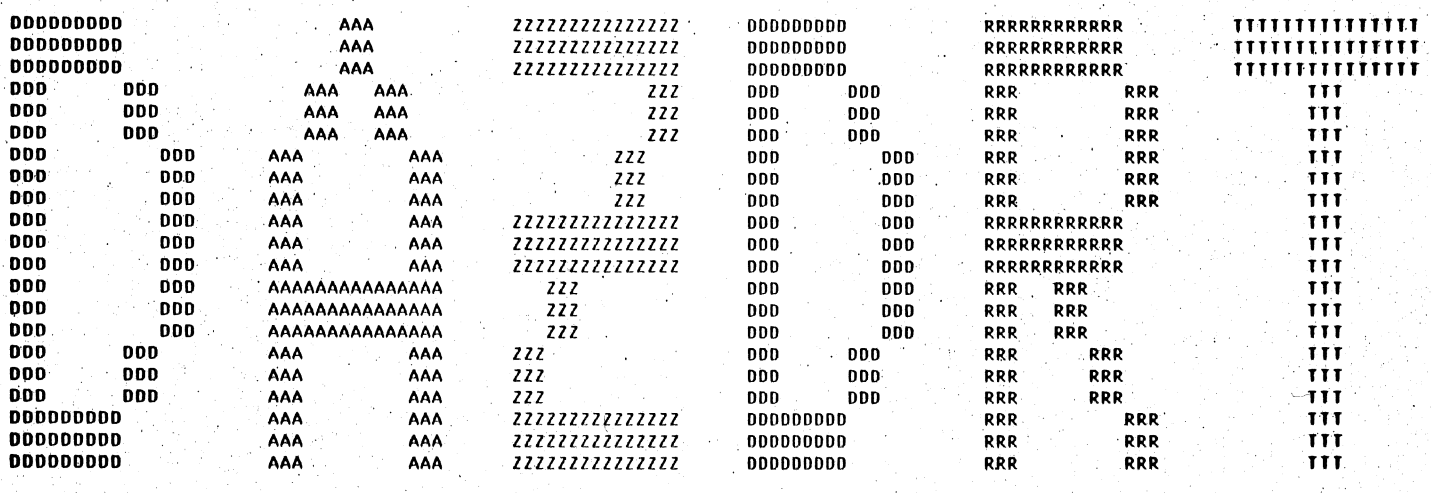
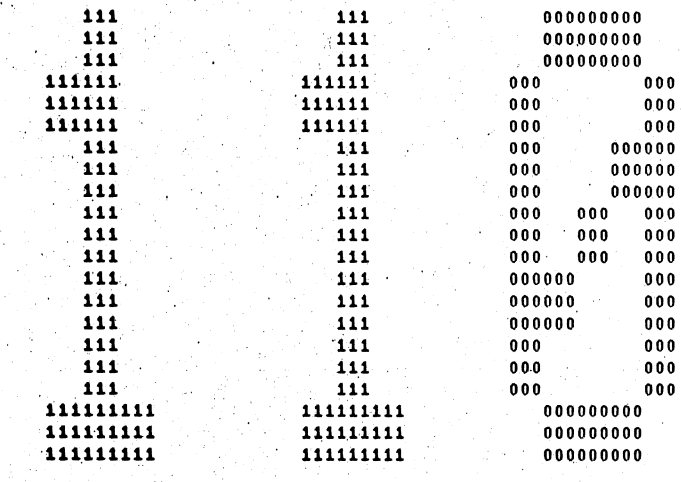
AI:KLH; CBF 07:17:44 TUEDAY, ER 23,1976 LQ+11H.9M.17S. CREATED MARCH 23,1976 06:56:54



AI:KLH; CBF 07:17:44 TUEDAY, ER 23,1976 LQ+11H.9M.17S. CREATED MARCH 23,1976 06:56:54



SWITCH SETTINGS: L[MIDAS] % A C M 85V 120W X Z
FONTS: F[FONTS;18FG KST,,]

```
001                     title dazdrt
002                     subtti Accumulator Definitions and Assembly Parameters
003                     .milit==1
004
005                     f=0     ;flag register
006                     a=1
007                     b=2
008                     c=3
009                     d=4
010                     e=5
011                     t=6     ;temps
012                     tt=7
013                     pwb=10  ;index to player with beam (always !!!!)
014                     plr=11  ;player index (random)
015
016                     DP=14   ;DISP BLKO PTR
017                     DL=15   ;DISP LIST PTR
018                     ia=16   ;APR interrupt acc
019                     p=17    ;pdl acc
020
021                     nplyrs==4       ;# of players
022
023
024             DEFINE SETF TEXT,FLG            ;useful set-flag macro!
025  001 028    IFDEF FLG,.STOP
026             .TAG FOOBAR
027             PRINTC "TEXT
028             FLG="
029             .TTYMAC FLAG
030             IFSE FLAG,YES,FLG==1
031             IFSE FLAG,NO,FLG==0
032             IFSE FLAG,Y,FLG==1
033             IFSE FLAG,N,FLG==0
034  001 028    IFNDEF FLG,FLG==FLAG
035             TERMIN
036  001 028    IFNDEF FLG,.GO FOOBAR
037             TERMIN
038
039  001 024            SETF [Run under ITS?]$ITS
```

```
001
002                     subttl Processor state definitions
003             ;---------- bit definitions ------------------
004             ;produces BIT2.3, BIT35 type defs
005             radix 10.
006             define bitdef a,b,c
007   001 007   bit!a!.!b==1_<<a-1>*9.+<b-1>>
008   001 008   bit!c==1_<35.-c>
009             termin
010             %%3==-1
011   002 010   repeat 4,[%%1==.rpcnt+1 ? repeat 9.,[%%2==.rpcnt+1 ? %%3==%%3+1
012   002 010   bitdef \%%1,\%%2,\%%3
013             ]]
014             radix 8.
015
016             ; ------ PDP-6 APR condition flags
017             ;CONO
018                     A%rpov==bit18   ;reset the PDL OV flag
019                     A%rio== bit19   ;I/O reset
020                     A%rmp== bit22   ;reset the Memory Protection flag
021                     A%rnxm==bit23   ;reset the nonexistent Memory flag
022                     A%ccez==bit24   ;turn the Clock Count Enable flag off
023                     A%cceo==bit25   ;turn the Clock Count Enable flag on
024                     A%clkz==bit26   ;turn the Clock flag off
025                     A%pcez==bit27   ;turn the PC Change Enable flag off
026                     A%pceo==bit28   ;turn the PC Change Enable flag on
027                     A%pcfz==bit29   ;turn the PC Change flag off
028                     A%ovez==bit30   ;turn the OV flag enable off
029                     A%oveo==bit31   ;turn the OV flag enable on
030                     A%ovfz==bit32   ;turn the OV flag off
031                     ;33-35 = assign PI channel to APR flags (listed above)
032                             ;define name to reset/clear all flags and i/o
033   002 030           a%STRT==a%RPOV+a%RIO+a%RMP+a%RNXM+a%CCEZ+a%CLKZ+a%PCEZ+a%PCFZ+a%OVEZ+a%OVFZ
034             ;CONI
035                     A%pov== bit18   ;PDL OV flag set
036                     A%ilop==bit22   ;Illegal Instruction flag set
037                     A%nxm== bit23   ;non-existent Memory flag set
038                     A%cce== bit25   ;Clock Count enable on
039                     A%clk== bit26   ;Clock Count flag set
040                     A%pce== bit28   ;PC Change enable on
041                     A%pcf== bit29   ;PC Change flag set
042                     A%ove== bit31   ;OV enable on
043                     A%ovf== bit32   ;OV flag set
044                     ;33-35 = set to the current PI channel assignment
045
046
047             ;-------- PDP-6 Priority Interrupt system flags
048             ;CONO
049                     P%clr== bit23   ;clear the PI system
050                     P%act== bit24   ;activate interrupt on channels selected (29-35)
051                     P%cenb==bit25   ;enable channels selected (")
052                     P%cdsb==bit26   ;disable channels selected (")
053                     P%off== bit27   ;turn off the PI system
054                     P%on==  bit28   ;turn on the PI system
055                             ;29-35 = channel select: bit 29 selects channel 1,
056                             ;  bit 35 selects channel 7, etc.
057             ;CONI
058                     P%piup==bit28   ;PI system is on
059                     ;29-35 = if a bit is 1, corresponding channel is on.
060
061             ;---------- 340 display flags
062             ;CONO
063                     D%init==bit1.7  ;initialize display
064                     D%cont==bit1.8  ;resume display after special interrupt
065
066
067             ;---------- Flag register (F)
068
069                     %pfneg==bit1.1  ;for PFDVR,PFMPR etc
070                     %m1inf==bit1.2  ;for intrsc routine
071                     %m2inf==bit1.3  ;       "
072                     %zfire==bit1.4  ;set when beam being zapped
073                     %inchk==bit1.5
074                     %dexch==bit1.6  ;for drwlin routine
075                     %rstrt==bit1.7  ;indicates restart from lossage, dont reset scores
076
077             ;-----------------------------------------
078
079             aprchn==4       ;processor has highest priority pi channel
080             dspchn==5       ;special interrupt channel for display
081             dischn==6       ;normal 'done' channel for display
082             imxchn==7       ;IMX data channel (pots)
083             imx==574
084             ..d574==574
```

```
085
086                 aprpic==bit1.3  ;channels as represented in PI condition word
087                 dsppic==bit1.2
088                 dispic==bit1.1
089
090                 IFE $ITS,[
091   002 079       loc aprchn*2+40
092   004 005               jsr aprbrk        ;processor (clock) interrupt vector
093   002 080       loc dspchn*2+40
094   004 057               jsr dspbrk        ;special interrupt on display
095   002 081       loc dischn*2+40
096   001 016               blko dis,DP       ;display output instr for PI system
097   004 046               jsr disbrk        ;int vector when BLKO done
098                 ]
099                 IFN $ITS,[
100                 LOC 42
101   004 033               JSR TSINT
102                 ]
103
104                 loc 100
105   002 075               trza f,%rstrt     ;set that this is not a restart
106   002 075                tro f,%rstrt     ; if started at 10, this is a restart
107   005 004               jrst go
108
109         .       patch:  block 100
110
111                 pdllen==20
112   002 112       pdl:    -pdllen,,pdl
113   002 111               block pdllen
114
115                 cpie==3.1415926535
116                 cpie.5==1.5707963
117                 cpi1.5==4.7123889
118                 PIE:    3.1415926535
119                 PI$2:   6.2831853072
120                 PI.5:   1.5707963
121
```

1 ← <7 - channel #>

```
001
002                     subttl Macro definitions
003             maxflo: 377777,,-1      ;maximum floating point value
004             JOV=<jfcl 10,>          ;jump on overflow
005
006             DEFINE PFMPR ac,loc
007   003 004           jov .+1
008                     fmpr ac,loc
009   003 004           jov [caige ac,0
010   002 069                 troa f,%pfneg
011   002 069                 trz f,%pfneg
012                           movm ac,ac
013                           caml ac,[177400,,0]
014                           jrst [setz ac, ? jrst .+1]
015   003 003                 move ac,maxflo
016   002 069                 trze f,%pfneg
017                           movn ac,ac ? jrst .+1]
018             TERMIN
019
020             DEFINE PFDVR ac,loc
021   003 004           jov .+1
022                     skipn loc
023                     jrst [call ac,0
024   003 003                 skipa ac,maxflo
025   003 003                 movn ac,maxflo ? jrst .+3]
026                     fdvr ac,loc
027   003 004           jov [caige ac,0
028   002 069                 troa f,%pfneg
029   002 069                 trz f,%pfneg
030                           movm ac,ac
031                           caml ac,[201400,,0]
032                           jrst [setz ac, ? jrst .+1]
033   003 003                 move ac,maxflo
034   002 069                 trze f,%pfneg
035                           movn ac,ac ? jrst .+1]
036             TERMIN
037
038             DEFINE PFADR ac,loc
039   003 004           jov .+1
040                     fadr ac,loc
041   003 004           jov [call ac,0
042   003 003                 skipa ac,maxflo
043   003 003                 movn ac,maxflo
044                           jrst .+1]
045             TERMIN
046             ;find intersection of lines specified and leave xi,yi in a,b
047             ;xi= (b1-b2)/(m2-m1)
048             ;yi= m1*xi+b1
049
050             ;where b=y-mx
051
052             siplim: 1.0^30  ;criterion for 'infinite'; maxflo is approx 1.E38
053
054             DEFINE INTRSC DM1,DX1,DY1,DM2,DX2,DY2
055   001 006           move a,DM1
056   003 072           movem a,m1
057   001 006           move a,dx1
058   003 070           movem a,x1
059   001 006           move a,dy1
060   003 071           movem a,y1
061   001 006           move a,dm2
062   003 075           movem a,m2
063   001 006           move a,dx2
064   003 073           movem a,x2
065   001 006           move a,dy2
066   003 074           movem a,y2
067   003 077           pushj p,intcal
068             TERMIN
069
070             x1:     0
071             y1:     0
072             m1:     0
073             x2:     0
074             y2:     0
075             m2:     0
076
077   001 008   intcal: push p,c
078   001 009           push p,d
079   001 010           push p,e
080   002 071           trz f,%m1inf+%m2inf
081
082   003 072           movm c,M1
083   003 052           caml c,siplim
084   002 070           troa f,%m1inf
```

```
085  003 072          jrst [movn c,M1
086  003 070                fmpr c,X1
087  003 071                fadr c,Y1       ;B1=Y1-M1X1
088                         jrst .+1]
089  003 075          movm d,M2
090  003 052          caml d,slplim
091  002 071          troa f,%m2inf
092  003 075          jrst [movn d,M2
093  003 073                fmpr d,X2
094  003 074                fadr d,Y2       ;B2=Y2-M2X2
095                         jrst .+1]
096
097  003 072          move b,M1
098  001 007          move e,b
099  003 075          fsbr b,M2       ;M1-M2
100                         ;(b)=M1-M2 (c)=B1 (d)=B2 (e)=M1
101  002 071          trnn f,%m1inf+%m2inf    ;either infinite?
