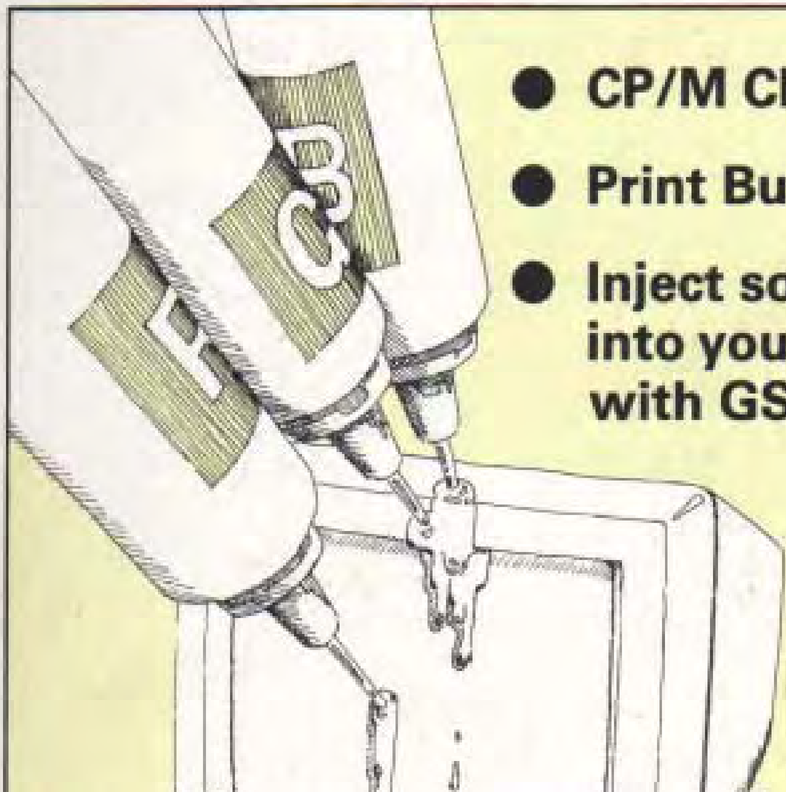
80-BUS NEWS

SEPTEMBER – OCTOBER 1984 VOL. 3 ISSUE 5



- CP/M CRC Program
- Print Buffer Software
- Inject some colour into your system with GSX

£1.50

The Magazine for GEMINI & NASCOM USERS

September–October 1984. 80-BUS NEWS Volume 3. Issue 5.

CONTENTS

SUBSCRIPTIONS

Annual Rates	UK	£9	Rest of World Surface	£12
	Europe	£12	Rest of World Air Mail	£20

Subscriptions to `Subscriptions´ at the address below.

EDITORIAL

Editor : Paul Greenhalgh   Associate Editor : David Hunt

Material for consideration to `The Editor´ at the address below.

ADVERTISING

Rates on application to `The Advertising Manager´ at the address below.

PRIVATE SALES

Free of charge to Subscribers by sending to the address below

ADDRESS: 80-BUS News,
c/o Gemini Microcomputers Ltd.,
Unit 4, Springfield Road,
Chesham, Bucks. HP5 1PU.

EDITORIAL

Questionnaire

I was wrong (just). In my last Editorial I said that I did not expect us to receive more than 20 per cent of the Questionnaires back. Well, they are still coming in (very very slowly) and there is a miniscule danger that we may reach 30 per cent. As we have never sent out anything like this before, I am uncertain as to how good a response this is, but it has certainly been giving Steve, who was `volunteered' to type the results into a database, plenty to be getting on with.

What has been fascinating is seeing the number of different ways people have been finding of filling them in. I thought, obviously somewhat foolishly, that there could only be one interpretation of how we expected the answers to be entered. Oh no, everybody has got their own idea, and some questionnaires have come back absolutely covered from wall to wall in writing. Steve has therefore had to spend a considerable amount of time working out how to extract useful information. We will be publishing the results soon, and I hope that all Steve's efforts have been worthwhile. A special thanks to all those who have responded. And if you are one of the 70 per cent (the silent majority) who has not returned the Questionnaire, then there may still be time, although you have not been included in the prize draw. On that subject, we will be publishing the names of the winners in the next issue.

Names and Addresses

Over the years various people have asked us to give them the names and addresses of other 80-BUS News subscribers in their areas. This seems like a very good way of getting people in touch with each other, as, for example, does either of two of the subscribers in Harrow, who live at numbers 70 and 77 of a particular road, know that there is another 80-BUS user so near? We have not previously released this sort of information as we respect that people may wish to maintain privacy. However what we have decided to do, with your permission, is publish your names and addresses in the magazine. When you receive your subscription reminder you will find that there is a `Yes' box and a `No' box alongside a question as to whether or not you would like your name and address being published. When we have received a reasonable number of these back then we will publish those that have agreed. This system was only started recently, and at the time of typing this we have had 54 of these resubscriptions back. The response looks very favourable, with 42 people ticking `Yes' and only 9 ticking `No'. I really don't know what to do with the 3 that didn't tick either!

Advertising

In one communication received recently there was a comment along the lines of "If Gemini think that 80-BUS is so wonderful, why don't they support the magazine that is dedicated to 80-BUS by advertising in 80-BUS News?". This made me realise that you non-subscribers don't realise that you are missing out!! Several times in the past, and again with this issue, Gemini have included various catalogues and data sheets with the subscription mailings. In addition the recent Questionnaire only went in with the subscription issues, apart from a few that were left over and that were therefore sent to one dealer. So remember, if you're not a fully paid up subscriber you may be missing vital information and opportunities!

Front covers

I ought here to belatedly thank Alf for his work. Alf has been responsible for drawing the front covers on the last few mags, and in my opinion he has made an excellent job of them. In Vol.3 Issue 3 I asked if anyone could understand its front cover. Well ONE person responded, on his Questionnaire (and unfortunately I haven't been able to find it in the pile again), and got it right. Well done sir. The answer?? Well the SVC has the ability of supporting a serial (cereal) keyboard!! (Yes, Alf may get 10 for quality, but only 2 for content!!)

Letters to the Editor

Pertec Mods.

I am a Nascom 2, Gemini 64K RAM card, GM829 FDC and Gemini IVC user. I am running CP/M 2.2 with Pertec FD250 drives.

I purchased surplus Pertec drives from the USA and had lots of problems. All the problems were a result of leaky decoupling capacitors. For those of you who intend to purchase surplus FD250, I suggest that all the decoupling capacitors should be replaced. Other than this, the drives are great.

There is a simple hardware and software modification to get these drives reading and writing 40 tracks instead of 35 tracks. Though I have done this independently, I understand "Henry's" is offering this modification. For those of you who are interested, please write and I will send full details.

Those of you who intend to make the printer-buffer as published in the June 1984 issue of BYTE, please note that the PIO output of the Nascom 2 or Gemini should be buffered and properly oriented with pull up resistors before it will work. I assume that the readers will have taken care of the other errors in the buffer hardware and software as published by BYTE

All correspondence on the above and other Nascom/Gemini subjects welcomed.

Yours truly, Hiten Patel, 4 Navyug Sagar, 183 Walkeshwar Road, BOMBAY 400 006, India.

Further thoughts on Hisoft Pascal

Following on from the random musings of Dr. Dark in the issue of July-August 1984, I am writing to tell of some procedures developed for just this need. The CP/M manual describes a file control block (FCB) which is used for random access, but this is not allowed for in the standard Hisoft Pascal file handler - the Pascal one is three bytes too short. So we want a definition of a FCB which can be used for random access, like this -

```
TYPE
    charll = ARRAY[1..11] OF CHAR;
{ charll is for the use of routines manipulating filenames
   unlike the Hisoft filename it coes not contain the : or .
   which are displayed in the directory listing so that
   a file will be   filenameext   and not filename.ext  }
    fcbtype = RECORD
      DRIVE : CHAR;
      NAME  : charll;
      EX : CHAR;
      S123 : ARRAY[1..3] OF CHAR;
      DIRECTY : ARRAY[1..16] OF CHAR;
      CREC : CHAR;
      RNDREC : INTEGER;
      RNDOVF : CHAR
    END;
VAR
{ whilst we are about it we can create three (or more) of the FCB's
   so that we can use more than one random file at a time }
    HEADER : ARRAY[1..3] OF fcbtype;
```

The other definitions of types and variables must be placed in the program as needed. The most important of these is a buffer area to store the record read from the file or about to be written to it. The simplest of definitions is simply an array of 128 bytes (or 64 integer numbers if you wish to use these or any other combination).

Having defined our FCB, there are a number of routines given in the CP/M manual which are of use apart from the obvious read random and write random. What of a function EXISTS which checks the existence of a file and returns a boolean value TRUE or FALSE ? Here it is -

```
FUNCTION EXISTS(title:char11):BOOLEAN;
VAR HEADER : fcbtype;
    i : INTEGER;
BEGIN
   i:=CPM(26,£80); (*reset DMA *)
   i:=fcbset(title,CHR(dkd),HEADER);
   i:=CPM(17,i); (* 255 if not found *)
   IF i = 255
      THEN EXISTS := FALSE
      ELSE EXISTS := TRUE
END; (* EXISTS *)
```

Note that we use the default DMA area (hex 80 to FF) to store the directory in as we read it, and the function used is number 17 which is search for first. Variable dkd is an integer giving the disk to be used 0=default 1=A 2=B etc. The function fcbset is used to copy the title and the disk number into the fcb area (called HEADER in this example), and also to set the extent,current record and overflow bytes to zero (ie CHR(0)). In my version of fcbset the function returns a value equal to the address of the fcb used but the value is ignored at present.

My versions of the random access routines (actually they are functions), use the fact that there is more than one fcb created, and calls this the channel number. The channel number is assigned by the programmer and all the function calls need this value (or else the wrong file would be used with unpredictable results). The routines are rather similar so I won't list them all (anyway I am not getting paid for this by the inch, and I earn my living as a professional programmer so I would be silly giving all my ideas away!!).

```
FUNCTION RDRAND
    (channel,bufad,recnum:INTEGER):INTEGER;
VAR i : INTEGER;
BEGIN
   i := CPM(26,bufad); (* set DMA *)
   HEADER[channel].RNDREC := recnum;
   i := ADDR(HEADER[channel]);
   RDRAND := CPM(33,i)
END; (* RDRAND  read random *)
```

Call the function with the channel number, the ADDRess of the buffer area to be used and the number of the record. The numbers are both integers as you see so this limits you to records in the range of 0 to 32767. As the max number of a record in a CP/M file is 65535 (ie 8 Megabytes) I leave it as an exercise to any user to work out a way of getting the top half of such a large file (or for a small fee..?). Since I have mentioned fees, for a small fee I might supply all these routines (and the ones which are not given here) on a disk, but this letter should have given enough ideas to get anyone going in the fascinating jungle of random-access files.

Yours sincerely, Godfrey Nix, 11 Whitechapel Street, Nottingham.

BIOS in EPROM

In an issue some time ago Richard Beal was discussing the considerable hassle of patching a new BIOS into the MOVCPM.COM file for relocating CP/M.

I have just finished my own custom BIOS (for a Nascom 2 and two 8" Shugart drives) and have adopted another system for relocating CP/M which does not involve relocating the BIOS.

My BIOS sits in EPROMs; in a 56k system these start from E000 where it is entered by a reset jump. It contains its own routines for loading the CCP and BDOS from disk, to the desired place in RAM. A `56k' CCP & BDOS would be loaded near the top of RAM, whereas a `32k' CCP and BDOS would be loaded about half-way up. Having done this, the BIOS then initialises the jump table just after the BDOS and transfers control ,to the CCP. This has the advantages that the BIOS can be in EPROMs since it is always in the same place, one can use anybody's system disk to run in your system, regardless of which BIOS is on the disk, there is no restriction on the size of the bootstrap loader because there isn't one (normally it has to fit on the first sector of the disk and has to be short) and full disk read error recovery routines can be implemented in it. One therefore has a better chance of loading the system tracks off a slightly suspect disk; this is important since without disks one cannot do anything except some trivial debugging using SIMON, or reverting the whole Nascom back to NAS-SYS3 and ZEAP and writing some test routines to find what is wrong. I have found no need for SIMON and a 6k BIOS size limit is handy if you want to modify CONOUT to drive one of the flashy graphics/alphanumeric cards; some need a lot of software for writing text.

I have used the standard MOVCPM to produce a `56k' system (with the MDS-800 BIOS on it which is not used) and my BIOS starts at E000. It can extend up to F800, a total of 6k which is enough for most requirements. Even then there is a big gap for stack/data and with a slightly smaller BIOS (same size as Mr. Beal's I think) a 60k system can be configured. With non-Nascom hardware a 64k system is possible, using the IVC card.

The only disadvantages I can think of is that EPROMs are bit more difficult to patch that the MOVCPM is when in RAM.

I feel that people who implement CP/M on their own hardware will be interested in this approach, since they will simply buy a CP/M 2.2 disk in their chosen format, configure it for the biggest RAM they can get and are not trying to produce a licensed and commercially saleable system.

Incidentally, I have been told by Digital Research that the [v] option should be `avoided' when copying large files. It does indeed produce spurious `Verify Error' messages and aborts, but my disk read/write routines do not report any errors. The appearance of this problem is consistent for a given file. Does anyone have any clues why this should be ?

Yours sincerely, P. Holy, Worthing.

Private Wants

WANTED: Processor, driver and power supply printed circuit boards for Epson MX80 or MX100 printer, either working or not working but must be mechanically undamaged to facilitate rebuild of damaged printer. Telephone (0742) 460609.

WANTED: Does anyone have any information on IBM 3270 interfacing/operation, i.e. manuals etc., OR would like said machine cheap with spares. Call Ian, Ipswich (0473) 831535.

DETERMINING THE NASCOM KEYBOARD STATUS

By Geoff Higgs

When Nas-Sys scans the keyboard it stores the state of all the keys in 9 "KMAP" positions, known as KMAP0 to KMAP8, at locations 0C01 to 0C09 hex, 3073 to 3081 decimal. These are updated every time the keyboard is scanned.

The chart shows the Nascom 2 keyboard as layed out. Beneath the legend for each key is the address and below that it's contents after a keyboard scan when that key is pressed. This is shown in both Hex and decimal notation. The contents remain the same on repeated scans until the key is released. Since each key is bit-mapped it can be detected irrespective of how many keys are simultaneously pressed. When several keys sharing the same map address are pressed, the content is the sum of the values for all the keys pressed.

Note that SHIFT does not change the contents for any key but only puts 10 (hex), 16 (dec) in KMAP0. Similarly GRAPH and CTRL are mapped as any other key.

When key presses are required to control features of programmes, the use of this table avoids involvement with repeat keyboard routines and their associated adjustable delays.

Example:

Assembly

```
SCANKB EQU 62H
KMAP0  EQU 0C01H

TESTKY LD HL,KMAP0+2
       SCAL SCANKB
       BIT 3,(HL)     ; "D" pressed, other keys "don't care
       JR Z,RTN1
       INC HL
       INC HL
       LD A,4
       CP (HL)        ; "8" pressed but no others using 0C05H
       JR Z,RTN2      ; or CALL Z
       JR TESTKY      ; or RET
```

Basic

```
10 K=USR(0):REM Scan keyboard user routine
20 IF PEEK(3075) AND 8=8 THEN 100:REM Go to routine 1
30 IF PEEK(3077)=4 THEN 200:REM or GOSUB
40 GOTO 10:REM or RETURN
```

KMAP0 is "duplicated" as KMAP8 at 0C09 hex (3081 decimal) and properly should be used instead. In practice I have never found any difficulty either way.

NASCOM KEYBOARD MAP CHART

1	2	3	4	5	6	7	8	9	0	-	[	]
0C07	0C07	0C06	0C08	0C02	0C03	0C04	0C05	0C06	0C07	0C01	0C07	0C08
10	08	08	04	04	04	04	04	04	04	04	40	40
3079	3079	3078	3080	3074	3075	3076	3077	3078	3079	3073	3079	3080
16	8	8	4	4	4	4	4	4	4	4	64	64

GRA	Q	W	E	R	T	Y	U	I	O	P	@	BS
0C06	0C06	0C05	0C04	0C08	0C02	0C03	0C04	0C05	0C06	0C07	0C01	0C01
40	10	08	08	20	20	20	20	20	20	20	20	01
3078	3078	3077	3076	3080	3074	3075	3076	3077	3078	3079	3073	3073
64	16	8	8	32	32	32	32	32	32	32	32	1

CTL	A	S	D	F	G	H	J	K	L	;	:	ENT	CH
0C01	0C05	0C04	0C03	0C02	0C08	0C02	0C03	0C04	0C05	0C06	0C07	0C01	0C01
08	10	10	08	08	01	01	01	01	01	01	01	02	40
3073	3077	3076	3075	3074	3080	3074	3075	3076	3077	3078	3079	3073	3073
8	16	16	8	8	1	1	1	1	1	1	1	2	64

SHF	Z	X	C	V	B	N	M	,	.	/	SHF
0C01	0C03	0C02	0C08	0C08	0C02	0C03	0C04	0C05	0C06	0C07	0C01
10	10	10	08	02	02	02	02	02	02	02	10
3073	3075	3074	3080	3080	3074	3075	3076	3077	3078	3079	3073
16	16	16	8	2	2	2	2	2	2	2	16

CL	CU	-------------- SPACE ------------	CD	CR
0C03	0C02	0C08	0C04	0C05
40	40	10	40	40
3075	3074	3080	3076	3077
64	64	16	64	64

Addresses	/ Hex		0C01	0C02	0C03	0C04	0C05	0C06	0C07	0C08
	\ Dec		3073	3074	3075	3076	3077	3078	3079	3080
Contents										
bit	Hex	Dec								
0	01	1	BS	H	J	K	L	;	:	G
1	02	2	ENT	B	N	M	,	.	/	V
2	04	4	-	5	6	7	8	9	0	4
3	08	8	CTL	F	D	E	W	3	2	C
4	10	16	SHF	X	Z	S	A	Q	1	SPC
5	20	32	@	T	Y	U	I	O	P	R
6	40	64	CH	CU	CL	CD	CR	GRA	[	]

GIANT INTELLIGENT PRINT BUFFER FOR GEMINI CPU CARDS By Richard Beal

This article gives you all the information and software which you need to set up a print buffer for a serial printer, using a Gemini GM813 CPU+RAM card with no other cards on its 80-BUS, or alternatively a Gemini GM811 CPU plus GM802 64K RAM combination. The print output, in a form suitable for a Centronics printer, is sent from the PIO of the host computer via the GIPB to the serial printer.

A large print buffer allows you to keep using your computer even when you have generated a very long printed report such as a program listing, without having to wait for the printer. This Giant Intelligent Print Buffer (GIPB) operates almost as fast as you can send data to it. For example when listing data to the screen using an SVC, which is very fast, there is no noticeable slowing up when sending the data to the GIPB at the same time.

You may like to develop the idea further, so here are some suggestions:-

- write a version which runs under a normal RP/M or CP/M;
- allow the display of characters in the buffer;
- allow buffering of the characters to an attached disk;
- develop a full automatic print spooling system;
- write a version with serial input and Centronics output.

The User Manual for the GIPB – Version 2.5

This program, called GIPB, is a special version of RP/M designed to perform the specific function of acting as a giant intelligent print buffer. Hardware requirements are:-

(a) a GM813 CPU+RAM card or a GM811 CPU card with extra 64K RAM card.
(b) a serial printer for output.
(c) a cable connecting the PIO socket to the PIO socket of another computer which is set up to output data to a Centronics printer. If the other computer if a GM811 or GM813, or a Nascom I/O card, a 26 way ribbon cable with a connector at each end is all that is needed.
(d) an optional serial keyboard on the printer, or a keyboard on the GM811.

No disk card or video card is required. Since there will normally be no video card the printer also acts as the console output device. See the RP/M documentation for details of operation without a video card. On the GM813 it is simply a matter of linking pin 1 to pin 14 on the link block labelled IC35. On the GM811 connect pin 6 to pin 7 on LKB1.

The card(s) may easily be added to an existing 80-BUS system by plugging it (them) in to the last connector(s) on the BUS. Since this would interfere with the BUS signals, cut all the lines on the motherboard except the power lines, which are 1 to 4 and 67 to 78.

As with RP/M, the UART speed for the printer may be changed by altering location F009 in the EPROM to hold the 2 byte UART divisor. This is normally 417 decimal, stored as 01 A1, giving 300 bps. Printer handshaking is supported in the normal way, if required. This is via pin 8 of the serial connector, which must be high to operate. Connect it to pin 2 if you have no handshake line.

The ports are used in a way compatible with the Gemini implementation of the Centronics interface, as follows:-

Port A is used in control mode.

Bit 0 is an output signal from the GIPB and is high when Busy and low when able to accept data.

Bit 1 is an input to the GIPB and is a strobe which goes low for a short time when data has been sent to port B.

Port B is used in control mode, as the GIPB input port. Bit 7 of the input data is ignored, and the output to the printer has even parity added to follow the normal standards.

Operation of the system is completely automatic, and all data received is printed as soon as possible. The program uses a circular buffer and compresses consecutive spaces to save memory. Up to 128 consecutive spaces can be held in one byte. Most listings contain many spaces, so the buffer will often be able to hold well over 100K in the 60K available. If the buffer becomes full the Busy line remains high so that no data can be lost.

If a keyboard is attached the following single character commands are available:-

Space — Halt the printer, or if halted start printing again. This does not affect the input of data to the buffer.

D — Delete the contents of the buffer and restart the program.

T — Delete the contents of the buffer and restart the program with a minimal buffer of only two bytes.

CR — Output CR/LF to the console device (normally the printer).

M — Output a status message to the console device. This shows the number of characters waiting to be printed, the number of bytes spare in the buffer, and whether or not the printer has been halted.

N — End the program and pass control to RP/M. RP/M operates as normal and can boot a disk system but does not have any cassette handling routines. The command G F000 will execute the program from RP/M.

! — Halt the processor.

Technical Notes

The GIPB operates on an interrupt driven basis, with an interrupt being generated when the input strobe goes high rather than low. It was necessary to do it this way because some host software does not initialise the ports correctly so that the first character is lost following a Reset. This method overcomes this problem and should not cause any problems. Some host software will send a null during initialisation. This is sent to the printer which is likely to ignore it.

The GIPB catches all characters transmitted by enabling interrupts and then setting the Busy line to 0. After about 8 instructions the Busy line is set back to 1. This should give the host machine plenty of time to notice that the line is not Busy, and decide to output the data. The GIPB waits for about 40 instructions after it sets the Busy line back to 1 before it disables interrupts. This gives the host machine more than enough time to send the data and make the strobe go low and then high again. If an interrupt occurs the Busy line is at once set back to 1 to ensure that a second character is not sent. This system should work correctly, although it would in theory be possible for someone to write a Centronics output routine which is so slow to send the strobe after examining the Busy line that the data is held until the interrupts are next enabled. This could in theory cause loss of characters. All known versions of the BIOS for Gemini systems, as well as RP/M itself, work perfectly as host machines.

The GIPB accepts input both when it is inactive, and when it is waiting for the handshake signal or the UART status to become ready during printing.

