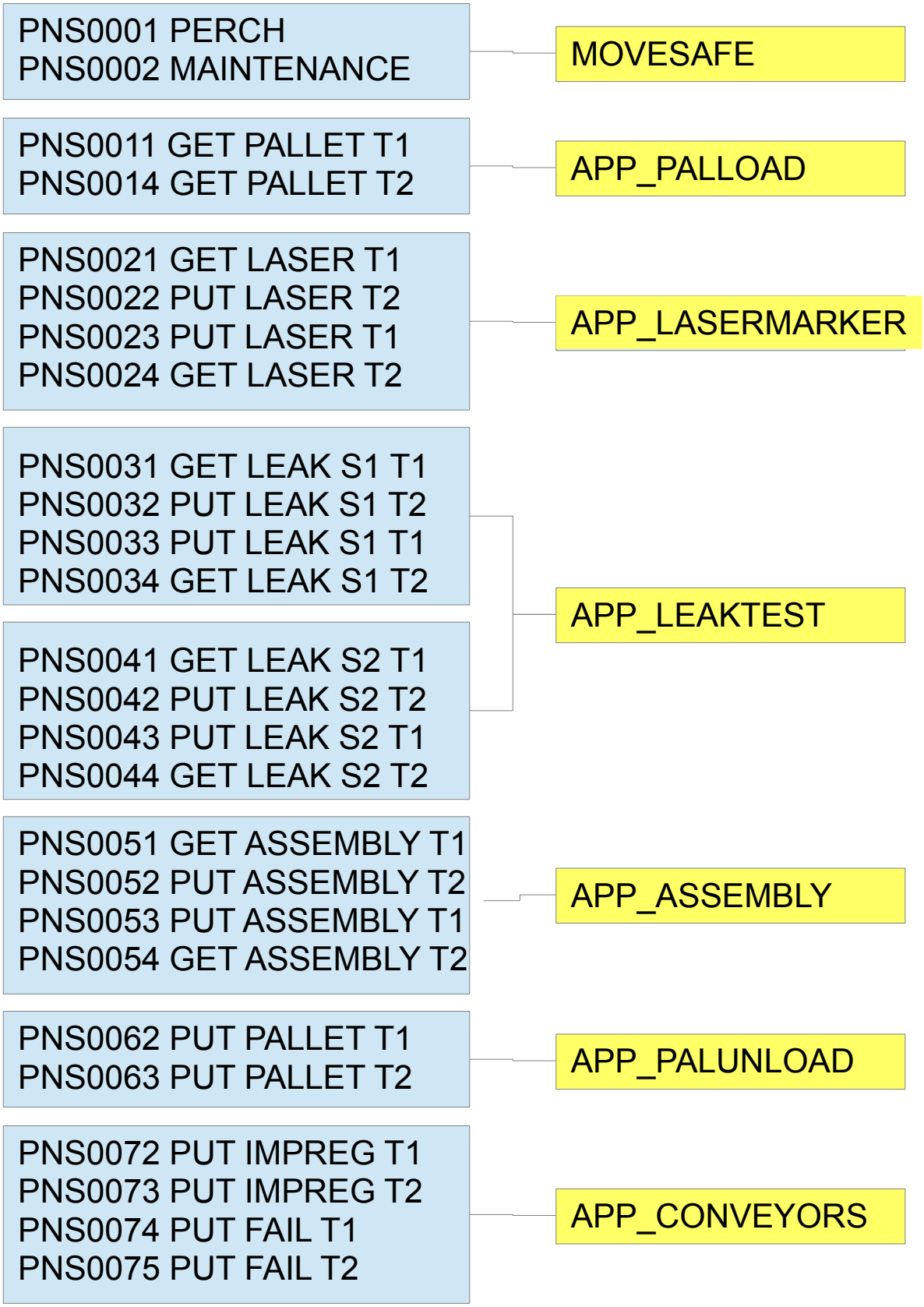
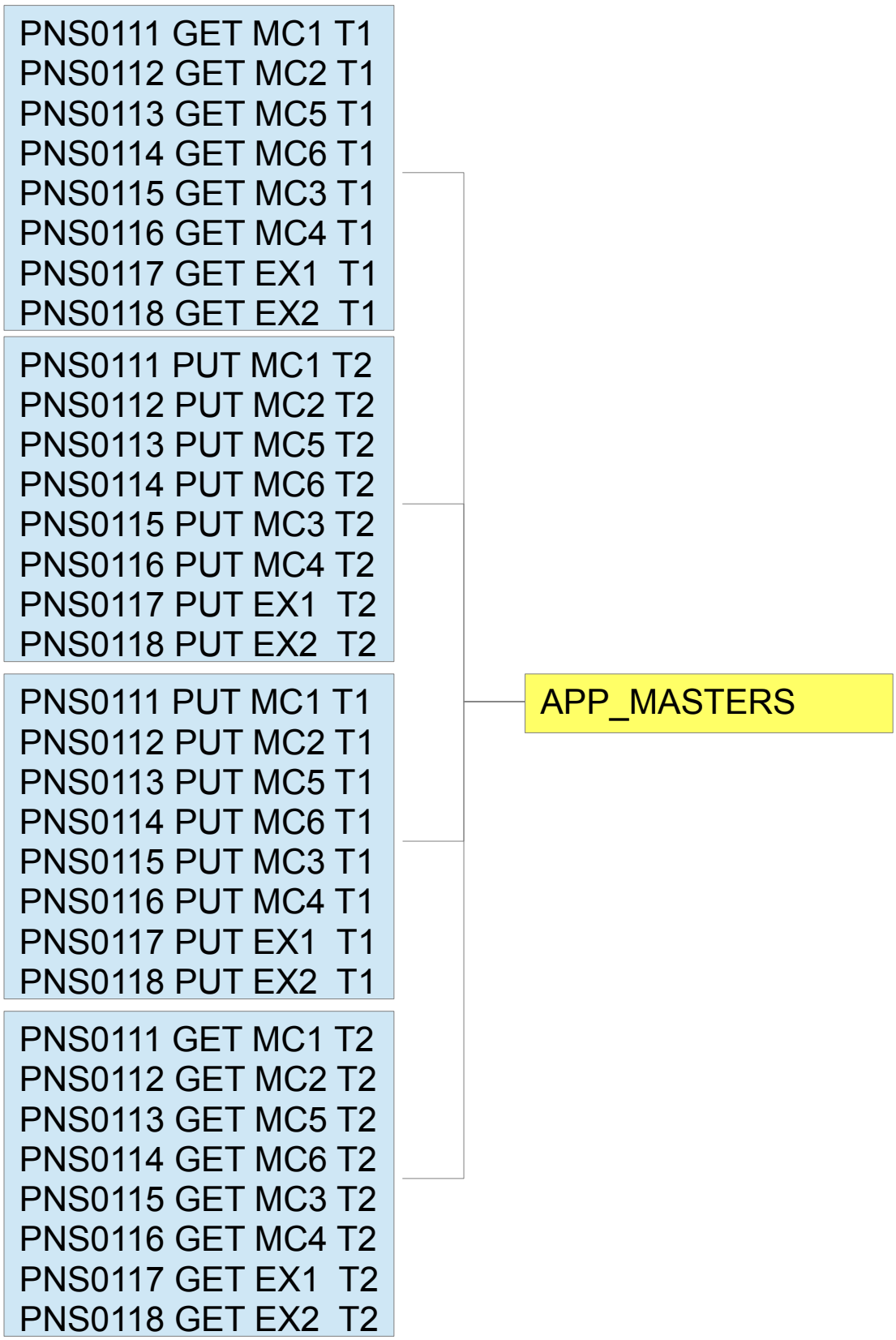
The main cell controller commands up to 66 codes to the Robot's Digital Input siganls to execute PNSxxxx programs which have been gruped to call an Application program.