102  003 120          jrst intrs2             ;no, normal crunching.
103  001 006          setz a,
104  002 070          trne f,%m1inf
105  002 071          jrst [trne f,%m2inf     ;1st infinite, 2nd also?
106  003 133                jrst intrs9       ;yes, no intersection
107  003 070                move a,X1
108  003 075                move b,M2
109  003 070                fmpr b,X1
110  001 009                fadr b,d        ;Y=B2+M2*X1
111            ;            move b,d        ;X=X1, Y=B2 if M1 infinite
112  003 128                jrst intrs7]
113  003 073          move a,X2
114  003 073          move b,X2
115  003 072          fmpr b,M1
116  001 008          fadr b,c
117            ;      move b,c                ;X=X2, Y=B1 if M2 infinite
118  003 128          jrst intrs7
119
120  003 133  intrs2: jumpe b,intrs9  ;if M1-M2 then lines parallel
121  001 009          move a,d
122  001 008          fsbr a,c        ;(B2-B1)
123  001 007          PFDVR a,b       ;(B2-B1)/(M1-M2) = X
124  001 006          move b,a
125  001 010          PFMPR b,e       ;X*M1
126  001 008          fadr b,c        ;X*M1+B1 = Y
127
128  001 010  intrs7: pop p,e
129  001 009          pop p,d
130  001 008          pop p,c
131  001 019          aos (p)
132  001 019          popj p,
133  001 010  intrs9: pop p,e
134  001 009          pop p,d
135  001 008          pop p,c
136  001 019          popj p,
137
138
139
140            define conc a,b
141  001 007   a!b!termin
142
143            define ssfix a,b
144  001 006          muli a,400
145  001 006          tsc a,a
146  001 006          ash a+1,-243+19.!b(a)
147            termin
148
149            ;floating to integer conversion -- works for pos/neg
150            define ifix a
151  001 006          push p,a+1
152  001 006          ssfix a,-19.
153  001 006          move a,a+1
154  001 006          pop p,a+1
155            termin
156
157            ;floating to fractional integer conversion; integer in LH, fraction in RH
158            define frifix a
159  001 006          push p,a+1
160  001 006          ssfix a,-1
161  001 006          move a,a+1
162  001 006          pop p,a+1
163            termin
```

```
001
002                     subttl interrupt handlers
003             IFE $ITS,[
004             ;processor interrupt handler
005             aprbrk: 0
006   001 018           coni apr,ia       ;get apr conditions into interrupt acc
007   002 039           trnn ia,A%clk     ;clock interrupt?
008   004 019           jrst aprbr2       ;no, don't do clock hackery...go check for bad news
009
010                     ;clock interrupt
011   004 025           sosle clkcnt      ;count off one tick
012   004 016           jrst aprbr1       ;and don't unset lock if haven't finished countdown
013   004 026           setom clksnk      ;ah! unset synchronization lock
014   004 024           move ia,steptm    ;get # tics per game step
015   004 025           movem ia,clkcnt   ;and reset count.
016   002 079   aprbr1: cono apr,a%cceo+A%clkz+aprchn   ;turn off clock flag and make sure enabled
017   004 005           jrst 12,@aprbrk           ;and dismiss interrupt
018
019   004 023   aprbr2: movem ia,aprcns
020   004 029           jsr death         ;for time being, all non-clock ints are no-no's...
021             ]
022
023             aprcns: 0       ;apr conditions if die
024             steptm: 1       ;duration of a game step in ticks (1 tick= 1/60 sec)
025             clkcnt: 0       ;tick countdown; when steptm ticks done, setom clksnk.
026             clksnk: 0       ;synch lock; each game step must wait until set to -1
027
028             DDT==34000      ;SA of DDT when present in 6.
029             death:  0       ;JSR'd here so can tell where came from.
030   004 028           jrst 4,DDT        ;stop with PC->DDT so 'continue' gets DDT
031
032             IFN $ITS,[
033             TSINT:  0
034                     0
035   004 033           SKIPGE IA,TSINT
036   004 033            .DISMIS TSINT+1
037   001 018           TLNN IA,200000
038   004 033            .DISMIS TSINT+1
039   004 026           SETOM CLKSNK
040                     .SUSET [.SAMASK,,[200000,,0]]
041   004 033           .DISMIS TSINT+1
042             ]
043
044
045             IFE $ITS,[
046             DISBRK: 0
047   001 017   DISBR0: TRNN DL,-1        ;ADDR OF NEXT ITEM IN RH?
048   018 046            MOVE DL,DISLST ;NO, GET PTR TO BEG OF CURRENT DISLIST
049   001 017           MOVE DL,(DL)      ;GET ITEM
050   001 017           HLRZ DP,DL        ;GET ADDR OF BLKO PTR FOR ITEM
051   004 047           JUMPE DP,DISBR0 ;NOTHING THERE, GET NEXT ITEM
052   001 016           SKIPL DP,(DP)     ;GET BLKO PTR
053   004 047            JRST DISBR0     ;AGAIN NOTHING THERE, GET NEXT.
054   004 046           JRST 12,@DISBRK ;RETURN
055
056
057             DSPBRK: 0                 ;SPECIAL DISPLAY INTERRUPT
058             ;       HLRZ DP,DL        ;GET ADDR OF BLKO THAT INTERRUPTED
059             ;       CAIN DP,BKPT1     ;IF WAS DOING FIRST BACKGROUND,
060             ;        MOVE DL,(DL)     ;SKIP NEXT (2ND BACKGND) ITEM.
061   001 017   DSPBR1: TRNN DL,-1
062   018 046            MOVE DL,DISLST
063   001 017           MOVE DL,(DL)
064   001 017           HLRZ DP,DL
065   004 061           JUMPE DP,DSPBR1
066   001 016           SKIPL DP,(DP)
067   004 061            JRST DSPBR1
068   002 081           CONO DIS,D%INIT+DSPCHN_3+DISCHN
069   004 057           JRST 12,@DSPBRK ;RETURN
070
071             DEFINE DSTART
072   018 046   MOVE DL,DISLST
073   001 016   MOVE DP,[-1,,[0]-1]
074   002 081   CONO DIS,D%INIT+<DSPCHN_3>+DISCHN
075             TERMIN
076             ]
077
078             IFN $ITS,[
079             DEFINE DSTART
080   018 046   .DSTART DISLST
081             .VALUE [ASCIZ /:$DISPLAY NOT AVAIL$
082             /]
083             TERMIN
084             ]
```

```
085
086  013 059  vbfend: pushj p,disran  ;display random stuffs
087  018 036         MOVE B,VDSW
088  018 054         move t,vdlipt    ;get addr to end of vlist currently being written
089  001 006         movei a,3000     ;get stop command
090  001 011         movem a,(t)      ;terminate vlist
091  001 011         HRRZ T,T
092  018 035         HRRZ A,VDCUR
093  001 006         SUBI T,A
094  001 011         CAIGE T,
095  001 011          SETZ T,
096  001 011         MOVN T,T
097  001 011         HRLZ T,T
098  018 035         HRR T,VDCUR
099  018 048         MOVEM T,VDPTR+1(B)
100  018 041         MOVEI A,DSTL+1(B)
101  018 046         MOVEM A,DISLST
102  018 036         SETCAB B,VDSW
103  018 037         MOVE A,VDPTAB+1(B)
104  018 035         MOVEM A,VDCUR
105  001 006         ADDI A,1
106  018 054         HRRZM A,VDLIPT
107  001 019         POPJ P,
108
```

```
001
002                     subttl main program
003                     ;one time initialization
004   002 112   go:     move p,pdl
005             IFE $ITS,[
006   002 079           cono apr,a%STRT+aprchn          ;clear all APR flags and i/o
007   002 079           cono apr,a%cceo+aprchn  ;enable clock ints
008                     cono 420,40     ;enable 36-bit input array
009   002 087           cono pi,p%clr+p%cenb+p%on+aprpic+dispic+dsppic  ;PI reset+assignments
010             ]
011             IFN $ITS,[
012   004 024           MOVE A,[600000,,STEPTM]
013   001 006           .REALT A,
014                     JFCL
015                     .SUSET [.SMASK,,[200000,,0]]
016                     .SUSET [.SPICLR,,[-1]]
017   002 082           .OPEN IMXCHN,[.BII,,'IMX]
018   002 083            .VALUE [ASCIZ /:$ OPEN OF IMX FAILED? $
019             /]
020             ]
021
022                     ;series initialization
023   009 012   series: pushj p,bsetup  ;set up board if necessary
024             IFE $ITS,[
025   002 063           cono dis,d%init ;reset/initialize display
026             ]
027
028   001 021           movsi plr,-nplyrs
029   008 051   ser1:    setzm plscor(plr)       ;whatever.
030   008 030           move a,splen    ;standard player length
031   008 047           movem a,plen(plr)
032   001 006           fmpr a,a
033   008 049           movem a,plensq(plr)
034   005 029           aobjn plr,ser1
035   002 075           trne f,%rstrt
036   005 041            jrst game      ; if this is restart, don't zero the scores
037   008 080           setzm tmscor    ;zero scores
038   008 080           setzm tmscor+1
039
040                     ;game initialization
041   001 005   game:   setzm f
042   010 043           pushj p,zassgn  ;decide which player gets beam (initialize PWB)
043   008 083           move a,ztime
044   008 084           movem a,ztmlft
045   001 006           imuli a,60.
046   008 085           movem a,ztim60
047   008 032           setzm a,passtm  ;make sure pass timer is zero
048   001 021           movsi plr,-nplyrs
049   008 010   game1:  move a,pinitx(plr)
050   008 035           movem a,plocx(plr)
051   008 014           move a,pinity(plr)
052   008 037           movem a,plocy(plr)
053   008 018           move a,pinita(plr)
054   008 039           movem a,plang(plr)
055   008 023           setzm input(plr)
056   012 003           pushj p,updat
057   013 004           pushj p,plrdis  ;update player (plr) and display him
058   005 049           aobjn plr,game1
059
060             ;       movei a,distrt-1        ;set up blko ptr
061             ;       movem a,displtr
062             ;       setzm vdlflg
063             ;       setzm vdlloc
064             ;       setom vdlapr
065
066   018 036           MOVE B,VDSW
067   018 037           HRRZ a,VDPTAB+1(B)
068   018 035           MOVEM A,VDCUR
069   001 006           ADDI A,1
070   018 054           movem a,vdlipt
071   018 017           MOVE A,DBDPT
072   018 050           MOVEM A,BDPT
073   018 044           MOVEI A,DSTL3
074   018 046           MOVEM A,DISLST
075   004 071           DSTART
076             ;       cono dis,d%init+dischn+dspchn_3 ;start display with PI assigned
```

```
001
002                     subttl main processing loop
003
004   004 086  gamlup: pushj p,vbfend  ;switch var. display buffers
005   004 026         aose clksnk     ;wait for clock synch
006             IFE $ITS,[
007                     jrst .-1
008   001 006         datai a ;for DDT hacking in the 6
009   001 006         trze a,1
010             break:  jfcl
011             ]
012             IFN $ITS,[
013                     .HANG
014   001 006         MOVE A,[-2,,[.SPIRQC,,[0] ? .SIMASK,,[200000,,0]]]
015   001 006         .SUSET A
016             ]
017
018   011 014         pushj p,inpget  ;get inputs for all players
019
020   001 021         movsi plr,-nplyrs
021   012 003         pushj p,updat
022   001 014         aobjn plr,.-1
023
024   006 024         setcmm ransw'
025   008 085         move a,ztim60   ;beam-possession time left in 60th's
026   004 024         sub a,steptm
027   006 038         jumpge a,gamlp0
028   001 021         movsi a,-nplyrs
029   008 058  gaml01: move b,tmon(a)
030   008 058         camn b,tmon(pwb)
031   006 029          aobjn a,gaml01
032   006 024         skipe ransw
033   001 006          addi a,1
034   001 006         movei pwb,(a)
035                     setzm passtm;when beam changes hands, pass timer is 0
036   008 083         move a,ztime
037   001 006         imuli a,60.
038   008 085  gamlp0: movem a,ztim60
039   001 006         addi a,59.
040   001 006         idivi a,60.
041   008 084         movem a,ztmlft  ;time left in seconds
042
043   001 013         move a,pwb      ;get index to player with beam * 4
044   001 006         ash a,2
045
046   008 032         skipg b,passtm  ;has the passing timer run out yet?
047   006 050         jrst tstpas
048   004 024          sub b,steptm   ;subtract the number of 60ths gone by from it
049   008 032          movem b,passtm
050   011 090  tstpas: skipl inval+fire(a)     ;is he passing beam?
051   006 056         jrst tstfir
052   008 053          move pwb,tmmate(pwb)   ;if so, transfer.
053   008 031          move b,paslim  ; set pass timer going
054   008 032          movem b,passtm
055
056   001 013  tstfir: move a,pwb      ;note that pwb may have changed...this is the idea.
057   001 006         ash a,2
058   011 090         skiple inval+fire(a)    ;is PWB firing?
059   002 072         troa f,%zfire   ;yes, set global flag and skip
060   002 072         trz f,%zfire    ;clear flag otherwise
061
062   001 021         movsi plr,-nplyrs
063   013 004  gamlp1: pushj p,plrdis          ;display new position etc.
064   006 063         aobjn plr,gamlp1
065
066   002 072         trnn f,%zfire   ;now see if beam was fired
067   006 004         jrst gamlup     ;nope, just loop back and wait.
068
069             ;aha! if skipped over to here, means must compute beam path!