The GIPB adjusts to the size of memory available, so that in fact only 2K of RAM at the start of memory is needed, although this would be of limited use. The reason that 64K is normally needed is that there must be 4K of RAM at the top of memory, occupying the same addresses as the EPROM. This is because the EPROM has to be paged out during use of the GIPB, and it copies itself to the same area in RAM. This is necessary because of a hardware feature of the card which prevents the PIO receiving the RETI instruction from code in the EPROM. Therefore the PIO can handle only one interrupt and then locks up.

An alternate version of the GIPB operates using only port B, with the Ready and Strobe lines for handshaking. This requires special interrupt handling software at the host end.

The Listings for GIPB

Below I have given the complete code needed. It is necessary only to create a 2732 EPROM and plug this into a GM813. The listing is shown in two halves, for convenience, and the CRCs for the two halves have been calculated separately, to make checking simpler.

CRC for first half: CRC = 6B A6
CRC for second half: CRC = A7 42

I have also given the source code of the routine which does the GIPB operation. It is self contained, and as you can see could easily be adapted to operation under any other operating system.

```
First half of the GIPB.
Record 0.
0100: 00:  C3 71 F0 C3 5C F1 C3 93  F1 A1 01 01 42 C3 00 F0  CqpC\qC. q!..BC.p
0110: 10:  C3 03 F0 C3 09 FC C3 24  FC C3 DE FC C3 F6 FC C3  C.pC.|C$ |C^|Cv|C
0120: 20:  0F FD C3 13 FD 21 83 FF  18 1C 21 86 FF 18 17 21  .}C.}!.. ..!....!
0130: 30:  89 FF 18 12 21 8C FF 18  0D 21 8F FF 18 08 21 92  ....!... .!....!.
0140: 40:  FF 18 03 21 95 FF ED 5B  4E 00 19 E9 20 62 79 74  ...!..m[ N..i byt
0150: 50:  65 73 20 2D 20 52 50 2F  4D 20 66 6F 72 20 47 65  es - RP/ M for Ge
0160: 60:  6D 69 6E 69 20 56 32 2E  35 20 2D 20 47 49 50 42  mini V2. 5 - GIPB
0170: 70:  24 16 64 01 FE F0 1E 0F  ED 59 1D 78 D6 10 47 30  $.d.~p.. mY.xV.G0
Record 1.
0180: 00:  F7 15 20 EF 3E 11 D3 FF  D3 B3 21 00 00 36 00 11  w. o>.S. S3!..6..
0190: 10:  01 00 01 FF 00 ED B0 21  00 00 7E 2F 77 BE 20 12  .....m0! ..~/w> .
01A0: 20:  2F 77 23 11 0C 00 1A 3C  12 FE 0A 20 ED AF 12 1B  /w£....< .~. m/..
01B0: 30:  18 F4 22 4E 00 11 C0 FF  19 F9 11 40 FF 19 3E C3  .t"N..@. .y.@..>C
01C0: 40:  32 05 00 22 06 00 77 23  11 06 F0 73 23 72 23 22  2.."..w£ ..ps£r£"
01D0: 50:  46 00 2A 4E 00 11 83 FF  19 32 00 00 22 01 00 EB  F.*N.... .2.."..k
01E0: 60:  1B 1B 1B 21 0D F0 01 18  00 ED B0 32 38 00 21 2A  ...!.p.. .m028.!*
01F0: 70:  FB 22 39 00 3A 0B F0 21  03 00 77 DB BE CB 77 28  {"9.:.p! ..w[>Kw(
Record 2.
0200: 00:  04 7E EE 01 77 01 FE 00  3E 0F ED 79 7E FE C3 ED  .~n.w.~. >.my~~Cm
0210: 10:  41 20 02 CB CE 21 00 02  22 4C 00 3A 0C F0 32 42  A .KN!.. "L.:.p2B
0220: 20:  00 DB B1 CD FB FD 2A 09  F0 22 3B 00 CD BD FE 3A  .[1M{}*. p";.M=~:
0230: 30:  1D FF FE FF C4 1D FF CD  26 F4 CD 62 FA 21 08 00  ..~.D..M &tMbz!..
0240: 40:  06 05 7E C6 30 5F E5 C5  CD 69 FA C1 E1 23 10 F2  ..~F0_eE MizAa£.r
0250: 50:  11 4C F0 CD 5D FA CD 3A  FB CC 49 FB 2A 4E 00 11  .LpM]zM: {LI{*N..
0260: 60:  C0 FF 19 F9 21 80 00 22  4A 00 AF 32 41 00 32 45  @..y!.." J./2A.2E
0270: 70:  00 32 53 00 3E FF D3 B6  3E FD D3 B6 3E FF D3 B7  .2S.>.S6 >}S6>.S7
Record 3.
0280: 00:  AF D3 B7 D3 B5 2F D3 B4  3A 20 FF FE FF C4 20 FF  /S7S5/S4 : .~.D .
0290: 10:  C3 70 F6 ED 53 57 00 21  00 00 22 59 00 ED 73 55  CpvmSW.! .."Y.msU
02A0: 20:  00 ED 7B 4E 00 21 F4 F1  E5 79 FE 1B D0 4B 21 BE  .m{N.!tq ey~.PK!>
02B0: 30:  F1 5F 16 00 19 19 5E 23  56 2A 57 00 EB E9 25 F0  q_....^£ V*W.ki%p
02C0: 40:  B1 F3 8E F2 B6 F3 3E F0  39 F0 BB F3 D2 F3 D7 F3  1s.r6s>p 9p;sRsWs
02D0: 50:  DC F3 DB F2 E2 F3 FD F1  E9 F3 FD F1 E9 F3 E9 F3  \s[rbs}q is}qisis
02E0: 60:  FD F1 FD F1 FD F1 E9 F3  E9 F3 E9 F3 FD F1 FD F1  }q}q}qis isis}q}q
02F0: 70:  FD F1 E9 F3 ED 7B 55 00  2A 59 00 7D 44 C9 21 54  }qism{U. *Y.}DI!T
```

```
Record 4.
0300: 00:   00 7E 36 00 B7 C0 C3 2F  F0 CD FE F1 CD 17 F2 D8  .~6.7@C/ pM~qM.rX
0310: 10:   F5 4F CD 8E F2 F1 C9 FE  0D C8 FE 0A C8 FE 09 C8  uOM.rqI~ .H~.H~.H
0320: 20:   FE 08 C8 FE 20 C9 3A 54  00 B7 20 1A CD 2A F0 E6  ~.H~ I:T .7 .M*pf
0330: 30:   01 C8 CD 2F F0 FE 13 20  0A CD 2F F0 FE 03 CA 00  .HM/p~.  .M/p~.J.
0340: 40:   00 AF C9 32 54 00 3E 01  C9 3A 50 00 B7 20 13 C5  ./I2T.>. I:P.7 .E
0350: 50:   CD 26 F2 C1 C5 CD 34 F0  C1 C5 3A 53 00 B7 C4 39  M&rAEM4p AE:S.7D9
0360: 60:   F0 C1 79 21 52 00 FE 7F  C8 34 FE 20 D0 35 7E B7  pAy!R.~. H4~ P5~7
0370: 70:   C8 79 FE 08 20 02 35 C9  FE 0D C0 36 00 C9 79 CD  Hy~. .5I ~.@6.IyM
Record 5.
0380: 00:   17 F2 30 0A F5 0E 5E CD  49 F2 F1 F6 40 4F 79 FE  .r0.u.^M Irqv@Oy~
0390: 10:   09 20 B6 0E 20 CD 49 F2  3A 52 00 E6 07 20 F4 C9  . 6. MIr :R.f. tI
03A0: 20:   CD A8 F2 0E 20 CD 34 F0  0E 08 C3 34 F0 0E 23 CD  M(r. M4p ..C4p.£M
03B0: 30:   49 F2 CD C4 F2 3A 52 00  21 51 00 BE D0 0E 20 CD  IrMDr:R. !Q.>P. M
03C0: 40:   49 F2 18 F1 0E 0D CD 49  F2 0E 0A C3 49 F2 0A FE  Ir.q..MI r..CIr.~
03D0: 50:   24 C8 03 C5 4F CD 8E F2  C1 18 F3 3A 52 00 32 51  $H.EOM.r A.s:R.2Q
03E0: 60:   00 2A 57 00 4E 23 E5 06  00 C5 E5 CD FE F1 E6 7F  .*W.N£e. .EeM~qf.
03F0: 70:   E1 C1 FE 0D CA AA F3 FE  0A CA AA F3 FE 08 20 0D  aA~.J*s~ .J*s~. .
Record 6.
0400: 00:   78 B7 28 E5 05 3A 52 00  32 50 00 18 50 FE 7F 20  x7(e.:R. 2P..P~.
0410: 10:   0A 78 B7 28 D4 7E 05 2B  C3 93 F3 FE 05 20 0B C5  .x7(T~.+ C.s~. .E
0420: 20:   E5 CD C4 F2 AF 32 51 00  18 C1 FE 10 20 0B E5 21  eMDr/2Q. .A~. .e!
0430: 30:   53 00 3E 01 96 77 E1 18  B0 FE 18 20 11 E1 3A 51  S.>..wa. 0~. .a:Q
0440: 40:   00 21 52 00 BE D2 DB F2  35 CD A0 F2 18 F0 FE 15  .!R.>R[r 5M r.p~.
0450: 50:   20 07 CD AD F2 E1 C3 DB  F2 FE 12 20 33 C5 CD AD   .M-raC[ r~. 3EM-
0460: 60:   F2 C1 E1 E5 C5 78 B7 28  0C 23 4E 05 C5 E5 CD 7E  rAaeEx7( .£N.EeM~
0470: 70:   F2 E1 C1 18 F0 E5 3A 50  00 B7 CA EB F2 21 52 00  raA.pe:P .7Jkr!R.
Record 7.
0480: 00:   96 32 50 00 CD A0 F2 21  50 00 35 20 F7 C3 EB F2  .2P.M r! P.5 wCkr
0490: 10:   23 77 04 C5 E5 4F CD 7E  F2 E1 C1 7E FE 03 78 20  £w.EeOM~ raA~~.x
04A0: 20:   05 FE 01 CA 00 00 B9 DA  E9 F2 E1 70 0E 0D C3 49  .~.J..9Z irap..CI
04B0: 30:   F2 CD 09 F2 18 2F CD 43  F0 18 2A 79 3C 28 07 3C  rM.r./MC p.*y<(.<
04C0: 40:   CA 2A F0 C3 34 F0 CD 2A  F0 B7 CA F4 F1 CD 2F F0  J*pC4pM* p7JtqM/p
04D0: 50:   18 13 3A 03 00 18 0E 21  03 00 71 C9 EB 4D 44 C3  ..:....! ..qIkMDC
04E0: 60:   CE F2 CD 26 F2 32 59 00  C9 AF C3 E5 F3 0D 0A 2A  NrM&r2Y. I/Ces..*
04F0: 70:   20 47 49 50 42 20 2A 20  20 20 20 20 20 20 30 20   GIPB *        0
Record 8.
0500: 00:   77 61 69 74 69 6E 67 20  20 20 20 20 20 30 20 73  waiting        0 s
0510: 10:   70 61 72 65 20 20 20 20  20 20 20 20 20 0D 0A 24  pare            ..$
0520: 20:   48 61 6C 74 65 64 21 00  F0 54 5D 01 00 10 ED B0  Halted!. pT]...m0
0530: 30:   3E 0F D3 BC 21 69 F5 22  60 01 F3 3E 03 D3 B7 D3  >.S<!iu" `.s>.S7S
0540: 40:   B6 21 47 F4 E5 ED 4D 3E  01 ED 47 ED 5E 3E 60 D3  6!GtemM> .mGm^>`S
0550: 50:   B6 3E CF D3 B7 3E FF D3  B7 3E CF D3 B6 3E FE D3  6>OS7>.S 7>OS6>~S
0560: 60:   B6 3E 01 D3 B4 3E B7 D3  B6 3E FD D3 B6 ED 4B 06  6>.S4>7S 6>}S6mK.
0570: 70:   00 C5 21 ED F3 11 00 01  01 33 00 ED B0 C1 21 00  .E!ms... .3.m0A!.
Record 9.
0580: 00:   02 E5 CD 9D F5 23 B7 ED  42 09 20 F6 E1 50 59 1B  .eM.u£7m B. vaPY.
0590: 10:   AF 12 E5 D5 C5 1E FF 0E  06 CD 05 00 C1 D1 E1 B7  /.eUE... .M..AQa7
05A0: 20:   28 5F FE 61 38 02 E6 5F  FE 44 28 C1 FE 54 20 05  (_~a8.f_ ~D(A~T .
05B0: 30:   01 02 02 18 BC FE 4E 20  05 3E 07 D3 BC C9 FE 0D  ....<~N  .>.S<I~.
05C0: 40:   20 08 E5 D5 C5 11 1D F4  18 0A FE 4D 20 0D E5 D5   .eUE..t ..~M .eU
05D0: 50:   C5 11 00 01 0E 09 CD 05  00 18 1A FE 20 20 1B E5  E.....M. ...~  .e
05E0: 60:   D5 C5 11 2A 01 01 06 00  1A FE 20 21 20 F4 28 03  UE.*.... .~ ! t(.
05F0: 70:   21 17 F4 ED B0 C1 D1 E1  18 98 FE 21 20 01 76 18  !.tm0AQa ..~! .v.
Record 10.
0600: 00:   91 3A 10 01 FE 20 20 30  3A 11 01 FE 30 20 29 AF  .:..~  0 :..~0 )/
0610: 10:   F5 3A 1F 01 FE 20 20 07  3A 20 01 FE 30 28 13 FB  u:..~  . : .~0(.{
0620: 20:   AF D3 B4 3E 03 3D 20 FD  3E 01 D3 B4 3E 14 3D 20  /S4>.= } >.S4>.=
0630: 30:   FD F3 F1 3D 20 DA 18 C0  3A 2A 01 FE 20 20 D0 7E  }sq= Z.@ :*.~  P~
0640: 40:   CB 7F 28 0C FE 80 28 05  35 3E 20 18 12 3E 20 77  K.(.~.(. 5> ..> w
0650: 50:   F5 CD 9D F5 F1 23 B7 ED  42 09 20 03 21 00 02 F5  uM.uq£7m B. .!..u
0660: 60:   CD BD F5 F1 CD EB F5 18  8F F5 E5 3E 01 D3 B4 DB  M=uqMku. .ue>.S4[
0670: 70:   B5 E6 7F FE 20 20 0E 1A  CB 7F 28 07 FE FF 28 03  5f.~  .. K.(.~.(.
Record 11.
0680: 00:   3C 18 12 3E 80 F5 CD E5  F5 F1 13 62 6B B7 ED 42  <..>.uMe uq.bk7mB
0690: 10:   20 03 11 00 02 12 CD B7  F5 E1 F1 ED 4D E5 21 20   .....M7 uaqmMe!
06A0: 20:   01 7E FE 20 20 04 36 31  E1 C9 FE 39 20 05 36 30  .~~  .61 aI~9 .60
06B0: 30:   2B 18 EE 3C 77 E1 C9 E5  21 11 01 18 E4 E5 21 11  +.n<waIe !...de!.
06C0: 40:   01 E5 7E FE 30 20 05 36  39 2B 18 F6 3D 77 FE 30  .e~~0 .6 9+.v=w~0
06D0: 50:   20 10 2B 3E 20 BE 20 0A  23 77 E1 BE 20 05 36 30   .+> > . £wa> .60
06E0: 60:   E1 C9 E1 E1 C9 E5 21 20  01 18 D6 B7 F5 EA F2 F5  aIaaIe!  ..V7ujru
06F0: 70:   EE 80 F5 3A 1F 01 FE 20  20 07 3A 20 01 FE 30 28  n.u:..~   .: .~0(
Record 12.
0700: 00:   12 FB AF D3 B4 3E 03 3D  20 FD 3E 01 D3 B4 3E 14  .{/S4>.=  }>.S4>.
0710: 10:   3D 20 FD DB BE CB 67 F3  28 D9 DB BD CB 6F 28 D3  = }[>Kgs (Y[=Ko(S
0720: 20:   F1 D3 B8 F1 C9 0D 0A 2A  2A 20 52 50 2F 4D 20 72  qS8qI..* * RP/M r
0730: 30:   65 61 64 79 20 2A 2A 24  57 68 61 74 3F 24 4E 6F  eady **$ What?$No
0740: 40:   20 73 75 63 68 20 63 6F  6D 6D 61 6E 64 24 54 6F   such co mmand$To
0750: 50:   6F 20 6D 61 6E 79 2F 66  65 77 20 76 61 6C 75 65  o many/f ew value
0760: 60:   73 24 0D 0A 2A 2A 20 54  72 61 70 20 61 74 20 24  s$..** T rap at $
0770: 70:   11 25 F6 CD 5D FA 1E 2A  CD 69 FA CD 87 FA CD 4E  .%vM]z.* MizM.zMN
Record 13.
0780: 00:   FA 28 F3 CD 7E FA 32 5C  00 13 21 76 F6 E5 FE 52  z(sM~z2\ ..!vve~R
0790: 10:   28 57 FE 57 28 53 FE 49  28 4F CD A3 FA 38 44 3A  (W~W(S~I (OM£z8D:
07A0: 20:   5C 00 FE 44 CA D3 F7 FE  53 CA 63 F8 FE 47 CA 02  \.~DJSw~ SJcx~GJ.
07B0: 30:   F9 FE 43 CA 20 F9 FE 46  CA 3D F9 FE 50 CA 5A F9  y~CJ y~F J=y~PJZy
07C0: 40:   FE 55 CA 9C F9 FE 4C CA  F4 F9 FE 4F CA 0F FA FE  ~UJ.y~LJ ty~OJ.z~
07D0: 50:   51 CA 24 FA FE 42 CA 39  FA 11 3E F6 18 08 11 4E  QJ$z~BJ9 z.>v...N
07E0: 60:   F6 18 03 11 38 F6 C3 5D  FA 21 5D 00 06 23 36 20  v...8vC] z!]..£6
07F0: 70:   23 10 FB AF 32 68 00 CD  4E FA 28 2F 06 09 21 5D  £.{/2h.M Nz(/..!]
Record 14.
0800: 00:   00 1A CD 7E FA FE 30 38  13 FE 5B 30 0F FE 3A 38  ..M~z~08 .~[0.~:8
0810: 10:   04 FE 41 38 07 77 13 23  10 E7 18 0F CD 4E FA 20  .~A8.w.£ .g..MNz
0820: 20:   0A 3A 5C 00 FE 49 CA B7  F7 18 0A 3A 5C 00 FE 49  .:\.~IJ7 w..:\.~I
0830: 30:   CA B7 F7 18 AE 3A 03 00  E6 01 28 A7 AF 32 7C 00  J7w..:.. f.(´/2|.
0840: 40:   11 5C 00 0E 0F CD 05 00  11 00 01 ED 53 69 00 0E  .\...M.. ...mSi..
0850: 50:   1A CD 05 00 3A 5C 00 FE  52 CA 88 F7 2A 3D 00 23  .M..:\.~ RJ.w*=.£
0860: 60:   22 65 00 2A 65 00 2B 7C  B5 28 44 22 65 00 11 5C  "e.*e.+| 5(D"e..\
0870: 70:   00 0E 15 CD 05 00 2A 69  00 11 80 00 19 22 69 00  ...M..*i ....."i.
Record 15.
0880: 00:   EB 0E 1A CD 05 00 18 DB  21 00 00 22 3D 00 11 5C  k..M...[ !.."=..\
0890: 10:   00 0E 14 CD 05 00 B7 20  16 2A 69 00 11 80 00 19  ...M..7  .*i.....
08A0: 20:   22 69 00 EB 0E 1A CD 05  00 2A 3D 00 23 18 DC 11  "i.k..M. .*=.£.\.
08B0: 30:   5C 00 0E 10 C3 05 00 21  80 00 35 23 54 5D 23 7E  \...C..! ..5£T]£~
08C0: 40:   CD 7E FA 12 13 23 B7 20  F6 21 70 F6 E3 21 00 01  M~z..£7  v!pvc!..
08D0: 50:   C3 17 F9 3A 60 00 FE 03  D2 DE F6 CD 1E FB FE 02  C.y:`.~. R^vM.{~.
08E0: 60:   30 05 11 80 00 18 06 B7  EB ED 52 EB 13 E5 D5 E5  0......7 kmRk.eUe
08F0: 70:   D5 CD 00 FB 06 02 CD 72  FA D1 E1 06 10 7E E5 D5  UM.{..Mr zQa..~eU
```

```
Second half of the GIPB.
Record 0.
0100: 00:  C5 CD 07 FB CD 6E FA C1  CD 59 F8 D1 E1 23 1B 7A  EM.{MnzA MYxQa£.z
0110: 10:  B3 28 02 10 E8 06 02 CD  72 FA D1 E1 06 10 7E E5  3(..h..M rzQa..~e
0120: 20:  D5 C5 FE 20 38 04 FE 7F  38 02 3E 2E 5F CD 69 FA  UE~ 8.~. 8.>._Miz
0130: 30:  C1 CD 59 F8 D1 E1 23 22  61 00 1B 7A B3 CA 62 FA  AMYxQa£" a..z3Jbz
0140: 40:  10 DC E5 D5 CD 62 FA 0E  0B CD 05 00 D1 E1 B7 28  .\eUMbz. .M..Qa7(
0150: 50:  9C 0E 01 CD 05 00 C3 62  FA 78 FE 09 C0 C5 CD 6E  ...M..Cb zx~.@EMn
0160: 60:  FA C1 C9 3A 60 00 FE 01  C2 DE F6 2A 61 00 22 61  zAI:`.~. B^v*a."a
0170: 70:  00 E5 CD 00 FB CD 6E FA  E1 7E E5 F5 CD 07 FB CD  .eM.{Mnz a~euM.{M
Record 1.
0180: 00:  6E FA F1 FE 20 38 10 FE  7F 30 0C F5 CD 7A FA F1  nzq~ 8.~ .0.uMzzq
0190: 10:  5F CD 69 FA CD 7A FA CD  62 FA 06 0B CD 72 FA CD  _MizMzzM bz..MrzM
01A0: 20:  87 FA E1 06 00 CD 4E FA  20 07 78 B7 20 C0 23 18  .za..MNz  .x7 @£.
01B0: 30:  BD 04 E5 CD C2 FA 7E B7  28 0D 23 23 7E B7 20 3B  =.eMBz~7 (.££~7 ;
01C0: 40:  2B 7E E1 77 23 18 DE E1  1A FE 2E 20 07 13 CD 4E  +~aw£.^a .~. ..MN
01D0: 50:  FA C8 18 28 FE 22 20 08  13 1A B7 28 1F 13 18 E3  zH.(~" . ..7(...c
01E0: 60:  FE 2D 20 04 2B 13 18 BD  FE 2F 20 10 13 CD C2 FA  ~- .+..= ~/ ..MBz
01F0: 70:  38 0A 7E B7 28 06 2A 5E  00 18 AA E1 CD E3 F6 C3  8.~7(.*^ ..*aMcvC
Record 2.
0200: 00:  6B F8 3A 60 00 FE 02 D2  DE F6 21 70 F6 E3 21 00  kx:`.~.R ^v!pvc!.
0210: 10:  01 B7 28 03 2A 61 00 E5  11 80 00 0E 1A C3 05 00  .7(.*a.e .....C..
0220: 20:  3A 60 00 FE 03 C2 DE F6  CD 1E FB B7 ED 52 19 30  :`.~.B^v M.{7mR.0
0230: 30:  09 0B EB 09 EB 09 03 ED  B8 C9 ED B0 C9 3A 60 00  ..k.k..m 8Im0I:`.
0240: 40:  FE 03 C2 DE F6 CD 1E FB  78 B7 C2 E3 F6 13 B7 ED  ~.B^vM.{ x7Bcv.7m
0250: 50:  52 19 C8 D2 E3 F6 71 23  18 F4 3A 60 00 FE 04 D2  R.HRcvq£ .t:`.~.R
0260: 60:  DE F6 CD 1E FB FE 03 28  03 01 00 60 FE 02 30 03  ^vM.{~.( ...`~.0.
0270: 70:  11 00 00 FE 01 30 03 21  00 C0 7A B7 C2 E3 F6 7B  ...~.0.! .@z7Bcv{
Record 3.
0280: 00:  FE 04 D2 E3 F6 3E 01 1C  1D 28 03 87 18 FA F6 10  ~.Rcv>.. .(...zv.
0290: 10:  D3 FF 11 00 01 ED B0 3E  11 D3 FF C9 3A 60 00 FE  S....m0> .S.I:`.~
02A0: 20:  01 DA DE F6 FE 03 D2 DE  F6 CD 1E FB 7C B7 C2 E3  .Z^v~.R^ vM.{|7Bc
02B0: 30:  F6 7D CD 7E FA FE 0C 20  0B 3A 60 00 FE 01 C2 DE  v}M~z~.  .:`.~.B^
02C0: 40:  F6 C3 B3 FE FE 0D C2 E3  F6 CD BD FE 3A 60 00 FE  vC3~~.Bc vM=~:`.~
02D0: 50:  01 C8 21 D3 FE 4E 23 46  23 78 B1 CA E3 F6 78 BA  .H!S~N£F £x1Jcvx:
02E0: 60:  20 04 79 BB 28 04 23 23  18 EB 4E 23 46 ED 43 3B   .y;(.££ .kN£FmC;
02F0: 70:  00 C3 BD FE 3A 60 00 FE  02 D2 DE F6 B7 28 06 2A  .C=~:`.~ .R^v7(.*
Record 4.
0300: 00:  61 00 22 3D 00 2A 3D 00  CD 00 FB CD 62 FA C9 3A  a."=.*=. M.{MbzI:
0310: 10:  60 00 FE 02 C2 DE F6 CD  1E FB 7A B7 C2 E3 F6 44  `.~.B^vM .{z7BcvD
0320: 20:  4D ED 59 C9 3A 60 00 FE  01 C2 DE F6 CD 1E FB 44  MmYI:`.~ .B^vM.{D
0330: 30:  4D ED 78 CD 07 FB C3 62  FA 3A 60 00 B7 20 06 21  MmxM.{Cb z:`.7 .!
0340: 40:  00 00 22 61 00 FE 02 D2  DE F6 C3 49 FB 13 1A FE  .."a.~.R ^vCI{..~
0350: 50:  20 28 FA FE 09 28 F6 FE  2C 28 F2 B7 C9 0E 09 CD   (z~.(v~ ,(r7I..M
0360: 60:  05 00 1E 0D CD 69 FA 1E  0A 0E 02 C3 05 00 1E 20  ....Miz. ...C...
0370: 70:  18 F7 C5 CD 6E FA C1 10  F9 C9 1E 22 18 EB FE 61  .wEMnzA. yI.".k~a
Record 5.
0380: 00:  D8 FE 7B D0 D6 20 C9 11  7F 00 3E 7E 12 0E 0A CD  X~{PV I. ..>~...M
0390: 10:  05 00 CD 62 FA 21 80 00  5E 16 00 19 23 36 00 11  ..Mbz!.. ^...£6..
03A0: 20:  81 00 C9 01 60 00 AF 02  CD C2 FA D8 7E B7 C8 23  ..I.`./. MBzX~7H£
03B0: 30:  03 7E 02 23 03 7E 02 21  60 00 34 7E FE 0B 38 E8  .~.£.~.! `.4~~.8h
03C0: 40:  37 C9 CD 4E FA 21 00 00  22 5E 00 AF 21 5D 00 77  7IMNz!.. "^./!].w
03D0: 50:  1A B7 C8 FE 20 C8 FE 09  C8 FE 2C C8 CD 7E FA D6  .7H~ H~. H~,HM~zV
03E0: 60:  30 D8 FE 0A 38 0B D6 07  FE 0A D8 FE 10 38 02 37  0X~.8.V. ~.X~.8.7
03F0: 70:  C9 13 34 23 ED 6F 23 ED  6F 2B 2B 28 D3 1B 37 C9  I.4£mo£m o++(S.7I
Record 6.
0400: 00:  7C E5 CD 07 FB E1 7D F5  1F 1F 1F 1F CD 10 FB F1  |eM.{a}u ....M.{q
0410: 10:  E6 0F C6 90 27 CE 40 27  5F 0E 02 C3 05 00 2A 61  f.F.'N@' _..C..*a
0420: 20:  00 ED 5B 63 00 ED 4B 65  00 C9 11 62 F6 0E 09 CD  .m[c.mKe .I.bv..M
0430: 30:  05 00 E1 2B CD 00 FB C3  00 00 3E 55 D3 E1 3D 20  ..a+M.{C ..>USa=
0440: 40:  FD 21 00 00 DB E1 FE 55  C9 CD 3A FB 11 DE FB 20  }!..[a~U IM:{.^{
0450: 50:  31 3E D0 CD D1 FB 3A 61  00 B7 20 01 3C D3 E4 3A  1>PMQ{:a .7 .<Sd:
0460: 60:  62 00 D3 E2 3E 5B CD D1  FB DB E0 47 0E 00 DB E0  b.Sb>[MQ {[`G..[`
0470: 70:  A8 B1 E6 02 4F DB E0 07  07 2F A1 20 08 2B 7C B5  (1f.O[`. ./! .+|5
Record 7.
0480: 00:  20 EC C3 5D FA 3E 0B CD  D1 FB 3E 88 D3 E0 21 80   1C]z>.M Q{>.S`!.
0490: 10:  00 0E E4 06 80 ED 78 28  FC DB E3 77 23 10 F6 ED  ..d..mx( |[cw£.vm
04A0: 20:  78 28 FC DB E3 FA 9F FB  DB E0 B7 11 E6 FB 20 D2  x(|[cz.{ [`7.f{ R
04B0: 30:  2A 80 00 11 47 47 ED 52  11 EF FB 20 C5 11 FA FB  *...GGmR .o{ E.z{
04C0: 40:  CD 82 FB 11 00 00 21 80  00 01 80 00 ED B0 C3 02  M.{...!. ....m0C.
04D0: 50:  00 D3 E0 3E 0A 3D 20 FD  DB E0 0F 38 FB C9 4E 6F  .S`>.= } [`.8{INo
04E0: 60:  20 64 69 73 6B 24 42 61  64 20 64 69 73 6B 24 57   disk$Ba d disk$W
04F0: 70:  72 6F 6E 67 20 64 69 73  6B 24 45 78 65 63 75 74  rong dis k$Execut
Record 8.
0500: 00:  69 6E 67 20 62 6F 6F 74  24 3A 45 00 B7 20 12 3A  ing boot $:E.7 .:
0510: 10:  40 00 B7 20 0C CD BC FD  38 02 AF C9 B7 C8 32 40  @.7 .M<} 8./I7H2@
0520: 20:  00 3E FF C9 3A 45 00 B7  28 19 2A 46 00 2B 5F 16  .>.I:E.7 (.*F.+_.
0530: 30:  00 19 7E FE 0D 20 07 AF  32 45 00 3E 0D C9 21 45  ..~~. ./ 2E.>.I!E
0540: 40:  00 34 C9 3A 40 00 B7 20  09 CD 86 FE CD CA FD CD  .4I:@.7  .M.~MJ}M
0550: 50:  AD FE 4F AF 32 40 00 79  21 03 00 CB 46 C8 21 4C  -~O/2@.y !..KFH!L
0560: 60:  00 BE C0 CD 8C FE CD CA  FD FE 03 20 0A 2A 46 00  .>@M.~MJ }~. .*F.
0570: 70:  36 03 23 36 0D 18 28 21  4D 00 BE 20 0B CD 55 FD  6.£6..(! M.> .MU}
Record 9.
0580: 00:  3E 0D 2A 46 00 77 18 17  CD 38 FE FE 0D 20 D7 CD  >.*F.w.. M8~~. WM
0590: 10:  7B FE ED 5B 46 00 CD 42  FE 12 13 FE 0D 20 F7 CD  {~m[F.MB ~..~. wM
05A0: 20:  AD FE 3E 01 32 45 00 2A  46 00 23 7E 2B FE 3E 20  -~>.2E.* F.£~+~>
05B0: 30:  0D 7E FE 41 38 08 FE 5B  30 04 3E 03 18 10 7E FE  .~~A8.~[ 0.>...~~
05C0: 40:  2A 28 09 FE 2D 28 05 FE  23 C2 24 FC 3E 02 32 45  *(.~-(.~ £B$|>.2E
05D0: 50:  00 3D CA 24 FC F5 3E 1D  CD 38 FE F1 18 F3 79 E5  .=J$|u>. M8~q.sye
05E0: 60:  21 03 00 CB 46 E1 28 1D  FE 0C 20 07 3E 0D CD F3  !..KFa(. ~. .>.Ms
05F0: 70:  FC 3E 0A C3 38 FE CD 19  FD CD 42 FD 79 21 03 00  |>.C8~M. }MB}y!..
Record 10.
0600: 00:  CB 7E C2 E5 FD CD B5 FD  79 CD 50 FD C3 AA FD 79  K~Be}M5} yMP}C*}y
0610: 10:  C3 AA FD CD DE FD 30 FB  C9 79 FE 0C C0 E1 22 43  C*}M^}0{ Iy~.@a"C
0620: 20:  00 3A 41 00 B7 20 07 3A  42 00 32 41 00 C9 0E 0D  .:A.7 .: B.2A.I..
0630: 30:  21 38 FD E5 2A 43 00 E9  0E 0A 21 21 FD E5 2A 43  !8}e*C.i ..!!}e*C
0640: 40:  00 E9 79 FE 0A C0 3A 41  00 B7 C8 3D 32 41 00 C9  .iy~.@:A .7H=2A.I
0650: 50:  B7 E8 EE 80 C9 CD 96 FD  06 50 3E 5F CD 9D FD 10  7hn.IM.} .P>_M.}.
0660: 60:  FB CD 96 FD CD 4A FE 21  00 00 CD 64 FE CD 7B FE  {M.}MJ~! ..Md~M{~
0670: 70:  CD 42 FE FE 0D 28 05 CD  9D FD 18 F4 CD 96 FD 24  MB~~.(.M .}.tM.}$
Record 11.
0680: 00:  7C FE 19 20 E5 2A 48 00  CD 64 FE 06 50 3E 7E CD  |~. e*H. Md~.P>~M
0690: 10:  9D FD 10 FB 18 00 3E 0D  CD 9D FD 3E 0A F5 C5 D5  .}.{..>. M.}>.uEU
06A0: 20:  E5 4F CD 39 F0 E1 D1 C1  F1 C9 F5 DB BD CB 6F 28  eOM9paQA qIu[=Ko(
06B0: 30:  FA F1 D3 B8 C9 DB BE CB  67 28 FA C9 CD FB FD D8  zqS8I[>K g(zIM{}X
06C0: 40:  CD 14 FE D8 CD DE FD CB  BF C9 CD BC FD 30 FB C9  M.~XM^}K ?IM<}0{I
06D0: 50:  CD D6 FD 30 FB C9 CD FB  FD D8 CD 14 FE D8 DB BD  MV}0{IM{ }XM.~X[=
06E0: 60:  1F D0 DB B8 C9 F5 DB B4  1F 38 FB F1 F5 D3 B5 00  .P[8Iu[4 .8{quS5.
06F0: 70:  3E FD D3 B4 00 3E FF D3  B4 F1 C9 3A 03 00 E6 02  >}S4.>.S 4qI:..f.
```

```
Record 12.
0700: 00:   C0 D5 3A 3F 00 5F DB B0  AB 28 07 AB 32 3F 00 07  @U:?._[0 +(.+2?..
0710: 10:   B7 1F D1 C9 3A 03 00 E6  01 C8 3E 1B CD 38 FE 3E  7.QI:..f .H>.M8~>
0720: 20:   6B CD 38 FE CD 42 FE B7  C8 3E 1B CD 38 FE 3E 4B  kM8~MB~7 H>.M8~>K
0730: 30:   CD 38 FE CD 42 FE 37 C9  F5 DB B2 1F 38 FB F1 D3  M8~MB~7I u[2.8{qS
0740: 40:   B1 C9 DB B2 17 38 FB DB  B1 C9 3E 1B CD 38 FE 3E  1I[2.8{[ 1I>.M8~>
0750: 50:   3F CD 38 FE CD 42 FE 32  49 00 CD 42 FE 32 48 00  ?M8~MB~2 I.MB~2H.
0760: 60:   CD 42 FE C9 3E 1B CD 38  FE 3E 3D CD 38 FE 7C C6  MB~I>.M8 ~>=M8~|F
0770: 70:   20 CD 38 FE 7D C6 20 CD  38 FE C9 3E 1B CD 38 FE   M8~}F M 8~I>.M8~
Record 13.
0780: 00:   3E 5A CD 38 FE C9 D5 11  08 48 18 04 D5 11 09 40  >ZM8~IU. .H..U..@
0790: 10:   F5 3A 03 00 E6 01 28 12  3E 1B CD 38 FE 3E 59 CD  u:..f.(. >.M8~>YM
07A0: 20:   38 FE 7A CD 38 FE 7B CD  38 FE F1 D1 C9 D5 11 08  8~zM8~{M 8~qQIU..
07B0: 30:   08 18 DD 3E 03 D3 BC 21  68 00 C3 C4 FE 3E 07 D3  ..]>.S<! h.CD~>.S
07C0: 40:   BC 2A 3B 00 3E 83 D3 BB  7C D3 B9 7D D3 B8 3E 03  <*;.>.S; |S9}S8>.
07D0: 50:   D3 BB C9 50 00 C4 09 75  00 83 06 10 01 70 04 34  S;IP.D.u .....p.4
07E0: 60:   01 A1 03 50 01 41 03 00  03 A1 01 00 06 D0 00 00  .!.P.A.. .!...P..
07F0: 70:   12 68 00 00 18 45 00 00  20 3F 00 00 24 34 00 00  .h...E..  ?..$4..
Record 14.
0800: 00:   36 23 00 00 48 1A 00 00  72 11 00 00 96 0D 00 2F  6£..H... r....../
0810: 10:   19 07 00 4F 38 03 00 0F  56 02 00 00 00 FF FF FF  ...08... V.......
0820: 20:   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ........ ........
0830: 30:   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ........ ........
0840: 40:   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ........ ........
0850: 50:   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ........ ........
0860: 60:   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ........ ........
0870: 70:   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ........ ........
Record 15.
0880: 00:   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ........ ........
0890: 10:   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ........ ........
08A0: 20:   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ........ ........
08B0: 30:   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ........ ........
08C0: 40:   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ........ ........
08D0: 50:   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ........ ........
08E0: 60:   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ........ ........
08F0: 70:   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ........ ........
```

