           3E5
      (bmcnt) EXTERNAL;                                            3E6
      (ltvsda) EXTERNAL;                                           3E7
      (savcfl) EXTERNAL;                                           3E8
      (arowon) EXTERNAL;                                           3E9
      (msgreset) EXTERNAL;                                        3E10
      (namereset) EXTERNAL;                                       3E11
      (lpbaud) EXTERNAL;                                          3E12
      (lpdlpad) EXTERNAL;                                         3E13
      (lpilpad) EXTERNAL;                                         3E14
      (lptab) EXTERNAL;                                           3E15
      (tracksend) EXTERNAL;                                       3E16
      (lppch) EXTERNAL;                                           3E17
      (lppcol) EXTERNAL;                                          3E18
      (lpplin) EXTERNAL;                                          3E19
      (lppjfn) EXTERNAL;                                          3E20
      (padstr) EXTERNAL;                                          3E21
      %won't be used anyway%
         (gcords) EXTERNAL;                                      3E22A
         (ssysname) EXTERNAL;                                    3E22B
         (dsubsys) EXTERNAL;                                     3E22C
         (dn) EXTERNAL;                                          3E22D
         (pbug) EXTERNAL;                                        3E22E
         (xsimdev) EXTERNAL;                                     3E22F
         (initdis) EXTERNAL;                                     3E22G
         (bmoff) EXTERNAL;                                       3E22H
         (rstlit) EXTERNAL;                                      3E22I
         (litline) EXTERNAL;                                     3E22J
         (ldspjfn) EXTERNAL;                                     3E22K
         (lplitline) EXTERNAL;                                   3E22L
         (lplitreset) EXTERNAL;                                  3E22M
         (lpxmax) EXTERNAL;                                      3E22N
         (lpymax) EXTERNAL;                                      3E22O
         (lptype) EXTERNAL;                                      3E22P
         (inittimer) EXTERNAL;                                   3E22Q
         (linkcnsl) EXTERNAL;                                    3E22R
         (defttysim) EXTERNAL;                                   3E22S
```

```
        (litreset) EXTERNAL;                                        3E22T
        (echda) EXTERNAL;                                           3E22U
        (nldevice) EXTERNAL ; %current terminal device%             3E22V
        (imlac0) EXTERNAL = 7777B;                                  3E22W
        (imlac1) EXTERNAL = 7777B;                                  3E22X
        (devlproc) EXTERNAL = 7777B;                                3E22Y
        (offline) EXTERNAL = 7777B;                                 3E22Z
        (litdahandle) EXTERNAL;                                    3E22A@
        (nlsdae) EXTERNAL;                                          3E22AA
        (dspjfn) EXTERNAL;                                          3E22AB
        (litapflag) EXTERNAL;                                       3E22AC
        (ttysim) EXTERNAL;                                          3E22AD
        (dev33) EXTERNAL = 0;                                       3E22AE
        (dev35) EXTERNAL = 1;                                       3E22AF
        (dev37) EXTERNAL = 2;                                       3E22AG
        (devtiex) EXTERNAL = 3;                                     3E22AH
        (nettty) EXTERNAL = 7;                                      3E22AI
        (strdev) EXTERNAL;                                          3E22AJ
    %device simulation%
        (initbt) EXTERNAL;                                           3E23A
        (initch) EXTERNAL;                                           3E23B
        (setdev) EXTERNAL;                                           3E23C

    %Process commands stuff%
        (auxsav) EXTERNAL;                                            3G1
        (auxtp1) EXTERNAL;                                            3G2
        (auxtp2) EXTERNAL;                                            3G3
        (auxmod) EXTERNAL;                                            3G4
        (auxmd2) EXTERNAL;                                            3G5
        (auxwrk) EXTERNAL;                                            3G6
        (mdemand) EXTERNAL;                                           3G7
        (typech) EXTERNAL;                                            3G8
        (basestateflag) EXTERNAL;                                     3G9

    %Record runfile stuff%
        (getchar) EXTERNAL;                                           3I1
        (ipgetchar) EXTERNAL;                                         3I2
        (ipaltgetchar) EXTERNAL;                                      3I3
        (cntrigetchar) EXTERNAL;                                      3I4
        (recrunfil) EXTERNAL;                                         3I5
        (inpwatchjfn) EXTERNAL = 0;                                   3I6
        (altinp) EXTERNAL;                                            3I7
        (inpjfn) EXTERNAL;                                            3I8
        (rawchr) EXTERNAL;                                            3I9
        (curchr) EXTERNAL;                                           3I10


    %From PSEDIT1/2 for allocating lists -- unused%
        (alloclist) EXTERNAL;                                         3L1

% It also contains comments on things we're ignoring for now but need
to fix later %

    %
    1.  The load command would like to fiddle with the Frontend grammar
    when loading a CML program.  Fix in PSEDIT1,gpget and
```

PSEDIT1,gtctrlbits as noted by commented out code.

%

FINISH of beignore

```
< NINE, BESYMMNP.NLS;7, >, 4-FEB-77 16:32 HGL ;;;;
FILE besymmnp % <ARCSUBSYS>XL10  <RELNINE>besymmnp % %
(arcsubsys,xl10,)  (RELNINE,besymmnp.rel,) %
   ALLOW!
   %Defines%
      (ddtlimit) CONSTANT = 3000B;                                          1B1
      (smanipumask) CONSTANT = 777748?;                                     1B2
      (optabl) CONSTANT = 24B;                                              1B3
      (global) CONSTANT = 1;                                                1B4
   % Record definitions %
      (inst10) RECORD    % PDP10 instruction parts %                       1C1
         addr10 [18],
         index10 [4],
         indir10 [1],
         accum10 [4],
         opcod10 [9];
      (instformat) RECORD                                                   1C2
         pdummy [21],
         longopcode [15];
   % Declarations %
      (regnames) _ ("R0  ", "R1  ", "R2  ", "R3  ", "R4  ", "R5  ", "R6
       ", " P  ", "MA  ", " S  ", " M  ", "RP  ", "A1  ", "A2  ", "A3
      ", "A4  ");                                                           1D1
      (loadlimit) _ 120B;                                                   1D2
      REGISTER r1 = 1, r2 = 2, r3 = 3, r4 = 4;
      REF ddtsptr;
   % Constanats %
      (strbyt) CONSTANT = 440700000001B;                                    1E1
      (leftbyteptrrj) CONSTANT = 4307B8;                                    1E2
   % Miscellaneous routines used by other NLS procedures %
      (nddtarm) PROCEDURE; % called from NLS %                              1F1
         LOCAL proc;
         LOCAL STRING nddtname[30];
         % Roll in NDDT "user" program. %
            % Increase buffer size to 40 pages. %
               IF NOT gpbsz( 40 ) THEN
                  BEGIN
                  err($"Cannot set program buffer for NDDT system!");
                  END;
            % Load the user program with NDDT code. %
               *nddtname* _ "rel-nls, nddt";
            % Get address of "entry point" calldt; put it in acalddt.
            %
               IF NOT ddtlookup($"CALLDDT", FALSE, % get procedure in
               user program. %: proc) THEN
                  BEGIN
                  err($"Cannot find NDDT entry procedure");
                  END;
               acalddt _ proc;
         sethint();
         RETURN;
         END.
      (nddtdisarm) PROCEDURE; % called from NLS %                           1F2
         unsethint();
         % reset acalddt %
            acalddt _ 0;
```

```
    RETURN;
    END.
(sethint) PROCEDURE;                                                    1F3
    %Set up control H as interrupt%
    levtb1 _ levtab.RH;
    chntab[5] _ $ddtint .V 186;
    r1 _ 485;
    r2 _ $chntab;
    !HRLI 2,levtab;
    !JSYS sir;
    r1 _ 10000005B;
    !JSYS ati;
    r1 _ 485;
    r2 _ 1810;
    !JSYS aic;
    RETURN;
    %DDT Interrupt handling stuff%
        (ddtint):                                                       1F3N1
        !SOS @levtb1; %Point to actual instruction interrupted%
        %Check for stack manipulation instruction%
            !HRR 0,@levtb1;
            !MOVE 0,@0;
            !MOVEM 0,ddtinst; %Save off inst in case we want to
            execute it later%
            !AND 0,smanipumask;
            !CAME 0,27044B7;
            !JRST ddtnt1;
            %By here, interrupted at stack manipulation
            instruction...execute it%
                !XCT ddtinst;
                !SOS @levtb1;
        (ddtnt1): %Call ddt %                                           1F3N4
            !HRL 0,acalddt;
            !HRR 0,@levtb1;
            !HLRM 0,@levtb1;
            !JSYS debrk;
    END.
(unsethint) PROCEDURE;                                                  1F4
    %remove control H as interrupt%
    r1 _ 10B;
    !JSYS dti;
    RETURN;
    END.
(octnum) PROCEDURE (value);                                             1F5
    LOCAL bp;
    %Uses ddtlit%
    bp _ r1 _ 440700000001B + $ddtlit;
    r2 _ value;
    r3 _ 8;
    IF NOT SKIP !JSYS nout THEN err($"Error in Nout");
    ddtlit.L _ slngth(bp, r1);
    RETURN($ddtlit);
    END.
(tblsearch) PROCEDURE (tbl, tblsize, entsize, value);                   1F6
    %Generalised table searcher...return TRUE, address of entrym
    and entry number if ok, FALSE otherwise%
```

```
    LOCAL c;
    REF tbl;
    c _ -entsize;
    WHILE (c _ c+entsize) < tblsize DO
       IF tbl[c] = value THEN RETURN(TRUE, $tbl[c], c/entsize);
    RETURN(FALSE);
    END.
(findbp) PROCEDURE (value);                                          1F7
    %IF value < 10 then assume it is a break point number,
    otherwise an address of an instruction%
    LOCAL bpa, bpn;
    IF value < 10 THEN
       BEGIN
       bpa _ $bptbl + value*4;
       IF [bpa] = 0 THEN RETURN(0);
       RETURN(bpa, value);
       END;
    IF NOT tblsearch($bptbl, 40, 4, value: bpa, bpn) THEN
    RETURN(0);
    RETURN(bpa, bpn);
    END.
(typval) PROCEDURE (value, mode);                                    1F8
    LOCAL bp;
    LOCAL STRING tempsr[75];
    REF db;
    db[16] _ value;
    IF mode = 1 THEN
       IF value.LH # 0 THEN
          *tempsr* _ *[octnum(value.LH)]*, ",,",
          *[octnum(value.RH)]*
       ELSE
          *tempsr* _ *[octnum(value)]*
    ELSE IF mode = 3 THEN
       BEGIN
       bp _ leftbyteptr[j + $value;
       WHILE bp.bpadr = $value DO IF ^bp # 0 THEN typech(.bp);
       RETURN;
       END
    ELSE
       ddgtsymbol($tempsr, value);
    dismes(2,$tempsr);
    RETURN;
    END.
% NDDT support %
   % Stack variables %
      (ddmrk)        EXTERNAL [10];  % mark stack %                1G1A
      (ddmrkm)       EXTERNAL _ 9;   % max stack depth %          1G1B
      (ddmrkx)       EXTERNAL _ 0;   % current stack depth %      1G1C
      (db)           EXTERNAL;       % ptr to current data block %
                                                                    1G1D
   % Breakpoint support %
      (bptbl)        EXTERNAL [40];  % breakpoint table 10*bpentsz %
                                                                    1G2A
      (ddtrloc)      EXTERNAL;       % breakpoint address %       1G2B
      (ddtrp)        EXTERNAL;       % trap address %             1G2C
      (ddtinst)      EXTERNAL;       % interrupted instruction %  1G2D
```

```
    (ddtjunk)       EXTERNAL;       % for processing breakpoints %
                                                                    1G2E
    (ddtrg1)        EXTERNAL;                                       1G2F
    (ddtrg2)        EXTERNAL;                                       1G2G
    (ddtrg3)        EXTERNAL;                                       1G2H
% NDDT hash symbol table %
    (hsymtb)        EXTERNAL [300]; % symbol table space %          1G3A
    (hstkhed)       EXTERNAL [193]; % hash table %                  1G3B
    (hsymptr)       EXTERNAL;       % ptr to symbol table entry %
                                                                    1G3C
% table of replaced procedures %
    (proctbl)       EXTERNAL [10];                                  1G4A
% misc. data %
    (acalddt)       EXTERNAL _ 0;   % addr of NDDT entry point or
    0%                                                              1G5A
    (levtbl)        EXTERNAL;       % control-h interrupt table %
                                                                    1G5B
    (dsplayflag)    EXTERNAL;                                       1G5C
      % flag for saving and restoring display areas %
% general temporary character string %
    (ddtlit)        EXTERNAL STRING [150];                          1G6A
%used by dn%
    (dnstr)         EXTERNAL STRING [50];                           1G7A
%* DDT Symbol Table Utility Routines%

(ddgtsymbol) % generates a symbolic representation for a value %
                                                                    1H1
  PROCEDURE(
  % FORMAL PARAMETERS %
    astr, % ptr to target character string %
    value); % fullword value to be converted to symbolic form %
  % DETAILED DESCRIPTION OF PROCEDURE %
    % ALGORITHM %
      % if the value word is a pdp10 instruction, then
      instructionmode is set to TRUE and the corresponding
      PDP10 assembly instruction is dis-assembled.  If not,
      then the word is decoded into halfword format if the
      left half is non zero, else it decodes the right half of
      the word %
    % NORMAL RETURNS %
      % symbol string is built in the parameter string passed
      %
    % ABNORMAL RETURNS %
      % NONE %
  % GLOBALS USED %
  LOCAL
    instructionmode; % set by getins to TRUE if word is a PDP10
    instruction, FALSE otherwise. %
  REF
    astr;
  %-------------%
  % set the result string to NULL %
    *astr* _ NULL;
  % process left half of value word, checking for inst. %
    IF value.LH # 0 THEN % may be instruction %
      BEGIN
```

```
        IF NOT (instructionmode _ getins(&astr,value) ) THEN
           BEGIN
           IF NOT ddsymget(&astr, value.LH) THEN
              *astr* _ *astr*, *[octnum(value.LH)]*;
           *astr* _ *astr*, ",,";
           END;
        END;
    % now do the right half word %
      IF NOT ddsymget(&astr, value.RH) THEN
        *astr* _ *astr*, *[octnum(value.RH)]*;
    % edit index field if PDP10 instruction %
      IF instructionmode THEN
        IF value.index10 THEN
           BEGIN
           *astr* _ *astr*, "(";
           ddsymget(&astr, value.index10);
           *astr* _ *astr*, ")";
           END;
    RETURN;
    END.

(ddsymget) % find a DDT symbol associated with a value %        1H2
    PROCEDURE(
    % FORMAL PARAMETERS %
      astr, % ptr to target char string - result appended to
      string %
      value ); % address value for which symbol is sought %
    % DETAILED DESCRIPTION OF PROCEDURE %
      % FUNCTION %
        % this routine searches the ddt symbol table for a match
        with the given value %
      % ALGORITHM %
        %  Seccessive blocks of the DDT symbol table are checked
        to see if the target value is represented by some symbol
        in the block.  If so, then the address values of the
        symbol table entries are compared with the target value
        and the symbol having a value closest to the target
        value is selected.  If no symbol is close enough to the
        target symbol (determined by the global value
        "ddtlimit") then FALSE is retruned.  If the closest
        symbol is a local symbol, then the file name is edited
        into the symbol string %
        % the search loop has been recoded using inline
        instructions and should be faster than the normal L10
        stuff, but it a bit unreadable %
        % registers are used by this routine as follows:
           r1    current pointer to symbol value. LH contains -
           length
                 remaining in the block
           r2    sought after symbol table value
           r3    -length of the current symbol table block
           r4    current symbol table test value %
      % NORMAL RETURNS %
        % TRUE -- a symbolic representation (symbol + offset)
        was generated in the parameter string %
        % FALSE -- no appropriate symbol was "close enough" to
```

```
        the value to be decoded %
      % ABNORMAL RETURNS %
        % NONE %
    LOCAL temp,lowptr,difference, closest;
    LOCAL blockptr;
    LOCAL closestblkptr;
    LOCAL STRING tempstr[6];
    REF astr, symloc;
    %----------%
    % move the compare value to r2 for faster access %
      r2 _ value;
    % if value represents a register, no search needed %
      IF r2 IN [$db, $db[15]] THEN r2 _ r2 - $db + 1;
      IF r2 IN [0, 17B] THEN
        BEGIN %Return a reasonable symbol%
        tempstr.L _ 2;
        tempstr[1] _ regnames[r2];
        *astr* _ *astr*, *tempstr*;
        RETURN (TRUE);
        END;
    % initialize for search through symbol table blocks %
      lowptr _ symloc.RH; % lowest symbol table address %;
      blockptr _ (lowptr - symloc.LH).A 18M -1; % highest symbol
      table address %
      difference _ ddtlimit;
    % loop through each block in the symbol table %
      WHILE blockptr > lowptr DO
        BEGIN
        !HLRZ r3,@blockptr; % length of block to r3 %
        r1 _ (blockptr + r3 + 4) .A 18M; % addr of first r2 in
        block %
        % check range of block %
        IF r2 INC [blockptr].RH, [r1-2]+ddtlimit] THEN
          BEGIN
          r1.LH _ r3+4; % effctive range of table %
          GOTO 11; % start searching up the blocks %
          (testdifference): % come here to check for a possible
          hit %                                          1H2M1E4
            !MOVE r4,0(r1); % get value word from symbol table
            %
            IF r2 - r4 IN [0,difference) THEN
              BEGIN
              closest _ r1;
              closestblkptr _ blockptr;
              IF (difference _ r2 - r4) = 0 THEN GOTO
              exactmatch;
              END;
            GOTO continuesearch;
          (11): % top of seach loop %                    1H2M1E5
            !CAML r2,0(r1); % compare with r2 %
            !JRST 0,testdifference;
          (continuesearch):                              1H2M1E6
            !AOBJN r1,12; % bump ptr by 1 %
          (12):                                          1H2M1E7
            !AOBJN r1,11; % fall through loop in no hits %
          END;
```

```
        blockptr _ (blockptr + r3) .A 18M;
        END;
    % exit, returning FALSE if no appropriate symbol was found %
      IF difference >= ddtlimit THEN RETURN( FALSE );
    % generate appropriate symbolic represenation %
      (exactmatch):                                                    1H2C1
      temp _ [closest.RH-1];
      IF temp .A 10B10 THEN % symbol is a LOCAL symbol %
        BEGIN
        r50toa( [closestblkptr-1] .A 32M, $tempstr); % convert
        block name %
        *astr* _ *astr*, "[, *tempstr*, "];
        END;
      % convert symbol name %
      r50toa( temp .A 32M, $tempstr);
      *astr* _ *astr*, *tempstr*; % append instruction mnemonic %
      IF difference > 0 THEN
        *astr* _ *astr*, "+, *[octnum(difference)]*;
    RETURN( TRUE );
    END.

    (getins) PROCEDURE( astr, instruction);                            1H2E
      LOCAL opcode, i , j, k;
      REF astr;
      LOCAL STRING temp[6];
      opcode _ instruction.opcod10;
      IF NOT( opcode IN [1, 700B]) OR
        ((i_optab[opcode]) = 0) THEN RETURN( FALSE );
      IF i = 1 THEN % indeterminate -look some more %
        LOOP
          BEGIN
          j _ instruction.longopcode;
          FOR k _ 0 UP UNTIL >= optab1 DO
            IF optab2[k] = j THEN
              BEGIN
              i _ optab3[k];
              EXIT LOOP 2;
              END;
          RETURN (FALSE);
          END;
      % i contains opcode mnemonic in sixbit form %
      *temp* _ NULL;
      FOR j _ 1 UP UNTIL > 6 DO
        BEGIN % convert to ascii by shifting out each digit %
        k _ i .A 6M; % pick off right hand 6 bit character %
        *temp* _ k + 32, *temp*;
        i _ shr(i, 6); % shift in next digit %
        END;
      % append to astr %
      *astr* _ *astr*, *temp*, "  ";
      % edit accumulaor field %
      IF instruction.accum10 THEN
        BEGIN
        *temp* _ NULL;
        ddsymget( $temp, instruction.accum10);
        *astr* _ *astr*, *temp*, ",;
```

```
        END;
      % indirect flag %
      IF instruction.indir10 THEN *astr* _ *astr*, "@;
      RETURN ( TRUE );
      END.
      (shl)  PROCEDURE( value, shiftcount );                    1H2R19
         % this routine performs the logical left shift of the
         specified number %
         r1 _ value;
         r2 _ 242040B6 + (shiftcount .A 18M); %construct a LSH
         instruction in r2 %
         !XCT r2; % execute the shift - result left in r1 %
         RETURN (r1);
         END.
      (shr) PROCEDURE( value, shiftcount );                     1H2R20
         % this routine performs the logical right shift of the
         specified number %
         r1 _ value;
         r2 _ 242040B6 + ((0-shiftcount) .A 18M); %construct a
         LSH instruction in r2 %
         !XCT r2; % execute the shift - result left in r1 %
         RETURN (r1);
         END.
   (r50toa) PROCEDURE (value, astr);                                1H3
      % this procedure converts a RADIX50 symbol representation into
      a NLS string %
      LOCAL j;
      REF astr;
      *astr* _ NULL;
      LOOP % convert from radix 50 to ascii %
         BEGIN
         DIV value/50B, value, j;
         IF NOT j OR astr.L = astr.M THEN EXIT LOOP;
         j _ IF (j _ j + 47) > 57 THEN j+7 ELSE j;
         *astr* _ j , *astr*;
         END;
      RETURN;
      END.
   (ddtenter) PROCEDURE (astr, value, type);                        1H4
      % this routine enters a symbol and value into the ddt symbol
      table
      the current range of the symbol table is contained in "symloc"
      the highest load address is contained in "[loadlimit].LH". %
      LOCAL ptr, i, blockptr;
      REF symloc ,astr, blockptr;
      % find end of current block -- look for a word of all zeroes %
      &blockptr _ symloc.RH + 2;
      UNTIL blockptr = 0 DO
         &blockptr _ &blockptr + 2; % try next entry %
      &blockptr _ &blockptr - 1;
      % LH of [blockptr] ccntaines -l where l is the length of the
      block%
      ptr _ (&blockptr + blockptr.LH -1) .A 18M;
      % check for table overflow %
      IF ptr < [loadlimit].LH THEN
         err( $"Symbol Table Overflow" ); % out of space %
```

```
    % update symloc to reflect new extent of table %
    symloc _ symloc - 2000002B;
    % increase length by -2 in header word %
    blockptr _ blockptr - 2B6;
    % make new header for block
      first word is zeroes
      second word contains the highest address in the block %
    [ptr] _ 0;
    BUMP ptr;
    [ptr] _  MAX([ptr+2],value);
    % add new entry to block %
    BUMP ptr;
    [ptr] _ ddtname( &astr, type);
    BUMP ptr;
    [ptr] _ value;
    RETURN( TRUE );
    END.
(ddtlookup) PROCEDURE (astr, skiplocals);                          1H5
    % IF the second argument is TRUE then the mark stack is
    ignored and the entire DDT symbol table is searched bottom up
    %
    LOCAL i,j,v,blockptr,address,value;
    REF astr, symloc;
    % convert symbol to a radix 50 value %
    v _ ddtname(&astr, global);
    % now search down through the symbol table %

    IF NOT skiplocals THEN
       FOR i _ ddmrkx-1 DOWN UNTIL < 0 DO
          % search through marked table for local symbols first %
          IF ddtsrchblk( ddmrk[i], v .X 14B10: value,address)
             % search for global symbols in any marked blocks next
             %
             OR ddtsrchblk( ddmrk[i], v: value,address)
             THEN RETURN(TRUE, value, address);

    % now search the whole damn table %
    j _ symloc.RH; % lowest symblol table address %
    blockptr _ (j - symloc.LH) .A 18M - 1; % higheat sym table
    address %
    WHILE blockptr > j DO
       BEGIN
       IF ddtsrchblk( blockptr, v : value, address) THEN
          RETURN( TRUE, value, address);
       blockptr _ (blockptr + [blockptr].LH) .A 18M;
       END;
    RETURN(FALSE);
    END.
(mrkglookup) PROCEDURE (astr);                                     1H6
    % Looks for the symbol specified by astr as a global in the
    mark stack%
    %used by gpget to get the subsystem dispatch record given the
    subsystem name%
    LOCAL i,j,v,blockptr,address,value;
    REF astr, symloc;
    % convert symbol to a radix 50 value %
```

```
    v _ ddtname(&astr, 1);
    % now search down through the symbol table %

    FOR i _ ddmrkx-1 DOWN UNTIL < 0 DO
       % search for global symbols in any marked blocks first %
       IF ddtsrchblk( ddmrk[i], v: value,address)
          THEN RETURN(TRUE, value, address);

    RETURN(FALSE);
    END.
(ddtmark) PROCEDURE (astr);                                        1H7
    % this routine markes the current position in the ddtsymbol
    table and creates a new block with the name given the value
    saved in the mark stack is the block pointer for the marked
    block%
    LOCAL namevalue,ptr1,blockptr;
    REF symloc, astr;
    % search through the ddt symbol table looking for the named
    block %
    namevalue _ ddtname( &astr, 0) .A 32M;
    blockptr _ (symloc.RH - symloc.LH - 1) .A 18M;
    LOOP BEGIN
       IF [blockptr-1] = namevalue THEN EXIT LOOP % found block %
       ELSE BEGIN
          blockptr _ (blockptr + [blockptr].LH ) .A 18M;
          IF blockptr <= symloc.RH THEN
             BEGIN % not found -- must allocate a new one %
             % allocate 4 words for new header is there is room %
             symloc _ symloc - 4000004B;
             ptr1 _ symloc.RH;
             IF ptr1 < [loadlimit].LH THEN err($"No More Room For
             Symbol Table");
             % initialize the 4 word header %
             [ptr1] _ 0;
             BUMP ptr1;
             [ptr1] _ 0;
             BUMP ptr1;
             [ptr1] _ namevalue; % radix50 name %
             BUMP ptr1;
             [ptr1].LH _ -4;
             [ptr1].RH _ [ptr1+2];
             EXIT LOOP;
             END;
          END;
       END;
    ddmrk[ddmrkx] _ blockptr; % save away current blockptr %
    IF (ddmrkx _ ddmrkx + 1) > ddmrkm THEN
       BEGIN
       ddmrkx _ ddmrkm;
       err($"Symbol Table Mark Stack Overflowed");
       END;
    RETURN;
    END.

(ddtname) PROCEDURE (astr, type);                                  1H8
    % this routine returns a radix 50 interpretation of the string
```

```
    and incorporates the type information with it to produce a
    ddtsymbol.  Only Upper or Lower case alphabetic and numeric
    characters are allowed. %
    LOCAL value, i ,j;
    REF astr;
    % make radix 50 value from symbol %
    value _ 0;
    FOR i _ 1 UP UNTIL > MIN(astr.L,6) DO
       BEGIN
       j _ CASE j_*astr*[i] OF
          INT 60B, 71B]:   % digits %
             j - 57B;
          INT101B,132B]:   % Upper case alpha %
             j - 66B;
          INT141B,172B]:   % lower case alpha %
             j - 126B;
          ENDCASE err($"Invalid char in identifier");
       value _ 40 * value + j;
       END;
    % set appropriate type mask %
    value _ value .V (IF type = global THEN 4B10 ELSE 10B10);
    RETURN (value);
    END.
(ddtsrchblk) PROCEDURE (blockptr, symbolhash);                    1H9
    % this  routine searches through a specified block of the
    symbol table, looking for a match with a given symbol value.
    Inline coding has been used to speed up the search loop and
    registers are used as follows:
       r1  pointer to current location in the symbol table
       r2  test value
       r3  temporary value %
    !HLRZ r3,@blockptr; % get length of symbol table block to r3 %
    r1 _ (blockptr + r3 + 3);
    !ADDI r3,4;
    !HRL r1,r3; % set length of r1 %
    r2 _ symbolhash; % load up with symbol hash value %
    IF [r1-2] # 0 THEN err( $" bad blockptr in ddtsrchblk ");
    (srch1): !CAMN r2,0(r1); % compare %                          1H9H
       !JRST 0,srchhit; % found it %
       !AOBJN r1,srch2;
    (srch2): !AOBJN r1,srch1; % loop to srch1 if any left to
    compare %                                                     1H9I
    RETURN (FALSE);
    (srchhit): % found the symbol %                               1H9K
       RETURN( TRUE, [r1+1], r1+1)
    END.
(ddtpop) PROCEDURE;                                               1H10
    % this routine pops the current routine from the marked symbol
    table stack %
    % decrement stack index and error check %
    IF (ddmrkx _ ddmrkx - 1) < 0 THEN
       BEGIN
       ddmrkx _ 0;
       err($"Symbol Table Pop Stack Error");
       END;
    % recover symbol table space if at the end of the table %
```

```
        ddtchkp( ddmrk[ddmrkx] );
        RETURN;
        END.
      (ddtchkp) PROCEDURE (blockptr);                                  1H11
        LOCAL ddtloc;
        REF symloc;
        % this procedure checks to see if the symbol table block
        defined by blockptr is at the bottom of the DDT symbol table,
        and if so, it recovers the space by modifying symloc
        appropriately %
        IF (blockptr + [blockptr].LH + 1).A 18M = symloc.RH
           THEN BEGIN
           symloc.LH _ symloc.LH - [blockptr].LH;
           symloc.RH _ blockptr + 1;
           % reset ddt's alt i -1 pointer also %
              ddtloc _ ddtsptr;
              [ddtloc] _ symloc;
           END;
        RETURN;
        END.
FINISH of besymtab
```

```
< NINE, BINTNLS.NLS;35, >, 30-May-78 16:25 HGL ;;;;
FILE bintnls  %  <ARCSUBSYS>L109  to <RELNINE>bintnls % % (arcsubsys,
L109,) (RELNINE,bintnls,) %
ALLOW!
%.....Compile-time switches.....%
   SET NSW = FALSE;  %TRUE => (RELNINE,wmbintnls.rel,) %
%......Declarations......%
   EXTERNAL tt, gt, st, nswnls;
   (randm) CONSTANT = 0;                                                4B
   (sequential) CONSTANT = 1;                                           4C
   (cmdisize) EXTERNAL CONSTANT = 30;  %size of display package command
   list%                                                                4D
   (arglsize) EXTERNAL CONSTANT = 10;  %size of display package
   argument list%                                                       4E
   REF prwndw, dtwndw;
   SET lowsegtop = 120B, symtbltop = 116B;
%  initialization which once was in FINTNLS %
   (backstart) PROCEDURE; %initialize the world%                        5A
      %here when NLS back-end created.  intializes back-end local
      environment and goes to routine to set up middle-end and never
      returns from middle-end%
      %----------------%
      LOCAL sysmsg, priority3 = 3;
      LOCAL i, subsname REF;
      LOCAL CONSTANT ckeep = TRUE;
      INVOKE (strterr);
      % initialize subsystem default directory to be that from which we
      came unless we're running in the NSW in which case the subsystem
      default directory has been set at presave time %
         IF NOT nswsw THEN dirfrom($subdfdir);
      % Init middle end if not done before %
         IF npkgs = 0 THEN
            BEGIN
            initmiddle($nlsprname, $cnvarg, $cmdfinish, 0, $nlserr,
            $xinit, $xfterm);
            % create a package for NLS %
               curpkg _ prvpkg _ crtlpkg($pkgname, $base, $subdfdir,
               ckeep, $pnhasht, lprime);
            % create a package for all subsystems loaded with NLS %
               i _ 0;
               WHILE &subsname _ nlssubs[i] DO
                  BEGIN
                  crtlpkg(&subsname, nlssubs[i + 1], $subdfdir, 0, 0,
                  0);
                  i _ i+2;
                  END;
            END;
      % Set up PCP required information %
         IF NOT pcpinit() THEN err($"pcp init error");
      % Get information about  local (backend) host and monitor number,
      etc. %
         beinfo();
      % Initialize interrupt system%
         setinterrupts();
      % transfer back to middle end to start command parsing%
         DROP( strterr );
```

```
      execmiddle(howpcp, priority3);
      RETURN;  %Here only in one-fork version%
      (strterr) CATCHPHRASE(:sysmsg);                                  5A12D
         BEGIN
         IF sysmsg AND [sysmsg].L <= [sysmsg].M AND [sysmsg].L IN
         [1, 2007 THEN
            btypestr(sysmsg);
         !JSYS 147B; %Reset jsys-- cannot use symbol because of
         conflict%
          LOOP !haltf();
            %In case someone is foolish enough to try to contnue%
         CONTINUE;
         END;

   END.

(beinfo) PROCEDURE; % Set up backend information %                      5B
   lhostn _ hostnumber(); %save logical host number%
   oprtty _ IF lhostn = utilhost THEN utiloprtty ELSE arcoprtty;
   tenex _ tenexverno();
   inptrf _ FALSE; %rubout flag%
   %misc. for background, load averag checks%
      nsdcktime _ 0; %time (gtad) to check if system going down%
      !sysgt(getsbn($"SYSTAT"));
      !HRLI R2,14B; lavtabad _ R2; %getab table address of load
      average%
   %Check to see if we are to rur library routines%
      %libfig _ IF libflg THEN libflg ELSE checklib();
      IF libflg = 1 TREN
         BEGIN
         setpriority( 202B);
         libflg _ 11B;
         END;%
   % initialize measurement tables %
      IF msflag THEN intmeas();
   % initialize globals about monitor tables %
      !sysgt(getsbn($"JOBTTY"));
      ttytbl _ R2.RH;  % table number %
      !sysgt(getsbn($"JOBDIR"));
      jobtbl _ R2.RH;  % table number %
   % initialize special back end globals %
      nortnrings _ FALSE;
   RETURN;
   END.

(feinfo)   % GB: ;  get information from the front-end %
PROCEDURE (wchrct REF % => rvalue BOOLEAN %);                          5C
   % Procedure description
      FUNCTION
         Do calls on front-end to get information.
            get-window characteristics
      ARGUMENTS
         wchrct:  window chacterstic LIST filled by the front-end
      RESULTS
         rvalue:  TRUE if calls on front-end succssful.  otherwise
         FALSE
```

```
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %
   % Declarations %
      LOCAL rvalue;
   %Get window charactersitics%
      dfegtw(&wchrct);
   % Return %
      RETURN;
   END.

(wminfo) PROCEDURE;                                                    5D
   % the following information should be obtained and set up at
   create process time.  The information will be optained from the
   WM.  For now, we get it by the old methods. %
      %
      userstr
      identstr
      cuno
      cinit
      %
      !gjinf();
      cuno _ R1;
      conndir _ R2;
      IF SKIP !dirst($userstr + chbmty, cuno) THEN
         userstr.L _ slngth($userstr + chbmty, R1)
      ELSE *userstr* _ NULL;
      IF NOT initsr.L THEN getuident(cuno);  %get user's ident%
      identwheel _ FALSE;
   RETURN( TRUE );
   END.

(initid) PROCEDURE; % open initial file and set up return stacks % 5E
   LOCAL da REF, fileno, fl, srr;
   LOCAL STRING locstr[200], gs[100];
   REF fl, srr;
   %Get display area of primary window%
      &da _ prwndw.widdarea;
   %PROTOCOL%
      curmkr _ 0;
      curmkr.stfile _ da.dacsp.stfile _ fileno _ openid();
         curmkr.stpsid _ origin;
         curmkr[1] _ 1;
      &fl _ flntadr(fileno);
      %-NSW%                                                           5E5D
         filnam( fileno, $locstr );
      %-NSW%                                                           5E5E
      %+NSW%                                                           5E5F
         rfilnam( fileno, $locstr );
      %+NSW%                                                           5E5G
   pushfrring(da.dalink, $locstr, fileno);
   readfrring(da.dalink, 0 : &srr);
   pushsrring(&srr, da.dacsp, da.daccnt, da.davspec, da.davspc2);
   RETURN;
```

```
    END.

(tenexverno) PROCEDURE;  %get version number of current TENEX%      5F
    %always returns 13301 following jdh's change to <nls, sintnls,
    tenexverno> %
    RETURN(13301);
    %following code is comment out
    %%returns number of te form 12900 for tenex version 1.29.00%%
    %%build 6-bit string for sysgt jsys%%
       R1 _ getsbn($"VERNUM");
    !JSYS sysgt;
    RETURN(R1);
    %
    END.

(hostnumber)PROC; %get logical host number from tenex or global%    5G
    IF nonetworkflag THEN RETURN(nonetworkflag);
    R1 _ getsbn($"LHOSTN");
    !JSYS sysgt;
    !HRLI R1,0;
    !HRR R1,R2;
    IF R2 AND SKIP !JSYS getab THEN RETURN(R1);
    RETURN (0);
    END.
% text area and display area initialization %
 (setdis)   % CL:LB ;  set up window and display areas %
 PROCEDURE;                                                          6A
    % Procedure description
       FUNCTION
          Get window characteristics from the frontend.
          Set up the primary window, the detault text window and the
          viewspec window.  Set up the display areas for the default
          text window and primary window.  There is no display area
          for the viewspec window.
       ARGUMENTS
          none
       RESULTS
          none
       NON-STANDARD CONTROL
          none
       GLOBALS
          SET
             dacnt -- no. of active da's
             wacnt -- no. of active wa's
             tda -- address of primary window display area (DNLS)
                -- address of only display area (TNLS)
             msgda, ttyda -- address of default text window display
             area
       %
    % Declarations %
       LOCAL LIST wchrst;  %window charactersitics%
       REF dspcmd, dsparg, prwndw, dtwndw, msgda, tda, ttyda, dttyda;
    % invoke catchphrase %
       INVOKE(initcat);
    % get frontend information %
       teinfo($wchrst );
```

```
   %Initialize initial window areas and set display type%
      wacnt _ 0;
      intwa ($wchrst);
   %Initialize display area default text window (tty window) in
   special display area block %
      &dttyda _ &ttyda _ dtwndw.widdarea _ $nlsdas;
      ttyda.dawid _ dtwndw.widindex;  %set window id in da%
      intdafl ( &ttyda );
      ttyda.daseq _ TRUE;  %sequential window%
   %Initialize display area for primary window %
      dacnt _ 0;
      prwndw.widdarea _ &tda _ newda();
      tda.dawid _ prwndw.widindex;
      intdafl ( &tda );
      IF nlmode = typewriter THEN
         tda.daseq _ TRUE;  %sequential window%
   %Initialize other windows if any%
   %set display  parameters%
      dpset(dspallf, endfil, endfil, endfil) ;
   % set current window id global to primary window id %
      cwndow _ prwndw.widindex;
   % Return %
      #wchrst# _ ;
      DROP (initcat);
      RETURN;
   % catchphrases %
      (initcat)  CATCHPHRASE();                                    6A12A
         CASE SIGNAL OF
            = return, = unwind :
               BEGIN
               DISABLE (initcat);
               NULL-LISTS;
               END;
         ENDCASE CONTINUE;
   END.

(setabs)     % new bit table version of settabs %                    6B
   PROCEDURE
      (tabtbl);       % address of tab table %                     6B1A
   LOCAL twrd1, twrd2, twrd3;
   REF tabtbl;
      !gjinf();                % get user info %
      IF R4 = -1 THEN RETURN;          % NOP if detached %
   %This routine should be doing calls on the front-end to get user
   and tab information%
   % convert NLS internal form to TENEX form %
      R1 _ tabtbl;
      R2 _ tabtbl[1];
      !SETCA 1,0;              %NLS bits are inverted%
      !SETCA 2,0;
      twrd1 _ R1;
      twrd2 _ R2;
      R2 _ tabtbl[2];
      !SETCA 2,0;
      twrd3 _ R2;
   % set tabs unless monitor words are the same %
```

```
      !gtabs(18M);
      IF twrd1 = R2 AND twrd2 = R3 AND twrd3 = R4 THEN RETURN
      ELSE !stabs(18M, twrd1, twrd2, twrd3);
   RETURN;
   END.