PNS0111 GET MC1 T1
PNS0112 GET MC2 T1
PNS0113 GET MC5 T1
PNS0114 GET MC6 T1
PNS0115 GET MC3 T1
PNS0116 GET MC4 T1
PNS0117 GET EX1 T1
PNS0118 GET EX2 T1
PNS0111 PUT MC1 T2
PNS0112 PUT MC2 T2
PNS0113 PUT MC5 T2
PNS0114 PUT MC6 T2
PNS0115 PUT MC3 T2
PNS0116 PUT MC4 T2
PNS0117 PUT EX1 T2
PNS0118 PUT EX2 T2
PNS0111 PUT MC1 T1
PNS0112 PUT MC2 T1
PNS0113 PUT MC5 T1
PNS0114 PUT MC6 T1
PNS0115 PUT MC3 T1
PNS0116 PUT MC4 T1
PNS0117 PUT EX1 T1
PNS0118 PUT EX2 T1
PNS0111 GET MC1 T2
PNS0112 GET MC2 T2
PNS0113 GET MC5 T2
PNS0114 GET MC6 T2
PNS0115 GET MC3 T2
PNS0116 GET MC4 T2
PNS0117 GET EX1 T2
PNS0118 GET EX2 T2
APP_MASTERS

Program Listing of the APPLICATION Program APP_PALLOAD

```
/PROG  APP_PALLOAD
   1:  !Pick from Input Conv ;
   2:  IF (F[1:ToolA_is_Empty]=OFF) THEN ;
   3:      CALL APPFAULTS(2) ;
   4:      END ;
   5:  ENDIF ;
   6:   ;
   7:  CALL MOVESAFE(R[11:CONTINUOUS]) ;
   8:  !Configure ;
   9:  PR[73:CONFIG PALLET]=PR[70:SAFE COPY]    ;
  10:  PR[73,4:CONFIG PALLET]=90    ;
  11:   ;
  12:  PR[73,5:CONFIG PALLET]=(-90)    ;
  13:  IF (R[6:TOOLNUM]=1) THEN ;
  14:      PR[73,6:CONFIG PALLET]=180    ;
  15:  ELSE ;
  16:      PR[73,6:CONFIG PALLET]=0    ;
  17:  ENDIF ;
  18:J PR[73:CONFIG PALLET] 50% CNT100    ;
  19:   ;
  20:  !Set Positions ;
  21:  R[17:STEP POS IDX]=R[4:PREG INDEX]    ;
  22:  CALL STEP(90,0,(-400),325) ;
  23:  CALL STEP(91,0,(-100),100) ;
  24:  CALL STEP(92,0,(-40),50) ;
  25:  CALL STEP(93,0,2,50) ;
  26:  CALL STEP(94,0,2,350) ;
  27:  CALL STEP(95,0,(-400),450) ;
  28:   ;
  29:L PR[90:Clr1] 250mm/sec FINE    ;
  30:  COL DETECT ON ;
  31:   ;
  32:L PR[91:Clr2] 150mm/sec FINE    ;
  33:L PR[92:Clr3] 150mm/sec FINE    ;
  34:   ;
  35:  !Appro POS ;
  36:  CALL APPRO_DEPART(25,50) ;
  37:  !Move to POS ;
  38:  CALL ATTAUGHTPOS    ;
  39:   ;
  40:L PR[93:Clr4] 50mm/sec FINE    ;
  41:L PR[94:Clr5] 100mm/sec FINE    ;
  42:  COL DETECT OFF ;
  43:L PR[95:Clr6] 200mm/sec CNT100    ;
  44:  CALL MOVESAFE(R[12:FINE]) ;
/END
```

Program Listing of the APPLICATION Program APP_LASERMARKER

```
/PROG  APP_LASERMARKER
   1:  CALL MOVESAFE(R[11:CONTINUOUS]) ;
   2:  !LASER MARKER PICK & PLACE ;
   3:   ;
   4:  !SET Position STEPS ;
   5:  R[17:STEP POS IDX]=R[4:PREG INDEX]    ;
   6:  CALL STEP(90,(-700),0,125) ;
   7:  CALL STEP(91,0,0,125) ;
   8:  CALL STEP(92,0,0,35) ;
   9:   ;
  10:J PR[90:Clr1] 25% CNT10    ;
  11:  COL GUARD ADJUST 100 ;
  12:   ;
  13:  !ABOVE POSITION ;
  14:L PR[91:Clr2] 200mm/sec CNT10    ;
  15:L PR[92:Clr3] 150mm/sec CNT10    ;
  16:   ;
  17:  !Move to POS ;
  18:  CALL ATTAUGHTPOS    ;
  19:  !Depart POS ;
  20:  CALL APPRO_DEPART(25,50) ;
  21:  COL DETECT OFF ;
  22:   ;
  23:  !MOVE  CLEAR OF LASERMARKER ;
  24:L PR[91:Clr2] 150mm/sec CNT25    ;
  25:L PR[90:Clr1] 200mm/sec CNT50    ;
  26:  CALL MOVESAFE(R[12:FINE]) ;
/END
```

Program Listing of the APPLICATION Program APP_LEAKTEST

```
/PROG  APP_LEAKTEST
   1:  CALL MOVESAFE(R[11:CONTINUOUS]) ;
   2:   ;
   3:  ! Clear dist in Xy ;
   4:  R[15:ClearY]=500    ;
   5:  R[19:ClearX]=(-800)    ;
   6:   ;
   7:  !SET Position STEPS ;
   8:  R[17:STEP POS IDX]=R[4:PREG INDEX]    ;
   9:  IF (R[14:PICK/PLACE]=1) THEN ;
  10:      !PICK ;
  11:      CALL STEP(90,R[19:ClearX],R[15:ClearY],100) ;
  12:      CALL STEP(91,0,0,100) ;
  13:      CALL STEP(92,0,0,50) ;
  14:  ELSE ;
  15:      !PLACE ;
  16:      CALL STEP(90,R[19:ClearX],R[15:ClearY],200) ;
  17:      CALL STEP(91,0,0,200) ;
  18:      CALL STEP(92,0,0,50) ;
  19:  ENDIF ;
  20:   ;
  21:J PR[90:Clr1] 25% CNT50    ;
  22:L PR[91:Clr2] 150mm/sec CNT25    ;
  23:   ;
  24:  !AIR BLAST SECTION ;
  25:  IF (R[14:PICK/PLACE]=2 AND DI[31:RY_31_Ena_AirBlast]=ON) THEN ;
  26:      !simulate a place for PLC ;
  27:      !Func to OPEN/SHUT Gripper ;
  28:      CALL GRIPFUNCTION    ;
  29:      IF (R[6:TOOLNUM]=1) THEN ;
  30:          //RO[5:AIRJET 1A]=PULSE,4.0sec ;
  31:      ENDIF ;
  32:      IF (R[6:TOOLNUM]=2) THEN ;
  33:          //RO[5:AIRJET 1A]=PULSE,4.0sec ;
  34:      ENDIF ;
  35:      JMP LBL[1] ;
  36:  ENDIF ;
  37:   ;
  38:L PR[92:Clr3] 100mm/sec FINE    ;
  39:  COL GUARD ADJUST 100 ;
  40:   ;
  41:  !Move to POS ;
  42:  CALL ATTAUGHTPOS    ;
  43:  !Depart POS ;
  44:  CALL APPRO_DEPART(25,50) ;
  45:   ;
  46:  COL DETECT OFF ;
  47:   ;
  48:  LBL[1:airblast mode on] ;
  49:   ;
  50:L PR[91:Clr2] 200mm/sec CNT25    ;
  51:L PR[90:Clr1] 200mm/sec CNT25    ;
  52:   ;
  53:  !default, clr air blasts ;
  54:  RO[5:AIRJET 1A]=OFF ;
  55:  RO[7:AIRJET 2 B]=OFF ;
  56:  CALL MOVESAFE(R[12:FINE]) ;
/END
```

Program Listing of the APPLICATION Program APP_ASSEMBLY

```
/PROG  APP_ASSEMBLY
   1:  CALL MOVESAFE(R[11:CONTINUOUS]) ;
   2:   ;
   3:  !SET Position STEPS ;
   4:  R[17:STEP POS IDX]=R[4:PREG INDEX]    ;
   5:  CALL STEP(90,700,400,175) ;
   6:  CALL STEP(91,0,0,175) ;
   7:  CALL STEP(92,0,0,25) ;
   8:   ;
   9:J PR[90:Clr1] 50% CNT100    ;
  10:  COL GUARD ADJUST 100 ;
  11:L PR[91:Clr2] 250mm/sec CNT10    ;
  12:L PR[92:Clr3] 150mm/sec CNT1    ;
  13:   ;
  14:  !Move to POS ;
  15:  CALL ATTAUGHTPOS    ;
  16:  !Depart POS ;
  17:  CALL APPRO_DEPART(25,50) ;
  18:   ;
  19:L PR[92:Clr3] 150mm/sec CNT10    ;
  20:L PR[91:Clr2] 200mm/sec CNT10    ;
  21:  COL DETECT OFF ;
  22:L PR[90:Clr1] 250mm/sec CNT100    ;
  23:  CALL MOVESAFE(R[12:FINE]) ;
/END
```