The source code of the GIPB, as a routine to be linked into a program, follows.

```
        title CPMGG  Giant Intelligent Print Buffer (GIPB)
        .z80
; Written to enable a Gemini running under RP/M to operate as a
; giant intelligent print buffer.
; Input is via a PIO, interrupt driven.
; Output is to a printer on the serial port.
; Provision is made for the output to be controlled from a keyboard
; on the serial port.

; Written by Richard Beal in 1982, originally using only port B with
; strobe and ready lines, using interrupt handling on both machines.

; This version developed in 1985 to add the option to emulate a
; parallel printer using a Centronics interface, so that interrupt
; handling is no longer required on the host.

; Port B is the input data port in both versions.
; For the printer emulator version Port A is the control port.
; From the point of view of the host, bit 0 is an input printer Busy line
; with 1=Busy, 0=Ready.  Bit 1 is an output strobe which is made low
; for a short time to show when a character is ready to be printed.

false   equ 0
true    equ not false

; Select type of parallel interface
cent    equ true        ; Set true to emulate Centronics interface
                        ; Set false for original version

; GIPB routine
        global gipb

; Dummy routines
        global reset,open,close,read,write
        global make,setdma

; Set returned value (for dummy routines)
        external bret

; Addresses
jbdos   equ 0005h       ; RP/M entry point
abdos   equ 0006h       ; Address of top of memory+1
work    equ 0100h       ; Start of work area
inttab  equ 0160h       ; Interrupt address table
stk     equ 0200h       ; Start of buffer

; RP/M routines
conio   equ 6           ; Direct console I/O
prts    equ 9           ; Print string
```

```
; Characters
lf      equ 0ah         ; Line feed
cr      equ 0dh         ; Carriage return

; Ports
uartd   equ 0b8h        ; UART data port
uartm   equ uartd+4     ; Modem control
uarts   equ uartd+5     ; Line status
uarth   equ uartd+6     ; Modem status
pd1     equ 0b4h        ; PIO data port A
pc1     equ pd1+2       ; PIO control port A
pd2     equ 0b5h        ; PIO data port B
pc2     equ pd2+2       ; PIO control port B

; Dummy routines
reset:
open:
close:
read:
write:
make:
setdma:
        xor a
        jp bret

; Heading message and work area
head:   defb cr,lf,"* GIPB *       "
hw:     defb "0 waiting      "
hs:     defb "0 spare  "
hh:     defb "     "
hcr:    defb cr,lf,"$"
headl   equ $-head              ; Length of message
wait    equ work+hw-head
spare   equ work+hs-head
headh   equ work+hh-head
haltm:  defb "Halted"
gipb:

; Copy RP/M to RAM at same address to avoid hardware problem
; (RETI from ROM not received by PIO)
        ld hl,0f000h
        ld d,h
        ld e,l
        ld bc,1000h
        ldir

; Switch out the ROM
        ld a,0fh        ; RS232, no ROM
        out (uartm),a

; Write interrupt address table to work area
        ld hl,procl
        ld (inttab),hl

; Disable CPU interrupts
        di

; Disable PIO
        ld a,03h
        out (pc2),a     ; Port B disabled
        if cent
        out (pc1),a     ; Port A disabled
        endif

; Ensure PIO happy
        ld hl,eph
        push hl
        reti

; Load I register
eph:    ld a,high(inttab)
        ld i,a

; Interrupt mode
        im 2

; Interrupt vector
        ld a,low(inttab)
        if cent
        out (pc1),a     ; Port A
        else
        out (pc2),a     ; Port B
        endif

        if cent
; PIO port B to mode 3, control
        ld a,0cfh
        out (pc2),a
; Direction control word - all bits are input
        ld a,0ffh
        out (pc2),a
; PIO port A to mode 3, control
        ld a,0cfh
        out (pc1),a
; Direction control word - define bit 0 as output and rest as input
        ld a,0feh
        out (pc1),a
; Output value to data port A to show printer busy
        ld a,01h
        out (pd1),a
; Interrupt control word - enable interrupts for low to high, mask follows
        ld a,0b7h
        out (pc1),a
; Interrupt mask - only interrupt on input bit 1 (when it goes low)
        ld a,0fdh
        out (pc1),a
        else
```

```
; PIO port B to mode 1, input
        ld a,4fh
        out (pc2),a
; Interrupt control word - enable interrupts for port B
        ld a,87h
        out (pc2),a
; Dummy read to start handshake
        in a,(pd2)
        endif

; PIO is now ready

; BC is used to point to the end of the buffer+1
st2:    ld bc,(abdos)

; Set up work area
st3:    push bc
        ld hl,head
        ld de,work
        ld bc,headl
        ldir
        pop bc

; Set number of bytes spare
        ld hl,stk       ; Start of buffer
        push hl
siz:    call sparep
        inc hl
        or a
        sbc hl,bc
        add hl,bc
        jr nz,siz

; HL is used to point to character to be output
        pop hl          ; Set to start of buffer

; DE is used to point to position to store input
        ld d,b          ; Set to end of buffer
        ld e,c
        dec de
; Reset bit 7 to not represent compressed blanks
        xor a
        ld (de),a

; Scan for keyboard input
tin:    push hl
        push de
        push bc
        ld e,0ffh
        ld c,conio
        call jbdos
        pop bc
        pop de
        pop hl
        or a
        jr z,noin
        cp "a"          ; Accept lower case
        jr c,yin1
        and 5fh
; If "D" restart with empty buffer
yin1:   cp "D"
        jr z,st2
; If "T" restart with 2 byte buffer
        cp "T"
        jr nz,yin2
        ld bc,stk+2
        jr st3
; If "N" return to RP/M
yin2:   cp "N"
        jr nz,yin3
        ld a,07h        ; RS232, ROM
        out (uartm),a
        ret
; If CR output CR/LF to printer
yin3:   cp cr
        jr nz,yin4
        push hl
        push de
        push bc
        ld de,hcr
        jr yin5
; If "M" output message to printer
yin4:   cp "M"
        jr nz,yin6
        push hl
        push de
        push bc
        ld de,work
yin5:   ld c,prts
        call jbdos
        jr yine
; If " " flip "Halted"
yin6:   cp " "          ; Space entered
        jr nz,yin8
        push hl
        push de
        push bc
        ld de,headh     ; Move to work area
        ld bc,6
        ld a,(de)
        cp " "
        ld hl,haltm     ; Move "Halted"
        jr z,yin7
        ld hl,hh        ; Move spaces
yin7:   ldir
yine:   pop bc
        pop de
        pop hl
tinx:   jr tin
```

```
; If "!" halt for debug
yin8:   cp "!"
        jr nz,yin9
        halt
; If "R" handshake (not usually needed)
yin9:   if cent
        jr tin          ; Does not apply unless host uses interrupt lines
        else
        cp "R"
        jr nz,tin
        in a,(pd2)
        jr tinx
        endif

; No input from keyboard
; Test if chars are waiting
noin:   ld a,(wait-1)
        cp " "
        jr nz,n6
        ld a,(wait)
        cp "0"
        jr nz,n6
; None waiting, so test if some spare
n2:     xor a
n3:     push af
        ld a,(spare-1)
        cp " "
        jr nz,n4
        ld a,(spare)
        cp "0"
        jr z,n5
; Still some spare so allow PIO input
n4:     if cent
        ei              ; Enable interrupts
        xor a           ; Show printer not busy
        out (pd1),a
        ld a,3          ; Allow time for host to realise (assume tight loop)
n4a:    dec a
        jr nz,n4a
        ld a,01h        ; Show printer busy again
        out (pd1),a
        ld a,20         ; Allow plenty of time for host to send data
n4b:    dec a
        jr nz,n4b
        di              ; Disable interrupts
        else
        ei              ; Enable interrupts
        nop
        nop
        di              ; Disable interrupts
        endif
n5:     pop af
        dec a
        jr nz,n3        ; Loop
        jr tinx

; Some waiting, so test if halted
n6:     ld a,(headh)
        cp " "
        jr nz,n2

; Chars are waiting for output
        ld a,(hl)
; Decompress spaces
        bit 7,a
        jr z,p4
        cp 80h
        jr z,p3
        dec (hl)
        ld a," "
        jr p6
p3:     ld a," "
        ld (hl),a
; Add 1 to Spare
p4:     push af
        call sparep
        pop af
; Test for end of buffer
        inc hl
        or a
        sbc hl,bc
        add hl,bc
        jr nz,p6
        ld hl,stk
; Subtract 1 from Waiting
p6:     push af
        call waitm
        pop af

; Call output routine
        call out        ; Includes EI/DI
        jr tinx


; ***** INTERRUPT HANDLING INPUT ROUTINE *****
procl:  push af
        push hl
        if cent
        ld a,01h        ; Set port A bit 0 to show busy
        out (pd1),a
        endif
        in a,(pd2)      ; Get data
        and 07fh        ; Strip parity
        cp " "          ; Check for blank
        jr nz,pr5
; Compress blanks
        ld a,(de)       ; Character in buffer
        bit 7,a         ; Test for first blank
        jr z,pr4
        cp 0ffh         ; Test for too many blanks
        jr z,pr4
```

```
        inc a       ; Not too many so increment
        jr pr6
; Compress first blank
pr4:    ld a,80h
; Subtract 1 from Spare
pr5:    push af
        call sparem
        pop af
; Test for end of buffer
        inc de
        ld h,d
        ld l,e
        or a
        sbc hl,bc
        jr nz,pr6
        ld de,stk   ; Set to start
; Store character in buffer
pr6:    ld (de),a
; Add 1 to Waiting
pr8:    call waitp
        pop hl
        pop af      ; Do not enable interrupts
        reti


; ***** ARITHMETIC ROUTINES *****
; Add 1 to chars spare
sparep: push hl
        ld hl,spare
ascinc: ld a,(hl)
        cp " "
        jr nz,gotdig
        ld (hl),"1"
        pop hl
        ret
gotdig: cp "9"
        jr nz,not9
        ld (hl),"0"
        dec hl
        jr ascinc
not9:   inc a
        ld (hl),a
        pop hl
        ret

; Add 1 to chars waiting
waitp:  push hl
        ld hl,wait
        jr ascinc

; Subtract 1 from chars waiting
waitm:  push hl
        ld hl,wait
ascsub: push hl
ascdec: ld a,(hl)
        cp "0"
        jr nz,ntzero
        ld (hl),"9"
        dec hl
        jr ascdec
ntzero: dec a
        ld (hl),a
        cp "0"
        jr nz,wfin
        dec hl
        ld a," "
        cp (hl)
        jr nz,wfin
        inc hl
        ld (hl),a
        pop hl
        cp (hl)
        jr nz,wfin2
        ld (hl),"0"
        pop hl
        ret
wfin:   pop hl
wfin2:  pop hl
        ret

; Subtract 1 from chars spare
sparem: push hl
        ld hl,spare
        jr ascsub


; ***** OUTPUT ROUTINE *****
; Output character to printer
out:    or a
        push af
        jp pe,out0        ; Make parity even
        xor 80h
; Decide if interrupts enabled while waiting
out0:   push af           ; Low level I/O routine
out1:   ld a,(spare-1)    ; Test if any spare
        cp " "
        jr nz,out2
        ld a,(spare)
        cp "0"
        jr z,out4
out2:   ei                ; Enable interrupts
        if cent
        xor a             ; Show printer not busy
        out (pdl),a
        ld a,3            ; Allow time for host to realise
                                (assume tight loop)
out2a:  dec a
        jr nz,out2a
        ld a,01h          ; Show printer busy again
        out (pdl),a
        ld a,20           ; Allow plenty of time for host to send data
out2b:  dec a
        jr nz,out2b
        endif
out4:   in a,(uarth)      ; Test handshake
        bit 4,a           ; Test CTS
        di                ; Disable interrupts
        jr z,out1
        in a,(uarts)      ; See if free yet
        bit 5,a
        jr z,out1         ; Wait until free
        pop af
        out (uartd),a     ; Output data
        pop af
        ret

        end
```