(intdafl) %initialize da record to standard values for a text area%
                                                                    6C
   PROCEDURE (da);
   LOCAL fv, index, cpos;
   REF da;
   %dislayarea record starting values%
      da.daseg _ FALSE;  %random display area%
      da.daauxiliary _ FALSE;    %for display purposes%
      da.davspec _ stdvsp;
      da.davspc2 _ stdvsp[1];
      da.dastrt _ 0;  %no string table yet%
      da.dacsp _ orgstid; %fileno later%
      da.daccnt _ 1;
      da.daind _ indcnt * hinc;
      da.damind _ tamind * da.daind;
      da.dalftjst _ 0;
      % set margins and related fields to default (entire primary
      window) %
         setdabnd(&prwndw, &da);
      da.daempty _ FALSE;
      da.daexis _ TRUE;
   %set default tabstop values%
      da.datab0 _ stdtab;
      da.datab1 _ stdtab[1];
      da.datab2 _ stdtab[2];
      IF nlmode = typewriter AND NOT autostrt THEN setabs($stdtab);
   % set previous viewspec words %
      (da.dapvs, da.dapvs2) _ (da.davspec, da.davspc2);
   RETURN;
   END.

(intwa)   %  Set up initial window areas and terminal type  %
PROCEDURE (wchrct REF LIST ) ;                                      6D
   % Procedure description
      FUNCTION
         Process data obtained from initial get window
         characterstics call.  Allocates wa for default text window
         and other windows if any.
      ARGUMENTS
         wchrct:  window charactertics list from front-end
      RESULTS
         none
      NON-STANDARD CONTROL
         SIGNALS generated
            ABORT(ermillarg,$"Invalid terminal type - intwa") if
            non-NLS terminal
      GLOBALS
         Set
            nlmode:  typewriter or fulldisplay
            nodisplay:  TRUE if typewriter terminal, otherwise FALSE
```

```
            prwndw:  Address of primary window wa
            vswndw:  Address of viewspec window wa.
            dtwndw:  Address of default text window wa
         %
      % Declarations %
         LOCAL
            hdplxty = 1,  %half duplex typewriter terminal%
            fdplxty = 2,  %full duplex typewriter terminal%
            fdplxds = 3;  %full duplex display terminal%
         REF prwndw, vswndw;
      % Set terminal type%
         CASE ELEM #wchrct# [1] OF
            = hdplxty, = fdplxty:  %full duplex typewriter%
               BEGIN
               nlmode _ typewriter;
               nodisplay _ TRUE;
               setabs($stdtab);
               END;
            = fdplxds:  %full duplex display%
               BEGIN
               nlmode _ fulldisplay;
               nodisplay _ FALSE;
               END;
            ENDCASE ABORT(ermillarg,$"Invalid terminal type - intwa");
      % Set up default text window %
         setwa(&dtwndw _ newwa(), ELEM #wchrct#[2]);
         dtwndw.wfrozen _ TRUE;
      % Set up other windows %
         IF nlmode # typewriter THEN
            BEGIN
            setwa(&prwndw _ newwa(), ELEM #wchrct#[3]);  %Primary%
            prwndw.wfrozen _ FALSE;
            IF wchrct.L >= 4 THEN  % check for viewspec window %
               setwa(&vswndw _ newwa(), ELEM #wchrct#[4])
            ELSE &vswndw _ 0;
            END
         ELSE &prwndw _ &dtwndw;
      % Return %
         RETURN;
      END.

% initialization for calls on external procedures %
% user ident stuff--  delete when we talk to WP %
   (getuident)   % CL: ;  get user's ident %
   PROCEDURE (user);                                                    8A
      % Procedure description
         FUNCTION
            get user's ident.
            If NSW, use 'nswnode'
            If or NLS wheel , ask user for ident.  Otherwise, open
            group.index file and check user's ident.  If there is only
            one ident associated with this directory, use it.  If there
            are several, ask the user for his ident and check that it
            is in the list.  If there are no idents associated with
            this login directory, go to 'err'.
         ARGUMENTS
```

```
        user:  login directory number
      RESULTS
        proc-value
      NON-STANDARD CONTROL
        none
      GLOBALS
        none
      %
    % Declarations %
      LOCAL i, jfn, chain, bp, id[2], wheelf, pcap, winit, idfrflag;
      LOCAL STRING errorstr[200];
      LOCAL validid _ FALSE, savechain;
    wheelf _ IF (pcap _ enablw()) = -1 THEN 0 ELSE 1;
    disablw(pcap:=-1);
    IF nswsw THEN *initsr* _ *[nswnode]*
      ELSE IF nwheelf THEN
        idfrmuser($initsr)  %NSW or NLS wheel --ask user%
        ELSE  %not NSW or NLS wheel -- try file%
          BEGIN
          idfrflag _ nwheelf;
          jfn _ 0;
          INVOKE( clsjfn );
          %read id from file or get from user%
            IF NOT idfrflag AND NOT (jfn _ sgtjfn(1B11,
            $grpfname, $errorstr)) THEN
              idfrflag _ TRUE;
            IF NOT idfrflag THEN
              BEGIN
              IF NOT (jfn _ sgtjfn(1B5, $grpfname, $errorstr))
              THEN
                err($"File associating login directories with
                NLS idents is bad.  Please report to ARC
                personnel.");
              IF NOT sysopen(jfn, rthawed, bintyp, $errorstr)
              THEN
                err($"File associating login directories with
                NLS idents is bad.  Please report to ARC
                personnel.");
              %get ident from group.index file or user%
                IF NOT SKIP !sfptr(jfn, user*4) THEN
                  err($"File associating login directories
                  with NLS idents is bad.  Please report to
                  ARC personnel.");
                !bin(jfn); %get user directory number and group
                number%
                IF R2.LH NOT= user THEN
                  err($"File associating login directories
                  with NLS idents is bad.  Please report to
                  ARC personnel.");
                !bin(jfn); %get first word of ident or link to
                chain of idents%
              END;
            IF R2 = 0 OR idfrflag THEN
              BEGIN
              IF wheelf OR idfrflag THEN idfrmuser ($initsr)
              ELSE err($"no idents associated with this login
```

```
            directory");
            END
         ELSE IF R2.LH = 18M THEN %chain of idents%
            BEGIN
            savechain _ R2.RH;
            DO  %loop until user type in valid ident%
               BEGIN
               %get ident from user%
                  idfrmuser ($initsr);
               chain _ savechain;
               LOOP
                  BEGIN
                  IF NOT SKIP !sfptr(jfn, chain) THEN
                     err($"cant set position in file
                     associating login directories an idents.
                     Please report to ARC personnel.");
                  !bin(jfn); %first word of ident%
                  id _ R2;
                  !bin(jfn); %second word of ident%
                  id[1] _ R2;
                  IF initsr[1] = id AND initsr[2] = id[1] THEN
                     BEGIN
                     validid _ TRUE;
                     EXIT LOOP; %found a match, everything is
                     ok%
                     END;
                  !bin(jfn); %follow chain to next entry%
                  IF (chain _ R2.RH) = 0 THEN %end of chain%
                     BEGIN
                     IF NOT wheelf THEN
                        dismes(1,$"ident not in list
                        associated with this login
                        directory");
                     EXIT LOOP;
                     END;
                  END;
               END
            UNTIL validid;
            END
         ELSE %get the ident%
            BEGIN
            initsr[1] _ R2;
            !bin(jfn);
            initsr[2] _ R2;
            bp _ chbmty + $initsr;
            FOR i _ 1 UP UNTIL > 10 DO
               IF ^bp = 0 THEN EXIT LOOP;
            initsr.L _ i-1;
            END;
         CASE initsr.L OF
            > 4:
               err($"ident too long for use with NLS (max
               length is 4 characters)");
            < 1: err($"Illegal ident.  Please report to ARC
            system personnel.");
            ENDCASE;
```

```
            IF jfn THEN sysclose(jfn);
            jfn _ 0;
            END;
   %now put initials into cinit, packed into 5-bit%
      cinit _ setcinit($initsr);
   %see if pseudo wheel%
      i _ 0;
      WHILE NOT nwheelf AND (winit_nwheecinitsⅠi:=i+1⅁) # 0 DO
         IF winit=cinit THEN nwheelf_1;
   DROP( clsjfn );
   % Return %
      RETURN;
   (clsjfn) CATCHPHRASE;                                          8A10
      BEGIN
      DISABLE (clsjfn);
      IF jfn THEN sysclose(jfn);
      jfn _ 0;
      CONTINUE;
      END;
   END.

(setcinit) PROCEDURE (strng);                                      8B
   LOCAL count, char, val;
   REF strng;
   count _ val _ 0;
   UNTIL (count _ count + 1) > strng.L DO
      BEGIN
      CASE (char _ *strng*Ⅰcount⅁) OF
         IN Ⅰ'A, 'Z⅁: char _ char-100B;
         IN Ⅰ'2, '6⅁: char _ char - 27B;
         ENDCASE err($"Illegal character in id");
      CASE count OF
         =1: val.cint1 _ char;
         =2: val.cint2 _ char;
         =3: val.cint3 _ char;
         ENDCASE val.cint4 _ char;
      END;
   RETURN (val);
   END.

(idfrmuser)   % LB: collect ident from user %
PROCEDURE (str REF);                                               8C
   % Procedure description
      FUNCTION
         Do HELP call on FE to get ident from user.  Calls itself
         recursively if HELP call fails.
      ARGUMENTS
         str:  address of string in which to put user ident
      RESULTS
         none
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %
   % Declarations %
```

```
      LOCAL LIST res1[1];  %results of HELP call%
    % Ask user for ident via HELP call to FE %
      IF helpfe(htypef, 0, 0, $res1) THEN
         *str* _ *[ ELEM #res1#[1] ]*  %successful HELP call%
      ELSE idfrmuser(&str);  %HELP call failed - ask again%
      astruc(&str);  %force to upper case%
    % Return %
      RETURN;
    END.

% interrupts -- new version %
  (setinterrupts)PROC;                                                    9A
    %set up illegal instruction, page write etc. interrupt%
      %IF YOU CHANGE THIS CODE, SEE ALSO (utilty, trapcc) AND
      (utilty, notrapcc)%
      chntab[9] _ $sysovr .V 1B6;
      chntab[11] _ $iodaterr .V 1B6;
      chntab[15] _ $ilspsi .V 1B6;  % illegal instruction %
      chntab[17] _ $wrtpsi .V 2B6;
      R1 _ 4B5;
      R2 _ $chntab;
      !HRLI R2, levtab;
      !JSYS sir;
    % special stuff for control-c (^C)%
      R1 _ 4B5;
      !JSYS rpcap;
      R3 _ R3 .V 4B11;
      !JSYS epcap; %Permits process to assign ^C for pseudo
      interrupt%
    %enable interrupts%
      R1 _ 4B5;
      !JSYS eir;
    %activate io data error and write pseudo interrupt%
      R1 _ 4B5;
      R2 _ 10586; %channel 11, 15, 17%
      !JSYS aic;
    RETURN;
    END.

% save stuff %
  (savent) PROC; %Dummy procedure head-- site of pre-save entry points
  %                                                                     10A
    (st): % entry point to do partial initialization prior to ssave %
                                                                        10A1
      bndchk _ FALSE;  % dont do boundary checks %
      GOTO tt;
    (qt): % entry point to do partial initialization prior to ssave %
                                                                        10A2
      bndchk _ TRUE;  % do boundary checks %
      GOTO tt;
    (tt): % entry point to do partial initialization prior to ssave %
                                                                        10A3
      startup _ $saveit;
      recover _ $horrible;
      % set things up so calls can be made %
        !JRST 110start;
```

```
    END.

(saveit) PROCEDURE; %code used at save time%                              10B
    LOCAL char, sysmsg;
    LOCAL i, subsname REF;
    LOCAL CONSTANT ckeep = TRUE;
    INVOKE( saverr );
    presave();  %do general presave stuff%
    LOOP
        BEGIN
        btypestr($"Enable Journal System?? ");
        !pbin();
        CASE R1 OF
            = 'y, = 'Y: %yes%
                BEGIN
                jdebug _ 0;
                btypestr($"es.
                    Password for Directory Journal (CR for default) = ");
                *jnlpsw* _ NULL;
                LOOP
                    BEGIN
                    char _ getcr();
                    IF char = CR THEN EXIT;
                    *jnlpsw* _ *jnlpsw*, char;
                    END;
                IF jnlpsw.L = 0 THEN *jnlpsw* _ "JPD";
                EXIT LOOP;
                END;
            = 'n, = 'N: %no%
                BEGIN
                btypestr($"o.
                ");
                *jnlpsw* _ "JPD";
                jdebug _ 0;  %debugging journal mode%
                EXIT LOOP;
                END;
            ENDCASE
                BEGIN
                btypestr($" ? (y or n)
                ");
                REPEAT LOOP;
                END;
        END;
    LOOP
        BEGIN
        btypestr($"Enable Help System?? ");
        !pbin();
        CASE R1 OF
            = 'y, = 'Y: %yes%
                BEGIN
                hdebug _ 0;
                btypestr($"es.
                ");
                EXIT LOOP;
                END;
            = 'n, = 'N: %no%
```

```
            BEGIN
            btypestr($"o.
            ");
            hdebug _ TRUE;  %help system not enabled%
            EXIT LOOP;
            END;
         ENDCASE
            BEGIN
            btypestr($" ? (y or n)
            ");
            REPEAT LOOP;
            END;
   END;
LOOP
   BEGIN
   btypestr($"startup message?? ");
   !pbin();
   CASE R1 OF
      = 'y, = 'Y: %yes%
         BEGIN
         btypestr($"es.
         ");
         gtintmsg();
         EXIT LOOP;
         END;
      = 'n, = 'N: %no%
         BEGIN
         btypestr($"o.
         ");
         EXIT LOOP;
         END;
      ENDCASE
         BEGIN
         btypestr($" ? (y or n)
         ");
         REPEAT LOOP;
         END;
   END;
% Set entry vector %
   entvec _ $nswnls;
   setvec();
% initialize 110 loader %
% initialize user program stuff %
   %gpgmrst();%%Commented out until code fixed. %
DROP( saverr );
% save registers 3-15 in image registers for restoration at
initialization %
   !MOVEM R15,stac15;
   !HRRI R15,stac3;
   !HRLI R15,3;
   !BLT R15,stac14;
   !MOVE R15,stac15;
% save core image as disk file %
   nisave();
% terminate partial initialization %
   LOOP !haltf();
```

```
   (saverr) CATCHPHRASE(:sysmsg);                                    10B16
      BEGIN
      IF sysmsg AND [sysmsg].L <= [sysmsg].N AND [sysmsg].L IN [1,
      200] THEN
         btypestr(sysmsg);
      !JSYS 147B; %Reset jsys-- cannot use symbol because of
      conflict%
       LOOP !haltf();
         %In case someone is foolish enough to try to contnue%
      CONTINUE;
      END;

   END.

(presave)   % LB: ;  code executed at presave time %
PROCEDURE;                                                           10C
   % Procedure description
      FUNCTION
         Check for loading page boundaries overflow and do presave
         code that is common for all protocols
      ARGUMENTS
         none
      RESULTS
         none
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %

   %Declarations%
      LOCAL i, subsname REF;
   % high que so this stuff goes fast %
      setpriority(202B);
   % initialize terminal stuff if not running detached%
      !gjinf();  %get job info%
      IF R4 NOT= -1 THEN  %R4 has -1 if detached%
         BEGIN  %R4 has terminal number - job is not detached%
         !rfmod(18M); % read mode word for controlling tty %
         !sfmod( 18M, R2 .V 1704); % set to wake up all characters %
         END;
   % check for loading page boundaries overflow %
      intmsf _ FALSE;  % no warning initially %
      IF bndchk THEN
         BEGIN
         IF lowsegtop.LH >= symtbltop.RH THEN
            BEGIN
            loderrm($"CODE RUNS INTO SYMBOL TABLE");
            intmsf _ TRUE;
            END
         ELSE
            IF (lowsegtop.LH + 2000B) >= symtbltop.RH THEN
               BEGIN
               loderrm($"LESS THAN 2 PAGES FOR NEW SYMBOLS");
               intmsf _ TRUE;
               END;
```

```
        IF intmsf THEN *intmsg* _ CR, LF, "WARNING: SEE LOAD MAP
        FOR ERROR MESSAGES", 0;
        END;
    % initialize free storage zone %
     makezone($dspblk, 10000B, 8, $dspbke - $dspblk-1);
    % host dependent%
     CASE (lhostn _ hostnumber()) OF
        =isichost: oprtty _ arcoprtty;
        =office1: NULL;
        =isirlhost: %NELC, was for ISID before NOV 76%
           %may want it for NELC%
           BEGIN
           nojournalflag _ TRUE;
           cpcldelim _ cpcrdelim _ "$;
           END;
        =isidhost, =isiehost: %may want to drop isid, depending on
        what isi does when isid goes to San diego%
           BEGIN
           tops20flag _ TRUE;
           END;
        =sriai20host: %20-50%
           BEGIN
           tops20flag _ TRUE;
           END;
        =sriaikahost:
           BEGIN
           *idnfname* _ "<NETSYS>IDENTS.MASTER";
           nojournalflag _ TRUE;
           END;
        =nsahost:
           BEGIN
           outremsiteflag _ inremsiteflag _ FALSE;
           *prtdir* _ "PRINTER";
           oprtty _ nsaoprtty;
           END;
        ENDCASE
           BEGIN
           *opname* _ "<SUBSYS>NOUTPRC.SAV
           ";
           *subdir* _ "SUBSYS";
           nojournalflag _ TRUE;
           END;
      IF tops20flag THEN
         BEGIN
         fvrdchar _ ". ;
         t2fnsupdate($t2fsbegin+1, $t2fsend); %change semicolons to
         periods in system filenames%
         cpcldelim _ cpcrdelim _ "$;
         *savext* _ "EXE";
         END;
      IF nswsw THEN
         BEGIN  %change globals for NSW%
         opnmod _ readwrite;  %access mode for original file open%
         uoext _ $"UO";  %extension for USEROPTION file%
         % change directory for compilers and compactor and set
         subsystem default directory to relnine %
```

```
            *1101ink* _ "<relnine, 1109,>";
            *cmllink* _ "<relnine, cml10,>";
            *subdfdir* _ "relnine";
         END;
   % initialize sequence work areas %
      sqinit();
   % initialize system default viewspecs %
      defvs1 _ 0;
      defvs2 _ 0;
      defvs1.vstrnc _ defvs1.vslev _ 63;
      defvs1.vsindf _ defvs1.vsnamf _ defvs1.vsdaft _defvs1.vsnamf _
      defvs1.vspagf _  TRUE;
   % initialize file stuff %
      filinit();
   % initialize strings for ident system%
      *idnamdel* _ "?, CR, EOL;
   % random stuff %
      exquery _ FALSE;        % normal is not query %
   % Return %
      RETURN;
   END.

(setpriority)PROC(qcode);                                              10D
   LOCAL pcap;
   IF (pcap _ enablw()) = -1 THEN RETURN;
   R1 _ 485;
   R2 _ qcode;
   !JSYS 243B;
   IF pcap # -1 THEN disablw(pcap);
   RETURN;
   END.
(loderrm)PROCEDURE(str);                                               10E
   %put out 4-line loading message%
   REF str;
   crlf(); crlf();
   typeas($"**********************************");
   crlf();
   typeas(&str);
   crlf();
   typeas($"**********************************");
   crlf();
   RETURN;
   END.
(nlsave) PROCEDURE; % save nls as disk file %                          10F
   LOCAL jfn;
   LOOP % get save file name from user %
      BEGIN
      btypestr($"save filename: ");
      IF NOT SKIP !gtjfn(400003B6,100000101B) THEN
         BEGIN
         btypestr($" ?");
         crlf();
         REPEAT LOOP;
         END;
      jfn _ R1;
      EXIT LOOP;
```

```
      END;
   % save core image as dsk file %
      !ssave(400000B6 .V jfn.RH, 777000B6 .V 520000B, 0);
         % MODIFY BOUNDS LATER %
   RETURN;
   END.

(t2fnsupdate) PROCEDURE(tabstrt, tabend);                              10G
   %change first semicolon in filename string to period for block of
   strings%
   LOCAL TEXT POINTER z1, z2;
   REF tabstrt;
   WHILE tabstrt AND &tabstrt < tabend DO
      BEGIN
      IF FIND SF(*tabstrt*) [";] ^z2 ^z1 _z1 THEN ST z1 z2 _ ". ;
      &tabstrt _ &tabstrt + (tabstrt.H )/5 + 2;
      END;
   RETURN;
   END.
(setvec)PROCEDURE;                                                     10H
   %Sets entry vector for frontend%
   entvec[0] _ entvec .V 254B9; %'entvec' set before call 'setvec'%
   %entvec[1] _ $rentr .V 254B9; %%reenter nls%
   %entvec[2] _ $tnls .V 254B9; %%start TNLS%
   %entvec[5] _ $query .V 254B9; %%start query%
   % comment out for now
      entvec[0] _ $binit .V 254B9; start backend%
   %entvec[3] _ $dex .V 254B9;% %start DEX%
   %entvec[4] _ $jbackground .V 254B9;% %journal background process%
   %entvec[6] _ $net [1] .V 254B9;% %start from Network %
   !sevec( 4B5, 1B6 .V $entvec); %set entry vector%
   RETURN;
   END.
(init1)PROCEDURE;                                                      10I
   (nswnls);                                                          10I1
   % save registers set by FE in image: 0-2 %
      !MOVEM R0,stacs;
      !MOVEM R1, stac1;
      !MOVEM R2, stac2;
   %Restore the registers set at presave time from runtime in image
   registers 3-15 %
      !HRLI R15, stac3;
      !HRRI R15, 3;
      !BLT R15, R15;
   backstart();
   LOOP !haltf();
   END.
(gtintmsg) PROCEDURE;                                                  10J
   btypestr($"  Message = ");
   LOOP
      BEGIN
      !pbin();
      IF R1 = CA THEN EXIT;
      *intmsg* _ *intmsg*, R1;
      END;
   IF intmsg.L THEN
```

```
      BEGIN
      *intmsg* _ *intmsg*, 0;
      intmsf _ TRUE;
      END;
   RETURN(intmsg.L);
   END.

(getcr)   % CL: ;  get next character converting EOL to CR %
PROCEDURE;                                                          10K
   % Procedure description
      FUNCTION
         Read next character using pbin changing an EOL to a CR.  If
         it's a CR, read next character assuming it's an LF.  If
         it's not an LF, do a bkjfn to backup character in file
      ARGUMENTS
         none
      RESULTS
         Character read or CR if character was an EOL.
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %
   % Declarations %
      LOCAL char;
   %Read character%
      !pbin();
      char _ R1;
      CASE char OF
         = EOL:
            char _ CR;
         = CR:
            BEGIN  %read next character%
            !pbin();
            IF R1 NOT= LF THEN  %backup character if not linefeed%
               IF NOT SKIP !bkjfn(100) THEN NULL;
            END;
         ENDCASE;
   % Return %
      RETURN(char);
   END.

%.....Initialization support routines.....%
   (tiinit) PROCEDURE;  %init file handling mechanisms%             11A
      LOCAL
         fileno, count, fz, da, astrng, fl, fo, fp, end, hdr, sndptr,
         char;
      REF sndptr, fl, fo, fp, astrng, fz, da;
      %initialize file list%
         nxpg _ 1;  %for lodrfb%
         &fl _ $filst;
         end _ &fl + filmax*filstl;
         DO fl _ 0 UNTIL (&fl _ &fl + 1) = end;
         &fo _ $fopncnt;
         end _ &fo + filmax + 1;
         DO fo _ 0 UNTIL (&fo _ &fo + 1) = end;
```

```
   %Initialize filepart table%
      &fp _ $filepart;
      end _ &fp + filmax;
      DO fp _ 0 UNTIL (&fp_&fp+1)=end;
   %Initialize frozen list%
      &fz _ fzfree _ $fzlsts;
      end _ &fz + (frzmax-1)*frzl;
      DO
         fz.fznext _ &fz + frzl
      UNTIL (&fz _ &fz + frzl) = end;
      fz.fznext _ 0;
   RETURN
   END.

(checklib)PROC;                                                    11B
   %This procedure checks the system flags to see if it should try
   to start up one of the background processors automatically.
      If so, it tries to start it up.
      If successful, it returns the library number for the
      processor,
      .
      OtherWise, it may call err, or it may return 0 depending on
      the error%
   LOCAL libndx;
   IF nojournalflag THEN RETURN(0);
   libndx _ 0;
   IF libstrt(2) THEN RETURN(1); %will start as auto job if not
   logged in%
   %Make sure this was not a false start%
      !gjinf();
      IF R1 = 0 THEN
         killib(); %Not logged in--kill job%
   RETURN(0);
   END.

(libstrt)PROCEDURE(libno);                                         11C
   LOCAL libdirno, sysmsg;
   LOCAL STRING errstr[200];
   %This procedure tries to login and start a background processor.
   It may do a directory connect too depending on the processor%
   %Set up to catch signals%
      INVOKE( liberr );
   %see if job is logged in%
      IF (libdirno_!gjinf()) # 0 AND NOT autostrt THEN
      RETURN(FALSE);
   %Set "XXX" ident%
      *initsr* _ "XXX";
      cinit _ 30614000; %could use setcinit, but it is in the
      frontend%
      nwheelf _ 1;
   %Get dirno of user for login%
      !stdir(0, chbmty + $"BACKGROUND");
         GOTO libfal;
         GOTO libfal;
   %Handle "alrady logged in"%
      IF libdirno  # 0 THEN
```

```
        BEGIN
        RETURN(IF R1.RH = libdirno THEN TRUE ELSE FALSE);
        END
     ELSE libdirno _ R1;
  %enable login capability%
     enablLogin();
  %By here, dir exists, no in R1.rh..see if login ok%
     IF libdirno.b0 THEN
        (libfal):                                              11C10A1
           BEGIN
           specttyout(0, $"Illegal Library User");
           specttyout(oprtty, $"Illegal Library User");
           killib();
           END;
  %log him in%
     R1 _ R1.RH;
     R2 _ 4407B8 + (CASE lhostn OF
        =utilhost: $libsw1;
        ENDCASE $libsw2);
     IF NOT SKIP !login(R1, R2, 5B11 .V 1) THEN
        BEGIN
        *errstr* _ "Could Not Log In Background Job", CR, LF,
        STRING(R1);
        specttyout(0, $errstr);
        specttyout(oprtty, $errstr);
        killib();
        END;
  autostrt _ TRUE;
  DROP( liberr );
  RETURN(libno);
  (liberr) CATCHPHRASE(:sysmsg);                               11C15
     BEGIN
     IF NOT sysmsg THEN sysmsg _ $"Error";
     *errstr* _ "Library Start Fail: ", *[sysmsg]*;
     specttyout(0,$errstr);
     specttyout(oprtty, $errstr);
     killib();
     CONTINUE;
     END;
  END.

DECLARE libsw1=46237014283, libsw2=516070141B3;
(killib)PROC;                                                  11E
   !logout(-1); %log job out%
   END.

(enablLogin)PROC;                                              11F
   %Enable the Login capability%
   !rpcap(4B5);
   IF NOT R2.bit3 THEN ABORT(0, $"Log Not in Assigned
   Capabilities");
   R3.bit3 _ 1;
   !epcap();
   RETURN;
   END.
```

```
(gthstn)     % returns host number for input string text pointers %
                                                                    11G
   PROCEDURE
      (hp1,  % text pointer to start of host name %
      hp2    % text pointer to end of host name %
      );
   LOCAL
      lhostn,          % index for looping through HOSTN table %
      sindex,          % index into HSTNAM table ( gotten from HOSTN
      table) %
      jhstnm,          % index for getting asciz string from HSTNAM
      table %
      rhstn, % remote host number for an entry in HOSTN table %
      kindex % index for doing string word compares %
      ;
   LOCAL STRING
      rhoststr[45],  % input host name string %
      tstring[45]    % temp string for HSTNAM table asciz string %
      ;
   REF hp1, hp2;

   % ALGORITHM %
      % if the input text pointers point to a null string then the
      host number of the local host is reaturned.
      This procedure also sets up the following globals:
         hostni.LH - gets table number of HOSTN table for getab jsys
         hostni.RH - gets number of entries in HOSTN table
         hostsi.LH - gets table number of HSTNAM table for getab
         jsys
         hostsi.RH - gets number of entries in HSTNAM table
      %
   % RETURNS %
      % returns host number for input texp pointers (local host
      number if text pointers delimit a null string) or -1 for
      invalid host name input %

   % setup string for input text pointers %
      *rhoststr* _ + hp1 hp2;
   % if null input then return local host number %
      IF NOT rhoststr.L THEN RETURN( lhostn );
   % setup string for word compares later %
      *rhoststr* _ *rhoststr*, 0, 0, 0, 0, 0, 0;
      rhoststr.L _ rhoststr.L - 6;
   % setup hostni with HOSTN number and length %
      IF hostni < 0 THEN
         BEGIN
         !sysgt( R1 _ getsbn( $"HOSTN" ) );
         IF NOT R2 THEN err( $"sysgt jsys error" );
         hostni.LH _ R2.RH;  % HOSTN table number %
         !HLRE R2,R2;
         hostni.RH _ -R2;     % number of entries %
         END;
   % setup hostsi with HSTNAM number and length %
      IF hostsi < 0 THEN
         BEGIN
         !sysgt( R1 _ getsbn( $"HSTNAM" ) );
```

```
        IF NOT R2 THEN err( $"sysgt jsys error" );
        hostsi.LH _ R2.RH;  % HSTNAM table number %
        !HLRE R2,R2;
        hostsi.RH _ -R2;    % number of entries %
        END;
    % loop through HOSTN table looking for this host string %
     FOR ihostn _ hostni.RH DOWN UNTIL <= 0 DO
        BEGIN
        R1.LH _ ihostn - 1; % table starts at 0 %
        R1.RH _ hostni.LH;
        IF NOT SKIP !getab( R1 ) THEN err( $"getab jsys error" );
        % get index into HSTNAM and host # this entry %
           sindex _ R1.hstnmi;
           rhstn _ R1.hstnmn;
        % now get asciz host name string from HSTNAM %
           FOR jhstnm _ sindex UP UNTIL < 0 DO
              BEGIN
              R1.LH _ jhstnm;
              R1.RH _ hostsi.LH;
              IF NOT SKIP !getab( R1 ) THEN err( $"getab jsys
              error" );
              tstring[jhstnm - sindex + 1] _ R1;
              IF NOT ( R1 .A 377B ) THEN EXIT LOOP;
              END;
        % see if this is the desired host %
           FOR kindex _ 1 UP UNTIL > (jhstnm - sindex + 1) DO
              IF (rhoststr[kindex] .A 777777777776B) #
              tstring[kindex] THEN REPEAT LOOP 2;   % look at next
              entry %
        % if we're here we got the right host so return %
           RETURN( rhstn );
        END;
    % got here means we've exhausted HOSTN and not found a match %
     RETURN( -1 );

    END.

    % %
```

```
%measurement stuff%
   SET
      begloc=0, endloc=1, begins=2, endins=3,
      measmx=4, meascr=5, frstclk=6, totclk=7;
   SET ZERO = 0;
   (intmeas) PROCEDURE;  %measurement initialization%                    12C
      LOCAL curitm, jsrbins, jsreins, count;
      REF curitm;
      %wsd-meas%
         !JSYS jobtm;
         strttm _ R1;
         ldrnct _ ldrfct _ ldsdct _ fchsct _ 0;
      &curitm _ $mslst;
      jsrbins _ 264B9 .V $begmeas;
      jsreins _ 264B9 .V $endmeas;
      count _ 0;
      DO
         IF (curitm.msbegin _ msaddr[count]) THEN
            BEGIN
            IF NOT (curitm.msend _ msaddr[count _ count + 1]) THEN
               BEGIN
               dismes (2, $"Measurement begin address has no
               corresponding end address");
               EXIT LOOP;
               END;
            curitm.msbinstn _ [curitm.msbegin] := jsrbins;
            curitm.mseinstn _ [curitm.msend] := jsreins;
            BUMP curitm.msbegin, curitm.msend, count;
            curitm.msmax _ msaddr[count := count + 1];
            curitm.mscurr _ curitm.ms1clk _ curitm.mstclk _ 0;
            END
         ELSE count _ count + 3
      UNTIL (&curitm _ &curitm + msentl) >= $mslste;
      RETURN;
      %the following assembly code is executed when the measurement
      routines are called%
         %this routine records the clock time to begin measurement%
            (begmeas): !ZERO;                                          12C11A1
            !MOVEM R1,msr1sv;  %save R1, R2, R3%
            !MOVEM R2,msr2sv;
            !MOVEM R3,msr3sv;
            !MOVEI R1,mslst;  %index to possible measurement entry%
            (bmsfnd): !MOVE R2,begloc(R1);                             12C11A6
            !HRRZ R3,begmeas;
            !CAMN R2,R3;
            !JRST bmstim;  %this is the entry%
            !ADD R1,msentl;
            !CAIGE R1,mslste;
            !JRST bmsfnd;
            !MOVE R1,=255B9;  %entry not here -- fake return%
            !MOVEM R1,bmsrin;
            !JRST bmsrtn;
            (bmstim): !MOVEM R1,measnt;                               12C11A16
            !MOVE R1,=-5;
            !JSYS jobtm;
            !MOVE R3,measnt;
```

```
        !MOVEM R1,frstclk(R3);
        !MOVE R2,begins(R3);
        !MOVEM R2,bmsrin;
        (bmsrtn): !MOVE R1,begmeas;                                  12C11A23
        !HRRM R1,bmsrloc;
        !MOVE R1,msr1sv;
        !MOVE R2,msr2sv;
        !MOVE R3,msr3sv;
        (bmsrin): !ZERO;                                             12C11A28
        (bmsrloc): !JRST 0;                                          12C11A29
    %this routine records the clock time ard performs the
    necessary calculations at the end of the measurement interval;
    to read the measurement at the end of the specified number of
    passes, two breakpoints may be set
        at msmbrk: at this point R1 will contain the elapsed time
        in milliseconds;
        at mssbrk: at this point R1 will contain the elapsed time
        in seconds, and R2 the remainder, in milliseconds%
        (endmeas): !ZERO;                                            12C11B3
        !MOVEM R1,msr1sv;  %save R1, R2, R3%
        !MOVEM R2,msr2sv;
        !MOVEM R3,msr3sv;
        !MOVEI R1,mslst;  %index to possible measurement entry%
        (emsfnd): !MOVE R2,endloc(R1);                               12C11B8
        !HRRZ R3,endmeas;
        !CAMN R2,R3;
        !JRST emstim;  %this is the entry%
        !ADD R1,msentl;
        !CAIGE R1,mslste;
        !JRST emsfnd;
        !MOVE R1,=255B9;  %entry not here -- fake return%
        !MOVEM R1,emsrin;
        !JRST emsrtn;
        (emstim): !MOVEM R1,measnt;                                  12C11B18
        !MOVE R1,=-5;
        !JSYS jobtm;
        !MOVE R3,measnt;
        !SUB R1,frstclk(R3);
        !ADDM R1,totclk(R3);
        !AOS 1,measct(R3);
        !CAMGE R1,measmx(R3);
        !JRST emscon;
        !MOVE 1,totclk(R3);
        (msmbrk): !IDIVI R1,1000;                                    12C11B28
        (mssbrk): !SETZM measct(R3);                                 12C11B29
        !SETZM totclk(R3);
        (emscon): !MOVE R2,endins(R3);                               12C11B31
        !MOVEM R2,emsrin;
        (emsrtn): !MOVE R1,endmeas;                                  12C11B33
        !HRRM R1,emsrloc;
        !MOVE R1,msr1sv;
        !MOVE R2,msr2sv;
        !MOVE R3,msr3sv;
        (emsrin): !ZERO;                                             12C11B38
        (emsrloc): !JRST 0;                                          12C11B39
END.
```

```
(nlserr) PROCEDURE (errcod REF, errmsg REF) ;                          13
   %nls base error routine.  called from ABORT pprocessor catchphrase
   in middle end.  at present simply displays message. %
      IF &errcod = nomod THEN RETURN(TRUE) ELSE
         BEGIN
         %dismes (1, &errmsg) ; Removed 'cause f.e. also prints
         message%
         RETURN (FALSE) ;
         END ;
      END.
%.....Entry points.....%
   DECLARE submission=1;
FINISH



   %Code from initid%
      %PROTOCOL%
         % check for new journal mail%
            !gtfdb (fl.florig, 1000024B, $R3);
            IF R3.ojdelf THEN %new mail%
               BEGIN
               R2 _ 0; R2.ojdelf _ 1;
               chnfdb (fl.florig, 24B, R2, 0);
               *gs* _ "You have new Journal mail";
               % Have FE print out message %
               END;
      % do initial process commands %
         CASE stupstr.L OF
            > 0:
               BEGIN
               INVOKE( termsig, exitcase );
               FIND SF(*stupstr*) ^p1 SE(*stupstr*) ^p2;
               p3 _ origin;
               p3.stfile _ prwndw.dacsp.stfile;
               p3[1] _ 1;
               caddexp ($p1, $p2, $prwndw, $p3 );
               IF p3 = endfil THEN EXIT CASE;
               p4 _ getend( p3 );
               FIND SE(p4) ^p4;
               auxstartup( $p3, $p4 );
               END;
            ENDCASE;
         (exitcase): DROP( termsig );                                  15A2B
      (termsig) CATCHPHRASE;                                           15A3
         TERMINATE;
   (setdis) PROCEDURE; %set up display areas%                          15B
      %issue jsys's to allocate display area's for CFL, L-T area,
      viewspec area, name area,subsystem-name area, literal feedback
      area, and a message area.  Set up globals subda, cflda, ltvsda,
      vspcda, namda, litda, msgda, and assume dpyarea is set up for
      text area .%
      %------------------%
      LOCAL  end, da, ls, save, consolenum, entryptr;
```

```
   LOCAL STRING send[10], str[40];
   REF ls, da;
   cdtype _ dspno;
   %reset the display%
      dassnbit($dabt, -1, dabtsize);  %deassign all daid's%
   %text area%
      &da _ $dpyarea;
      UNTIL &da >= $dpyend DO
         BEGIN
         IF da.daexis AND NOT da.dasuppress THEN
            <NLS, DSPGEN, alocda>(&da);
         &da _ &da + dal;
         END;
   RETURN;
   END.

(dex) PRLCEDURE; %start DEX%                                          15C
   !JSP R1,110init;
   initback($dxin);
   LOOP !haltf();
   END.

(dxin)PROCEDURE; % Roll in dxctl file %                               15D
   LOCAL proc, result[10]; % Address of DXSTRT procedure in the user
   program. %
   LOCAL TEXT POINTER tp;
   LOCAL STRING dxname[20];
   REF proc;
   % Roll in DEX user program. %
      % Increase buffer size to 30 pages. %
         IF NOT gpbsz( 30 ) THEN
            BEGIN
            btypestr($"Cannot set program buffer for DEX system!");
            LOOP !haltf();
            END;
      % Reset the user program buffer. %
         gpgmrst();
      % Load the user program with DEX code. %
         *dxname* _ "rel-nls, dxctl";
         INVOKE( nodex );
         getuprog($dxname);
         DROP( nodex );
      % Get address of DXSTRT %
         IF NOT ddtlookup($"DXSTRT", FALSE, % get procedure in user
         program. %: &proc) THEN
            BEGIN
            btypestr($"Cannot find entry procedure");
            LOOP !haltf();
            END;
   % Run DEX. %
      proc();
   btypestr($"Completed.");
   LOOP !haltf();
   (nodex) CATCHPHRASE;                                               15D9
      BEGIN
      btypestr($"Cannot load DEX program");
```

```
      LOOP !haltf();
      CONTINUE;
      END;
   END.

