SYDTRUG NEWS

NOVEMBER 1982

HELP WANTED

The material for this newsletter has been contributed by members of the Sydney TRS-80 Users Group (SYDTRUG) for the benefit of other members. Continuation of the Newsletter can only continue if other members of the group also pass on their experience and contribute. Listings of original software would be appreciated.

Contributions may be left with our genial host Mick Rowney, at Patterson's Florist, 1120 Botany Road, Botany, or with the editor Colin Elphick, 37 Hall Drive, Menai.

USING THE "VTOS" OPERATING SYSTEM

COMMUNICATION PACKAGE

by Robert Gareb

A number of people have been asking for a quick summary of how to use VTOS to communicate with a mainframe/bulletin board. This will give a quic guide, but it is still best to read the manual properly yourself and experiment.

Following is a list of the various local functions available on VTOS :

1	KI	!	DUPLEX
2	DO	"	ECHO
3	PR	#	----
4	CL	$	AUTO-LF
5	FI	%	REWIND
6	FO	&	PEOF
7	----	'	----
8	----	(	----
9	ID	)	VTCOMM
Ø	RESET	_	----
:	ON	*	----
-	OFF	=	VTOS

To use these extra keys:
For the keys in the left column, press <clear> and the relevant key, while for the keys on the right column press <clear> <shift> and the relevant numeral.

I will only describe the basics. For more detail, either see me at any meeting, or read the manual.

Firstly, you have to set the communication parameters. This is easily done by either setting the DIP switches internally (the easiest method, because you will generally always use the same values, e.g. 3ØØ BAUD, 8 Bit No Parity for most bulletin boards and T.A.B.) or when you initialise the VTOS driver.

To tell VTOS that the info is coming in the RS232 line, you type:

LINK *CL TO RS232 <enter>

or if you have to set the baud rate, then type:

LINK *CL TO RS232 (BAUD=3ØØ) <enter>

For more info on all the other parameters refer to the operatins manual.

Now we must tell it to run the VTCOMM package and use the incoming data from the RS232 port:

VTCOMM *CL <enter>

Now everything coming in via the RS232 port will be displayed on the monitor.

OPTION 1

To get a hard copy of all data coming in as it is coming in:
Refer to the local control keys listed above.
The left side (e.g. 3 PR) keys are obtained by pressing <clear> and 3 respectively.
The right column keys are obtained by pressing <clear> <shift> and the respective key simultaneously.

To get a hard copy, you must turn the printer on:

press <clear>-3 <clear>-:
i.e. press clear and 3, then clear and :

This turns the printer on. Now all incoming data will go to the printer as well as the monitor.
To turn off, press <clear>-3 <clear>--
i.e. clear 3 clear -

OPTION 2

To send all incoming data to disk, you must:
1) Set up a file for incoming data
2) Turn on this file.
3) When finished, turn off the file.
4) Reset the file.

****
You **MUST** do these in this order. If you do not, you will not have the data on the disk ready to use at a later time.
Resetting before turning off resets the pointer, and the new data overwrites the old, while not resetting stops the update of the directory.
****
Procedure:

After setting up as above (i.e. VTCOMM), when you want the data to go to disk press:

<clear> 6
<clear> 9

The system will then ask you for a filename. Include extensions and drive no. if required.

You will find that you generally will first establish communications, and if using a bulletin board you will not want all the sign on data and messages to goto disc. With this system you can give the file name initially, and turn on and off the I/O at any time.
To turn on the I/O, i.e. to say "now write to disk":

<clear> 6
<clear> :

Then as you type in and as the host system sends data, all the information will be written to disc.
To turn I/O off type :

<clear> 6
<clear> -

This on off procedure can be repeated i.e. if you do not want everything to go to disc.
As you have already seen (hopefully) <clear> 6 says to the system that it is a file being received. Other commands can be similarly used for file to be sent.

***** Now we have turned our I/O off, what must we do?

(Refer to above if necessary)
Reset the disc I/O and thus fix the directory:

<clear> 6
<clear> 0

That will close the file, and all is complete.

