SECTION 2

COMMAND ROUTINES

COMMAND INDEX

```
MAIN  LOOP

;  Program start

STARM           CALL   CLRSC     ;  Clear the TRS8Ø Screen
                LD     HL,SIGON  ;  Point to sign on message
                CALL   PSTR      ;  Print it
                CALL   PNEWL     ;  Print a new line
                CALL   INIT      ;  Set up system
QUES1           CALL   DELT      ;  Small delay
                LD     HL,QUESS  ;  Point to menue string
                CALL   PSTR      ;  Print it
                CALL   GCHRA     ;  Get response and print it
                CALL   PNEWL     ;  Print new line
                CP     NL        ;  Is response a newline
                JR     Z,QUES1   ;  Yes then ignore
                CP     'L'       ;  Is response an 'L'
                JP     Z,LEARN   ;  Yes do learn section
                CP     'E'       ;  Is it an 'E'
                JP     Z,EDIT    ;  Yes do edit
                CP     'R'       ;  Is it an 'R'
                JP     Z,READ    ;  Yes then do read command
                CP     'W'       ;  Is it a 'W'
                JP     Z,WRITE   ;  Yes do write command
                CP     'C'       ;  Is it a 'C'
                JP     Z,CHECK   ;  Yes do check routine
                CP     'S'       ;  Is it an 'S'
                JP     Z,SETAM   ;  Yes then do arm set
                CP     'T'       ;  a 'T'
                JP     Z,TOSTM   ;  Yes then move arm to start
                CP     'G'       ;  a 'G'
                JP     Z,GO      ;  Do execute movements stored
                CP     'D'       ;  a 'D'
                JP     Z,DISP    ;  Yes then display ARST array
                CP     'B'       ;  a 'B'
                JP     Z,BOOT    ;  Yes then restart system
                CP     'M'       ;  an 'M'
                JP     Z,MANU    ;  Yes the Manual control of arm
                CP     'F'       ;  a 'F'
                JP     Z,FREARM  ;  Yes then clear all motors
                CP     'Q'       ;  a 'Q'
                JP     Z,FINSH   ;  Yes then quit program
                LD     HL,QMESS  ;  Point to 'PARDON' message
                CALL   PSTR      ;  Print it
                JP     QUES1     ;  Try for next command
```

THE LEARN ROUTINE

```
;  This section deals with the recording
;  of an arm sequence

LEARN       LD     HL,RELNS   ;  Point to learn message
            CALL   PSTR       ;  Print the message
            CALL   GCHRA      ;  Get response and print it
            CALL   PNEWL      ;  Print a new line
            CP     '.'        ;  Response a '.'
            JP     Z,QUES1    ;  Back to main loop is uder types a '.
            CP     'S'        ;  Response an 'S'
            JR     Z,WAIT1    ;  Learn sequence from start
            CP     'C'        ;  a 'C'
            JR     Z,NOINT    ;  Continue learning from end of
                              ;  sequence
            CALL   PNEWL      ;  output a new line
            JR     LEARN      ;  Bad answer so try again
WAIT1       CALL   MOVTO      ;  Move arm to start position
            CALL   INIT       ;  Clear variables
WAIT2       LD     HL,CASRD   ;  Point to waiting message
            CALL   PSTR       ;  Print it
            CALL   GCHRA      ;  Get response and print it
            CALL   PNEWL      ;  Print new line character
            CP     '.'        ;  Response a '.'
            JP     QUES1      ;  Exit to main loop if so
            CP     SPAC       ;  Is it a space?
            JR     NZ,WAIT2   ;  If not then bad input, try again
            CALL   TORQUE     ;  Switch motors on
            JR     STLRN      ;  Do rest of learn
NOINT       LD     HL,(COUNT) ;  Get current count
            LD     A,L        ;
            OR     H          ;  Is it zero?
            JR     Z,NOSTR    ;  Yes then can't add to nothing
STLRN       XOR    A          ;  Clear manual flag
            LD     (MAN)A     ;  Because we are in learn mode
CONLN       CALL   KEYIN      ;  Drive motors and store sequence
            OR     A          ;  Zero key pressed
            JR     NZ,CONLN   ;  No then continue
            CALL   MOVTO      ;  Move arm to start position
            JP     QUES1      ;  Back to main loop
```