Program Listing of the APPLICATION Program APP_PALUNLOAD

```
/PROG  APP_PALUNLOAD
   1:  CALL MOVESAFE(R[11:CONTINUOUS]) ;
   2:   ;
   3:  !Configure ;
   4:  PR[73:CONFIG PALLET]=PR[70:SAFE COPY]    ;
   5:  PR[73,4:CONFIG PALLET]=90    ;
   6:  IF (R[6:TOOLNUM]=1) THEN ;
   7:      PR[73,6:CONFIG PALLET]=(-180)    ;
   8:  ELSE ;
   9:      PR[73,6:CONFIG PALLET]=0    ;
  10:  ENDIF ;
  11:J PR[73:CONFIG PALLET] 100% CNT100    ;
  12:   ;
  13:  !Place at Input Conv ;
  14:  IF (R[6:TOOLNUM]=1) THEN ;
  15:      R[33:Pal Entry POSREG]=81    ;
  16:  ELSE ;
  17:      R[33:Pal Entry POSREG]=82    ;
  18:  ENDIF ;
  19:   ;
  20:  !Set Positions ;
  21:  R[17:STEP POS IDX]=R[33:Pal Entry POSREG]    ;
  22:  CALL STEP(90,(-300),(-200),450) ;
  23:  CALL STEP(91,0,0,400) ;
  24:  R[17:STEP POS IDX]=R[4:PREG INDEX]    ;
  25:  CALL STEP(92,0,3,25) ;
  26:  CALL STEP(93,0,10,250) ;
  27:  CALL STEP(94,0,(-100),0) ;
  28:  CALL STEP(95,0,(-100),50) ;
  29:  CALL STEP(96,0,(-100),400) ;
  30:   ;
  31:  !Move In ;
  32:L PR[90:Clr1] 250mm/sec CNT10    ;
  33:L PR[91:Clr2] 200mm/sec CNT10    ;
  34:  COL GUARD ADJUST 100 ;
  35:L PR[R[33]] 100mm/sec FINE    ;
  36:L PR[92:Clr3] 25mm/sec FINE    ;
  37:   ;
  38:  !Move to POS ;
  39:  CALL ATTAUGHTPOS    ;
  40:  !Depart POS ;
  41:  CALL APPRO_DEPART(25,25) ;
  42:  COL DETECT OFF ;
  43:   ;
  44:  !Move Out ;
  45:L PR[94:Clr5] 100mm/sec CNT20    ;
  46:L PR[95:Clr6] 200mm/sec CNT20    ;
  47:L PR[96:Clr7] 200mm/sec CNT20    ;
  48:  !Exit same as Entry ;
  49:L PR[90:Clr1] 200mm/sec CNT20    ;
  50:  CALL MOVESAFE(R[12:FINE]) ;
/END
```

Program Listing of the APPLICATION Program APP_CONVEYORS

```
/PROG  APP_CONVEYORS
   1:  CALL MOVESAFE(R[11:CONTINUOUS]) ;
   2:  !Configure ;
   3:  IF (R[6:TOOLNUM]=1) THEN ;
   4:      PR[72,6:CONFIG CONVEYOR]=90    ;
   5:  ENDIF ;
   6:  IF (R[6:TOOLNUM]=2) THEN ;
   7:      PR[72,6:CONFIG CONVEYOR]=270    ;
   8:  ENDIF ;
   9:J PR[72:CONFIG CONVEYOR] 25% CNT100    ;
  10:   ;
  11:  !SET Position STEPS ;
  12:  R[17:STEP POS IDX]=R[4:PREG INDEX]    ;
  13:  CALL STEP(90,100,0,250) ;
  14:  CALL STEP(91,0,0,25) ;
  15:   ;
  16:L PR[90:Clr1] 200mm/sec CNT50    ;
  17:L PR[91:Clr2] 100mm/sec CNT50    ;
  18:   ;
  19:  COL GUARD ADJUST 100 ;
  20:   ;
  21:  !Move to POS ;
  22:  CALL ATTAUGHTPOS    ;
  23:  !Depart POS ;
  24:  CALL APPRO_DEPART(75,50) ;
  25:   ;
  26:  COL DETECT OFF ;
  27:   ;
  28:L PR[90:Clr1] 250mm/sec CNT50    ;
  29:  CALL MOVESAFE(R[12:FINE]) ;
/END
```

Program Listing of the APPLICATION Program APP_MASTERS

```
/PROG  APP_MASTERS
   1:  CALL MOVESAFE(R[11:CONTINUOUS]) ;
   2:  !Pick or Place from Masters Rack ;
   3:   ;
   4:  !OPEN DOOR ;
   5:  CALL DOOR(0) ;
   6:   ;
   7:  IF (DO[8:RX_08_TPENBL]=OFF) THEN ;
   8:      R[18:CUBE INPUT NUM]=SR[1]    ;
   9:      IF (DI[R[18]]=OFF) THEN ;
  10:          MESSAGE[WATING TO ENTER CUBE] ;
  11:          WAIT DI[R[18]]=ON    ;
  12:      ENDIF ;
  13:  ENDIF ;
  14:   ;
  15:  !SET Position STEPS ;
  16:  R[17:STEP POS IDX]=R[4:PREG INDEX]    ;
  17:  R[19:ClearX]=100    ;
  18:  R[16:MOD]=R[3:PNS INDEX] MOD 2    ;
  19:  IF (R[16:MOD]=1) THEN ;
  20:      R[19:ClearX]=(-1)*R[19:ClearX]    ;
  21:  ENDIF ;
  22:   ;
  23:  CALL STEP(90,R[19:ClearX],700,130) ;
  24:  CALL STEP(91,R[19:ClearX],0,130) ;
  25:  CALL STEP(92,0,0,130) ;
  26:  CALL STEP(93,0,0,50) ;
  27:   ;
  28:  !MOVE IN ;
  29:L PR[90:Clr1] 150mm/sec CNT10    ;
  30:L PR[91:Clr2] 150mm/sec CNT10    ;
  31:L PR[92:Clr3] 100mm/sec FINE    ;
  32:L PR[93:Clr4] 100mm/sec CNT10    ;
  33:   ;
  34:  COL GUARD ADJUST 100 ;
  35:  !Appro POS ;
  36:  CALL APPRO_DEPART(25,50) ;
  37:  !Move to POS ;
  38:  CALL ATTAUGHTPOS    ;
  39:  !Depart POS ;
  40:  CALL APPRO_DEPART(25,50) ;
  41:   ;
  42:  !MOVE OUT ;
  43:L PR[92:Clr3] 100mm/sec CNT20    ;
  44:L PR[91:Clr2] 100mm/sec CNT20    ;
  45:L PR[90:Clr1] 100mm/sec CNT20    ;
  46:  COL DETECT OFF ;
  47:   ;
  48:  !SHUT DOOR ;
  49:  CALL DOOR(1) ;
  50:  CALL MOVESAFE(R[12:FINE]) ;
/END
```

PNSXXXX LISTINGS

```
/PROG  PNS0001
   1:  !Move to Perch ;
   2:  IF (DO[7:RX_07_ATPERCH]=OFF) THEN ;
   3:      CALL ASSIGNREGS(1,0,1,3,0) ;
   4:      CALL FLAGSSET    ;
   5:      !Force a CONFIG Move ;
   6:      F[6:MoveConfig First]=(ON) ;
   7:      CALL OPS    ;
   8:  ENDIF ;
/END
/PROG  PNS0002
   1:  !Move to Maintenance ;
   2:  IF (DO[56:RX_56_Safe_Maint]=OFF) THEN
;
   3:      CALL ASSIGNREGS(2,0,2,3,0) ;
   4:      CALL FLAGSSET    ;
   5:      !Force a CONFIG Move ;
   6:      F[6:MoveConfig First]=(ON) ;
   7:      CALL OPS    ;
   8:  ENDIF ;
/END
/PROG  PNS0010
   1:  !MOVETO SAFE PAL LOAD ;
   2:  CALL ASSIGNREGS(10,0,10,3,0) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0011
   1:  !Pick from Pallet Load T1 ;
   2:  CALL ASSIGNREGS(11,11,10,1,1) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0014
   1:  !Pick Pallet Load T2 ;
   2:  CALL ASSIGNREGS(14,12,10,2,1) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0020
   1:  !MOVETO SAFE LASER MARKER ;
   2:  CALL ASSIGNREGS(20,0,14,3,0) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0021
   1:  !Pick from Laser Marker T1 ;
   2:  CALL ASSIGNREGS(21,15,14,1,1) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0022
   1:  !Place at Laser T2 ;
   2:  CALL ASSIGNREGS(22,16,14,2,2) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
```

```
/PROG  PNS0023
   1:  !Place Laser Marker T1 ;
   2:  CALL ASSIGNREGS(23,15,14,1,2) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0024
   1:  !Pick from Laser Marker T2 ;
   2:  CALL ASSIGNREGS(24,16,14,2,1) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0030
   1:  !MOVETO SAFE LEAK TEST LH ;
   2:  CALL ASSIGNREGS(30,0,18,3,0) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0031
   1:  !Pick from Leaktest LH T1 ;
   2:  CALL ASSIGNREGS(31,19,18,1,1) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0032
   1:  !Place at Leaktest LH T2 ;
   2:  CALL ASSIGNREGS(32,20,18,2,2) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0033
   1:  !Place at Leaktest LH T1  ;
   2:  CALL ASSIGNREGS(33,19,18,1,2) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0034
   1:  !Pick Laser Marker T2 ;
   2:  CALL ASSIGNREGS(34,20,18,2,1) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0040
   1:  !MOVETO SAFE LEAK TEST RH ;
   2:  CALL ASSIGNREGS(40,0,22,3,0) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0040
   1:  !MOVETO SAFE LEAK TEST RH ;
   2:  CALL ASSIGNREGS(40,0,22,3,0) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
```