070             ;first must initialize beam variables
071
072   008 135         setzm zbounc    ;# of bounces
073   008 039         move a,plang(pwb)        ;angle of beam
074   002 120         fadr a,pi.5     ;get angle+90 deg
075   002 119         caml a,pi$2     ;normalize
076   002 119         fsbr a,pi$2
077   008 124         movem a,zangle
078   008 041         move a,plsin(pwb)
079   008 126         movnm a,zcos    ;cos (ang+90)= -sin (ang)
080   008 043         move a,plcos(pwb)
081   008 125         movem a,zsin    ;sin (ang+90)= cos(ang)
082   008 035         move a,plocx(pwb)
083   008 122         movem a,zstrtx
084   008 037         move a,plocy(pwb)
```

```
085  008 123          movem a,zstrty
086  008 125          move a,zsin
087  008 126          PFDVR a,zcos     ;find (possibly infinite) slope
088  008 127          movem a,zslope
089  008 129          movem pwb,zfrom ;indicate which player beam coming from
090  002 073          trz f,%inchk     ;clear check-boundary-lines flag.
091
092                   ;loops once for each straight line path
093  001 021  zlup:   movsi plr,-nplyrs
094  008 131  zlup0:  setom zbestd
095  008 132          setom zbestp     ;clear locs which save closest termination of this path
096
097                   ;loops thru each player seeing if this path hits anyone
098  001 014  zlup1:  hrrz a,plr       ;get # of player being checked
099  008 129          camn a,zfrom     ;same as player beam is coming from?
100  006 138          jrst zlup50      ;yep, don't check this one.
101
102  008 037          intrsc zslope,zstrtx,zstrty,plslop(plr),plocx(plr),plocy(plr)
103  006 138          jrst zlup50      ;no skip if no intersection
104
105  002 073          trne f,%inchk
106  006 114          jrst zlup2       ;skip within-ness check if doing boundaries
107  008 088          caml a,blowx     ;check if intersection within board limits
108  008 089          camle a,bhighx
109  006 138          jrst zlup50      ;out-of-bounds
110  008 090          caml b,blowy
111  008 091          camle b,bhighy
112  006 138          jrst zlup50      ;out
113
114  007 013  zlup2:  pushj p,angchk
115  006 138          jrst zlup50      ;not pointing in right direction
116
117  002 073          trne f,%inchk
118  006 125          jrst zlup25      ;skip hit-it check if doing boundaries
119  008 035          move c,plocx(plr)        ;see if intersction is close enough to player to hit him
120  008 037          move d,plocy(plr)        ;set up to find dist
121  007 004          pushj p,distsq  ;leave result in c
122  008 049          camle c,plensq(plr)      ;compare with player radius squared
123  006 138          jrst zlup50      ;no skip=nope
124
125  008 122  zlup25: move c,zstrtx    ;find distance from start of this path to the intersection.
126  008 123          move d,zstrty
127  007 004          pushj p,distsq  ;find distance (squared) in c
128  008 131          skipge zbestd    ;have any hits already?
129  006 133          jrst zlup3       ;nope, skip the closeness check
130  008 131          caml c,zbestd    ;see if it's any closer than closest so far
131  006 138          jrst zlup50      ;nope, forget about it
132
133  008 131  zlup3:  movem c,zbestd  ;ah! closer...store its vars
134  008 133          movem a,zbestx
135  008 134          movem b,zbesty
136  008 132          hrrzm plr,zbestp
137
138  006 098  zlup50: aobjn plr,zlup1 ;loop on thru players
139  008 132          skipge plr,zbestp        ;found any hits?
140  002 073          jrst [troe f,%inchk      ;nope, go check board boundaries.
141  004 029                jsr death         ;couldn't find boundary hit
142  001 021                move plr,[-4,,nplyrs]
143  006 094                jrst zlup0]
144  002 073          trne f,%inchk
145  006 179          jrst zlup90      ;boundary hit!
146                   ;found a solid hit on a player. now must bounce it off.
147  008 122          move a,zstrtx    ;draw line from starting pt
148  008 123          move b,zstrty
149  008 133          move c,zbestx    ;to bounce pt
150  008 134          move d,zbesty
151  015 055          pushj p,drwlin   ;do it
152
153  008 135          aos a,zbounc     ;increment bounce cnt
154  008 137          hrrzm plr,zbncit-1(a)    ;record the player bounced off of
155  008 136          cail a,zmaxbc    ;skip if not yet reached max # bounces.
156  008 132          jrst [move pwb,zbestp    ;ah! beam ownership transfers!
157  008 032                setzm passtm      ;zero the pass timer
158  008 083                move a,ztime      ;but start a new ownership timout
159  008 084                movem a,ztmlft
160  001 006                imuli a,60.
161  008 085                movem a,ztim60
162  006 004                jrst gamlup]
163  008 122          movem c,zstrtx  ;else do again, relative to new starting place
164  008 123          movem d,zstrty
165  007 039          pushj p,reflct   ;find reflected angle of beam
166  008 124          movem a,zangle   ;store
167  001 006          push p,a
168  007 054          pushj p,cos      ;now get vars related to angle.(sin,cos,tan)
```

```
169  008 126          movem a,zcos
170  001 006          pop p,a
171  007 055          pushj p,sin
172  008 125          movem a,zsin
173  008 126          PFDVR a,zcos
174  008 127          movem a,zslope
175  008 129          hrrzm pir,zfrom
176  006 093          jrst zlup       ;now do another straight beam path....
177
178                   ;boundary hit!
179  008 122  zlup90: move a,zstrtx
180  008 123          move b,zstrty
181  008 133          move c,zbestx
182  008 134          move d,zbesty
183  015 055          pushj p,drwlin  ;draw final leg
184
185                   ;now determine if hit a goal...
186  008 135          skipg zbounc    ;bounced off someone?
187  006 004          jrst gamlup     ;no, back to game.
188  001 014          movei pir,-4(pir)       ;adjust index (not pointing into player tables now)
189  008 099          skipn bglflg(pir)       ;is line a goal line?
190  006 004          jrst gamlup             ;nope. loop back
191  008 133          move a,zbestx   ;get x coord of hit
192  008 104          camle a,bglow(pir)      ;test low value coordinate of boundary
193  008 105          caml a,bghigh(pir)      ;test high  "
194  006 004          jrst gamlup     ;if no hit
195
196                   ;aha! hit goal! whoopee etc...
197  010 011          pushj p,goalht  ;go do whatever
198  005 041          jrst game       ;back to another game
```

```
001
002                     subttl calculational subroutines
003
004  001 009  distsq: push p,d
005  001 006          fsbr c,a
006  001 007          fsbr d,b
007  001 008          fmpr c,c
008  001 009          fmpr d,d
009  001 009          fadr c,d
010  001 009          pop p,d
011  001 019          popj p,
012
013  001 008  angchk: push p,c
014  001 009          push p,d
015  001 010          push p,e
016  001 011          push p,t
017  001 006          move c,a        ;x coord of hit
018  001 007          move d,b        ;y coord of hit
019  008 122          fsbr c,zstrtx   ;get coords relative to start of beam
020  008 123          fsbr d,zstrty
021  001 008          movm e,c
022  001 009          movm t,d
023  001 011          camg e,t
024  008 125          jrst [move c,d ? xor c,zsin ? jumpge c,angchw ? jrst angchi]
025  008 126          xor c,zcos
026  007 032          jumpge c,angchw
027  001 011  angchi: pop p,t
028  001 010          pop p,e
029  001 009          pop p,d
030  001 008          pop p,c
031  001 019          popj p,
032  001 011  angchw: pop p,t
033  001 010          pop p,e
034  001 009          pop p,d
035  001 008          pop p,c
036  001 019          aos (p)
037  001 019          popj p,
038
039  008 039  reflct: move a,plang(plr)       ;get ang2
040  001 006          fmpr a,[2.0]    ;2*ang2
041  008 124          fsbr a,zangle   ;reflected angle = 2*ang2 - ang1
042  002 119          jumpl a,[fadr a,pi$2 ? popj p,]
043  002 119          caml a,pi$2
044  002 119          fsbr a,pi$2
045  001 019          popj p,
046
047
048           ;FLOATING POINT SINE AND COSINE.  REENTERABLE.
049
050  001 006  SIND:   FMPR A,[.01745329251994]        ;PI/180
051  007 055          JRST SIN
052
053  001 006  COSD:   FMPR A,[.01745329251994]
054  007 087  COS:    FADR A,SC1      ;PI/2
055  007 091  SIN:    CAMG A,SC9      ;.000211431983 IS SUFFICIENT FOR IDENTITY, 10**-15 IS NECESSARY NOT TO UNDERFLOW
056  001 006          CAMGE A,[-.000211431983]        ;ABS X MIGHT CAUSE POLYNOMIAL UNDERFLOW
057                   JRST .+2
058  001 019          POPJ P, ;AND IS SMALL ENOUGH FOR SIN X _ X
059  007 087          FDVR A,SC1      ;PI/2
060  001 006          PUSH P,A
061  001 007          PUSH P,B
062  001 006          MULI A,400
063  001 006          TSC A,A ;CAML A,...SETZB B,-1(P)
064  001 006          ASH B,-243(A)
065  001 007          MOVNS A,B
066  001 006          ANDCMI A,1
067  001 006          TLC A,232000
068  001 006          FAD A,A
069  001 019          FADRB A,-1(P)
070  001 007          TRNE B,2
071  001 019          MOVNS A,-1(P)
072  001 006          FMP A,A
073  007 091          MOVE B,SC9
074  001 006          FMP B,A
075  007 090          FAD B,SC7
076  001 006          FMP B,A
077  007 089          FAD B,SC5
078  001 006          FMP B,A
079  007 088          FAD B,SC3
080  001 007          FMP A,B
081  007 087          FADR A,SC1
082  001 019          FMPRM A,-1(P)
083  001 007          POP P,B
084  001 006          POP P,A
```

```
085  001 019          POPJ P,
086
087           SC1:    1.5707963267
088           SC3:    -0.64596371106
089           SC5:    0.07968967928
090           SC7:    -0.00467376557
091           SC9:    0.00015148419
092                   CONSTANTS
093                   VARIABLES
094
```

```
001
002                     subttl data bases and constants
003
004             ;data base for players
005             plrbri: 20112   ;poarameter mode hafiwds to set brightness and scale
006             pwbbri: 20114   ;and go into point mode
007
008             pinitd: 150.0   ;initial dist. from center of board (x and y derived from this)
009
010             pinitx: 300.0   ;initial x value for player positions at start of game
011                     500.0
012                     700.0
013                     500.0
014             pinity: 500.0   ;initial y
015                     700.0
016                     500.0
017                     300.0
018     002 116 pinita: cpie.5  ;initial angle
019                     0.0
020     002 117         cpi1.5
021     002 115         cpie
022
023     001 021 input:  block nplyrs    ;holds inputs for each player during pass
024             speed:  3.0             ;points/step to move if moving
025             angrat: .03             ;radians/step to turn if turning
026             angspd: 4.0     ;factor to scale pot*angrat by when using pots
027             fuzzm:  .3      ;factor for which pots are considered close enough to be zero
028             fuzzr:  .3      ;same thing except for rotation instead of motion
029             dconst: 1.0     ;value digital inputs are assumed to be
030             splen:  40.0    ;standard player length on a side
031             paslim: 10      ;number of 60th's that must go by between passes
032             passtm: 0       ;timer for above
033
034
035     001 021 plocx:  block nplyrs    ; x coord of player
036                     block 4 ;boundary line x coords (kludge)
037     001 021 plocy:  block nplyrs    ; y coord
038                     block 4
039     001 021 plang:  block nplyrs    ;heading angle in radians
040                     block 4
041     001 021 plsin:  block nplyrs    ;sin(plang)
042                     block 4
043     001 021 plcos:  block nplyrs    ;cos(plang)
044                     block 4
045     001 021 plslop: block nplyrs    ; tan(plang)   i.e. slope
046                     block 4
047     001 021 plen:   block nplyrs    ;player length
048                     block 4
049     001 021 plensq: block nplyrs    ;player length squared
050                     block 4
051     001 021 plscor: block nplyrs
052
053             tmmate: 1       ;index of other teammate(s) (circular list if more than 2)
054                     0
055                     3       ;teams are (0,1) and (2,3)
056                     2
057
058             tmon:   0       ;team # player is on
059                     0
060                     1
061                     1
062
063             control:        ; says whether to get input from A/D pots or bits
064     008 023 play1:  070 ? 070 ? 070 ? .bp 20,input+1?.bp 100,input+1
065     008 023 play2:  070 ? 070 ? 070 ? .bp 20,input+1?.bp 100,input+1
066     008 023 play3:  070 ? 070 ? 070 ? .bp 20,input+3?.bp 100,input+3
067     008 023 play4:  070 ? 070 ? 070 ? .bp 20,input+3?.bp 100,input+3
068
069             rotright==0     ;symbols defined to make patching control table easy
070             rotleft==1
071             Xright==2
072             Xleft==3
073             Yup==4
074             Ydown==5
075             fire==6
076             pass==7
077
078             ltmscr: 0       ;last team that had beam when goal hit.
079             lglhit: 0       ;last goal hit (team # of)
080             tmscor: 0       ;score for team 0 (0 and 1)
081                     0       ;team 1 (2 and 3)
082
083             ztime:  30.     ;# of seconds team can possess beam
084             ztmlft: 0       ;count of seconds left til beam possession lost
```

```
085             ztim60: 0       ;count in 60ths of sec
086             ;boundary lines
087
088             blowx:  0       ;left
089             bhighx: 0       ;right
090             blowy:  0       ;bottom
091             bhighy: 0       ;top
092
093             pbdgap: 2.0     ;boundary-player gap
094             plowx:  0       ;limits of trvael for players.
095             plowy:  0
096             phighx: 0
097             phighy: 0
098
099             bglfig: 0       ;nonzero if line has a goal on it
100                     1
101                     0
102                     1
103
104             bglow:  block 4 ;lower bound of coordinate for goal on line (if any)
105             bghigh: block 4 ;upper bound  "
106             bgwide: 100.0   ;width of goal
107
108             bdlenx: 700.0   ;width of board (x coord)
109             bdleny: 900.0   ;height of board (y coord)
110             bln.5x: 0       ;half bdlenx
111             bln.5y: 0       ;half bdleny
112             bdcenx: 0       ;coord of center axis (x)
113             bdceny: 0       ;coord of center axis (y)
114
115             bprot:  160.0   ;radius of protective shield around goal
116             bprosq: 0       ;set to square of above
117
118             frztim: 3       ;# seconds to freeze action on beam hit.