EDIT FUNCTION

```
EDIT    LD      HL,(COUNT)      ;  Get row count
        LD      A,L             ;
        OR      H               ;  Test for zero
        JP      Z,NOSTR         ;  Yes then nothing in store
EDSRT   LD      HL,ECOMS        ;  Print edit message
        CALL    PSTR            ;
        CALL    GCHRA           ;  Get response
        CALL    PNEWL           ;  Print a new line
        CP      'M'             ;  Is response an 'M'
        JR      Z,EDMOT         ;  Yes then edit motor
        CP      'R'             ;  Is response an 'R'
        JR      NZ,EDSRT        ;  No then try again
        LD      HL,COUTS        ;  HL = New row count message
        CALL    PSTR            ;  Print it
        CALL    GINT            ;  Get 16 bit signed integer
        JP      NZ,BADC         ;  Non zero return means bad input
        LD      A,H             ;  Test top bit of HC
        BIT     7,A             ;
        JP      NZ,BADC         ;  If negative then bad input
        LD      BC,(COUNT)      ;  Get count value
        PUSH    HL              ;  Save response
        OR      A               ;  Clear carry flag
        SBC     HL,BC           ;  See if response < current count
        POP     HL              ;  Restore response
        JR      NC,BADC         ;
        LD      (COUNT),HL      ;  Replace count with response
        JP      QUES1           ;  Back to main loop
EDMOT   LD      HL,EDSTR        ;
        CALL    PSTR            ;  Print 'row number'
        CALL    GINT            ;  Get integer response
        JR      NZ,BADC         ;  Bad answer
        LD      A,H             ;
        BIT     7,A             ;  No negative row count
        JR      NZ,BADC         ;  allowed
        LD      A,H             ;
        OR      L               ;  or zero row count
        JR      Z,BADC          ;
        LD      BC,(COUNT)      ;  Get row count into BC
        INC     BC              ;  Move count up one
        PUSH    HL              ;  Clear carry flag
        SBC     HL,BC           ;  Subtract count from response
        POP     HL              ;  Restore response
        JR      NC,BADC         ;  If greater than allowed error
EDOK    DEC     HL              ;  Move response down one
        ADD     HL,HL           ;  Double HL
        PUSH    HL              ;  Save it
        ADD     HL,HL           ;  Row count x 4
        POP     BC              ;  BC = row count x 2
```

```
        ADD     HL,BC       ;  HL = Row count x 6
        LD      BC,ARST     ;  Get store start address
        ADD     HL,BC       ;  Add row offset
        PUSH    HL          ;  Save resulting pointer
        LD      HL,MOTNS    ;  Print
        CALL    PSTR        ;  Motor number string
        CALL    GINT        ;  Get Answer
        JR      NZ,BADNM    ;  Bad answer
        LD      A,H         ;
        OR      A           ;
        JR      NZ,BADNM    ;  Response too large
        LD      A,L         ;
        CP      1           ;
        JR      C,BADUM     ;  No motor number < 1
        CP      7           ;
        JR      NC,BADNM    ;  No motor number > 6
        POP     HL          ;  Restore = Memory pointer
        DEC     A           ;  Motor offset Ø → 5
        LD      C,A         ;
        LD      B,Ø         ;  Add to memory pointer
        ADD     HL,BC       ;  Now we point to motor in store
        PUSH    HL          ;  Save pointer
        LD      HL,NVALS    ;
        CALL    PSTR        ;  Print new step value
        CALL    GINT        ;  Get response
        JR      NZ,BADNM    ;  Bad answer
        LD      A,H         ;
        CP      ØFFH        ;
        JR      NZ,PEDIT    ;  We have a positive response
        BIT     7,L         ;  New negative step value too
        JR      Z,BADNM     ;  large
        JR      MOTAS       ;  Step value OK
PEDIT   OR      A           ;  New positive step value too
        JR      NZ,BADNM    ;  large
        BIT     7,L         ;  so exit
        JR      NZ,BADNM    ;  else  ok
MOTAS   LD      A,L         ;  Get step value
        POP     HL          ;  Restore memory pointer
        LD      (HL),A      ;  Place step value in store
        JP      QUES1       ;  Go do next operation
BADNM   POP     HL          ;
BADC    LD      HL,BADMS    ;  Print error message and
        CALL    PSTR        ;
        JP      QUES1       ;  return to main loop
```