PNSXXXX LISTINGS

```
/PROG  PNS0041
   1:  !Pick from Leaktest RH T1 ;
   2:  CALL ASSIGNREGS(41,23,22,1,1) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0042
   1:  !Place at Leaktest RH T2 ;
   2:  CALL ASSIGNREGS(42,24,22,2,2) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0043
   1:  !Place at Leaktest RH T1 ;
   2:  CALL ASSIGNREGS(43,23,22,1,2) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0044
   1:  !Pick from Leaktest RH T1 ;
   2:  CALL ASSIGNREGS(44,24,22,2,1) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0050
   1:  !MOVETO SAFE ASSEMBLY ;
   2:  CALL ASSIGNREGS(50,0,26,3,0) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0051
   1:  !Pick from Assembly T1 ;
   2:  CALL ASSIGNREGS(51,27,26,1,1) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0052
   1:  !Place at Assembly T2 ;
   2:  CALL ASSIGNREGS(52,28,26,2,2) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0053
   1:  !Place at Assembly T2 ;
   2:  CALL ASSIGNREGS(53,27,26,1,2) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0054
   1:  !Pick from Assembly T2 ;
   2:  CALL ASSIGNREGS(54,28,26,2,1) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
```

```
/PROG  PNS0060
   1:  !MOVETO PALLET LOAD ;
   2:  CALL ASSIGNREGS(60,0,30,3,0) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0062
   1:  !Place at Pallet unload T1 ;
   2:  CALL ASSIGNREGS(62,32,30,2,2) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0063
   1:  !Place at Unload Pallet T1 ;
   2:  CALL ASSIGNREGS(63,31,30,1,2) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0070
   1:  !MOVETO SAFE CONVEYORS ;
   2:  CALL ASSIGNREGS(70,0,34,3,0) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0072
   1:  !Place at Conveyor Impreg T2 ;
   2:  CALL ASSIGNREGS(72,36,34,2,2) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0073
   1:  !Place at Conveyor Impreg T1 ;
   2:  CALL ASSIGNREGS(73,35,34,1,2) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0074
   1:  !Place at Conv Fail T2 ;
   2:  CALL ASSIGNREGS(74,38,34,2,2) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0075
   1:  !Place at Conv Fail T1 ;
   2:  CALL ASSIGNREGS(75,37,34,1,2) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0100
   1:  !MOVETO SAFE MASTER RACKS ;
   2:  CALL ASSIGNREGS(100,0,40,3,0) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
```

PNSXXXX LISTINGS

```
/PROG  PNS0111
   1:  !GET MC1 T1 ;
   2:  CALL ASSIGNREGS(111,41,40,1,1) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0112
   1:  !GET MC2 T1 ;
   2:  CALL ASSIGNREGS(112,42,40,1,1) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0113
   1:  !GET MC5 T1 ;
   2:  CALL ASSIGNREGS(113,43,40,1,1) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0114
   1:  !GET MC6 T1 ;
   2:  CALL ASSIGNREGS(114,44,40,1,1) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0115
   1:  !GET MC3 T1 ;
   2:  CALL ASSIGNREGS(115,45,40,1,1) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0116
   1:  !GET MC4 T1 ;
   2:  CALL ASSIGNREGS(116,46,40,1,1) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0117
   1:  !GET EX1 T1 ;
   2:  CALL ASSIGNREGS(117,47,40,1,1) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0118
   1:  !GET EX2 T1 ;
   2:  CALL ASSIGNREGS(118,48,40,1,1) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0121
   1:  !PUT MC1 T2 ;
   2:  CALL ASSIGNREGS(121,49,40,2,2) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
```

```
/PROG  PNS0122
   1:  !PUT MC2 T2 ;
   2:  CALL ASSIGNREGS(122,50,40,2,2) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0123
   1:  !PUT MC5 T2 ;
   2:  CALL ASSIGNREGS(123,51,40,2,2) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0124
   1:  !PUT MC6 T2 ;
   2:  CALL ASSIGNREGS(124,52,40,2,2) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0125
   1:  !PUT MC3 T2 ;
   2:  CALL ASSIGNREGS(125,53,40,2,2) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0126
   1:  !PUT MC4 T2 ;
   2:  CALL ASSIGNREGS(126,54,40,2,2) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0127
   1:  !PUT EX1 T2 ;
   2:  CALL ASSIGNREGS(127,55,40,2,2) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0128
   1:  ! PUT EX2 T2 ;
   2:  CALL ASSIGNREGS(128,56,40,2,2) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0131
   1:  !PUT MC1 T1 ;
   2:  CALL ASSIGNREGS(131,41,40,1,2) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0132
   1:  !PUT MC2 T1 ;
   2:  CALL ASSIGNREGS(132,42,40,1,2) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
```

PNSXXXX LISTINGS

```
/PROG  PNS0132
   1:  !PUT MC2 T1 ;
   2:  CALL ASSIGNREGS(132,42,40,1,2) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0133
   1:  !PUT MC5 T1 ;
   2:  CALL ASSIGNREGS(133,43,40,1,2) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0134
   1:  !PUT MC6 T1 ;
   2:  CALL ASSIGNREGS(134,44,40,1,2) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0135
   1:  !PUT MC3 T1 ;
   2:  CALL ASSIGNREGS(135,45,40,1,2) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0136
   1:  !PUT MC4 T1 ;
   2:  CALL ASSIGNREGS(136,46,40,1,2) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0137
   1:  !PUT EX1 T1 ;
   2:  CALL ASSIGNREGS(137,47,40,1,2) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0138
   1:  !PUT EX2 T1 ;
   2:  CALL ASSIGNREGS(138,48,40,1,2) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0141
   1:  !GET MC1 T2 ;
   2:  CALL ASSIGNREGS(141,49,40,2,1) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0142
   1:  !GET MC2 T2 ;
   2:  CALL ASSIGNREGS(142,50,40,2,1) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
```

```
/PROG  PNS0143
   1:  !GET MC5 T2 ;
   2:  CALL ASSIGNREGS(143,51,40,2,1) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0144
   1:  !GET MC6 T2 ;
   2:  CALL ASSIGNREGS(144,52,40,2,1) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0145
   1:  !GET MC3 T2 ;
   2:  CALL ASSIGNREGS(145,53,40,2,1) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0146
   1:  !GET MC4 T2 ;
   2:  CALL ASSIGNREGS(146,54,40,2,1) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0147
   1:  !GET EX1 T2 ;
   2:  CALL ASSIGNREGS(147,55,40,2,1) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
/PROG  PNS0148
   1:  !GET EX2 T2 ;
   2:  CALL ASSIGNREGS(148,56,40,2,1) ;
   3:  CALL FLAGSSET    ;
   4:  CALL OPS    ;
/END
```

ALL the PNSxxxx Programs are written in the same format.

CALL ASSIGNREGS with parameters.

Argument 1 PNS0xxx Number

Argument 2 Safe Position Register Number

Argument 3 App Position Register Number

Argument 4 TCP Number to use.

Argument 5 Gripper Action at Position.