QUICK SUMMARY OF HOW TO STORE INCOMING DATA TO DISC

<clear> 6	<clear> 9	Give incoming file a name.
<clear> 6	<clear> :	Turn on disc I/O.
<clear> 6	<clear> -	Turn off disc I/O.
<clear> 6	<clear> Ø	Reset file, update dir.

The same can be applied for LDOS except some of the names are different, i.e.
use SET *CL TO RS232R (for TRSØ)
use LCOMM instead of VTCOMM.

Once again I must emphasize:
IF ALL ELSE FAILS, READ THE MANUAL.

THE WESTERN DIGITAL F.D.C.

by Michael Cooper

EVERYTHING YOU WANTED TO KNOW ABOUT YOUR FLOPPY DISK CONTROLLER, BUT WERE TOO FRIGHTENED TO ASK...

So you've just bought the latest hot-shot programme and the diabolical author has issued it on a "protected" disk. Well, well. How is it possible that your TRS-80 can load the disk O.K. but then damn thing won't copy it? The answer to that riddle is to be found in the depths of the Floppy Disk Controller (F.D.C.). I hope by the time you've stumbled through this story you may just have the answer.

Before we begin, let's get one thing straight. Just because you can drive a car doesn't make you Jack Brabham, thus I am not setting myself up as any type of expert on this matter, but I do have a passing aquaintance with programming the F.D.C. Also, the hardware aspects of interfacing the F.D.C. to the TRS-80 buss will not be covered because I don't feel it necessary, just as Poke 15360,48 will put a "0" in the top left hand corner of the screen, how it gets there doesn't really matter, you can still use that command to achieve the desired result.

The Western Digital 1771 F.D.C. as used in the Model 1 can be thought of as a microprocessor with an instruction set of 11 commands. Using these commands it is possible to perform all the functions you need to read, write or format a disk. The F.D.C. also has 5 registers (8 bit each) which hold control information. These are:

(1). Data register
(2). Command register
(3). Sector register
(4). Track register
(5). Status register

Like everything else in the Model 1, these registers are memory mapped, which means that they can be found at certain memory addresses. The other crucial address used in disk operation is the Drive select latch, which allows you to turn on the drive motors and select the drive you wish to use. These addresses are as follows, please put them in your little black book.

Drive Select	37E1H (14304)
Data register	37EFH (14319)
Command Register	37ECH (14316)
Sector Register	37EEH (14318)
Track Register	37EDH (14317)
Status Register	37ECH (14316)

Each of the registers has a specific function, these are:

Data register is used to pass the data byte to or from the disk.
Sector register contains the current disk sector number.
Track register contains the current disk track number.
Command register contains the current F.D.C. command instruction.
Status register contains the current F.D.C. status and is used for error detection.

The first thing to notice here is that both the Command register and the Status register occupy the same address. This presents no problem as commands are a write operation (Poke) whereas status is a read operation (Peek).

The first task in using the F.D.C. is to understand the commands and as I mentioned earlier there are 11 of them and they are known as:

(1).	Restore	(Move the head to track zero)
(2).	Seek	(Find the track specified in the track register)
(3).	Step	(Step the head 1 track in the previous direction)
(4).	Step In	(Step the head 1 track in)
(5).	Step Out	(Step the head 1 track out)
(6).	Read	(Read 1 byte of data from disk)
(7).	Write	(Write 1 byte of data to disk)
(8).	Read address	(Read identification field - put on disk during format)
(9).	Read Track	(Read entire track)
(10).	Write Track	(Write entire track)
(11).	Force Interrupt	(Terminate operation)

Because the command register is 8 bits long, all of these commands consist of an 8 bit pattern which is sent to the command register. This causes the F.D.C. to do its thing. The following is a description of the bit patterns

used in each command and a description of the function of each bit. Note that these commands can be grouped into 2 separate groups. The head positioning group and the data transfer group. I will deal with the head positioning group first. In all commands the format is Bit 7 - Bit 0. Bit 7 is the left most bit and Bit 0 is the right most bit: i.e. 7 6 5 4 3 2 1 0