(quin)PROCEDURE(nlsparse); % Roll in query file %                    15E
   LOCAL proc, result[10]; % Address of QPORT procedure in the user
   program. %
   LOCAL TEXT POINTER tp;
   LOCAL STRING quname[20];
   REF proc;
   % Roll in QUERY user program. %
      % Reset the user program buffer. %
         gpgmrst();
      % Load the user program with QUERY code. %
         *quname* _ "programs, querysys";
         INVOKE( noquery );
         getuprog( $quname );
         DROP( noquery );
      % Get address of QPORT %
         IF NOT ddtlookup($"QPORT", FALSE, % get procedure in user
         program. %: &proc) THEN
            BEGIN
            btypestr($"Cannot find entry procedure");
            LOOP !haltf();
            END;
   % Run QUERY. %
      proc(nlsparse);
   btypestr($"Completed.");
   LOOP !haltf();
   (noquery) CATCHPHRASE;                                           15E9
      BEGIN
      btypestr($"Cannot load QUERY program");
      LOOP !haltf();
      CONTINUE;
      END;
   END.

(jbackground) PROCEDURE; %start Journal background process%         15F
   !JSP R1,l10init;
   initback($jnlin);
   LOOP !haltf();
   END.

(jnlin)PROCEDURE; % Roll in JNLDEL file %                            15G
   LOCAL proc, result[10]; % Address of JOUTIL procedure in the user
   program. %
   LOCAL TEXT POINTER tp;
   LOCAL STRING jname[20];
   REF proc;
   % Initialize. %
      jnlprog _ FALSE;
      %nddtarm();% %arm ^H to nddt in case we want it%
   % Roll in Journal user program. %
      % Increase buffer size to 30 pages. %
         IF NOT gpbsz( 30 ) THEN
```

```
            BEGIN
            btypestr($"Cannot set program buffer for JOurnal
            system!");
            LOOP !haltf();
            END;
        % Reset the user program buffer. %
          gpgmrst();
        % Load the user program with Journal delivery/utility code. %
          *jname* _ "netsys, jnldel";
          INVOKE( nojnldel );
          getuprog($jname); % loads user program, does  not put out
          messages. %
          DROP( nojnldel );
        % Set flag indicating that user program has been rolled in. %
          jnlprog _ TRUE;
        % Get address of joutil %
          IF NOT ddtlookup($"JOUTIL", FALSE, % get procedure in user
          program. %: &proc) THEN
            BEGIN
            btypestr($"No JOUTIL procedure:  error in DDT lookup");
            LOOP !haltf();
            END;
    IF libflg THEN % run journal delivery %
       BEGIN
       proc( libflg _ libflg .V oldflg, jdid % list of idents. %);
       btypestr($"Completed.");
       END
    ELSE btypestr($"No library functions set");
    RETURN;
    (nojnldel) CATCHPHRASE;                                    15G10
       BEGIN
       btypestr($"Cannot load jnldel user program");
       LOOP !haltf();
       CONTINUE;
       END;
    END.

(getuprog) PROCEDURE                                           15H
    %FORMAL PARAMETERS%
       (stptr);       %ptr to a string containg program name (or a
       directory name followed by a comma follwed by the file name.)%
                                                               15H1A
    % This procedure calls gpget with a flag saying not to put out
    program loading messages except for errors. %
    LOCAL
       adstr[40];
    LOCAL TEXT POINTER tpt1;
    LOCAL STRING locstr[50];
    REF stptr;
    % set up local string and text pointer%
       *locstr* _ "< , *stptr* , ",>";
       FIND SF(*locstr*) ^tpt1;
       lnkprs($tpt1,$adstr);
       RETURN(gpget($adstr, FALSE % don't print out messages %));
    END.
(net) PROCEDURE; % Start NLS from the Net %                     15I
```

```
    R1.LH _ 7; % first AC to save is 7 %
    R1.RH _ $xacs [7]; % that's where it's to be saved %
    R2 _ $xacs [15]; % AC 17 is last one to be saved %
    !BLT 1,(2); % save ACs 7-17 %
    !JSP R1,110init;
    initback($netdsp);
    LOOP !haltf();
    END.

(netdsp) PROCEDURE; % Dispatch to HIOP or NJS %                    15J
    % Declarations %
        LOCAL reason;   % reason for dispatch %
    % Save dispatch reason %
        R1 _ xacs [7]; % get xwd dispatch-reason, xxx %
        ! HLRZS 1; % clean dispatch reason %
        reason _ R1; % save it %
        IF reason = submission THEN njs($xacs) ELSE hiop($xacs);
        END.

%... old version of procedure ...%
(setinterrupts)PROC;                                               16A
    %set up rubout, command delete, and page write interrupt%
        %IF YOU CHANGE THIS CODE, SEE ALSO (utilty, trapcc) AND
        (utilty, notrapcc)%
        %chntab[1] _ $stopline .V 1B6; %% ^P %
        %note: the ^C dispatch is altered/restored by the journal%
        chntab[2] _ $ctrlc .V 1B6; %handle ^C to do imlac stuff%
        %chntab[3] _ $rubout .V 1B6; %% treat ^U like a rubout %
        %chntab[4] _ $msgpsi .V 1B6;% % for message fork to use %
        chntab[9] _ $sysovr .V 1B6;
        chntab[11] _ $iodaterr .V 1B6;
        chrtab[15] _ $ilspsi .V 1B6;  % illegal instruction %
        chntab[17] _ $wrtpsi .V 2B6;
        %chntab[24] _ $gotohelp .V 1B6;%
        R1 _ 4B5;
        R2 _ $chntab;
        !HRLI R2, levtab;
        !JSYS sir;
    % special stuff for control-c (^C)%
        R1 _ 4B5;
        !JSYS rpcap;
        R3 _ R3 .V 4B11;
        !JSYS epcap; %Permits process to assign ^C for pseudo
        interrupt%
    %enable interrupts%
        R1 _ 4B5;
        !JSYS eir;
    %activate io data error and write pseudo interrupt%
        R1 _ 4B5;
        R2 _ 10586; %channel 11, 15, 17%
        !JSYS aic;
    %activate rubout, etc.%
        %^O and ^P used by frontend
        R1 _ 20000001B; %%^P%%
        !JSYS ati;
        R1 _ 170000036; %%^O%%
```

```
      !JSYS ati;
      %
      %R1 _ 21000030B;% %^Q%
      %!JSYS ati;%
      IF nldevice = imlac0 OR nldevice = imlac1 OR ((tenex >= 13200)
      AND (nldevice = devlproc)) THEN
         BEGIN
         R1 _ 3000002B; %^C%
         !JSYS ati;
         %R2 _ 340000004000B;% %activate rubout, ^P, ^Q, ^C, ^O%
         R2 _ 340000000000B;% activate ^p, ^c, ^o %
         END
      ELSE % R2 _ 240000004000B;% %activate ^P, ^Q and ^O%
         R2 _ 240000000000B;% activate ^p, ^c %
      R1 _ 4B5;
      !JSYS aic;
   RETURN;
   END.

(idfrmuser) PROCEDURE (str);  %collect ident from user in tty mode%
   LOCAL char;
   REF str;
   *str* _ "XXX"; % Essentially a dummy for now;  this is a FE
   function %
   RETURN;
   END.
```

(acctyp)	<nine, brecords, 0346>	FIELD - 3	5B3
(b0)	<nine, brecords, 0302>	FIELD - 1	5T5
(bit1)	<nine, brecords, 0301>	FIELD - 1	5T4
(bit2)	<nine, brecords, 0300>	FIELD - 1	5T3
(bit3)	<nine, brecords, 0299>	FIELD - 1	5T2
(bitfiller)	<nine, brecords, 0298>	FIELD - 32	5T1
(blnk)	<nine, brecords, 0327>	FIELD - 1	5K5
(canxt)	<nine, brecords, 0151>	FIELD - 18	5A5
(carbp)	<nine, brecords, 0150>	FIELD - 36	5A4
(card)	<nine, brecords, 0146>	RECORD	5A
(carnc)	<nine, brecords, 0148>	FIELD - 18	5A2
(catbp)	<nine, brecords, 0149>	FIELD - 36	5A3
(catnc)	<nine, brecords, 0147>	FIELD - 18	5A1
(corcpy)	<nine, brecords, 0165>	FIELD - 36	5B3
(cords)	<nine, brecords, 0108>	RECORD	3A
(corlck)	<nine, brecords, 0163>	FIELD - 36	5B1
(corpag)	<nine, brecords, 0162>	RECORD	5B
(corpgr)	<nine, brecords, 0166>	RECORD	5C
(cortim)	<nine, brecords, 0164>	FIELD - 36	5B2
(ctfile)	<nine, brecords, 0168>	FIELD - 5	5C2
(ctfroz)	<nine, brecords, 0170>	FIELD - 3	5C4
(ctfull)	<nine, brecords, 0167>	FIELD - 1	5C1
(ctpnum)	<nine, brecords, 0169>	FIELD - 9	5C3
(daidr)	<nine, brecords, 0111>	RECORD	3B
(daidr1)	<nine, brecords, 0113>	FIELD - 6	3B2
(daidr2)	<nine, brecords, 0112>	FIELD - 6	3B1
(delhc)	<nine, brecords, 0183>	FIELD - 1	5D3
(deliverymodes)	<nine, brecords, 0180>	RECORD	5D
(delnet)	<nine, brecords, 0182>	FIELD - 1	5D2
(delol)	<nine, brecords, 0181>	FIELD - 1	5D1
(dig)	<nine, brecords, 0333>	FIELD - 1	5K11
(dlfbal)	<nine, brecords, 052>	FIELD - 18	2I5
(dlfbap)	<nine, brecords, 057>	FIELD - 18	2I10
(dlfbcp)	<nine, brecords, 056>	FIELD - 18	2I9
(dlfbff)	<nine, brecords, 053>	FIELD - 18	2I6
(dlfbfl)	<nine, brecords, 055>	FIELD - 18	2I8
(dlfbgk)	<nine, brecords, 050>	FIELD - 36	2I3
(dlfbl)	<nine, brecords, 059>	EXT	2I10A
(dlfbnb)	<nine, brecords, 048>	FIELD - 18	2I1
(dlfbpb)	<nine, brecords, 049>	FIELD - 18	2I2
(dlfbpf)	<nine, brecords, 054>	FIELD - 18	2I7
(dlfbsk)	<nine, brecords, 051>	FIELD - 36	2I4
(dlfbst)	<nine, brecords, 061>	RECORD	2J
(dlflbk)	<nine, brecords, 047>	RECORD	2I
(dlgbgk)	<nine, brecords, 071>	FIELD - 36	2K5
(dlgbl)	<nine, brecords, 073>	EXT	2K5A
(dlgbnb)	<nine, brecords, 067>	FIELD - 18	2K1
(dlgbnu)	<nine, brecords, 070>	FIELD - 18	2K4
(dlgbpb)	<nine, brecords, 068>	FIELD - 18	2K2
(dlgbsc)	<nine, brecords, 069>	FIELD - 18	2K3
(dlgrbk)	<nine, brecords, 066>	RECORD	2K
(dlmbal)	<nine, brecords, 078>	FIELD - 11	2L2
(dlmbfl)	<nine, brecords, 081>	FIELD - 12	2L5
(dlmbfp)	<nine, brecords, 084>	FIELD - 18	2L8
(dlmbtr)	<nine, brecords, 079>	FIELD - 1	2L3
(dlmbl)	<nine, brecords, 086>	EXT	2L8A

(dimblk)	<nine, brecords, 075>	RECORD	2L
(dimbln)	<nine, brecords, 077>	FIELD - 11	2L1
(dimbnm)	<nine, brecords, 083>	FIELD - 18	2L7
(dimbpr)	<nine, brecords, 080>	FIELD - 1	2L4
(dimbzn)	<nine, brecords, 082>	FIELD - 18	2L6
(dlstlg)	<nine, brecords, 062>	FIELD - 1	2J1
(dlstnl)	<nine, brecords, 064>	FIELD - 1	2J3
(dlstpc)	<nine, brecords, 063>	FIELD - 1	2J2
(doi)	<nine, brecords, 0332>	FIELD - 1	5K10
(dpt)	<nine, brecords, 0331>	FIELD - 1	5K9
(dstar)	<nine, brecords, 0235>	FIELD - 1	5J13
(dvstar)	<nine, brecords, 0237>	FIELD - 1	5J15
(estar)	<nine, brecords, 0233>	FIELD - 1	5J11
(exp2)	<nine, brecords, 0330>	FIELD - 2	5K8
(exsign)	<nine, brecords, 0329>	FIELD - 2	5K7
(fbind)	<nine, brecords, 0188>	FIELD - 9	5F2
(fbnull)	<nine, brecords, 0187>	FIELD - 36	5F1
(fbpnum)	<nine, brecords, 0189>	FIELD - 9	5F3
(fbtype)	<nine, brecords, 0190>	FIELD - 5	5F4
(fdbaar)	<nine, brecords, 033>	FIELD - 1	2F2
(fdbact)	<nine, brecords, 094>	EXT ADDRESS =10B	2M6
(fdbadl)	<nine, brecords, 034>	FIELD - 1	2F3
(fdbarc)	<nine, brecords, 038>	FIELD - 1	2F7
(fdbart)	<nine, brecords, 031>	RECORD	2F
(fdbarr)	<nine, brecords, 025>	RECORD	2D
(fdbart)	<nine, brecords, 0102>	EXT ADDRESS =20B	2M14
(fdbbfr)	<nine, brecords, 028>	RECORD	2E
(fdbbkf)	<nine, brecords, 0101>	EXT ADDRESS =17B	2M13
(fdbbyr)	<nine, brecords, 042>	RECORD	2H
(fdbbyv)	<nine, brecords, 095>	EXT ADDRESS =11B	2M7
(fdbcnr)	<nine, brecords, 039>	RECORD	2G
(fdbcnt)	<nine, brecords, 0100>	EXT ADDRESS =16B	2M12
(fdbcre)	<nine, brecords, 091>	EXT ADDRESS =5B	2M3
(fdbcrv)	<nine, brecords, 097>	EXT ADDRESS =13B	2M9
(fdbctl)	<nine, brecords, 089>	EXT ADDRESS =1B	2M1
(fdbctr)	<nine, brecords, 015>	RECORD	2C
(fdbctu)	<nine, brecords, 016>	FIELD - 18	2C1
(fdbdel)	<nine, brecords, 021>	FIELD - 1	2C6
(fdbdmp)	<nine, brecords, 036>	FIELD - 1	2F5
(fdbeph)	<nine, brecords, 017>	FIELD - 1	2C2
(fdblng)	<nine, brecords, 019>	FIELD - 1	2C4
(fdbmrk)	<nine, brecords, 035>	FIELD - 1	2F4
(fdbnar)	<nine, brecords, 037>	FIELD - 1	2F6
(fdbnex)	<nine, brecords, 022>	FIELD - 1	2C7
(fdbnun)	<nine, brecords, 018>	FIELD - 11	2C3
(fdbnxf)	<nine, brecords, 020>	FIELD - 1	2C5
(fdbprb)	<nine, brecords, 08>	RECORD	2B
(fdbprm)	<nine, brecords, 023>	FIELD - 1	2C8
(fdbprr)	<nine, brecords, 04>	RECORD	2A
(fdbprt)	<nine, brecords, 090>	EXT ADDRESS =4B	2M2
(fdbref)	<nine, brecords, 099>	EXT ADDRESS =15B	2M11
(fdbsiz)	<nine, brecords, 096>	EXT ADDRESS =12B	2M8
(fdbtdm)	<nine, brecords, 0103>	EXT ADDRESS =21B	2M15
(fdbtfa)	<nine, brecords, 0104>	EXT ADDRESS =22B	2M16
(fdbtmp)	<nine, brecords, 024>	FIELD - 1	2C9
(fdbtsa)	<nine, brecords, 0105>	EXT ADDRESS =23B	2M17

(fdbunu)	<nine, brecords, 032>	FIELD - 11	2F1
(fdbuse)	<nine, brecords, 092>	EXT ADDRESS =6B	2M4
(fdbusw)	<nine, brecords, 0106>	EXT ADDRESS =24B	2M18
(fdbver)	<nine, brecords, 093>	EXT ADDRESS =7B	2M5
(fdbwrt)	<nine, brecords, 098>	EXT ADDRESS =14B	2M10
(fdd)	<nine, brecords, 0123>	FIELD - 1	3C6
(fds)	<nine, brecords, 0124>	FIELD - 1	3C7
(fhflgs)	<nine, brecords, 0184>	RECORD	5E
(fhunused)	<nine, brecords, 0342>	FIELD - 35	5E2
(fileblockheader)	<nine, brecords, 0186>	RECORD	5F
(filmax)	<nine, brecords, 0344>	EXT	5G15B
(filst1)	<nine, brecords, 0211>	EXT	5G15A
(filstr)	<nine, brecords, 0196>	RECORD	5G
(flaccm)	<nine, brecords, 0202>	FIELD - 8	5G6
(flart1)	<nine, brecords, 030>	FIELD - 18	2F2
(flastr)	<nine, brecords, 0208>	FIELD - 18	5G12
(flbpart)	<nine, brecords, 0206>	FIELD - 18	5G10
(flbpcst)	<nine, brecords, 0210>	FIELD - 18	5G14
(flbrws)	<nine, brecords, 0199>	FIELD - 1	5G3
(flbyts)	<nine, brecords, 045>	FIELD - 6	2H3
(fld1)	<nine, brecords, 0325>	FIELD - 6	5K3
(fld2)	<nine, brecords, 0324>	FIELD - 6	5K2
(fld3)	<nine, brecords, 0323>	FIELD - 6	5K1
(fldfr)	<nine, brecords, 046>	FIELD - 6	2H4
(fldirno)	<nine, brecords, 0203>	FIELD - 12	5G7
(fldmpt)	<nine, brecords, 029>	FIELD - 18	2E1
(flexis)	<nine, brecords, 0197>	FIELD - 1	5G1
(flfill)	<nine, brecords, 044>	FIELD - 6	2H2
(flfsat)	<nine, brecords, 027>	FIELD - 18	2D2
(flhead)	<nine, brecords, 0198>	FIELD - 9	5G2
(fllock)	<nine, brecords, 0200>	FIELD - 1	5G4
(flnmrd)	<nine, brecords, 040>	FIELD - 18	2G1
(flnmwr)	<nine, brecords, 041>	FIELD - 18	2G2
(flnocios)	<nine, brecords, 0204>	FIELD - 1	5G8
(flnsw)	<nine, brecords, 0319>	FIELD - 18	5G15
(florig)	<nine, brecords, 0207>	FIELD - 18	5G11
(flpart)	<nine, brecords, 0205>	FIELD - 18	5G9
(flpcread)	<nine, brecords, 0201>	FIELD - 1	5G5
(flpcst)	<nine, brecords, 0209>	FIELD - 18	5G13
(flpgsz)	<nine, brecords, 043>	FIELD - 18	2H1
(flprap)	<nine, brecords, 011>	FIELD - 1	2B3
(flprex)	<nine, brecords, 012>	FIELD - 1	2B4
(flprgr)	<nine, brecords, 06>	FIELD - 6	2A2
(flpri1)	<nine, brecords, 010>	FIELD - 1	2B2
(flprnu)	<nine, brecords, 09>	FIELD - 1	2B1
(flprpu)	<nine, brecords, 05>	FIELD - 6	2A1
(flprre)	<nine, brecords, 014>	FIELD - 1	2B6
(flprse)	<nine, brecords, 07>	FIELD - 6	2A3
(flprwr)	<nine, brecords, 013>	FIELD - 1	2B5
(flsnat)	<nine, brecords, 026>	FIELD - 18	2D1
(frmatr)	<nine, brecords, 0115>	RECORD	3C
(frozen)	<nine, brecords, 0127>	RECORD	3D
(frzl)	<nine, brecords, 0129>	EXT	3D5A
(fstar)	<nine, brecords, 0234>	FIELD - 1	5J12
(fzexis)	<nine, brecords, 0340>	FIELD - 1	3D5
(fznext)	<nine, brecords, 0339>	FIELD - 18	3D4

(ojtitt)	<nine, brecords, 0254>	FIELD - 12	5N1
(opdisflg)	<nine, brecords, 0384>	FIELD - 1	5O4
(opflgs)	<nine, brecords, 0258>	RECORD	5O
(opform)	<nine, brecords, 0259>	FIELD - 1	5O1
(opsimtf)	<nine, brecords, 0260>	FIELD - 1	5O2
(opwtpb)	<nine, brecords, 0385>	FIELD - 1	5O3
(prvsts)	<nine, brecords, 0185>	FIELD - 1	5E1
(pvsrecord)	<nine, brecords, 0139>	RECORD	4A
(rfcore)	<nine, brecords, 0268>	FIELD - 9	5P6
(rfexis)	<nine, brecords, 0263>	FIELD - 1	5P1
(rffree)	<nine, brecords, 0267>	FIELD - 10	5P5
(rfnull)	<nine, brecords, 0265>	FIELD - 2	5P3
(rfpart)	<nine, brecords, 0264>	FIELD - 1	5P2
(rfstr)	<nine, brecords, 0262>	RECORD	5P
(rfused)	<nine, brecords, 0266>	FIELD - 10	5P4
(round)	<nine, brecords, 0326>	FIELD - 5	5K4
(sbas)	<nine, brecords, 0292>	FIELD - 36	5R3
(sbc1cnt)	<nine, brecords, 0296>	FIELD - 12	5S1
(sbc2cnt)	<nine, brecords, 0356>	FIELD - 12	5S2
(sbc3cnt)	<nine, brecords, 0357>	FIELD - 12	5S3
(sbdp)	<nine, brecords, 0291>	FIELD - 36	5R2
(sbrp)	<nine, brecords, 0290>	FIELD - 36	5R1
(sbtt)	<nine, brecords, 0293>	FIELD - 36	5R4
(sdd)	<nine, brecords, 0117>	FIELD - 1	3C2
(sds)	<nine, brecords, 0118>	FIELD - 1	3C3
(seqr)	<nine, brecords, 0358>	RECORD	5Q
(shed)	<nine, brecords, 0289>	RECORD	5R
(sign)	<nine, brecords, 0335>	FIELD - 2	5K13
(sqwrkl)	<nine, brecords, 0381>	EXT ADDRESS =seqr.SIZE	5Q22
(substr)	<nine, brecords, 0295>	RECORD	5S
(swalloc)	<nine, brecords, 0373>	FIELD - 1	5Q14
(swaport)	<nine, brecords, 0366>	FIELD - 36	5Q8
(swbport)	<nine, brecords, 0367>	FIELD - 36	5Q9
(swcacode)	<nine, brecords, 0363>	FIELD - 18	5Q5
(swclvl)	<nine, brecords, 0371>	FIELD - 6	5Q12
(swcstid)	<nine, brecords, 0359>	FIELD - 36	5Q1
(swflg)	<nine, brecords, 0374>	FIELD - 1	5Q15
(swfill)	<nine, brecords, 0377>	FIELD - 1	5Q18
(swkflg)	<nine, brecords, 0372>	FIELD - 1	5Q13
(swlbstid)	<nine, brecords, 0360>	FIELD - 36	5Q2
(swslvl)	<nine, brecords, 0370>	FIELD - 6	5Q11
(swstid)	<nine, brecords, 0361>	FIELD - 36	5Q3
(swstkalloc)	<nine, brecords, 0376>	FIELD - 1	5Q17
(swstkdec)	<nine, brecords, 0375>	FIELD - 1	5Q16
(swstkloc)	<nine, brecords, 0378>	FIELD - 18	5Q19
(swstksz)	<nine, brecords, 0379>	FIELD - 18	5Q20
(swsvw)	<nine, brecords, 0369>	FIELD - 18	5Q10
(swusqcod)	<nine, brecords, 0362>	FIELD - 18	5Q4
(swvsp2)	<nine, brecords, 0365>	FIELD - 36	5Q7
(swvSpec)	<nine, brecords, 0364>	FIELD - 36	5Q6
(swword)	<nine, brecords, 0380>	FIELD - 36	5Q21
(tenexbits)	<nine, brecords, 0297>	RECORD	5T
(usrblk)	<nine, brecords, 0303>	RECORD	5U
(usrtlgs)	<nine, brecords, 0307>	FIELD - 18	5U3
(usrgrp)	<nine, brecords, 0308>	FIELD - 36	5U4
(usrjob)	<nine, brecords, 0309>	FIELD - 36	5U5

(usrlnk)	<nine, brecords, 0304>	FIELD - 36	5U1
(usrno)	<nine, brecords, 0306>	FIELD - 18	5U2
(usrrt)	<nine, brecords, 0312>	FIELD - 36	5U8
(usrtim)	<nine, brecords, 0313>	FIELD - 36	5U9
(usrtod)	<nine, brecords, 0311>	FIELD - 36	5U7
(usrttv)	<nine, brecords, 0310>	FIELD - 36	5U6
(ustar)	<nine, brecords, 0236>	FIELD - 1	5J14
(vs1)	<nine, brecords, 0140>	FIELD - 36	4A1
(vs2)	<nine, brecords, 0141>	FIELD - 36	4A2
(vscacode)	<nine, brecords, 0142>	FIELD - 18	4A3
(vstar)	<nine, brecords, 0232>	FIELD - 1	5J10
(vsusgcod)	<nine, brecords, 0143>	FIELD - 18	4A4
(xc)	<nine, brecords, 0130>	RECORD	3E
(xc1)	<nine, brecords, 0132>	FIELD - 6	3E2
(xc2)	<nine, brecords, 0131>	FIELD - 6	3E1
(xcord)	<nine, brecords, 0109>	FIELD - 18	3A1
(xyds)	<nine, brecords, 0116>	FIELD - 1	3C1
(yc)	<nine, brecords, 0134>	RECORD	3F
(yc1)	<nine, brecords, 0136>	FIELD - 6	3F2
(yc2)	<nine, brecords, 0135>	FIELD - 6	3F1
(ycord)	<nine, brecords, 0110>	FIELD - 18	3A2

```
< NINE, BRECORDS.NLS;4, >, 19-Apr-78 13:39 HGL ;;;;
FILE brecords % (arcsubsys,xl10,) (RELNINE,brecords.rel,) %
% RECORDS FOR USE WITH THE DIRECTORY COMMANDS %
   (fdbprr) RECORD  % description of fdbprt word in fdb %                2A
      flprpu  [6],  %    public protection bits %
      flprgr  [6],  %    group protection bits %
      flprse  [6];  %    self protection bits %

   (fdbprb) RECORD  % description of protection field bits %             2B
      flprnu  [1],  %    not used %
      flprli  [1],  %    list protection %
      flprap  [1],  %    append access %
      flprex  [1],  %    execute access %
      flprwr  [1],  %    write access %
      flprre  [1];  %    read access %

   (fdbctr) RECORD  % description of fdbctl word in fdb %               2C
      fdbctu [18],  %    unused bits %
      fdbeph  [1],  %    file is ephemeral %
      fdbnun [11],  %    more unused bits %
      fdblng  [1],  %    this is a long file %
      fdbnxf  [1],  %    file doesn't exist yet - unused %
      fdbdel  [1],  %    file is deleted %
      fdbnex  [1],  %    no ext for this fdb yet - unused %
      fdbprm  [1],  %    file is permanent %
      fdbtmp  [1];  %    file is temporary %

   (fdbarr) RECORD  % description of fdbart word in fdb %               2D
      flsnat [18],  %    second archive tape number %
      flfsat [18];  %    first archive tape number %

   (fdbbfr) RECORD  % description of fdbbkf word in fdb %               2E
      fldmpt [18],  %    most recent dump tape number %
      flarfl [18];  %    archive flags (see fdbarf record) %

   (fdbarf) RECORD  % description of archive flags field %             2F
      fdbunu [11],  %    unused bits %
      fdbaar  [1],  %    file already archived %
      fdbadl  [1],  %    don't delete file after archiving %
      fdbmrk  [1],  %    archived but not marked complete - unused %
      fdbdmp  [1],  %    dumped but not marked complete - unused %
      fdbnar  [1],  %    don't archive this file %
      fdbarc  [1];  %    archive pending %

   (fdbcnr) RECORD  % description of fdbcnt word in fdb %               2G
      flnmrd [18],  %    number of times this files read %
      flnmwr [18];  %    number of times this file written %

   (fdbbyr) RECORD  % bits within fdbbyv word in fdb %                  2H
      flpgsz [18],  %    size of file in pages %
      flfill  [6],  %    unused %
      flbyts  [6],  %    byte size %
      flgtr   [6];  %    default no. of versions to keep %

   (dlflbk) RECORD  % directory list file block %                       2I
      dlfbnb [18],  %    pointer to next file block this group %
```

```
    dlfbpb [18],  %    pointer to previous file block this group %
    dlfbgk [36],  %    group key for this file %
    dlfbsk [36],  %    sort key for this file %
    dlfbal [18],  %    astr - link for this file %
    dlfbff [18],  %    address of file fdb %
    dlfbpf [18],  %    address of pc fdb %
    dlfbfl [18],  %    status bits this entry (see dlfbst RECORD) %
    dlfbcp [18],  %    chain pointer for second pass %
    dlfbap [18];  %    astr - name of this file or its pcname%

       (dlfbl) EXTERNAL = 6;
                                                                    2I10A
(dlfbst) RECORD  % definition of dlfbfl in dlflbk RECORD %            2J
    dlstig  [1],  %    this entry redundant - ignore it %
    dlstpc  [1],  %    this file is a partial copy %
    dlstnl  [1];  %    this is an NLS file for which a pc exists %

(dlgrbk) RECORD  % directory list grouping block %                    2K
    dlgbnb [18],  %    pointer to next block %
    dlgbpb [18],  %    pointer to previous block %
    dlgbsc [18],  %    pointer to start of data chain this group %
    dlgbnu [18],  %    not used %
    dlgbgk [36];  %    group key for this group %

       (dlgbl) EXTERNAL = dlgrbk.SIZE;
                                                                   2K5A
(dlmblk) RECORD  % directory list - master blocks %                   2L
    dlmbln [11],  %    length of block requested%
    dlmbal [11],  %    actual length of block%
    dlmbfr  [1],  %    true if block is on a free list%
    dlmbpr  [1],  %    true if previous block is on a free list%
    dlmbfl [12],  %    filler %
       % these first 5 fields must parallel the record blkptr %
    dlmbzn [18],  %    zone to which this master block belongs %
    dlmbnm [18],  %    pointer to next master block %
    dlmbfp [18];  %    free storage pointer %

       (dlmbl) EXTERNAL = dlmblk.SIZE;
                                                                   2L8A
% offsets within an fdb %
   (fdbctl) EXTERNAL ADDRESS =  1B;  % ctrl word  (see fdbctr rec) %
                                                                    2M1
   (fdbprt) EXTERNAL ADDRESS =  4B;  % prot word (see fdbprr rec) %
                                                                    2M2
   (fdbcre) EXTERNAL ADDRESS =  5B;  % crt time & date of orig ver %
                                                                    2M3
   (fdbuse) EXTERNAL ADDRESS =  6B;  % LH = last write dir. # %     2M4
   (fdbver) EXTERNAL ADDRESS =  7B;  % LH is version number %       2M5
   (fdbact) EXTERNAL ADDRESS = 10B;  % accourt information %        2M6
   (fdbbyv) EXTERNAL ADDRESS = 11B;  % see fdbbyr record %          2M7
   (fdbsiz) EXTERNAL ADDRESS = 12B;  % byte count to end of file %
                                                                    2M8
   (fdbcrv) EXTERNAL ADDRESS = 13B;  % crt time & date this ver % 2M9
   (fdbwrt) EXTERNAL ADDRESS = 14B;  % time & date of last write %
                                                                   2M10
   (fdbref) EXTERNAL ADDRESS = 15B;  % time & date of last read %
```

```
                                                                    2M11
    (fdbcnt) EXTERNAL ADDRESS = 16B;  % see fdbcnr record %         2M12
    (fdbbkf) EXTERNAL ADDRESS = 17B;  % see fdbbfr record %         2M13
    (fdbart) EXTERNAL ADDRESS = 20B;  % see fdbarr record %         2M14
    (fdbtdm) EXTERNAL ADDRESS = 21B;  % time & date of last dump %
                                                                    2M15
    (fdbtfa) EXTERNAL ADDRESS = 22B;  % time & date of first archive
    %                                                               2M16
    (fdbtsa) EXTERNAL ADDRESS = 23B;  % time & date of secnd archive
    %                                                               2M17
    (fdbusw) EXTERNAL ADDRESS = 24B;  % user settable word %
                                                                    2M18
% RECORDS FOR FORMATTING DISPLAYS %
  (cords)  RECORD  % cursor word coordinates %                       3A
    xcord  [18],  %    x-coordinate %
    ycord  [18];  %    y-coordinate %

  (daidr)  RECORD  % daid record for remote displays %              3B
    daidr2  [6],  %    low-order 5 bits %
    daidr1  [6];  %    high-order 5 bits %

  (irmatr) RECORD % format byte in STRDA command to remote displays %
                                                                    3C
    xyds    [1],  %    TRUE: use old coordinates %
      % FALSE: read coordinates from command string %
    sdd     [1],  %    TRUE: use default da character size %
    sds     [1],  %    TRUE: use old string character size %
      % sdd=FALSE AND sds=FALSE: read character size from command
      string %
    idd     [1],  %    TRUE: use default da horizontal increment %
    ids     [1],  %    TRUE: use old string horizontal increment %
      % idd=FALSE AND ids=FALSE: read horizontal increment from
      command string %
    fdd     [1],  %    TRUE: use default da font%
    fds     [1];  %    TRUE: use old string font%
      % fdd=FALSE AND fds=FALSE: read font from command string %

  (frozen) RECORD                                                   3D
    fzvspec[36],
    fzvspc2[36],
    fzstid [36],
    fznext [18],
    fzexis  [1];

      (frzl) EXTERNAL = frozen.SIZE;
                                                                    3D5A
  (xc)     RECORD  % xcord record for remote displays %             3E
    xc2     [6],  %    low-order 5 bits %
    xc1     [6];  %    high-order 5 bits %

  (yc)     RECORD  % ycord record for remote displays %             3F
    yc2     [6],  %    low-order 5 bits.%
    yc1     [6];  %    high-order 5 bits %

% VIEWSPEC RETURN RECORD %
  (pvsrecord) RECORD  % viewspec return record %                    4A
```

```
    vs1        [36],  %    updated viewspec word 1 %
    vs2        [36],  %    updated viewspec word 2 %
    vscacode   [18],  %    ptr to user content analyzer pgm %
    vsusqcod   [18];  %    ptr to user sequence gen. pgm %
       % a viewspec collection astring follows the pvsrecord in the
       function state record %

% MISCELLANEOUS RECORD DEFINITIONS %
  (card)   RECORD  % substitute candidate record %                    5A
     catnc  [18],  %    length of test string %
     carnc  [18],  %    length of replacement string %
     catbp  [36],  %    byte ptr to test string %
     carbp  [36],  %    byte ptr to replacement string %
     canxt  [18];  %    next card for this initial char %

  (corpag) RECORD  % group allocation data page format %              5B
     corlck [36],  %    lock for this page %
     cortim [36],  %    time of last locking %
     corcpy [36];  %    not used %

  (corpgr) RECORD  % core page status record %                        5C
     ctfull  [1],  %    true if the page is in use %
     ctfile  [5],  %    file to which the page belongs %
     ctpnum  [9],  %    page number within the file %
     ctfroz  [3];  %    number of reasons why frozen %
       % a single word; gives status for a given core page for random
       files %
       % The array CORPST is the core page status table and is made
       up of instances of the above record.  RFPMAX gives the number
       of core pages that may contain file pages.  The core pages are
       located at positions indicated by the array CRPGAD (core page
       address).  CORPST is indexed by numbers in the range [1,
       RFPMAX).  The starting location of page k is given by crpgad
       [k]. %

  (deliverymodes) RECORD  % journal delivery flags%                   5D
     delol   [1],  %    Deliver on-line %
     delnet  [1],  %    Deliver Network %
     delhc   [1];  %    Deliver hard copy %

  (thflgs) RECORD % flag word in file header %                        5E
     prvsts    [1],
     fhunused [35];

  (fileblockheader) RECORD  % fbhdl is length %                       5F
     fbnull [36],  %    unused %
     fbind   [9],  %    status table index %
     fbpnum  [9],  %    page number in file of this block %
     fbtype  [5];  %    type of this block
        hdtyp   = header
        sdbtyp  = data
        rngtyp  = ring
        jnktyp  = misc (such as keyword, viewchange etc.)%

  (filstr)  RECORD  % file status table record %                      5G
     flexis   [1],  %    true: entry represents an existant file %
```

```
   flhead   [9],  %    crgpad index of the file header %
   flhrws   [1],  %    this file in browse mode %
   fllock   [1],  %    file was locked by another user when loaded %
   flpcread [1],  %    PC read only-- write open failed (openpc) %
   flaccm   [8],  %    file access mask %
   fldirno [12],  %    directory number for the original file %
   flnoclos [1],  %     do not close this file %
   flpart  [18],  %    JFN for the partial copy %
   flbpart [18],  %    JFN for the browse partial copy %
   florig  [18],  %    JFN for the original file %
   flastr  [18],  %    address of the file name string %
   flpcst  [18],  %    address of partial copy name string %
   flbpcst [18],  %    address of browse partial copy name string %
   flnsw   [18];  %    address of nsw filename string %

      (filstl) EXTERNAL = filstr.SIZE;                              5G15A
      (filmax) EXTERNAL = 25;
                                                                    5G15B
(lock)    RECORD  % lock word in file descriptor block %              5H
   lkinit  [20],
   lkdirn  [10],
   acctyp   [3],
   ojdelf   [1];

(isqfwa)  RECORD  % length is 3 words %                             5I
   isstid  [36],  %    last completed stid %
   isbuff  [18],  %    sequential file buffer address %
   islevs  [18],  %    location of string for levadj %
   isinfno  [8],  %    sequential file number %
   istatnd  [8],  %    character that signals end of statement %
   islevb   [8],  %    number of leading blanks per level %
   islstb   [8],  %    number of leading blanks, last statement %
   isdpth   [8],  %    depth from initial psid %
   isfiltyp [1];  %    TRUE if tenex file, else from 940 %

(lhjfn)   RECORD  % lefthalf bits from gtjfn %                      5J
   lhjrun   [4],  %    unused rightmost bits %
   lhjb13   [1],  %    bit 13: always zero %
   lhjb12   [1],  %    bit 12: complement of B8 in gtjfn call %
   lhjb11   [1],  %    bit 11: ;T given %
   lhjb10   [1],  %    bit 10: account given %
   lhjb9    [1],  %    bit  9: protection given %
   lhjb8    [1],  %    bit  8: use lowest version %
   lhjb7    [1],  %    bit  7: use next highest version %
   lhjb6    [1],  %    bit  6: use highest version %
   vstar    [1],  %    asterisk for version number %
   estar    [1],  %    asterisk for extension name %
   tstar    [1],  %    asterisk for file name %
   dstar    [1],  %    asterisk for directory name %
   ustar    [1],  %    asterisk for unit %
   dvstar   [1],  %    asterisk for device %
   lfjlun  [18];  %    unused leftmost bits %

(mask)    RECORD  % calculator %                                    5K
   fld3     [6],
   fld2     [6],
```

```
    fld1      [6],
    round     [5],
    blnk      [1],
    oflo      [1],
    exsign    [2],
    exp2      [2],
    dpt       [1],
    dol       [1],
    dig       [1],
    just      [2],
    sign      [2];

(measrec) RECORD  % measurement record %                              5L
    msbegin [36],  %    location to begin measurement %
    msend   [36],  %    location to end measurements %
    msbinstn[36],  %    instruction displaced to begin measurement %
    mseinstn[36],  %    instruction displaced to end measurement %
    msmax   [36],  %    maximum number of passes to measure %
    mscurr  [36],  %    current number of passes measured %
    mslclk  [36],  %    clock at beginning of interval %
    mstclk  [36];  %    total clock time, this series %

        (msentl) EXTERNAL = measrec.SIZE;
                                                                    5L8A
(mrker)    RECORD                                                    5M
    mkname  [36],
    mkpsid  [18],
    mkfix    [1],
    mkccnt  [12],
    mkexis   [1];

        (mkrl) EXTERNAL = mrker.SIZE;
                                                                    5M5A
(ojtime)  RECORD  % on-line delivery time format %                    5N
    ojtitt  [12],  %    hour * 64 + fractional part of hour (64ths) %
    ojtnn    [6],  %    # deliveries (at ojtiii hour intervals) %
    ojtiii   [6],  %    delivery interval, 0 defaults to 1 %
    ojtijjj  [9];  %    slinker op flag %