(1 SHUT) (2 OPEN)

CALL FLAGSET Internal IO

CALL OPS (short for OperrationS)

```
/PROG  ASSIGNREGS
   1:  R[3:PNS INDEX]=AR[1]    ;
   2:  R[4:PREG INDEX]=AR[2]    ;
   3:  R[5:SAFE INDEX]=AR[3]    ;
   4:  R[6:TOOLNUM]=AR[4]    ;
   5:  R[14:PICK/PLACE]=AR[5]    ;
   6:  END ;
   7:   ;
   8:  !Passed Parameter ;
   9:  !AR[1] PNS Number ;
  10:  !AR[2] Position Register ;
  11:  !AR[3] Safe Position Reg ;
  12:  !AR[4] Tool A or B ;
  13:  !AR[5] (1=PICK) (2=PLACE) ;
/END
```

The ASSIGNREGS Program sets REGISTER's from the passing ARGUMENTS to be used in the Appication Programs.

```
/PROG  FLAGSSET
   1:  !ASSIGN USER FLAGS ;
   2:   ;
   3:  !Reset Error Flag ;
   4:  R[10:ERRORCODE]=0    ;
   5:   ;
   6:  !Empty Tool1 ;
   7:  F[1:ToolA_is_Empty]=(RI[5:Mhev 1 Prx ]=OFF AND RI[6:Phev 1 prx ]=OFF) ;
   8:   ;
   9:  !Empty Tool2 ;
  10:  F[2:ToolB_is_Empty]=(RI[7:Mhev 2 prx]=OFF AND RI[8:Phev 2 prx]=OFF) ;
  11:   ;
  12:  !Robot @ Safe Pos ;
  13:  F[3:RobAtSafePosn]=(DO[7:RX_07_ATPERCH]=ON OR
                          DO[21:RX_21_Safe10]=ON OR
                          DO[22:RX_22_Safe20]=ON OR
                          DO[23:RX_23_Safe30]=ON OR
                          DO[24:RX_24_Safe40]=ON OR
                          DO[25:RX_25_Safe50]=ON OR
                          DO[26:RX_26_Safe60]=ON OR
                          DO[27:RX_27_Safe70]=ON OR
                          DO[28:RX_28_Safe100]=ON OR
                          DO[56:RX_56_Safe_Maint]=ON) ;
  14:   ;
  15:  !Pick from Rack ;
  16:  F[4:Master Rack LH]=(R[3:PNS INDEX]>=111 AND
                           R[3:PNS INDEX]<=114 OR
                           R[3:PNS INDEX]>=121 AND
                           R[3:PNS INDEX]<=124 OR
                           R[3:PNS INDEX]>=131 AND
                           R[3:PNS INDEX]<=134 OR
                           R[3:PNS INDEX]>=141 AND
                           R[3:PNS INDEX]<=144) ;
  17:   ;
  18:  !Place @ Rack ;
  19:  F[5:Master Rack RH]=(R[3:PNS INDEX]>=115 AND
                           R[3:PNS INDEX]<=118 OR
                           R[3:PNS INDEX]>=125 AND
                           R[3:PNS INDEX]<=128 OR
                           R[3:PNS INDEX]>=135 AND
                           R[3:PNS INDEX]<=138 OR
                           R[3:PNS INDEX]>=145 AND
```

```
                          R[3:PNS INDEX]<=148) ;
  20:   ;
  21:   !Robot @ Perch or Maint  ;
  22:   F[6:MoveConfig First]=(DO[7:RX_07_ATPERCH]=ON OR DO[56:RX_56_Safe_Maint]=ON) ;
  23:   ;
  24:   !OP Flags from PNSxxxx ;
  25:   F[10:OP Pallet Load   ]=(R[3:PNS INDEX]=11 OR R[3:PNS INDEX]=14) ;
  26:   ;
  27:   F[11:OP Laser Marker ]=(R[3:PNS INDEX]>=21 AND R[3:PNS INDEX]<=24) ;
  28:   ;
  29:   F[12:OP Leak Test LH]=(R[3:PNS INDEX]>=31 AND R[3:PNS INDEX]<=34) ;
  30:   ;
  31:   F[13:OP Leak Test RH]=(R[3:PNS INDEX]>=41 AND R[3:PNS INDEX]<=44) ;
  32:   ;
  33:   F[14:OP Assembly]=(R[3:PNS INDEX]>=51 AND R[3:PNS INDEX]<=54) ;
  34:   ;
  35:   F[15:OP Pallet Unload]=(R[3:PNS INDEX]=62 OR R[3:PNS INDEX]=63) ;
  36:   ;
  37:   F[16:OP Conv FAIL]=(R[3:PNS INDEX]=72 OR R[3:PNS INDEX]=73) ;
  38:   ;
  39:   F[17:OP Conv IMPREG]=(R[3:PNS INDEX]=74 OR R[3:PNS INDEX]=75) ;
  40:   ;
  41:   F[18:OP Master Racks]=(F[4:Master Rack LH]=ON OR F[5:Master Rack RH]=ON) ;
  42:   ;
  43:   F[19:OP MOVE SAFE]=(R[3:PNS INDEX]=1 OR
                            R[3:PNS INDEX]=2 OR
                           R[3:PNS INDEX]=10 OR
                           R[3:PNS INDEX]=20 OR
                           R[3:PNS INDEX]=30 OR
                           R[3:PNS INDEX]=40 OR
                           R[3:PNS INDEX]=50 OR
                           R[3:PNS INDEX]=60 OR
                           R[3:PNS INDEX]=70 OR
                           R[3:PNS INDEX]=100) ;
/END
```

The FLAGSET program sets an Internal Flag setting to ether a ON or OFF state using the PNS register value is within a range of numbers. The REGISTER[3] has been previously set by the ASSIGNREGS program. There are flags on the state of the robot at a SAFE Position, The State of the Proxcimity Sensors on the Gripper.

```
/PROG  OPS
   1:  !Set TCP ;
   2:  UTOOL_NUM=R[6:TOOLNUM] ;
   3:  UFRAME_NUM=0 ;
   4:   ;
   5:  !Check @ SAFE POS ;
   6:  IF (F[3:RobAtSafePosn]=OFF),JMP LBL[99] ;
   7:   ;
   8:  !Get Cube IO Numbers ;
   9:  CALL CUBEIO    ;
  10:   ;
  11:  !APPLICATIONS PROGS ;
  12:  IF (F[10:OP Pallet Load   ]=ON),CALL APP_PALLOAD ;
  13:  IF (F[11:OP Laser Marker ]=ON),CALL APP_LASERMARKER ;
  14:  IF (F[12:OP Leak Test LH]=ON),CALL APP_LEAKTEST ;
  15:  IF (F[13:OP Leak Test RH]=ON),CALL APP_LEAKTEST ;
  16:  IF (F[14:OP Assembly]=ON),CALL APP_ASSEMBLY ;
  17:  IF (F[15:OP Pallet Unload]=ON),CALL APP_PALUNLOAD ;
  18:  IF (F[16:OP Conv FAIL]=ON),CALL APP_CONVEYORS ;
  19:  IF (F[17:OP Conv IMPREG]=ON),CALL APP_CONVEYORS ;
  20:  IF (F[18:OP Master Racks]=ON),CALL APP_MASTERS ;
  21:  IF (F[19:OP MOVE SAFE]=ON) THEN ;
  22:      CALL MOVESAFE(R[12:FINE]) ;
  23:  ENDIF ;
  24:   ;
  25:  END ;
  26:   ;
  27:  LBL[99:ROBOT NOT FOUND ] ;
  28:  CALL APPFAULTS(1) ;
/END
```

The OPS (Operations) Program uses the FLAGS that have been set in the FLAGSET program to ask if a FLAG as been set to ON then to call the requied APPICATION Program.

F[10] APP_PALLOAD
F[11] APP_LASERMARKER
F[12] APP_LEAKTEST (S1)
F[13] APP_LEAKTEST (S2)
F[14] APP_ASSEMBLY
F[15] APP_PALUNLOAD
F[16] APP_CONVEYORS (FAIL)
F[17] APP_CONVEYORS (IMPREG)
F[18] APP_MASTERS
F[19] MOVESAFE (PERCH) OR (MAINENANCE) OR (MANUAL SAFE'S)

```
/PROG  STEP
   1:  !Null Tcp Offset ;
   2:  PR[5:TCPOFFSET]=PR[4:NULLOFFSET]    ;
   3:  !Set XYZ from Arguments ;
   4:  PR[5,1:TCPOFFSET]=AR[2]    ;
   5:  PR[5,2:TCPOFFSET]=AR[3]    ;
   6:  PR[5,3:TCPOFFSET]=AR[4]    ;
   7:   ;
   8:  !STEP = POSREG+OFFSET ;
   9:  PR[AR[1]]=PR[R[4]]+PR[5:TCPOFFSET]    ;
  10:  END ;
  11:  !AR[1] STEP NUMBER ;
  12:  !AR[2] X DISTANCE ;
  13:  !AR[3] Y DISTANCE ;
  14:  !AR[4] Z DISTANCE ;
/END
```

In the APPICATION Programs there is a program called STEP to set a working position register of a XYZ displacement or (offset) which normally from the target position register.

The Working Registers 90-96 have been used, they are updated on each application program that has been call upon. When used they then move the robot through it's motion to the target position and back out clear of the equipment.