READ ROUTINE

```
;  Reads stored sequence from cassette
;  into memory

READ    LD      HL,CASRD    ;  Point to wait message
        CALL    PSTR        ;  Print it
        CALL    GCHRA       ;  Get response
        CALL    PNEWL       ;  Print new line
        CP      '.'         ;  Is response a dot?
        JP      Z,QUES1     ;  Yes then exit
        CP      SPAC        ;  Is it a space?
        JR      NZ,READ     ;  No then try again
        XOR     A           ;  Clear A=Drive zero
        CALL    CASON       ;  Switch on drive zero
        CALL    DELS        ;  Short delay
        CALL    RDHDR       ;  Read header from tape
        CALL    READC       ;  Read first character
        LD      B,A         ;  Put in B
        CALL    READC       ;  Read second character
        LD      C,A         ;  Place in C
        OR      B           ;  BC now equals count
        JP      Z,NOSTR     ;  Count zero, so exit
        LD      (COUNT),BC  ;  Set count = read count
        LD      HL,ARST     ;  Point to start of store
ROWNR   PUSH    BC          ;  Same count
        LD      E,Ø         ;  E = Check sum for a row
        LD      B,6         ;  B = Column Count
RDBYT   CALL    READC       ;  Read a row element
        LD      (HL),A      ;  Store it
        ADD     A,E         ;  Add it to check sum
        LD      E,A         ;  Store in check sum
        INC     HL          ;  Inc memory pointer
        DJNZ    RDBYT       ;  Do next element
        POP     BC          ;  Restore row count
        CALL    READC       ;  Read check digit
        CP      E           ;  Same as calculated?
        JR      NZ,RDERR    ;  No then error
        DEC     BC          ;  Decrement row count
        LD      A,B         ;  See if row count
        OR      C           ;  is zero
        JR      NZ,ROWNR    ;  No then read next row
        CALL    CASOF       ;  Switch cassette off
        JP      TAPEF       ;  exit
RDERR   LD      HL,RDMSG    ;  Error message for tape
        CALL    PSTR        ;  Print it
        JP      QUES1       ;  Go to main loop
```

WRITE ROUTINE

```
;  Writes a stored sequence to tape

WRITE       LD      BC,(COUNT)      ;  Get row count
            LD      A,B             ;
            OR      C               ;
BADWI       JP      Z,NOSTR         ;  If zero exit
            LD      HL,CASRD        ;  print message
            CALL    PSTR            ;
            CALL    GCHRA           ;  Get answer
            CALL    PNEWL           ;  Print new line
            CP      '.'             ;  Is answer a dot
            JP      Z,QUES1         ;  Yes then exit
            CP      SPAC            ;  Is answer a space
            JR      NZ,BADWI        ;  No then try again
            XOR     A               ;  Clear drive number
            CALL    CASON           ;  Switch on drive zero
            CALL    DELT            ;  delay
            CALL    WRLDR           ;  Write Leader
            CALL    DELT            ;  delay
            LD      BC,(COUNT)      ;  Get count into BC
            LD      A,B             ;
            CALL    WRBYA           ;  Write higher byte
            LD      A,C             ;  Get lower byte of count into A
            CALL    DELT            ;  delay
            CALL    WRBYA           ;  Write lower byte
            LD      HL,ARST         ;  Point to start of sequence of store
ROWNW       PUSH    BC              ;  Save row count
            LD      E,Ø             ;  Clear check sum
            LD      B,6             ;  Six motor slots per row
WRBYT       LD      A,(HL)          ;  Get motor  slot N
            CALL    DELS            ;  delay
            CALL    WRBYA           ;  Write it
            CALL    DELS            ;  delay
            ADD     A,E             ;  add to check sum
            LD      E,A             ;
            INC     HL              ;  Inc memory pointer
            DJNZ    WRBYT           ;  Do for all six motors
            CALL    WRBYA           ;  Write check sum
            POP     BC              ;  Restore row count
            DEC     BC              ;  Decrement row count
            LD      A,B             ;
            OR      C               ;  Test if zero
            JR      NZ,ROWNW        ;  No then try again
            CALL    CASOF           ;  Switch cassette off
            JP      QUES1           ;  Back to main loop
```