GSX THE GRAPHICS INTERFACE

By Dave Russ

Who among you has suffered the trauma of having purchased at great expense a wonderful new colour card and then realised that you have weeks of work ahead of you creating some sort of software interface to your favourite language? As is often the case you will have to create a library of low level primitives armed only with boundless enthusiasm and a manual whose flashy cover does not reflect its contents. Fear not, for the cavalry is on its way. The Digital Research GSX graphics system will relieve you of this tiresome task, leaving you time to get on with what you originally had in mind (Maybe seeing your family and friends once in a while.)

GSX (Graphics System Extension) allows you to write application programs in any language that supports BDOS calls, and provides you with an environment that is independent of the device(s) that will eventually display the end product. Along with 2 other DR products, GSS Kernel and GSS Plot, you are able to program graphics applications that conform to the emerging Graphical Kernel system (GKS), a draft international graphics standard. Kernel and Plot are not essential to you, and you do not have to use them in order to produce graphical routines, but they do provide a friendlier interface to GSX giving you access to a standard library accessable from popular high level languages. DR have specified Pascal, Fortran and PL/I so far. The whole thing is similar in concept to the relationship between the BDOS and BIOS in that you have a standard interface to custom built device drivers.

Having decided that this is for you, off you trot to your friendly software dealer with your loot in your hand and swop it for the GSX80 (or GSX86) disk. For your money you will have received the following:

GSX.SYS - This is the actual GDOS that will load itself into memory and process all your graphics calls.

GENGRAF.COM - A utility program which is run against a graphics program once it has reached the .COM stage. GENGRAF attaches a GSX loader to your program. The GSX loader receives control as soon as the program is run, its purpose is to handle the loading of GSX.SYS, the rearrangement of the BDOS pointers, and then the loading of the assignment table ASSIGN.SYS (see below) along with the first device driver that is specified, which must also be the biggest. Once it has finished its work the loader brings down the application program from its position above the loader to the start of the TPA at 100H and executes it.

ASSIGN.SYS - This is an ASCII file containing the device driver numbers and names that you want to use in your particular system. As it is possible to have only one device driver in memory at any one time GSX has to refer to the table contained in ASSIGN.SYS in order to select new drivers when they are required by the system. The copy that you find on your master disk will contain the names of a few of the sample device drivers supplied to you on the disk, and as such will have to be altered to suite the drivers that you will be using.

A number of ready to go device drivers - This sounds good doesn't it? We have just bought the disk and we are off already. However, the bottom line here is that unless you own a Hewlett Packard 7220 plotter, a Digital Engineering Retro-Graphics colour monitor, or an Epson MX-80 with Graftrax plus, these drivers are not going to be of much use to you.

Referring to the last item, it seems that most members of the 80-BUS fraternity will have to stop and think at this point. "I now have GSX, but what about the device drivers for MY system?" Well you have a choice of two options, the first being to write your own device driver or secondly, wait for the one you want to be released.

Writing your own does seem to defeat the object of the exercise, doesn't it? However it is possible, you are able to implement as much or as little of the standard as you wish depending on the capabilities of the device. A word of warning here. The device driver specifications supplied with the GSX disk are attractivly bound and the contents well laid out, but trying to write a GSX device driver from the knowledge contained therein should not be attempted unless you are sure of your sanity and/or you have a hot line to Digital Research in Newbury. During the creation of the Gemini device driver for the Pluto board I have needed to refer to both the GSX80 and GSX86 manuals for information, the GSX86 one being by far the better of the two. Test software is yet another problem, as writing your own will not confirm that you have got it right. All testing for the Pluto driver so far has been done using the DR DRAW drawing package, a fine piece of software, but it will cost you £232 at current retail prices. The DR compiled BASIC, CBASIC, will also help you as it contains inbuilt commands that allow you access to GSX, and you will find yourself £393 the poorer for this experience. So in a nutshell, unless you are sure that you are committed to the subject it might be better if you waited for your device driver to appear on the scene.

But will they arrive? Well Gemini will soon be releasing a device driver for the popular Pluto board that will be configurable for the 640 and 768 versions in both high and low res. Input routines have been written for keyboard, digitiser and mouse, and seperate drivers may be supplied for each of these devices. They also have a driver for their GM837 colour card under development though no release date can be forecast for this just yet. As far as other devices are concerned, who knows, but I suppose if the demand is there others will appear.

So how does it work? Lets first take the source program that you yourself will write. You forget all about the target display machine and its limitations, frame size, aspect ratio etc, as GSX will sort all that out for you. This concept means that your program is capable of being displayed on any graphics device for which you have a GSX device driver. You have at your disposal up to 33 graphical routines depending on the particular device driver you have installed, these include old favourites such as line drawing and text display, along with extra goodies like complex polygon fills that will cater for a number of fill patterns. Each GSX function is invoked by a special call to the BDOS (115 in C register). All the data associated with a function call will have been stored in special arrays of your own creation and their start addresses passed across using the GDOS parameter passing conventions. (This is where GSX starts getting a little complex - but more on this later.)

The 32767 X 32767 virtual frame size means that you can afford to be lavish with your coordinates and even include some form of zoom feature in your emerging bijou of a program, providing that those around you don't take offence at the constant stream of expletives and apparent recurrence of brain death associated with such activities.

So you have typed in the source, and as usual it has compiled first time (bliss), you link and load it to produce the .COM version, nothing new so far. Now you enter GENGRAF <filename><RET> and the utility will attach the GSX loader to your program. Your graphics program is now ready to run.

When run, the GSX loader gets the first look in as previously mentioned, loads GSX.SYS to create the GDOS interface, loads the assignment table and the first named device driver contained therein. The space now occupied by the device driver is now referred to as the GIOS, which lives just below the BDOS and its workmate the BIOS. Refer to `Nuts and bolts´ for more detail. The application program is moved down to 100h and executed.

The first command of any program will be GSX opcode 1 `open workstation´. This will inform the GDOS which of the available device drivers is to be used. If it is already in memory, entry one of the GIOS is called. If another driver is specified, it will be loaded into the GIOS area from disk. It can now be seen why the first entry in ASSIGN.SYS must be the name of the biggest driver available to the system, as GSX determines the amount of memory to allocate the GIOS solely from inquiring the size of this first named driver. If a subsequently loaded driver is bigger than the allocated GIOS size confusion will follow.

`Open workstation´ calls the first entry in to the GIOS, and firstly informs the GIOS of any defaults that the application requires, such as line colour, marker type etc. More importantly though, this function returns to the GDOS information concerning the device that it is currently working with. On exit from open workstation the GDOS will have details contained in it on the exact capabilities of the device. These details include X and Y axis resolution, aspect ratios, no. of colours, available fonts, and more. In fact 57 16 bit values are returned to reflect the device specifications. Not only does the GDOS use this to prepare itself for the following commands, but this information is also available to the calling pogram if it needs it.

So they´re off!! Your much awaited graphics program will now spring into colourful life, and all the lines and circles etc whos coordinates you programmed inside the GSX 32k X 32k virtual frame size now appear on your screen or whatever, which may only be 640 X 288 for example. Whats more your circles are circular, because the GDOS has received information on the aspect ratio of your screen.

In the time taken for your display to plot, the GDOS has intercepted all calls to the BDOS in which the C register contains the value 115, any others it passes on to the BDOS as normal. The control array is interrogated to see if you wanted to open a new workstation, if so another device driver is loaded, if not all coordinates contained in the array PTSIN are scaled to device size and control is passed to the GIOS. So as you can see the job of the GIOS has been simplified as the device has been passed coordinates that it can understand.

If information has to be returned to the calling program, such as in the case of `Inquire input locator´, i.e. where is the graphics cursor now, device coordinates are returned to the GDOS and are likewise scaled before control is passed back.

NUTS & BOLTS.

I will now try and explain the technicalities of working with GSX. These will be clarified with the aid of diagrams (a picture's worth etc), as is does seem rather complex at first. It is worth mentioning that once you, as the applications or device driver writer, have created a routine that allows you to easily reference the data arrays concerned, the task is not quite so daunting as it first seems.

As calls to the BDOS involve the use of the BC and DE register to inform of your intentions, the problem is how do you manage to pass sometimes large amounts of data over using only one 16 bit value. Of course the answer is with the use of pointers as usual. Don't forget that the C register contains 115 on all calls to GDOS regardless and therefore cannot be used for pointer work.

The GSX standard expects the application programmer to have set up 5 arrays, and to give them their proper names, these are:

PARAMETER BLOCK

This 5x16 bit array contains the start addresses of the other data arrays described below. On a call to GDOS the DE register pair must contain the start address of this array.

PB pointed to by DE.

Fig 1. On call to BDOS requiring a GSX function.

```
 -------------
| C REG = 115 |
 -------------
 -------------
|  DE REG     |
 -------------
     |
     |                                                ------------------
     |                           /------------------| control array |
     |Points to                 /                     ------------------
     |                         /
     |parameter block         /                       --------------------------
 -------------------         /     /-----------------| input parameter array |
| PB      | PB+1    |-------/     /                     --------------------------
 -------------------              /
| PB+2    | PB+3    |----------/
 -------------------                                  ---------------------
| PB+4    | PB+5    |----------------------------------| input point array |
 -------------------                                  ---------------------
| PB+6    | PB+7    |----------\
 -------------------            \                     --------------------------
| PB+8    | PB+9    |-------\    \
 -------------------         \    \-----------------| output parameter array |
                               \                      --------------------------
                                \
                                 \                    ---------------------
                                  \
                                   \------------------| output point array |
                                                      ---------------------
```

Parameter block contents.

PB	Address of control array
PB+2	Address of input parameter array
PB+4	Address of input point coordinate array
PB+6	Address of output parameter array.
PB+8	Address of output point coordinate array

CONTROL ARRAY

This area is used by the GDOS and GIOS for control of the selected function. For example control(1) will contain the number of the required graphics routine (Remember open workstation - opcode 1). The remaining fields are used to contain values representing the amount of valid data contained in the other arrays on both entry to, and return from, the GIOS. These are usually extracted by the GIOS and used as loop counters.

INTIN ARRAY

Contains information to be used by the GIOS in a called function. These are not usually point coordinates but colour change values, text strings, input device modes and the suchlike.

PTSIN ARRAY

Contains point coordinates passed to the GIOS from the calling program. Used to contain line coordinates for example. This array is scaled to device coordinates by the GDOS before control is passed to the GIOS.

INTOUT ARRAY

Similar to intin but used by the GIOS to return data to the calling program. Typical entries are text rotation achieved as opposed to rotation asked for, input samples flagged as successful or not, linestyle selected etc.

PTSOUT ARRAY

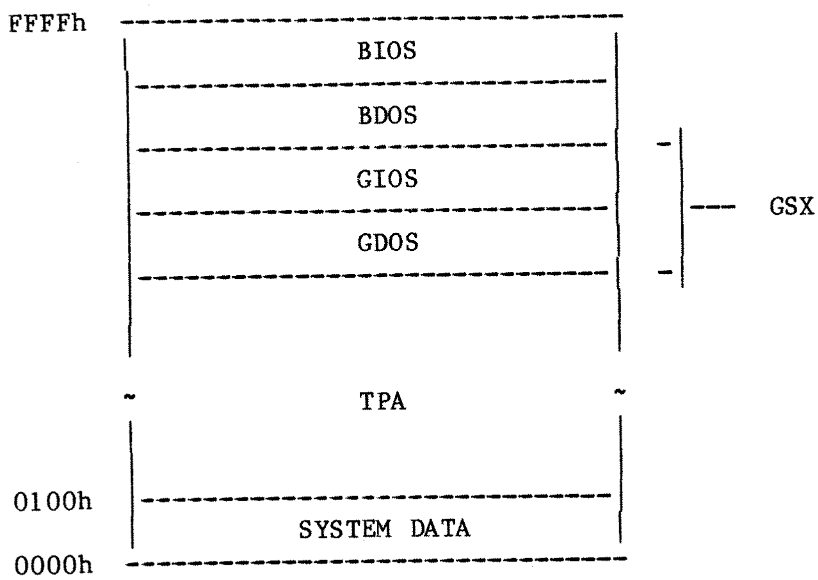
Similar to ptsin but used by the GIOS to pass coordinates back to the calling program. This array is also scaled by the GDOS before control is passed back to the caller, but this time to the 32k X 32k virtual frame size.

The GSX standard dictates that all array elements are 16 bit values, even ASCII text strings. All array references in the documentation are 1 based, which can be a source of bugs if you forget that array(1) is really array(0).

Memory arrangement.

Those of you au fait with the CP/M map may care to take a look at fig 2, which shows where the GDOS and GIOS live when at home. All calls to BDOS are rerouted through the GDOS first (via a modified 0005h) and passed on to the BDOS if it is not a graphics request.

Fig 2. GSX memory arrangement for CP/M 80.



<u>The final washup</u>

Well, what do you think? Is GSX the answer to the maidens prayer or is it too cumbersome in its constructs to be of any use to you. One thing is for sure, you will never be able to achieve the same fast animated graphics capability that you can get by `Bareback' colour board programming, due to the number of processes that have to be gone through before an element is displayed. But it is a much needed standard that will allow Joe Public to tap a variety of software sources, without ever having to find out which ports his cards are mapped to.

However if you were to give GSX the push you would never be able to use the new generation of graphical software that will shortly become available; I'm referring in particular to the desktop emulator that DR were showing off at COMPEC this year, it is similar to the things that we have seen from Macintosh so I'll say no more except that as a CP/M user you know it will run on more than one type of machine.

Before I return to "The happy hackers' holiday home" (Quote P.Greenhalgh, Gemini), I would like to mention the hours of innocent fun that I have had using DR DRAW to test out the Pluto GIOS. I would thouroughly recommend it to anyone who is passing through their second childhood, (and serious business users of course). The latest fun activity being the creation of a picture of a door, which to all intents and purposes is quite harmless, but if you zoom in on the keyhole and have a peep through you will find out whatever it was that made the butler blush...........

SUMMARY OF GSX OPCODES

Opcode	Description.
1	Initialise Workstation. Loads the device driver if necessary and sets default values.
2	Close Workstation. Halts graphics output to this workstation.
3	Clear Workstation. This clears the device and is equivalent to CLS if used on a CRT device.
4	Update Workstation. Display all pending graphics.
5	Escape. Enable device dependent operation. These deal mainly with character output if the device has an alpha mode with addressable character cells. Function 5 is called and an escape sequence ID is passed to GSX in control(6).

ID	Description
1	- Inquire addressable character cells.
2	- Enter graphics mode.
3	- Exit graphics mode.
4	- Alpha cursor up.
5	- Alpha cursor down.
6	- Alpha cursor right.
7	- Alpha cursor left.
8	- Home alpha cursor.
9	- Erase to end of screen.
10	- Erase to end of line.
11	- Direct cursor address (Move to row and column).
12	- Output cursor addressable text.
13	- Reverse video on.
14	- Reverse video off.
15	- Inquire current cursor address.
16	- Inquire tablet status. (Is a digitiser connected?)
17	- Hard copy. e.g. Dump a graphics screen to printer.
18	- Place graphics cursor at location.
19	- Remove graphics cursor. This turns the cursor off.
20 - 50	Reserved for future expansion.
51 - 100	Unused and available.

Opcode	Description.
6	Polyline. Output lines from data in PTSIN array.
7	Polymarker. Output markers at positions given in PTSIN. These markers are typically (. * O X +).
8	Text. Output text from machine font at specified position.
9	Filled area. Display and fill a polygon.
10	Cell array. Create a pixel array from colour data given in the INTIN array and at a position given in PTSIN.

SUMMARY OF GSX OPCODES - continued

Opcode	Description.
11	Generalised drawing primitive. These routines give you an easy way to display bars, arcs, pie slices and circles. These are not always fully implemented.
12	Set character height. Not possible of course if the Pluto font is used but should be implemented for plotter device drivers and the suchlike.
13	Set character up vector. This allows you to rotate character strings if the device will allow it.
14	Set colour representation. Will allow you to specify the red, green and blue intensity associated with a colour index. (Presumably this is for use in palette systems.)
15	Set linetype. You should be able to choose from solid, dashed, dotted or dashed-dotted.
16	Set line width.
17	Set line colour.
18	Set marker type.
19	Set marker scale.
20	Set marker colour.
21	Set hardware text font. (Only one to choose from in Pluto.)
22	Set text colour.
23	Set fill interior style. You should be able to choose from outline only, solid fill, pattern fill or hatch pattern fill.
24	Set fill style index. This allows you to specify the type of pattern or hatch fill you require from the selection available.
25	Set fill colour index. Having chosen the type of fill you require you can now say what colour you want it done in.
26	Inquire colour representation. Returns the RGB intensities of a requested colour index.
27	Inquire cell array. Returns the pixel colour values of the requested area.

SUMMARY OF GSX OPCODES - continued

Opcode	Description.
28	Input locator. This function serves as the interface between GSX and the outside world. Typically this will call the digitiser or mouse to return information on its whereabouts. When in request mode a cursor will appear on the screen and move according to the action of the specified input device. When in sample mode the current locator position is immediately returned and is often used in conjunction with ESCAPE 18 to plot a graphics cursor while at the same time displaying a rubber banding line.
29	Inquire input valuator. If some sort of analogue device is connected to the workstation, then the current value of its status is returned. Could be of use if graphical displays of external monitoring devices is required.
30	Inquire choice device. The choice device may be something like a keypad or function keys. For use in menu driven applications I would imagine.
31	String device. This returns a string from an input device which will of course be the keyboard in most cases.
32	Set writing mode. This affects the way in which lines or filled areas etc. are placed on the screen. The modes available are replace, overstrike, complement (XOR) and erase.
33	Set input mode. This lets you specify the type of input device you will require next i.e. locator, valuator, choice or string. You have also to specify whether this device is to operate in request or sample mode. In request mode the device waits until an event occurs such as the digitiser pen being pressed down to terminate input, in sample mode the current status or location of the device is returned without waiting.

Private Sales

Micropolis 400K Single/Sided Disk Drive, £85.00. Climax Colour Card and Software, £75.00. Mr Ward, Macclesfield (0625) 610678.

Large 19" rack case suitable for Gemini/Nascom with 8-slot card rack, floppy disk mounts, power supply. £85.00 ono. Plus Nascom 2, 64K RAM A card (4 MHz), cased with power supply and earom programmer. Software includes Nas-Sys 3, Zeap, Nas-Pen, Nas-Debug, Tool Kit, Documentation, etc, £220.00 ono and Gemini GM812 IVC card, £120.00 ono. Please make me an offer, I may not be able to refuse it. St Albans (0727) 73057.

Nascom IMP printer with Imprint operating system, spare ribbon cartridges and various electronics spares in original packing, £110.00. Nascom 1 with genuine Nascom 2 keyboard and PSU card, £60.00. Bits & PCs hexadecimal and control key-pad kit, £12.00. Gemini GM804 5v/12v PSU for twin disk drives, £15.00. Twin 19" matching instrument cases, to accommodate 5-card rack and PSU, and 2 standard height disk drives with PSU, £20.00. Carriage at cost. Telephone (0742) 460609.

IBM Selectric KBD Printer, ex 2741, with Hardware/Software interface for Nascom 2/Nas-Sys or any system with 8-bit port/Z80. In excellent working order, with Service Manuals. Haggle around £115.00, Ian on Ipswich (0473) 831353. (PS: Faster than some cheap daisy wheels, and better print Q.)