(optlgs)  RECORD  % format of flags word passed to OP %              5O
    opform   [1],  %    TRUE then send form feeds %
    opsimff  [1],  %    TRUE if simulate form feeds %
    opwtpb[1],       %TRUE if wait on page breaks%
    opdisflg[1];     %TRUE if output (local) terminal from display%
(rfstr)   RECORD  % random file block status record %                5P
    rfexis   [1],  %    true if the block exists in the file %
    rfpart   [1],  %    true if block comes from partial copy %
    rfnull   [2],  %    unused %
    rfused  [10],  %    used word count for the block %
    rffree  [10],  %    free pointer for the block %
    rfcore   [9];  %    0 then not in core, else page index %

    % unallocated if entirely zero %

    % The table RFBS is broken into two sections each of which
    contains a block a records of the above type.  The first
```

```
        section includes RNGM entries from RFBS[RNGBAS] up to and
        including RFBS[RNGBAS+RNGM-1] and contains information about
        the ring blocks in the file.  The second section includes DTBM
        entries from RFBS[DTBBAS] up to and including
        RFBS[DTBBAS+DTBM-1] and contains information about the data
        blocks in the file.  The entry RFBS[RNGBAS+i] may also be
        referenced as RNGST[i]; likewise RFBS[DTBBAS+i] may be
        referenced as DTBST[i].  The index in RFBS of a block is the
        actual page number of the block in the file. %

        % Data blocks are allocated in the file starting with page
        DTBBAS.  Up to DTBM data blocks may be allocated, with data
        blocks given internal numbers from 0 to DTBM-1.  The array
        DTBST in the file header is the data block status table and
        contains a one word record for each potential data block.  A
        zero entry means that the block does not exist in the file,
        otherwise the entry is as described in the above record
        defintion.  A pointer to an SDB (PSDB) consists of a nine bit
        data block number in the range [0,DTBM) and a nine bit
        displacement from the start of the block.  The variable DTBL
        is maintained in each file header as the current upper bound
        on allocated data blocks for that file.  This is used to limit
        the search for a location for a new SDB.  The variable DTBLST
        contains the index of the block from which an SDB was last
        allocated or freed. %

(seqr) RECORD  % sequence generator work area -- sqwrkl words long %
                                                                    5Q
    swcstid[36],    % real STID of current statement %              5Q1
    swlbstid[36],   % STID of statement heading last branch in the
    sequence %                                                      5Q2
    swstid[36],     % "stid" of last item "port-sent" from this
    sequence -- it may be an ENDFIL, a stastr (pointer to an
    a-string), or the same as SWCSTID %                             5Q3
    swusqcod[18],   % address of user sequence generator code or 0 %
                                                                    5Q4
    swcacode[18],   % address of content anylyzer code or 0 %       5Q5
    swvspec[36],    % first word of viewspecs %                     5Q6
    swvsp2[36],     % second word of viewspecs %                    5Q7
    swaport[36],     % port id for stack establishing this sequence
    (stack a) %                                                     5Q8
    swbport[36],     % port id for stack containing this sequence
    (stack b) %                                                     5Q9
    swsvw[18],      % address of statement vector work area %      5Q10
    swslvl[6],      % level at which sequence was started %        5Q11
    swclvl[6],      % current statement's level %                  5Q12
    swktig[1],      % has first item in this sequence been port-send
    -- used for k viewspec %                                       5Q13
    swalloc[1],     % whether or not this work area is allocated % 5Q14
    swfflg[1],      % is this the first call on the content filter? %
                                                                   5Q15
    swstkdec[1],      %True if there is a stack declared for this
    work area%
    swstkalloc[1],    %whether or not the stack has been allocated %
    swfill[1],              %currently unused %
    swstkloc[18],     % address of the stack associated with this
```

```
    sequence work area or 0 implying a stack must be
    allocated%
    swstksz[18],              %size of stack associated with this
    sequence%
    swword[36]; % initialized to 0 in openseq: may be used by user
    filters, etc. %
    (sgwrkl) EXTERNAL ADDRESS = seqr.SIZE;
                                                                    5Q22

(shed)      RECORD  % substitute header record %                    5R
    sbrp     [36],  %    pointer to next space for card %
    sbdp     [36],  %    pointer to dispatcher %
    sbas     [36],  %    pointer to A-string for strings %
    sbtt     [36];  %    type of entity being substituted %
       %this field can be shortened; only needs to hold numbers up to
       32, at the outside %

(substr)    RECORD  % substitute stack entry record %               5S
    sbc1cnt  [12],
    sbc2cnt  [12],
    sbc3cnt  [12];

(tenexbits)RECORD   % bits as numbered by Tenex %                   5T
    bitfiller[32],
    bit3      [1],
    bit2      [1],
    bit1      [1],
    b0        [1];

(usrblk)    RECORD  % group allocation userjob entry record %       5U
    usrlnk   [36],  %    lh = back link to previous user block %
       % rh = forward link %
    usrno    [18],  %    user directory number (rh!) %
    usrflgs  [18],  %    flags (see defs) (lh!) %
    usrgrp   [36],  %    user group number %
    usrjob   [36],  %    user job number %
    usrtty   [36],  %    user tty number (-1 if detached) %
    usrtod   [36],  %    time logged in (for connectime) %
    usrrt    [36],  %    runtime for this job %
    usrtim   [36];  %    todclk of last runtime update %

       (lcard) EXTERNAL = card.SIZE; %length of card in words%    5U9A
       (lshed) EXTERNAL = shed.SIZE; %length of shed in words%
                                                                   5U9B
FINISH
```

```
< NINE, CATNUM.NLS.2, >, 8-Jul-77 13:23 KJM ;;;;( NLS, CATNUM.NLS;36,
), 23-MAY-74 12:58 HGL ;

FILE catnum  % <ARCSUBSYS>XL10  <RELNINE>catnum %  % (arcsubsys,xL10,)
(RELNINE,catnum.rel,) %

ALLOW!
%General Number System Library Routines%
   (gcatnums) %Get count catalog numbers for subcollection(s) substr.
   type is passed to getcnum. Return numbers (separated by EOL's) in
   retstr if Non-zero%
   PROCEDURE (retstr, idntsr,  substr, count, type);                    3A
      LOCAL numfstid;
      LOCAL STRING cnumber[10];
      REF retstr, substr, idntsr;

      IF &retstr THEN *retstr* _ NULL;
      conjdir(TRUE);
      numfstid _ 0;
      INVOKE (sigcls);
      numfstid _ openlock(0, jflname($"tcnumbers"));
      WHILE (count _ count-1) >= 0 DO
         BEGIN
         %now assign a number, and mark it used%
            getcnum(&idntsr, $cnumber, &substr, type, numfstid);
            IF &retstr THEN *retstr* _ *retstr*, *cnumber*, EOL;
         END;
      close(numfstid.stfile := 0);
      conjdir(FALSE);
      DROP (sigcls);
      RETURN;

      (sigcls) CATCHPHRASE;                                             3A16
         BEGIN
         DISABLE (sigcls);
         IF numfstid.stfile THEN sigclose(numfstid.stfile := 0);
         conjdir(FALSE);
         END;

      END.

   (ckcnum) %This procedure accepts a catalog number in *number*, an
   identification-list in *idlist*, and searches the preassigne-number
   list for that number.%
   PROCEDURE (type, number, idlist, nstid);                             3B
      % The action it takes depends on the parameter TYPE:
         =0: if the number is preassigned to this user then
            return true
         =1: if the number is preassigned to this user and not inuse
         then
            mark number as in use
            return true
         =2: if the number is preassigned to this user and inuse then
            mark not in use
            return true
      Assumes connected to journal directory.
```

```
%
LOCAL stid, savestid, retval, numstid;
LOCAL TEXT POINTER  z1, z2;
LOCAL STRING reservelist[50], idstr[20];
REF number, idlist;

numstid _ 0;
%Set up SIGNAL for errors%
   INVOKE (sigcls);
%open number file, and look for number%
   numstid _ IF nstid THEN nstid
      ELSE  openlock(0, jflname($"tcnumbers"));
   enablaccess(numstid.stfile, jrnlaccess);
   IF NOT numtype(numstid, &number, $"PREASSIGNED": savestid)
   THEN
      err($"Number Not Reserved");
   IF NOT FIND SF(savestid) [*)] $NP $PT $NP ^z1 ([*;] /  [D *-]
   < $PT > / [NP]) ^z2 _z2 THEN
      BEGIN
      lockjo(1);
      err($"Illegal number file format -- Please report this to
      the NIC");
      END;
   *reservelist* _ +z1 z2; %ident field%
   FIND SF(*idlist*) ^z2 ^z1;
   LOOP %process list of idents%
      BEGIN
      FIND z2 > $(SP/",) ("( [*)] $(SP/",) /) ^z1 $(LD/"-) ^z2;
      *idstr* _ +z1 z2;
      IF NOT idstr.L THEN EXIT LOOP; %end of ident list%
      IF FIND SF(*reservelist*) [*idstr*] THEN
         BEGIN
         CASE type OF
            = 0: %number is reserved by this user%
               retval _ TRUE;
            = 1: %mark as in-use%
               IF retval _ ( NOT FIND SF(savestid) [" INUSE "] )
               THEN
                  ST savestid _ SF(savestid) SE(savestid), "
                  INUSE "
            = 2:  %mark as not-in-use%
               IF retval _ (FIND SF(savestid) [" INUSE "] ^z2 <
               ['I] CH ^z1) THEN
                  ST z1 z2 _ NULL;
            ENDCASE err ($"Illegal operation type encountered in
            ckcnum");
         IF NOT nstid THEN close(numstid.stfile := 0);
         DROP (sigcls);
         RETURN(retval, savestid);
         END;
      END;
   err($"number reserved by someone else!");

(sigcls) CATCHPHRASE;                                          3B13
   BEGIN
   DISABLE (sigcls);
```

```
      IF NOT nstid AND numstid.stfile THEN
         sigclose(numstid.stfile := 0);
      CONTINUE;
      END;

   END.


(ckrfcnum) %This procedure accepts a catalog number in *number*, an
RFC number in *rfcnumber*, an identification-list in *idlist*, and
searches the preassigne-number list for that number.%
PROCEDURE (type, rfcnumber, number, idlist, rstid, nstid);          3D
   % The action it takes depends on the parameter TYPE:
      =0: if the numbers are preassigned to this user then
         return true
      =1: if the numbers are preassigned to this user and not inuse
      then
         mark numbers as in use
         return true
      =2: if the numbers are preassigned to this user and inuse then
         mark not in use
         return true
   Assumes connected to journal directory.
   %
   LOCAL stid, nsavstid, rsavstid, retval, numstid, rnumstid;
   LOCAL TEXT POINTER  z1, z2;
   LOCAL STRING rfcreservelist[50], reservelist[50], idstr[20];
   REF number, rfcnumber, idlist;

   numstid _ rnumstid _ 0;
   %Set up SIGNAL for errors%
      INVOKE (sigcls);
   %open number file, and look for number%
      numstid _ IF nstid THEN nstid
         ELSE  openlock(0, jflname($"tcnumbers"));
      rnumstid _ IF rstid THEN rstid
         ELSE  openlock(0, jflname($"RFCNUMBERS"));
      enablaccess(numstid.stfile, jrnlaccess);
      enablaccess(rnumstid.stfile, jrnlaccess);
      IF NOT numtype(rnumstid, &rfcnumber, $"PREASSIGNED": rsavstid)
      THEN
         err($"RFC Number Not Reserved");
      IF NOT numtype(numstid, &number, $"PREASSIGNED": nsavstid)
      THEN
         err($"Catalog Number Not Reserved");
      IF NOT FIND SF(rsavstid) [")] $NP $PT $NP ^z1 ([";] /  [D "-]
      < $PT > / [NP]) ^z2 _z2 THEN
         BEGIN
         lockjo(1);
         err($"Illegal RFC number file format -- Please report this
         to the NIC");
         END;
      *rfcreservelist* _ +z1 z2; %ident field%
      IF NOT FIND SF(nsavstid) [")] $NP $PT $NP ^z1 ([";] /  [D "-]
      < $PT > / [NP]) ^z2 _z2 THEN
         BEGIN
```

```
      lockjo(1);
      err($"Illegal Catalog number file format -- Please report
      this to the NIC");
      END;
   *reservelist* _ +z1 z2; %ident field%
   FIND SF(*idlist*) ^z2 ^z1;
   LOOP %process list of idents%
      BEGIN
      FIND z2 > $(SP/",) ("( [")] $(SP/",) /) ^z1 $(LD/"-) ^z2;
      *idstr* _ +z1 z2;
      IF NOT idstr.L THEN EXIT LOOP; %end of ident list%
      IF (FIND SF(*rfcreservelist*) [*idstr*]) AND (FIND
      SF(*reservelist*) [*idstr*]) THEN
         BEGIN
         CASE type OF
            = 0: %number is reserved by this user%
               retval _ TRUE;
            = 1: %mark as in-use%
               BEGIN
               IF retval _ ( NOT FIND SF(rsavstid) [" INUSE "] )
               THEN
                  ST rsavstid _ SF(rsavstid) SE(rsavstid), "
                  INUSE ";
               IF NOT FIND SF(nsavstid) [" INUSE "] THEN
                  ST nsavstid _ SF(nsavstid) SE(nsavstid), "
                  INUSE "
               END;
            = 2:  %mark as not-in-use%
               BEGIN
               IF retval _ (FIND SF(rsavstid) [" INUSE "] ^z2 <
               ["I] CH ^z1) THEN
                  ST z1 z2 _ NULL;
               IF FIND SF(nsavstid) [" INUSE "] ^z2 < ["I] CH ^z1
               THEN
                  ST z1 z2 _ NULL;
               END;
            ENDCASE err ($"Illegal operation type encountered in
            ckcnum");
         IF NOT nstid THEN close(numstid.stfile := 0);
         IF NOT rstid THEN close(rnumstid.stfile := 0);
         DROP (sigcls);
         RETURN(retval, rsavstid, nsavstid);
         END;
      END;
   err($"Number reserved by someone else!");

   (sigcls) CATCHPHRASE;                                    3D110
      BEGIN
      IF NOT nstid AND numstid.stfile THEN
         sigclose(numstid.stfile := 0);
      DISABLE (sigcls);
      IF NOT rstid AND rnumstid.stfile THEN
         sigclose(rnumstid.stfile := 0);
      CONTINUE;
      END;
```

```
   END.

(getcnum) %This is the routine which is used for reserving catalogue
numbers%
PROCEDURE (identsr, numbsr, typesr, destination, stid);                3E
   % Formal Parameters:
      IDENTSR is a string containing the ident of the requestor of
      the number
      NUMBSR is the string which will be used to return the reserved
      number
      TYPESR contains the address of a string which identifies the
      process requesting the number (e.g. JOURNAL or XDOC)
      DESTINATION contains: 0 if number is to be moved to "IN USE
      branch, 1 for "PREASSIGNED", and 2 for "USED"%
   LOCAL  count, numstid, trap, capsav;
   LOCAL TEXT POINTER z1, z2, z3, z4;
   LOCAL STRING tempsr[15];
   REF identsr, numbsr, typesr;

   %assumes connected to journal directory%
   %Abort if journal not in operation%
      IF jolock() THEN
         werr($"number system temporarily unavailable");
   %Set up SIGNAL for errors%
      numstid _ 0;
      trap _ FALSE;
      INVOKE (sigcls);
   IF stid THEN
      BEGIN
      numstid _ stid;
      numstid.stpsid _ origin;
      END
   ELSE numstid _ openlock(0, jflname($"tcnumbers"));
   enablaccess(numstid.stfile, jrnlaccess); %let him write it%
   IF (numstid _ namelook(numstid, $"FREE")) = endfil THEN
      err($"Illegal Number File Format -- report to NIC");
   IF NOT FIND SF(numstid) ["C/D] ^z1 THEN
      err($"Number File Exhausted -- report to NIC");
   %Check to see if it is an interval%
   trap _ TRUE;
   capsav _ trapcc();
   IF FIND z1 < "C > THEN
      BEGIN
      %it is an interval, so we will have to work a little%
      FIND z1 $D ^z2 %First number%
         $(-D) ^z3 $D ^z4; %second number%
         *numbsr* _ z1 z2;
         *tempsr* _ z3 z4;
         IF *tempsr* = *numbsr*  THEN %equal...delete whole schmere%
            BEGIN
            FIND z1 _z1 ["]] ^z2;
            ST z1 z2 _ NULL;
            END
         ELSE
            BEGIN %add one from first number %
            *tempsr* _ *numbsr*;
```

```
            bumpstr($tempsr);
            ST z1 z2 _ *tempsr*;
            END;
      END
   ELSE
      BEGIN %single number%
      FIND > _z1 z1 $D ^z2;
      *numbsr* _ z1 z2;
      ST z1 z2 _ NULL;
      END;
   %Now move number entry to proper list%
      CASE destination OF
         = 2: *tempsr* _ "USED";
         = 1: *tempsr* _ "PREASSIGNED";
         ENDCASE *tempsr* _ "INUSE";
      %move it to the branch%
         numstid _ movecnum($tempsr, &numbsr, &identsr, &typesr,
         numstid);
   notrapcc(capsav);
   trap _ FALSE;
   IF NOT stid THEN close(numstid.stfile := 0);
   DROP (sigcls);
   RETURN(numstid);

   (sigcls) CATCHPHRASE;                                          3E27
      BEGIN
      DISABLE( sigcls);
      IF trap THEN notrapcc(capsav);
      IF stid = 0 THEN sigclose(numstid.stfile := 0);
      CONTINUE;
      END;

   END.

(movecnum) %Move designated number under deisgnated list (branch
name). If the list doesn't exist, create it%
PROCEDURE (brname, numbsr, identsr, typesr, stid);                3F
   LOCAL tstid;
   LOCAL TEXT POINTER z1, z2;
   LOCAL STRING tempsr[200];
   REF brname, numbsr, identsr, typesr;

   IF (tstid _ namelook(stid, &brname)) = endfil THEN
      BEGIN %branch doesn't exist..create it%
      *tempsr* _ "(, *brname*, ") ";
      FIND SF(*tempsr*) ^z1 SE(*tempsr*) ^z2;
      stid _ cinssta(getail(getsub(stid)), levsuc, $z1, $z2);
      END
   ELSE stid _ tstid;
   *tempsr* _ NULL;
   %  !gtad();  no good for TOPS20%
   dtfrmt(-1, $tempsr);
   astruc(&identsr);
   *tempsr* _ "(C",*numbsr*,  ") ", *typesr*, SP,  *identsr*, "; ",
   *tempsr*;
   FIND SF(*tempsr*) ^z1 SE(*tempsr*) ^z2;
```

```
    stid _ cinssta(stid, levdown, $z1, $z2);
    RETURN(stid);
    END.

(numtype) %Compares name of branch containing numstr to string
typesr, and returns true if same, false if not.  Returns stid of
number statement as second result%
PROCEDURE (stid, numstr, typesr);                                    3G
    LOCAL savestid, retstid;
    LOCAL STRING tempsr[50];
    REF numstr, typesr;

    *tempsr* _ "C, *numstr*;
    IF (savestid _ retstid _ namelook(stid, $tempsr)) = endfil THEN
       err($"Number not reserved!");
    %Now check that it is in right branch%
       WHILE (stid _ getup(savestid)).stpsid # origin DO savestid _
       stid;
       CCPOS SF(savestid);
       xtrnam($tempsr, $swork, "(, "); %read name into tempsr%
    RETURN(*tempsr* = *typesr*, retstid);
    END.

(usedcnum) %This routine moves the number in numbsr from the in-use
or pre-assigned list to the used list%
PROCEDURE (numbsr, nstid);                                           3H
    LOCAL numstid, stid, xstid;
    LOCAL TEXT POINTER z1;
    LOCAL STRING tempsr[100];
    REF numbsr;

    %Set up SIGNAL for errors%
       numstid _ 0;
       INVOKE (sigcls);
    numstid _ IF nstid THEN nstid
       ELSE openlock(0, jflname($"tcnumbers"));
    enablaccess(numstid.stfile, jrnlaccess);
    %now find number%
       *tempsr* _ "C, *numbsr*;
       IF (stid _ namelook(numstid, $tempsr)) = endfil THEN
          BEGIN
          *tempsr* _ *numbsr*, " Not Reserved";
          err($tempsr);
          END;
       numstid _ stid; %stid of statement%
       WHILE (xstid _ getup(stid)).stpsid # origin DO stid _ xstid;
       %by here, stid contains head of branch...check name%
       CCPOS SF(stid);
       xtrnam($tempsr, $swork, "(, "); %read statement name into
       tempsr%
       IF (*tempsr* # "INUSE") AND (*tempsr* # "PREASSIGNED") THEN
          err($"Number neither reserved nor in use -- command
          aborted.");
       IF (stid _ namelook(stid, $"USED")) = endfil THEN err($"Bad
       USED branch missing from the Number File -- report to NIC");
       stid _ cmovsta(stid, levdown, numstid, FALSE, 0); %move
```

```
      statement%
      %Now strip off in use if necessary%
         IF FIND SF(stid) ["INUSE"] < ["I] ^z1 > THEN ST stid _
         SF(stid) z1;
   IF NOT nstid THEN close(numstid.stfile := 0);
   DROP (sigcls);
   RETURN;

   (sigcls) CATCHPHRASE;                                          3H14
      BEGIN
      DISABLE( sigcls);
      IF nstid = 0 THEN sigclose(numstid.stfile := 0);
      CONTINUE;
      END;

   END.

%Support for RFC numbers%
   (rfcex) %reserve an RFC (Request For Comments) and a Journal number
   given the currently available info contained in the string pointed
   to by STID.  Reserve it on behave of the user whose ident is in
   authrsr%
   PROCEDURE (authorstr, titlestr, sendstr, onlineflag, rfcnum,
   catnum);                                                        4A
      LOCAL STRING tempsr[250];
      LOCAL rnumstid;
      REF authorstr, titlestr, sendstr, rfcnum, catnum;
      %-----------------%
      rnumstid _0;
      INVOKE (sigcls);
      conjdir(TRUE);
      %Get RFC Number%
         %open RFC Number file%
            rnumstid _ openlock(0, jflname($"RFCNUMBERS"));
         rnumstid _ getcnum(&authorstr, $rfcnum, $" NWG/RFC ", 1,
         rnumstid);
      %Get catalog number%
         getcnum(&authorstr, $catnum, $" NWG/RFC ", 1, 0);
      %Now fix up rfc number file to reflect all of the desired info%
         IF onlineflag THEN
            *tempsr* _
               "Online
               Distribution: ", *sendstr*, EOL, "Title: ", *titlestr*
         ELSE
            *tempsr* _
               "Offline
               Distribution: ", *sendstr*, EOL, "Title: ", *titlestr*;
         ST SF(rnumstid) _ SF(rnumstid) SE(rnumstid), EOL,
            "#, *catnum*, EOL, *tempsr*;
         close(rnumstid.stfile := 0);
         conjdir(FALSE);
      %Type onto logging tty%
         *tempsr* _ *catnum*, " RFC#", *rfcnum*;
         jlog($tempsr, $"Pre-Assigned RFC", &authorstr);
         DROP (sigcls);
      RETURN;
```

```
    (sigcls) CATCHPHRASE;                                              4A14
       BEGIN
       DISABLE (sigcls);
       IF rnumstid.stfile THEN sigclose(rnumstid.stfile := 0);
       conjdir(FALSE);
       CONTINUE;
       END;

    END.


(rfcsyscheck) %see if the RFC system is enabled for this host%
PROCEDURE;                                                             4B
    IF lhostn # rfchost THEN
       err($"RFC Numbers Not Available on This Host");
    RETURN;
    END.
FINISH
```

```
< NINE, CCALC.NLS;1, >, 28-Apr-78 14:17 KIRK ;;;;
FILE ccalc % L10 <RELNINE>CCALC.rel %% (arcsubsys, x110,)
(RELNINE,CCALC.rel,) %
ALLOW!
% DECLARATIONS %
   (expf) FIELD=[38,9:27];                                                    3A
   (fone)=201488;                                                             3B
   (ften)=204588;                                                             3C
   (ften1)=15189;                                                             3D
   (ftenth)=175631463146B;                                                    3E
   (ften1)=142314631463B;                                                     3F
   (acsizdef) STRING = "Format too small for accumulator
   ACCUMULATOR SET TO ZERO";                                                  3G
   (acsizerr) STRING = "Format too small for accumulator
   FORMAT RESET TO DEFAULT";                                                  3H
   (insizerr) STRING = "Format too small for input";                          3I
% ARITHMETIC %
   (qcmult) PROCEDURE (ar,ma);   % floating multiply ar _ ar * ma %          4A
      %parameters are all addresses%
      !MOVE R3,ma;
      !MOVE R2,ar;
      !MOVE R4,0(R2);
      !MOVE R5,1(R2);
      !MOVEM R4,R6;
      !FMPR R6,1(R3);
      !FMPR R5,0(R3);
      !UFA R5,R6;
      !FMPL R4,0(R3);
      !UFA R5,R6;
      !FADL R4,R6;
      !MOVEM R4,0(R2);
      !MOVEM R5,1(R2);
      RETURN; END.

   (qcadd) PROCEDURE (ar,ma);   % floating add ar _ ar + ma %               4B
      !MOVE R3,ma;
      !MOVE R2,ar;
      !MOVE R4,0(R2);
      !MOVE R5,1(R2);
      !UFA R5,1(R3);
      !FADL R4,0(R3);
      !UFA R5,R6;
      !FADL R4,R6;
      !MOVEM R4,0(R2);
      !MOVEM R5,1(R2);
      RETURN END.

   (qcdiv) PROCEDURE (ar REF,ma REF);   % floating divide ar _ ar / ma
   %                                                                          4C
      LOCAL quo[2];
      qcdivw(&ar,&ma,$quo);
      ar _ quo;
      ar[1] _ quo[1];
      RETURN; END.

   (qcdivw) PROCEDURE (ar,ma,guo);   % floating divide quo _ ar / ma %
```

4D

```
    %parameters are all addresses%
    !MOVE R3,ma;
    !MOVE R2,ar;
    !MOVE R4,0(R2);
    !MOVE R5,1(R2);
    !FDVL R4,0(R3);
    !MOVN R6,R4;
    !FMPR R6,1(R3);
    !UFA R5,R6;
    !FDVR R6,0(R3);
    !FADL R4,R6;
    !MOVE R2,quo;
    !MOVEM R4,0(R2);
    !MOVEM R5,1(R2);
    RETURN END.

(qcsub) PROCEDURE (ar,ma);   % floating subtract ar _ ar - ma %    4E
    !MOVE R3,ma;
    !MOVE R2,ar;
    !MOVE R4,0(R2);
    !MOVE R5,1(R2);
    !DFN R4,R5;
    !UFA R5,1(R3);
    !FADL R4,0(R3);
    !UFA R5,R6;
    !FADL R4,R6;
    !DFN R4,R5;
    !MOVEM R4,0(R2);
    !MOVEM R5,1(R2);
    RETURN; END.

(qcneg) PROCEDURE (ar);   % floating negate ar _ -ar %              4F
    !MOVE R2,ar;
    !MOVE R4,0(R2);
    !DFN R4,1(R2);
    !MOVEM R4,0(R2);
    RETURN; END.

(qfloat) PROCEDURE (ar REF,ch);   % convert character (ascii 0-9) to
floating point number %                                             4G
    %ar is address of result%
    LOCAL expa;

    expa _ 2008;
    R3_0;
    .expf_(ch-"0");
    WHILE SKIP !TLNN R3,777000B DO
        BEGIN
        !LSH R3,-1;
        BUMP expa;
        END;
    .expf_expa;
    ar_R3;
    BUMP &ar;
    R3_0;
```

```
    .expf_expa-27;
    ar_R3;
    RETURN; END.

(nfloat) PROCEDURE (instring REF,fl1 REF,fl2 REF);   % convert
string to floating point number %                                    48
    %convert character string to double precision floating point%
    %instring is address of input string, fl1, fl2 will contain
    results%
    LOCAL chr,fchr,fchr1,term,term1,sflg;

    %Set up flag for negative umber%
    sflg _ FALSE;
    %set READC pointer to head of string%
    CCPOS  SF(*instring*);
    fl1 _ fl2 _ 0; %clear result area%
    %take care of portion to left of decimal point%
    CASE (chr _ READC) OF
        IN ['0,'9]:
            BEGIN
            qcmult(&fl1, $ften);
            qfloat($fchr,chr);
            qcadd(&fl1, $fchr);
            REPEAT CASE;
            END;
        = '-:
            BEGIN
            sflg _ TRUE;
            REPEAT CASE;
            END;
        = '.:
            %take care of portion to right of decimal point%
            BEGIN
            term_fone;
            term1_0; %temporaries%
            CASE (chr _ READC) OF
                IN ['0,'9]:
                    BEGIN
                    qcmult($term,$ftenth);
                    qfloat($fchr,chr);
                    qcmult($fchr,$term);
                    qcadd(&fl1,$fchr);
                    REPEAT CASE;
                    END;

                = '-:
                    BEGIN
                    sflg _ TRUE;
                    REPEAT CASE;
                    END;

                = ENDCHR:
                    EXIT CASE;

            ENDCASE REPEAT CASE; %drop editing characters%
            END;
```

```
      = ENDCHR: EXIT CASE;

      ENDCASE REPEAT CASE; %drop editing characters%

   RETURN (sflg);
   END.

% SUPPORT ROUTINES %
   (errx) PROCEDURE (whence,ar REF);                                    5A
      %JSYS dfout has returned with an error condidtion. R4 contains
      error mnemonics. FLOTX1 and  FLOTX2 indicate column overflow. All
      other error returns are either calulator or tenex bugs. If the
      error occurred on an accumulator and the format is the default,
      maximum no. of digits, the accum will we set to zero.  If the
      format is for less than the maximum, the format will be changed
      to the maximum.%
      IF whence = 1 THEN err($insizerr)
      ELSE
         BEGIN
         IF dfoutm.round < 12 THEN
            BEGIN
            dfoutm_0640141202000;
            err($acsizerr);
            END
         ELSE
            BEGIN
            ar _ ar[1] _ 0;
            err($acsizdef);
            END;
         END;
      RETURN;
      END.


   (cacfile) PROCEDURE; %get/create calc file%                          5B
      %If user has done a Quit Return, take care of his file, which had
      the partial copy closed to make it read only.  Otherwise, load
      his calc-ident file, if he has one.  If not, create a null
      calc-ident file%
      LOCAL fileno,  %file number%
            stid;  %origin of file if Quit Return was done%
      LOCAL STRING flnam[50],filestr[65];
      INVOKE (cinitsig,cbdfle);
      clcname($flnam); %get calc-ident file name%
      IF NOT (fileno _ opwk(0, $flnam)) THEN RETURN(FALSE, FALSE);
      RETURN(TRUE,fileno);
      (cinitsig) CATCHPHRASE;                                           5B9
         BEGIN
         CASE SIGNALTYPE OF = aborttype:
            CASE SIGNAL OF = -5: %bad file%
               TERMINATE; %let calling routine handle it%
            ENDCASE;
         ENDCASE;
         CONTINUE;
         END;
```

```
   (cbdfle): RETURN(FALSE, fileno);                                    5B10
   END.

(clcname) PROCEDURE(flnam REF);%set calculator file name to login
directory CALC file%                                                   5C
   !JSYS gjinf; %get login directory number%
   gdname(R1,&flnam); %conver to string%
   *flnam* _ "<,*flnam*,">CALC-",*initsr*;
   RETURN;
   END.

(qcins) PROCEDURE (cfloat REF,opstrng REF); %insert value%            5D
   %into calc-ident file - updates global CASTID %
   %cfloat = value to insert at stid. csign is sign and/or original
   operator. opstrng is address of final formatted operand.%
   LOCAL tempstid;
   LOCAL TEXT POINTER tp1, tp2;
   REF cda;

   qfloutp(&cfloat,&opstrng,1);
   *opstrng* _ *opstrng*, SP, *signstr*;
   tempstid _ castid;
   tp1 _ tp2 _ $opstrng;
   tp1.stastr _ 1;
   tp2.stastr _ 1;
   tp1[1] _ 1;
   tp2[1] _ opstrng.[ + 1;
   castid _ cinssta(tempstid,sucdir,$tp1,$tp2);
   %recreate display and take care of scrolling%
   IF nlmode = fulldisplay AND NOT cda.daauxiliary THEN
      BEGIN
      IF cda.dacrow < cda.damrow THEN
         BEGIN
         dpset(dspstrc,tempstid,endfil,endfil);
         seldsp();
         END
      ELSE
         BEGIN
         cda.dacsp _ castid;
         dafrmt(&cda, 0);
         END;
      END;
   IF NOT cda.daauxiliary THEN bmoff();
   RETURN;
   END.

(qfloutp) PROC(ar, os,whence);%gfloating output a value at address
ar to string at address os%                                           5E
   LOCAL TEXT POINTER tp1, tp2, tp3;
   LOCAL i, j, k, fst;
   LOCAL STRING cos[16];
   REF os;
   *cos* _ "                ";
   !MOVE R3,ar;
   !MOVE R2,0(R3);
   !MOVE R3,1(R3);
```

```
    R4_dfoutm; %indicator of precision%
    R1 _ &os + 440700000001B; %TENEX string designator constant%
    IF NOT SKIP !JSYS dfout THEN errx(whence,ar);
    os.L _ slngth(&os + chbmty, R1);
    IF *os*[1] = "- THEN fst _ 2
    ELSE fst _ 1;
    FOR i _ fst UP UNTIL > os.L DO
       IF *os*[i] = "0 THEN *os*[i] _ SP
       ELSE EXIT;
    IF cacflg THEN % comma formatting %
       BEGIN
       j _ os.L + 2;
       FOR i _ os.L DOWN UNTIL = 1 DO
          IF *os*[i] # ". THEN
             BEGIN
             *cos*[j] _ *os*[i];
             j _ j-1;
             END
          ELSE EXIT;
       *cos*[j] _ ".;
       i_ i-1;
       j _ j-1;
       LOOP
          BEGIN
          k _ i;
          FOR i DOWN UNTIL = k-3 DO
             BEGIN
             IF i < 1 THEN EXIT LOOP 2;
             IF *os*[i] = SP THEN EXIT LOOP 2;
             IF *os*[i] = "- THEN EXIT LOOP 2;
             *cos*[j] _ *os*[i];
             j _ j-1;
             END;
          IF *os*[i] = SP THEN EXIT LOOP;
          IF i < 1 THEN EXIT LOOP;
          IF *os*[i] # "- THEN
             BEGIN
             *cos*[j] _ ",;
             j_ j-1;
             REPEAT LOOP;
             END
          ELSE EXIT LOOP;
          END;
       IF calflg THEN BEGIN
          FIND SP(*cos*) $SP (D/".) ^tp1 _tp1;
          FIND SE(*cos*) $NP ^tp2;
          IF *os*[1] = "- THEN k _ 2
          ELSE k _ 1;
          *os*[k TO os.M] _ tp1 tp2;
          END
       ELSE
          BEGIN
          FIND SE(*cos*) ^tp3 $NP ("./) ^tp2;
          ST tp2 tp3 _ NULL;
          IF *os*[1] = "- THEN k _ 2
          ELSE k_ 1;
```

```
            os.L _ k + cos.L;
            *os*[k TO os.L] _ *cos*;
            END;
         END;
      FIND SE(*os*) $NP ('./) ^tp1;
      os.L _ tp1[1] - 1;
      RETURN END.

FINISH of CCALC
```

```
< NINE, COLSRT.NLS.3, >, 6-Jul-77 08:03 KJM ;;;;< NLS, COLSRT.NLS;58,
>, 12-JUN-74 16:29 KEV ;
FILE colsrt   % <ARCSUBSYS>XL10  <RELNINE>colsrt % % (arcsubsys,xL10,)
(RELNINE,colsrt.rel,) %
ALLOW!
%Declarations%
   (less) = -1;                                                          3A
   (gtr) = 1;                                                            3B
   (random) = 4;                                                         3C
   (ovrflo) = 0;                                                         3D
   (rlleft) FIELD = [1,18:18];                                           3E
   (chif) FIELD = [0,7:35];                                              3F
   REF colda, tda, colsw;
%...new stuff for new sort-merge primitives...%
   %v   %
   %vcv   %
   %vcvtp  %
   %merge-update working size parameters%
      %mersiz  block size%
      %merrff  merge block reformat flag%
   %update cells%
      %vcvu  key value for present call%
      %vcvul  list of stid'S for decision proc%
      %vcvulp end+1 of vcvul%
%..Default key procedure...%
   (defkey)PROCEDURE(stid,outb,num);                                     5A
      LOCAL pst,val,flg,char,cnt,wdcnt,firstf;
      REF outb;
      CCPUS SF(stid);
      wdcnt _ 0;
      flg _ FALSE;
      firstf _ TRUE;
      WHILE num > 0 DO BEGIN
         cnt_0;
         val_0;
         pst_chif+$val; %pointer into val%
         WHILE cnt<5 AND NOT flg DO BEGIN
            IF ((char_READC) = "@ AND NOT firstf) OR char = ENDCHR THEN
            BEGIN
               flg_TRUE;
               EXIT; END;
            ^pst_IF char IN ["a,"z] THEN char-40B ELSE char;
            BUMP cnt;
            firstf _ FALSE;
            END;
         outb_val;
         BUMP wdcnt;
         BUMP &outb;
         BUMP DOWN num;
         IF flg THEN EXIT;
         END;
      RETURN (flg,wdcnt); END.
%...On-line Commands...%
   (kprget)PROC(window);  %get key procedure address%                    6A
      LOCAL kpr;
      REF kpr;
```

```
    &colda _ dsparea(window);
    IF (&kpr _ colda.daukeycode) = 0 THEN &kpr _ $defkey;
    RETURN (&kpr);
    END.
(colgdest)PROC(stid);                                                  6B
    IF getfhd(stid) THEN RETURN (getup(stid),levdown)
       ELSE RETURN (getprd(stid),levsuc);
    END.
(sortbranch)PROC(stidx, window);                                       6C
    LOCAL stid1,stid2;
    IF (stid1_getsub(stidx))=stidx THEN RETURN;
    stid1_plxset (stid1 : stid2);
    sort (stidx,stid1,stid2,levdown,kprget(window),FALSE);
    RETURN; END.
(sortgrp)PROC(bug1,bug2, window);                                      6D
    LOCAL stid1,stid2,stidx,rlevcnt;
    stid1 _ grptst (bug1, bug2 : stid2);
    stidx_colgdest (stid1 : rlevcnt);
    sort (stidx,stid1,stid2,rlevcnt,kprget(window),FALSE);
    RETURN; END.
(sortplx)PROC(bug1, window);                                           6E
    LOCAL stid1,stid2;
    stid1 _ plxset(bug1 : stid2);
    sort(getup(stid1),stid1,stid2,levdown,kprget(window),FALSE);
    RETURN; END.
%...Sort and friends...%
(sort) PROC(stidx,stid1,stid2,rlevcnt,kpr,copyf);                      7A
    LOCAL i,cxxg,stid,stidi;
    REF cxxg, keypr;
    %map out pages for buffer%
       hmapout();
    INVOKE (sigsort);
    vcvtp _ $vcv;
    stid1_grptst(stid1,stid2:stid2);
    IF copyf THEN
       stid1 _ ccopgro(stidx, rlevcnt, stid1, stid2, 0: stid2)
    ELSE
       cmovgro(stidx, rlevcnt, stidi_stid1, stid2, 0);
    &keypr_kpr;
    vtop_0;
    WHILE (stid_gsstid($stidi,stid2))#endfil DO a3ddrec(stid);
    i_vtop+1;
    WHILE (i_i-1) >= 2 DO BEGIN
       s3iftup (1,i);
       v[1]_v[i]:=v[1];
       END;
    %rearrange fast-fast-fast%
       IF (stid_[v[vtop]+4]) # stid2 THEN
          ctragro (stid, stid, stid2, stid2, FALSE, 0);
       IF stid = stid1 THEN stid1 _ stid2;
       IF (stid_[v[1]+4]) # stid1 THEN
          ctragro (stid, stid, stid1, stid1, FALSE, 0);
       FOR i_2 UP UNTIL > vtop DO stosuc (stid, stid _ [v[i]+4]);
    %map nls pages back in%
       hmapin();
    DROP (sigsort);
```

```
   RETURN;

   (sigsort) CATCHPHRASE;                                          7A19
      BEGIN
      DISABLE (sigsort);
      hmapin();
      CONTINUE;
      END;

   END.
(s3iftup)PROC(i,n);                                                7B
   %sort tree into vector...from floydd%
   LOCAL j;
   v_v[i];
   LOOP
      IF (j _ i+i) <= n THEN BEGIN
         IF j<n THEN IF  c3cmpare(j, j+1) THEN BUMP j;
         IF NOT c3cmpare(j, 0) THEN BEGIN
            v[i:=j] _ v[j]
            END ELSE EXIT;
      END ELSE EXIT;
   v[i] _ v;
   RETURN END.
(a3ddrec)PROC(stid);                                               7C
   %add stid into vector%
   LOCAL index, index1, wdcnt;
   IF (index _ vtop _ vtop + 1) > vmax THEN ABORT(sorterr,$"Too Many
   Input statements");
   v[vtop] _ vcvtp;
   [vcvtp+3] _ keypr (stid, vcvtp,3 : wdcnt);
   [vcvtp+4]_stid;
   WHILE wdcnt < 3 DO [vcvtp + (wdcnt:=wdcnt+1)] _ 0;
   vcvtp _ vcvtp+5;
   WHILE (index1 _ index / 2) > 0 DO
      IF NOT  c3cmpare(index, index1) THEN BEGIN
         v[index1] _ v[index := index1] := v[index1]
         END ELSE EXIT;
   RETURN END.