CHECK ROUTINE

```
;   Checks tape with sequence in store

CHECK       LD      BC,(COUNT)      ;  Get row count
            LD      A,B             ;
            OR      C               ;
            JP      Z,NOSTR         ;  If zero exit
BADCI       LD      HL,CASRD        ;  Print wait message
            CALL    PSTR            ;
            CALL    GCHRA           ;  Get answer
            CALL    PNEWL           ;  Print new line
            CP      '.'             ;  Is response a '.'
            JP      Z,QUES1         ;  Yes then go to main loop
            CP      SPAC            ;  Is it a space
            JR      NZ,BADCI        ;  No then try again
            XOR     A               ;  Clear cassette number
            CALL    CASON           ;  Switch drive zero on
            CALL    RDHDR           ;  Read header from tape
            LD      BC,(COUNT)      ;  Get row count
            CALL    READC           ;  Read first section
            CP      B               ;  Same?
            JR      NZ,RDERR        ;  No then error
            CALL    READC           ;  Read lower byte of count
            CP      C               ;  Same?
            JR      NZ,RDERR        ;  No then error
            OR      B               ;  Zero count from tape
            JP      Z,NOSTR         ;  So exit
            LD      HL,ARST         ;  Point to start of memory
ROWNC       PUSH    BC              ;  Save count
            LD      E,Ø             ;  Check sum is zero
            LD      B,6             ;  Count is 6
CKBYT       CALL    READC           ;  Read a motor step element
            CP      (HL)            ;  Same as in store?
            JP      NZ,RDERR        ;  Not the same so error
            ADD     A,E             ;
            LD      E,A             ;  Add to check sum
            INC     HL              ;  Advance memory pointer
            DJNZ    CKBYT           ;  Do next row element
            POP     BC              ;  Restore row count
            CALL    READC           ;  Read check sum
            CP      E               ;  Same as check sum calculated
            JP      NZ,RDERR        ;  No then error
            DEC     BC              ;  Decrement count
            LD      A,B             ;
            OR      C               ;  Is count zero?
            JP      NZ,ROWNC        ;  No then do next row
            CALL    CASOF           ;  Switch cassette off
TAPEF       LD      HL,TAPOK        ;  Print tape off message
            CALL    PSTR            ;
            JP      QUES1           ;  and back to main loop
```

BOOT AND FINISH COMMANDS

```
;  This routine restarts the program


BOOT    LD      HL,BOOTS        ;  Print "DO YOU REALLY
        CALL    PSTR            ;  WANT TO RESTART?"
        CALL    GCHRA           ;  Get answer
        CP      'Y'             ;  User typed 'Y'?
        JP      Z,STARM         ;  Yes then restart program
        CP      'N'             ;  No 'N'?
        JR      NZ,BOOT         ;  Then try again
        CALL    PNEWL           ;  else print new line and
        JP      QUES1           ;  back to main loop

;  This is the exit from program Section to TRS80
;  system level

FINSH   LD      HL,RELYQ        ;  Print "REALLY QUIT"
        CALL    PSTR            ;
        CALL    GCHRA           ;  Get answer
        CP      'Y'             ;  User typed a 'Y'
        JR      NZ,TRYNO        ;  No then try 'N'
        LD      HL,SIGOF        ;  Print ending message
        CALL    PSTR            ;  and then
        JP      FINAD           ;  return to TRS80 System
TRYNO   CP      'N'             ;  User typed an 'N'
        JR      NZ,FINSH        ;  No then try again
        CALL    PNEWL           ;  Print a new line
        JP      QUES1           ;  Back to main loop
```