CRC PROGRAM – VERSION 5.0

By Richard Beal

There is a program in the CP/M user group library which is so useful that if you don't have it and you don't have access to the library then it is worth typing it all in - so it is listed in full below.

This program is called CRC.COM and its function is to calculate CRCs, which are a special almost infallible type of checksum, on files. The program can be used to confirm that a file has not been corrupted, even if it has passed through many different computer systems and communications links.

In its simplest form, you enter the command:-
    CRC filename
and the CRC for the file is displayed, as two hex numbers.

The filename can be ambiguous, so if for example you type:-
    CRC B:*.*
all the files on drive B are read and the CRCs displayed.

Now the shortcoming of this is that if you received a file you would need to know what its original CRC was in order to be sure that the current CRC was correct. But this is where CRC is so useful, because if you enter an F after the command as in:-
    CRC B:*.* F
then the resulting CRCs are not only displayed on the screen but are also written to a file called CRCKLIST.CRC. So when a disk of software is prepared, just before it is issued the CRC program is run, and the CRCKLIST.CRC file is added to the disk. The user of the disk has only to run the CRC program to the screen, or using Control-P to a printer, and compare the results with the values in the CRCKLIST.CRC file, which can be seen or printed using the TYPE command. Wouldn't it be a good idea if Gemini did this!

When version 5 of CRC.COM appeared, it had grown in size by more than 1K, and it didn't seem any different to the earlier versions. But it turned out to have a quite amazing feature which is well worth the extra space (and your time to type it in). If a disk has a CRCKLIST.CRC file on it and you just type the command CRC with nothing after it, then it will read in the file and then calculate the CRCs of all the files on the disk, reporting on whether they are different to those in the CRC file. It also reports on missing files. This means that with a single command you can verify the entire contents of a disk and be certain that it is the same as when it left the supplier. And if the software doesn't work, the supplier can't get away with the old excuse "It must be a bad disk - send it back and we'll replace it". And suppliers can save the trouble of replacing disks which are in fact correct.

For those who are curious to know how the CRCs are calculated, here is the equation, which is a CCITT standard polynomial:
    X^16 + X^15 + X^13 + X^7 + X^4 + X^2 + X + 1

I don't have a copy of the source code unfortunately, and I haven't found it in the CP/M library, so perhaps an enterprising person will disassemble it and comment it nicely. If they do, 80-BUS NEWS would like to print it, as it is quite short and must be a fine example of compact software.

Since you have to type in the code of CRC.COM, the first thing you should then do is type the command:-

CRC CRC.COM

which should give the answer B2 07, proving both that it works and that you haven't made any mistakes.

Perhaps when I tell you that hundreds of disks full of free software like this are available from the CP/M user group, and that it produces an interesting journal several times a year, you will send a cheque for £7.50 for your individual annual subscription to:-

CP/M Users Group (UK)
72 Mill Road
Hawley
Dartford
Kent DA2 7RZ

This is also the address of Derek Fordred, the software librarian, who can give you information about the amazing service which he offers.

The object code for CRCK V5.0 is given elsewhere.

LOST CHARACTERS IN CP/M

By Richard Beal

One of the advantages of having a buffered keyboard like that provided by the Gemini GM812 and GM832 video boards is that you can key ahead. However when using CP/M these characters can sometimes get lost. There are several reasons for this. One is that some programs check the keyboard and "gobble" any characters they find. Some programs, like MBASIC and WORDSTAR can gobble one character while they are starting up, and then it can reappear when the program is exited. But the most common problem and the one which is most annoying is that one character gets lost when a warm boot occurs, for example at the end of a PIP command. This is because the one character workspace in the BDOS is overwritten during a warm boot, and its contents are lost.

This article describes how to cure the problem of characters getting lost during a warm boot. I have used this patch for a long time, and have found it a useful improvement. It is very dangerous to make any alteration to the BDOS, since this leads to a non standard system, but this small change is harmless. I do not recommend any other changes to the BDOS. The SYS BIOS has implemented this alteration by patching the BDOS after each warm boot, but this article shows how to make the same change to the standard Gemini versions of CP/M, including the excellent new version called BIOS 3.

The solution is to move the location of the one byte workspace out of the BDOS into a spare location in the BIOS, by altering all references in the BDOS to this location. The location in the BIOS must be zero initially, otherwise a spurious character will appear on the screen after a cold boot. The method of installing the patch is to change the CP/M system which is generated by running either MOVCPM in the case of older versions of the BIOS, or GENSYS in the case of BIOS 3.

Having generated the system, use your debugging program to load the CP/M image, and examine location 13FC. This will contain 0E. (If it doesn't - STOP!) Change this to 8D. Now examine the next location, 13FD. Take the value in this byte, add 0B to it, and replace it. Now repeat the above for 1424-1425 and 1443-1444. Then SAVE the CP/M image to disk and use SYSGEN to write it to the system tracks. Use CONFIG as usual. This will place the workspace in the 32 byte patch area provided in the Gemini BIOS. No other changes are needed.

```
Listing of CRC.COM
Record 0.
0100:00:  C3 23 01 43 52 43 2E 43  4F 4D 20 35 2E 30 20 36  C£.CRC.C OM 5.0 6
0110: 10:  2F 31 38 2F 38 32 00 43  52 43 4B 46 49 4C 45 3F  /18/82.C RCKFILE?
0120: 20:  3F 3F 00 21 00 00 39 22  87 0C 31 87 0C CD 4D 09  ??.!..9" ..1..MM.
0130: 30:  CD 40 09 43 52 43 20 56  65 72 20 35 2E 30 0D 0A  M@.CRC V er 5.0..
0140: 40:  43 54 4C 2D 53 20 70 61  75 73 65 73 2C 20 43 54  CTL-S pa uses, CT
0150: 50:  4C 2D 43 20 61 62 6F 72  74 73 0D 0A 00 3A 5D 00  L-C abor ts...:].
0160: 60:  FE 20 C2 4F 06 CD 40 09  2B 2B 53 65 61 72 63 68  ~ BO.M@. ++Search
0170: 70:  69 6E 67 20 66 6F 72 20  43 52 43 4B 4C 49 53 54  ing for  CRCKLIST
Record 1.
0180: 00:  20 66 69 6C 65 2B 2B 00  CD 06 02 CA C7 01 CD 40   file++. M..JG.M@
0190: 10:  09 4E 6F 77 20 73 65 61  72 63 68 69 6E 67 20 66  .Now sea rching f
01A0: 20:  6F 72 20 22 43 52 43 4B  46 49 4C 45 22 20 66 69  or "CRCK FILE" fi
01B0: 30:  6C 65 2B 2B 00 11 0A 02  21 17 01 01 0B 00 CD 18  le++.... !.....M.
01C0: 40:  0A CD 06 02 C2 01 05 31  87 0C CD 2E 03 C2 29 0A  .M..B..1 ..M..B).
01D0: 50:  11 46 08 21 D7 0B 06 00  0E 0B CD F1 03 C2 00 02  .F.!W... ..Mq.B..
01E0: 60:  CD A1 05 C2 00 02 11 5D  08 0E 1F CD F1 03 C2 00  M!.B...] ...Mq.B.
01F0: 70:  02 CD B8 04 C2 00 02 21  E2 0B CD FD 03 C3 C7 01  .M8.B..! b.M}.CG.
Record 2.
0200: 00:  CD 94 03 C3 C7 01 C3 30  02 00 43 52 43 4B 4C 49  M..CG.C0 ..CRCKLI
0210: 10:  53 54 3F 3F 3F 00 00 43  52 43 4B 46 49 4C 45 3F  ST???..C RCKFILE?
0220: 20:  3F 3F 00 21 00 00 39 22  87 0C 89 0C 00 20 4D 09  ??.!..9" ..... M.
0230: 30:  C3 A0 02 2A 2C 02 EB 2A  2E 02 7D 93 7C 9A DA 89  C .*,.k* ..}.|.Z.
0240: 40:  02 21 00 00 22 2E 02 EB  2A 2C 02 7B 95 7A 9C D2  .!.."..k *,.{.z.R
0250: 50:  7B 02 2A 2A 02 19 EB 0E  1A CD 05 00 11 09 02 0E  {.**..k. .M......
0260: 60:  14 CD 05 00 B7 C2 75 02  11 80 00 2A 2E 02 19 22  .M..7Bu. ...*..."
0270: 70:  2E 02 C3 47 02 2A 2E 02  22 2C 02 11 80 00 0E 1A  ..CG.*.. ",......
Record 3.
0280: 00:  CD 05 00 21 00 00 22 2E  02 EB 2A 2A 02 19 EB 2A  M..!..". .k**..k*
0290: 10:  2C 02 7D B4 3E 1A C8 1A  2A 2E 02 23 22 2E 02 C9  ,.}4>.H. *..£"..I
02A0: 20:  AF 32 15 02 32 29 02 21  00 20 22 2C 02 22 2E 02  /2..2).! . ",."..
02B0: 30:  0E 0F 11 09 02 CD 05 00  3C C2 D9 02 0E 09 11 C7  .....M.. <BY....G
02C0: 40:  02 CD 05 00 C3 EE 08 0D  0A 4E 4F 20 46 49 4C 45  .M..Cn.. .NO FILE
02D0: 50:  43 52 43 20 46 49 4C 45  24 21 0A 02 11 D7 0B 01  CRC FILE $!...W..
02E0: 60:  08 00 CD 18 0A 3E 2E 12  13 01 03 00 CD 18 0A AF  ..M..>.. ....M../
02F0: 70:  12 CD 40 09 09 43 68 65  63 6B 69 6E 67 20 77 69  .M@..Che cking wi
Record 4.
0300: 00:  74 68 20 66 69 6C 65 20  2D 20 00 21 D7 0B 7E B7  th file  - .!W.~7
0310: 10:  CA 1A 03 CD 54 09 23 C3  0E 03 CD 4D 09 CD 4D 09  J..MT.£C ..MM.MM.
0320: 20:  AF 32 AD 0B 32 AE 0B 32  AF 0B 32 B0 0B C9 21 D7  /2-.2..2 /.20.I!W
0330: 30:  0B 3E 00 32 D5 0B E5 CD  33 02 E1 FE 1A C2 6F 03  .>.2U.eM 3.a~.Bo.
0340: 40:  3A D5 0B B7 C2 90 03 3A  B8 0B B7 CA 16 08 CD 40  :U.7B..: 8.7J..M@
0350: 50:  09 2A 2A 2A 4E 6F 20 43  52 43 20 46 69 6C 65 73  .***No C RC Files
0360: 60:  20 66 6F 75 6E 64 2A 2A  2A 24 00 3E 1A B7 C9 E6   found** *$.>.7If
0370: 70:  7F FE 0D CA 89 03 FE 0A  CA 89 03 77 23 3A D5 0B  .~.J..~. J..w£:U.
Record 5.
0380: 00:  3C 32 D5 0B FE 50 DA 36  03 3A D5 0B B7 CA 2E 03  <2U.~PZ6 .:U.7J..
0390: 10:  36 00 AF C9 CD 40 09 43  61 6E 20 6E 6F 74 20 70  6./IM@.C an not p
03A0: 20:  61 72 73 65 20 73 74 72  69 6E 67 0D 0A 00 21 D7  arse str ing...!W
03B0: 30:  0B 7E B7 CA BD 03 CD 54  09 23 C3 B1 03 CD 4D 09  .~7J=.MT .£C1.MM.
03C0: 40:  21 D7 0B 04 05 CA D5 03  3E 09 BE CA D0 03 36 20  !W...JU. >.>JP.6
03D0: 50:  23 05 C2 C8 03 36 5E 23  36 00 21 D7 0B 7E B7 CA  £.BH.6^£ 6.!W.~7J
03E0: 60:  E9 03 CD 54 09 23 C3 DD  03 21 AF 0B 34 CD 4D 09  i.MT.£C] .!/.4MM.
03F0: 70:  C9 1A BE C0 23 13 04 78  B9 C8 C3 F1 03 E5 11 5D  I.>@£..x 9HCq.e.]
Record 6.
0400: 00:  00 01 08 00 CD 18 0A 23  01 03 00 11 65 00 CD 18  ....M..£ ....e.M.
0410: 10:  0A AF 77 E1 7E B7 CA 20  04 CD 54 09 23 C3 14 04  ./wa~7J  .MT.£C..
0420: 20:  3E 01 32 B7 0B CD 40 09  20 2D 20 00 CD 73 09 D2  >.27.M@.  - .Ms.R
0430: 30:  4D 04 CD 40 09 46 69 6C  65 20 6E 6F 74 20 66 6F  M.M@.Fil e not fo
0440: 40:  75 6E 64 0D 0A 00 21 B0  0B 34 C3 09 09 3E 01 32  und...!0 .4C..>.2
0450: 50:  D2 0B CD 6F 08 C0 3A B5  0B 47 3A B2 0B B8 C2 84  R.Mo.@:5 .G:2.8B.
0460: 60:  04 3A B4 0B 47 3A B3 0B  B8 C2 84 04 CD 40 09 20  .:4.G:3. 8B..M@.
0470: 70:  2A 4D 61 74 63 68 2A 0D  0A 00 AF 32 B8 0B 21 AD  *Match*. ../28.!-
Record 7.
0480: 00:  0B 34 AF C9 CD 40 09 20  3C 2D 2D 20 69 73 2C 20  .4/IM@.  <-- is,
0490: 10:  77 61 73 20 2D 2D 3E 20  00 3A B4 0B CD 29 09 3E  was -->  .:4.M).>
04A0: 20:  20 CD 54 09 3A B5 0B CD  29 09 CD 4D 09 AF 32 B8   MT.:5.M ).MM./28
04B0: 30:  0B 21 AE 0B 34 AF 3C C9  CD 1C 06 C0 32 B4 0B 7E  .!..4/<I M..@24.~
04C0: 40:  23 FE 20 CA EB 04 CD 40  09 4E 6F 74 20 61 20 73  £~ Jk.M@ .Not a s
04D0: 50:  70 61 63 65 20 62 65 74  77 65 65 6E 20 43 52 43  pace bet ween CRC
04E0: 60:  20 76 61 6C 75 65 73 00  AF 3C C9 CD 1C 06 C0 32   values. /<IM..@2
04F0: 70:  B5 0B 47 AF C9 2D 43 41  54 41 4C 4F 47 3F 3F 3F  5.G/I-CA TALOG???
Record 8.
0500: 00:  00 CD 40 09 2B 2B 20 4E  6F 77 20 73 65 61 72 63  .M@.++ N ow searc
0510: 10:  68 69 6E 67 20 66 6F 72  20 22 2D 43 41 54 41 4C  hing for  "-CATAL
0520: 20:  4F 47 22 20 66 69 6C 65  2B 2B 00 11 0A 02 21 F5  OG" file ++....!u
0530: 30:  04 01 0B 00 CD 18 0A CD  06 02 C2 29 0A CD 2E 03  ....M..M ..B).M..
0540: 40:  C2 29 0A 06 00 21 D7 0B  CD E5 05 C2 94 05 3E 2E  B)...!W. Me.B..>.
0550: 50:  BE C2 94 05 23 04 CD F4  05 C2 94 05 CD CD 05 C2  >B..£.Mt .B..MM.B
0560: 60:  94 05 CD A1 05 C2 94 05  CD CD 05 C2 9B 05 CD F8  ..M!.B.. MM.B..Mx
0570: 70:  05 C2 9B 05 CD F8 05 CA  74 05 FE 4B C2 9B 05 CD  .B..Mx.J t.~KB..M
Record 9.
0580: 00:  CD 05 C2 9B 05 CD B8 04  C2 9B 05 2A AB 0B CD FD  M.B..M8. B..*+.M}
0590: 10:  03 C3 3D 05 3A B8 0B B7  C2 3D 05 CD 94 03 C3 3D  .C=.:8.7 B=.M..C=
05A0: 20:  05 22 AB 0B CD 0A 06 C0  0E 07 CD 0A 06 CA B3 05  ."+.M..@ ..M..J3.
05B0: 30:  FE 20 C0 0D C2 AA 05 7E  23 04 FE 2E C0 0E 03 CD  ~ @.B*.~ £.~.@..M
05C0: 40:  0A 06 CA C8 05 FE 20 C0  0D C2 BF 05 C9 7E FE 20  ..JH.~ @ .B?.I~~
05D0: 50:  CA D6 05 FE 09 C0 23 04  7E FE 20 CA D6 05 FE 09  JV.~.@£. ~~ JV.~.
05E0: 60:  CA D6 05 BF C9 CD F4 05  C0 CD F8 05 C8 FE 2E C0  JV.?IMt. @Mx.H~.@
05F0: 70:  2B 05 BF C9 CD F8 05 C0  7E 23 04 FE 30 DA 07 06  +.?IMx.@ ~£.~0Z..
Record 10.
0600: 00:  FE 3A D2 07 06 BF C9 FE  30 C9 7E 23 04 FE 21 DA  ~:R..?I~ 0I~£.~!Z
0610: 10:  19 06 FE 7F D2 19 06 BF  C9 FE 41 C9 CD 2E 06 C0  ..~.R..? I~AIM..@
0620: 20:  07 07 07 07 4F CD 2E 06  C0 47 79 B0 BF C9 7E 23  ....OM.. @Gy0?I~£
0630: 30:  04 FE 30 DA 4D 06 D6 30  FE 0A DA 4B 06 E6 1F D6  .~0ZM.V0 ~.ZK.f.V
0640: 40:  07 FE 0A DA 4D 06 FE 10  D2 4D 06 BF C9 B7 C9 3A  .~.ZM.~. RM.?I7I:
0650: 50:  6D 00 32 B1 0B FE 46 C2  53 07 C3 84 06 00 43 52  m.21.~FB S.C...CR
0660: 60:  43 4B 4C 49 53 54 24 24  24 00 05 C2 9B 05 CD F8  CKLIST$$ $..B..Mx
0670: 70:  05 C2 9B 05 CD F8 05 CA  74 05 FE 4B C2 9B 89 2C  .B..Mx.J t.~KB..,
Record 11.
0680: 00:  00 20 C2 9B C3 09 07 F5  2A 80 06 EB 2A 82 06 7D  . B.C..u *..k*..}
0690: 10:  93 7C 9A DA F9 06 21 00  00 22 82 06 EB 2A 80 06  .|.Zy.!. .."..k*..
06A0: 20:  7B 95 7A 9C D2 EB 06 2A  7E 06 19 EB 0E 1A CD 05  {.z.Rk.* ~..k..M.
06B0: 30:  00 11 5D 06 0E 15 CD 05  00 B7 C2 CA 06 11 80 00  ..]...M. .7BJ....
06C0: 40:  2A 82 06 19 22 82 06 C3  9C 06 0E 09 11 D6 06 CD  *..."..C .....V.M
06D0: 50:  05 00 F1 C3 EE 08 0D 0A  44 49 53 4B 20 46 55 4C  ..qCn... DISK FUL
06E0: 60:  4C 3A 20 43 52 43 46 49  4C 45 24 11 80 00 0E 1A  L: CRCFI LE$.....
06F0: 70:  CD 05 00 21 00 00 22 82  06 EB 2A 7E 06 19 EB F1  M..!..". .k*~..kq
```

```
Record 12.
0700: 00:   12 2A 82 06 23 22 82 06  C9 AF 32 69 06 32 7D 06   .*..£".. I/2i.2}.
0710: 10:   21 00 20 22 80 06 21 00  00 22 82 06 0E 13 11 5D   !. "..!. ."....]
0720: 20:   06 CD 05 00 0E 16 11 5D  06 CD 05 00 3C C2 53 07   .M.....] .M..<BS.
0730: 30:   0E 09 11 3B 07 CD 05 00  C3 EE 08 0D 0A 4E 4F 20   ...;.M.. Cn...NO
0740: 40:   44 49 52 20 53 50 41 43  45 3A 20 43 52 43 46 49   DIR SPAC E: CRCFI
0750: 50:   4C 45 24 31 87 0C 3E 00  32 D2 0B CD 73 09 D2 20   LE$1..>. 2R.Ms.R
0760: 60:   08 3A B7 0B B7 CA 7E 07  CD 23 0A 2B 2B 46 49 4C   .:7.7J~. M£.++FIL
0770: 70:   45 20 4E 4F 54 20 46 4F  55 4E 44 2B 2B 24 3A B1   E NOT FO UND++$:1
Record 13.
0780: 00:   0B FE 46 C2 16 08 2A 82  06 7D E6 7F C2 92 07 22   .~FB..*. .}f.B.."
0790: 10:   80 06 3E 1A F5 CD 87 06  F1 C2 86 07 0E 10 11 5D   ..>.uM.. qB.....]
07A0: 20:   06 CD 05 00 3C C2 CA 07  0E 09 11 B3 07 CD 05 00   .M..<BJ. ...3.M..
07B0: 30:   C3 CA 07 0D 0A 43 41 4E  4E 4F 54 20 43 4C 4F 53   CJ...CAN NOT CLOS
07C0: 40:   45 20 43 52 43 46 49 4C  45 24 C3 EE 07 00 43 52   E CRCFIL E$Cn..CR
07D0: 50:   43 4B 4C 49 53 54 43 52  43 00 53 4B 20 46 55 4C   CKLISTCR C.SK FUL
07E0: 60:   4C 3A 20 43 52 43 46 49  4C 45 24 11 80 00 0E 13   L: CRCFI LE$.....
07F0: 70:   11 CD 07 CD 05 00 C3 0D  08 E5 01 10 00 09 1A 77   .M.M..C. .e.....w
Record 14.
0800: 00:   13 23 0D C2 FE 07 D1 0E  17 CD 05 00 C9 21 5D 06   .£.B~.Q. .M..I!].
0810: 10:   11 CD 07 CD F9 07 CD 90  0A 0D 0A 44 4F 4E 45 24   .M.My.M. ...DONE$
0820: 20:   21 65 00 CD 09 0A CA 53  07 21 5D 00 11 51 08 01   !e.M..JS .!]..Q..
0830: 30:   08 00 CD 18 0A 21 65 00  11 5A 08 01 03 00 CD 18   ..M..!e. .Z....M.
0840: 40:   0A CD 40 09 0D 0A 2D 2D  3E 20 46 49 4C 45 3A 20   .M@...-- > FILE:
0850: 50:   20 58 58 58 58 58 58 58  58 2E 58 58 58 09 09 43    XXXXXXX X.XXX..C
0860: 60:   52 43 20 3D 20 00 CD 6F  08 CA 53 07 C3 29 0A 11   RC = .Mo .JS.C)..
0870: 70:   5C 00 0E 0F CD 05 00 3C  C2 91 08 CD 40 09 2B 2B   \...M..< B..M@.++
Record 15.
0880: 00:   4F 50 45 4E 20 46 41 49  4C 45 44 2B 2B 00 C3 09   OPEN FAI LED++.C.
0890: 10:   09 21 00 00 22 B2 0B 21  00 01 22 D3 0B 2A D3 0B   .!.."2.! .."S.*S.
08A0: 20:   7C FE 00 CA C8 08 0E 0B  CD 05 00 B7 CA B9 08 0E   |~.JH... M..7J9..
08B0: 30:   01 CD 05 00 FE 03 CA 29  0A 11 5C 00 0E 14 CD 05   .M..~.J) ..\...M.
08C0: 40:   00 B7 C2 D6 08 21 80 00  7E 32 B6 0B 23 22 D3 0B   .7BV.!.. ~26.£"S.
08D0: 50:   CD 0C 09 C3 9D 08 FE 01  C2 EE 08 3A B3 0B CD 29   M..C..~. Bn.:3.M)
08E0: 60:   09 3E 20 CD 54 09 3A B2  0B CD 29 09 AF C9 CD 40   .> MT.:2 .M)./IM@
08F0: 70:   09 09 09 2B 2B 46 49 4C  45 20 52 45 41 44 20 45   ...++FIL E READ E
Record 16.
0900: 00:   52 52 4F 52 2B 2B 0D 0A  00 AF 3C C9 2A B2 0B 7C   RROR++.. ./<I*2.|
0910: 10:   E6 80 F5 29 3A B6 0B 85  6F F1 CA 25 09 7C EE A0   f.u):6.. oqJ%.|n
0920: 20:   67 7D EE 97 6F 22 B2 0B  C9 F5 1F 1F 1F 1F CD 32   g}n.o"2. Iu....M2
0930: 30:   09 F1 E6 0F FE 0A DA 3B  09 C6 07 C6 30 C3 54 09   .qf.~.Z; .F.FOCT.
0940: 40:   E3 7E CD 54 09 23 7E B7  C2 41 09 E3 C9 3E 0D CD   c~MT.£~7 BA.cI>.M
0950: 50:   54 09 3E 0A C5 D5 E5 E6  7F 5F D5 CD 68 09 D1 0E   T.>.EUef ._UMh.Q.
0960: 60:   02 CD 05 00 E1 D1 C1 C9  3A B1 0B FE 46 C0 7B CD   .M..aQAI :1.~F@{M
0970: 70:   87 06 C9 0E 1A 11 80 00  CD 05 00 AF 32 68 00 32   ..I..... M../2h.2
Record 17.
0980: 00:   7C 00 3A B7 0B B7 CA B2  09 21 5C 00 11 BA 0B 01   |.:7.7J2 .!\..:..
0990: 10:   0C 00 CD 18 0A 3A 5C 00  32 C6 0B 21 BA 0B 11 5C   ..M..:\. 2F.!:..\
09A0: 20:   00 01 0C 00 CD 18 0A 0E  11 11 5C 00 CD 05 00 C3   ....M... ..\.M..C
09B0: 30:   DA 09 21 C6 0B 11 5C 00  01 0C 00 CD 18 0A 0E 11   Z.!F..\. ...M....
09C0: 40:   11 5C 00 CD 05 00 21 BA  0B 11 5C 00 01 0C 00 CD   .\.M..!: ..\....M
09D0: 50:   18 0A 0E 12 11 5C 00 CD  05 00 3C 37 C8 3D E6 03   .....\.M ..<7H=f.
09E0: 60:   87 87 87 87 87 C6 81 6F  26 00 E5 11 C7 0B 01 0B   .....F.o &.e.G...
09F0: 70:   00 CD 18 0A E1 11 5D 00  01 0B 00 CD 18 0A AF 32   .M..a.]. ...M../2
Record 18.
0A00: 00:   68 00 32 7C 00 32 B7 0B  C9 CD 11 0A C0 CD 11 0A   h.2|.27. IM..@M..
0A10: 10:   C0 7E E6 7F FE 24 23 C9  7E 12 23 13 0B 78 B1 C2   @~f.~$£I ~.£..x1B
0A20: 20:   18 0A C9 D1 0E 09 CD 05  00 3A B1 0B FE 46 C2 7D   ..IQ..M. .:1.~FB}
0A30: 30:   0A 2A 82 06 7D E6 7F C2  3D 0A 22 80 06 3E 1A F5   .*..}f.B =."..>.u
0A40: 40:   CD 87 06 F1 C2 31 0A 0E  10 11 5D 06 CD 05 00 3C   M..qB1.. ..].M..<
0A50: 50:   C2 75 0A 0E 09 11 5E 0A  CD 05 00 C3 75 0A 0D 0A   Bu....^. M..Cu...
0A60: 60:   43 41 4E 4E 4F 54 20 43  4C 4F 53 45 20 43 52 43   CANNOT C LOSE CRC
0A70: 70:   46 49 4C 45 24 0E 13 11  5D 06 CD 05 00 CD 90 0A   FILE$... ].M..M..
Record 19.
0A80: 00:   0D 0A 0D 0A 2B 2B 41 42  4F 52 54 45 44 2B 2B 24   ....++AB ORTED++$
0A90: 10:   D1 0E 09 CD 05 00 3A D2  0B B7 CA 7E 0B AF 32 B1   Q..M..:R .7J~./21
0AA0: 20:   0B CD 4D 09 CD 40 09 51  75 61 6E 74 69 74 79 20   .MM.M@.Q uantity
0AB0: 30:   6F 66 20 66 69 6C 65 20  43 52 43 20 74 68 61 74   of file  CRC that
0AC0: 40:   20 6D 61 74 63 68 65 64  20 2D 20 00 3A AD 0B CD    matched  - .:-.M
0AD0: 50:   83 0B 3A AE 0B B7 CA 10  0B CD 4D 09 CD 40 09 51   ..:..7J. .MM.M@.Q
0AE0: 60:   75 61 6E 74 69 74 79 20  6F 66 20 66 69 6C 65 20   uantity  of file
0AF0: 70:   43 52 43 20 74 68 61 74  20 64 69 64 20 6E 6F 74   CRC that  did not
Record 20.
0B00: 00:   20 6D 61 74 63 68 20 2D  20 00 3A AE 0B CD 83 0B    match -  .:..M..
0B10: 10:   3A AF 0B B7 CA 4A 0B CD  4D 09 CD 40 09 51 75 61   :/.7JJ.M M.M@.Qua
0B20: 20:   6E 74 69 74 79 20 6F 66  20 6C 69 6E 65 73 20 66   ntity of  lines f
0B30: 30:   61 69 6C 65 64 20 70 61  72 73 65 20 74 65 73 74   ailed pa rse test
0B40: 40:   20 2D 20 00 3A AF 0B CD  83 0B 3A B0 0B B7 CA 7E    - .:/.M ..:0.7J~
0B50: 50:   0B CD 4D 09 CD 40 09 51  75 61 6E 74 69 74 79 20   .MM.M@.Q uantity
0B60: 60:   6F 66 20 66 69 6C 65 28  73 29 20 6E 6F 74 20 66   of file( s) not f
0B70: 70:   6F 75 6E 64 20 2D 20 00  3A B0 0B CD 83 0B 2A 87   ound - . :0.M..*.
Record 21.
0B80: 00:   0C F9 C9 1E 00 06 64 CD  94 0B 06 0A CD 94 0B C6   .yI...dM ....M..F
0B90: 10:   30 C3 54 09 0E FF 0C 90  D2 96 0B 80 57 79 B3 CA   OCT..... R...Wy3J
0BA0: 20:   A9 0B F6 30 CD 54 09 1E  30 7A C9 74 69 00 00 00   ).vOMT.. OzIti...
0BB0: 30:   00 00 00 00 69 6C 00 01  01 00 00 00 00 00 00 00   ....il.. ........
0BC0: 40:   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0BD0: 50:   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0BE0: 60:   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0BF0: 70:   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
End of listing of CRC.COM
```