119
120             ;---------------
121             ;beam hit variables used while computing beam path
122             zstrtx: 0       ;start of beam, x coord
123             zstrty: 0       ;start of beam, y coord
124             zangle: 0       ;direction of beam in radians
125             zsin:   0       ;sin(zangle)
126             zcos:   0       ;cos(zangle)
127             zslope: 0       ;slope = tan(zangle)
128             zslopx: 0       ;non-zero if slope infinite
129             zfrom:  0       ;index of player beam from (via fire or bounce)
130
131             zbestd: 0       ;holds distance to best hit seen so far (closest)
132             zbestp: 0       ;holds index of player implicated by zbestd
133             zbestx: 0       ;x coord of best hit
134             zbesty: 0       ;y coord of best hit
135             zbounc: 0       ; # times beam has bounced off something so far
136             zmaxbc==3       ;max # bounces before beam ownership transfers
137  008 136    zbnclt: block zmaxbc    ;stores record of players bounced off
138             zbri:   20113   ;beam brightness/scale, go into point mode
```

```
001
002                     subttl board setup
003
004             define dline x1,y1,x2,y2
005     003 070         move a,x1
006     003 071         move b,y1
007     003 073         move c,x2
008     003 074         move d,y2
009     015 055         pushj p,drwlin
010             termin
011
012     008 115 bsetup: move a,bprot    ;get protective-circle radius
013     001 006         fmpr a,a        ;square it
014     008 116         movem a,bprosq  ;store for use by player updat
015     008 030         move a,splen    ;get player length
016     001 006         fadr a,[2.0]    ;add 2 units for safety
017     008 088         movem a,blowx   ;and use that as lower limit of board
018     008 090         movem a,blowy   ;in both directions.
019     008 093         fadr a,pbdgap
020     008 094         movem a,plowx
021     008 095         movem a,plowy
022             IRP xory,,[x,y]
023     001 007         move b,[1024.0] ;get absolute limit of coord
024     001 008         move c,bdlen!xory       ;and desired width
025     001 006         fadr c,a        ;adjust out
026     001 008         fsbr b,c        ;and check
027     001 007         camle a,b       ;skip if enough room
028     004 029         jsr death       ;foo.
029     001 008         movem c,bhigh!xory
030     008 093         fsbr c,pbdgap   ;spacing
031     001 008         movem c,phigh!xory
032     001 007         move b,bdlen!xory
033     001 007         fmpr b,[0.5]    ;find 1/2 length
034     001 007         movem b,bln.5!xory
035     001 007         fadr b,blow!xory
036     001 007         movem b,bdcen!xory      ;center of board line
037             TERMIN
038
039     001 011         movei t,3       ;loop 4 times storing line data
040     001 011 bsetp1: cain t,0
041     008 109         jrst [move a,blowx ? move b,bdceny ? move c,pi.5 ? move d,bdleny ? jrst bset2]
042     001 011         cain t,1
043     008 108         jrst [move a,bdcenx ? move b,bhighy ? setz c, ? move d,bdlenx ? jrst bset2]
044     001 011         cain t,2
045     008 109         jrst [move a,bhighx ? move b,bdceny ? move c,pi.5 ? move d,bdleny ? jrst bset2]
046     001 011         cain t,3
047     008 108         jrst [move a,bdcenx ? move b,blowy ? setz c, ? move d,bdlenx ? jrst bset2]
048     008 035 bset2:  movem a,plocx+nplyrs(t)
049     008 037         movem b,plocy+nplyrs(t)
050     008 039         movem c,plang+nplyrs(t)
051     008 047         movem d,plen+nplyrs(t)
052     001 009         fmpr d,d
053     008 049         movem d,plensq+nplyrs(t)
054     001 008         move a,c        ;setup for trig stuff
055     007 054         pushj p,cos
056     008 043         movem a,plcos+nplyrs(t)
057     001 008         move a,c
058     007 055         pushj p,sin
059     008 041         movem a,plsin+nplyrs(t)
060     008 043         PFDVR a,plcos+nplyrs(t) ;find tan
061     008 045         movem a,plslop+nplyrs(t)
062     009 040         sojge t,bsetp1
063
064                     ;initialize player positions
065     001 008         movei c,3       ;loop thru players
066     008 112 bset3:  move a,bdcenx
067     008 113         move b,bdceny   ;coords of center of board
068     001 008         cain c,0
069     008 008         fsbr a,pinitd
070     001 008         cain c,1
071     008 008         fadr b,pinitd
072     001 008         cain c,2
073     008 008         fadr a,pinitd
074     001 008         cain c,3
075     008 008         fsbr b,pinitd
076     008 010         movem a,pinitx(c)
077     008 014         movem b,pinity(c)
078     009 066         sojge c,bset3
079
080                     ;find where to start number display
081     008 089         move a,bhighx
082     001 006         fadr a,[80.0]
083     013 073         movem a,dtx
084     013 073         movem a,dtx+1
```

```
085  013 073          movem a,dtx+2
086  008 113          move b,bdceny
087  013 076          movem b,dty+1    ;timeout cntr
088  001 007          fadr b,[200.0]
089  013 076          movem b,dty      ;score, team 0
090  001 007          fsbr b,[400.0]
091  013 076          movem b,dty+2    ;score, team 1
092
093  008 106          move a,bgwide    ;find desired goal width
094  001 006          fmpr a,[0.5]     ;divide
095  008 112          move b,bdcenx    ;center of board down middle
096  001 006          fsbr b,a
097  008 104          movem b,bglow+1
098  008 104          movem b,bglow+3
099  008 112          move b,bdcenx
100  001 006          fadr b,a
101  008 105          movem b,bghigh+1
102  008 105          movem b,bghigh+3
103
104  018 054          push p,vdlipt    ;save deposit ptr
105  018 019          movei a,dlstrt   ;start board here
106  018 054          movem a,vdlipt
107  010 007          move a,bdbri     ;get board brightness, scale
108  018 054          movem a,@vdlipt
109  018 054          aos vdlipt
110
111  008 091          dline blowx,blowy,blowx,bhighy
112  008 091          dline blowx,bhighy,bhighx,bhighy
113  008 090          dline bhighx,bhighy,bhighx,blowy
114  008 090          dline bhighx,blowy,blowx,blowy
115
116  008 090          move a,blowy
117  010 004          fsbr a,bgldep    ;goal depth
118  010 005          movem a,glowy
119  008 091          move a,bhighy
120  010 004          fadr a,bgldep
121  010 006          movem a,ghighy
122
123  010 006          dline bglow+1,bhighy,bglow+1,ghighy
124  010 006          dline bglow+1,ghighy,bghigh+1,ghighy
125  008 091          dline bghigh+1,ghighy,bghigh+1,bhighy
126
127  010 005          dline bglow+3,blowy,bglow+3,glowy
128  010 005          dline bglow+3,glowy,bghigh+3,glowy
129  008 090          dline bghigh+3,glowy,bghigh+3,blowy
130
131                   ;display protective semi-cricles around goals
132  008 112          move a,bdcenx    ;right in the center X
133  008 090          move b,blowy     ;and bottom Y line
134  001 008          move c,[0,0]     ;from 0.0 radians
135  001 009          move d,[3.14159265358] ;to Pi radians
136  008 115          move e,bprot     ;radius of protective circle
137  016 008          pushj p,dcirc    ;draw a circle there
138
139  008 112          move a,bdcenx    ;same but on top now
140  008 091          move b,bhighy
141  001 008          move c,[3.14159265358]
142  001 009          move d,[6.2831853]      ;or 0.0 ?
143  008 115          move e,bprot
144  016 008          pushj p,dcirc    ;and drw the top circle
145
146  018 054          move a,vdlipt
147  018 019          subi a,dlstrt
148  010 008          movem a,bdsiz
149  018 054          pop p,vdlipt
150  001 019          popj p,
```

```
001
002                     subttl goal scoring, beam assignment, random generator
003
004             bgldep: 25.0    ;goal depth
005             glowy:  0       ;bottom y coord of bottom goal
006             ghighy: 0       ;top y coord of top goal
007             bdbri:  20111   ;board brightness
008             bdsiz:  0       ;# words needed in board display list
009
010                     ; score a goal
011  008 134    goalht: move a,zbesty   ;find which goal hit
012  001 007            setz b,         ;assume #0 (top)
013  008 113            camge a,bdceny  ;above or below center?
014  001 007            movei b,1       ;below, bottom goal (team #1)
015  008 079            movem b,lglhit  ;save # of goal hit
016  001 007            movei b,1       ;now set up default score
017  008 113            camge a,bdceny  ;test again
018  008 058            jrst [  skipe c,tmon(pwb)       ;team hitting bottom goal should be #0
019  001 007                    hrrei b,-2      ;oops! shot self!
020  010 023                    jrst golht0]
021  008 058            skipn c,tmon(pwb)       ;team hitting top goal should be #1
022  001 007            hrrei b,-2      ;oops again
023  008 080    golht0: addm b,tmscor(c)        ;add to score
024  008 078            movem c,ltmscr  ;save # as last team to score (whether hit theirs or own)
025  004 086            pushj p,vbfend  ;wrap up current variable buffer and switch it
026  008 118            move b,frztim   ;secs to freeze
027  001 007            imuli b,60.     ;60ths
028  004 024            idiv b,steptm   ;find # of go-aheads to wait
029  004 026    golht1: aose clksnk
030             IFE $ITS,[
031                     jrst .-1
032             ]
033             IFN $ITS,[
034                     .HANG
035  001 006            MOVE A,[-2,,[.SPIRQC,,[0] ? .SIMASK,,[200000,,0]]]
036  001 006            .SUSET A
037             ]
038  010 029            sojg b,golht1
039
040  001 019            popj p,
041
042                     ;find which player gets beam at start of game
043  008 079    zassgn: move c,lglhit   ;get team # of last goal hit
044  010 051    zass1:  pushj p,prandm
045  001 021            idivi a,nplyrs
046  008 058            came c,tmon(b)  ;see if selected player on that team
047  010 044            jrst zass1      ;no, get another
048  001 007            move pwb,b      ;remainder is from 0-nplyrs
049  001 019            popj p,
050
051  010 054    prandm: pushj p,random
052  001 006            movm a,a        ;return positive random number
053  001 019            popj p,
054  008 085    random: skipn a,ztim60  ;hash it up with random time
055  010 059            move a,ran
056  010 059            fmpb a,ran
057  001 006            tsc a,a
058  001 019            popj p,
059             ran:    123456,,654321  ;initial seed
```

```
001
002                     subttl input
003
004             ;---- input bits--
005             i%fire==bit1.1  ;firing beam
006             i%pass==bit1.2  ;passing to teammate
007             i%clkw==bit1.3  ;rotate clockwise (right)
008             i%ccw== bit1.4  ;rotate counter-clockwise(left)
009             i%posy==  bit1.5        ;move up
010             i%negy==bit1.6  ;move down
011             i%posx==bit1.7  ;move right
012             i%negx==bit1.8  ;move left
013
014   001 006   inpget: push p,a
015   001 007           push p,b
016   011 095           skipn inmode
017   001 006           datai a
018   011 095           skipe inmode
019   001 006           datai 420,a
020   001 006           setca a,a               ;bits are 1's until 0 imposed.
021   001 014           movsi plr,-4
022   001 006   inpgt1: lshc a,-9.
023   001 007           lsh b,-27.
024   008 023           movem b,input(plr)
025   011 022           aobjn plr,inpgt1
026
027   011 096           skipe switches
028   011 033            jrst getpots
029   011 090           move a,[inval-1,,inval] ;setup overlapping blt
030   011 090           blt a,inval-1+16.       ;if no pots are to be read, zero inval
031   011 075           jrst timout
032
033             getpots:                ; read pot values
034             ife $its,[
035   001 006   wait:    move a,20.      ; dont wait much more than 40 microseconds
036   011 075           sojl a,timout
037                     conso 574,1_43  ; see if processor can access A/D
038                      jrst .-2
039
040   001 006           movsi a,-16.            ; index
041                     cono 574,200    ; rate of 1, non-sequential, non-pack, no intrpt
042   011 079   loop5:  skipn b,chans(a)        ; get channel # form channel table
043   011 053            jrst next
044   001 007           datao 574,b     ; select channel
045
046   001 007           move b,20.
047   011 053           sojl b,next     ; if much more than 40 microseconds, just use last val
048                     conso 574,10    ; is data here yet?
049                      jrst .-2
050
051   011 089           datai 574,inputs(a)     ; input pot value and place in list
052
053   011 042   next:    aobjn a,loop5
054             ]
055             ifn $its,[
056                     ;IMX device should already have been opened in block mode
057   011 089           move a,[chans,,inputs]  ; copy channel numbers into inputs variables
058   011 089           blt a,inputs+16.-1
059   011 089           move a,[-16.,,inputs]   ; get values
060   002 082           .iot imxchn,a
061             ]
062   001 006           movsi a,-16.
063   011 089   loop6:  move b,inputs(a) ; get input value
064   001 007           fsc b,233       ; float the number
065   001 007   ife $its, fadr b,[0.0]  ; on 6, must normalize before dividing!!
066   001 007           fdvr b,[1000.0] ; scale to number from 0.0 to 2.0 (almost)
067   001 007           fsbr b,[1.0]    ; change to -1.0 to 1.0
068   011 079           skipn chans(a)  ; this is to make sure if we didn't input for that
069   001 007            setz b,        ;  channel the value is still zero
070   011 084           skipe negpot(a)
071   001 007            movn b,b
072   011 090           movem b,inval(a) ; stick into input value
073   011 063           aobjn a,loop6
074
075   001 007   timout: pop p,b
076   001 006           pop p,a
077   001 019           popj p,
078
079             chans:  73 ? 70 ? 71 ? 0        ;players 1 and 2 each side of New Pot Box
080                     74 ? 77 ? 76 ? 0
081                     10 ? 2  ?  3 ? 0        ;player 2 and 3 , AMF arm console pots
082                     30 ? 5  ?  6 ? 0
083
084             negpot: 0 ? 0 ? 0 ? 0           ;if non-0, means negate sense of pot.