(lodkey)PROC(n,stid);                                              7D
   LOCAL vcvn,wdcnt;
   vcvn _ n*mersiz + $vcv;
   v _ vcvn+2;
   IF ([vcvn] _ stid) = endfil THEN BEGIN
      [vcvn+1] _ 7777760000001B;
      [vcvn+2] _ 377777777777B; %sentinal%
      RETURN (FALSE);
      END;
   [vcvn+3]_[vcvn+4]_0;
   !PUSH S,R1;
   R1 _ keypr (stid, vcvn+2, 3 : wdcnt);
   .r1left _ -wdcnt;
   [vcvn+1] _ R1;
   !POP S,R1;
   RETURN (TRUE);
   END.
```

```
(gsstid)PROC (st1,st2);                                                 7E
   LOCAL stid;
   REF st1;
   IF (stid_st1)#endfil THEN BEGIN
      IF stid = st2 THEN st1 _ endfil ELSE st1_getsuc(stid);
      END;
   RETURN (stid);
   END.
(c3cmpare)PROC(i1,i2);                                                  7F
   LOCAL stid1,stid2,skcnt,corder;
   corder_ (IF (R1 _ v[i1]) < (R2 _ v[i2]) THEN 1 ELSE 0);
   !HRLI R1,-3;
   (ccma): !MOVE R3,0(R1); !CAMN R3,0(R2); GOTO ccme;                   7F4
   A1 _ 0; !CAMG R3,0(R2); BUMP A1; RETURN;
   (ccme): BUMP R2; !AOBJN R1,ccma;                                     7F6
   stid1 _ [R1+1];
   stid2 _ [R2+1];
   R2 _ R2 - 1000001B;
   skcnt _ 3;
   GOTO ccmxh;
   (ccmxa): BUMP skcnt;                                                 7F12
      !MOVE R3,0(R1); !CAMN R3,0(R2); GOTO ccmxe;
      A1 _ 0; !CAMG R3,0(R2); BUMP A1; RETURN;
   (ccmxe): !AOBJP R1,ccmxh;                                            7F13
   (ccmxf): !AOBJN R2,ccmxa;                                            7F14
      IF kysrdone (stid2,R2,skcnt:R2) THEN RETURN (FALSE);
      GOTO ccmxa;
   (ccmxh):                                                             7F15
      !PUSH S,R2;
      IF kysrdone (stid1,R1,skcnt:R1) THEN BEGIN
         !POP S,R2;
         !AOBJN R2,ccmxha;
         IF kysrdone (stid2,R2,skcnt:R2) THEN RETURN (corder);
         (ccmxha): RETURN (TRUE);                                       7F15B4
         END;
      !POP S,R2;
      GOTO ccmxf;
   END.
(kysrdone)PROC(stid,loc,skcnt);                                         7G
   LOCAL wdcnt,done;
   REF loc;
   IF loc > 0 THEN RETURN (TRUE);
   IF loc = 0 THEN BEGIN
      IF vcvtp > vcvend THEN ABORT (sorterr, $"Sort buffer full,
      aborted");
      !PUSH S,R1;
      IF skcnt>0 THEN BEGIN
         R1_vcvtp-skcnt;
         .r1left_-skcnt;
         (kysra): !PUSH S,0(R1); !AOBJN R1,kysra;                       7G4C3
         END;
      done _ keypr (stid,vcvtp-skcnt,skcnt+10:wdcnt);
      IF skcnt>0 THEN BEGIN
         R1 _ vcvtp; .r1left_ skcnt; !TRNA 0,0;
         (kysrb): !POP S,0(R1);                                         7G4E2
         R1 _ R1 - 1000001B;
```

```
            !JUMPGE R1,kysrb;
            END;
         R1 _ vcvtp; .r1left_skcnt-wdcnt;
         vcvtp _ vcvtp+wdcnt-skcnt;
         loc_R1;
         [vcvtp]_done;
         BUMP vcvtp;
         !POP S,R1;
         END;
      RETURN (FALSE,loc);
      END.
FINISH
%...Merge, Update and friends...%
   MOVED AFTER FINISH TO SAVE SPACE IN THE SYSTEM
   (mergbranch)PROC(bug1,bug2, window);                                    9B
      LOCAL stid1,stid2,stid3,stid4,zerw,stidx,rlevcnt,kpr;
      REF kpr;
      zerw_0;
      &kpr_kprget(window);
      stidx_bug1;
      rlevcnt_levdown;
      IF (stid1_getsub(bug1)) = bug1 THEN stid1_endfil
          ELSE stid1_plxset(stid1 : stid2);
      IF (stid3_getsub(bug2)) = bug2 THEN RETURN;
      stid3 _ plxset(stid3 : stid4);
      WHILE merge ($stidx,$stid1,&kpr,110,FALSE) # -2 DO NULL;
      RETURN; END.
   (merggrp)PROC(bug1,bug2,bug3,bug4, window);                             9C
      LOCAL zerw,stidx,rlevcnt,kpr; %BEWARE bug1 to zerw are sequenced%
      REF kpr;
      zerw_0;
      bug1_grptst(bug1,bug2 : bug2);
      bug3_grptst(bug3,bug4 : bug4);
      stidx_colgdest(bug1 : rlevcnt);
      &kpr _ kprget(window);
      WHILE merge($stidx,$bug1,&kpr,110,FALSE) # -2 DO NULL;
      RETURN; END.
   (mergplx)PROC(bug1,bug2, window);                                       9D
      LOCAL stid1,stid2,stid3,stid4,zerw,stidx,rlevcnt,kpr;
      REF kpr;
      zerw_0;
      &kpr_kprget(window);
      stid1_plxset(bug1 : stid2);
      stidx_getup(stid1); rlevcnt_levdown;
      stid3_plxset(bug2: stid4);
      WHILE merge ($stidx,$stid1,&kpr,110,FALSE) # -2 DO NULL;
      RETURN; END.
   (merge)PROC(stidx,inlst,kpr,percnt,copyf);                              9E
      LOCAL rlevcnt,ngrp,cxxb,n,stidn,cdtbm,crngm,cdtbloc,crngloc;
      REF stidx,cxxb,cdtbloc,crngloc,keypr;
      %map out nls pages%
         hmapout();
      &keypr_kpr;
      rlevcnt_stidx[1];
      &cxxb_IF copyf THEN $ccopgro ELSE $cmovgro;
      mersiz_40B;
```

```
   merrff_FALSE;
   % set structure and date block bounds %
      cdtbm_dtbm*percnt/100;
      crngm_rngm*percnt/100;
      &cdtbloc_$dtbi+&crngloc_fileheadEstidx.stfileJ-$filhed;
      &crngloc_&crngloc+$rngl;
   n_inlst; ngrp_0;
   WHILE En] # 0 DO BEGIN
      BUMP ngrp; n_n+2;
      END;
   LOOP BEGIN
      FOR n _ 0 UP UNTIL >= ngrp DO BEGIN
         R1_$v; .rlleft_-n; R2_v;
         (lp): !EXCH R2,1(R1); !AOBJN R1,lp;                    9E12A2
         lodkey (n,gostid(n+n+inlst));
         merorder(n+1);
         END;
      %test for no input%
         IF Ev-2J = endfil THEN BEGIN hmapin(); RETURN(-2); END;
      LOOP BEGIN
         IF cdtbloc>=cdtbm OR crngloc>=crngm THEN BEGIN hmapin();
         RETURN(-1); END;
            %this is turned off until outfull is written%
         IF merrff THEN BEGIN
            mersiz_mersiz+40B;
            merrff_ FALSE;
            EXIT;
            END;
         n_ (v-$vcv)/mersiz;
         stidn_g3nstid(n+n+inlst);
         stidx _cxxb (stidx, rlevcnt, Ev-2J, Ev-2J, FALSE, 0);
         rlevcnt _ levsuc;
         IF NOT lodkey (n, stidn) THEN BEGIN hmapin(); RETURN(n);
         END;
         merorder(ngrp);
         END;
      END;
   END.
(merorder)PROC(n);                                              9F
   LOCAL i;
   FOR i _ 1 UP UNTIL>= n DO BEGIN
      R2 _ vEiJ; R1_vEi-1J;
      IF cmmpare() # 0 THEN RETURN;
      vEiJ _ vEi-1J := vEiJ;
      END;
   RETURN; END.
(cmmpare)PROC;                                                  9G
   !HLL R1,-1(R1); !HLL R2,-1(R2);
   (cmma): !MOVE R3,0(R1); !CAMN R3,0(R2); GOTO cmme;           9G2
   IF SKIP !CAMG R3,0(R2) THEN RETURN(FALSE);
   RETURN(TRUE);
   (cmme): !AOBJP R1,cmmx;                                      9G5
   (cmmf): !AOBJN R2,cmma;                                      9G6
   IF keydone($R2) THEN RETURN (FALSE);
   GOTO cmma;
   (cmmx): IF keydone($R1) THEN BEGIN                           9G9
```

```
      !AOBJN R2,cmmxa;
      IF keydone($R2) THEN RETURN(-1);
      (cmmxa): RETURN(TRUE);                                          9G9C
      END;
   GOTO cmmf;
   END.
(keydone)PROC(loc);                                                   9H
   LOCAL r,wdcnt,sloc,n,vcvl;
   REF loc;
   vcvl_$vcv;
   IF loc < vcvl THEN vcvl_$vcvu;
   n _ (loc-vcvl)/mersiz;
   r _ vcvl + n*mersiz;
   IF [r+1] .A 7777B = 0 THEN BEGIN
      wdcnt _ (sloc_loc)-r+11;
      IF wdcnt+3 > mersiz THEN BEGIN
         merrff_ TRUE;
         RETURN (TRUE);
         END;
      !PUSH S,R1;
      !PUSH S,R2;
      R1 _ keypr ([r], r+2, wdcnt : wdcnt);
      .r1left _ -wdcnt;
      [r+1] _ R1;
      R1 _ sloc .A 777777B;
      .r1left _ R1-wdcnt - 2 - r;
      A1_R1;
      !POP S,R2;
      !POP S,R1;
      loc_A1;
      RETURN (FALSE);
      END;
   RETURN (TRUE);
   END.
(update)PROC(stidx,inlst,kpr,udpr,percnt,initf);                      9I
   LOCAL ngrp,n,dhold,cdtbm,crngm,cdtbloc,crngloc;
   REF udpr,initf,cdtbloc,crngloc;
   REF keypr,vcvulp;
   &keypr_kpr;
   mersiz_40B;
   merrff_FALSE;
   % set structure and date block bounds %
      cdtbm_dtbm*percnt/100;
      crngm_rngm*percnt/100;
      &cdtbloc_$dtbl+&crngloc_filehead[[stidx].stfile]-$filhed;
      &crngloc_&crngloc+$rngl;
   IF initf THEN BEGIN
      initf_dhold_FALSE;
      &vcvulp_$vcvul;
      vcvulp_0;
      END;
   LOOP BEGIN
      n_inlst; ngrp_0;
      WHILE [n] # 0 DO BEGIN
         BUMP ngrp; n_n+2;
         END;
```

```
      FOR n _ 0 UP UNTIL >= ngrp DO BEGIN
         R1_$v; .r1left _-n; R2_v;
         (lpa): !EXCH R2,1(R1); !AOBJN R1,lpa;                    9I9C2
         lodkey (n,gostid(n+n+inlst));
         merorder(n+1);
         END;
      LOOP BEGIN
         IF merrff THEN BEGIN %reformat key area%
            mersiz_mersiz+40B;
            merrff_FALSE;
            EXIT;
            END;
         IF cdtbloc>=cdtbm OR crngloc>=crngm THEN RETURN(-1);
         R1 _ $vcvu+2; R2_v;
         IF dhold THEN
         CASE cmmpare() OF
            =1: BEGIN %send list%
               CALL udpr($vcvul, stidx);
               &vcvulp_$vcvul;
               dhold_FALSE;
               END;
            =0:  NULL; %error, file out of order%
            ENDCASE;
         IF  [v-2] = endfil THEN RETURN (-2);
         IF NOT dhold THEN BEGIN %copy to hold value%
            R1 _ [v-1]; R2_ -(.r1left) + 2; R2 _ R2 .A 777777B;
            R1_$vcvu; .r1left _ v-2; !BLT R1,vcvu(R2);
            dhold_TRUE;
            END;
         %add item to list%
            vcvulp _ [v-2];
            BUMP &vcvulp;
            vcvulp _ n _ (v-$vcv)/mersiz;
            BUMP &vcvulp;
            vcvulp _ 0;
         %get new item%
            IF NOT lodkey (n,g3nstid (n+n+inlst)) THEN RETURN (n,
            $vcvul);
            merorder(ngrp);
         END;
      END;
   END.
(gostid)PROC(loc);                                                  9J
   LOCAL lb;
   REF loc,lb;
   &lb_&loc+1;
   IF loc#endfil THEN loc_grptst(loc,lb:lb);
   RETURN (loc);
   END.
(g3nstid)PROC(loc);                                                 9K
   LOCAL lb;
   REF loc,lb;
   &lb_&loc+1;
   IF loc # endfil THEN BEGIN
      IF loc = lb THEN loc _ endfil ELSE loc _ getsuc(loc);
      END;
```

```
      RETURN (loc);
      END.
%old collector-sorter
   (colsrt)PROC;                                                    10A
      LOCAL count, stmtfg;
      STATE _ colstate, spec;
      LOOP BEGIN
         crlf();
         count _ 0;
         WHILE (count _ count+1) <= fedind DO todco(SP);
         todco("-);
         echoff();
         CASE inpcuc() OF
            ="D: BEGIN echo($"Delete Keys? ");
               strpkey _ answer() END;
            ="E: BEGIN echo($"Execute Quit");
               getctc(); reset() END;
            ="F: BEGIN echo($"File List ");
               IF stmtfg _ (lookc() # SP) THEN BEGIN %read names from a
               statement%
                  tbug($b1);
                  CCPOS SF(b1);
               END ELSE input();
               csetif(stmtfg); END;
            ="G: BEGIN echo($"Go ");
               getctc();
               chkprm();
               ccontrol(&tda); END;
            ="I: BEGIN echo($"Initialise ");
               getctc();
               ninfil _ 0; *outnam* _ NULL;
               strpkey _ lngflg _ sortfg _ FALSE;
               smtmax _ 12000; END;
            = "L: BEGIN echo($"Length Key ?");
               lngflg _ answer() END;
            ="M: BEGIN echo($"Max Number Stmts/outtput file =");
            smtmax _ gttnum(smtmax) END;
         ="O: BEGIN echo ($"Output File Name: ");
            coutnam() END;
         ="S: BEGIN echo($"Sort ?");
            sortfg _ answer() END;
         ="V: tv();
      ENDCASE error();
      END;
      END.
   (chkprm)PROC;                                                    10B
      IF outnam.L = 0 THEN err($"No Output File Name");
      IF ninfil = 0 THEN err($"No Input Files");
      RETURN END.
   (csetif)PROC(stmtfg);                                            10C
      %READ list of file nams from keyboard fro now%
      LOCAL infastr, maxchr, chrused, strlng, char; REF infastr;
      ninfil _ chrused _ 0;
      maxchr _ 100*5;
      &infastr _ $nambuf;
      IF NOT stmtfg THEN echon();
```

```
   LOOP BEGIN
      IF ninfil = 50 THEN err($"Too Many Input File Names");
      maxchr _ maxchr-5; %room o for astring control word%
      infastr.LH _ maxchr-chrused;
      *infastr* _ NULL;
      IF stmtfg THEN BEGIN
         WHILE (char _ READC) = SP DO NULL;
         IF char = ENDCHR THEN EXIT;
         WHILE (char # ",) AND (char # SP) AND (char # ENDCHR) DO
         BEGIN *infastr* _ *infastr*, char; char _ READC END;
      END ELSE BEGIN
         crlf();
         IF curchr = CA THEN EXIT;
         IF rdlit(&infastr, $rnmdel) = ovrflo THEN err($"Name space
         exhausted");
      END;
      namlst[ninfil] := ninfil+1] _ &infastr;
      &infastr _ &infastr+ (strlng _ infastr.L+4)/5+1; %address of
      next room%
      chrused _ chrused+strlng+5;
   END;
   namlst[ninfil] _ -1; %end marker%
   RETURN END.
(coutnam)PROC;                                                    10D
   %read outfile name%
      *outnam* _ NULL;
      echon();
      crlf();
      IF rdlit($outnam, $rnmdel) = ovrflo THEN err($"Name Too
      Long");
      cversion _ 1;
      RETURN END.
(ccmpare)PROC(i1,i2);                                             10E
   LOCAL ccval;
   %Use p1 through p4 globally%
   IF (ccval_vcv[i2]-vcv[i1])>0
      THEN RETURN (TRUE)
      ELSE IF ccval<0 THEN RETURN (FALSE);
   IF vcv[i1] = 0 THEN RETURN (FALSE);
   ccval_v[i1]; CCPOS SF(ccval); FIND ^p1;
   ccval_v[i2]; CCPOS SF(ccval); FIND ^p3;
   LOOP BEGIN
      IF NOT setkey($p1, $p2) THEN RETURN(TRUE);
      IF NOT setkey($p3, $p4) THEN RETURN(FALSE);
      *stn* _ p2 p1;
      *lit* _ p4 p3;
      CASE cmpstr($stn, $lit) OF
         =less: RETURN(TRUE);
         =gtr: RETURN(FALSE);
         ENDCASE;
      END;
      END.
(setkey)PROC(ptr1, ptr2);                                         10F
   REF ptr1, ptr2;
   CCPOS ptr1;
   RETURN(FIND '@ ^ptr2 ['@] ^ptr1 _ptr1, ptr1, ptr1[1]);
```

```
   END.
(cccval) PROCEDURE(stid);                                          10G
   LOCAL ps,char,val,cnt;
   CCPOS SF(stid); FIND ^p1;
   IF NOT setkey ($p1, $p2) THEN RETURN (0);
   ps_chif+$val;
   cnt_val_0;
   IF lngflg THEN BEGIN
      ^ps _ MIN (77B, p1[1] - p2[1]);
      BUMP cnt;
      END;
   CCPOS p2;
   WHILE cnt < 5 AND (char_READC) # "@ DO BEGIN
      ^ps_char;
      BUMP cnt;
      END;
   RETURN (val); END.
(addrec)PROC(stid);                                                10H
   %add stid into vector%
   LOCAL index, index1;
   IF (index _ vtop _ vtop + 1) > vmax THEN creror($"Too Many Input
   statements");
   v[vtop] _ stid;
   vcv[vtop] _ cccval(stid);
   WHILE (index1 _ index / 2) > 0 DO
      IF NOT  ccmpare(index, index1) THEN BEGIN
         vcv[index1] _ vcv[index] := vcv[index1];
         v[index1] _ v[index := index1] := v[index1]
         END ELSE EXIT;
   RETURN END.
(gnstid)PROC;                                                      10I
   %Return next stid in sequence%
   LOCAL stid;
   (gnstdloop): IF NOT infopn THEN IF NOT opninf() THEN
   RETURN(endfil);                                                 10I3
   IF (stid _ seqgen(&colsw)) = endfil THEN BEGIN
      closeseq(&colsw);
      close(colda.dacsp.stfile :=0);
      infopn _ 0;
      GOTO gnstdloop END;
   RETURN(stid) END.

(opstmt)PROC(stid, s4ort);                                         10J
   %copy next input statemnt to file%
   LOCAL TEXT POINTER z1, z2;
   IF outcnt >= smtmax THEN BEGIN
      %file full...open next one%
      IF NOT s4ort THEN IF cversion > 1 THEN closeu(cstid.stfile);
      cstid.stfile _ copen(cversion := cversion+1, s4ort);
      cstid.stpsid _ origin;
      outcnt _ 0;
      END;
   FIND SF(stid) ^z1 SE(stid) ^z2;
   cstid _IF stid.stastr THEN
      cinssta(cstid, levsuc, $z1, $z2)
   ELSE ccopsta(cstid, levsuc, stid, FALSE, 0);
```

```
    BUMP outcnt;
    IF s4ort THEN addrec(cstid);
    RETURN END.
(ccontrol)PROC(da);                                                    10K
    LOCAL stid;
    LOCAL STRING messtr[20];
    REF da;
    &colda _ &da;
    initcl(); %initialise things%
    WHILE (stid _ gnstid()) # endfil DO opstmt(stid, sortfg);
    *messtr* _ EOL, "Input Finished";
    dismes (1, $messtr);
    IF sortfg THEN finsort();
    cwrpup();
    reset() END.

% (cmpstr) has been moved to (utilty,cmpstr) %
(opninf)PROC;                                                          10M
    %1--open nexxt input file in list
    2-- set up seqwrk to read in first psid (not origin)
    3----Retturn TRUE if ok, false if error or no more files%
    LOCAL wrkstr, colstd, sv1, sv2;
    REF wrkstr;
    &wrkstr _ getstring(1000, $dspblk);
    INVOKE (sigfree);
    (opninloop):                                                       10M8
    IF namlst[namndx _ namndx+1] <= 0 THEN
       BEGIN
       freestring(&wrkstr, $dspblk);
       DROP (sigfree);
       RETURN(FALSE); %no more files%
       END;
    dismes(0);
    cpysr(namlst[namndx], &wrkstr); %copy into wrkstr for openning%
    IF NOT colda.dacsp.stfile _  open(cgtjfn(&wrkstr), &wrkstr) THEN
       BEGIN
       *wrkstr* _ *wrkstr*, "--Open Fail";
       dismes (1, &wrkstr);
       GOTO opninloop
       END
    ELSE colda.dacsp.stpsid _ origin;
    dismes (1, &wrkstr);
    colda.dacsp_getsub(colda.dacsp);
    &colsw_openseq(colstd _ colda.dacsp, seqend(colstd, sv1 _
    colda.davspec, sv2 _ colda.davspc2),sv1,
       sv2,colda.dausqcod,colda.dacacode);
    infopn _ TRUE;
    DROP (sigfree);
    freestring(&wrkstr, $dspblk);
    RETURN;

    (sigfree) CATCHPHRASE;                                             10M22
       BEGIN
       DISABLE (sigfree);
       freestring(&wrkstr, $dspblk);
       CONTINUE;
```

```
      END;

   END.

(cgtjfn)PROC(astrng);                                                  10N
   %RETURN jfn of file indicated by astrng%
   LOCAL jfn;
   IF NOT (jfn _ lgetjfn (0, astrng, $nlsext, gtjoif, astrng)) THEN
      err (astrng);
   RETURN(jfn) END.
(initcl)PROC;                                                          10O
   vtop _ infopn _ 0;
   cversion _ 1;
   namndx _ -1;
   outcnt _ smtmax+1;
   IF [flntadr(colda.dacsp.stfile)].flexis THEN
   close(colda.dacsp.stfile := 0);
   RETURN END.

(copen)PROC(version, s4ort);                                           10P
   %Open next output file, and return NLS file numbeer%
   LOCAL fileno, jfn, flgbits, nfl;
   IF s4ort THEN *lit* _ "xxxsort", version+60B ELSE
   *lit* _ *outnam*, version+60B;
   IF NOT (jfn _ lgetjfn (0, $lit, $nlsext, getgtjflg(write, origff,
   oldvrsn), $lit)) THEN creror($"Output File Open Fail");
   jfnstr(jfn, $lit);
   fileno _ addfile(jfn, $lit);
   %now do dummy open and close to create it%
      IF NOT sysopen(jfn, write, random, $lit) THEN err($lit);
      IF NOT sysclose(jfn .V 4B11, $lit) THEN err($lit);
   IF NOT opnfil(fileno, $lit) THEN err($lit);
   IF lit.L > empty THEN
      BEGIN
      dismes (2, $lit);
      <AUXCOD, pause> (2000);
      END;
   resetf(fileno); %reset file%
   RETURN(fileno) END.
(creror)PROC(errcod);                                                  10Q
   nlmode _ typewriter;
   IF errcod THEN dismes (1, errcod);
   cwrpup();
   err($"Fatal Error In COLSORT");
   END.

(cwrpup)PROC;                                                          10R
    closeu(cstid.stfile := 0);
   closeall();
   *outnam* _ *outnam*, "1; %oper and load first output file%
   colda.dacsp.stfile _ open(cgtjfn($outnam), $outnam);
   colda.dacsp.stpsid _ origin;
   RETURN END.
(tinsort)PROC;                                                         10S
   LOCAL i, fflag, oldtime, timeused;
   LOCAL STRING messtr[25];
```

```
    oldtime _ 0;
    i _ vtop+1;
    WHILE (i_i-1) >= 2 DO BEGIN
       !runtm(4B11); timeused _ R1/R2;
       IF timeused - oldtime > 60 THEN BEGIN %type out a message%
          oldtime _ timeused;
          *messtr* _ "Sort Running, time = ";
          dismes (1, $messtr);
          IF NOT SKIP !nout(101B, oldtime, 10) THEN err(0);
          dismes (1, $" Seconds, node = ");
          IF NOT SKIP !nout(101B, vtop-i+1, 10) THEN err(0);
          END;
       siftup(1,i);
       v[1] _ v[i] := v[1];
       vcv[1] _ vcv[i] := vcv[1];
       END;
    %now put files in order%
       cversion _ 1;
       i _ 0;
       outcnt _ smtmax+1;
       WHILE (i _ i+1) <= vtop DO BEGIN opstmt(v[i], FALSE); IF
       strpkey THEN BEGIN CCPOS SF(cstid);
          FIND ^p1;
          fflag _ FALSE;
          WHILE setkey($p1, $p2) DO fflag _ TRUE;
          IF fflag THEN ST p1 _ p2 SE(p2);
       END;
       END;
       %delete keys here%
    RETURN END.

(siftup)PROC(i,n);                                                    10T
    %sort tree into vector...from floydd%
    LOCAL j;
    v_v[i]; vcv_vcv[i];
    LOOP
       IF (j _ i+i) <= n THEN BEGIN
          IF j<n THEN IF  ccmpare(j, j+1) THEN BUMP j;
          IF NOT ccmpare(j, 0) THEN BEGIN
             vcv[i] _ vcv[j];
             v[i:=j] _ v[j]
             END ELSE EXIT;
       END ELSE EXIT;
    v[i] _ v; vcv[i] _ vcv;
    RETURN END.
```

```
< NINE, CORESUPPORT.NLS;1, >, 11-JUN-75 11:14 KJM
;;;;<NLS>CORESUPPORT.NLS;1, 26-NOV-73 21:49 CHI ;
FILE coresupport   % <ARCSUBSYS>XL10  <RELNINE>coresupport %  %
(arcsubsys,xL10,) to (RELNINE,coresupport,) %
ALLOW!
% inpfbk %
   (chrpos) PROCEDURE (char, astrng); %scan string for character%      3A
      %chr in first parameter, a-string addr in second%
      %return TRUE if char in string, FALSE otherwise.%
      %-----------------%
      LOCAL tmpchr;
      REF astrng;
      CCPOS *astrng*;
      LOOP
         IF (tmpchr _ READC) = char THEN RETURN(TRUE)
         ELSE IF tmpchr = ENDCHR THEN RETURN(FALSE);
      END.

   (clrbut) %clear the todas input and (optionally) output%            3B
      %if outflg is non-zero the output buffer is also cleared.
      nothing is  returned%
      %-----------------%
      PROCEDURE (outflg);
         %Dummy procedure (commented out)
         IF outflg THEN
            BEGIN
            %%jsys to clear output buffer%%
               r1 _ 101B;
               !JSYS cfobf;
            END;
         %%jsys to clear input buffer%%
            r1 _ 100B;
            !JSYS cfibf;
         buffn _ buffs; %% reset input buffer pointers %%
         %
         RETURN;
         END.

FINISH
```

(cdr)	<nine, delimiters, 045>	PROCEDURE	4A
(findr)	<nine, delimiters, 0127>	PROCEDURE	4J
(grpar)	<nine, delimiters, 014>	PROCEDURE	3B
(iddr)	<nine, delimiters, 099>	PROCEDURE	4G
(idldr)	<nine, delimiters, 0107>	PROCEDURE	4H
(idr)	<nine, delimiters, 051>	PROCEDURE	4B
(ldr)	<nine, delimiters, 0133>	PROCEDURE	4K
(littoink)	<nine, delimiters, 0143>	LOCAL	5A
(ndr)	<nine, delimiters, 070>	PROCEDURE	4D
(nmdr)	<nine, delimiters, 089>	PROCEDURE	4F
(pixdr)	<nine, delimiters, 029>	PROCEDURE	3C
(stdr)	<nine, delimiters, 05>	PROCEDURE	3A
(tdr)	<nine, delimiters, 058>	PROCEDURE	4C
(vdr)	<nine, delimiters, 0121>	PROCEDURE	4I
(wdr)	<nine, delimiters, 082>	PROCEDURE	4E

```
< NINE, DELIMITERS.NLS;3, >, 28-OCT-75 22:16 LLC ;;;;
FILE delimiters % <ARCSUBSYS>XL10 to <RELNINE>DELIMITERS %%
(arcsubsys,xl10,) (RELNINE,delimiters.rel,) %
%THIS code serves as a dummy back end for the middle end. The
procedures contained here are those in the PE called by the middle-end%
% DELIMITER ROUTINES %
   (stdr) PROCEDURE( % truncates a bug selection to a statement
   selection %                                                          3A
      bug,      % text ptr for bug %
      tptr1,% first text pointer %
      tptr2);% second text pointer %
      REF bug, tptr1, tptr2;
      %---------------%
      FIND SF(bug) ^tptr1 SE(bug) ^tptr2;
      RETURN;
      END.

   (grpdr) PROCEDURE( % delimits a structural group %                   3B
      % FORMAL ARGUMENTS %
         tptr1, % addrss of first text pointer %
         tptr2, % address of second text pointer %
         ret1,  % address of the first result text pointer %
         ret2); % address of the second result text pointer %
      % RETURNS %
         % text pointers ret1 and ret2 are set up to delimit the group
         identified by the bug selections tptr1 and tptr2 %
      REF % VARIABLES %
         tptr1, tptr2, ret1, ret2;
      %------------%
      ret1[1] _ ret2[1] _ 1;
      ret1 _ grptst( tptr1, tptr2: ret2);
      RETURN;
      END.

   (plxdr) PROCEDURE( % delimits a structural plex %                    3C
      % FORMAL ARGUMENTS %
         tptr1, % addrss of first text pointer %
         ret1,  % address of the first result text pointer %
         ret2); % address of the second result text pointer %
      % RETURNS %
         % text pointers ret1 and ret2 are set up to delimit the plex
         identified by the bug selection tptr1 %
      REF % VARIABLES %
         tptr1, ret1, ret2;
      %------------%
      ret1[1] _ ret2[1] _ 1;
      ret1 _ plxset( tptr1: ret2);
      RETURN;
      END.

% entity finding routines - known as delimiters %
   (cdr) PROCEDURE (bug,ptr1,ptr2);                                     4A
      REF bug, ptr1, ptr2;
      FIND bug > ^ptr1 CH ^ptr2;
      RETURN;
      END.
```

```
(idr) PROCEDURE (bug,ptr1,ptr2);                                        4B
   LOCAL TEXT POINTER tp1;
   REF bug, ptr1, ptr2;
   FIND bug ^tp1 < $NP ^ptr1 tp1 > CH $NP ^ptr2;
   RETURN;
   END.

(tdr) PROCEDURE (bug1,bug2,ptr1,ptr2);                                  4C
   REF bug1, bug2, ptr1, ptr2;
   LOCAL TEXT POINTER tp1;
   IF POS SF(bug1) # SF(bug2) THEN err($"invalid text selection")
   ELSE
      IF POS bug1 < bug2 THEN
         FIND bug1 ^ptr1 bug2 > CH ^ptr2
      ELSE
         FIND bug2 ^tp1 bug1 > CH ^ptr2 tp1 ^ptr1;
   RETURN;
   END.

(ndr) PROCEDURE (bug, ptr1, ptr2); %verify number %                     4D
   LOCAL sflg;
   REF bug, ptr1,ptr2;
   sflg _ TRUE;
   IF NOT FIND bug < ($(D/ './ ',/ '-/ '+) ^ptr1)
   THEN err($"invalid number selection");
   IF NOT FIND ptr1 > ('-/'+/FR sflg) ((1$3D $(', 3D) ('. 1$D/-D)
   ^ptr2)/
      (1$D ('. 1$D/TRUE) ^ptr2)/('. 1$D ^ptr2))
   THEN err($"invalid number selection");
   FIND ptr2 < -D ^ptr2;
   RETURN;
   END.

(wdr) PROCEDURE (bug,ptr1,ptr2);                                        4F
   % This procedure delimits a "WORD" given a pointer to a character
   in a string.  The pointers PTR1 and PTR2 are set to the first and
   last characters, respectively, of the word.%
   REF bug, ptr1, ptr2;
   FIND bug > CH $LD ^ptr2 bug < $LD ^ptr1;
   RETURN;
   END.

(nmdr) PROCEDURE (bug,ptr1,ptr2);                                       4F
   % statement name (and statement number) delimiter %
   REF bug, ptr1, ptr2;
   FIND bug >
      $(LD / '- /'' / '@  ) ^ptr2
      bug < $(LD / '- /'' / '@  ) ^ptr1;
      % @ is alphabetic zero %
   RETURN;
   END.

(iddr) PROCEDURE (bug,ptr1,ptr2); %IDENT delimiter routine%             4G
   REF bug, ptr1, ptr2;
   FIND bug >
```

```
        $1("&/"^) $(LD / "- /"" ) ^ptr2
      bug < $(LD / "- /"" ) ^ptr1;
   RETURN;
   END.

(idldr) PROCEDURE (bug1,bug2,ptr1,ptr2);                                4H
   REF bug1, bug2, ptr1, ptr2;
   LOCAL TEXT POINTER tp1, tp2;
   IF POS SF(bug1) # SF(bug2) THEN err($"invalid IDENTLIST selection
   -- must be contained in one statement")
   ELSE
      IF POS bug1 < bug2 THEN
         FIND bug1 ^ptr1 bug2 > CH ^ptr2
      ELSE
         FIND bug2 ^tp1 bug1 > CH ^ptr2 tp1 ^ptr1;
   iddr(&bug1, &ptr1, $tp2);
   iddr(&bug2, $tp1, &ptr2);
   RETURN;
   END.

(vdr) PROCEDURE (bug,ptr1,ptr2);                                        4I
   REF bug, ptr1, ptr2;
   FIND bug > CH $PT ^ptr2 bug < $PT ^ptr1;
   RETURN;
   END.

(findr) PROCEDURE (bug,ptr1,ptr2); %file name -- TENEX syntax%          4J
   REF bug, ptr1, ptr2;
   FIND bug > $(LD/"./";/">/"-) ^ptr2 bug < $(LD/"./";/">/"-/"<)
   ^ptr1;
   RETURN;
   END.

(idr) PROCEDURE (bug,ptr1,ptr2);                                        4K
   REF bug, ptr1, ptr2;
   FIND bug ^ptr1 ^ptr2;
   RETURN;
   END.

%support  routines%
   (littolnk)   % turn a lit into a link maybe %                        5A
      PROCEDURE
         (type, % TRUE -> should be old/new filelink %
         astrng REF  %addr of literal string%
         );
      LOCAL temp, dstrng REF;
      LOCAL TEXT POINTER tp1, tp2;

      &dstrng _ &astrng;
      temp _ type;
      FIND SE(*astrng*) ^tp2 SF(*astrng*) ^tp1 ;
      FIND $(SP/TAB);
      IF FIND ( "( / "< / "--" ) THEN temp _ TRUE;
      LOOP
         CASE READC OF
            = "), = ">, = ",, = ":
```

```
            BEGIN
            temp _ TRUE;
            END;
         = '':
            CASE READC OF
               = ENDCHR:
                  err($"Missing Character After Single Quote");
               ENDCASE;
         = '":
            LOOP
               CASE READC OF
                  = '": EXIT LOOP;
                  = ENDCHR:
                     BEGIN
                     &dstrng _ getstring(astrng.L + 4, $dspblk);
                     *dstrng* _ tp1 tp2, '" ;
                     EXIT LOOP 2;
                     END;
                  ENDCASE;
         = ENDCHR: EXIT LOOP;
         ENDCASE;
   IF temp OR type THEN
      BEGIN
      &dstrng _ getstring(astrng.L + 4, $dspblk);
      *dstrng* _ tp1 tp2;
      IF NOT FIND tp1 > $(SP/TAB) ( '( / '< / "--" ) THEN
         *dstrng* _ '<, tp1 tp2;
      FIND SE(*dstrng*) $(SP/TAB);
      CASE type OF
         = TRUE:
            CASE READC OF
               = '), ='>: NULL;
               = ',:
                  *dstrng* _ *dstrng*, '> ;
               ENDCASE
                  *dstrng* _ *dstrng*, ",>" ;
         ENDCASE
            CASE READC OF
               = '), ='>: NULL;
               ENDCASE
                  *dstrng* _ *dstrng*, '> ;
      END;
   RETURN(&dstrng);
   END.