THE DH BITS **By David Hunt**

A few days ago a customer of ours popped into the shop and gave us a long tale about having exported some gear in 1982 and because of some customs problem could they have a copy receipt for the goodies? Well much as we try to please, trying to find a receipt for goods supplied on an unknown date in 1982 is a little too much. So, as we remembered the customer, and were therefore pretty certain that he had bought something a long time ago, we suggested that we could give him a new receipt, but dated 1982, altogether a lot less hassle. This was fine, but could we make sure that the receipt was not dated on a Sunday or Bank Holiday or some other such suspicious date? I wonder how many of you have a 1982 diary or calendar to hand? We certainly didn't. This whole thing was taking on the proportions of a farce, as trying to find a 1982 calendar looked like turning into as much fun as trying to find the original receipt. Then a thought occurred. A very long time ago, when Nascom first grew BASIC, I cobbled together a Calendar program. Did it still exist, had it been converted to run under CP/M, etc? A quick consult of the CAT program revealed CALENDAR.BAS, which when fired up, worked.

This turned out to be an interesting program, short and to the point, so for my sins, it's offered here. It may even have been published before, although a quick look through the old back issues of INMC didn't reveal it. Looking closely at the program, I'm pretty sure it's not all original DH, so I must have pinched it, or bits of it, from somewhere. The basic algorithm for working out the start date seems to be attributable to David Ahl's `101 Basic Programs', but different, and I'm pretty sure, left to my own devices I wouldn't have calculated the screen TAB positions the way it's done here, but 1979 was a long time ago and premature senility is creeping over me, so I don't remember. One thing I do remember was the trap for the 1752 start for the Gregorian calendar, and the nasty business of the (MOD 4000)+2000 over the fact that the year 2000 won't be a leap year when it should be. This doesn't happen again until the year 6000, so it looks like I `short cut' the procedure and checked at the 1000 year boundary. This makes the effective operating range the 1250 odd year span from the year 1753 to the year 2999. The 1752 trap won't allow earlier dates, and the 2999 trap stops the program thinking that the year 3000 is not a leap year when it should be. All pretty esoteric really as I doubt that anyone will be interested in dates outside a century span anyway. See Listing One.

Another thing which has been causing problems lately (and still on the subject of dates and times) is using machine code routines under dBASE II. Now versions 2.4 or later allow machine code calls, and the favourite for these is making the cursor display different on the SVC/IVC card, or reading either of the Gemini clocks (the clock on the GM816 I/O card or the GM822 RTC) in as dBASE data.

The cursor first. The various permutations of Gemini CP/M do different things with the cursor when either waiting for a CP/M command, or actually executing a program (it's documented in the manual, so I won't repeat it here). Normally the cursor blinks whilst in the CP/M command mode, and becomes a non-blinking cursor when executing a program. I say normally, as the Winchester based Quantum machine I borrow when I visit Gemini, turns the cursor off completely when executing a program. I find this infuriating if the program I'm using doesn't turn it back on again, and of course, dBASE is one program that doesn't. I know this is very easily patchable with the program

CONFIG, but I can't be bothered to do it to a machine which doesn't belong to me in the first place.

All that apart, I use dBASE with the reply prompts highlighted, that is, in inverse video, and I like to see a flashing cursor under these circumstances anyway. Further, the prompt is usually on the eighth line which looks untidy as it overlays the underhangs on characters like lower case g, j, y, etc. So I like to see a flashing cursor on the 9th line when using dBASE. How can this be achieved?

Well the obvious is to send a command string to the SVC/IVC to turn the cursor on, blinking, and move it down one line. It appears nice and easy, just work out the control words and then print them:

```
STORE CHR(27)+"Y"+CHR(72)+CHR(8) TO curs
? curs
```

Not so, dBASE says different. The main problem in this instance is the CHR(8), instantly recognisable as 08h, or backspace. Now dBASE does not send a backspace, it translates the 08h into a cursor left movement. To achieve a back space you use the DEL key and the 7fh code is translated into a three byte string: 08h (to move the cursor back one), 20h (to delete the character at the cursor, which also advances the cursor) and 08h to move the cursor back again. (This problem is not uncommon, several control codes are converted either by the BDOS or the application program, so dBASE is not an isolated instance. The characters usually affected would be 08h, backspace, 09h, tab, 0ah, line feed and 0dh carriage return.) The answer is to use the PUTVID program in the SVC/IVC manual, but this means making a machine code patch for dBASE (or whatever).

According to the manual dBASE uses memory up to about a000H, so any address above this could be used for the patch, so I chose C000H for convenience. The machine code listing is given in Listing Two.

All it has is a data table at c011h and the PUTVID routine enclosed in a loop to shove the four characters in turn at the video card. dBASE uses POKES like Basic, but the `call' procedure is slightly different, you use the SET function to set the call address, then use CALL to call it. So the (decimalized) dBASE version of the above is as follows, (49152 is the decimal equivalent of c000h):

```
* Enable cursor type
SET CALL TO 49152
POKE 49152,6,4,33,17,192,219,178,15,56,251,126,35,211,177,16,245
POKE 49168,201,27,89,72,8
CALL
```

You could do the same with MBASIC, using the same POKE addresses and data, thus:

```
10 CURS=49152
20 FOR A=49152 TO 49172: READ B: POKE A,B: NEXT
30 DATA 6,4,33,17,192,219,178,15,56,251,126,35,211,177,16,245
40 DATA 201,27,89,72,8
50 CALL CURS
```

But be warned, MBASIC starts it's workspace and stack from the top of the TPA. The stack and/or workspace could come crashing through the program if you're not careful. So POKEing this into RAM in MBASIC is not a clever idea. Far sneakier is a method suggested by Carl Lloyd-Parker, and that makes use of the fact that although MBASIC knows where strings are in a program, it doesn't pull then out into the workspace area until some additive or subtractive manuipulation is carried out on it. In other words, the string stays where it was originally written in the program unless you do something nasty to it. Carl's method is to define a string of asterisks somewhere in the program, the string being as long as the machine code to be POKEd into it. Then calculate the position of the string using VARPTR, then POKE the code into the string as the example above. The address calculated from VARPTR is also the call address for the program. There are two examples of this, one by Carl in his IVC HIRES programs and another by me in the BASIC demo CLOCK program in the Gemini GM816 manual. The fun part of this method is if the string is placed as the first line of the program, then, when the program has been run, the string is full of lots of interesting junk, making the program unlistable (if you start the list at the first line).

Perhaps you're wondering how I decimalize the HEX code from the programs as assembled into a form useable by dBASE or MBASIC, or whatever, at least without making too many mistakes. Simple I use BASIC to do it! First I assemble the program using M80 and L80 as usual, then I load it up under ZSID, DDT, GEM-DEBUG or some other debugger. I then clear out the memory around the location where the program is to reside and move the program to the working address. This gives me the program in memory at its correct place with some 00h's before and after it. (Nice and identifiable that way.)

Next into MBASIC, work out the start and end addresses using the &H function in BASIC, note that this gives negative answers if not treated right. Take the instance above:

```
? 65536 + &Hc000
 49152
Ok
? 65536 + &Hc011
 49169
Ok
```

Now for the crafty bit, open a sequential file and write the code to it, I do this in the command mode like so:

```
OPEN "O",£1,"CODE": FOR A=49152 TO 49169: PRINT£1,PEEK(A);: NEXT: CLOSE
```

Surpise surpise, this gives me an ASCII file that I can bash into a text editor and edit to suit. My favourite address for machine code to be used in this way is c000h, as for some reason I can always remember 49152. When it comes to the end addresses of these programs, having calculated it I usually have a quick PEEK around the calculated address to see if I got right, hence the nulls either end of the program. The whole process takes about as long as it took to write up and being done by machine is not susceptible to human error.

Listing One

```
10 '                  *** CALENDAR ***
20 '
30 '    By D. R. Hunt.                  2 October 1979
40 '
50 '    Suitable for Nascom 1/2 fitted with NASBUG `T'
60 '    or NAS-SYS monitors.
70 '    This program occupies approx. 1.5K.
80 '
90 '    Modified for Gemini CP/M and MBASIC. 22 March 1981
100 '
110 CLS$=CHR$(26): PRINT CLS$;"Calendar"
120 '
130 ' ** Get inputs
140 RESTORE: F=0: F1=0
150 PRINT: PRINT "Do you require continuous output ? ";
160 I$=INKEY$: IF I$="" THEN 160 ELSE IF I$=CHR$(3) THEN END
170 I$=CHR$(ASC(I$) AND &HDF): PRINT I$
180 IF I$="Y" THEN F1=1: GOTO 190 ELSE IF I$<>"N" THEN 140
190 INPUT "What year ";Y: IF Y>=1753 AND Y=<2999 THEN 230
200 PRINT "Out of range !": GOTO 190
210 '
220 ' ** Calculate starting day
230 J=Y:GOSUB 490
240 '
250 ' ** Print calendar
260 FOR M=1 TO 12
270   READ M$: PRINT CLS$: PRINT TAB(34-INT((LEN(M$)+6)/2));M$;" ";Y
280   PRINT"======================================================================"
290   PRINT" SUN       MON       TUE       WED       THU       FRI       SAT "
300   PRINT"======================================================================"
310   READ DY: IF F=1 THEN IF M=2 THEN DY=DY+1
320   FOR D=1 TO DY
330     PRINT TAB(10*C+1);: IF D<10 THEN PRINT" ";
340     PRINT D;: C=C+1: IF C=7 THEN PRINT: PRINT: PRINT: C=0
350   NEXT D
360   IF C<>0 THEN PRINT
370   PRINT"======================================================================"
380   IF F1=1 OR M=12 THEN 410
390   PRINT: PRINT TAB(42) "Hit any key to continue."
400   I$=INKEY$: IF I$="" THEN 400 ELSE IF I$=CHR$(3) THEN END
410 NEXT M: GOTO 140
420 '
430 ' ** Data for months and days
440 DATA "JANUARY",31,"FEBRUARY",28,"MARCH",31,"APRIL",30,"MAY",31,"JUNE",30
450 DATA "JULY",31,"AUGUST",31,"SEPTEMBER",30,"OCTOBER",31,"NOVEMBER",30
460 DATA "DECEMBER",31
470 '
480 ' ** Is it a leap year ? F=1 says yes.
490 IF J/1000=INT(J/1000) THEN 550
500 IF J/400=INT(J/400) THEN F=1: GOTO 550
510 IF J/100=INT(J/100) THEN 550
520 IF J/4=INT(J/4) THEN F=1 ELSE GOTO 550
530 '
540 ' ** What's the first day ?
550 J1=J-1
560 K=2+J1+INT(J1/4)-INT(J1/100)
570 K=K+INT(J1/400)-INT(J1/1000)
580 C=K-INT(K/7)*7
590 RETURN
600 '
610 END
```

Listing Two

```
To Load the IVC/SVC Cursor register     MACRO-80 3.44          PAGE   1

                                .z80
 0000'                          aseg
                                org     0100h
                                .phase  0c000h

 001B                   esc     equ     1bh

 C000    06 04                  ld      b,4           ; Four chars to send
 C002    21 C011                ld      hl,chars
 C005    DB B2          rdy:    in      a,(0b2h)      ; See if IVC/SVC ready
 C007    0F                     rrca
 C008    38 FB                  jr      c,rdy
 C00A    7E                     ld      a,(hl)        ; Get the character
 C00B    23                     inc     hl
 C00C    D3 B1                  out     (0b1h),a
 C00E    10 F5                  djnz    rdy
 C010    C9                     ret

 C011    1B 59 48 08 chars:     defb    esc,"Y",48h,8h  ; Cursor control

                                end
```

Listing Three

```
GM816 Clock Read Routine     MACRO-80 3.44     09-Dec-82     PAGE     1

                                .z80
 0000'                          aseg
                                org     100h
                                .phase  0c000h

 0020                   clock   equ     20h           ; Base port of clock
 000B                   nmrreg  equ     11            ; Number of regs to read

 C000    C3 C014                jp      start
 C003                           defs    6             ; Workspace for results
 C009                   regs:   defs    11            ; Workspace for registers
```

```
C014    0E 21     start:  ld      c,clock+1       ; Read into workspace
C016    06 0B             ld      b,nmrreg
C018    21 C009           ld      hl,regs
C01B    0C        read:   inc     c
C01C    ED A2             ini
C01E    20 FB             jr      nz,read

                  ; Now see if any changed during read and convert into HEX
C020    06 04             ld      b,4
C022    21 C009           ld      hl,regs
C025    11 C008           ld      de,regs-1
C028    CD C03C   scan:   call    scanl
C02B    28 E7             jr      z,start
C02D    10 F9             djnz    scan
C02F    CD C055           call    test
C032    28 E0             jr      z,start
C034    12                ld      (de),a
C035    1B                dec     de
C036    CD C03C           call    scanl
C039    28 D9             jr      z,start
C03B    C9                ret                     ; Exit, all done

                  ; Take two bytes and convert to HEX
C03C    CD C055   scanl:  call    test            ; Do the low byte
C03F    C8                ret     z
C040    4F                ld      c,a             ; Save it in C
C041    CD C055           call    test            ; Do the high byte
C044    C8                ret     z
C045    C5                push    bc
C046    CB 27             sla     a               ; Multiply by 10
C048    4F                ld      c,a
C049    CB 27             sla     a
C04B    CB 27             sla     a
C04D    81                add     a,c
C04E    C1                pop     bc
C04F    81                add     a,c             ; Add low to high
C050    12                ld      (de),a          ; Save it in workspace
C051    1B                dec     de
C052    AF                xor     a               ; Clear any Z flag
C053    3D                dec     a
C054    C9                ret

                  ; Take a byte and test the change flag
C055    7E        test:   ld      a,(hl)
C056    23                inc     hl
C057    E6 0F             and     0fh             ; Mask off top nibble
C059    FE 0F             cp      0fh             ; Test for change
C05B    C9                ret

                          end
```

Listing Four

```
* Get the time and display on SVC
CALL
IF PEEK(s:time+6)<=9
    STORE "0"+STR(PEEK(s:time+6),1) TO hr
ELSE
    STORE STR(PEEK(s:time+6),2) TO hr
ENDIF
IF PEEK(s:time+7)< 9
    STORE "0"+STR(PEEK(s:time+7),1) TO min
ELSE
    STORE STR(PEEK(s:time+7),2) TO min
ENDIF
IF PEEK(s:time+8)<=9
    STORE "0"+STR(PEEK(s:time+8),1) TO sec
ELSE
    STORE STR(PEEK(s:time+8),2) TO sec
ENDIF
STORE CHR(27)+"t"+hr+min+sec+CHR(27)+"tE" TO tt
? tt

* Get todays date, add to time, store as a logon string
STORE "JanFebMarAprMayJunJlyAugSepOctNovDec" TO m
STORE "SunMonTueWedThuFriSat" TO d
STORE hr+"."+min TO hm
STORE $(m,3*PEEK(s:time+3)-2,3) TO m
STORE $(d,3*PEEK(s:time+4)-2,3) TO d
STORE "Log on time "+hm+"   "+d+" "+STR(PEEK(s:time+5),2)+" "+m TO s:logonl
```

Listing Five.

```
POKE 49152,195,105,192
POKE 49167,62,255,211,30,62,255,211,30,201,62,255,211,31,62,255,211,31
POKE 49184,201,62,255,211,29,62,255,211,31,62,16,211,31,201,237,81
POKE 49200,237,89,219,28,237,81,201,205,33,192,205,15,192,33,3,192
POKE 49216,6,12,22,236,30,140,14,29,205,46,192,230,15,254,15,40
POKE 49232,230,119,35,21,29,16,241,201,126,35,205,99,192,70,35,128
POKE 49248,18,19,201,7,71,7,7,128,201,205,15,192,205,24,192,205
POKE 49264,55,192,33,3,192,84,93,205,88,192,237,160,6,4,197,205
POKE 49280,88,192,193,16,249,201,0,0,0,0,0,0,0,0,0,62
POKE 49296,255,211,30,62,255,211,30,201,62,255,211,31,62,255,211,31
POKE 49312,201,62,255,211,29,62,255,211,31,62,16,211,31,201,237,81
POKE 49328,237,89,219,28,237,81,201,205,33,192,205,15,192,33,3,192
POKE 49344,6,12,22,236,30,140,14,29,205,46,192,230,15,254,15,40
POKE 49360,230,119,35,21,29,16,241,201,126,35,205,99,192,70,35,128
POKE 49376,18,19,201,7,71,7,7,128,201,205,15,192,205,24,192,205
POKE 49392,55,192,33,3,192,84,93,205,88,192,237,160,6
```

Ok, now on to clocks and dBASE. The same process is used, just the programs are different. Firstly the clock call routine for the GM816, this is likely to be the more popular. I don't include any utilities for setting the clock as both the GM816 and the GM822 are sufficiently reliable to only require setting every now and then by separate utilities described in their respective manuals.