```

```
085                     0 ? 0 ? 0 ? 0
086                     1 ? 1 ? 1 ? 0
087                     1 ? 1 ? 1 ? 0
088
089             inputs: block 4*4
090             inval:  block 4*4       ; place to put input values, should be inited to 0
091             rotate==0
092             xmove==1
093             ymove==2
094             fire==3
095             inmode: 1       ;0= data switches, 1=extra switch bank
096             switches: 1     ;1= use A/D or switches depending on control table
097                             ;0= use switches in the old fashion
```

```
001
002                     subttl player updating
003             updat:          ; update inval variables according to directions
004   008 029           move c,dconst   ; digital inputs have the value dconst
005   001 014           movei e,(plr)
006   001 010           ash e,2         ; 4 * player
007   011 096           skipe switches
008   012 020            jrst updat1
009   008 023           move b,input(plr)
010   011 090           trne b,i%posx ? movem c,inval+xmove(e)
011   011 090           trne b,i%negx ? movnm c,inval+xmove(e)
012   011 090           trne b,i%posy ? movem c,inval+ymove(e)
013   011 090           trne b,i%negy ? movnm c,inval+ymove(e)
014   011 090           trne b,i%clkw ? movem c,inval+rotate(e)
015   011 090           trne b,i%ccw ? movnm c,inval+rotate(e)
016   011 090           trne b,i%pass ? movem c,inval+fire(e)
017   011 090           trne b,i%fire ? movnm c,inval+fire(e)
018   012 038           jrst updat4
019
020   001 014   updat1: movei a,(plr)
021   001 006           ash a,3         ; get # of player * 8 in a
022   001 014           movei e,(plr)
023   001 010           ash e,2
024   001 010           hrli e,-4       ; -4,,player # * 4
025   008 063   loop7:  skipn b,control(a) ; does control table say to use pots?
026   012 035            jrst nextsw    ; use the pot input for this variable
027   011 090           setzm inval(e)  ; zap any pot reading there may already be
028   001 007           ldb b,b         ; pick up bit using byte pointer in control tab
029   001 007           trne b,1        ; if its on, set corresponding inval to (dconst)
030   011 090            movem c,inval(e)
031   008 063           ldb b,control+1(a) ; pick up bit representing opposite condition
032                                     ;e.g. left rotate instead of right rotate..
033   001 007           trne b,1        ;if its on, set inval to -(dconst)
034   011 090            movnm c,inval(e)
035   001 006   nextsw: addi a,2        ; advance a to next control tabl entry
036   012 025           aobjn e,loop7   ; advance e to next inval
037
038   001 014   updat4: movei e,(plr)   ; e gets plr*4
039   001 010           ash e,2
040
041                     ;update X motion of player
042   001 006           setz a,
043   011 090           move a,inval+xmove(e)
044   001 006           movm t,a        ; takes abs val
045   008 027           camg t,fuzzm    ; is the value small enough to be zero?
046   001 006            setz a,        ; then make it exactly zero
047   008 024           fmpr a,speed    ; multiply by speed const
048                     jfcl            ; for patching more interesting calculations
049   008 035           fadr a,plocx(plr) ; add the x position to new movment
050   008 096           camle a,phighx  ; truncate new position if its outside board
051   008 096            move a,phighx
052   008 094           camge a,plowx
053   008 094            move a,plowx
054
055   001 007           setz b,         ; do same things for Y movement
056   011 090           move b,inval+ymove(e)
057   001 007           movm t,b
058   008 027           camg t,fuzzm
059   001 007            setz b,
060   008 024           fmpr b,speed
061                     jfcl
062   008 037           fadr b,plocy(plr)
063   008 097           camle b,phighy
064   008 097            move b,phighy
065   008 095           camge b,plowy
066   008 095            move b,plowy
067
068                     ; check for moving within edge of protective circle:
069   008 112           move c,bdcenx
070   008 091           move d,bhighy   ;coords of center of top goal
071   008 113           camge b,bdceny  ;use above if player is in upper half
072   008 090           move d,blowy    ;else check lower goal
073   007 004           pushj p,distsq  ;get dist
074   008 116           caml c,bprosq   ;compare w/square of protective circle radius
075   012 095           jrst updat2     ;okay...
076   011 090           skipe inval+xmove(e) ;within circle! moving in x dir?
077   008 112            jrst [camn a,bdcenx    ;test for critical point
078   012 087                   jrst updat1
079   008 112                   camg a,bdcenx   ;see which side of bd he's on
080   011 090                   jrst [skipl inval+xmove(e)      ;ok to move in neg. dir
081   008 035                         move a,plocx(plr)         ;but not OK in pos dir(non-neg)
082   012 087                         jrst updat1]
083   011 090                   skipg inval+xmove(e) ;ok to move in pos dir
084   008 035                   move a,plocx(plr)
```

```
085  012 087                  jrst updat1]
086
087  011 090  updat1: skipe inval+ymove(e)    ;moving in y dir?
088  008 113           jrst [camg b,bdceny    ;see which side
089  011 090                 jrst [skipg inval+ymove(e)      ;ok to move up
090  008 037                       move b,plocy(plr)
091  012 095                       jrst updat2]
092  011 090                 skipl inval+ymove(e)
093  008 037                 move b,plocy(plr)
094  012 095                 jrst updat2]
095  008 035  updat2: movem a,plocx(plr)      ;store normalized coords
096  008 037          movem b,plocy(plr)
097
098  001 014          movei c,(plr)           ;get 8 * plr into c
099  001 008          ash c,3
100
101  011 090          skipn a,inval+rotate(e) ; is rotation non-zero?
102  012 132           jrst updat3
103  011 090          movm b,inval+rotate(e)
104            ;      movm t,b         ;if absval of rotate is too small
105  008 028          camg b,fuzzr
106  012 132           jrst updat3
107  008 025          fmpr b,angrat
108  008 026          fmpr b,angspd    ;scale-up for pots
109                   jfcl
110  011 096          skipe switches  ;if rotate was digital, us angrat
111  008 063           skipe control(c)
112  008 025            move b,angrat
113  001 006          caile a,
114  001 007           movn b,b                ;CCW is - in inval, CW is +.
115  008 039          fadrm b,plang(plr)
116  008 039          move a,plang(plr)        ;normalize
117  002 119          camle a,pi$2     ;make angle between 0 and pi*2
118  002 119           fsbr a,pi$2
119  001 006          caige a,
120  002 119           fadr a,pi$2
121  008 039          movem a,plang(plr)
122
123            ; if analog use value as direct position
124            ;      skipn b,inval+rotate(e) ;now check for rotation.
125            ;       jrst updat3     ;skip if not
126            ;      fadr b,[1.0]     ;make range 0.0 to 2.0
127            ;      fmpr b,[1.2]     ;don't want having to turn pot all the way
128            ;      fmpr b,pie       ;now angle is between 0 and 2*pi (well, not really)
129            ;      movem b,plang(plr)
130
131                   ;derive necessary trigonometric values
132  008 039  updat3: move a,plang(plr)
133  007 055          pushj p,sin
134  008 041          movem a,plsin(plr)
135  008 039          move a,plang(plr)
136  007 054          pushj p,cos
137  008 043          movem a,plcos(plr)
138  008 041          move a,plsin(plr)
139  008 043          PFDVR a,plcos(plr)
140  008 045          movem a,plslop(plr)
141
142  001 019          popj p,
```

```
001
002                     subttl display updating
003
004   008 005   plrdis: move a,plrbri   ;get standard player bri.
005   001 014           hrrz b,plr
006   001 013           camn b,pwb      ;this one is pwb?
007   008 006           move a,pwbbri   ;if so, use increased pwb brightness
008   018 004           move t,dlplst(plr)      ;get addr to display list
009   001 011           movem a,(t)     ;store brightness
010
011   008 035           move a,plocx(plr)
012   008 037           move b,plocy(plr)
013   014 045           pushj p,getpt   ;get display wd for pt specified
014   001 008           andcmi c,bit2.7+bit2.6+bit2.5   ;flush mode bits
015   001 008           iori c,bit2.7   ;go into vector mode after this hwd
016   001 011           movem c,1(t)    ;store
017
018   008 041           move b,plsin(plr)
019   008 043           move a,plcos(plr)
020   008 047           fmpr a,plen(plr)
021   008 047           fmpr b,plen(plr)
022   014 066           pushj p,getvec  ;get vector for this displacement
023   001 008           hrl c,c         ;copy in LH
024   001 008           ior c,[bit2.8,,0]       ;make visible vector for first one
025   001 008           xori c,bit2.7+bit1.8    ;reverse sign for invisible vector(to return to strt)
026   001 011           movem c,2(t)    ;store
027
028   001 006           movn a,a
029   001 007           movn b,b
030   014 066           pushj p,getvec
031   001 008           hrl c,c
032   001 008           ior c,[bit2.8,,bit2.9]  ;escape into parameter mode after this one
033   001 008           xori c,bit2.7+bit1.8
034   001 011           movem c,3(t)    ;store second side of player
035   001 006           movei a,20000   ;go into point mode
036   001 011           movem a,4(t)
037
038   008 035           move a,plocx(plr)
039   008 037           move b,plocy(plr)
040   008 041           move c,plsin(plr)
041   008 043           move d,plcos(plr)
042   018 002           pushj p,@dpradr(plr)    ;do ID stuff (points)
043
044   001 019           popj p,
045
046             define pushae pdac,aclist
047             IRP ac,,[aclist]
048                     push pdac,ac
049             TERMIN
050             TERMIN
051             define popae pdac,aclist
052             IRP ac,,[aclist]
053                     pop pdac,ac
054             termin
055             termin
056
057             ;routine to display random numbers and text as indicated in tables
058
059   001 010   disran: pushae p,[a,b,c,d,e]
060   013 071           movsi e,-dtmax  ;get aobjn thru random-display tables
061   013 082   disrn1: skipe d,dtloc(e)        ;skip til something found(get addr of it)
062   013 079           jrst [move c,dtscbr(e)  ;get scale/brightness
063   013 076                 move b,dty(e)     ;y coord to start at
064   013 073                 move a,dtx(e)     ;x coord
065   014 005                 pushj p,dtdisp    ;display it
066                           jrst .+1]
067   013 061           aobjn e,disrn1
068   001 006           popae p,[e,d,c,b,a]
069   001 019           popj p,
070
071             dtmax==3        ;current max slots in random-display tables
072
073             dtx:    900.0   ;x coord of start of text
074                     900.0
075                     900.0
076             dty:    950.0   ;y coord
077                     700.00
078                     450.0
079             dtscbr: 20173   ; scale/brightness parameter wd
080                     20175
081                     20173
082   008 080   dtloc:  tmscor  ;location of integer number (or setz [asciz/string/] )
083   008 084           ztmift
084   008 080           tmscor+1
```

```
001
002                ;displays text; takes starting x and y in a,b  scale/bri param wd in c, and
003                ;addr of number (or addr of text string, with sign bit set as in setz)
004
005   018 054  dtdisp: movem c,@vdlipt ;store paramter wd. (later, an idpb c,vdlipt)
006   018 054          aos vdlipt
007   014 045          pushj p,getpt    ;convert a,b into point-mode word in c
008   001 008          andcmi c,bit2.5+bit2.2  ;take out point-mode bit and don't intensify
009   001 008          iori c,bit2.6+bit2.5    ;go into text mode after this one
010   018 054          movem c,@vdlipt ;store
011   018 054          aos vdlipt
012   001 006          movei a,440600  ;get LH to turn VDLIPT
013   018 054          hrlm a,vdlipt   ;into a 6-bit byte pointer.
014   001 006          movei a,35      ;go into upper case
015   018 054          idpb a,vdlipt
016                ;     jumpl d,dtex    ;jump if doing ASCIZ text
017
018                    ;doing a number. get it
019   001 009          move a,@d       ;more general than (d)
020   014 032          pushj p,dnum    ;do the chars
021   014 023          jrst dtdsp9     ;done
022
023   001 006  dtdsp9: movei a,37      ;esc. to parameter mode
024   018 054          idpb a,vdlipt
025   001 006          setz a,         ;now do kludge to ensure core zeroed til next param wd.
026   001 007          movei b,6
027   018 054          idpb a,vdlipt
028   001 007          sojg b,.-1      ;repeat for a word's length
029   018 054          hrrzs vdlipt    ;now clear LH of vdlipt ptr, and all's well.
030   001 019          popj p,
031
032   018 054  dnum:   jumpl a,[push p,b ? movei b,55 ? idpb b,vdlipt ? pop p,b        ;minus sign
033   001 006                movn a,a        ;make positive
034                          jrst .+1]
035   001 006  dnum1:  idivi a,10.     ;base 10.