FINISH
```

```
< NINE, DSPGEN.NLS;32, >, 28-Jul-78 15:15 SKO ;;;;;
FILE dspgen    % <ARCSUBSYS>L109  <RELNINE>dspgen % % (arcsubsys,L109,)
(RELNINE,dspgen.rel,) %
%.....Declarations.....%
   % other relevant declarations are in srecords, brecords, and bdata %
   ALLOW! % for use of bkjfn:  get it out of here and into an OSI
   procedure! %
   (partdebug) EXTERNAL CONSTANT = FALSE;  %FALSE if debugging
   daupdate%                                                          2C
   (mt) STRING = "Empty";                                             2D
   (dotstr) STRING = ".............................";
   (unknown) STRING = "<UKC>" %unknown char detected in npstrad%;     2E
    %source codes for LSRT%
      (srcnul) EXTERNAL CONSTANT = 0;                                 2F1
      (srcstat) EXTERNAL CONSTANT = 1;                                2F2
      (srcstno) EXTERNAL CONSTANT = 2;                                2F3
      (srcsig) EXTERNAL CONSTANT = 3;                                 2F4
      (srcast) EXTERNAL CONSTANT = 4;                                 2F5
      (srcdot) EXTERNAL CONSTANT = 5;                                 2F6
      (eos) CONSTANT = 800;                                           2F7
   (strtype) EXTERNAL CONSTANT = 1;                                    2G
   (lstype) EXTERNAL CONSTANT = 2;                                     2H
   % selector codes %
      (never) EXTERNAL CONSTANT = 1; %  never buggable %              2I1
      (always) EXTERNAL CONSTANT = 2; % always buggable %             2I2
      (textonscrn) EXTERNAL CONSTANT = 3; % text on screen buggable %
                                                                      2I3
   % attributes for the Front End display procedures %
      (sigatt) EXTERNAL CONSTANT = 1;                                 2J1
      (blnkatt) EXTERNAL CONSTANT = 1;                                2J2
      (snatt) EXTERNAL CONSTANT = 1;                                  2J3
      (stateatt) EXTERNAL CONSTANT = 1; % not highlighted, visible,
      non-printings invisible %                                       2J4
   % window types for the Front End display procedures %
      (rwtype) EXTERNAL CONSTANT = 1;  %randowm window%               2K1
      (swtype) EXTERNAL CONSTANT = 2;  %sequential window%            2K2
      (gwtype) EXTERNAL CONSTANT = 3;  %graphics window%              2K3
      (pwtype) EXTERNAL CONSTANT = 4;  %printer window%               2K4
   %display commands: indices to FE%
      (getwindows) EXTERNAL CONSTANT = 1; % get-windows %             2L1
      (batchdis) EXTERNAL CONSTANT = 2; % batch-display-commands %    2L2
      (crewind) EXTERNAL CONSTANT = 3; %create-window %               2L3
      (delwind) EXTERNAL CONSTANT = 4; % delete-window %              2L4
      (clrwind) EXTERNAL CONSTANT = 5; % clear-window %               2L5
      (scrwind) EXTERNAL CONSTANT = 6; % scroll-window %              2L6
      (setwindatt) EXTERNAL CONSTANT = 7; %set-window-attributes %    2L7
      (writstr) EXTERNAL CONSTANT = 8; % write-string %               2L8
      (writls) EXTERNAL CONSTANT = 9; % write-line-segment %          2L9
      (replstr) EXTERNAL CONSTANT = 10; % replace-string %           2L10
      (delstr) EXTERNAL CONSTANT = 11; % delete-string %             2L11
      (repost) EXTERNAL CONSTANT = 12; % reposition-string %         2L12
      (setstratt) EXTERNAL CONSTANT = 13; % set-string-attributes %  2L13
      (repls) EXTERNAL CONSTANT = 15; % replace-line-segment %       2L14
      (repols) EXTERNAL CONSTANT = 14; %reposition-line-segment %    2L15
      (dells) EXTERNAL CONSTANT = 16; % delete-line-segment %        2L16
      (setlsatt) EXTERNAL CONSTANT = 17; % set-line-segment-attributes
```

```
    %                                                                   2L17
    (writlit) EXTERNAL CONSTANT = 18; % write-literal %                 2L18
    (ctrack) EXTERNAL CONSTANT = 19; %turn cursor tracking on/off %
                                                                        2L19
  % generic codes for windows %
    (genfull) EXTERNAL CONSTANT = 1001;  %full screen%                  2M1
    (gentty) EXTERNAL CONSTANT = 1002;  %tty window%                    2M2
    (gencf) EXTERNAL CONSTANT = 1003;  %command feed back window%       2M3
  % data structure direction constants %
    %forward = 1, declared elsewhere %
    (backward) EXTERNAL CONSTANT = 2;                                   2N2
%....Display format control....%
  (dafrmt)  PROCEDURE (da REF, oldsw REF); %display area formatter%     3A
    %This routine generates the display image for a text display
    area, assuming the universal display.  It is driven entirely by
    the information available in the display area record (da).
    Returns the t-pointer for the next statement character after the
    last one which was displayed.  oldsw is the address of an
    existing sequence work area or zero.  If zero, then a sequence is
    opened and closed in this procedure.%
    %----------------%
    LOCAL stid, charct, sa REF, entry REF, endflg, fmtstd, lststid,
    chrsfmtd, sv1, sv2;
    REF dspcmd;  %address of commands list%
    % Null out commands list %
      IF dspcmd.L THEN #dspcmd# _ ;
    &sa _ 0;
    % Invoke catchphrases %
      INVOKE (catdafmt);
    IF NOT da.daexis THEN
      ABORT(badda, $"Fatal display error in DAFRMT");
    IF NOT da.davspec.vsdaft THEN
      BEGIN %dafrmt viewspec not set%
      % For efficiency %
        DROP (ALL);
      RETURN(endfil, 1, TRUE);
      END;
    %..INITIALIZATION....%
      %..update previous file in da - for jump routines....%
        da.dapstf _ da.dacsp.stfile;
      %..if csp is not valid, set to origin..%
        IF NOT da.daempty AND
        NOT goodrng(da.dacsp) THEN
          BEGIN
          da.dacsp.stpsid _ origin;
          da.daccnt _ 1;
          END;
      %..initialize vertical beam position..%
        da.dacrow _ 0;
      % Clear command feedback window if not yet done and not mouse
      buttons %
        IF dspccf THEN
          BEGIN  %need to clear command feedback window%
          bldclw(gencf, &dspcmd);  %build command%
          dspccf _ FALSE;  %only do clear once each call on DE%
          END;
```

```
      clearda(&da, &dspcmd);%clear the display image %
      clrall(&da, TRUE); % zero LSRT and all associated data
      structures %
   %..Display Generation..%
      IF da.daempty OR da.dacsp = endfil THEN
         BEGIN
         damt(&da, &dspcmd);
         % For efficiency %
            #dspcmd# _ ;
            DROP (ALL);
         RETURN(endfil, 1, TRUE);
         END;
      % format frozen statements if viewspec is on %
         IF da.davspec.vsfrzf THEN frzfrmt(&da, &dspcmd, 0);
      % open sequence %
         &sa _  IF &oldsw THEN &oldsw
            ELSE openseq (fmtstd _ da.dacsp, seqend(fmtstd, sv1 _
            da.davspec, sv2 _ da.davspc2), sv1, sv2, da.dausqcod,
            da.dacacode);
      charct _ da.daccnt;
      UNTIL da.dacrow > da.damrow
      OR (stid _ seqgen (&sa)) = endfil DO
         BEGIN
         chrsfmtd _ stfrmt (&da, stid, charct, sa.swclvl, sa.swslvl,
          sa.swsvw, sa.swvspec, sa.swvsp2, &dspcmd, 0 %stringid%:
         &entry);
         lststid _ stid;
         charct _ 1;
         END;
      % Build command to display viewspecs in viewspec window %
         da.davspec _ sa.swvspec;
         da.davspc2 _ sa.swvsp2;
         dspvsp(da.davspec, da.davspc2, &dspcmd);
      %process the accumulated display commands%
         prcmds(&da, &dspcmd, 0, 0);
      % If endflg is FALSE, we haven't reached the end of the
      material to be displayed  before the end of the screen. %
      endflg _
         IF stid # endfil AND da.dacrow > da.damrow THEN FALSE
         ELSE TRUE;
      % these should be returned by stfrmt %
         stid _ lststid;  %dgstid, dgstml, dgstl set by stfrmt%
         charct _ chrsfmtd + 1;
   %..termination..%
      %see if screen is really empty - could be i-viewspec didn't
      pass any statements%
         IF da.dacrow = 0  OR
         entry.stsrce = srcdot THEN
            BEGIN
            damt(&da, &dspcmd);
            stid _ endfil;
            charct _ 1;
            END;
      IF NOT &oldsw THEN closeseq (&sa:=0);
      % For efficiency %
         #dspcmd# _ ;
```

```
        DROP (ALL);
    RETURN(stid, charct, endflg);
    % Catchphrases %
      (catdafmt)  CATCHPHRASE();                                    3A14A
        BEGIN
        DISABLE (catdafmt);
        CASE SIGNALTYPE OF
          = notetype:
            CASE SIGNAL OF
              = return, = unwind :
                BEGIN
                IF NOT &oldsw AND &sa THEN closeseg (&sa:=0);
                #dspcmd# _ ;
                END;
            ENDCASE CONTINUE;
        ENDCASE CONTINUE;
        END;
    END.

(daupdate)   % CL:LB ;  selectively update display %
PROCEDURE (da REF);                                                 3B
    % Procedure description
      FUNCTION
        This routine updates the window associated with the display
        area "da". It uses stfrmt for statements that have changed
        or were not previously in the window.  It keeps the old
        string table while it is creating the new one so that it
        can check for statements that are already on the screen.
        Statements that were not previously in the window get
        placed there with write string commands.  Statements that
        were previously there but totally changed get updated with
        replace string commands.  Statements that were partially
        changed get updated with replace line segment commands.
        Statements that were unchanged in content but changed
        position get updated by reposition or scroll commands.
        Delete line segment and delete statement commands are set
        up for data that is no longer in the window.  Scrolls are
        done whereever possible.
        A list of addresses of entries in the string table is
        maintained for use in prcmds.  There is an entry in the
        string table address list for any element in the commands
        list for which there are return values from the front-end,
        i.e. write string, replace string and write line segment.
        Finally the old string table is deleted.
        If fewer than 'mincopy' lines of data remain on the screen,
        a total refresh of the screen via "dafrmt" is done.
      ARGUMENTS
        da:  displayarea address
      RESULTS
        none
      NON-STANDARD CONTROL
        none
      GLOBALS
        cdtype, cdstd1, cdstd2, cdstop, dspjpf
      %
    % Declarations %
```

```
    LOCAL
       entry REF,          %entry in old string table%
       block,       %block containing entry%
       newst REF,   %entry in new string table%
       lsptr,  %ptr to line segments list in replace string
       command %
       index,
       charct,
       sa REF,           %sequence work area%
       stid,        %current STID being considered for formatting%
       endstd,        % End stid in sequence to be formatted. %
       svstd,
       sv1,     %Viewspec word 1 saved from first sequence work
       area. %
       sv2,     %Viewspec word 2 saved from first sequence work
       area. %
       slvl,         %base level saved from first wequence work
       area%
       copycount _ 0,       %number of statements already on
       screen%
       row,  %save current row%
       prevrows,  %number of rows previously on screen%
       rsgflag,     %TRUE if copy caused sequence to be broken%
       srchentry REF,  %line segment entry in old table for srch%
       sblock,      %block containing srchentry%
       oldstrt;     %old string table%
    LOCAL LIST
       clscrolls,  %scroll commands list%
       clearwC1],  %clear window command%
       staddr;  %string table entry addresses%
    REF dspcmd;
  % Check for da existance and that terminal is display %
    IF nlmode = typewriter THEN
       err($"display attempted on typewriter terminal
       - proceed at own risk");
    IF NOT da.daexis THEN
       ABORT(badda, $"Fatal display error in DAUPDATE");
  % Check that viewspec for update of display is set%
    IF NOT da.davspec.vsdaft THEN RETURN;
  % If user sequence generator, do total refresh of screen %
    IF da.dausqcod AND da.davspec.vsusqf THEN
       BEGIN
       dafrmt(&da,0);
       RETURN;
       END;
  % if csp is not valid, set to origin and do total screen refresh%
    IF NOT da.daempty AND
    NOT goodrng(da.dacsp) THEN
       BEGIN
       da.dacsp.stpsid _ orgstid.stpsid;
       da.daccnt _ 1;
       dafrmt(&da, 0);
       RETURN;
       END;
  % Invoke catchphrases %
    &sa _ 0;
```

```
    INVOKE (catdaup);
% Null out commands list %
    IF dspcmd.L THEN #dspcmd# _ ;
%..update previous file in da, for jump routines%
    da.dapstf _ da.dacsp.stfile;
%do empty screen and return%
    IF da.daempty OR da.dacsp = endfil THEN
        BEGIN
        damt(&da, &dspcmd);
        % For efficiency %
            #dspcmd# _ ;
            #clscrolls# _ ;
            #clearw# _ ;
            #staddr# _ ;
            DROP (ALL);
        RETURN;
        END;
% Initialization %
    % keep track of number of rows prevously on screen%
        prevrows _ da.dacrow;
    %..initialize vertical beam position..%
        da.dacrow _ 0;
    % initialize pointer into old table for srch routine %
        &srchentry _ da.dastrt + dspbhl;
        sblock _ oldstrt _ da.dastrt;
        da.dastrt _ da.dacurallo _ 0;
    % initialize flags for data copied and sequence break %
        rsqflag _ FALSE;
% Take care of frozen statements %
    IF da.davspec.vsfrzf THEN
        BEGIN
        frzfrmt(&da, &dspcmd, $staddr);
        %set entry address for srch routine past frozen statements%
            UNTIL srchentry.stsrce = srcdot DO
                dgetnxt(&srchentry, sblock, forward: &srchentry,
                sblock);
        dgetnxt(&srchentry, sblock, forward: &srchentry, sblock);
        END;
% Start sequence at top of screen %
    &sa _ openseq(svstd _ da.dacsp, endstd _ seqend( svstd, sv1 _
    da.davspec, sv2 _ da.davspc2 ), sv1, sv2, da.dausqcod,
    da.dacacode);
% Display statements if there is at least one to display and the
screen is not full %
    charct _ da.daccnt;  %display of first statement may start in
    middle%
    IF da.dacrow <= da.damrow AND (stid _ seqgen (&sa)) NOT=
    endfil THEN
        BEGIN
        % Process statements up to statement where no more changes
        can occur%
            DO
                BEGIN
                %If statement was previously on screen, compare old
                ad new data  if explicit formatting check was
                requested or copy data from old table if no explicit
```

```
            request.  If statement was not previously on screen,
            format it.%
            IF rsgflag _ srch (stid, sa.swclvl, &srchentry,
            sblock: &entry, block) THEN
               BEGIN   %statement previously on screen%
               row _ da.dacrow;  %save row%
               IF stid = cdstd1 OR stid = cdstd2 THEN
                  %explicit formatting request made %
                  explct(charct, stid, &entry, block, &da, &sa,
                  &dspcmd, $staddr)
               ELSE
                  BEGIN
                  lscopy(charct, &da, &sa, &dspcmd, $staddr,
                  &entry, block);
                  IF cdtype = dspjpf THEN
                     BEGIN  %jump.  move srch pointer for
                     efficiency%
                     &srchentry _ &entry;
                     sblock _ block;
                     END
                  END;
               copycount _ copycount + da.dacrow - row;
                  %number of lines copied%
               END
            ELSE
               BEGIN
               stfrmt(&da, stid, charct, sa.swclvl, sa.swslvl,
               sa.swsvw, sa.swvspec, sa.swvsp2, &dspcmd, 0:
               &newst);
               #staddr# !_ &newst;  %save entry address%
               END;
            charct _ 1;
            END
         UNTIL da.dacrow > da.damrow OR
            stid = cdstop OR
            (stid _ seggen (&sa)) = endfil;
      % Display more statements if not end of file or screen full
      %
         IF stid NOT= endfile AND da.dacrow <= da.damrow THEN
            BEGIN
            IF rsgflag THEN  % current statment was on screen %
               %loop until all of old table has been copied or
               screen full%
                  WHILE da.dacrow <= da.damrow AND
                  dgetnxt(&entry, block, forward: &entry, block)
                  DO
                     BEGIN
                     row _ da.dacrow;
                     lscopy(charct, &da, &sa, &dspcmd, $staddr,
                     &entry, block);
                     copycount _ copycount + da.dacrow - row;
                        %number of lines copied%
                     END;
            IF da.dacrow <= da.damrow THEN %more room on screen--
            must format new statements%
               BEGIN
```

```
                IF rsqflag THEN %copy of old table broke sequence%
                   BEGIN
                   % save viewspecs from sequence work area in
                   case user seq code canged them. %
                      sv1 _ sa.swvspec;
                      sv2 _ sa.swvsp2;
                   %save base level from sequence work area in
                   case changed by impending openseq%
                      slvl _ sa.swslvl;
                   %close last sequence%
                      closeseq(&sa:=0);
                   %open new sequence at last stid copied%
                      &sa _ openseq (stmntid(&entry), endstd,
                      sv1, sv2, da.dausqcod, da.dacacode);
                      stid _ seqgen(&sa);%already formatted-throw
                      away%
                   %restore base level for this screen%
                      sa.swslvl _ slvl;
                   END;
                %do regular format on stmts now able to fit on
                screen for first time until screen full or end of
                file%
                   WHILE da.dacrow <= da.damrow AND (stid _
                   seqgen( &sa )) NOT= endfile DO
                      BEGIN
                      stfrmt(&da, stid, charct, sa.swclvl,
                      sa.swslvl, sa.swsvw, sa.swvspec, sa.swvsp2,
                      &dspcmd, 0: &newst);
                      #staddr# !_ &newst;
                      END;
                END;
             END;
       END;
% Copy viewspecs to da from seqarea %
   da.davspec _ sa.swvspec;
   da.davspc2 _ sa.swvsp2;
%if enough lines copied, do scrolls, statement repositions and
deletes%
   % If more than a minimum number old table entries was used in
   the new display, create scroll and reposition commands and
   search old table for any undeleted table entries and use list
   of addresses to decode return values from FE.  Otherwise,
   delete the new string table and refresh the entire screen via
   "dafrmt". %
   IF copycount >= prevrows/4  THEN
      BEGIN
      doscrolls(&da, &sa, oldstrt, &dspcmd, $clscrolls, $staddr);
      dodeletes(&da, oldstrt, &dspcmd, $clscrolls);
      dorepos (&da, $clscrolls);
      IF clscrolls.L THEN
         #dspcmd# _ MOVE #clscrolls#, MOVE #dspcmd#;
      % Build command to display viewspecs in viewspec window %
         dspvsp(da.davspec, da.davspc2, &dspcmd);
      prcmds(&da, &dspcmd, $staddr, 0);
      % Delete old string table %
         dlstbl(oldstrt, TRUE  %delete allocated strings%);
```

```
        % Close sequence area %
          closeseq(&sa:= 0);
        END
      ELSE %too few entries were copied, delete new string table and
      call 'dafrmt'%
        BEGIN
        % Close sequence area %
          closeseq(&sa:= 0);
        distbl(da.dastrt, TRUE);  % Delete new string table %
        da.dastrt _ oldstrt;  %restore da to point to old table%
        dafrmt(&da, 0);
        END;
   % Null out lists for efficiency %
     #dspcmd# _ ;
     #clscrolls# _ ;
     #clearw# _ ;
     #staddr# _ ;
     DROP (ALL);
   % Return %
     RETURN;
   % Catchphrases %
     (catdaup)  CATCHPHRASE();                                    3819A
        BEGIN
        DISABLE (catdaup);
        CASE SIGNALTYPE OF
          = notetype:
            CASE SIGNAL OF
              = return, = unwind :
                BEGIN
                % For efficiency %
                  #dspcmd# _ ;
                  #clscrolls# _ ;
                  #clearw# _ ;
                  #staddr# _ ;
                END;
            ENDCASE CONTINUE;
        ENDCASE CONTINUE;
        END;
   END.

(damt)     PROCEDURE (da REF, commands REF LIST); %empty da routine%
                                                                  3C
   %This routine constructs the empty message in a text
   display area%
   %------------%
   LOCAL ls REF, truncf, size, sizetr;
   LOCAL STRING string[10];
   % Initialize first string ID block %
     IF da.dastrt THEN
        clrall(&da, TRUE); %get rid of old one before getting new
        one %
     &ls _ allost( &da);
     ls.stx1 _ ls.sty _ ls.stey _ 0;
     ls.stlev _ ls.statt _ ls.stccnt _ 1;
     ls.stselector _ never;
     ls.statt _ 1;
```

```
      ls.ststid.stastr _ TRUE;
      ls.ststid.stadr _ $mt;
      ls.stsrce _ srcast;
      IF (size _ ls.stx1 + (mt.L-1)*hinc) > (sizetr _ da.daright -
      da.daleft) THEN
         BEGIN
         ls.stx2 _ sizetr;
         *string* _ *mt*[1 TO sizetr];
         truncf _ TRUE;
         END
      ELSE
         BEGIN
         ls.stx2 _ size;
         truncf _ FALSE;
         END;
   buildcom(&commands, &da, &ls, IF truncf THEN $string ELSE $mt, 0
   %strngid %);
   prcmds ( &da, &commands, 0, 0 ) ; % send commands to fe;  break
   up returned list, put results (ids) in data structures. %
   RETURN;
   END.

(trzfrmt)  PROCEDURE (da REF, commands REF LIST, staddr REF LIST);
%display frozen statements%                                          3D
   %This routine formats the frozen statements for a
   display area%
   %------------------%
   LOCAL  ls REF, fz REF, davs1, davs2, sa REF, stid, rt REF, size,
   sizetr, truncf, newst REF;
   LOCAL STRING string[100];
   davs1 _ da.davspec; %save view specs from display area record%
   davs2 _ da.davspc2;
   IF &fz _ da.dafrz1 THEN %there are frozen statements%
      DO
         IF fz.fzexis THEN
            BEGIN  %statement frozen...display it%
            % set up a sequence that will only return the first
            statement if it passes -- by setting the L viewspec to E
            + 0 %
               davs1 _ setlt('e, fz.fzvspec, fz.fzvspc2 : davs2);
            &sa _ openseq (fz.fzstid, fz.fzstid, davs1, davs2,
            da.dausqcod, da.dacacode);
            UNTIL (stid _ seqgen (&sa)) = endfil DO
               BEGIN
               stfrmt(&da, stid, 1, sa.swclvl, sa.swslvl, sa.swsvw,
               sa.swvspec, sa.swvsp2, &commards, 0 %stringid%:
               &newst);
               IF &staddr THEN  %partial update%
                  #staddr# !_ &newst;  % prcmds needs addr%
               END;
            closeseq (&sa);
            IF da.dacrow > da.damrow THEN RETURN; %filled display
            area%
            END
         ELSE err($"NLS error: Frozen statement list incorrect
         Turn off viewspec o")
```

```
      UNTIL (&fz _ fz.fznext) = 0;
   %display dotted line%
      % Initialize first string ID block %
         &ls _ allost( &da);
         ls.stx1 _ 0;
         ls.sty _ ls.stey _ da.dacrow := da.dacrow + vinc;
         ls.stlev _ ls.statt _ ls.stccnt _ 1;
         ls.stselector _ never;
         ls.statt _ 1;
         ls.ststid.stastr _ TRUE;
         ls.ststid.stadr _ $dotstr;
         ls.stsrce _ srcast;
         IF (size _ ls.stx1 + (dotstr.L-1)*hinc) > (sizetr _
         da.daright - da.daleft) THEN
            BEGIN
            ls.stx2 _ sizetr;
            *string* _ *dotstr*[1 TO sizetr];
            truncf _ TRUE;
            END
         ELSE
            BEGIN
            ls.stx2 _ size;
            truncf _ FALSE;
            END;
      ls.stsrce _ srcdot;
   buildcom(&commands, &da, &ls, IF truncf THEN $string ELSE
   $dotstr, 0 %strngid%);
   IF &staddr THEN  %partial update%
      #staddr# !_ &ls;  %  prcmds needs addr%
   RETURN;
   END.

%....Display format support....%
   (stfrmt)   % LB: ;  format a statement %
   PROCEDURE (da REF, stid, charct, level, origlev,  stvec, viewspec1,
   vwspc2, commands REF LIST, stringid % => [meta-res] chrsfmtd,
   &entry, &block %);                                                    4A
      % Procedure description
         FUNCTION
            Given an STID and a DISPLAY AREA ADDRESS, STFRMT
           "formats" the corresponding text into the space available
           on the window following line truncation viewspecs and
           indenting by level.  (Level truncation has been handled in
           the calling procedure.)  Statement numbers, signatures, and
           blank lines are also created as per viewspec request.
           Sets up commands for the FE if 'stringid' is zero.  A
           non-zero 'stringid' indicates that the statement is already
           on the screen and the line segments must be checked to
           determine which data must be re-displayed.
           none
           To format a statement, stfrmt loops over the  lines in the
           text (determined by the da record specified width,
           (remaining) height, and user option specified break
           positions (wrap column and virtual display wrap) and line
           segments in each line by calling LSGFRMT.  STFRMT builds a
           string record for the statement and an associated list of
```

```
      line segment records.  LSGFRMT formats single line segments
      (ie, creates a line segment record) breaking at an
      appropriate place dependent on types of characters read
      and:
         user specified wrap column
         virtual display rightmost column
         hardware/display window right column
      Commands to be sent to the front end to "write" this
      statement are appended to the commands list passed.
      This routine formats a statement according to the
      information in the text display area record, da, the level
      of the statement, level, and the statement vector work
      area, stvec.  The formating begins at the character
      indicated by charct.
      NOTE:  special handling of indentation--if indent=OFF and
      branch/plex only, integrity of structure maintained
      viewspec1 is first word of viewspecs, as obtained from
      seggen
   ARGUMENTS
      da:  address of display area
      stid:  stid of statement
      charct:  last formatted character in statement
      level:  level of statement
      origlev:  level of statement on top of screen
      stvec:  statement vector work area
      viewspec1:  first viewspec word
      vwspc2:  second viewspec word
      commands:  address of commands list for FE commands
      stringid:  0 if statement is not on screen
         FE string id if statement is on screen
   RESULTS
      chrsfmtd: the last formatted character count.
      entry:  address of string table entry for this statement
      block:  address of block of string table containing 'entry'
   NON-STANDARD CONTROL
      Go to 'err' with message "Bad File" if 'lodprop' fails.
   GLOBALS
      none
   %
% Declarations %
   LOCAL
      maxchar, % maximum characters in the statement to be
      formatted. %
      wrpcnt, % number of times this "line" has been wrapped
      around %
      chrsfmtd, % number of characters which HAVE BEEN formatted
      %
      numberadr REF, %temp for stmt # address%
      stringadr REF, % used for signature %
      endcon,   %return from lsgfrmt: reason for ending line
      segment%
      ebptr,
      numflg,
      leftsno, % If sno on right is on a separate line, this
      local has the left boundary column.  It is used to check if
      the signature will fit on that same line. %
```

```
        sdbadr REF,
        trunc,   %current line for truncation viewspec%
        indent,  %left margin offset%
        st REF,  % string segment address %
        ls REF,  % line segment address%
        entry REF,
        block REF,
        blnklnf; %flag to note blank line already out%
    % Return if gross error-- illegal display area %
      IF NOT da.daexis THEN
        err($"Fatal display error in STFRMT");
      IF stid = endfil THEN RETURN(0, 0, 0);
    % General initialization %
      % get text if this is a statement; set maximum character
      length %
        IF NOT stid.stastr THEN
          BEGIN
           IF NOT lodprop(stid, txttyp : &sdbadr) THEN
             err ($"Bad file -- stfrmt");
          maxchar _ sdbadr.schars;
          END
        ELSE maxchar _ [stid.stadr].L;
      %initialize general variables%
        trunc _ viewspec1.vstrnc; % truncation counter %
        blnklnf _ FALSE; % TRUE if stmt # at right on separate
        line%
        wrpcnt _ 0; % number of times we have wrapped around for
        this statement-- initialized to be zero %
      %indentation%
        IF (viewspec1.vsrind AND ( viewspec1.vsbrof OR
        viewspec1.vsplxf)) THEN  %relative identing%
          indent _ MAX (0, MIN (da.daind * (level-origlev),
          da.damind))
        ELSE IF NOT viewspec1.vsindf THEN  %no identing%
          indent _ 0
        ELSE  %identing according to statement level only%
          indent _ MAX (0, MIN (da.daind * (level-1), da.damind));
      % set up count to first character to be formatted, skipping
      over name if necessary:  the starting character %
        IF charct <= 1 THEN
          BEGIN
          charct _ 1;
          IF NOT viewspec1.vsnamf THEN %do not show statement
          name%
            IF NOT stid.stastr THEN %a-string%
              charct _ sdbadr.sname;%skip over name%
          END;
        chrsfmtd _ charct - 1; % the number of characters which
        HAVE BEEN formatted %
      % get the next string element for this display area (or first
      one if necessary);  get the first line segment for that string
      element:  initialize them.  %
        &entry _ &ls _ &st _ allost( &da );
        &block _ da.dacurallo;
      % Initialize additional first fields for line segment/string
      element: statement dependent fields will be initialized after
```

```
    statement number code below %
       ls.stx1 _ ls.stx2 _ st.stex _ da.daccol _ indent;
       ls.sty _ st.stey _ da.dacrow;
       ls.stlev _ level;
       ls.stcomplete _ TRUE;  %FALSE if statement doesn't fit%
 % Format statement number: put it out if it is on the right.
 (Putting it out implies stuffing a command on the batch commands
 list and closing off the corresponding line/string segments.  If
 this is done, we must get a new string and initialize it before
 proceeding to the body of the statement.)  If statement numbers
 go on the right, simply collect the string now and set a flag
 which will be tested later. %
    numflg _ FALSE; % number not on right %
    &numberadr _ 0;
    IF NOT stid.stastr AND viewspec1.vsstnf AND
    (stid.stpsid NOT= origin OR viewspec1.vssidf) THEN
       BEGIN  %statement number must be formatted %
       &numberadr _ getstring(30, $dspblk);
       IF viewspec1.vssidf  % display line #'s as sids %
          THEN
             *numberadr* _ '0, STRING( getsid(stid) )
          ELSE
             fechnm(stvec, &numberadr);
                %convert statement vector to string%
       IF NOT viewspec1.vsstnr THEN
          BEGIN %statement number on left%
          % fill out rest of string/line segment record %
             ls.ststid _ &numberadr;
             ls.stccnt _ 1;
             ls.stbps _ chbmty + &numberadr;
             ls.stsrce _ srcstno;
             ls.stselector _ never;
             ls.statt _ snatt;
             % Truncate statement number if it is too long %
                IF da.damcol - ls.stx1 < numberadr.L THEN
                   BEGIN
                   numberadr.L _ da.damcol - ls.stx1;
                   *numberadr*[numberadr.L] _ '!;
                   %to note that some of stmt # is lost%
                   END;
             ls.stbpe _ chbptr(numberadr.L) + &numberadr;
             ls.stx2 _ st.stex _ ls.stx1 + (numberadr.L-1);
          % put the command on the command list. %
             IF NOT stringid THEN
                buildcom( &commands, &da, &ls, &numberadr,
                stringid);
          % increment columns to offset statement numbers;
          otherwise, increment line. %
             IF ls.stx2 + 2 < da.damcol THEN % we increment by 2
             to have s-nums offset. %
                da.daccol _  ls.stx2 + 2 * hinc
             ELSE
                BEGIN
                da.dacrow _ da.dacrow + vinc; % increment line by
                1 %
                da.daccol _ indent;
```

```
                END;
            % get next line (string) segment.  %
               &ls _ allols( &st );
               ls.stx1 _ ls.stx2 _ da.daccol;
               ls.sty _ st.stey _ da.dacrow;
               ls.stlev _ level;
            END
         ELSE %numbers on right%
            numflg _ TRUE;
         END;
   % Pre-statement body line segment initialization %
      %set up byte pointers for reading characters from string or
      statement%
         IF NOT stid.stastr THEN
            ls.stbps _ ls.stbpe _ stbptr(charct - 1) + &sdbadr +
            sdbhdl
         ELSE
            ls.stbps _ ls.stbpe _ chbptr(charct - 1) + stid.stadr;
   % Format statement body %
      endcon _ FALSE;
      LOOP % over lines for the statement --
      UNTIL trunc <= 0 OR da.dacrow > da.damrow %
         BEGIN %once per line%
         %format a line%
            LOOP % over line segments in a line %
               BEGIN
               ls.ststid _ stid;
               ls.stccnt _ chrsfmtd+1;
               ls.stsrce _ srcstat;
               ls.stselector _ textonscrn;
               ls.statt _ stateatt;
               % format the line segment %
                  endcon _ lsgfrmt(&da, &ls, &commands, maxchar,
                  chrsfmtd, wrpcnt, stringid: chrsfmtd, wrpcnt,
                  ebptr );
                     % ls.stbpe and da.daccol updated %
                  ls.stx2 _  MAX( 0, da.daccol-hinc );
               IF endcon = EOL OR endcon = ENDCHR THEN EXIT LOOP;
                  % Done with this line %
               % Get and initialize next line segment for this line
               %
                  &ls _ allols( &st );
                  ls.stbps _ ls.stbpe _ ebptr; % The end of the last
                  line segment is the beginning of this one. %
                  ls.stx1 _ ls.stx2 _ da.daccol;
                  ls.sty _ da.dacrow;
                  ls.stlev _ level;
               END;
         IF endcon = ENDCHR OR
         (trunc _ trunc - 1) <= 0 OR
         (da.dacrow _ da.dacrow + vinc) > da.damrow THEN
            EXIT LOOP; % over statement text %
         % Initialize x and y coordinates for next line %
            &ls _ allols( &st );
            % initialize indentation for the line: x position at
            start %
```

```
            ls.stbps _ ls.stbpe _ ebptr; % The end of the last
            line segment is the beginning of this one. %
            ls.stx1 _ ls.stx2 _ da.daccol _ indent;
            ls.sty _ st.stey _ da.dacrow;
            ls.stlev _ level;
         END; % Of statement text format %
      % Put out ending coordinates of string %
         st.stex _ ls.stx2;
         st.stey _ ls.sty;
   % Set statement complete flag to FALSE if statement did not fit
   on screen%
      IF endcon NOT= ENDCHR THEN entry.stcomplete _ FALSE;
   % Put out statement numbers on the right %
      IF numflg THEN % statement number to be put out on right %
         BEGIN
         IF (da.dacrow <= da.damrow)
            % if the end of screen has intervened; free the number
            string and return %
         AND &numberadr
            % There really is a string %
         AND NOT (da.daccol > da.damcol - hinc*(numberadr.L+1) AND
         da.dacrow = da.damrow)
            % There is room on this line for the number or, if not,
            there is room for a new line. %
         THEN
            BEGIN %statement # on right at end of stmt unless both
            line and screen full%
            % get new string segment for statement number and put it
            out on the right. %
               &ls _ allols(&st);
               ls.stlev _ level;
               % fill out rest of string/line segment record %
                  ls.ststid _ &numberadr;
                  ls.stccnt _ 1;
                  ls.stbps _ chbmty + &numberadr;
                  ls.stsrce _ srcstno;
                  ls.stselector _ never;
                  ls.statt _ snatt;
                  % Truncate statement number if it is too long %
                     IF da.damcol < numberadr.L THEN
                        BEGIN
                        % Truncate statement if it will not fit in
                        the window width.  It will go on a separate
                        line flush right if there is not enough room
                        on this one. %
                        numberadr.L _ da.damcol;
                        *numberadr*[numberadr.L] _ "!;
                        %to note that some of stmt # is lost%
                        END;
                  % If statement number won't fit on this line,
                  advance one line %
                     IF (da.damcol-da.daccol) < numberadr.L THEN
                        BEGIN  %have to put number on separate line%
                        da.dacrow _ da.dacrow + vinc;
                        blnklnf _ TRUE;
                        END
```

```
                    ELSE blnklnf _ FALSE;
                ls.sty _ st.stey _ da.dacrow;
                ls.stbpe _ chbptr(numberadr.L) + &numberadr;
                % Make statement number flush right. %
                    ls.stx2 _ st.stex _ da.damcol;
                    ls.stx1 _ MAX(ls.stx2 - (numberadr.L -1), 0);
                leftsno _ IF blnklnf THEN ls.stx1 ELSE da.damcol;
              % put the command on the command list. %
                IF NOT stringid THEN
                    buildcom( &commands, &da, &ls, &numberadr,
                    stringid);
            END
        ELSE % There was a statement number string allocated, but
        it was not put out.  It should be deallocated since it will
        be deallocated automatically only if a string segment
        pointing to it is freed. %
            BEGIN
            IF &numberadr THEN freestring( &numberadr:= 0, $dspblk
            );
            da.dacrow _ da.dacrow + vinc;
            entry.stcomplete _ FALSE;  %statement not complete%
            RETURN(chrsfmtd, &entry, &block);
            END;
        END;
    % Put out signatures and/or blank lines %
      IF viewspecl.vsidtf AND NOT stid.stastr AND da.dacrow <=
      da.damrow  THEN
        BEGIN
        % get the signature %
            &stringadr _ getstring( 30, $dspblk);
            fechsig(stid, &stringadr);
        % If this is the last line in the window and the signature
        must go on the next line, free the signature string and
        return %
            IF da.dacrow = da.damrow AND
            ( (blnklnf AND (leftsno -  (stringadr.L + 3)*hinc) <
            indent) OR (NOT blnklnf) ) THEN
                BEGIN
                freestring(&stringadr, $dspblk);
                da.dacrow _ da.dacrow + vinc;
                entry.stcomplete _ FALSE;  %statement not complete%
                RETURN(chrsfmtd, &entry, &block);
                END;
        % get and initialize string %
            &ls _ allols(&st);
            ls.stlev _ level;
            ls.ststid _ &stringadr;
            ls.stccnt _ 1;
            ls.stbps _ chbmty + &stringadr;
            ls.stsrce _ srcsig;
            ls.stselector _ never;
            ls.statt _ sigatt;
        % increment line if necessary %
            IF blnklnf THEN
                BEGIN
                IF (leftsno - (stringadr.L + 3)*hinc < indent) THEN
```

```
                BEGIN  %signature won't fit on line%
                da.dacrow _ da.dacrow + vinc;
                ls.stx2 _ st.stex _ da.damcol;
                END
             ELSE
                ls.stx2 _ leftsno - 3 * hinc;
             END
          ELSE
             BEGIN
             ls.stx2 _ st.stex _ da.damcol;
             blnklnf _ TRUE;
             da.dacrow _ da.dacrow + vinc;
             END;
          ls.sty _ st.stey _ da.dacrow;
       % Truncate string if necessary; set left x coordinate %
          IF ls.stx2 < stringadr.L THEN
             BEGIN
             stringadr.L _ ls.stx2;
             *stringadr*[stringadr.L] _ "!;
             entry.stcomplete _ FALSE;  %statement not complete%
             END;
          ls.stx1 _ MAX(ls.stx2-(stringadr.L-1), 0);
          ls.stbpe _ chbptr(stringadr.L) + &stringadr;
       %Put it out %
          IF NOT stringid THEN
             buildcom( &commands, &da, &ls, &stringadr, stringid);
          END;
       IF viewspec1.vsblkf AND NOT blnklnf AND da.dacrow < da.damrow
       THEN
          BEGIN
          % Put out blank line segment %
             &ls _ allols( &st );
             ls.stlev _ level;
             ls.ststid _ ls.stccnt _ 0;
             ls.stbps _ ls.stbpe _ chbmty;
             ls.stsrce _ srcnul;
             ls.statt _ blnkatt;
             ls.stx1 _ ls.stx2 _ st.stex _ indent;
             ls.sty _ st.stey _ da.dacrow _ da.dacrow + vinc;
             IF NOT stringid THEN
                buildcom( &commands, &da, &ls, $"", stringid);
          END
       ELSE IF viewspec1.vsblkf AND NOT blnklnf THEN
          entry.stcomplete _ FALSE;  %statement not complete%
    % Update current row %
       da.dacrow _ da.dacrow + vinc;
    % Return %
       RETURN(chrsfmtd, &entry, &block);
    END.