See Listing Three

This lot comes down to a neat and tidy little piece so:

```
STORE 49152 TO s:time
SET CALL TO s:time

POKE 49152,195,20,192
POKE 49172,14,33,6,11,33,9,192,12,237,162,32,251,6,4,33,9
POKE 49188,192,17,8,192,205,60,192,40,231,16,249,205,85,192,40,224
POKE 49204,18,27,205,60,192,40,217,201,205,85,192,200,79,205,85,192
POKE 49220,200,197,203,39,79,203,39,203,39,129,193,129,18,27,175,61
POKE 49236,201,126,35,230,15,254,15,201
```

Note that in this routine the 11 registers are first read into an 11 byte workspace, the results are then converted from the decimal one byte per digit into HEX numbers in a second workspace, as dBASE requires the numbers stored in HEX. It is then a simple matter of PEEKing the workspace to extract the time and date. The order is thus:

```
s:time+3 = month
s:time+4 = day of week
s:time+5 = day of month
s:time+6 = hours
s:time+7 = minutes
s:time+8 = seconds
```

Listing Four is an extract from my radio logbook program which firstly shoves the correct time at the SVC and then picks up a logon string for later use. This is for the GM816. The same is true for the GM822 hung on a PIO device. The routine is quite a bit larger, but the output format is the same. In this instance the port decode was 1ch - 1fh, if you want it any different, then you can unscramble it and disassemble it yourself. See Listing Five.

Naturally these routines could be used with any high level language which has the ability to PEEK and POKE and to CALL user subroutines. The principles are the same regardless, but care should be taken as to where they are put as some parts of programs could crash into them if they are located at c000h, or worse, they could be moved by the program itself.

SYSTEM ROUTINES IN POLYDOS AND POLYDOS DISK BASIC

By Geoff Higgs

The values of $TAB, $OUT, $UOUT, $IN, $UIN and $NMI are variously initialized by Nas-Sys 3, ROM BASIC, PolyDos and PolyDos Disc Basic. A table of these values might save some head scratching when incorporating user routines or patches.

Fctn	Wkspc. Add.	Nas-Sys 3	ROM BASIC	PolyDos	Disc Basic
$TAB	0C71 (3185)	0700 (1792)	0700 (1792)	C07E (-16258)	C07E (-16258)
$OUT	0C73 (3187)	0779 (1913)	0779 (1913)	0779 (1913)	B138 (-20168)
$UOUT	0C78 (3192)	002F (47)	002F (47)	C240 (-15808)	B13A (-20166)
$IN	0C75 (3189)	077C (1916)	077C (1916)	D416 (-11242)	D416 (-11242)
$UIN	0C7B (3195)	002F (47)	002F (47)	002F (47)	002F (47)
$NMI	0C7E (3198)	0475 (1141)	FEDE (-290)	unaltered	(see note *)

* The byte at 0C7D is set to £C3 (jump) by Nas-Sys initialization. 0C7E/F is set to 0475 by Nas-Sys PARSE calling INLS at 02E8 each time. Therefore if on power up neither Nas-Sys or ROM BASIC is implemented then state of 0C7E/F is indeterminate. If Nas-Sys STMON is called (as by PolyDos) but Nas-Sys command input is not used then the byte £C3 is set but not the subsequent address.

Note that PolyDos copies out the routine table STABA to its workspace. The base is C07E and the table actually begins at C100. Within this table the addresses of MRET, CRT, NNIM, and BLINK are altered. RKBD, SP2 and SCALI are altered as these routines are written into PolyDos so as to make it compatible with Nas-Sys 1.

Note further that PolyDos Disc Basic extension to ROM basic once again alters the address of MRET and also alters the address of INLIN.

These changes are tabulated below. (decimal values in brackets)

Routine	Nas-Sys address	PolyDos address	PolyDos table pos'n	PolyDos Basic add.
MRET	03FE (1022)	D09D (-12140)	C134 (-16076)	B079 (-20359)
CRT	0190 (400)	D3C7 (-11321)	C148 (-16056)	
NNIM	0742 (1858)	D410 (-11248)	C16E (-16018)	
BLINK	0078 (120)	D419 (-11239)	C174 (-16012)	
RKBD	0082 (142)	D481 (-11135)	C178 (-16008)	
SP2	0362 (866)	D504 (-11004)	C17A (-16006)	
SCALI	05B5 (1461)	D509 (-10999)	C17C (-16004)	
INLIN	02F0 (752)	not altered	C144 (-16060)	BDEC (-16916)

Since PolyDos keeps STABA in RAM then a routine to trap carriage returns before the CRT routine, such as shown above, can be "patched in" rather than written as an user routine. It is only necessary to alter the address at C148 (-16056), normally D3C7 (-11321) to the "patch" address and end the "patch" routine with a jump to CRT (C3 C7 D3).

POLYDOS FILE NAME LISTING

By M. J. R. Gibbs

This program is for making a listing on a printer of all the file names on a Gemini GM809/GM815 system with Polydos 2.0. The output consists of a listing of file names as they appear on the disks and a sorted list of file names, as well as a usage summary of all the user disks owned. For this to work properly the user must have some way of identifying his disks, I have decided to use the last two digits of the twenty digit disk name (see the FORMAT utility or the NAME command).

Another way of obtaining a list is to use the DIR;ELD command but this takes a long time for a large number of disks and does not give a sorted listing and a summary. With only a few disks it is quite easy to remember where things are kept, this becomes increasingly more difficult as the number of disks increases. (Also I have a remarkably bad memory).

The following is a sample output from running this program against two disks the first disk has an identifier D2 and the second B2:-

```
        DISK INDEX LISTING
        ==================

EDIT2    BS  D2  ACCOUNT  BS  D2  MEMTEST2 GO  B2  CHRHEX   Z2  B2
EDIT1    BS  D2  PAYROLL  BS  D2  MEMTEST2 Z2  B2  TAPECOPY GO  B2
ACCTDATA DT  D2  INDXDATA DT  D2  MEMTEST3 GO  B2  TAPECOPY Z2  B2
EDITFILE DT  D2  TEST     GO  D2  MEMTEST3 Z2  B2  UPDATE   GO  B2
VORTEX   GO  D2  INDEX    BS  D2  CHECKSUM GO  B2  UPDATE   Z2  B2
SPAOLD   DT  D2  DIRFILE  TX  D2  CHECKSUM Z2  B2  DUMP     Z2  B2
SPA      BS  D2  DIRFILE  GO  D2  CHRDIS   GO  B2  DUMP     GO  B2
SPADATA  TX  D2  MEMTEST1 GO  B2  CHRDIS   Z2  B2
SPADATA  DT  D2  MEMTEST1 Z2  B2  CHRHEX   GO  B2

        DISK INDEX LISTING
        ==================

ACCOUNT  BS  D2  DIRFILE  TX  D2  MEMTEST1 Z2  B2  SPAOLD   DT  D2
ACCTDATA DT  D2  DUMP     GO  B2  MEMTEST2 GO  B2  TAPECOPY GO  B2
CHECKSUM GO  B2  DUMP     Z2  B2  MEMTEST2 Z2  B2  TAPECOPY Z2  B2
CHECKSUM Z2  B2  EDIT1    BS  D2  MEMTEST3 GO  B2  TEST     GO  D2
CHRDIS   GO  B2  EDIT2    BS  D2  MEMTEST3 Z2  B2  UPDATE   GO  B2
CHRDIS   Z2  B2  EDITFILE DT  D2  PAYROLL  BS  D2  UPDATE   Z2  B2
CHRHEX   GO  B2  INDEX    BS  D2  SPA      BS  D2  VORTEX   GO  D2
CHRHEX   Z2  B2  INDXDATA DT  D2  SPADATA  DT  D2
DIRFILE  GO  D2  MEMTEST1 GO  B2  SPADATA  TX  D2

        DISK INDEX LISTING
        ==================

        DISK NAME             ------ FILES ----- . ---- SECTORS -----
                                USED  DEL  FREE .  USED  DEL  FREE
        BASIC SYSTEMS 1   D2     32     0    18 .   646    0   614
        UTILIES           B2     33     0    17 .   253    0  1007
                                ----  ---- ---- .  ----  ---- ----
                                 65     0    35 .   899    0  1621
                                ====  ==== ==== .  ====  ==== ====

        NUMBER OF DISKS ..:-     2
```

This program is split into two portions. Firstly there is a machine code program that gathers up the data and secondly a BASIC program that processes the output and sorts the data into order. The machine code section is loaded into RAM at £1000 and executed at £1000, the user is asked to load the disk into drive 0 and press enter, this reads into RAM the directory and is a very quick operation. When all the directorys of the disks have been read in then the user presses the `ESC´ key, you are asked to insert into drive 0 the disk which will contain the completed directory file. This will be the disk containing the BASIC program. The BASIC program should then be run, it reads the file generated by the machine code programme, prints it out, sorts it into alphabetical order and prints it out again. After this it prints out the disk summary. The BASIC program is able to remove certain files name from the listings, for example all disks have the file `Exec´ so there is no need to list it out. The user can change or add to lines 1260 - 1480 which is where all the unwanted file names are removed. The file produced by the machine code program can be viewed using the Polydos LIST option or changed using the EDIT facility (be careful as this may well stop the BASIC program operating correctly).

A word of warning; as the directory data is held in RAM then there must be a limit to how many disks can be used although I have used 40 disks without any trouble. Should this be a limitation then the job should be split up into sections and the BASIC program changed to merge several files together. Alternatively change the machine code program to remove unwanted files which would significantly reduce the ammount of data in RAM.

There follows a full Assembly listing including a sorted symbol table, also a dump of the program using a modified version of the disk dump published in Vol.1 issue 2 of 80-BUS NEWS (the numbers on the left hand side are the RAM locations). The BASIC program is also included.

```
PolyZap V2.2    ASSEMBLER    PAGE  1
----------------------------------------------------------------
0000            ;*********************************
0000            ;* CREATE A DIRECTORY FILE ON DISK *
0000            ;* =============================== *
0000            ;*                                 *
0000            ;* M.J.R.GIBBS 11-12-82            *
0000            ;*                                 *
0000            ;*********************************
0000                            ;NAS-SYS COMMAND
0018    NASSYS  EQU     £18
0060    ARGS    EQU     £60
007A    B1HEX   EQU     £7A
0068    B2HEX   EQU     £68
007B    BLINK   EQU     £7B
007C    CPOS    EQU     £7C
006A    CRLF    EQU     £6A
006B    ERRM    EQU     £6B
005E    FFLP    EQU     £5E
0063    INLIN   EQU     £63
0062    KBD     EQU     £62
005F    MFLP    EQU     £5F
005F    MOTFLP  EQU     £5F
005B    MRET    EQU     £5B
0064    NUM     EQU     £64
0079    RLIN    EQU     £79
005C    SCALJ   EQU     £5C
006D    SOUT    EQU     £6D
0069    SPACE   EQU     £69
0067    TBCD1   EQU     £67
0066    TBCD3   EQU     £66
005D    TDEL    EQU     £5D
006C    TXI     EQU     £6C
0071    NOM     EQU     £71
0072    NIM     EQU     £72
0077    NNOM    EQU     £77
0078    NNIM    EQU     £78
0000                            ;OTHER RESET COMMANDS
0020    BRKPT   EQU     £20
0008    CHIN    EQU     £08
0030    CRT     EQU     £30
0028    PRS     EQU     £28
0010    RCAL    EQU     £10
0030    ROUT    EQU     £30
0008    RIN     EQU     £08
0000                            ;
0000                            ;CONTROL CHARACTERS
0000                            ;
0008    BACKSP  EQU     £08
000C    CLEAR   EQU     £0C
000D    CRET    EQU     £0D
001B    ESC     EQU     £1B
```

```
PolyZap V2.2    ASSEMBLER     PAGE  2
------------------------------------
0000                                    ;
0000                                    ;DISK COMMANDS
0000                                    ;
0080        ZDSIZE  EQU     £80         ;DISK SIZE
0081        ZDRD    EQU     £81         ;READ
0082        ZDWR    EQU     £82         ;WRITE
0083        ZRDIR   EQU     £83         ;READ DIRECTORY
0084        ZWDIR   EQU     £84         ;WRITE DIRECTORY
0085        ZCFS    EQU     £85         ;CONVET FILE SPECIFIER
0087        ZENTER  EQU     £87         ;UPDATE DIRECTORY
0088        ZCOV    EQU     £88         ;CALL OVERLAY
0089        ZCOVR   EQU     £89         ;CALL OVERLAY RESTORE
008A        ZCKER   EQU     £8A         ;CHECK FOR ERRORS
008B        ZCBRK   EQU     £8B         ;CHECK FOR BREAK
008C        ZCFMA   EQU     £8C         ;ABORT COMMAND MODE
0086        ZLOOK   EQU     £86         ;LOOKUP FILE DIRECTORY
0063        ZINLIN  EQU     £63         ;MONITOR INPUT LINE
008E        ZJUMP   EQU     £8E         ;JUMP ROUTINE
008F        POUT    EQU     £8F         ;PRINTER OUTPUT
```

```
DUMP V1.0       DUMP DIRECTORY PROGRAM                          PAGE 02

1380  C9 75 00 10 27 E8 03 64  00 0A 00 01 00 00 00 00  .u..´..d........
1390  00 00 00 00 00 49 4E 44  58 44 41 54 41 44 54 00  .....INDXDATADT.
13A0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

```
NASCOM BASIC LISTING ..... PAGE :- 04
-------------------------------------

20000 PRINT CHR$(4);"        DISK INDEX LISTING"
20010 PRINT CHR$(4);"        =================="
20020 RETURN
```

```
PolyZap V2.2    ASSEMBLER     PAGE  3
------------------------------------
0000                                    ;
0000                                    ;POLYDOS WORKAREA
0000                                    ;
C000                ORG     £C000
C000 + 0001 MDRV    DS      1           ;MASTER DIRECTORY DRIVE
C001 + 0001 DDRV    DS      1           ;MASTER DIRECTORY NUMBER
C002 + 0001 DRVCOD  DS      1           ;DRIVE  CODE
C003 + 0001 FIRST   DS      1           ;POWER  UP FLAG
C004 + 0001 ERRFLG  DS      1           ;ERROR  FLAG
C005 + 0001 ERRCOD  DS      1           ;ERROR  CODE
C006 + 0002 BREAK   DS      2           ;BREAK  ADDRESS
C008 + 0002 BRAM    DS      2           ;RAM    BUFFER ADDRESS
C00A + 0001 BNSC    DS      1           ;RAM    BUFFER LENGTH
C00B + 0001 CFFLG   DS      1           ;COMMAND FILE FLAG
C00C + 0001 CFDRV   DS      1           ;COMMAND FILE DRIVE
C00D + 0002 CFSEC   DS      2           ;COMMAND SECTOR ADDRESS
C00F + 0001 CFNSC   DS      1           ;COMMAND SECTOR COUNTER
C010 + 0001 CFSBP   DS      1           ;COMMAND SECTOR POINTER
C011 + 0001 RKROW   DS      1           ;KEYBOARD ROW
C012 + 0001 RKBIT   DS      1           ;KEYBOARD BIT MASK
C013 + 0001 RKVAL   DS      1           ;REPEAT KEY VALUE
C014 + 0002 RKCNT   DS      2           ;REPEAT DELAY COUNT
C016 + 0001 BLINKF  DS      1           ;BLINK  ROUTINE FLAG
C017 + 0001 PLCT    DS      1           ;PRINT  LINE COUNTER
C018 + 0001 PPOS    DS      1           ;PRINT  HEAD POSITION
C019 + 0002 CLINP   DS      2           ;COMMAND LINE POINTER
C01B + 0030 CLIN    DS      48          ;COMMAND LINE BUFFER
C04B + 000A OVFCB   DS      10          ;OVERLAY FCB
C055        S1FCB   EQU     $
C055 + 0008 F1NAM   DS      8           ;FILE   NAME
C05D + 0002 F1EXT   DS      2           ;FILE   EXTENSION
C05F + 0001 F1SFL   DS      1           ;SYSTEM FLAG
C060 + 0001 F1UFL   DS      1           ;USER   FLAG
C061 + 0002 F1SEC   DS      2           ;SECTOR ADDRESS
C063 + 0002 F1NSC   DS      2           ;NUMBER SECTORS
C065 + 0002 F1LDA   DS      2           ;LOAD   ADDRESS
C067 + 0002 F1EXA   DS      2           ;EXEC   ADDRESS
C069        S2FCB   EQU     $
C069 + 0008 F2NAM   DS      8           ;FILE   NAME
C071 + 0002 F2EXT   DS      2           ;FILE   EXTENSION
C073 + 0001 F2SFL   DS      1           ;SYSTEM FLAG
C074 + 0001 F2UFL   DS      1           ;USER   FLAG
C075 + 0002 F2SEC   DS      2           ;SECTOR NUMBER
C077 + 0002 F2NSC   DS      2           ;NUMBER SECTORS
C079 + 0002 F2LDA   DS      2           ;LOAD   ADDRESS
C07B + 0002 F2EXA   DS      2           ;EXEC   ADDRESS
C07D + 0006 DSKWSP  DS      6           ;DISK   I-O WORKSPACE
C083 + 003D SYSWSP  DS      61          ;MISC   COMMAND WORKSP
C0C0                                    ;
C0C0                                    ;DIRECTORY BUFFER
C0C0                                    ;
C400        DIRBUF  EQU     £C400       ;DIRECTORY BUFFER
C414        NXTSEC  EQU     £C414       ;NEXT AVAILABLE SECTOR
C416        NXTFCB  EQU     £C416       ;A(NEXT FCB)
C418        FCBS    EQU     £C418       ;50 FCBS
```

```
PolyZap V2.2     ASSEMBLER     PAGE  4
--------------------------------------
1000              START   ORG    £1000
1000                      IDNT   £1000,£1000
1800              OUTLOC  EQU    £1800
1000 DD210018             LD     IX,OUTLOC          ;LOAD OUTPUT AREA
1004 217B13               LD     HL,OUTPUT
1007 22780C               LD     (£0C78),HL         ;POINT TO USER OUTPUT
100A CDDC12       LOOP    CALL   PAGE
100D EF                   RST    PRS
100E 20202020             DB     ´      Load Disk in Drive 0 Press´
1012 204C6F61
1016 64204469
101A 736B2069
101E 6E204472
1022 69766520
1026 30205072
102A 657373
102D 2022456E             DB     ´ "Enter ".´,CRET,CRET
1031 74657220
1035 222E0D0D
1039 20202020             DB     ´      To End the Programme Press´
103D 20546F20
1041 456E6420
1045 74686520
1049 50726F67
104D 72616D6D
1051 65205072
1055 657373
1058 20224573             DB     ´ "Escape".´,0
105C 63617065
1060 222E00
1063 DF                   RST    NASSYS
1064 7B                   DB     BLINK
1065 FE1B                 CP     ESC                ;IS ESCAPE
1067 2806                 JR     Z,ENDIT
1069 CD4B11               CALL   DIR
106C C30A10               JP     LOOP               ;NEXT DISK
106F 218113       ENDIT   LD     HL,OUTTAB          ;FLIP OUTPUT
1072 DF                   RST    NASSYS
1073 71                   DB     NOM
1074 EF                   RST    PRS
1075 0D                   DB     CRET
1076 454F460D             DB     ´EOF´,CRET,0
107A 00
107B DF                   RST    NASSYS
107C 77                   DB     NNOM               ;BACK TO NORMAL
107D 3E00                 LD     A,0
107F 0600                 LD     B,0
1081 DD7700       L10     LD     (IX+0),A           ;OUTPUT NULLS
1084 DD23                 INC    IX
1086 10F9                 DJNZ   L10
1088 CDDC12       L20     CALL   PAGE
108B EF                   RST    PRS
108C 20202020             DB     ´            ´
1090 20202020
1094 20202020
1098 4C6F6164             DB     ´Load File Insert disk.´
```

```
PolyZap V2.2     ASSEMBLER     PAGE  5
--------------------------------------
109C 2046696C
10A0 6520496E
10A4 73657274
10A8 20646973
10AC 6B2E
10AE 0D                   DB     CRET
10AF 20202020             DB     ´
10B3 20202020
10B7 20202020
10BB 2D2D2D2D             DB     ´---------------------´
10BF 2D2D2D2D
10C3 2D2D2D2D
10C7 2D2D2D2D
10CB 2D2D2D2D
10CF 2D2D
10D1 00                   DB     0
10D2 DF                   RST    NASSYS
10D3 7B                   DB     BLINK
10D4 3EFF                 LD     A,£FF
10D6 3201C0               LD     (DDRV),A           ;FORCE DIR READ
10D9 AF                   XOR    A                  ;DIRECTORY DRIVE IN 0
10DA 4F                   LD     C,A                ;READ DIRECTORY
10DB DF                   RST    NASSYS
10DC 83                   DB     ZRDIR              ;READ DIRECTORY
10DD 20A9                 JR     NZ,L20             ;ERROR TRY AGAIN
10DF DDE5                 PUSH   IX                 ;CALC NUMBER OF SECTORS
10E1 E1                   POP    HL                 ;HL = A(END)
10E2 110018               LD     DE,OUTLOC          ;DE = A(START)
10E5 B7                   OR     A
10E6 ED52                 SBC    HL,DE              ;HL = L(DATA)
10E8 E5                   PUSH   HL                 ;KEEP LENGTH
10E9 210018               LD     HL,OUTLOC          ;HL = A(START)
10EC ED5B14C4             LD     DE,(NXTSEC)        ;DE = A(NEXT FREE SECTR)
10F0 ED53A113             LD     (FXSEC),DE         ;LOAD FCB
10F4 C1                   POP    BC                 ;RECOVER NUMBER OF SECTS
10F5 48                   LD     C,B                ;C  = NUMBER OF SECTORS
10F6 0600                 LD     B,0                ;B  = 0
10F8 ED43A313             LD     (FXNSC),BC         ;LOAD FCB
10FC 41                   LD     B,C                ;B  = NUMBER OF SECTORS
10FD 0E00                 LD     C,0                ;C  = DDRV ie 0
10FF DF                   RST    NASSYS
1100 82                   DB     ZDWR               ;WRITE IT OUT
1101 DF                   RST    NASSYS
1102 8A                   DB     ZCKER
1103 219513               LD     HL,FXFCB           ;HL = A(FCB)
1106 DF           ENTER   RST    NASSYS
1107 87                   DB     ZENTER             ;TRY  ENTER IN DIRECTORY
1108 283F                 JR     Z,STOPIT
110A FE31                 CP     £31                ;FILE EXISTS
110C 2802                 JR     Z,ENT20
110E DF           ENT10   RST    NASSYS
110F 8A                   DB     ZCKER              ;ANOTHER ERROR
1110 E5           ENT20   PUSH   HL                 ;KEEP HL
1111 210A00               LD     HL,FXSFL-FXNAM     ;REL POS OF SYSTEM
1114 19                   ADD    HL,DE              ;FIND SAME NAME FCB
1115 CB46                 BIT    0,(HL)             ;IS  IT LOCKED
```