036   014 040          jumpe a,dnum2   ;ah, finished
037   001 007          push p,b        ;save remainder for later typeout
038   014 035          pushj p,dnum1   ;deposit number in a
039   001 007          pop p,b         ;restore remainder-digit
040   001 007  dnum2:  movei b,"0(b)   ;convert to ascii
041   018 054          idpb b,vdlipt   ;mapping for #'s is equiv. to ascii
042   001 019          popj p,
043
044
045   001 006  getpt:  push p,a
046   001 007          push p,b
047   003 158          frifix a
048   003 158          frifix b
049   001 006          hlrz c,a
050   001 007          hll c,b
051   001 008          and c,[1777,,1777]
052   001 008          ior c,[220000,,022000]
053   001 007          pop p,b
054   001 006          pop p,a
055   001 019          popj p,
056
057                    ;takes floating x,y in a,b as vector increment from present position
058   001 008  drwvec: push p,c
059   014 066          pushj p,getvec
060   001 008          iori c,bit2.8   ;make visible vector
061   018 054          movem c,@vdlipt ;deposit
062   018 054          aos vdlipt
063   001 008          pop p,c
064   001 019          popj p,
065
066   001 006  getvec: push p,a
067   001 007          push p,b
068   003 158          frifix a
069   003 158          frifix b
070   001 006          movm c,a
071   001 008          hlrz c,c
072   001 008          andi c,177
073   001 006          skipge a
074   001 008          tro c,200       ;sign bit for x
075   001 007          movm a,b
076   001 006          hlrz a,a
077   001 006          andi a,177
078   001 007          skipge b
079   001 006          tro a,200       ;sign bit for y
080   001 008          dpb a,[101000,,c]       ;deposit in c
081   001 007          pop p,b
082   001 006          pop p,a
083   001 019          popj p,
```

```
001
002  001 006  drawl:  push p,a
003  001 007          push p,b
004  001 008          push p,c
005  001 009          push p,d
006  001 010          push p,e
007  001 008          push p,c
008  014 045          pushj p,getpt    ;get display wd for pt specified
009  001 008          andcmi c,bit2.7+bit2.6+bit2.5    ;flush mode bits
010  001 008          iori c,bit2.7    ;go into vector mode after this hwd
011  001 011          movem c,(t)
012  001 011          aoj t,           ;starting point is now set
013  001 008          pop p,c
014  001 006          fsbr c,a         ;find relative coords
015  001 007          fsbr d,b
016  001 008          move b,c         ;don't mung c yet
017  003 020          PFDVR b,d        ;find slope
018  001 007          movm a,b
019  001 006          caml a,[2048.0] ;no sense in more precision for vertical
020  001 007          jrst [skipge b
021  001 007                  skipa b,[-2048.0]
022  001 007                  move b,[2048.0]
023                           jrst .+1]
024  003 158          frifix b         ;y rise per
025  001 006          move a,[1,,0]    ;single x step
026  001 009          skipge d
027  001 006          movn a,a         ;in whatever direction
028  001 006          push p,a
029  001 007          push p,b
030  001 006          movm a,a
031  001 007          movm b,b
032  001 007          camge a,b        ;if x changest fastest, do normal stuff
033            %dxch==bit2.9
034  015 033          troa f,%dxch     ;if y changes fastest, set flag
035  015 033          trz f,%dxch
036  001 007          pop p,b
037  001 006          pop p,a
038  015 033          trne f,%dxch
039  001 009          jrst [  exch a,b ? exch c,d ? jrst .+1]
040  001 006          movm e,a
041  001 009          move d,[177,,0]
042  001 010          idivi d,e        ;find pct of total vecto length
043  001 009          imul b,d         ;adjust smaller step accordingly
044  001 006          skipl a
045  001 006          skipa a,[177,,0]
046  001 006          movn a,[177,,0]
047  003 158          frifix c         ;find limit of controlling coord
048  001 008          movm d,c         ;magn.
049  001 009          idiv d,[177,,0] ;find # loops in it
050
051            drwin5:
052
053
054
055  001 006  drwlin: push p,a
056  001 007          push p,b
057  001 008          push p,c
058  001 009          push p,d
059  001 010          push p,e
060  001 012          push p,tt
061  001 008          push p,c
062  008 138          move c,zbri      ;beam brightness/scale
063  018 054          movem c,@vdlipt ;store
064  018 054          aos vdlipt
065  014 045          pushj p,getpt    ;set location
066  001 008          andcmi c,bit2.7+bit2.6+bit2.5    ;flush mode
067  001 008          iori c,bit2.7    ;go into vector mode after this
068  018 054          movem c,@vdlipt
069  018 054          aos vdlipt
070  001 008          pop p,c
071  001 006          fsbr c,a         ;del_x
072  001 007          fsbr d,b         ;del_y
073  001 008          movm e,c         ;magn of del_x
074  001 009          movm tt,d        ;magn of del_y
075  015 116          camg e,c127.
076  015 116          camle tt,c127.
077                   skipa
078  015 103          jrst [move a,c ? move b,d ? jrst nofig] ;then dont have to figure
079  002 074          trz f,%dexch
080  001 012          camge e,tt       ;skip if del_y < del_x
081  002 074          jrst [tro f,%dexch ? exch c,d ? exch e,tt ? jrst .+1]   ;if del-y > del-x
082
083  003 020          pfdvr d,c        ;find m = <lesser del>/<larger del>
084  015 116  xcont:  move a,c127.     ;amount of x to be moved
```

```
085  001 008          skipg c          ;if del_x < 0
086  001 006          movn a,a         ;move -127.
087  001 006          move b,a
088  001 009          fmpr b,d         ;get y (m*x increment)
089  002 074          trne f,%dexch
090  001 007          exch a,b         ;exchange coords if y is really in a
091  014 058          pushj p,drwvec   ;draw this
092  015 116          fsbr e,c127.     ;|del_x| = 127 - new del_x
093  015 116          camle e,c127.    ;done yet?
094  015 084          jrst xcont
095
096  001 010          move a,e
097  001 008          skipg c
098  001 006          movn a,a
099  001 006          move b,a
100  001 009          fmpr b,d
101  002 074          trne f,%dexch
102  001 007          exch a,b
103  014 066  nofig:  pushj p,getvec
104  001 008          ior c,[bit2.9+bit2.8]   ;escape to param mode
105  018 054          movem c,@vdlipt
106  018 054          aos vdlipt
107
108  001 012          pop p,tt
109  001 010          pop p,e
110  001 009          pop p,d
111  001 008          pop p,c
112  001 007          pop p,b
113  001 006          pop p,a
114  001 019          popj p,          ;return
115
116            c127.:        127.0
117            yinc:         0.0
118            xinc:         0.0
```

```
001
002                     subttl circle drawer
003             %ixpos==bit2.9
004             %ixneg==bit2.8
005             %iypos==bit2.7
006             %iyneg==bit2.6
007
008  001 011  dcirc:   push p,t
009  001 012           push p,tt
010  001 006           move t,a
011  001 007           move tt,b
012  001 006           move a,[0.5]
013  001 010           fdvr a,e
014  001 008           fsbr d,c
015  001 006           fdvr d,a
016  003 150           ifix d
017  001 006           move b,a
018  007 055           pushj p,sin
019  016 075           movem a,csini
020  001 007           move a,b
021  007 054           pushj p,cos
022  016 074           movem a,ccosi
023
024  001 008           move a,c
025  007 055           pushj p,sin
026  001 006           move b,a
027  001 008           move a,c
028  007 054           pushj p,cos
029  001 011           exch a,t
030  001 012           exch b,tt
031  001 010           fmpr t,e
032  001 010           fmpr tt,e
033  001 011           fadr a,t
034  001 012           fadr b,tt
035  001 006           push p,a
036  016 158           move a,bpbri
037  018 054           movem a,@vdlipt
038  018 054           aos vdlipt
039  001 006           pop p,a
040  001 008           push p,c
041  014 045           pushj p,getpt
042  001 008           andcmi c,bit2.7+bit2.6+bit2.5
043  001 008           iori c,bit2.7+bit2.6
044  018 054           movem c,@vdlipt
045  018 054           aos vdlipt
046  001 008           pop p,c
047
048  001 009           move e,d
049  001 011           move c,t
050  001 012           move d,tt
051  001 008           move a,c
052  001 009           move b,d
053  016 116           pushj p,incrst
054
055  001 008  cirlup:  move a,c          ;x
056  016 074           fmpr a,ccosi      ;x cosi
057  001 009           move b,d
058  016 075           fmpr b,csini      ;y sini
059  001 007           fsbr a,b          ;new x = x*cosi - y*sini
060  016 075           fmpr c,csini      ;x sini
061  016 074           fmpr d,ccosi      ;y cosi
062  001 009           fadr c,d          ;new y = x*sini + y*cosi
063  001 008           move b,c
064
065  001 006           move c,a
066  001 007           move d,b
067  016 077           pushj p,drincp
068  016 055           sojg e,cirlup
069  016 128           pushj p,incre
070  001 012           pop p,tt
071  001 011           pop p,t
072  001 019           popj p,
073
074             ccosi:  0
075             csini:  0
076
077  003 150  drincp:  ifix a
078  003 150           ifix b
079  001 011           sub a,t
080  001 012           sub b,tt
081
082  016 005           trz f,%ixneg+%ixpos+%iyneg+%iypos
083  016 087           jumpe a,drin1
084  001 006           call a,
```

```
085  016 003          troa f,%ixpos
086  016 004          tro f,%ixneg
087  016 091  drin1:  jumpe b,drin2
088  001 007          call b,
089  016 005          troa f,%iypos
090  016 006          tro f,%iyneg
091  016 005  drin2:  trnn f,%ixneg+%ixpos+%iyneg+%iypos
092  001 019          popj p,
093  001 006          add t,a
094  001 007          add tt,b
095  016 157          move a,incwd
096  016 156          move b,inccnt
097  016 003          trne f,%ixneg+%ixpos
098  016 136          jrst [tro a,@ixenb(b)
099  016 004                trne f,%ixneg
100  016 141                tro a,@ixbit(b)
101                         jrst .+1]
102  016 005          trne f,%iyneg+%iypos
103  016 146          jrst [tro a,@iyenb(b)
104  016 006                trne f,%iyneg
105  016 151                tro a,@iybit(b)
106                         jrst .+1]
107  018 054          soji b,[idpb a,vdlipt
108  001 006                 movei a,200000
109  001 007                 movei b,3
110                          jrst .+1]
111  016 157          movem a,incwd
112  016 156          movem b,inccnt
113  001 019          popj p,
114
115
116  003 150  incrst: ifix a
117  003 150          ifix b
118  001 006          move t,a
119  001 007          move tt,b
120  001 006          movei a,200000
121  016 157          movem a,incwd
122  001 007          movei b,3
123  016 156          movem b,inccnt
124  001 006          movei a,442200
125  018 054          hrlm a,vdlipt
126  001 019          popj p,
127
128  001 006  incre:  movei a,400000
129  018 054          idpb a,vdlipt
130  001 006          setz a,
131  018 054          idpb a,vdlipt
132  018 054          idpb a,vdlipt
133  018 054          hrrzs vdlipt
134  001 019          popj p,
135
136            ixenb:  bit1.4
137                    bit1.8
138                    bit2.3
139                    bit2.7
140
141            ixbit:  bit1.3
142                    bit1.7
143                    bit2.2
144                    bit2.6
145
146            iyenb:  bit1.2
147                    bit1.6
148                    bit2.1
149                    bit2.5
150
151            iybit:  bit1.1
152                    bit1.5
153                    bit1.9
154                    bit2.4
155
156            inccnt: 0
157            incwd:  0
158            bpbri:  20112
```

```
001
002                     subttl utility subroutines and data bases
003             define player num,list
004     017 005                 dlplr!num=%%dp
005     017 005                 %%dp=%%dp+5     ;bright/scale wd,point wd,2 vector wds,+back to pt mode
006             dwplr!num:
007     017 006 relfig num,[list]
008             termin
009
010             ;macro for routine to display pts relative to starting pt and angle.
011             ;1 is along angle, 2 is 45 deg ccw, 3 is 90 deg ccw to angle, etc.
012             define relfig num,list
013     001 006                 push p,a
014     001 007                 push p,b
015     001 008                 push p,c
016     001 009                 push p,d
017             ;               push p,e        ;if using maskoff--allows wraparound on screen
018     003 158                 frifix a        ;convert x and
019     003 158                 frifix b        ;y to fractional integer form
020     017 056                 pushj p,csncvt  ;convert sin/cos in c/d to various needed values
021             ;               move e,[776000,,776000] ;if using maskoff--flush high order bits of coords
022     001 009                 move d,[220000,,022000] ;point mode bits IOR'd with coords
023             irpc pt,,[list]
024     017 084 ifse [pt][1][relpnt rsin,rcos, ? .stop]
025     017 088 ifse [pt][2][relpnt rsn45,rcs45, ? .stop]
026     017 083 ifse [pt][3][relpnt rcos,nrsin, ? .stop]
027     017 087 ifse [pt][4][relpnt rcs45,nrsn45, ? .stop]
028     017 085 ifse [pt][5][relpnt nrsin,nrcos, ? .stop]
029     017 089 ifse [pt][6][relpnt nrsn45,nrcs45, ? .stop]
030     017 082 ifse [pt][7][relpnt nrcos,rsin, ? .stop]
031     017 086 ifse [pt][8][relpnt nrcs45,rsn45, ? .stop]
032     001 007 ifse [pt][S][   push p,a ? push p,b
033             ]
034     001 006 ifse [pt][U][   pop p,b ? pop p,a
035             ]
036             termin
037             ;               pop p,e ;if using maskoff
038     001 009                 pop p,d
039     001 008                 pop p,c
040     001 007                 pop p,b
041     001 006                 pop p,a
042     001 019                 popj p,
043             termin
044
045             define relpnt yrel,xrel
046     001 006                 add a,xrel
047     001 007                 add b,yrel
048     001 007                 hllz c,b        ;y coord first
049     001 006                 hlr c,a
050             ;               and c,e ;to mask bits off if necessary
051     001 009                 ior c,d
052     017 005                 movem c,%%dp
053     017 005 %%dp=%%dp+1
054             termin
055
056     001 006 csncvt: push p,a
057     001 007                 push p,b
058     001 008                 move a,c
059     001 009                 move b,d
060     003 158                 frifix c
061     017 082                 movem c,rsin
062     017 083                 movnm c,nrsin
063     001 006                 move c,a        ;get floating sin back
064     001 007                 fadr c,b
065     001 008                 fmpr c,[.7071]  ; (1/sqrt(2))*(sin + cos)=sin(ang+45 deg)
066     003 158                 frifix c
067     017 086                 movem c,rsn45
068     017 087                 movnm c,nrsn45
069
070     003 158                 frifix d
071     017 084                 movem d,rcos
072     017 085                 movnm d,nrcos
073     001 006                 fsbr b,a        ;cos-sin
074     001 007                 fmpr b,[.7071]
075     003 158                 frifix b
076     017 088                 movem b,rcs45
077     017 089                 movnm b,nrcs45
078     001 007                 pop p,b
079     001 006                 pop p,a
080     001 019                 popj p,
081
082             rsin:   0
083             nrsin:  0
084             rcos:   0
```

```
085          nrcos:  0
086          rsn45:  0
087          nrsn45: 0
088          rcs45:  0
089          nrcs45: 0
```

```
001
002  003 140  dpradr: repeat nplyrs,[conc dwplr,\.rpcnt+1
003           ]              ;addr of routine for each player that mungs dlist
004  018 021  dlplst: repeat nplyrs,[conc dlplr,\.rpcnt+1,+dpstrt
005           ]              ;addr of first word in display list for player
006
007           ;routines to update player ID, given isin/icos/isin45/icos45 and x,y in c,d
008           ;--all using fractional integer format.