(lsgfrmt)   % CL: ;  format a line segment %
PROCEDURE (da REF, ls REF, commands REF LIST, maxchar, chrsfmtd,
wrpcnt, stringid % => endcon, chrsfmtd, wrpcnt, bptre %);          4B
    % procedure description
       FUNCTION
          Format 1 line segment and build commands to send to FE if
```

```
      statement is not on the screen (strngid = 0).
      Reasons for terminating a line segment:
         end of statement: RETURN(ENDCHR),
         end of line: RETURN(EOL),
         end of line segment: RETURN(eos),
      none
   ARGUMENTS
      da REF:  address of display area
      ls REF:  address of line segment -has stid, charct, stsrce,
      x1, y, stev fields set
      commands REF LIST:  address of batch commands
      maxchar:  Length of statement or string
      chrsfmtd:  number of characters which have been formatted
      so far.
      wrpcnt:  number of times we have wrapped around for this
      line
      stringid:  0 for write, non-zero for replace
   RESULTS
      endcon:  reason for terminating line segment
         values:  ENDCHR (end of statement), EOL (end of line)
         or eos (special character  e.g. tab, non-printing,
         escape)
      chrsfmtd:  number of characters which have been formatted
      so far
      Wrpcnt:  number of times we have wrapped around for this
      line
      bptre:  byte pointer to end of line segment
   NON-STANDARD CONTROL
      none
   GLOBALS
      none
   %
% Declarations %
   LOCAL
      strlength,  % length of text of line segment %
      char, % The most recently read character: pointed to by
      bptr %
      col, % current physical column being formatted %
      charcol, % column to which TAB and NP strings will take the
      segment. %
      x2lim, % maximum columns which may be formatted in the
      remaining virtual line %
      wwidth, % maximum width of the physical display window %
      wdthflag,  % flag used in determining truncation of TAB or
      NP strings. %
      endcon, % reason for ending segment: eos, ENDCHR or EOL %
      bptr, % starting bytepointer for this line segment %
      firstchar,
      firstseg,
      frmtdorig, % number of characters formatted before this
      call  %
      tabsiz,
      gaptr,
      gapcol,
      wrptr,
      wrapgap,
```

```
        wrapcol;
      LOCAL STRING strngspcef100];  %used for nonprintables and
      tabs%
    % Initialize counts and pointers %
      x2lim _ da.damcol - (wrapcol _ udpwrapcol*wrpcnt);
         % The maximum which may be formatted in the remaining
         virtual line %
      wwidth _ da.daright - da.daleft;
         % The maximum width of the window %
      frmtdorig _ chrsfmtd;
      gaptr _ wrptr _ bptr _ ls.stbps;
      gapcol _ wrapgap _ 0;
      col _ da.daccol;
      firstchar _ TRUE;
    % Format the line segment - up to virtual line width %
      DO
         BEGIN
         IF (chrsfmtd _ chrsfmtd + 1) > maxchar THEN
            BEGIN
            IF col = ls.stx1 THEN
               BEGIN %This line segment is empty%
               % A null segment as the only segment in the statement
               should display aas a space so the statement ma be
               seleceted.  If there are other segments in the
               statement, display as an actual NULL in the FE.  In
               both cases the stsrce type in the backend record is
               srcnul in case we want to know. %
                  ls.stsrce _ srcnul;
                  firstseg _ ls.stflag;
               % Really put out the "null" segment. %
                  IF NOT stringid THEN
                     buildcom( &commands, &da, &ls,  IF firstseg
                     THEN $" " ELSE $"", stringid );
                  da.daccol _ col + (IF firstseg THEN hinc ELSE 0);
                  % increment for space %
               RETURN( ENDCHR, chrsfmtd-1, wrpcnt, ls.stbpe _ bptr
               );
               END;
            % put out last line segment in the statement. %
               IF NOT stringid THEN
                  buildcom(&commands, &da, &ls, buildstring(
                  ls.stbps, bptr: strlength ), stringid);
            % Advance relevant fields %
               da.daccol _ col;
            RETURN(ENDCHR, chrsfmtd-1, wrpcnt, ls.stbpe _ bptr);
            END;
         CASE  char _ ^bptr OF %get next character from statement%
            IN [*1,*"]:  NULL;
               %punctuation, lower case alpha, upper
               case alpha, numbers%
            =SP:
               BEGIN %a gap character--note its position for line
               break%
               IF (x2lim = wwidth AND col >= x2lim) THEN EXIT; % we
               have reached the limit of the window %
                  % otherwise keep going.  Extraneous spaces "bleed"
```

```
            off end. %
         % set up backup state info %
            gaptr _ bptr;
            gapcol _ col;
         END;
      =CR, =EOL:
         BEGIN
         % Even if there is no other character after the EOL,
         a new line segment must be generated to create the
         blank line! %
         % put out line segment.  (In this case a new line
         segment at the end is legitimate.) %
            IF NOT stringid THEN
               buildcom( &commands, &da, &ls, buildstring(
               ls.stbps, bptr: strlength), stringid);
            da.daccol _ col+hinc;
            RETURN(EOL, chrsfmtd, wrpcnt _ 0, ls.stbpe _ bptr
            );
         END;
      ENDCASE %=TAB, =CA,=0, =LF, =CD, =BC, =BW, =$ascalt, IN
      [1B,32B], IN [34B,36B]:% %a TAB or non-printing
      character%
         BEGIN
         % If this is the first character read in this call on
         lsgfrmt, this nonprinting is the ONLY character in
         the line segment.  If other characters have been
         read, terminate the line segment (with EOS) before
         the NP, return to stfrmt to start up a new line
         segment.  The next time through, the NP will be put
         out. %
         IF firstchar THEN
            BEGIN
            % Get string representation into STRNGSPCE %
               IF char = TAB THEN
                  BEGIN
                  tabsiz _ fndtab(&da, col/hinc + wrapcol) *
                  hinc - wrapcol - hinc - col;
                     % subtract hinc otf because fndtab is one
                     column greater than da.daccol
                     coordinates. %
                  buildtab( $strngspce, tabsiz );
                  END
               ELSE
                  BEGIN
                  *strngspce* _ *[ npstrad(char) ]*;
                  END;
            % Set end column %
               charcol _ strngspce.L*hinc + col;
            % Truncate string if necessary %
               wdthflag _ FALSE;
               IF ( wdthflag _ (charcol > x2lim) ) OR
               (udpwrapcol>0 AND charcol>udpwrapcol) THEN
                  BEGIN
                  charcol _ IF wdthflag THEN wwidth ELSE
                  udpwrapcol;
                  strngspce.L _ charcol - col;
```

```
                END;
            %Build commands; set line segment buggable as
            single character %
               ls.stcbug _ FALSE;
               IF NOT stringid THEN
                  buildcom( &commands, &da, &ls, $strngspce,
                  stringid);
            % Determine end condition %
               IF chrsfmtd + 1 > maxchar THEN
                  endcon _ ENDCHR
               ELSE IF charcol = wwidth OR charcol >= x2lim
               THEN
                  BEGIN
                  wrpcnt _ 0;
                  endcon _ EOL
                  END
               ELSE IF (udpwrapcol > 0 AND charcol >=
               udpwrapcol) THEN
                  BEGIN
                  wrpcnt _ wrpcnt + 1;
                  endcon _ EOL;
                  END
               ELSE endcon _ eos;
            % Set DACCOL and RETURN %
               da.daccol _ charcol;
               RETURN( endcon, chrsfmtd, wrpcnt, ls.stbpe _
               bptr);
            END
         ELSE
            BEGIN
            % Back up one character; put out "current" line
            segment.  The NP will be handled next time
            through. %
               IF SKIP !bkjfn( bptr ) THEN
                  ls.stbpe _ R1
               ELSE ls.stbpe _ bptr;  % should be error! %
               IF NOT stringid THEN
                  buildcom( &commands, &da, &ls, buildstring(
                  ls.stbps, ls.stbpe: strlength), stringid);
               da.daccol _ col; % to back up one character %
               % always eos: we wold not be here in any other
               case %
               RETURN(eos, chrsfmtd-1, wrpcnt, ls.stbpe );
            END;
         END;
      IF (col _ col + hinc) > udpwrapcol AND udpwrapcol > 0 THEN
         BEGIN
         wrptr _ bptr;
         wrapgap _ col;
         wrpcnt _ wrpcnt + 1;
         EXIT LOOP;
         END;
      firstchar _ FALSE;
      END
   UNTIL col > x2lim % OR col > udpwrapcol %;
% We get here only if the virtual line width has been exceeded or
```

```
if the gap column has been reached.  We must permit spaces and
TABs and EOLs to bleed into the extra space permitted on the
terminal. %
   IF chrsfmtd = maxchar OR (gapcol = 0 AND wrapgap = 0) THEN
      BEGIN %no gap char found or at last character%
      % Break in the middle of the text:  make the current
      character a gap. %
      IF NOT stringid THEN
         buildcom( &commands, &da, &ls, buildstring( ls.stbps,
         bptr : strlength), stringid);
      da.daccol _ col;
      endcon _ IF chrsfmtd + 1 > maxchar THEN ENDCHR ELSE EOL;
      RETURN(endcon, chrsfmtd, wrpcnt _ 0, ls.stbpe _ bptr );
      END
   ELSE
      BEGIN
      % There was a backup character found and we  will back up
      to that point before putting the line segment out.   If
      there is a group of invisibles following that backup point
      and they will fit in the slack space between x2lim and
      wwidth, we will advance the breaking point as far as we
      can. %
      IF ((gapcol = x2lim AND wrapgap = 0) AND (char = SP AND
      chrsfmtd < maxchar)) THEN
         BEGIN
         WHILE (x2lim _ x2lim + hinc) < MIN(wwidth, IF udpwrapcol
         THEN udpwrapcol ELSE 1000) DO
            BEGIN
            IF (chrsfmtd _ chrsfmtd + 1) > maxchar THEN
               BEGIN
               % put out last line segment in the statement. %
                  IF NOT stringid THEN
                     buildcom(&commands, &da, &ls, buildstring(
                     ls.stbps, bptr: strlength ), stringid);
               % Advance relevant fields %
                  da.daccol _ col + hinc;
               RETURN(ENDCHR, chrsfmtd-1, wrpcnt, ls.stbpe _
               bptr);
               END;
            CASE char _ ^bptr OF
               =SP:
                  BEGIN
                  gapcol _ col _ col + hinc;
                  gaptr _ bptr;
                  END;
               =EOL, =CR:
                  BEGIN
                  % put out line segment.  (In this case a new
                  line segment at the end is legitimate.) %
                     IF NOT stringid THEN
                        buildcom( &commands, &da, &ls,
                        buildstring( ls.stbps, bptr: strlength),
                        stringid);
                     da.daccol _ col + hinc;
                     RETURN(EOL, chrsfmtd, wrpcnt _ 0, ls.stbpe _
                     bptr );
```

```
                    END;
                 ENDCASE
                    BEGIN
                    % send out the line segment; end the line
                    before the current character %
                       IF SKIP !bkjfn( bptr ) THEN % backup one
                       character %
                          ls.stbpe _ P1
                       ELSE ls.stbpe _ bptr;  % should be error! %
                       IF NOT stringid THEN
                          buildcom( &commands, &da, &ls,
                          buildstring( ls.stbps, ls.stbpe:
                          strlength), stringid);
                       da.daccol _ col;
                       RETURN(EOL, chrsfmtd -1, wrpcnt _ 0,
                       ls.stbpe );
                    END;
              END;
           wrpcnt _ 0;
           END;
        IF (wrapgap AND (wrapgap < gapcol OR gapcol = 0)) THEN
           BEGIN
           IF NOT stringid THEN
              buildcom( &commands, &da, &ls, buildstring( ls.stbps,
              wrptr: strlength), stringid)
           ELSE strlength _ slngth(ls.stbps, wrptr);
           da.daccol _ wrapgap+hinc;
           endcon _ IF (chrsfmtd _ frmtdorig + strlength) + 1 >
           maxchar THEN
              ENDCHR ELSE EOL;
           RETURN(endcon, chrsfmtd, wrpcnt _ wrpcnt + 1, ls.stbpe _
           wrptr );
           END;
        % the window has been filled; we are through with this
        line, so reset wrpcnt.  We have bled all spaces into the
        extra space between x2lim and wwidth that we could. %
           IF NOT stringid THEN
              buildcom( &commands, &da, &ls, buildstring( ls.stbps,
               gaptr: strlength), stringid)
           ELSE strlength _ slngth(ls.stbps, gaptr);
           da.daccol _ gapcol+hinc;
           endcon _ IF (chrsfmtd _ frmtdorig + strlength) + 1 >
           maxchar THEN
              ENDCHR ELSE EOL;
           RETURN(endcon, chrsfmtd, wrpcnt _ 0, ls.stbpe _ gaptr );
        END;
   % Return %
      RETURN;  %returns done above %
   END.

(buildstring)   % LB: ;  build a string in the list zone %
PROCEDURE (bpts, bpte % => string %);                                4C
   % Procedure description
      FUNCTION
         Allocate a string in the list zone and build a string in
         it.  This procedure assumes that the allocated string will
```

```
        be used in a LIST and consequently freed when the LIST is
        nulled out.
      ARGUMENTS
        bpts:  byte pointer to beginning of text
        bpte:  byte pointer to end of text
      RESULTS
        string:  address of allocated string
        count:  length of string
      NON-STANDARD CONTROL
        none
      GLOBALS
        lstzone:  references
      %
   % Declarations %
      LOCAL coptr, count, string REF;
   % Allocate string %
      &string _ getstring(slngth(bpts, bpte), lstzone);
   % Build string %
      coptr _ chbmty + &string;
      count _ 0;
      UNTIL bpts = bpte DO
        BEGIN
        BUMP count;
        ^coptr _ ^bpts;
        END;
      string.L _ count;
   % Return %
      RETURN(&string, count);
   END.

(buildtab) PROCEDURE                                                  4D
   (string REF,
   size);
   IF size < 1 THEN size _ 1;
   FOR size DOWN UNTIL <= 1 DO
      *string* _ *string*, SP;
   RETURN;
   END.

(allost)   PROCEDURE (da REF);  %allocate next string for this da or
the first one if none has been allocated yet%                         4E
   % not initialized here:
      ststid
      stcoord
      stringid
      stlev
      stsrce
   %
   %-----------------%
   (st) REF;                                                          4E4
   &st _ dnxt( IF da.dacurallo THEN da.dacurallo
      ELSE dbnew( &da, 0, strtype) );
   % default initalization %
      st.stcurallo _ st.stlsid _ st.stlsptr _ st.strngid _ 0;
      st.stwid _ da.dawid;
      % Set default flags to be TRUE %
```

```
        st.stexis _
        st.stnew _
        st.stflag _ % The first line segment in the string %
        st.stcbug _ TRUE; % Each character is buggable individually
        %
    % selector and attributes must be filled in by caller %
  RETURN(&st);
  END.

(allols)   PROCEDURE (st REF); %allocate next line segment for this
string or the first one if none has been allocated yet%            4F
  % not initialized here
    rtbps
    rtbpe
    rtstid
    rtx1
    rtx2
    rty
    rtsid
    rtlev
    rrtccnt
    rtlsid
    rtsrce
  %
  %-----------------%
  (ls) REF;                                                        4F4
  &ls _ dnxt( IF st.stcurallo THEN st.stcurallo
    ELSE dbnew( &st, 0, lstype) );
  % default initialization %
    ls.stwid _ st.stwid;
    ls.stexis _ ls.stnew _ ls.stcbug _ TRUE;
    ls.stflag _ FALSE;
    % selector and attributes must be filled in by caller %
  RETURN(&ls);
  END.

(npstrad) PROCEDURE (npchar);%get symbol for non-printing%         4G
  %npstrad accepts a non-printing character code, npchar, as input,
  and returns the address of the appropriate global string used to
  represent that character in the display%
  LOCAL index;  %for matching npcodes and npstrings%
    FOR index _ 0 UP 1 UNTIL >= npsize DO
      IF npcodes[index] = npchar THEN RETURN (npstrings[index]);
    *ukcstr*[1] _ npchar;
    RETURN ($ukcstr);
  END.

(dspvsp)   % LB: ;  display viewspecs %
PROCEDURE (vspec1, vspec2, commands REF);                          4H
  % Procedure description
    FUNCTION
      Check the parameters against the viewspecs previously
      displayed in the viewspec window.  If the viewspecs have
      changed, update the global viewspec words and append to the
      commands list a write-literal command to display the
      viewspecs in the viewspec window.
```

```
      ARGUMENTS
         vspec1:  first viewspec word
         vspec2:  second viewspec word
         commands:  address of the commands list
      RESULTS
         none
      NON-STANDARD CONTROL
         none
      GLOBALS
         vspsav,vspsav[1]
      %
   % Declarations %
      LOCAL firstd, secndd;
      LOCAL STRING vspstr[16];
      REF vswndw;
   % Check against previously displayed viewspecs %
      IF &vswndw AND ((vspec1 NOT= vspsav) OR (vspec2 NOT=
      vspsav[1])) THEN
         BEGIN  %viewspecs have changed%
         (vspsav,vspsav[1]) _ (vspec1,vspec2);
         bldvsstr(vspec1, vspec2, $vspstr);
         % Build write-literal command %
            bldwli(&vswndw, $vspstr, &commands);
         END;
   % Return %
      RETURN;
   END.

(bldvsstr)   % LB: ;  build viewspec string %
PROCEDURE (vspec1, vspec2, vspstr REF);                          4I
   % Procedure description
      FUNCTION
         Compose a character string to display the specified
         viewspecs in the viewspecs window.
      ARGUMENTS
         vspec1:  first viewspec word
         vspec2:  second viewspec word
         vspstr:  address of the character string
      RESULTS
         none
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %
   LOCAL firstd, secndd;
   % Set up level and lines part of viewspec string %
      IF vspec1.vsrlev THEN
         BEGIN
         *vspstr* _ "R;
         IF vspec1.vslevd THEN *vspstr* _ *vspstr*, "-
         ELSE *vspstr* _ *vspstr*, "+;
         *vspstr* _ *vspstr*, STRING(vspec1.vslev), SP;
         END
      ELSE
         CASE vspec1.vslev OF
```

```
            >= 63:   *vspstr* _ "ALL ";
            <= 9:    *vspstr* _ "  ", vspec1.vslev + '0, SP;
            ENDCASE
               BEGIN
               DIV vspec1.vslev / 10, firstd, secndd;
               *vspstr* _ SP, firstd + '0, secndd + '0, SP;
               END;
      CASE vspec1.vstrnc OF
         >= 63:   *vspstr* _ *vspstr*, "ALL";
         <= 9:    *vspstr* _ *vspstr*, "  ", vspec1.vstrnc + '0;
         ENDCASE
            BEGIN
            DIV vspec1.vstrnc / 10, firstd, secndd;
            *vspstr* _ *vspstr*, SP, firstd + '0, secndd + '0;
            END;
   % Set up hjuCP part of viewspec string %
      *vspstr* _ EOL, *vspstr*, EOL,
         IF vspec1.vsbrof THEN 'g ELSE IF vspec1.vsplxf THEN 'l ELSE
         'h,
         IF vspec1.vscapf THEN 'i ELSE IF vspec1.vscakf THEN 'k ELSE
         'j,
         IF vspec1.vsdaft THEN 'u ELSE 'v,
         IF vspec1.vsnamf THEN 'C ELSE 'D,
         IF vspec1.vsusgf THEN 'O ELSE 'P;
   RETURN;                                                          4I5
   END.
                                                                    4I6
(explct)   % CL:LB ;  explicit request for content checking %
PROCEDURE (charct, stid, oldls REF, oblock, da REF, sa REF, commands
REF LIST, staddr REF LIST);                                         4J
   % Procedure description
      FUNCTION
         This routine processes a statement that is already on the
         screen and for which explicit content checking has been
         requested.  It uses stfrmt to format the statement.  If the
         statement is on a new position on the screen, it issues a
         replace statement command.  Otherwise it issues replace
         line segment commands for each line segment that was
         previously on the screen and write line segment commands
         for each new statement has more line segments than the old.
         This routine assumes that statement number/sids and
         signatures are line segments in the string containing the
         text of the statement and not separate strings.
      ARGUMENTS
         charct:  position in statement at which to begin display
         stid:  stid of statement to be replaced
         oldls:  address of first line segment old display of
         statement
         oblock:  address of block containing oldls
         da:  address of display area
         commands:  list of commands
         staddr:  list of string table entry addresses
      RESULTS
         none
      NON-STANDARD CONTROL
         none
```

```
    GLOBALS
       none
    %
% Declarations %
    LOCAL festrngid, length, newls REF, nblock, moren, bp;
%Format statement and set flags %
    stfrmt(&da, stmntid(&oldls), charct, sa.swclvl, sa.swslvl,
    sa.swsvw, sa.swvspec, sa.swvsp2, &commands, oldls.strngid:
    &newls, nblock);
    oldls.stold _ 0;
    festrngid _ oldls.strngid;
%See if statement is at same position on screen%
    IF newls.sty NOT= oldls.sty THEN
       BEGIN  %new position.%
       #staddr# !_ &newls;  %save string table entry%
       newls.strngid _ festrngid;  %set fe string id%
       DO  % build replace statement command%
          BEGIN  %call buildcom for each line segment%
          buildcom(&commands, &da, &newls, lstext(&newls),
          festrngid);
          END
       WHILE dnxtls(&newls, nblock, forward: &newls, nblock);
       DO  %flag old line segment as not needing to be deleted%
          oldls.strew _ FALSE
       WHILE (dnxtls(&oldls, oblock: &oldls, oblock));
       END
    ELSE  %same position.  Use replace line sements%
       BEGIN
       %Build replace line segement command for each line segment.
        Copy FE string id and line segment id into new table %
          DO  %loop until out of line segments in old or new
          statement%
             BEGIN
             oldls.stnew _ FALSE;  %don't need delete command%
             newls.strngid _ festrngid;
             newls.stlsid _ oldls.stlsid;
             bldrpl(&newls, &da, &commands);
             END
          WHILE (moren _ dnxtls(&newls, nblock, forward: &newls,
          nblock)) AND dnxtls(&oldls, oblock: &oldls, oblock);
       %check for change in number of line segements and build
       write line segment commands as needed %
          IF moren THEN
             DO %more line segments in new%
                BEGIN
                newls.strngid _ festrngid;
                bldwls(&newls, &da, &commands);
                #staddr# !_ &newls;
                END
             WHILE dnxtls(&newls, nblock, forward: &newls,
             nblock);
       END;
% Return %
    RETURN;
END.
```

```
(srch)   % CL:LB ;  search old string table for matching stid %
PROCEDURE (stid, level, entry REF, block % => found, entry, block
%);                                                              4K
   % Procedure description
      FUNCTION
         Search old string table from "entry" for an entry with
         matching stid and level
      ARGUMENTS
         stid:  stid to search for
         level:  level at which statement is in new display
         entry:  entry in old table at which to begin search
         block:  block containing entry
      RESULTS
         found:  TRUE if match found, otherwise FALSE
         entry:  matching entry in table or 0 if no match
         block:  block containing entry or 0 if no match
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %
   % Declarations %
   %loop from "top" until end of table or entry found%
      DO
         IF entry.stlev = level AND stid = stmntid(&entry) THEN
            RETURN (TRUE, &entry, block)
      WHILE dgetnxt(&entry, block, forward: &entry, block);
   % Return %
      RETURN( FALSE, 0, 0);
   END.

(stmntid)   % CL: ;  get statement id for string table entry %
PROCEDURE (entry REF % => stid %);                               4L
   % Procedure description
      FUNCTION
         Given the address of the first line segment in an entry
         string in the string table, find the stid for the
         statement.  If the entry represents a string, rather than a
         statement, the procedure value is FALSE.  Otherwise it will
         be the ststid field of the first line segment if statment
         numbers/id's are not on the left or the second line segment
         if they are on the left.
      ARGUMENTS
         entry:  address entry in string table
      RESULTS
         stid:  stid if statement or FALSE
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %
   % Declarations %
      LOCAL stid, block;
   % Get ststid of first or second line segment if entry represents
   a statement %
      IF entry.stsrce = srcstat OR (entry.stsrce = srcstno AND
```

```
      dnxtls(&entry, block: &entry, block) AND entry.stsrce =
      srcstat) THEN
         stid _ entry.ststid
      ELSE stid _ FALSE;
   % Return %
      RETURN( stid );
   END.

(lscopy)   % CL:LB ;  copy line segments from old table to new %
PROCEDURE (charct, da REF, sa REF, commands REF LIST, staddr REF
LIST, oldls REF, oblock);                                        4M
   % Procedure description
      FUNCTION
         This routine processes a statement that was in the previous
         display and remained unchanged in content.  If the
         statement is at the same position on the screen or it had
         been complete in the previous display and can fit again,
         the data is copied from the old table to the new.  If the
         position of the statement changed, the screen coordinates
         are changed in the new table and the reposition flag is set
         in the new string table.  If the statement changed position
         on the screen and was not complete in the previous display
         write line segment commands are used to add any new line
         segments that can now fit on the screen.
      ARGUMENTS
         charct:  position to begin display in this statement
         da:  address of display area
         sa:  address of sequence area
         commands:  commands list
         staddr:  list of string table entry addresses
         oldls:  address of first line segment in statement
         oblock:  address of block containing oldls
      RESULTS
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %
   % Declarations %
      LOCAL index,offset,  nstmnt REF, newls REF, newst REF, nblock,
       saveblock, savelsptr, saveindex, savelength,  lslist REF,
      festrngid, moren;
   %Copy data if statement is at same position on screen or was
   complete and will be complete%
      IF (oldls.sty = da.dacrow) OR (oldls.stcomplete AND da.dacrow
      + oldls.stey - oldls.sty <= da.damrow) THEN
         BEGIN
         offset _ da.dacrow - oldls.sty;
         &newls _ 0;
         DO  %loop until all line segments copied%
            BEGIN
            IF &newls THEN  % allocate next line segment %
               &newls _ allols( &newst )
            ELSE  % allocate first line segment%
               &newls _ &newst _ allost( &da );
            saveblock _ newls.stcurallo;
```

```
            savelsptr _ newls.stlsptr;
            FOR index _ 0 UP UNTIL = strtl DO
               newls[index] _ oldls[index];
            oldls.stnew _ FALSE;
            oldls.stold _ &newls;  %point to corresponding new
            entry%
            newls.stcurallo _ saveblock;
            newls.stlsptr _ savelsptr;
            oldls.stkeep _ TRUE;  % keep block for string %
            IF offset THEN
               BEGIN
               newls.sty _ newls.sty + offset;
               newls.stey _ newls.stey + offset;
               newls.strepo _ TRUE;
               newls.stold _ &oldls;
               END
            ELSE newls.strepo _ FALSE;
            END
         WHILE dnxtls(&oldls, oblock: &oldls, oblock);
         da.dacrow _ newls.sty + 1;  %update current row %
         END
      ELSE  %must re-format statement%
         BEGIN
         stfrmt(&da, stmntid(&oldls), charct, oldls.stlev,
         sa.swslvl, sa.swsvw, sa.swvspec, sa.swvsp2, $commands,
         oldls.strngid: &newls, nblock);
         newls.strepo _ TRUE;  % statement is repositioned %
         newls.stold _ &oldls;  % save address to form scroll %
         oldls.stold _ 0;
         % copy FE string ids and line segment ids into line
         segments that were already on the screen %
            festrngid _ oldls.strngid;
            DO  %loop until out of line segments in old or new
            statement%
               BEGIN
               oldls.stnew _ FALSE;  %don't need delete command%
               newls.strngid _ festrngid;
               newls.stlsid _ oldls.stlsid;
               END
            WHILE (moren _ dnxtls(&newls, nblock, forward: &newls,
            nblock)) AND dnxtls(&oldls, oblock: &oldls, oblock);
         % do write line segment commands if more line segments can
         now fit on the screen.  old line segements that are no
         longer on the screen will be automatically deleted by the
         scroll or reposition string command %
            IF moren THEN
               DO %more line segments in new%
                  BEGIN
                  newls.strngid _ festrngid;
                  bldwls(&newls, &da, &commands);
                  #staddr# !_ &newls;
                  END
               WHILE dnxtls(&newls, nblock, forward: &newls, nblock)
            ELSE
               DO  %fewer line segements now%
                  oldls.stnew _ FALSE  %prevent Delete ls commands%
```

```
            WHILE dnxtls(&oldls, oblock: &oldls, oblock);
        END;
    % Return %
        RETURN;
    END.

(complsg) PROCEDURE (bps1, bps2, length);  %compare two line
segments -- Return TRUE is they contain the same string of
characters or length is zero%                                    4N
    IF length THEN
        FOR length DOWN UNTIL <= 0 DO
            IF ^bps1 NOT= ^bps2 THEN RETURN (FALSE);
    RETURN( TRUE );
    END.

(dodeletes)    % CL:LB ;  create delete commands %
PROCEDURE (da REF, block, commands REF LIST, clscrolls REF LIST); 40
    % Procedure description
        FUNCTION
            Search old string table for any line segments that must be
            deleted, i.e. the stnew is TRUE.  The scroll commands list
            is searched for each undeleted line segment/string and
            delete commands are created only for data that will not be
            automatically deleted by scroll commands.
            Note that if the stnew flag of the first line segment in a
            string is TRUE, the entire string must be deleted (either
            explicitly or automatically by a scroll) but if the the
            stnew flag of first line segment of a string is FALSE,
            there may still be other line segmerts in the string to
            delete.
            Also, if one segment of a string is automatically deleted
            by a scroll, all other line segements are still examined to
            see if they too are deleted by a scroll.
        ARGUMENTS
            da:  display area
            block:  address of old stable
            commands:  delete command list
            clscrolls:  scroll comands list
        RESULTS
            none
        NON-STANDARD CONTROL
            none
        GLOBALS
            none
        %
    % Declarations %
        LOCAL entry REF, ls REF, lsblock, examine;
    %Loop through string table%
        &entry _ block + dspbhl;
        DO
            BEGIN
            &ls _ &entry;
            lsblock _ block;
            examine _ TRUE;  %set to FALSE if whole string deleted%
            DO  %loop through line segments in string%
                IF ls.stnew AND NOT scrfall(&da, &ls, &clscrolls) THEN
```

```
            BEGIN  %line segment must be deleted%
            IF ls.stflag THEN    %if first line segment%
               BEGIN  %delete entire string%
               bldds t(&ls, &commands);
               examine _ FALSE;  %stop examining line segements%
               END
            ELSE  %just some line segments disappear%
               bldd ls(&ls, &commands);  %delete line segment%
            END
         WHILE examine AND dnxtls(&ls, lsblock: &ls, lsblock)
         END
      WHILE dgetnxt(&entry, block, forward: &entry, block);
   % Return %
      RETURN;
   END.

(scrfall)   % CL:LB ; see if line segment is deleted by a scroll %
PROCEDURE (da REF, ls REF, commands REF LIST % => fall %);          4P
   % Procedure description
      FUNCTION
         Examine scrolls commands to see if any sub-window includes
         line segment
      ARGUMENTS
         da:  diaplay area address
         ls:  address of line segment entry
         commands:  scroll command list
         scrindex:  index of first possible scroll command
         scrlast:  index of last possible scroll command
      RESULTS
         fall: TRUE if line segment falls out of sub-window, i.e. is
         automatically deleted in frontend; otherwise FALSE
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %
   % Declarations %
      LOCAL scrindex _ 1, fall _ FALSE, ptr REF;
   %Check commands until end of scrolls or sub-window found that
   contains line segment.  Note that the y coordinate of the line
   segment must be converted to the front-end coordinate for the
   check%
      UNTIL scrindex > commands.L OR fall DO
         BEGIN
         &ptr _ ELEM #commands#[scrindex];  %command%
         &ptr _ ELEM #ptr#[2];  %scroll parameter list%
         IF (da.dabottom - ls.sty) IN [ ELEM #ptr#[4], ELEM #ptr#[3]
         ] THEN
            fall _ TRUE
         ELSE BUMP scrindex;
         END;
   % Return %
      RETURN( fall );
   END.

(doscrolls)   % CL:LB ;  create scroll window commands %
```

```
PROCEDURE (da REF, sa REF, oblock, commands REF LIST, clscrolls REF
LIST, staddr REF LIST);                                              4Q
   % Procedure description
      FUNCTION
         search through new string table for any strings that have
         the reposition flag TRUE.  create scroll commands for any
         string or group of strings that can be scrolled and
         reposition string commands for any strings that cannot be
         scrolled.  scroll commands are put in a separate list so
         that they can be inserted before all other commands in the
         commands list.
         create at most one scroll command.  in certain cases with
         viewspec 'e', there is a bug in the BE if more than one
         scroll command is created.
      ARGUMENTS
         da:  display area address
         oblock:  start of old table
         commands:  command list
         clscrolls:  scroll commands list
         staddr:  list of string table entry addresses
      RESULTS
         none
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %
   % Declarations %
      LOCAL oldst REF, entry REF, block, bblock, bottom REF;
   % look at entries from top to bottom of new table %
      &oldst _ oblock + dspbhl;
      block _ da.dastrt;
      &entry _ block + dspbhl;
      DO
      IF entry.strepo THEN
         BEGIN
         &bottom _ scrsub(&entry, block: bblock);
         IF okscroll(&da, &sa, &oldst, oblock, &entry, block,
         &bottom, &clscrolls, &commands, &staddr) THEN
            BEGIN  %scroll built.%
            &entry _ &bottom;  %continue table scan below scroll%
            block _ bblock;
            END;
         END
      WHILE clscrolls.L = 0 AND dgetnxt(&entry, block, forward:
      &entry, block);
   % Return %
      RETURN;
   END.

(scrsub)   % CL:LB ;  get bottom string for scroll command
sub-window %
PROCEDURE (top REF, block % => bottom, block%);                      4R
   % Procedure description
      FUNCTION
         Search  the new string table to find other strings that
```

```
        could be repositioned in a scroll command whose sub-window
        starts with the entry "top"
      ARGUMENTS
        top:  entry in new string table.  first string in scroll
        block:  block containing top
      RESULTS
        bottom:  entry in new string table.  last string in scroll
        block:   block containing bottom
      NON-STANDARD CONTROL
        none
      GLOBALS
        none
      %
    % Declarations %
      LOCAL change, next REF, nextblock, bottom REF;
    %initialize%
      &bottom _ &top;
      change _ [top.stold].sty - top.sty;  %position change%
    %loop until at end of string table or at an entry that could not
    be included in scroll command%
      WHILE dgetnxt(&bottom, block, forward: &next, nextblock) AND
      next.strepo AND ([next.stold].sty) - next.sty = change DO
        BEGIN
        &bottom _ &next;
        block _ nextblock;
        END;
    % Return %
      RETURN(&bottom, block, change);
    END.

(okscroll)   % CL:LB ;  create scroll command if it is okay %
PROCEDURE (da REF, sa REF, oldst REF, block, top REF, tblock, bottom
REF, clscrolls REF LIST, commands REF LIST, staddr REF LIST% =>
okflag %);                                                        4S
    % Procedure description
      FUNCTION
        Check that scroll command for strings "top" through
        "bottom" will not delete any data that should remain on the
        screen.  If the scroll command is okay in this respect,
        build the scroll.
      ARGUMENTS
        da:  display area address
        oldst:  address of entry in old string table
        block:  block containing oldst
        top:  table entry.  first to be repositioned
        tblock:  block containing top
        bottom:  table entry.  last to be repositioned
        clscrolls:  scroll commands list
        commands:  commands list
        staddr:  list of entries in string table
      RESULTS
        okflag:  TRUE if scroll command built;  otherwise FALSE
      NON-STANDARD CONTROL
        none
      GLOBALS
        vinc
```

```
    %
% Declarations %
    LOCAL oldtop REF, oldbottom REF, change, okflag = TRUE,
    subtop, subbottom, outct = 0, scrct, oldls REF, lsblock;
    LOCAL LIST fallout;
% invoke catchphrases %
    INVOKE (catdaup);
% set boundaries of scroll sub-window and number of lines
scrolled%
    &oldtop _ top.stold;
    &oldbottom _ bottom.stold;
    change _ oldtop.sty - top.sty;  %position change%
    IF change < 0 THEN
        BEGIN  %data below strings would fall out of sub-window%
        subtop _ oldtop.sty;
        subbottom _ MIN((oldbottom.stey - change), da.dabottom);
        END
    ELSE
        BEGIN  %data above strings would fall out of sub-window%
        subtop _ oldtop.sty - change;
        subbottom _ oldbottom.stey;
        END;
    scrct _ oldbottom.stey - oldtop.sty + 1;
% search old string table for any strings that would fall out of
scroll sub-window and should not be deleted %
    DO
        IF NOT (oldst.sty IN [oldtop.sty, oldbottom.stey]) THEN
            BEGIN  % string is not one of scrolled strings %
            IF oldst.sty IN [subtop, subbottom] AND NOT oldst.stnew
            THEN  % whole string falls out %
                IF oldst.stold THEN
                    BEGIN  %count lines that have to be re-written%
                    outct _ outct + oldst.stey - oldst.sty + 1;
                    #fallout# !_ oldst.stold
                    END
                ELSE  % Cannot do scroll if scroll would delete a
                string for which write line segment commands exist %
                    okflag _ FALSE
            ELSE  % check if any line segment would fall out %
                BEGIN
                &oldls _ &oldst;
                DO
                    IF oldls.sty IN [subtop, subbottom] AND NOT
                    oldls.stnew THEN
                        okflag _ FALSE
                WHILE okflag AND dnxtls(&oldls, lsblock: &oldls,
                lsblock);
                END;
            END
    WHILE okflag AND outct < scrct AND dgetnxt(&oldst, block,
    forward: &oldst, block);
% build scroll command if more lines get scrolled than need to be
re-written%
    IF okflag AND scrct > outct THEN
        BEGIN
        bldscr(&da, subtop, subbottom, change, &clscrolls);
```

```
      IF outct  THEN
         refrmt($fallout, &da, &commands, &staddr);
      % Do not reposition scrolled strings %
         FOR &top UP strtl UNTIL > &bottom DO
            top.strepo _ FALSE;
      END
   ELSE okflag _ FALSE;
% Return %
   #fallout# _ ;
   DROP (ALL);
   RETURN( okflag );
% Catchphrases %
   (catdaup)  CATCHPHRASE();                                          4S8A
      BEGIN
      DISABLE (catdaup);
      CASE SIGNAL OF
         = return, = unwind :
            BEGIN
            NULL-LISTS;
            END;
      ENDCASE CONTINUE;
      END;
END.

(reIrmt)   % CL:LB ;  reformat string %
PROCEDURE ( fallout REF LIST, da REF, commands REF LIST, staddr REF
LIST);                                                                4T
   % Procedure description
      FUNCTION
         Build write string commands for strings that will get
         automatically deleted by a scroll command.  The entries for
         these strings already exists in the new string table.
      ARGUMENTS
         fallout:  addresses of affected entries in new string table

         da:  display area address
         commands:  commands list
         staddr:  list of addresses of string table entries
      RESULTS
         none
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %
   % Declarations %
      LOCAL block, index, entry REF;
   % Build write string commands %
      FOR index _ 1 UP UNTIL > fallout.L DO
         BEGIN
         &entry _ ELEM #fallout#[index];
         #staddr# !_ &entry;
         entry.strepo _ FALSE;
         DO  % build write string command%
            BEGIN  %call buildcom for each line segment%
            buildcom(&commands, &da, &entry, lstext(&entry), 0);
```

```
            END
         WHILE dnxtls(&entry, block, forward: &entry, block);
         END;
   % Return %
      RETURN;
   END.

(dorepos)   % CL:LB ;  create reposition commands %
PROCEDURE (da REF, commands REF LIST);                                  4U
   % Procedure description
      FUNCTION
         Search through new string table and build reposition string
         commands for any strings that have to be repositioned
      ARGUMENTS
         da:  display area address
         commands:  commands list
      RESULTS
         none
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %
   % Declarations %
      LOCAL entry REF, block;
   % Look for repositioned strings %
      block _ da.dastrt;
      &entry _ block + dspbhl;
      DO
         IF entry.strepo THEN bldrpo(&da, &entry, &commands)
      WHILE dgetnxt(&entry, block, forward: &entry, block);
   % Return %
      RETURN;
   END.