There can not be used to go directly to the targent position as it is not guarantee that the target position is in the same XYZ plane as the robot due floor or equipment levelling.

```
/PROG  APPRO_DEPART
   1:  !Approch or Depart Tool -Z ;
   2:  PR[5:TCPOFFSET]=PR[4:NULLOFFSET]    ;
   3:  !Set DIST & SPEED ;
   4:  !AR[1] DIST AR[2] SPEED ;
   5:  PR[5,3:TCPOFFSET]=(-1)*AR[1]    ;
   6:  R[2]=AR[2]    ;
   7:   ;
   8:  !Appro/Depart Move ;
   9:  IF (AR[1]>25) THEN ;
  10:    L PR[R[4]] R[2]mm/sec CNT20 Tool_Offset,PR[5:TCPOFFSET]    ;
  11:  ELSE ;
  12:    L PR[R[4]] R[2]mm/sec FINE Tool_Offset,PR[5:TCPOFFSET]    ;
  13:  ENDIF ;
/END
```

The Program APPRO_DEPART will position the robot inline with the Target position using the it's TOOL with the guide pins, distances and speed argments are used for this function.

```
/PROG  ATTAUGHTPOS
   1:  !Skip if output comes on ;
   2:  SKIP CONDITION DO[6:RX_06_FAULT]=ON    ;
   3:  ![---- THE TAUGHT POS ---- ;
   4:L PR[R[4]] 25mm/sec FINE Skip,LBL[1]    ;
   5:  !---- THE TAUGHT POS ----] ;
   6:   ;
   7:  !FAULT ;
   8:  COL DETECT OFF ;
   9:  ABORT ;
  10:   ;
  11:  LBL[1:At Pos No Fault] ;
  12:  !Func OPEN/SHUT Gripper ;
  13:  CALL GRIPFUNCTION    ;
/END
```

The Program ATTAUGHTPOS is the program that finally goes to the target position. In the case of a re-teach it the only require needed as REGISTER[4] has been set with the correct POSITION REGISTER Number.

```
/PROG  GRIPFUNCTION
   1:  !At POS action the gripper ;
   2:   ;
   3:  !SHUT GRIPPER ON A PICK ;
   4:  IF (R[14:PICK/PLACE]=1) THEN ;
   5:      CALL AIRVALVES(1) ;
   6:      CALL GRIPPERCHK(1) ;
   7:  ENDIF ;
   8:   ;
   9:  !OPEN GRIPPER ON A PLACE ;
  10:  IF (R[14:PICK/PLACE]=2) THEN ;
  11:      CALL AIRVALVES(0) ;
  12:      CALL GRIPPERCHK(0) ;
  13:  ENDIF ;
/END
```

If the Motion to the Target Position as been successful then PROGRAM GRIPFUNCTION is called and then to energize the robot digital output signal to the gripper using Register information of ether OPEN or SHUT the Gripper via the AIRVALVES Program.

```
/PROG  AIRVALVES
   1:  R[7:VALVESTATE]=AR[1]    ;
   2:   ;
   3:  IF (R[7:VALVESTATE]=0) THEN ;
   4:      IF (R[6:TOOLNUM]=1) THEN ;
   5:          RO[1:Release 1A]=ON ;
   6:          RO[2:Grip 1A]=OFF ;
   7:      ELSE ;
   8:          RO[3:Release B2]=ON ;
   9:          RO[4:Grip B2]=OFF ;
  10:      ENDIF ;
  11:  ELSE ;
  12:      IF (R[6:TOOLNUM]=1) THEN ;
  13:          RO[2:Grip 1A]=ON ;
  14:          RO[1:Release 1A]=OFF ;
  15:      ELSE ;
  16:          RO[4:Grip B2]=ON ;
  17:          RO[3:Release B2]=OFF ;
  18:      ENDIF ;
  19:  ENDIF ;
/END
```

```
/PROG  GRIPPERCHK
   1:  R[7:VALVESTATE]=AR[1]    ;
   2:   ;
   3:  IF (R[7:VALVESTATE]=0) THEN ;
   4:      IF (R[6:TOOLNUM]=1) THEN ;
   5:          WAIT RI[1:TOOLA is OPEN]=ON    ;
   6:      ELSE ;
   7:          WAIT RI[3:TOOLB is OPEN]=ON    ;
   8:      ENDIF ;
   9:  ELSE ;
  10:      IF (R[6:TOOLNUM]=1) THEN ;
  11:          WAIT RI[2:TOOLA is SHUT ]=ON    ;
  12:      ELSE ;
  13:          WAIT RI[4:TOOLB is SHUT]=ON    ;
  14:      ENDIF ;
  15:  ENDIF ;
  16:   ;
  17:  !Ensure end strokes ;
  18:  WAIT    .50(sec) ;
/END
```

The GRIPPERCHK program then checks and wait for the STATE of the gripper's proxcimity sensors.

(NOTE: Timeout to be implemited and the error to sent to the cell controller, then the best action following the fault TBD)

```
/PROG  GRIPRDY
   1:  !OPEN & CHECK VALVES FOR PICKING ;
   2:  IF (R[14:PICK/PLACE]<>1) THEN ;
   3:      END ;
   4:  ENDIF ;
   5:   ;
   6:  IF (R[6:TOOLNUM]=1 AND F[1:ToolA_is_Empty]=OFF) THEN ;
   7:      CALL APPFAULTS(2) ;
   8:      END ;
   9:  ENDIF ;
  10:   ;
  11:  IF (R[6:TOOLNUM]=2 AND F[2:ToolB_is_Empty]=OFF) THEN ;
  12:      CALL APPFAULTS(3) ;
  13:      END ;
  14:  ENDIF ;
  15:   ;
  16:  CALL AIRVALVES(0) ;
  17:  CALL GRIPPERCHK(0) ;
/END
```

A general program GRIPRDY called when a GET PNSxxxx has been called to check for a PART Present already in the appointed gripper, normany used when the robot is being used by a operator.

Also to open and check the gripper before moving to the target position.

```
/PROG  MOVESAFE
   1:  !Check if at Perch/Mant ;
   2:  IF (F[6:MoveConfig First]=ON) THEN ;
   3:    J PR[3:CONFIG] 50% FINE    ;
   4:      F[6:MoveConfig First]=(OFF) ;
   5:  ENDIF ;
   6:   ;
   7:  !Motion Types to Safe Pos ;
   8:  R[13:MOTION TYPE]=AR[1]    ;
   9:   ;
  10:  IF (R[13:MOTION TYPE]=R[11:CONTINUOUS]) THEN ;
  11:      PR[70:SAFE COPY]=PR[R[5]]    ;
  12:      IF (R[6:TOOLNUM]=1) THEN ;
  13:          IF (F[10:OP Pallet Load   ]=OFF AND F[15:OP Pallet Unload]=OFF) THEN ;
  14:              PR[70,6:SAFE COPY]=0    ;
  15:          ENDIF ;
  16:      ENDIF ;
  17:      IF (R[6:TOOLNUM]=2) THEN ;
  18:          IF (F[10:OP Pallet Load   ]=OFF AND F[15:OP Pallet Unload]=OFF) THEN ;
  19:              PR[70,6:SAFE COPY]=180    ;
  20:          ENDIF ;
  21:      ENDIF ;
  22:    J PR[70:SAFE COPY] 50% CNT100    ;
  23:  ENDIF ;
  24:  IF (R[13:MOTION TYPE]=R[12:FINE]) THEN ;
  25:    J PR[R[5]] 50% FINE    ;
  26:  ENDIF ;
/END
```

The Program MOVESAFE is called on the START and END of an APPICATION program aswll as as stand alone for the PERCH, MAINENANCE and manual SAFE moves.

An Argument passed for a CNT (Continues Move) or FINE (reach the destination position)

CNT used to flow through goring to start and FINE after the event to ensure the robot is in position and the digital out is sent to the cell controlleer.

While in the CNT mode a rotation value is give to prepare the Joint 6 to be verticall ready for TOOL 1 or TOOL 2 form the normal stacncde of the gripper being horzontal.

```
/PROG  APPFAULTS
   1:  !FAULT INFORM PLC/OPERATOR ;
   2:  R[10:ERRORCODE]=AR[1]    ;
   3:  IF (DO[8:RX_08_TPENBL]=ON) THEN ;
   4:      !MODE MANUAL ;
   5:      JMP LBL[R[10]] ;
   6:  ELSE ;
   7:      !MODE AUTO ;
   8:      GO[1:Error Code]=AR[1] ;
   9:      WAIT DI[41:RY_41_ClrGripERR]=ON    ;
  10:      GO[1:Error Code]=0 ;
  11:      WAIT DI[41:RY_41_ClrGripERR]=OFF    ;
  12:  ENDIF ;
  13:  !goto the program END ;
  14:  END ;
  15:  !Manual Mode Messages ;
  16:  LBL[1] ;
  17:  MESSAGE[NOT Found at any SAFE's] ;
  18:  JMP LBL[99] ;
  19:  LBL[2] ;
  20:  MESSAGE[ToolA NOT Empty to PICK] ;
  21:  JMP LBL[99] ;
  22:  LBL[3] ;
  23:  MESSAGE[ToolB NOT Empty to PICK] ;
  24:  JMP LBL[99] ;
  25:  LBL[4] ;
  26:  MESSAGE[ToolA Failed to OPEN] ;
  27:  JMP LBL[99] ;
  28:  LBL[5] ;
  29:  MESSAGE[ToolB Failed to OPEN] ;
  30:  JMP LBL[99] ;
  31:  LBL[6] ;
  32:  MESSAGE[ToolA Failed to SHUT] ;
  33:  JMP LBL[99] ;
  34:  LBL[7] ;
  35:  MESSAGE[ToolB Failed to SHUT] ;
  36:  JMP LBL[99] ;
  37:  LBL[8] ;
  38:  MESSAGE[T1/2 NOT EMPTY RACKPICK] ;
  39:  JMP LBL[99] ;
  40:  LBL[9] ;
  41:  MESSAGE[T2Full for T1RACK Place] ;
  42:  JMP LBL[99] ;
  43:  LBL[10] ;
  44:  MESSAGE[T1Full for T2RACK Place] ;
  45:  JMP LBL[99] ;
  46:  LBL[11] ;
  47:  MESSAGE[NOT Clear of SPACE CUBE] ;
  48:  LBL[99] ;
/END
```