009
010                   if1 %%dp=0
011  018 021          if2 %%dp=dpstrt
012  017 003  player 1,[1111144444466666611111]       ;single triangle
013  017 003  player 2,[S22222288888U444444666666]   ;double triangle
014  017 003  player 3,[111333333555555777777]       ;single square
015  017 003  player 4,[333333S111111777777U555555777777]     ;double square
016
017  018 019  DBDPT:  -DLLEN,,DLSTRT-1
018
019           dlstrt: block 500        ;reserved for board
020                   bit4.9           ;to break out of vector mode just in case
021  017 005  dpstrt: if1 loc .+%%dp  ;reserve however many words the board needs
022  017 005          if2 repeat %%dp-., 0
023  018 019  DLLEN==.-DLSTRT
024                   3000     ;interrupt display when reach end of "fixed" d-list
025
026           vdllen==500      ;max size of variable-length display list
027           VDIS1:
028  018 026  vdlst1: block vdllen     ;idea is that 340 crunches one while APR creates the other
029           VDIS2:
030  018 026  vdlst2: block vdllen
031
032
033  018 026  VDSLEN==VDLLEN
034           VDBR:   0
035           VDCUR:  0
036           VDSW:   0
037  018 027  VDPTAB: -VDSLEN,,VDIS1-1
038  018 029          -VDSLEN,,VDIS2-1
039
040
041  018 050  DSTL:   VDPTR,,[BDPT,,0]
042  018 050          VDPTR+1,,[BDPT,,0]
043
044  018 050  DSTL3:  BDPT,,0
045
046           DISLST: 0                ;PTR TO CURRENT DISLIST, SET BY DSTART.
047
048           VDPTR:  0        ;HOLDS BLKO FOR VDIS1
049                   0        ;FOR VDIS2
050           BDPT:   0
051
052
053
054           vdlipt: 0        ;holds ptr into a vdlist for APR deposit of halfwords
055
056           vdlapr: 0        ;# of vdl that APR is munging
057           vdl340: 0        ;# of vdl that 340 is munging
058           vdl1st: 0        ;status of vdl 1; -1 = 340, 0=free waiting for 340, 1= APR
059           vdl2st: 0        ;status for vdl 2
060
061           vbfnum: 0        ;flag for 340 routine, -1 means hacking vlist, 0 mainlist
062           vdlcnt: 0        ;# times display has shown list since start of use
063           vdlflg: 0        ;when APR wants to switch vlists, puts addr of new one here.
064           vdlloc: 0        ;340 routine stores current vlist start addr here.
065           disptr: 0        ;BLKO ptr wd.
066
067  005 004  end go
```

```
%%1     = 002 011   BPROSQ     008 116   FLG      = 001 029   P%CENB   = 002 051   TIMOUT    011 075
%%2     = 002 011   BPROT      008 115   FLG      = 001 030   P%CLR    = 002 049   TMMATE    008 053
%%3     = 002 010   BREAK      006*010   FLG      = 001 031   P%OFF    = 002*053   TMON      008 058
%%3     = 002 011   BSET2      009 048   FLG      = 001 032   P%ON     = 002 054   TMSCOR    008 080
%%DP    = 017 005   BSET3      009 066   FLG      = 001 033   P%PIUP   = 002*058   TSINT     004 033
%%DP    = 017 053   BSETP1     009 040   FRIFIX   M 003 158   PASLIM   008 031   TSTFIR    006 056
%%DP    = 018 010   BSETUP     009 012   FRZTIM   008 118   PASS     = 008*076   TSTPAS    006 050
%%DP    = 018 011   C       = 001 008   FUZZM    008 027   PASSTM   008 032   TT      = 001 012
%DEXCH  = 002 074   C       = 002 008   FUZZR    008 028   PATCH    002*109   UPDAT     012 003
%DXCH   = 015 033   C127.      015 116   GAME     005 041   PBDGAP   008 093   UPDAT1    012 087
%IXNEG  = 016 004   CCOSI      016 074   GAME1    005 049   PDL      002 112   UPDAT2    012 095
%IXPOS  = 016 003   CHANS      011 079   GAML01   006 029   PDLLEN   = 002 111   UPDAT3    012 132
%IYNEG  = 016 006   CIRLUP     016 055   GAMLP0   006 038   PFADR    M 003*038   UPDAT4    012 038
%IYPOS  = 016 005   CLKCNT     004 025   GAMLP1   006 063   PFDVR    M 003 020   UPDATI    012 020
%LNCHK  = 002 073   CLKSNK     004 026   GAMLUP   006 004   PFMPR    M 003 006   VBFEND    004 086
%M1INF  = 002 070   CONC    M 003 140   GETPOTS   011 033   PHIGHX     008 096   VBFNUM    018*061
%M2INF  = 002 071   CONTROL    008 063   GETPT    014 045   PHIGHY     008 097   VDBR      018*034
%PFNEG  = 002 069   COS        007 054   GETVEC   014 066   PI$2       002 119   VDCUR     018 035
%RSTRT  = 002 075   COSD       007*053   GHIGHY   010 006   PI.5       002 120   VDIS1     018 027
%ZFIRE  = 002 072   CPI1.5  = 002 117   GLOWY    010 005   PIE        002*118   VDIS2     018 029
..D574  = 002*084   CPIE    = 002 115   GO       005 004   PINITA     008 018   VDL1ST    018*058
.MLLIT  = 001*003   CPIE.5  = 002 116   GOALHT   010 011   PINITD     008 008   VDL2ST    018*059
A       = 001 006   CSINI      016 075   GOLHT0   010 023   PINITX     008 010   VDL340    018*057
A%CCE   = 002*038   CSNCVT     017 056   GOLHT1   010 029   PINITY     008 014   VDLAPR    018*056
A%CCEO  = 002 023   D       = 001 009   I%CCW    = 011 008   PLANG      008 039   VDLCNT    018*062
A%CCEZ  = 002 022   D%CONT  = 002*064   I%CLKW   = 011 007   PLAY1      008*064   VDLFLG    018*063
A%CLK   = 002 039   D%INIT  = 002 063   I%FIRE   = 011 005   PLAY2      008*065   VDLIPT    018 054
A%CLKZ  = 002 024   DBDPT      018 017   I%NEGX   = 011 012   PLAY3      008*066   VDLLEN  = 018 026
A%ILOP  = 002*036   DCIRC      016 008   I%NEGY   = 011 010   PLAY4      008*067   VDLLOC    018*064
A%NXM   = 002*037   DCONST     008 029   I%PASS   = 011 006   PLAYER   M 017 003   VDLST1    018*028
A%OVE   = 002*042   DDT     = 004 028   I%POSX   = 011 011   PLCOS      008 043   VDLST2    018*030
A%OVEO  = 002*029   DEATH      004 029   I%POSY   = 011 009   PLEN       008 047   VDPTAB    018 037
A%OVEZ  = 002 028   DISBRO     004 047   IA       = 001 018   PLENSQ     008 049   VDPTR     018 048
A%OVF   = 002*043   DISBRK     004 046   IFIX     M 003 150   PLOCX      008 035   VDSLEN  = 018 033
A%OVFZ  = 002 030   DISCHN  = 002 081   IMX      = 002 083   PLOCY      008 037   VDSW      018 036
A%PCE   = 002*040   DISLST     018 046   IMXCHN   = 002 082   PLOWX      008 094   WAIT      011*035
A%PCEO  = 002*026   DISPIC  = 002 088   INCCNT     016 156   PLOWY      008 095   X1        003 070
A%PCEZ  = 002 025   DISPTR     018*065   INCRE      016 128   PLR      = 001 014   X2        003 073
A%PCF   = 002*041   DISRAN     013 059   INCRST     016 116   PLRBRI     008 005   XCONT     015 084
A%PCFZ  = 002 027   DISRN1     013 061   INCWD      016 157   PLRDIS     013 004   XINC      015*118
A%POV   = 002*035   DISTSQ     007 004   INMODE     011 095   PLSCOR     008 051   XLEFT   = 008*072
A%RIO   = 002 019   DL      = 001 017   INPGET     011 014   PLSIN      008 041   XMOVE   = 011 092
A%RMP   = 002 020   DLINE   M 009 004   INPGT1     011 022   PLSLOP     008 045   XRIGHT  = 008*071
A%RNXM  = 002 021   DLLEN   = 018 023   INPUT      008 023   POPAE    M 013 051   Y1        003 071
A%RPOV  = 002 018   DLPLST     018 004   INPUTS     011 089   PRANDM     010 051   Y2        003 074
A%STRT  = 002 033   DLSTRT     018 019   INTCAL     003 077   PUSHAE   M 013 046   YDOWN   = 008*074
ANGCHK    007 013   DNUM        014 032   INTRS2     003 120   PWB      = 001 013   YINC      015*117
ANGCHL    007 027   DNUM1       014 035   INTRS7     003 128   PWBBRI     008 006   YMOVE   = 011 093
ANGCHW    007 032   DNUM2       014 040   INTRS9     003 133   RAN        010 059   YUP     = 008*073
ANGRAT    008 025   DP       = 001 016   INTRSC   M 003 054   RANDOM     010 054   ZANGLE    008 124
ANGSPD    008 026   DPRADR     018 002   INVAL      011 090   RANSW    V 006 024   ZASS1     010 044
APRBR1    004 016   DPSTRT     018 021   IXBIT      016 141   RCOS       017 084   ZASSGN    010 043
APRBR2    004 019   DRAWL      015*002   IXENB      016 136   RCS45      017 088   ZBESTD    008 131
APRBRK    004 005   DRIN1      016 087   IYBIT      016 151   REFLCT     007 039   ZBESTP    008 132
APRCHN  = 002 079   DRIN2      016 091   IYENB      016 146   RELFIG   M 017 012   ZBESTX    008 133
APRCNS    004 023   DRINCP     016 077   JOV      = 003 004   RELPNT   M 017 045   ZBESTY    008 134
APRPIC  = 002 086   DRWLIN     015 055   LGLHIT     008 079   ROTATE   = 011 091   ZBNCLT    008 137
B       = 001 007   DRWLN5     015*051   LOOP5      011 042   ROTLEFT  = 008*070   ZBOUNC    008 135
B       = 002 007   DRWVEC     014 058   LOOP6      011 063   ROTRIGHT = 008*069   ZBRI      008 138
BDBRI     010 007   DSPBR1     004 061   LOOP7      012 025   RSIN       017 082   ZCOS      008 126
BDCENX    008 112   DSPBRK     004 057   LTMSCR     008 078   RSN45      017 086   ZFROM     008 129
BDCENY    008 113   DSPCHN  = 002 080   M1         003 072   SC1        007 087   ZLUP      006 093
BDLENX    008 108   DSPPIC  = 002 087   M2         003 075   SC3        007 088   ZLUP0     006 094
BDLENY    008 109   DSTART   M 004 071   MAXFLO     003 003   SC5        007 089   ZLUP1     006 098
BDPT      018 050   DSTART   M 004 079   NEGPOT     011 084   SC7        007 090   ZLUP2     006 114
BDSIZ     010 008   DSTL       018 041   NEXT       011 053   SC9        007 091   ZLUP25    006 125
BGHIGH    008 105   DSTL3      018 044   NEXTSW     012 035   SER1       005 029   ZLUP3     006 133
BGLDEP    010 004   DTDISP     014 005   NOFIG      015 103   SERIES     005*023   ZLUP50    006 138
BGLFLG    008 099   DTDSP9     014 023   NPLYRS   = 001 021   SETF     M 001 024   ZLUP90    006 179
BGLOW     008 104   DTLOC      013 082   NRCOS      017 085   SIN        007 055   ZMAXBC  = 008 136
BGWIDE    008 106   DTMAX   = 013 071   NRCS45     017 089   SIND       007*050   ZSIN      008 125
BHIGHX    008 089   DTSCBR     013 079   NRSIN      017 083   SLPLIM     003 052   ZSLOPE    008 127
BHIGHY    008 091   DTX        013 073   NRSN45     017 087   SPEED      008 024   ZSLOPX    008*128
BITDEF  M 002 006   DTY        013 076   NUM        017 006   SPLEN      008 030   ZSTRTX    008 122
BLN.5X    008*110   E       = 001 010   NUM      = 017 004   SSFIX    M 003 143   ZSTRTY    008 123
BLN.5Y    008*111   F       = 001 005   P        = 001 019   STEPTM     004 024   ZTIM60    008 085
BLOWX     008 088   FIRE    = 008 075   P%ACT    = 002*050   SWITCHES   011 096   ZTIME     008 083
BLOWY     008 090   FIRE    = 011 094   P%CDSB   = 002*052   T        = 001 011   ZTMLFT    008 084
BPBRI     016 158   FLG     = 001 028
```

```
%%1     002=011 002-012
%%2     002=011 002-012
%%3     002=010 002=011 002-011 002-012
%%DP    017=005 017=053 018=010 018=011 017-004 017-005 017-052 017-053 018-021 018-022
%DEXCH  002=074 015-079 015-081 015-089 015-101
%DXCH   015=033 015-034 015-035 015-038
%IXNEG  016=004 016-082 016-086 016-091 016-097 016-099
%IXPOS  016=003 016-082 016-085 016-091 016-097
%IYNEG  016=006 016-082 016-090 016-091 016-102 016-104
%IYPOS  016=005 016-082 016-089 016-091 016-102
%LNCHK  002=073 006-090 006-105 006-117 006-140 006-144