% Routines to build commands for FE display package procedures %
   (buildcom)   % LB ;  build write/replace string command %
   PROCEDURE (commands REF LIST, da REF, entry REF, stringadr REF,
   stringid);                                                           5A
      % Procedure description
         FUNCTION
            build write/replace string command and append to command
            list or build line segment and append to current command
            line segment list element
         ARGUMENTS
            commands REF LIST:  address of commands list
            da REF:address of display area
            entry:  address of line segment table entry
            stringadr REF:  address of line segement text
            stringid:  string-id if replace string; 0 if write string
         RESULTS
            proc-value
         NON-STANDARD CONTROL
            none
         GLOBALS
            dsplslist:  sets (and consequently nulls out previous
```

```
      contents)
    %
% Problem:  If list is too long, should break the string into two
strings or else we blow the list zone.  Maybe even have to call
prcmds in stages! Occurs when too many line segments in single
statement. %
% Declarations %
   (procpars) LOCAL REF _ 0;  % parameters to the procedure % 5A3A
   (stngatlst) LOCAL REF _ 0;  % string attribute list %      5A3B
   (cordlst) LOCAL REF _ 0;  % coordinates of origin %        5A3C
   REF dsplslist;
% Build command or append line segment %
   IF entry.stflag THEN
      BEGIN  %first line segment - build command%
      &dsplslist _ getlst(30);  %allocate line segment list%
      &procpars _ getlst(4); % parameters to the procedure %
      &stngatlst _ getlst(3); % string attribute list %
      &cordlst _ getlst(2); % coordinates of origin %
      #cordlst# _
         entry.stx1, da.dabottom-entry.sty;
      #stngatlst# _
         USE makedesc( ulist, &cordlst, TRUE ),
         USE makedesc( uindex, entry.statt, FALSE),
         USE makedesc( uindex, entry.stselector, FALSE);
      #dsplslist# _ USE makedesc( ulist, formls(0, &stringadr,
      0), TRUE );
      IF NOT stringid THEN
         BEGIN % parameters for write string %
         #procpars# _
            USE makedesc( uindex, da.dawid, FALSE),
            USE makedesc( ulist, &stngatlst, TRUE ),
            USE makedesc( ulist, &dsplslist, TRUE );
         END
      ELSE
         BEGIN % parameters for replace string %
         #procpars# _
            USE makedesc( uindex, da.dawid, FALSE),
            USE makedesc( uindex, stringid, FALSE),
            USE makedesc( ulist, &stngatlst, TRUE ),
            USE makedesc( ulist, &dsplslist, TRUE );
         END;
      addtobatch( &commands, &procpars, IF NOT stringid THEN
      writstr ELSE replstr );
      END
   ELSE  %not the first line segment in string%
      BEGIN
      % Append the line segment to the list of line segments
      associated with this string.  'dsplslist' has address of
      line segment list. %
      #dsplslist# !_
         USE makedesc( ulist, formls(&entry, &stringadr, &da),
         TRUE );
      END;
% Return %
   RETURN;
END.
```

```
(bldrpl)   % LB: ;  build replace line segment for batch command %
PROCEDURE (entry REF, da REF, commands REF LIST);                    5B
   % Procedure description
      FUNCTION
         Build replace line segment command for batch command and
         append to command list.  Use 'lstext' to create string of
         the line segment text.  This string gets freed when the
         commands list gets nulled.
      ARGUMENTS
         entry:  address of line segment table entry
         da:  address of display area
         commands:  address of batch commands list
      RESULTS
         none
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %
   % Declarations %
      (batchpar) LOCAL REF _ 0;;                                     5B2A
   % allocate storage for list %
      &batchpar _ getlst(4);
   % set up list of paramters for replace line segment %
      #batchpar# _
         USE makedesc( uindex, entry.stwid, FALSE),
         USE makedesc( uindex, entry.strngid, FALSE),
         USE makedesc( uindex, entry.stlsid, FALSE),
         USE makedesc (ulist, formls(&entry, lstext(&entry), &da),
         TRUE);
   % append to commands list %
      addtobatch(&commands, &batchpar, repls);
   % Return %
      RETURN;
   END.

(bldwls)   % LB: ;  build write line segment for batch command %
PROCEDURE (entry REF, da REF, commands REF );                        5C
   % Procedure description
      FUNCTION
         build write line segment for batch command and append to
         command list.  Use 'lstext' to create a string of the text
         of the lne segment.  This string gets freed when command
         list gets nulled.
      ARGUMENTS
         entry:  address of entry in line segment table
         da:  address of display area
         commands :  command list
      RESULTS
         proc-value
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %
```

```
   % Declarations %
      (batchpar) REF LIST _ 0;  %address of parameter list%      5C2A
   %allocate list from list zone%
      &batchpar _ getlst(3);
   % set up list of paramters for write line segment %
      #batchpar# _
         USE makedesc( uindex, entry.stwid, FALSE),
         USE makedesc( uindex, entry.strngid, FALSE),
         USE makedesc (ulist, formls(&entry, lstext(&entry), &da),
         TRUE);
   % append to commands list %
      addtobatch(&commands, &batchpar, writls);
   % Return %
      RETURN;
   END.

(bldscr)   % LB ;  build scroll for batch command %
PROCEDURE (da REF, subtop, subbottom, change, commands REF LIST);  5D
   % Procedure description
      FUNCTION
         Change sub-window coordinates to front-end coordinates
         (display manipulates y coordinates as if upper left corner
         is 0,0.  In  front-end lower left corner is 0,0) and append
         command to scroll command list
      ARGUMENTS
         da:  address of display area
         subtop:  top of scroll sub-window
         subbottom:  bottom of sub-window
         change:  amount and direction of change
         commands:  command list
      RESULTS
         none
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %
   % Declarations %
      (batchpar) LOCAL REF _ 0;                                  5D2A
   % set up list of paramters for scroll %
      &batchpar _ getlst(4);  %allocate from list zone%
      #batchpar# _
         USE makedesc( uindex, da.dawid, FALSE),
         change,
         da.dabottom-subtop,
         da.dabottom-subbottom;
   % append to commands list %
      addtobatch(&commands, &batchpar, scrwind);
   % Return %
      RETURN;
   END.

(bldrpo)   % LB ;  build reposition string for batch command %
PROCEDURE (da REF, entry REF, commands REF LIST);                  5E
   % Procedure description
      FUNCTION
```

```
      Change y coordinate to front-end coordinate and append
      command to reposition command list
   ARGUMENTS
      da:  display area address
      entry:  entry in new string table
      commands:  command list
   RESULTS
      none
   NON-STANDARD CONTROL
      none
   GLOBALS
      none
   %
% Declarations %
   (coords) LOCAL REF _ 0;  %address of coords list%          5E2A
   (batchpar) LOCAL REF _ 0;  %address of parameter list %    5E2B
% set up list of coordinates %
   &coords _ getlst(2);  %allocate from list zone%
   #coords# _ entry.stx1, da.dabottom - entry.sty;
% set up list of paramters for write line segment %
   &batchpar _ getlst(3);  %allocate from list zone%
   #batchpar# _
      USE makedesc( uindex, entry.stwid, FALSE),
      USE makedesc( uindex, entry.strngid, FALSE),
      USE makedesc (ulist, &coords, TRUE);
% append to commands list %
   addtobatch(&commands, &batchpar, repostr);
% Return %
   RETURN;
END.

(blddls)   % LB: ;  build delete line segment for batch command %
PROCEDURE (entry REF, commands REF LIST);                      5F
   % Procedure description
      FUNCTION
         build delete line segment for batch command and append to
         command list
      ARGUMENTS
         entry:  entry in old string table
         commands:  command list
      RESULTS
         proc-value
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %
   % Declarations %
      (batchpar) LOCAL REF _ 0;  %address parameter list%      5F2A
   % set up list of paramters for delete line segment %
      &batchpar _ getlst(3);  %allocate from list zone%
      #batchpar# _
         USE makedesc( uindex, entry.stwid, FALSE),
         USE makedesc( uindex, entry.strngid, FALSE),
         USE makedesc( uindex, entry.stlsid, FALSE);
   % append to commands list %
```

```
      addtobatch(&commands, &batchpar, dells);
   % Return %
      RETURN;
   END.

(bldst)   % LB: ;  build delete string for batch command %
PROCEDURE (entry REF, commands REF LIST);                              5G
   % Procedure description
      FUNCTION
         build delete string for batch command and append to command
         list
      ARGUMENTS
         entry;:  entry in old string table-first line segment in
         string
         commands:  command list
      RESULTS
         proc-value
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %
   % Declarations %
      (batchpar) LOCAL REF _ 0;  %address of parameter lit%        5G2A
   % set up list of paramters for delete line segment %
      &batchpar _ getlst(2);  %allocate from list zone%
      #batchpar# _
         DSE makedesc( uindex, entry.stwid, FALSE),
         DSE makedesc( uindex, entry.strngid, FALSE);
   % append to commands list %
      addtobatch(&commands, &batchpar, delstr);
   % Return %
      RETURN;
   END.

(bldcw)   % LB ;  build create window for batch command %
PROCEDURE (wa REF, commands REF LIST);                                 5H
   % Procedure description
      FUNCTION
         build create window command and append command to command
         list
      ARGUMENTS
         wa:  window area address
         commands:  commands list
      RESULTS
         proc-value
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %
   % Declarations %
      (batchpar) LOCAL REF _ 0;  %address of parameter list%       5H2A
   % Form parameter list %
      &batchpar _ getlst(4);  %allocate from list zone%
      argcrw(&wa, &batchpar);
```

```
   % Append to commands list %
      addtobatch( &commands, &batchpar, crewind);
   % Return %
      RETURN;
   END.

(bldclw)   % LB ;  build clear window for batch command %
PROCEDURE (wa REF, commands REF LIST);                              5I
   % Procedure description
      FUNCTION
         build delete window command and append to batch command
      ARGUMENTS
         wa:  window area address
         commands:  commands list
      RESULTS
         proc-value
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %
   % Declarations %
      (batchpar) LOCAL REF _ 0;  %address of parameter list%     5I2A
   % Form delete window parameter list and append to batch command
   list %
      &batchpar _ getlst(1);  %allocate from list zone%
      #batchpar# _ USE makedesc( uindex, wa.widindex, FALSE);
      addtobatch( &commands, &batchpar, delwind);
   % Return %
      RETURN;
   END.

(bldclw)   % LB ;  build clear window for batch command %
PROCEDURE (wa REF, commands REF LIST);                              5J
   % Procedure description
      FUNCTION
         Build clear window command and append to batch command.  If
         wa is within the window area record block, assume it is the
         address of a window area.  If it is outside the block,
         assume it is the window id.
      ARGUMENTS
         wa:  window area address or window id
         commands:  commands list
      RESULTS
         proc-value
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %
   % Declarations %
      (batchpar) LOCAL REF _ 0;  %address of parameter list%     5J2A
   % Form clear window parameter list and append to batch command
   list %
      &batchpar _ getlst(1);  %allocate from list zone%
      #batchpar# _
```

```
        USE makedesc( uindex,
        IF &wa IN [$wndwarea,$wndwend] THEN wa.widindex ELSE &wa,
        FALSE);
      addtobatch( &commands, &batchpar, clrwind);
   % Return %
      RETURN;
   END.

(bldwli)   % LB ;  build write-literal command %
PROCEDURE (wa REF, string REF, commands REF);                          5K
   % Procedure description
      FUNCTION
         Build write-literal command and append to commands list.
         If string is not in list zone, this procedure make a copy
         of the string in the list zone.  The copy (or the original
         string if it is in the list zone), gets freed when the
         commands list gets nulled.
      ARGUMENTS
         wa:  address of window area
         string:  address of literal string
         commands:  address of commands list
      RESULTS
         proc-value
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %
   % Declarations %
      (procpar) LOCAL REF _ 0;  %address of parameter list%      5K2A
      (copyptr) LOCAL REF _ 0;                                   5K2B
   % Copy string if it is not in list zone %
      IF &string IN [$fsblist, $fsblend] THEN
         &copyptr _ &string  %it is in list zone%
      ELSE  %it is not in list zone%
         BEGIN
         &copyptr _ getstring(string.L, lstzone);  %allocate string%
         *copyptr* _ *string*;  %copy string%
         END;
   % Build write-literal command %
      &procpar _ getlst(2);
      #procpar# _
         USE makedesc( uindex, wa.widindex, FALSE),  %window id%
         USE makedesc( ustring, &copyptr, TRUE);  %literal string%
   % Add to commands list %
      addtobatch (&commands, &procpar, writlit);
   % Return %
      RETURN;
   END.

(addtobatch)   % LB: ;  append to batch-command's parameter list %
PROCEDURE (commands REF LIST, procpar REF, procop);                    5I
   % Procedure description
      FUNCTION
         Append a call to a FE procedure to parameter list of FE
         batch-command.  This procedure assumes that the parameter
```

```
      list of the FE call to be appended is in the list zone and
      does not make a copy of the list but instead makes a
      descriptor to point to it.
   ARGUMENTS
      commands:  address of parameter list of batch command
      procpar:  address of the parameter list of the FE procedure
      procop:  op code of the FE procedure
   RESULTS
      none
   NON-STANDARD CONTROL
      none
   GLOBALS
      none
   %
% Declarations %
   (batchpar) LOCAL REF _ 0;  %address of batch command list elt%
                                                              5L2A
% set up element for batch command parameter list %
   &batchpar _ getlst(2);  %allocate in list zone%
   #batchpar# _
      USE makedesc( uindex, procop, FALSE),
      USE makedesc( ulist, &procpar, TRUE );
% append to batch command parameter list %
   #commands# !_
      USE makedesc( ulist, &batchpar, TRUE );
% Return %
   RETURN;
END.

(argcrw)   % LB :  build create window command argument list %
PROCEDURE (wa REF, arglist REF LIST);                          5M
% Procedure description
   FUNCTION
      build create window command argument list according to
      specified window area
   ARGUMENTS
      wa:  window area address
      arglist:  address of argument list
   RESULTS
      proc-value
   NON-STANDARD CONTROL
      none
   GLOBALS
      none
   %
% Declarations %
% Form parameter list %
   #arglist# _
      USE makedesc( uindex, wa.widowner, FALSE),
      USE makedesc( uindex, wa.wtype, FALSE),
      LIST (wa.wuplx, wa.wuply, wa.wlrx, wa.wlry),  %diags%
      LIST(  %window attributes%
         USE makedesc( uindex, wa.wipriority, FALSE),
         LIST(  %string attributes%
            USE makedesc( unull, 0, FALSE),  %coords ignored%
            USE makedesc( uindex, wa.wiatt, FALSE),
```

```
                USE makedesc( uindex, wa.wiselector, FALSE)
                )
            );
    % Return %
      RETURN;
    END.

(argclw)   % LB ;  build clear window command argument list %
PROCEDURE (wndid, arglist REF LIST);                                    5N
    % Procedure description
      FUNCTION
         build clear window command argument list according to
         specified window ID
      ARGUMENTS
         wndid:  window ID
         arglist:  address of argument list
      RESULTS
         proc-value
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %
    % Declarations %
    % Form parameter list %
      #arglist# _
         USE makedesc( uindex, wndid, FALSE);
    % Return %
      RETURN;
    END.

(formls)   % LB ;  form line segment list %
PROCEDURE (entry REF, stringadr REF, da REF);                           5D
    % Procedure description
      FUNCTION
         Form line segment list for commands that require one.
         Check address of string.  If it not is in the list zone
         (fsblist-fsblend), this procedure makes a copy of string in
         list zone.  If 'entry' is 0, the line segment attributes
         are null in the line segment list.  The copied string (or
         the original string if it was in the list zone will be
         freed when the commands list is nulled.
      ARGUMENTS
         entry:  address of line segment table entry / 0
         stringadr:  address of text of line segment; assumed to be
         in list zone
         da:  address of display area
      RESULTS
         address of list formed (allocated dynamically by this
         procedure from list zone)
      NON-STANDARD CONTROL
         none
      GLOBALS
         fsblist, fsblend
      %
    % Declarations %
```

```
      (cordlst) LOCAL REF _ 0;  % coordinates of origin %          502A
      (linlst) LOCAL REF _ 0;  % line segment element list %       502B
      (lnatlst) LOCAL REF _ 0;  % line segment attribute list %    502C
      (copystr) LOCAL REF _ 0;  % address of string for list elt %
                                                                   502D
   % Allocate line segment element list %
     &linlst _ getlst(2); % line segment element list %
   % Copy string if not in shared pages %
     IF &stringadr NOT IN [$fsblist, $fsblend] THEN
        BEGIN  %string is not in shared pages%
        &copystr _ getstring(stringadr.L, lstzone);
        *copystr* _ *stringadr*;
        END
     ELSE &copystr _ &stringadr;  %string is in shared pages%
   % Build line segment list %
     IF &entry THEN
        BEGIN  %use line segment attributes%
        &cordlst _ getlst(2); % coordinates of origin %
        &lnatlst _ getlst(3); % line segment attribute list %
        #cordlst# _
           entry.stx1, da.dabottom-entry.sty;
        #lnatlst# _
           USE makedesc( ulist, &cordlst, TRUE ),
           USE makedesc( uindex, entry.statt, FALSE),
           USE makedesc( uindex, entry.stselector, FALSE);
        #linlst# _
           USE makedesc( ulist, &lnatlst, TRUE),
           USE makedesc( ustring, &copystr, TRUE);
        END
     ELSE  %null line segment attributes - first line segment%
        #linlst# _
           USE makedesc( unull, 0, FALSE),
           USE makedesc( ustring, &copystr, TRUE);
   % Return %
     RETURN(&linlst);
   END.

(lstext)   % LD: ;  create string of line segment text %
PROCEDURE (lineseg REF % => adrstr %);                             5P
   % Procedure description
     FUNCTION
        Create a string of the text of the line segment  using a
        string allocated in the list zone.  If text is a special
        text ('stcbug' field is FALSE indicating TAB or
        non-printable), use spaces or non-printable string
        representatin (e.g. <LF>) in place of actual character in
        line segment.
        The line segment entry is complete when this procedure is
        called.
     ARGUMENTS
        lineseg:  address of line segment entry in line segment
        table
     RESULTS
        adrstr:  address of created text string
     NON-STANDARD CONTROL
        none
```

```
      GLOBALS
         none
      %
   % Declarations %
      (adrstr) REF ;  %address of string from list zone%          5P2A
      (bptr) LOCAL;                                               5P2B
      (char) LOCAL;                                               5P2C
      (length) LOCAL;                                             5P2D
      (locstr) STRING [10];                                       5P2E
   % build (TAB or NP) string or use buildstring %
      IF lineseg.stcbug THEN  %display actual text from line
      segment%
         &adrstr _ buildstring(lineseg.stbps, lineseg.stbpe)
      ELSE  %TAB or non-printable%
         BEGIN
         &adrstr _ getstring( ( length _ lineseg.stx2 - lineseg.stx1
         + 1), lstzone);  %length of line segment%
         bptr _ lineseg.stbps;  %byte ptr. to first char in text%
         IF (char _ ^bptr ) = TAB THEN  %use spaces for TAB%
            FOR length DOWN UNTIL = 0 DO
               *adrstr* _ *adrstr*, SP
            ELSE  %use string representation of non-printable%
               BEGIN
               *locstr* _ *[npstrad(char)]*;
               locstr.L _ length;  %truncate if necessary%
               *adrstr* _ *locstr*;
               END;
         END;
   % Return %
      RETURN(&adrstr);
   END.

%...routines to call FE display package and support routines...%
   (prcmds) PROCEDURE (da REF, commands REF LIST, staddr REF LIST,
   batchres REF LIST);                                            6A
      % Call the frontend display processor batch commands routine.  If
      the call is from dafrmt or daupdate, the FE will return a list of
      string and line segment IDs which must be decoded and placed into
      the appropriate locations in the string tables associated with
      the display area.  If the call is from dafrmt (total recreation
      of screen), the string and line segment IDs are placed according
      to the order of entries in the string table.  If the call is from
      daupdate (selective screen update), the list of string table
      stentry addresses (staddr) is used to determine the
      correspondence between string and line segment IDs and string
      table entries. %
      % ARGUMENTS
         da:  display area address
            0 if FE results are not to be processed here
         commands:  address of list of commands to the fe
         staddr:  if call from dafrmt, =FALSE
            if call from daupdate, address of list of string table
            stentry addresses - an entry for each command for which
            there will be return values from the FE
         batchres:  address of list in which to store batch command's
         results from the FE
```

```
        ignored if FE results are processed here
    %
    LOCAL rvalue, opcode, cindex, rindex = 1, block, stentry REF,
    clptr REF, rlptr REF, blptr REF, retlist REF;
    REF dsparg;
    % Invoke catchphrases %
       INVOKE (catprc);
    % return if no commands %
       IF NOT commands.L THEN
          BEGIN
          DPOP (ALL);
          RETURN;
          END;
    % call FE via batch commands %
       #dsparg# _ USE makedesc( ulist, &commands, FALSE );
       IF NOT extcall( feident %process handle or mailbox%,
       $"batch-display-commands", fedpypkg %package handle or
       callmode%, &dsparg: &retlist) THEN
          ABORT (erdfedisplay,$"Error in FE - prcmds");
       &blptr _ ELEM #retlist# [1] ;  %addr. of command return list%
    % decode return list if display area specified %
       IF &da THEN
          BEGIN
          %initialize index into return list and set pointers  to
          first stentry in string table if call from dafrmt%
             IF NOT &staddr THEN
                BEGIN
                block _ da.dastrt;
                &stentry _ block + dspbhl;
                END;
          %examine each command to see if there is any return for it%
             FOR cindex _ 1 UP UNTIL > commands.L DO
                BEGIN
                &clptr _ ELEM #commands#[cindex];
                CASE ELEM #clptr# [1] OF  %opcode in batch command%
                   = writstr:  %write string%
                      BEGIN
                      IF rindex > blptr.L THEN
                         err($"too few return values from FE --
                         prcmds");
                      IF &staddr THEN
                         &stentry _ ELEM #staddr#[rindex];
                      &rlptr _ ELEM #blptr#[rindex];
                      stentry.strngid _ ELEM #rlptr#[1];
                      prclsid(&stentry, ELEM #rlptr#[2],
                      stentry.strngid);
                      IF NOT &staddr THEN
                         dgetnxt(&stentry, block, forward: &stentry,
                         block);
                      BUMP rindex;
                      END;
                   = replstr:  %replace string%
                      BEGIN
                      IF rindex > blptr.L THEN
                         err($"too few return values from FE --
                         prcmds");
```

```
                    &stentry _ ELEM #staddr#[rindex];
                    prclsid(&stentry, ELEM #blptr#[rindex],
                    stentry.strngid);
                    BUMP rindex;
                    END;
                = writls:  %write line segment%
                    BEGIN
                    IF rindex > blptr.L THEN
                       err($"too few return values from FE --
                       prcmds");
                    &stentry _ ELEM #staddr#[rindex];
                    stentry.stlsid _ ELEM #blptr#[rindex];
                    BUMP rindex;
                    END;
                ENDCASE;  %no return values for other commands%
             END;
          END
       ELSE  % Move results into calling routine's list %
          IF blptr.L THEN #batchres# _ MOVE #blptr#;
    % Return %
       DROP (ALL);
       #dsparg# _ ;
       RETURN;
    % Catchphrases %
       (catprc)  CATCHPHRASE();                                    6A11A
          BEGIN
          DISABLE (catprc);
          CASE SIGNAL OF
             = return, = unwind :
                BEGIN
                #dsparg# _ ;
                END;
          ENDCASE CONTINUE;
          END;
    END.

(prclsid)  %process front end line segment ids in return list%
PROCEDURE (stentry REF, lslist REF LIST, festid);                   6B
    LOCAL index, block;
    index _ 1;
    DO  %loop through line segments in string%
       BEGIN
       IF index > lslist.L THEN
          err($"too few return values from FE -- prclsid");
       stentry.stlsid _ ELEM #lslist#[index];  %fe line segment id%
       stentry.strngid _ festid;  %fe string id%
       BUMP index;
       END
    WHILE dnxtls(&stentry, block: &stentry, block);
    RETURN;
    END.
(dfecrw)   % LP :  build and send FE create window command %
PROCEDURE (wa REF);                                                 6C
    % Procedure description
       FUNCTION
          build create window command according to specified window
```

```
        area and send to FE
      ARGUMENTS
        wa:  window area address
      RESULTS
        wid:  window id returned from FE
      NON-STANDARD CONTROL
        none
      GLOBALS
        none
      %
   % Declarations %
      LOCAL retval, retlist REF;
      REF dsparg;  %address of argument list%
   % Invoke catchphrases %
      INVOKE (catdfecw);
   % Form parameter list %
      argcrw(&wa, &dsparg);
   % Send to Front End %
      IF NOT extcall( feident %process handle or mailbox%,
      $"create-window", fedpypkg %package handle or callmode%,
      &dsparg: &retlist) THEN
         ABORT (erdfedisplay,$"Error in FE - dfecw");
   % Return %
      retval _ ELEM #retlist# [1];
      #dsparg# _;
      DROP (ALL);
      RETURN( retval ); %return window id%
   % Catchphrases %
      (catdfecw)  CATCHPHRASE();                                    6C7A
         BEGIN
         DISABLE (catdfecw);
         CASE SIGNALTYPE OF
            = notetype:
               CASE SIGNAL OF
                  = return, = unwind :
                     BEGIN
                     DROP (ALL);
                     #dsparg# _ ;
                     END;
               ENDCASE CONTINUE;
         ENDCASE CONTINUE;
         END;
   END.

(dfegtw)   % LB :  build and send FE get-window characteristics %
PROCEDURE (wchrct REF LIST);                                          6D
   % Procedure description
      FUNCTION
         build get-window characteristcs command and send to FE
      ARGUMENTS
         wchrct:  address of list in which to place window
         characteristics returned from FE
      RESULTS
         none
      NON-STANDARD CONTROL
         none
```

```
      GLOBALS
         none
      %
   % Declarations %
      LOCAL lptr REF, retlist REF;
      REF dsparg;
   % Invoke catchphrases %
      INVOKE (catdfegtw);
   % Call FE and get result %
      IF NOT extcall(feident %process handle or mailbox%,
      $"get-windows", fedpypkg %package handle or callmode%,
      &dsparg: %call requires argument list% &retlist %return list%)
      THEN
         ABORT(erdfedisplay, $"Error in FE - dfegtw");
      % copy window characteristic list into calling routine's list%
         &lptr _ ELEM #retlist# [1];  %addr. of command results
         list%
         #wchrct# _ COPY #lptr#;
   % Return %
      DROP (ALL);
      RETURN;
   % Catchphrases %
      (catdfegtw)  CATCHPHRASE();                                    6D6A
         BEGIN
         DISABLE (catdfegtw);
         CASE SIGNALTYPE OF
            = notetype:
               CASE SIGNAL OF
                  = return, = unwind :
                     BEGIN
                     DROP (ALL);
                     #dsparg# _;
                     END;
               ENDCASE CONTINUE;
         ENDCASE CONTINUE;
         END;
   END.

(wrlstr)     % LB : build and send FE write literal string %
PROCEDURE (wa REF, string REF, clear);                                 6E
   % Procedure description
      FUNCTION
         Build command to write a literal string in a window and
         call FE to do it.  In addition, if 'clear' is TRUE', build
         and send command to clear window before writing the string.
          The commands are sent to the FE via 'prcmds' which means
         that the FE procedure 'batch-commands' is used.
      ARGUMENTS
         wa:  address of window area
         string:  address of string
         clear:  TRUE to clear window before writing; FALSE for no
         clear
      RESULTS
         none
      NON-STANDARD CONTROL
         none
```

```
      GLOBALS
         none
      %
   % Declarations %
      LOCAL cmdlst REF _ 0;
   % Initialize cmdlst %
      &cmdlst _ getlst(2);
   % Invoke catchphrase %
      INVOKE (catwrlstr);
   % Null out global commands list %
      #cmdlst# _ ;
   % Build command(s) and send to FE%
      IF clear THEN
         clearda(wa.widdarea, &cmdlst);  %clear window command%
      bldwli(&wa, &string, &cmdlst);  %write literal command%
      prcmds(0, &cmdlst, 0, 0);  %send to FE%
   % Drop catch phrase and null out global commands list %
      DROP (catwrlstr);
      #cmdlst# _ ;
   % Return %
      RETURN;
   % Catchphrases %
      (catwrlstr)  CATCHPHRASE();                                    6E9A
         BEGIN
         DISABLE (catwrlstr);
         CASE SIGNALTYPE OF
            = notetype:
               CASE SIGNAL OF
                  = return, = unwind :
                     BEGIN
                     #cmdlst# _ ;
                     END;
               ENDCASE CONTINUE;
         ENDCASE CONTINUE;
         END;
   END.

(dfeclw)   % LR :  build and send FE clear window command %
PROCEDURE (windid);                                                  6F
   % Procedure description
      FUNCTION
         build clear window command according to specified window id
         and send to FE
      ARGUMENTS
         wndid:  window id
      RESULTS
         none
      NON-STANDARD CONTROL
         none
      GLOBALS
         none
      %
   % Declarations %
      REF dsparg;  %address of argument list%
   % Invoke catchphrases %
      INVOKE (catdfecw);
```

```
    % Form parameter list %
      #dsparg# _
         USE makedesc( uindex, windid, FALSE);
    % Send to Front End %
      IF NOT extcall( feident %process handle or mailbox%,
      $"clear-window", fedpypkg %package handle or callmode%,
      &dsparg) THEN
         ABORT (erdfedisplay,$"Error in FE - dfeclw");
    % Return %
      #dsparg# _;
      DROP (ALL);
      RETURN;
    % Catchphrases %
      (catdfecw)  CATCHPHRASE();                                    6F7A
         BEGIN
         DISABLE (catdfecw);
         CASE SIGNALTYPE OF
            = notetype:
               CASE SIGNAL OF
                  = return, = unwind :
                     BEGIN
                     DROP (ALL);
                     #dsparg# _ ;
                     END;
               ENDCASE CONTINUE;
         ENDCASE CONTINUE;
         END;
    END.

% display block storage management %
   % default size (and increment) for string table is the number of
   lines in the display area; for a line segment table, always 5.These
   numbers are reasonable given the screen size and the nature of NLS
   statements.
      Link these blocks together rather than getting an incrementally
      larger block and copying everything else over.
      Within blocks, do not link segments together.  Rather, they will
      be pushed on top of each other: end of segments/strings indicated
      by non-existant lsrt.  End of da by non-existant string.
      Auxilliary blocks used so copying may be facilitated.
      %
   (dnxt) PROCEDURE % get next segment block %                       7B
      ( curblk REF); % block address from which next item is to be
      obtained.  If zero, none associated yet; this is an error.
      Should call dbnew and link to parent data structure before
      calling this procedure.  dnxt will call dbnew if this block runs
      out, but cannot get the first one!)  Returns address of next
      entry in structure to be filled. %
         % Note: when a block is filled and after a new one attached
         the fields have the following characteristics:
            dbprev: points to previous block or 0 if this is the first
            dbnxt: points to the next block
            denxt: points BEYOND the last element in this block.  NOT A
            LEGITIMATE ADDRESS!  The get procedure (dgetnxt) checks if
            the element address is beyond the llast element in the
            block.  Does NOT point to the first element in the next
```

```
        block.
        delst:  address of last element in the block.
        dparad, dbtype:  parent address and block type:  lstype or
        strtype
      %
   LOCAL entry REF; %address temp: is in blocks%
   CASE &entry _ (curblk.denxt := curblk.denxt + strtl) OF
      > curblk.delst:   % no room in this block%
        BEGIN
        &curblk _ dbnew( curblk.dparad, &curblk, curblk.dbtype );
        &entry _ (curblk.denxt := curblk.denxt + strtl);
        END;
      ENDCASE;   %room in this block%
   RETURN( &entry, &curblk);
   END.

(dbnew) PROCEDURE % get new block; size depends on type of block and
size of window. link it to the old block (or to the parent data
structure if this is the first; if &curblk is 0, this is the first
block %                                                          7C
   (parad REF,
   curblk REF,
   type);
   LOCAL newblk REF,
      size,
      number,
      length;
   length _ strtl;
   number _ IF type = lstype THEN 5 ELSE MIN( 44,(parad.dabottom +
   1) );
      % if a string type, then allocate a block with the number of
      elements corresponding to the number of lines in the window or
      44, whichever is smaller. %
   IF NOT &newblk _ getblk ((size _ number*length + dspbhl),
   $dspblk) %allocate new block% THEN
      BEGIN
      ABORT(spacerr, $"FATAL DISPLAY ERROR - type ^C then RESET
      Reenter NLS by retyping 'NLS'"); %he'S in trouble%
      END;
   IF &curblk THEN
      BEGIN
      curblk.dbnxt _ &newblk; %block header: new is the next block
      for old%
      newblk.dprev _ &curblk; % old is the previous block for the
      new%
      CASE type OF
         =lstype: parad.stcurallo _ &newblk;
         ENDCASE parad.dacurallo _ &newblk;
      END
   ELSE
      BEGIN
      CASE type OF
         =lstype: parad.stlsptr _ parad.stcurallo _ &newblk;
         ENDCASE parad.dastrt _ parad.dacurallo _ &newblk;
      END;
   newblk.dbnxt _ 0;
```

```
    newblk.dparad _ &parad;
    newblk.dbtype _ type;
    newblk.delst _ &newblk + size - length;
    newblk.denxt _ &newblk + dspbhl;
    RETURN( &newblk);
    END.

(dbtree) PROCEDURE % free the blocks, unlink them  %                    7D
    %starting at blkadress, free all display command blocks up to the
    last in the chain (has no dbnxt  field).%
    (blkaddress REF);  % starting block in chain: 0 start (and if
    appropriate current) address  in calling procedure!%
    LOCAL savblk;
    REF savblk;
    &savblk _ &blkaddress; %original starting block--last used%
    LOOP
        BEGIN
        % If we are at the end of the chain, or if this is an illegal
        address, exit. %
            IF NOT &savblk THEN EXIT LOOP;
        % Advance the block marker and free the current one. %
            freeblk( &savblk := savblk.dbnxt, $dspblk );
        END;
    RETURN;
    END.

(dnxtls)   % get next line segment in statement %
PROCEDURE (ls REF, block % => flag, ls, block %);                      7E
    % Procedure description
        ARGUMENTS
            ls:  entry in string or line segment table
            block:  block containing ls
        RESULTS
            flag:  TRUE if more line segments in statement; FALSE if
            not
            ls:  line segement entry address.  updated to next if next
            exists, otherwise same as on entry
            block:  block containing ls
        %
    %If first line segment in statement, follow pointer into line
    segment table and get first line segment there.  Otherwise get
    next line segment%
        IF ls.stflag THEN
            IF ls.stlsptr THEN
                RETURN(TRUE, ls.stlsptr + dspbhl, ls.stlsptr)
            ELSE RETURN(FALSE, &ls, block)   %only one line segment%
        ELSE RETURN( dgetnxt(&ls, block, forward: &ls, block), &ls,
        block);
    END.

(dgetnxt) PROCEDURE (entry REF, block REF, direction);                 7F
    %updates entry and block in direction specified%
    CASE direction OF
        =forward:
            BEGIN
            &entry _ &entry + strtl;
```

```
            IF &entry <= block.delst THEN
               BEGIN  %get next entry in this block if entry exists%
               IF entry.stexis THEN RETURN(TRUE, &entry, &block);
               RETURN (FALSE, &entry - strtl, &block);  %does not
               exist%
               END;
            %get first entry in next block if block exists%
               IF NOT block.dbnxt THEN
                  RETURN(FALSE, &entry - strtl, &block); %no new block%
               &entry _ (&block _ block.dbnxt) + dspbhl;
               RETURN(TRUE, &entry, &block);
            END;
         =backward:
            BEGIN
            &entry _ &entry - strtl;
            IF &entry >= &block + dspbhl THEN RETURN(TRUE, &entry,
            &block);
            &entry _ &entry + strtl; %restore previous value%
            IF NOT block.dprev THEN RETURN(FALSE, &entry, &block);
            &block _ block.dprev;
            &entry _ block.delst; %last used%
            RETURN(TRUE, &entry, &block);
            END;
         ENDCASE err($"FATAL DISPLAY SYSTEM ERROR...
         Type ^C and RESET");
         END.

%...special da functions...%
   (clrall) % *** % PROCEDURE (da REF, free);  %reset LSRT entries for
   the da                                                          8A
      In general, if da=0 then these happen for all random display
      areas.  If free = TRUE, free strings.%
      %---------------%
      LOCAL
         daend,
         stblk REF;
      IF NOT &da THEN
         daend _ (&da _ $dpyarea) + dacnt * dal
      ELSE
         daend _ &da + dal;
      UNTIL &da >= daend
      DO
         BEGIN
         IF da.daexis AND da.dawid AND NOT da.daseq AND NOT
         da.daauxiliary THEN
            BEGIN
            IF NOT (&stblk _ da.dastrt) THEN
               BEGIN
               &da _ &da + dal;
               REPEAT LOOP; % first string table block for this da. %
               END;
            distbl( &stblk, free );
            da.dacrow _ da.daccol _ da.dacurallo _ da.dastrt _ 0;
            END;
         &da _ &da + dal;
         END;
```

```
    RETURN;
    END.

(distbl) PROCEDURE % delete string table and associated line segment
tables; if FREE TRUE, also delete allocated strings of signatures
and statement numbers %                                              8B
    (stblk REF,
    free);
    LOCAL stindex REF, ls REF, lsblock, fstblk REF;
    % delete string tables %
    &fstblk _ &stblk;
    &stindex _ &stblk + dspbhl;
    %free storage associated with strings%
       LOOP % over strings in this da %
          BEGIN
          &ls _ &stindex;
          % loop over line segments in string freeing string storage
          associated with allocated strings %
             DO
                IF free AND NOT ls.stkeep AND (ls.stsrce = srcstno OR
                ls.stsrce = srcsig) THEN
                   freestring (ls.ststid.stpsid, $dspblk)
             WHILE dnxtls(&ls, lsblock: &ls, lsblock);
          % Free line segments associated with this string %
             IF stindex.stlsptr THEN dbfree( stindex.stlsptr := 0);
             stindex.stcurallo _ 0;
          % advance to the next string in the da; exit if we are
          through with this da. %
             IF NOT dgetnxt( &stindex, &stblk, forward : &stindex,
             &stblk ) THEN
                EXIT LOOP;
          END;
    % free string table %
       dbfree( &fstblk );
    RETURN;
    END.

(clearda)% *** %  PROCEDURE(da REF, commands REF LIST); %clear
display area%                                                        8C
    %If &da is zero then zap the display image of all of the text
    display areas.  Otherwise, zap only the given text display area
    image.%
    % IMPORTANT:  Call clrall after clearda to free up the string and
    line segment tables and to delete allocated strings.  (Perhaps it
    can be incorporated in here?) %
    %----------------%
    LOCAL
       end;
    IF NOT &da THEN
       BEGIN
       end _ (&da _ $dpyarea) + dacnt*dal;
       END
    ELSE
       BEGIN
       end _ &da + dal;
       END;
```

```
    DO
        BEGIN  %build FE clear window command%
        IF da.daexis AND da.dawid THEN
            BEGIN
            % CLRALL must be called to delete internal data structures.
            %
            bldclw( findwa(da.dawid), &commands);
            END;
        END
    UNTIL (&da _ &da+dal) >= end;
    RETURN;
    END.

(supda) % *** % PROCEDURE(da REF, commands REF LIST); %suppress
display area%                                                    8D
    %If &da is zero then suppress the display image of all of the
    text display areas.  Otherwise, suppress only the given text
    display area image.%
    %----------------%
    LOCAL end;
    LOCAL LIST procpar[2], windatt[3];
    IF NOT &da THEN
        BEGIN
        end _ (&da _ $dpyarea) + dacnt*dal;
        END
    ELSE
        BEGIN
        end _ &da + dal;
        END;
    DO
        IF da.daexis AND NOT da.daseg AND da.dawid AND NOT
        da.daauxiliary THEN
            BEGIN
            da.dasuppress _ TRUE;
            % assume commands list passed and that this will be
            executed by batch commands.  we should permit immediate
            execution; maybe permit &commands to be 0 in which case wee
            execute now?  If there is not enough room in the FE to
            maintain the storage, we must do something differently %
            #windatt# _
                USE makedesc( unull, 0, FALSE),
                USE makedesc( uindex, 5 %invisible%, FALSE),
                USE makedesc( unull, 0, FALSE);
            #procpar# _
                USE makedesc( uindex, da.dawid, FALSE ), % window id %
                USE makedesc( ulist, $windatt, FALSE);
            addtobatch( &commands, $procpar, setwindatt );
            [findwa( da.dawid )].wiatt _ 5; %invisible %
            END
    UNTIL (&da _ &da+dal) >= end;
    NULL-LISTS;
    RETURN;
    END.

(resda) % *** % PROCEDURE (da REF, commands REF LIST); %restore
display area%                                                    8E
```