OTHER SHORT COMMANDS

```
;  SETAM  clears arm position array

SETAM     CALL    RESET   ;  Clear Arm array  (POSAR)
          JP      QUES1   ;  Back to main loop

;  TOSTM  moves the arm back to its start position

TOSTM     CALL    MOVTO   ;  Steps motors till POSAR elements
          JP      QUES1   ;  are zero then back to main loop

;  FREARM  frees all motors for user to move arm
;  by hand

FREARM    CALL    CLRMT   ;  Output all ones to motors
          JP      QUES1   ;  and now to main loop

;  MANU allows the user to move the arm using
;  the 1-6 keys and the 'Q' 'W' 'E' 'R' 'T' 'Y' keys
;  The movements made are not stored.

MANU      LD      A,1     ;  Set in manual mode for the
          LD      (MAN),A ;  keyin routine
MANUA     CALL    KEYIN   ;  Now get keys and move motors
          JP      NZ,MANUA;  If non zero then move to be done
          XOR     A       ;  Clear manual flag
          LD      (MAN),A ;
          JP      QUES1   ;  Back to main loop
```

THE GO COMMAND

```
;  This command causes the computer to step
;  through a stored sequence and makes the arm
;  follow the steps stored, if the sequence is to
;  be done forever then the arm resets itself at
;  the end of each cycle.

GO      CALL    PNEWL       ;  Print a new line
        CALL    MOVTO       ;  Move arm to start
        XOR     A           ;  Clear
        LD      (FORFG),A   ;  Forever Flag FORFG
        LD      HL,AORNM    ;  Print "DO ONCE OR FOREVER
        CALL    PSTR        ;  Message
        CALL    GCHRA       ;  Get answer and print it
        CALL    PNEWL       ;  Print a new line
        CP      'O'         ;  User typed an 'O'
        JR      Z,ONECY     ;  Do sequence till end
        CP      'F'         ;  User typed an 'F'
        JR      NZ,GO       ;  No then re-try
        LD      A,1         ;  Set forever flag
        LD      (FORFG),A   ;  to 1
ONECY   LD      A,'.'       ;  Print a '.'
        CALL    PUTCHR      ;  Using PUTCHR
        CALL    DOALL       ;  Execute the sequence
        LD      A,(FORFG)   ;  Test FORFG, if zero
        OR      A           ;  then we do not want
        JR      Z,NORET     ;  to carry on so exit
        CALL    DELT        ;  delay
        CALL    MOVTO       ;  Move arm to start
        CALL    DELLN       ;  Delay approx 1 second
        JR      ONECY       ;  Do next sequence
NORET   LD      HL,DONMS    ;  Print sequence done
        CALL    PSTR        ;
        JP      QUES1       ;  and go to main loop
```