%M1INF  002=070 003-080 003-084 003-101 003-104
%M2INF  002=071 003-080 003-091 003-101 003-105
%PFNEG  002=069 003-010 003-011 003-016 003-028 003-029 003-034
%RSTRT  002=075 002-105 002-106 005-035
%ZFIRE  002=072 006-059 006-060 006-066
..D574  002=084
.MLLIT  001=003
A       001=006 002-007 002-007 003-055 003-056 003-057 003-058 003-059 003-060 003-061 003-062 003-063 003-064 003-065
        003-066 003-103 003-107 003-113 003-121 003-122 003-123 003-124 003-141 003-144 003-145 003-145 003-146 003-146
        003-151 003-152 003-153 003-153 003-154 003-159 003-160 003-161 003-161 003-162 004-089 004-090 004-092 004-093
        004-100 004-101 004-103 004-104 004-105 004-106 005-012 005-013 005-030 005-031 005-032 005-032 005-033 005-043
        005-044 005-045 005-046 005-047 005-049 005-050 005-051 005-052 005-053 005-054 005-067 005-068 005-069 005-070
        005-071 005-072 005-073 005-074 006-008 006-009 006-014 006-015 006-025 006-026 006-027 006-028 006-029 006-031
        006-033 006-034 006-036 006-037 006-038 006-039 006-040 006-041 006-043 006-044 006-050 006-056 006-057 006-058
        006-073 006-074 006-075 006-076 006-077 006-078 006-079 006-080 006-081 006-082 006-083 006-084 006-085 006-086
        006-087 006-088 006-098 006-099 006-107 006-108 006-134 006-147 006-153 006-154 006-155 006-158 006-159 006-160
        006-161 006-166 006-167 006-169 006-170 006-172 006-173 006-174 006-179 006-191 006-192 006-193 007-005 007-017
        007-039 007-040 007-041 007-042 007-042 007-043 007-044 007-050 007-053 007-054 007-055 007-056 007-059 007-060
        007-062 007-063 007-063 007-064 007-065 007-066 007-067 007-068 007-068 007-069 007-071 007-072 007-072 007-074
        007-076 007-078 007-080 007-081 007-082 007-084 009-005 009-012 009-013 009-013 009-014 009-015 009-016 009-017
        009-018 009-019 009-020 009-021 009-025 009-027 009-041 009-043 009-045 009-047 009-048 009-054 009-056 009-057
        009-059 009-060 009-061 009-066 009-069 009-073 009-076 009-081 009-082 009-083 009-084 009-085 009-093 009-094
        009-096 009-100 009-105 009-106 009-107 009-108 009-116 009-117 009-118 009-119 009-120 009-121 009-132 009-139
        009-146 009-147 009-148 010-011 010-013 010-017 010-035 010-036 010-045 010-052 010-052 010-054 010-055 010-056
        010-057 010-057 011-014 011-017 011-019 011-020 011-020 011-022 011-029 011-030 011-035 011-036 011-040 011-042
        011-051 011-053 011-057 011-058 011-059 011-060 011-062 011-063 011-068 011-070 011-072 011-073 011-076 012-020
        012-021 012-025 012-031 012-035 012-042 012-043 012-044 012-046 012-047 012-049 012-050 012-051 012-052 012-053
        012-077 012-079 012-081 012-084 012-095 012-101 012-113 012-116 012-117 012-118 012-119 012-120 012-121 012-132
        012-134 012-135 012-137 012-138 012-139 012-140 013-004 013-007 013-009 013-011 013-019 013-020 013-028 013-028
        013-035 013-036 013-038 013-059 013-064 013-068 014-012 014-013 014-014 014-015 014-019 014-023 014-024 014-025
        014-027 014-032 014-033 014-033 014-035 014-036 014-045 014-047 014-049 014-054 014-066 014-068 014-070 014-073
        014-075 014-076 014-076 014-077 014-079 014-080 014-082 015-002 015-014 015-018 015-019 015-025 015-027 015-027
        015-028 015-030 015-030 015-032 015-037 015-039 015-040 015-044 015-045 015-046 015-055 015-071 015-078 015-084
        015-086 015-086 015-087 015-090 015-096 015-098 015-098 015-099 015-102 015-113 016-010 016-012 016-013 016-015
        016-017 016-019 016-020 016-022 016-024 016-026 016-027 016-029 016-033 016-035 016-036 016-037 016-039 016-051
        016-055 016-056 016-059 016-065 016-077 016-079 016-083 016-084 016-093 016-095 016-098 016-100 016-103 016-105
        016-107 016-108 016-111 016-116 016-118 016-120 016-121 016-124 016-125 016-128 016-129 016-130 016-131 016-132
        017-013 017-018 017-032 017-034 017-041 017-046 017-049 017-056 017-058 017-063 017-073 017-079
AXCCE   002=038
AXCCEO  002=023 004-016 005-007
AXCCEZ  002=022 002-033
AXCLK   002=039 004-007
AXCLKZ  002=024 002-033 004-016
AXILOP  002=036
AXNXM   002=037
AXOVE   002=042
AXOVEO  002=029
AXOVEZ  002=028 002-033
AXOVF   002=043
AXOVFZ  002=030 002-033
AXPCE   002=040
AXPCEO  002=026
AXPCEZ  002=025 002-033
AXPCF   002=041
AXPCFZ  002=027 002-033
AXPOV   002=035
AXRIO   002=019 002-033
AXRMP   002=020 002-033
AXRNXM  002=021 002-033
AXRPOV  002=018 002-033
AXSTRT  002=033 005-006
ANGCHK  007:013 006-114
ANGCHL  007:027 007-024
ANGCHW  007:032 007-024 007-026
ANGRAT  008:025 012-107 012-112
ANGSPD  008:026 012-108
APRBR1  004:016 004-012
APRBR2  004:019 004-008
APRBRK  004:005 002-092 004-017
APRCHN  002=079 002-091 004-016 005-006 005-007
```

```
D%INIT   002=063 004-068 004-074 005-025
DBDPT    018:017 005-071
DCIRC    016:008 009-137 009-144
DCONST   008:029 012-004
DDT      004=028 004-030
DEATH    004:029 004-020 006-141 009-028
DISBR0   004:047 004-051 004-053
DISBRK   004:046 002-097 004-054
DISCHN   002=081 002-095 004-068 004-074
DISLST   018:046 004-048 004-062 004-072 004-080 004-101 005-074
DISPIC   002=088 005-009
DISPTR   018:065
DISRAN   013:059 004-086
DISRN1   013:061 013-067
DISTSQ   007:004 006-121 006-127 012-073
DL       001=017 004-047 004-048 004-049 004-049 004-050 004-061 004-062 004-063 004-063 004-064 004-072
DLINE    009+004 009-111 009-112 009-113 009-114 009-123 009-124 009-125 009-127 009-128 009-129
DLLEN    018=023 018-017
DLPLST   018:004 013-008
DLSTRT   018:019 009-105 009-147 018-017 018-023
DNUM     014:032 014-020
DNUM1    014:035 014-038
DNUM2    014:040 014-036
DP       001=016 002-096 004-050 004-051 004-052 004-052 004-064 004-065 004-066 004-066 004-073
DPRADR   018:002 013-042
DPSTRT   018:021 018-004 018-011
DRAWL    015:002
DRIN1    016:087 016-083
DRIN2    016:091 016-087
DRINCP   016:077 016-067
DRWLIN   015:055 006-151 006-183 009-009
DRWLN5   015:051
DRWVEC   014:058 015-091
DSPBR1   004:061 004-065 004-067
DSPBRK   004:057 002-094 004-069
DSPCHN   002=080 002-093 004-068 004-074
DSPPIC   002=087 005-009
DSTART   004+071 004+079 005-075
DSTL     018:041 004-100
DSTL3    018:044 005-073
DTDISP   014:005 013-065
DTDSP9   014:023 014-021
DTLOC    013:082 013-061
DTMAX    013=071 013-060
DTSCBR   013:079 013-062
DTX      013:073 009-083 009-084 009-085 013-064
DTY      013:076 009-087 009-089 009-091 013-063
E        001=010 003-079 003-098 003-125 003-128 003-133 007-015 007-021 007-023 007-028 007-033 009-136 009-143 012-005
         012-006 012-010 012-011 012-012 012-013 012-014 012-015 012-016 012-017 012-022 012-023 012-024 012-027 012-030
         012-034 012-036 012-038 012-039 012-043 012-056 012-076 012-080 012-083 012-087 012-089 012-092 012-101 012-103
         013-059 013-060 013-061 013-062 013-063 013-064 013-067 013-068 015-006 015-040 015-042 015-059 015-073 015-075
         015-080 015-081 015-092 015-093 015-096 015-109 016-013 016-031 016-032 016-048 016-068
F        001=005 002-105 002-106 003-010 003-011 003-016 003-028 003-029 003-034 003-080 003-084 003-091 003-101 003-104
         003-105 005-035 005-041 006-059 006-060 006-066 006-090 006-105 006-117 006-140 006-144 015-034 015-035 015-038
         015-079 015-081 015-089 015-101 016-082 016-085 016-086 016-089 016-090 016-091 016-097 016-099 016-102 016-104
FIRE     008=075 011=094 006-050 006-058 012-016 012-017
FLG      001=028 001=029 001=030 001=031 001=032 001=033 001-025 001-034 001-036
FRIFIX   003+158 014-047 014-048 014-068 014-069 015-024 015-047 017-018 017-019 017-060 017-066 017-070 017-075
FRZTIM   008:118 010-026
FUZZM    008:027 012-045 012-058
FUZZR    008:028 012-105
GAME     005:041 005-036 006-198
GAME1    005:049 005-058
GAML01   006:029 006-031
GAMLP0   006:038 006-027
GAMLP1   006:063 006-064
GAMLUP   006:004 006-067 006-162 006-187 006-190 006-194
GETPOTS  011:033 011-028
GETPT    014:045 013-013 014-007 015-008 015-065 016-041
GETVEC   014:066 013-022 013-030 014-059 015-103
GHIGHY   010:006 009-121 009-123 009-124 009-124 009-125
GLOWY    010:005 009-118 009-127 009-128 009-128 009-129
GO       005:004 002-107 018-067
GOALHT   010:011 006-197
GOLHT0   010:023 010-020
GOLHT1   010:029 010-038
I%CCW    011=008 012-015
I%CLKW   011=007 012-014
I%FIRE   011=005 012-017
I%NEGX   011=012 012-011
I%NEGY   011=010 012-013
I%PASS   011=006 012-016
I%POSX   011=011 012-010
```

```
VDBR    018:034
VDCUR   018:035 004-092 004-098 004-104 005-068
VDIS1   018:027 018-037
VDIS2   018:029 018-038
VDL1ST  018:058
VDL2ST  018:059
VDL340  018:057
VDLAPR  018:056
VDLCNT  018:062
VDLFLG  018:063
VDLIPT  018:054 004-088 004-106 005-070 009-104 009-106 009-108 009-109 009-146 009-149 014-005 014-006 014-010 014-011
        014-013 014-015 014-024 014-027 014-029 014-032 014-041 014-061 014-062 015-063 015-064 015-068 015-069 015-105
        015-106 016-037 016-038 016-044 016-045 016-107 016-125 016-129 016-131 016-132 016-133
VDLLEN  018=026 018-028 018-030 018-033
VDLLOC  018:064
VDLST1  018:028
VDLST2  018:030
VDPTAB  018:037 004-103 005-067
VDPTR   018:048 004-099 018-041 018-042
VDSLEN  018=033 018-037 018-038
VDSW    018:036 004-087 004-102 005-066
WAIT    011:035
X1      003:070 003-058 003-086 003-107 003-109 009-005
X2      003:073 003-064 003-093 003-113 003-114 009-007
XCONT   015:084 015-094
XINC    015:118
XLEFT   008=072
XMOVE   011=092 012-010 012-011 012-043 012-076 012-080 012-083
XRIGHT  008=071
Y1      003:071 003-060 003-087 009-006
Y2      003:074 003-066 003-094 009-008
YDOWN   008=074
YINC    015:117
YMOVE   011=093 012-012 012-013 012-056 012-087 012-089 012-092
YUP     008=073
ZANGLE  008:124 006-077 006-166 007-041
ZASS1   010:044 010-047
ZASSGN  010:043 005-042
ZBESTD  008:131 006-094 006-128 006-130 006-133
ZBESTP  008:132 006-095 006-136 006-139 006-156
ZBESTX  008:133 006-134 006-149 006-181 006-191
ZBESTY  008:134 006-135 006-150 006-182 010-011
ZBNCLT  008:137 006-154
ZBOUNC  008:135 006-072 006-153 006-186
ZBRI    008:138 015-062
ZCOS    008:126 006-079 006-087 006-169 006-173 007-025
ZFROM   008:129 006-089 006-099 006-175
ZLUP    006:093 006-176
ZLUP0   006:094 006-143
ZLUP1   006:098 006-138
ZLUP2   006:114 006-106
ZLUP25  006:125 006-118
ZLUP3   006:133 006-129
ZLUP50  006:138 006-100 006-103 006-109 006-112 006-115 006-123 006-131
ZLUP90  006:179 006-145
ZMAXBC  008=136 006-155 008-137
ZSIN    008:125 006-081 006-086 006-172 007-024
ZSLOPE  008:127 006-088 006-102 006-174
ZSLOPX  008:128
ZSTRTX  008:122 006-083 006-102 006-125 006-147 006-163 006-179 007-019
ZSTRTY  008:123 006-085 006-102 006-126 006-148 006-164 006-180 007-020
ZTIM60  008:085 005-046 006-025 006-038 006-161 010-054
ZTIME   008:083 005-043 006-036 006-158
ZTMLFT  008:084 005-044 006-041 006-159 013-083
```