The Program APPFAULT send a Code Number to the Cell Controller or if manual mode a message is displayed on the Teach Pendant.

```
/PROG  DOOR
   1:  !Passed AR[1]  0 Open 1 Shut ;
   2:  R[9:OPEN / SHUT]=AR[1]    ;
   3:   ;
   4:  !T1 set & move to config ;
   5:  IF (R[6:TOOLNUM]=1) THEN ;
   6:      PR[59,6:DOOR CONFIG]=0    ;
   7:      IF (R[9:OPEN / SHUT]=0) THEN ;
   8:        J PR[59:DOOR CONFIG] 50% CNT50    ;
   9:      ELSE ;
  10:        L PR[59:DOOR CONFIG] 150mm/sec CNT20    ;
  11:      ENDIF ;
  12:  ENDIF ;
  13:   ;
  14:  !T2 set & move to config ;
  15:  IF (R[6:TOOLNUM]=2) THEN ;
  16:      PR[59,6:DOOR CONFIG]=180    ;
  17:      IF (R[9:OPEN / SHUT]=0) THEN ;
  18:        J PR[59:DOOR CONFIG] 50% CNT50    ;
  19:      ELSE ;
  20:        L PR[59:DOOR CONFIG] 150mm/sec CNT20    ;
  21:      ENDIF ;
  22:  ENDIF ;
  23:   ;
  24:  IF (DO[8:RX_08_TPENBL]=ON) THEN ;
  25:      //END ;
  26:  ENDIF ;
  27:   ;
  28:  !set door positions ;
  29:  IF (F[4:Master Rack LH]=ON),CALL DOOR_LH ;
  30:  IF (F[5:Master Rack RH]=ON),CALL DOOR_RH ;
  31:   ;
  32:  !door motions ;
  33:L PR[60:DOOR SAFE] 150mm/sec CNT10    ;
  34:L PR[61:DOOR START] 150mm/sec CNT10    ;
  35:   ;
  36:  //!Stop at Handle ;
  37:  //IF (F[4:Master Rack LH]=ON) THEN ;
  38:      //IF (R[9:OPEN / SHUT]=0) THEN ;
  39:          //L PR[62:DOOR LH P1] 150mm/sec FINE    ;
  40:      //ENDIF ;
  41:      //IF (R[9:OPEN / SHUT]=1) THEN ;
  42:          //L PR[63:DOOR LH P2] 150mm/sec FINE    ;
  43:      //ENDIF ;
  44:  //ENDIF ;
  45:   ;
  46:  //IF (F[5:Master Rack RH]=ON) THEN ;
  47:      //IF (R[9:OPEN / SHUT]=0) THEN ;
  48:          //L PR[64:DOOR RH P1] 150mm/sec FINE    ;
  49:      //ENDIF ;
  50:      //IF (R[9:OPEN / SHUT]=1) THEN ;
  51:          //L PR[65:DOOR RH P2] 150mm/sec FINE    ;
  52:      //ENDIF ;
  53:  //ENDIF ;
  54:   ;
  55:L PR[66:DOOR END] 100mm/sec FINE    ;
  56:L PR[67:DOOR FINISH] 150mm/sec CNT20    ;
  57:   ;
  58:L PR[59:DOOR CONFIG] 150mm/sec CNT20    ;
  59:  !move safe after door shut ;
  60:  IF (R[9:OPEN / SHUT]=1) THEN ;
  61:    J PR[70:SAFE COPY] 25% CNT20    ;
```

```
  62:  ENDIF ;
/END
/PROG  DOOR_LH
   1:  !OPEN DOOR ;
   2:  IF (R[9:OPEN / SHUT]=0) THEN ;
   3:      PR[60:DOOR SAFE]=PR[62:DOOR LH P1]    ;
   4:      PR[61:DOOR START]=PR[62:DOOR LH P1]    ;
   5:      PR[66:DOOR END]=PR[63:DOOR LH P2]    ;
   6:      PR[67:DOOR FINISH]=PR[63:DOOR LH P2]    ;
   7:       ;
   8:      PR[60,1:DOOR SAFE]=PR[60,1:DOOR SAFE]+100    ;
   9:      PR[60,2:DOOR SAFE]=PR[60,2:DOOR SAFE]+100    ;
  10:      PR[61,1:DOOR START]=PR[61,1:DOOR START]+100    ;
  11:      PR[67,1:DOOR FINISH]=PR[67,1:DOOR FINISH]+150    ;
  12:      PR[67,2:DOOR FINISH]=PR[67,2:DOOR FINISH]+75    ;
  13:  ENDIF ;
  14:   ;
  15:  !SHUT DOOR ;
  16:  IF (R[9:OPEN / SHUT]=1) THEN ;
  17:      PR[60:DOOR SAFE]=PR[63:DOOR LH P2]    ;
  18:      PR[61:DOOR START]=PR[63:DOOR LH P2]    ;
  19:      PR[66:DOOR END]=PR[62:DOOR LH P1]    ;
  20:      PR[67:DOOR FINISH]=PR[62:DOOR LH P1]    ;
  21:       ;
  22:      PR[60,1:DOOR SAFE]=PR[60,1:DOOR SAFE]-100    ;
  23:      PR[60,2:DOOR SAFE]=PR[60,2:DOOR SAFE]+100    ;
  24:      PR[61,1:DOOR START]=PR[61,1:DOOR START]-100    ;
  25:      PR[66,1:DOOR END]=PR[66,1:DOOR END]-45    ;
  26:      PR[67,1:DOOR FINISH]=PR[67,1:DOOR FINISH]-150    ;
  27:      PR[67,2:DOOR FINISH]=PR[67,2:DOOR FINISH]+75    ;
  28:  ENDIF ;
/END
/PROG  DOOR_RH
   1:  !DOOR OPEN ;
   2:  IF (R[9:OPEN / SHUT]=0) THEN ;
   3:      PR[60:DOOR SAFE]=PR[64:DOOR RH P1]    ;
   4:      PR[61:DOOR START]=PR[64:DOOR RH P1]    ;
   5:      PR[66:DOOR END]=PR[65:DOOR RH P2]    ;
   6:      PR[67:DOOR FINISH]=PR[65:DOOR RH P2]    ;
   7:       ;
   8:      !DOOR SHUT ;
   9:      PR[60,1:DOOR SAFE]=PR[60,1:DOOR SAFE]-100    ;
  10:      PR[60,2:DOOR SAFE]=PR[60,2:DOOR SAFE]+75    ;
  11:      PR[61,1:DOOR START]=PR[61,1:DOOR START]-100    ;
  12:      PR[67,1:DOOR FINISH]=PR[67,1:DOOR FINISH]-100    ;
  13:      PR[67,2:DOOR FINISH]=PR[67,2:DOOR FINISH]+75    ;
  14:  ENDIF ;
  15:   ;
  16:   ;
  17:  IF (R[9:OPEN / SHUT]=1) THEN ;
  18:      PR[60:DOOR SAFE]=PR[65:DOOR RH P2]    ;
  19:      PR[61:DOOR START]=PR[65:DOOR RH P2]    ;
  20:      PR[66:DOOR END]=PR[64:DOOR RH P1]    ;
  21:      PR[67:DOOR FINISH]=PR[64:DOOR RH P1]    ;
  22:       ;
  23:      PR[60,1:DOOR SAFE]=PR[60,1:DOOR SAFE]+100    ;
  24:      PR[60,2:DOOR SAFE]=PR[60,2:DOOR SAFE]+75    ;
  25:      PR[61,1:DOOR START]=PR[61,1:DOOR START]+100    ;
  26:      PR[66,1:DOOR END]=PR[66,1:DOOR END]+45    ;
  27:      PR[67,1:DOOR FINISH]=PR[67,1:DOOR FINISH]+100    ;
  28:      PR[67,2:DOOR FINISH]=PR[67,2:DOOR FINISH]+75    ;
  29:  ENDIF ;
```

```
/END
```