RESTORE 0 0 0 0 H V R R

The H field is the head load field. On the TRS-80, the H field has no effect as the head is loaded automatically when the motor is turned on. On other machines, if H=1 then head is loaded, if H=0 then the head is not loaded.
The V field is the verify field and is used to verify that the track number read from the disk is the same as the current track number in the track register. Now this is where you can start protecting your disks. If you format track 1 as track 234 then if the V field is set (as it is in Backup) then your poor machine gets thoroughly confused and usually bombs out. Anyway, back to the restore command. If V=1 then the F.D.C. will verify the track number, if V=0 then the F.D.C. won't bother. Best to set it to 0.
The R R field is the head stepping rate: 0 0 = 6ms, 0 1 = 12ms, 1 0 = 20ms, 1 1 = 40ms. On the TRS-80 it is best to use the 40ms rate as that covers all types of disk drives used.
Now, all up, that means setting the Restore field as 0 0 0 0 0 0 1 1 which is decimal 3. So to restore your head to track 0 you send a 3 to address 37ECH (14316) O.K.?

SEEK 0 0 0 1 H V R R

We have already discussed the H V R R field and it is the same as in restore. We will always set it to 0 0 1 1 so the seek command will be 0 0 0 1 0 0 1 1 which is decimal 19. However, before we can use this command we must have sent the track number we require to the data register. When a seek command is issued after the track number has been stored in the data register the F.D.C. will automatically position the head over the proper track.

STEP 0 0 1 U H V R R

Again here, we already know what the H V R R fields do, and we know that they are set to 0 0 1 1. The U field determines whether the track register is to be updated after the head is stepped. If U=1 then it is incremented or decremented as the case may be. If U=0 then no adjustment is made. Thus to step the head 1 track in the previous direction and to update the track register we send the command 0 0 1 1 0 0 1 1 which is 51 decimal to the command register at address 37ECH (14316).

STEP IN 0 1 0 U H V R R
STEP OUT 0 1 1 U H V R R

The fields are the same as we have already discussed, the only difference is Bit 6 is set for both commands, and Bit 5 is set for a Step Out. The Step In command moves the head 1 track towards the centre of the disk, the Step Out moves it 1 track towards the edge (i.e. Towards Track 0). These commands are most useful when dealing with "protected" disks, most of which usually have funny Track and Sector numbers formatted. Any attempt to use the "Seek" command on such disks will lead to disaster unless you know the exact formatting pattern used.

Now lets move on to the Read and Write commands.

The first thing to understand here is that the TRS-80 uses "Register I/O" to transfer data between the disk and memory. This has the advantage of hardware simplicity, but extracts a high cost in CPU overhead. The CPU is continually looping, testing the status register waiting to transfer the next byte of data. To keep up with the F.D.C. the CPU must transfer a byte every 7B microseconds which is within the capability of the Z-80. Prior to the Read or Write operation the F.D.C. sector register must be loaded with the sector number to be read or written. Also the head must be positioned over the required track using one of the aforementioned positioning commands. The sector register is loaded by a "store" instruction to address 37EEH (14318) of the sector number required eg.

```
LD   A,01H      ;Load
LD   (37EEH),A  ;Sector #1
```

Having done this we can now read or write sector number 1.

READ 1 0 0 M B E 0 0
WRITE 1 0 1 M B E X Y

There are 3 fields in the Read command and 4 fields in the Write command.

The M field is used to read or write single or multiple sectors. If M=0 then a single sector will be transferred. If M=1 then more than 1 sector will be transferred. Usually with the TRS-80 only 1 sector is transferred.
The B field is used to indicate whether the disk is formatted in the IBM standard or not. Standard disks, formatted by your favourite DOS will always be in IBM format, however some "protected" disks will not. If B=1 then IBM format is signified.
The E field is used to control a 10 millisecond delay to allow the head to stabilise before performing the read or write operation. In theory, since the head is always loaded in the TRS-80 this shouldn't be necessary, but Tandy use it, so we will too. If E=1 then the delay will occur.
The X and Y fields are only used in the write command, and control the writing of the Data Address Mark (DAM) on the disk - more on this later on when we discuss disk formatting. The standard IBM format uses value F8H, which is specified by setting X=0 and Y=0.