THE DISPLAY COMMAND

```
;  This command allows the user to display
;  the motor sequence so that he can then
;  alter the contents of a sequence by using
;  the edit command


DISP     LD      HL,DISPS      ;  Point to header string
         CALL    PSTR          ;  and display it
         CALL    POSDS         ;  Print out the relative position
         LD      HL,ARST       ;  Point to sequence start
         LD      BC,(COUNT)    ;  BC = how many rows to print
         LD      A,B           ;
         OR      C             ;  Test if count is zero
         JP      NZ,SETBC      ;  No then jump to rest of
NOSTR    LD      HL,NODIS      ;  display else print message
         CALL    PSTR          ;  telling user no display and
         JP      QUES1         ;  return to the main loop
SETBC    LD      BC,ØØØ        ;  Clear BC for row count
DOROW    PUSH    BC            ;  Save it
         PUSH    HL            ;  Save memory position
         LD      H,B           ;
         LD      L,C           ;  HL = row count
         INC     HL            ;  Now row count =N+1
         LD      IX,NUMAR      ;  IX  points to buffer for ASCII String
         CALL    CBTAS         ;  Convert HL to ASCII
         LD      HL,NUMAR      ;  Point to ASCII string
         CALL    PSTR          ;  now print it
         LD      A,'.'         :
         CALL    PUTCHR        ;  Print a '.'
         POP     HL            ;  Restore memory pointer
         LD      B,6           ;  Motor count to B (6 motors)
NEXTE    LD      A,(HL)        ;  Get step value
         PUSH    HL            ;  Save memory pointer
         PUSH    BC            ;  Save motor count
         BIT     7,A           ;  Test bit 7 of A for sign
         JR      Z,NUMPO       ;  If bit = Ø then positive step
         LD      H,ØFFH        ;  Make H = negative number
         JR      EVAL          ;  Do rest
NUMPO    LD      H,Ø           ;  Clear H for positive number
EVAL     LD      L,A           ;  Get low order byte into L
         LD      IX,NUMAR      ;  Point to result string
         CALL    CBTAS         ;  Call conversion routine
         LD      HL,NUMAR      ;  HL points to result
         CALL    PSTR          ;  Print resulting conversion
         LD      A,(381ØH)     ;  Get keyboard memory location
         BIT     Ø,A           ;  Test for zero key pressed
         JR      Z,NOSTP       ;  Not pressed, then skip
DOSTP    CALL    GCHR          ;  Wait till next character entered
         CP      '.'           ;  Is it a dot?
         JR      NZ,NOSTP      ;  No then carry on
         CALL    PNEWL         ;  else print a new line
         POP     BC            ;  and restore all the registers
         POP     HL            ;  and the stack level
```

```
                POP     BC          ;
                JP      QUES1       ; Jump back to main loop
NOSTP           POP     BC          ; Restore column count
                POP     HL          ; Restore memory pointer
                INC     HL          ; Increment memory pointer
                CALL    PSPAC       ; Print a space between
                                    ; numbers
                DJNZ    NEXTE       ; Do for six motors
                CALL    PNEWL       ; Print a new line
                POP     BC          ; Restore row count
                INC     BC          ; Increment row count
                LD      A,(COUNT)   ; Get lower count byte
                CP      C           ; Is it the same
                JR      NZ,DOROW    ; No then do next row
                LD      A,(COUNT+1) ; Get higher order count byte
                CP      B           ; Same?
                JR      NZ,DOROW    ; No then do next row else
                CALL    PNEWL       ; print a new line and then
                JP      QUES1       ; back to main loop
```

SUBROUTINES INDEX

DOALL.....Execute a stored sequence once

DRIVL.....Drives all motors directed by TBUF

INIT.....Set up system

MOVTO.....Use POSAR to rest system arm

TORQUE.....Turn on off motors

CLRMT.....Turn off all motors

SETDT.....Reset CTPOS elements to one

DRAMT.....Drive directed motors

STEPM.....Step motors via DRAMT

DNEWD.....Delay on direction change

SPAMT.....Update TBUF array during learn

KEYIN.....Scan keyboard and build up motors to move

CBTAS.....Convert 16 bit 2's complement number to ASCII

CLRMF.....Clear MOTBF array

CTBUF.....Clear TBUF, DRBUF & MOTBF arrays

GINT.....Get 16 bit signed value from keyboard

POSDS.....Display relative position array elements

POSIC..... .....Increment relative position array elements

STORE.....Copy TBUF to current ARST slice

RESET.....Clear POSAR array

PUTCHR.....Print a character

PSTR.....Print a string

PSPAC.....Print a space

PNEWL.....Print a carriage return