The DOOR opening and Closing Program for the Master Rack Enclosure.

```
/PROG  CUBEIO
   1:  !REGISTER CUBE'S IO and set TOOL ;
   2:   ;
   3:  IF (F[19:OP MOVE SAFE]=ON) THEN ;
   4:      END ;
   5:  ENDIF ;
   6:   ;
   7:  SR[1]=0    ;
   8:  R[18:CUBE INPUT NUM]=SR[1]    ;
   9:   ;
  10:  IF (F[10:OP Pallet Load   ]=ON),JMP LBL[1] ;
  11:  IF (F[11:OP Laser Marker ]=ON),JMP LBL[2] ;
  12:  IF (F[12:OP Leak Test LH]=ON),JMP LBL[3] ;
  13:  IF (F[13:OP Leak Test RH]=ON),JMP LBL[4] ;
  14:  IF (F[14:OP Assembly]=ON),JMP LBL[5] ;
  15:  IF (F[15:OP Pallet Unload]=ON),JMP LBL[6] ;
  16:  IF (F[16:OP Conv FAIL]=ON),JMP LBL[7] ;
  17:  IF (F[17:OP Conv IMPREG]=ON),JMP LBL[8] ;
  18:  IF (F[4:Master Rack LH]=ON),JMP LBL[9] ;
  19:  IF (F[5:Master Rack RH]=ON),JMP LBL[10] ;
  20:  END ;
  21:   ;
  22:  LBL[1:CUBE PALLET LOAD] ;
  23:  SR[1]=$RSPACE1[1].$DIN_INDX ;
  24:  $RSPACE1[1].$UTOOL_NUM=R[6:TOOLNUM] ;
  25:  END ;
  26:   ;
  27:  LBL[2:CUBE LASER MARKE] ;
  28:  SR[1]=$RSPACE1[2].$DIN_INDX ;
  29:  $RSPACE1[2].$UTOOL_NUM=R[6:TOOLNUM] ;
  30:  END ;
  31:   ;
  32:  LBL[3:CUBE LEAKTEST LH] ;
  33:  SR[1]=$RSPACE1[3].$DIN_INDX ;
  34:  $RSPACE1[3].$UTOOL_NUM=R[6:TOOLNUM] ;
  35:  END ;
  36:   ;
  37:  LBL[4:CUBE LEAKTEST RH] ;
  38:  SR[1]=$RSPACE1[4].$DIN_INDX ;
  39:  $RSPACE1[4].$UTOOL_NUM=R[6:TOOLNUM] ;
  40:  END ;
  41:   ;
  42:  LBL[5:CUBE ASSEMBLY] ;
  43:  SR[1]=$RSPACE1[5].$DIN_INDX ;
  44:  $RSPACE1[5].$UTOOL_NUM=R[6:TOOLNUM] ;
  45:  END ;
  46:   ;
  47:  LBL[6:CUBE PAL UNLOAD] ;
  48:  SR[1]=$RSPACE1[6].$DIN_INDX ;
  49:  $RSPACE1[6].$UTOOL_NUM=R[6:TOOLNUM] ;
  50:  END ;
  51:   ;
  52:  LBL[7:CUBE CONV FAIL] ;
  53:  SR[1]=$RSPACE1[7].$DIN_INDX ;
  54:  $RSPACE1[7].$UTOOL_NUM=R[6:TOOLNUM] ;
  55:  END ;
  56:   ;
  57:  LBL[8:CUBE CONV IMPREG] ;
  58:  SR[1]=$RSPACE1[8].$DIN_INDX ;
  59:  $RSPACE1[8].$UTOOL_NUM=R[6:TOOLNUM] ;
  60:  END ;
  61:   ;
```

```
  62:  LBL[9:CUBE RACK LH ] ;
  63:  SR[1]=$RSPACE1[9].$DIN_INDX ;
  64:  $RSPACE1[9].$UTOOL_NUM=R[6:TOOLNUM] ;
  65:  END ;
  66:   ;
  67:  LBL[10:CUBE RACK RH] ;
  68:  SR[1]=$RSPACE1[10].$DIN_INDX ;
  69:  $RSPACE1[10].$UTOOL_NUM=R[6:TOOLNUM] ;
  70:   ;
  71:  IF (F[18:OP Master Racks]=OFF) THEN ;
  72:      !Check ok to enter space ;
  73:      IF (DO[8:RX_08_TPENBL]=OFF) THEN ;
  74:          R[18:CUBE INPUT NUM]=SR[1]    ;
  75:          IF (DI[R[18]]=OFF) THEN ;
  76:              MESSAGE[WATING TO ENTER CUBE] ;
  77:              WAIT DI[R[18]]=ON    ;
  78:          ENDIF ;
  79:      ENDIF ;
  80:  ENDIF ;
/END
```

The CUBIO Program extracts the assigned Digital Input number that has been set for the SPACE CUBE accounted with the Application equipment. It helps in being able to display a message if the robot is wait for the signal to be ON to enter the equipment area.

```
/PROG  CUBEENABLE
   1:  !GET CURRENT STATE ;
   2:  R[21:CUBE ENA/DIS 0/1]=$RSPACE1[1].$ENABLED ;
   3:  !DISABLE on Teach ;
   4:  IF (DO[8:RX_08_TPENBL]=ON) AND (R[21:CUBE ENA/DIS 0/1]=1) THEN ;
   5:      $RSPACE1[1].$ENABLED=0 ;
   6:      $RSPACE1[2].$ENABLED=0 ;
   7:      $RSPACE1[3].$ENABLED=0 ;
   8:      $RSPACE1[4].$ENABLED=0 ;
   9:      $RSPACE1[5].$ENABLED=0 ;
  10:      $RSPACE1[6].$ENABLED=0 ;
  11:      $RSPACE1[7].$ENABLED=0 ;
  12:      $RSPACE1[8].$ENABLED=0 ;
  13:      $RSPACE1[9].$ENABLED=0 ;
  14:      $RSPACE1[10].$ENABLED=0 ;
  15:  ENDIF ;
  16:  !ENABLE on Auto ;
  17:  IF (DO[8:RX_08_TPENBL]=OFF) AND (R[21:CUBE ENA/DIS 0/1]=0) THEN ;
  18:      $RSPACE1[1].$ENABLED=1 ;
  19:      $RSPACE1[2].$ENABLED=1 ;
  20:      $RSPACE1[3].$ENABLED=1 ;
  21:      $RSPACE1[4].$ENABLED=1 ;
  22:      $RSPACE1[5].$ENABLED=1 ;
  23:      $RSPACE1[6].$ENABLED=1 ;
  24:      $RSPACE1[7].$ENABLED=1 ;
  25:      $RSPACE1[8].$ENABLED=1 ;
  26:      $RSPACE1[9].$ENABLED=1 ;
  27:      $RSPACE1[10].$ENABLED=1 ;
  28:  ENDIF ;
/END
```

The program CUBEENABLE does the work of allowing to enter the equipment area by disableing the SPACE function in MANUAL and RE-Enableing in AUTO by using the Rotate switch on the teach pendant.

```
/PROG  IO2CCLINK
   1:  !ECHO CCLINK INPUT ;
   2:  DO[36:RX_36_BG_ChkEcho]=(DI[36:RY_36_BG_Heartbeat]) ;
   3:   ;
   4:  !ECHO TYPE PROX INPUTS ;
   5:  DO[37:RX_37_MhevPart1]=(RI[5:Mhev 1 Prx ]) ;
   6:  DO[38:RX_38_PhevPart1]=(RI[6:Phev 1 prx ]) ;
   7:  DO[39:RX_39_MhevPaet2]=(RI[7:Mhev 2 prx]) ;
   8:  DO[40:RX_40_PhevPart2]=(RI[8:Phev 2 prx]) ;
   9:   ;
  10:  !ECHO GRIPPER INPUT ;
  11:  DO[49:RX_49_ToolA Open]=(RI[1:TOOLA is OPEN]) ;
  12:  DO[50:RX_50_ToolA Shut]=(RI[2:TOOLA is SHUT ]) ;
  13:  DO[51:RX_51_ToolB Open]=(RI[3:TOOLB is OPEN]) ;
  14:  DO[52:RX_52_ToolB Shut]=(RI[4:TOOLB is SHUT]) ;
  15:   ;
/END
```

Echo on the CCLINK the state of the Robot's Gripper and Part Present proximity sensors.