READ ADDRESS 1 1 0 0 0 1 0 0

The read address command reads six identification bytes from the disk. These bytes are put on the disk during the formatting process and are as follows:

(1). Track number (0 - 34/39/79) depends on track count of your drive
(2). String of zeroes (used as a delimiter)
(3). Sector number (0 - 9) single density only
(4). Sector length
(5). CRC byte 1
(6). CRC byte 2 (these are the checksum bytes)

These will make more sense later when we cover the formatting process.

READ TRACK 1 1 1 0 0 1 0 S

The S field is the only option here and is used to synchronise the F.D.C. with the byte flow. Since this command reads a whole track of bytes, including the control bytes put down during the formatting process, it can sometimes get a bit lost, (or should I say "a lost bit"). If the S=1 then the accumulation of bytes is synchronised to each address mark found. If S=0 then no synchronisation is done. The read track command is primarily used during the formatting process to verify that all is well.

WRITE TRACK 1 1 1 1 0 1 0 0

There are no options here. This command simply writes bytes to the track over which the head has been positioned. We use this command to format a disk by setting up the format bytes in memory and then transferring them to the disk. Obviously no data bytes are sent during this process, only the header information (Track number, sector number, sector length etc) and filler bytes (E5H) in the data blocks. The address marks are written to the disk by the F.D.C. when it encounters a byte in the range F7H - FEH. These are as follows:

F7H	Write CRC
FBH	Data Address Mark
FCH	Index Address Mark
FEH	ID Address Mark

FORCE INTERRUPT 1 1 0 1 A B C D

This command forces cessation of any current F.D.C. operation. In many systems it can generate an interrupt request, but on the TRS-80 it can only interrupt if interrupts are enabled. These interrupts depend on the fields set by A B C D. If they are all set to zeroes then the current command is terminated, but no interrupt is generated.
If D=1 then terminate on not ready to ready transition.
If C=1 then terminate on ready to not ready transition.
If B=1 then terminate on next index pulse.
If A=1 then immediate terminate/interrupt.
In the TRS-80 a Force Interrupt with no interrupt (1 1 0 1 0 0 0 0) is used to terminate a read or write of multiple sectors after the desired number have been reached.

The last detail we need to pursue is the Status Register. This is used during all of the F.D.C. commands to see if the operation was successful or if we have any problems. Again, each of the 8 bits are used to signify some condition. These conditions vary depending on whether we are using a positioning command or an I/O command. For the positioning commands the bits are as follows:

7 6 5 4 3 2 1 0

Bit 7	Ready/Not ready.	Is Reset (0) when the disk is being addressed.
Bit 6	Write protect.	If Set (1) then indicates the write protect notch on the disk is covered.
Bit 5	Head engaged.	Is always Set (1) when a disk operation is being performed.
Bit 4	Seek error.	Is set (1) if track not found and verification active.
Bit 3	CRC error.	Is set (1) if verification active.
Bit 2	Track 0.	Is set (1) if head is over track 0.
Bit 1	Index.	Is set (1) when index pulse detected from disk, reset (0) otherwise.
Bit 0	Busy	Is set (1) while command in progress, reset (0) otherwise.

Normally you would use the busy bit to test if the F.D.C. was still busy prior to issuing a new command. If the bit was set (1) then go into a loop, testing this bit until it resets (0). Track 0 bit may be tested after a Restore command to make sure the head is positioned correctly.

During the Read and Write commands the Status register holds the status as follows:

7 6 5 4 3 2 1 0

Bit 7	Not ready.	If reset (0) F.D.C. is ready.
Bit 6	Write Protect.	If write operation and bit set (1) then write protect notch covered.
	Record type.	If read operation used with bit 5 to indicate IBM record
Bit 5	Write fault.	If write operation and bit set (1) then a write fault has occurred.
	Record type.	If read operation used with bit 6 to indicate IBM record Bits 6 & 5 should be set to 0 1 for IBM
Bit 4	Record not found.	If set (1) indicates desired track or sector or both not found.
Bit 3	CRC error.	If set (1) indicates bad ID data or bad user data.
Bit 2	Lost Data.	If set (1) indicates the CPU did not respond fast enough to the data flow.
Bit 1	Data Request.	If set (1) indicates data register empty on write, or data register full on read.
Bit 0	Busy.	Is set (1) while read or write is active.

During the Read Address, Read Track or Write Track commands the status bits would have the same meaning as in the Read or Write commands.

FORMATTING THE DISK.

The disk is formatted by writing a sequence of control and filler bytes onto each disk track in turn. The start of each track is signalled by the index pulse which is generated by a flash of light passing through the little hole near the centre of each disk. The standard format pattern for single density disks is as follows:

```
    18 bytes of value FFH preceeding the 1st sector.
.............................................(start of sector)
     1 byte  of value FEH (ID address mark).
     1 byte  containing Track number.
     1 byte  of value 00H.
     1 byte  containing Sector number.
     1 byte  of value 01H.
     1 byte  of value F7H (to generate 2 CRC bytes)
    12 bytes of value FFH (filler)
     6 bytes of value 00H (filler)
     1 byte  of value FBH (Data Address Mark)
   256 bytes of value E5H (For user data)
     1 byte  of value F7H (to generate 2 CRC bytes)
    12 bytes of value FFH (filler)
     6 bytes of value 00H (filler)
.............................................(end of sector)
```

The last sector differs from the others in that it ends with a stream of bytes (up to 186 of value FFH) until the F.D.C. signals end of track.

BOOTING A DISK.

To end off this article I will now demostrate how you can load a disk sector into memory. This particular code was devised by yours truly to enable me to unprotect a certain utility that we all think is quite super. This code is located on track 0, sector 0 (BOOT/SYS) and is loaded into memory at address 4200H by ROM when you press Reset, and then executed. The sector loaded by this code (number 2) contained my own custom loader which was used to load the remainder of the disk, which was formatted in a non-standard fashion and thus was considered to be "protected" from the normal backup process. You may find it handy for your own applications. The only part that requires explaination is the delay sequence. Here you will notice I swap the top of stack with register HL four times. This is just one way to waste some time between issuing a command to the F.D.C. and testing the status bits. It should be remembered that if you give a read/write command to the F.D.C. it is necessary to delay testing the status for a few microseconds else you can get a false status from the F.D.C. This does not apply when using the head positioning commands as they have the built in head delay field bit.

Entry Conditions:
HL=37ECH (Points to F.D.C. Command and Status registers)
DE=37EFH (Points to F.D.C. data register)

```
;*****************************************************************
;
;
            ORG   4200H
ENTRY       LD    (HL),0D0H    ;FORCE FDC INTERRUPT
            LD    A,01H        ;SELECT
            LD    (37E1H),A    ; DRIVE 0
            LD    (HL),03H     ;ISSUE COMMAND TO RESTORE TO TRACK 0
L1          BIT   0,(HL)       ;TEST UNTIL FDC
            JR    NZ,L1        ; SIGNALS RESTORE COMPLETED
            LD    A,02H        ;LOAD FDC SECTOR
            LD    (37EEH),A    ; REGISTER WITH VALUE 2
            LD    BC,0A000H    ;MEMORY ADDRESS WHERE DATA GOES
            LD    (HL),8CH     ;READ 1 SECTOR, IBM FORMAT
            EX    (SP),HL      ;DELAY TO
            EX    (SP),HL      ; WASTE
            EX    (SP),HL      ;  SOME
            EX    (SP),HL      ;   TIME
            JR    L3           ;GO TO TEST PHASE
L2          BIT   0,(HL)       ;TEST FOR "END OF RECORD" SIGNAL
            JP    Z,0A000H     ;ALL DONE? - YES, JUMP TO CODE JUST LOADED
L3          BIT   1,(HL)       ;TEST FOR "DATA BYTE READY"
            JR    Z,L2         ;NOT READY - KEEP TESTING
            LD    A,(DE)       ;READY - GET DATA BYTE FROM FDC DATA REGISTER
            LD    (BC),A       ;AND PUT INTO MEMORY
            INC   BC           ;INCREMENT MEMORY POINTER
            JR    L2           ;BACK FOR NEXT BYTE
```