```
PolyZap V2.2    ASSEMBLER    PAGE  6
------------------------------------
1117 3E33               LD      A,£33
1119 202B               JR      NZ,ENT30        ;YES  STOP
111B EF                 RST     PRS
111C 0D                 DB      CRET
111D 20202020           DB                   <= OLD FILE DELETED =>'
1121 20202020
1125 20202020
1129 3C3D204F
112D 4C442046
1131 494C4520
1135 44454C45
1139 54454420
113D 3D3E
113F 0D00               DB      CRET,0
1141 CBCE               SET     1,(HL)          ;DELETE OLD FILE
1143 E1                 POP     HL              ;RECOVER
1144 18C0               JR      ENTER           ;TRY AGAIN
1146 E1         ENT30   POP     HL              ;RECOVER HL
1147 DF                 RST     NASSYS
1148 8A                 DB      ZCKER           ;CANNOT OVERWRITE LOCKED
1149 DF         STOPIT  RST     NASSYS
114A 5B                 DB      MRET            ;RET TO NASSYS
114B 183D       DIR     JR      DIR20           ;JUMP OVER ERROR
114D CDDC12     DIR10   CALL    PAGE
1150 EF                 RST     PRS
1151 0D                 DB      CRET
1152 20202020           DB      '      Disk not Loaded Correctly'
1156 20446973
115A 6B206E6F
115E 74204C6F
1162 61646564
1166 20436F72
116A 72656374
116E 6C79
1170 20547279           DB      ' Try AGAIN',0
1174 20414741
1178 494E00
117B DF                 RST     NASSYS
117C 7B                 DB      BLINK           ;GET  REPLY
117D F5                 PUSH    AF              ;KEEP REPLY
117E EF                 RST     PRS
117F 0D00               DB      CRET,0
1181 F1                 POP     AF              ;RECOVER REPLY
1182 FE1B               CP      ESC             ;IS   ESC
1184 2004               JR      NZ,DIR20        ;NO
1186 F1                 POP     AF
1187 C36F10             JP      ENDIT           ;TERMINATE
118A 3EFF       DIR20   LD      A,£FF           ;FORCE DIRECTORY READ
118C 3201C0             LD      (DDRV),A
118F EF                 RST     PRS
1190 0C00               DB      CLEAR,0
1192 3E00               LD      A,0             ;READ DIRECTORY DRIVE 0
1194 4F                 LD      C,A             ;C  = DRIVE NUMBER
1195 DF                 RST     NASSYS
1196 83                 DB      ZRDIR           ;READ DIRECTORY
1197 20B4               JR      NZ,DIR10
```

```
PolyZap V2.2    ASSEMBLER    PAGE  7
------------------------------------
1199 218113             LD      HL,OUTTAB       ;SETUP OUTPUT TABLES
119C DF                 RST     NASSYS
119D 71                 DB      NOM             ;CHANGE TO LOAD RAM
119E 0673               LD      B,£73           ;SCAN
11A0 C5         DIR30   PUSH    BC
11A1 EF                 RST     PRS
11A2 44726976           DB      'Drive ',00
11A6 652000
11A9 3A01C0             LD      A,(DDRV)
11AC C630               ADD     A,'0'
11AE F7                 RST     CRT
11AF EF                 RST     PRS
11B0 3A20               DB      ': '
11B2 00                 DB      0
11B3 0614               LD      B,20
11B5 2100C4             LD      HL,DIRBUF
11B8 7E         DIR40   LD      A,(HL)
11B9 F7                 RST     CRT
11BA 23                 INC     HL
11BB 10FB               DJNZ    DIR40
11BD DF                 RST     NASSYS
11BE 6A                 DB      CRLF
11BF 218D13             LD      HL,NUMFL
11C2 0608               LD      B,8
11C4 3600       DIR50   LD      (HL),0
11C6 23                 INC     HL
11C7 10FB               DJNZ    DIR50
11C9 0673               LD      B,£73
11CB 2169C0     DIR60   LD      HL,S2FCB
11CE DF                 RST     NASSYS
11CF 86                 DB      ZLOOK           ;LOOKUP FILE DIRECTORY
11D0 2027               JR      NZ,DIR90
11D2 D5                 PUSH    DE
11D3 3A73C0             LD      A,(F2SFL)
11D6 ED5B77C0           LD      DE,(F2NSC)
11DA CB4F               BIT     1,A             ;DELETED
11DC 200D               JR      NZ,DIR70        ;YES
11DE 218D13             LD      HL,NUMFL        ;INCREMENT NUMBER FILES
11E1 34                 INC     (HL)
11E2 2A9113             LD      HL,(NUMSC)
11E5 19                 ADD     HL,DE           ;ADD  NUMBER SECTORS
11E6 229113             LD      (NUMSC),HL
11E9 180B               JR      DIR80
11EB 218F13     DIR70   LD      HL,NUMFLD
11EE 34                 INC     (HL)            ;INCREMENT NUMBER FILES
11EF                                            ;      DELETED
11EF 2A9313             LD      HL,(NUMSCD)
11F2 19                 ADD     HL,DE           ;ADD  NUMBER OF SECTORS
11F3                                            ;      DELETED
11F3 229313             LD      (NUMSCD),HL
11F6 D1         DIR80   POP     DE
11F7 18D2               JR      DIR60
11F9 2A8D13     DIR90   LD      HL,(NUMFL)
11FC E5                 PUSH    HL
11FD CD4F13             CALL    PRTNUM
1200 EF                 RST     PRS
```

```
1201 2046696C          DB      ' Files   used, ',00
1205 65732020
1209 20757365
120D 642C2000
1211 2A8F13            LD      HL,(NUMFLD)
1214 E5                PUSH    HL
1215 CD4F13            CALL    PRTNUM
1218 EF                RST     PRS
1219 2044656C          DB      ' Deleted, ',00
121D 65746564
1221 2C2000
1224 E1                POP     HL
1225 D1                POP     DE
1226 19                ADD     HL,DE
1227 EB                EX      DE,HL
1228 213200            LD      HL,50           ;TOTAL NUMBER OF FILES
122B B7                OR      A
122C ED52              SBC     HL,DE
122E CD4F13            CALL    PRTNUM
1231 EF                RST     PRS
1232 20467265          DB      ' Free.',CRET,00
1236 652E0D00
123A 2A9113            LD      HL,(NUMSC)
123D 110400            LD      DE,4
1240 19                ADD     HL,DE
1241 E5                PUSH    HL
1242 CD4F13            CALL    PRTNUM
1245 EF                RST     PRS
1246 20536563          DB      ' Sectors used, ',00
124A 746F7273
124E 20757365
1252 642C2000
1256 2A9313            LD      HL,(NUMSCD)
1259 E5                PUSH    HL
125A CD4F13            CALL    PRTNUM
125D EF                RST     PRS
125E 2044656C          DB      ' Deleted, ',00
1262 65746564
1266 2C2000
1269 E1                POP     HL
126A D1                POP     DE
126B 19                ADD     HL,DE
126C EB                EX      DE,HL
126D 3A01C0            LD      A,(DDRV)
1270 4F                LD      C,A
1271 DF                RST     NASSYS
1272 80                DB      ZDSIZE          ;DISK SIZE
1273 B7                OR      A
1274 ED52              SBC     HL,DE
1276 CD4F13            CALL    PRTNUM
1279 EF                RST     PRS
127A 20467265          DB      ' Free.',CRET
127E 652E0D
1281 53656374          DB      'Sect Nsec Load Exec F Name'
1285 204E7365
1289 63204C6F
```



```
128D 61642045
1291 78656320
1295 46204E61
1299 6D65
129B 0D00                      DB      CRET,00
129D C1                        POP     BC
129E 0E0B                      LD      C,£0B
12A0 2155C0          DIR100    LD      HL,S1FCB
12A3 DF                        RST     NASSYS
12A4 86                        DB      ZLOOK           ;LOOKUP FILE DIRECTORY
12A5 2032                      JR      NZ,DIR150
12A7 C5                        PUSH    BC
12A8 D5                        PUSH    DE
12A9 210C00                    LD      HL,12
12AC 19                        ADD     HL,DE
12AD 0604                      LD      B,4
12AF 5E              DIR110    LD      E,(HL)          ;PRINT 4 PARMS
12B0 23                        INC     HL
12B1 56                        LD      D,(HL)
12B2 23                        INC     HL
12B3 EB                        EX      DE,HL
12B4 DF                        RST     NASSYS
12B5 66                        DB      TBCD3
12B6 EB                        EX      DE,HL
12B7 10F6                      DJNZ    DIR110
12B9 11F6FF                    LD      DE,-10
12BC 19                        ADD     HL,DE
12BD 3E20                      LD      A,' '
12BF CB4E                      BIT     1,(HL)
12C1 2804                      JR      Z,DIR120
12C3 3E44                      LD      A,'D'           ;DELETED FILE
12C5 1806                      JR      DIR130
12C7 CB46            DIR120    BIT     0,(HL)
12C9 2802                      JR      Z,DIR130
12CB 3E4C                      LD      A,'L'           ;LOCKED FILE
12CD F7              DIR130    RST     CRT
12CE DF                        RST     NASSYS
12CF 69                        DB      SPACE
12D0 D1                        POP     DE
12D1 C1                        POP     BC
12D2 CD2B13                    CALL    LSTFIL
12D5 DF              DIR140    RST     NASSYS
12D6 6A                        DB      CRLF
12D7 18C7                      JR      DIR100
12D9 DF              DIR150    RST     NASSYS
12DA 77                        DB      NNOM            ;NORMAL I-O
12DB C9                        RET
```

```
PolyZap V2.2    ASSEMBLER    PAGE 10
------------------------------------
12DC EF          PAGE    RST    PRS            ;--- PAGE HEADINGS ---
12DD 0C                  DB     CLEAR
12DE 20202020            DB     '           Disk Directory File Load'
12E2 20202020
12E6 20202044
12EA 69736B20
12EE 44697265
12F2 63746F72
12F6 79204669
12FA 6C65204C
12FE 6F6164
1301 0D                  DB     CRET
1302 20202020            DB     '           ========================'
1306 20202020
130A 2020203D
130E 3D3D3D3D
1312 3D3D3D3D
1316 3D3D3D3D
131A 3D3D3D3D
131E 3D3D3D3D
1322 3D3D3D
1325 0D0D0D0D            DB     CRET,CRET,CRET,CRET,0
1329 00
132A C9                  RET


132B D5          LSTFIL  PUSH   DE             ;--- LIST FILE ---
132C C5                  PUSH   BC
132D 01000A              LD     BC,£0A00
1330 1A          LST10   LD     A,(DE)
1331 13                  INC    DE
1332 FE20                CP     ' '
1334 F7                  RST    CRT
1335 0C                  INC    C
1336 78                  LD     A,B
1337 FE03                CP     £03
1339 2004                JR     NZ,LST20
133B 3E2E                LD     A,'.'
133D F7                  RST    CRT
133E 0C                  INC    C
133F 10EF        LST20   DJNZ   LST10
1341 79                  LD     A,C
1342 C1                  POP    BC
1343 D1                  POP    DE
1344 C9                  RET


1345 3E3A        PRTDRV  LD     A,':'          ;--- PRINT DRIVE NO ---
1347 F7                  RST    CRT
1348 3A01C0              LD     A,(DDRV)
```

```
PolyZap V2.2    ASSEMBLER    PAGE 11
------------------------------------
134B C630                ADD    A,'0'
134D F7                  RST    CRT
134E C9                  RET
134F 010004      PRTNUM  LD     BC,£0400       ;-- PRINT NO DECIMAL --
1352 118513              LD     DE,TENS+2
1355 D5                  PUSH   DE
1356 E3          PRTN10  EX     (SP),HL
1357 5E                  LD     E,(HL)
1358 23                  INC    HL
1359 56                  LD     D,(HL)
135A 23                  INC    HL
135B E3                  EX     (SP),HL
135C AF                  XOR    A
135D 3C          PRTN20  INC    A
135E ED52                SBC    HL,DE
1360 30FB                JR     NC,PRTN20
1362 19                  ADD    HL,DE
1363 3D                  DEC    A
1364 200D                JR     NZ,PRTN30
1366 0C                  INC    C
1367 0D                  DEC    C
1368 2009                JR     NZ,PRTN30
136A 78                  LD     A,B
136B 3D                  DEC    A
136C 2805                JR     Z,PRTN30       ;MUST PRINT LAST 0
136E 3E20                LD     A,' '          ;REPLACE WITH BLANK
1370 F7                  RST    CRT            ;PRINT IT
1371 1804                JR     PRTN40         ;NEXT DIGIT
1373 0D          PRTN30  DEC    C
1374 C630                ADD    A,'0'
1376 F7                  RST    CRT
1377 10DD        PRTN40  DJNZ   PRTN10
1379 D1                  POP    DE
137A C9                  RET
137B DD7700      OUTPUT  LD     (IX+0),A       ;--- OUTPUT LOAD RAM ---
137E DD23                INC    IX
1380 C9                  RET
1381 7500        OUTTAB  DB     £75,0
```

```
PolyZap V2.2      ASSEMBLER      PAGE  12
----------------------------------------
1383 1027          TENS      DW        10000
1385 E803                    DW        1000
1387 6400                    DW        100
1389 0A00                    DW        10
138B 0100                    DW        1
138D 0000          NUMFL     DW        0
138F 0000          NUMFLD    DW        0
1391 0000          NUMSC     DW        0
1393 0000          NUMSCD    DW        0


1395               FXFCB     EQU       $
1395 494E4458      FXNAM     DB        'INDXDATA'
1399 44415441
139D 4454          FXEXT     DB        'DT'
139F 00            FXSFL     DB        0
13A0 00            FXUFL     DB        0
13A1 0000          FXSEC     DW        0
13A3 0000          FXNSC     DW        0
13A5 0000          FXLDA     DW        0
13A7 0000          FXEDA     DW        0
```

```
PolyZap V2.2      ASSEMBLER      PAGE  13
----------------------------------------
ARGS     0060  B1HEX    007A  B2HEX    0068  BACKSP   0008  BLINK    007B
BLINKF   C016  BNSC     C00A  BRAM     C008  BREAK    C006  BRKPT    0020
CFDRV    C00C  CFFLG    C00B  CFNSC    C00F  CFSBP    C010  CFSEC    C00D
CHIN     0008  CLEAR    000C  CLIN     C01B  CLINP    C019  CPOS     007C
CRET     000D  CRLF     006A  CRT      0030  DDRV     C001  DIR      114B
DIR10    114D  DIR100   12A0  DIR110   12AF  DIR120   12C7  DIR130   12CD
DIR140   12D5  DIR150   12D9  DIR20    118A  DIR30    11A0  DIR40    11B8
DIR50    11C4  DIR60    11CB  DIR70    11EB  DIR80    11F6  DIR90    11F9
DIRBUF   C400  DRVCOD   C002  DSKWSP   C07D  ENDIT    106F  ENT10    110E
ENT20    1110  ENT30    1146  ENTER    1106  ERRCOD   C005  ERRFLG   C004
ERRM     006B  ESC      001B  F1EXA    C067  F1EXT    C05D  F1LDA    C065
F1NAM    C055  F1NSC    C063  F1SEC    C061  F1SFL    C05F  F1UFL    C060
F2EXA    C07B  F2EXT    C071  F2LDA    C079  F2NAM    C069  F2NSC    C077
F2SEC    C075  F2SFL    C073  F2UFL    C074  FCBS     C418  FFLP     005E
FIRST    C003  FXEDA    13A7  FXEXT    139D  FXFCB    1395  FXLDA    13A5
FXNAM    1395  FXNSC    13A3  FXSEC    13A1  FXSFL    139F  FXUFL    13A0
INLIN    0063  KBD      0062  L10      1081  L20      1088  LOOP     100A
LST10    1330  LST20    133F  LSTFIL   132B  MDRV     C000  MFLP     005F
MOTFLP   005F  MRET     005B  NASSYS   0018  NIM      0072  NNIM     0078
NNOM     0077  NOM      0071  NUM      0064  NUMFL    138D  NUMFLD   138F
NUMSC    1391  NUMSCD   1393  NXTFCB   C416  NXTSEC   C414  OUTLOC   1800
OUTPUT   137B  OUTTAB   1381  OVFCB    C04B  PAGE     12DC  PLCT     C017
POUT     008F  PPOS     C018  PRS      0028  PRTDRV   1345  PRTN10   1356
PRTN20   135D  PRTN30   1373  PRTN40   1377  PRTNUM   134F  RCAL     0010
RIN      0008  RKBIT    C012  RKCNT    C014  RKROW    C011  RKVAL    C013
RLIN     0079  ROUT     0030  S1FCB    C055  S2FCB    C069  SCALJ    005C
SOUT     006D  SPACE    0069  START    1000  STOPIT   1149  SYSWSP   C083
TBCD1    0067  TBCD3    0066  TDEL     005D  TENS     1383  TXI      006C
ZCBRK    008B  ZCFMA    008C  ZCFS     0085  ZCKER    008A  ZCOV     0088
ZCOVR    0089  ZDRD     0081  ZDSIZE   0080  ZDWR     0082  ZENTER   0087
ZINLIN   0063  ZJUMP    008E  ZLOOK    0086  ZRDIR    0083  ZWDIR    0084
```

```
DUMP V1.0          DUMP DIRECTORY PROGRAM                          PAGE 01

1000  DD 21 00 18 21 7B 13 22  78 0C CD DC 12 EF 20 20  .!..!{."x.....
1010  20 20 20 4C 6F 61 64 20  44 69 73 6B 20 69 6E 20     Load Disk in
1020  44 72 69 76 65 20 30 20  50 72 65 73 73 20 22 45  Drive 0 Press "E
1030  6E 74 65 72 20 22 2E 0D  0D 20 20 20 20 20 54 6F  nter "...     To
1040  20 45 6E 64 20 74 68 65  20 50 72 6F 67 72 61 6D   End the Program
1050  6D 65 20 50 72 65 73 73  20 22 45 73 63 61 70 65  me Press "Escape
1060  22 2E 00 DF 7B FE 1B 28  06 CD 4B 11 C3 0A 10 21  "...{..(..K....!
1070  81 13 DF 71 EF 0D 45 4F  46 0D 00 DF 77 3E 00 06  ...q..EOF...w>..
1080  00 DD 77 00 DD 23 10 F9  CD DC 12 EF 20 20 20 20  ..w..£......
1090  20 20 20 20 20 20 20 20  4C 6F 61 64 20 46 69 6C          Load Fil
10A0  65 20 49 6E 73 65 72 74  20 64 69 73 6B 2E 0D 20  e Insert disk..
10B0  20 20 20 20 20 20 20 20  20 20 20 2D 2D 2D 2D 2D             -----
10C0  2D 2D 2D 2D 2D 2D 2D 2D  2D 2D 2D 2D 2D 2D 2D 2D  ----------------
10D0  2D 00 DF 7B 3E FF 32 01  C0 AF 4F DF 83 20 A9 DD  -..{>.2...O.. ..
10E0  E5 E1 11 00 18 B7 ED 52  E5 21 00 18 ED 5B 14 C4  .......R.!...[..
10F0  ED 53 A1 13 C1 48 06 00  ED 43 A3 13 41 0E 00 DF  .S...H...C..A...
1100  82 DF 8A 21 95 13 DF 87  28 3F FE 31 28 02 DF 8A  ...!....(?.1(...
1110  E5 21 0A 00 19 CB 46 3E  33 20 2B EF 0D 20 20 20  .!....F>3 +..
1120  20 20 20 20 20 20 20 20  20 3C 3D 20 4F 4C 44 20           <= OLD
1130  46 49 4C 45 20 44 45 4C  45 54 45 44 20 3D 3E 0D  FILE DELETED =>.
1140  00 CB CE E1 18 C0 E1 DF  8A DF 5B 18 3D CD DC 12  ..........[.=...
1150  EF 0D 20 20 20 20 20 44  69 73 6B 20 6E 6F 74 20  ..     Disk not
1160  4C 6F 61 64 65 64 20 43  6F 72 72 65 63 74 6C 79  Loaded Correctly
1170  20 54 72 79 20 41 47 41  49 4E 00 DF 7B F5 EF 0D   Try AGAIN..{...
1180  00 F1 FE 1B 20 04 F1 C3  6F 10 3E FF 32 01 C0 EF  .... ...o.>.2...
1190  0C 00 3E 00 4F DF 83 20  B4 21 81 13 DF 71 06 73  ..>.O.. .!...q.s
11A0  C5 EF 44 72 69 76 65 20  00 3A 01 C0 C6 30 F7 EF  ..Drive .:...0..
11B0  3A 20 00 06 14 21 00 C4  7E F7 23 10 FB DF 6A 21  : ...!..~.£...j!
11C0  8D 13 06 08 36 00 23 10  FB 06 73 21 69 C0 DF 86  ....6.£...s!i...
11D0  20 27 D5 3A 73 C0 ED 5B  77 C0 CB 4F 20 0D 21 8D   ':s..[w..O .!.
11E0  13 34 2A 91 13 19 22 91  13 18 0B 21 8F 13 34 2A  .4*..."....!..4*
11F0  93 13 19 22 93 13 D1 18  D2 2A 8D 13 E5 CD 4F 13  ..."......*....O.
1200  EF 20 46 69 6C 65 73 20  20 20 75 73 65 64 2C 20  . Files   used,
1210  00 2A 8F 13 E5 CD 4F 13  EF 20 44 65 6C 65 74 65  .*....O.. Delete
1220  64 2C 20 00 E1 D1 19 EB  21 32 00 B7 ED 52 CD 4F  d, .....!2...R.O
1230  13 EF 20 46 72 65 65 2E  0D 00 2A 91 13 11 04 00  .. Free...*.....
1240  19 E5 CD 4F 13 EF 20 53  65 63 74 6F 72 73 20 75  ...O.. Sectors u
1250  73 65 64 2C 20 00 2A 93  13 E5 CD 4F 13 EF 20 44  sed, .*....O.. D
1260  65 6C 65 74 65 64 2C 20  00 E1 D1 19 EB 3A 01 C0  eleted, .....:..
1270  4F DF 80 B7 ED 52 CD 4F  13 EF 20 46 72 65 65 2E  O....R.O.. Free.
1280  0D 53 65 63 74 20 4E 73  65 63 20 4C 6F 61 64 20  .Sect Nsec Load
1290  45 78 65 63 20 46 20 4E  61 6D 65 0D 00 C1 0E 0B  Exec F Name.....
12A0  21 55 C0 DF 86 20 32 C5  D5 21 0C 00 19 06 04 5E  !U... 2..!.....^
12B0  23 56 23 EB DF 66 EB 10  F6 11 F6 FF 19 3E 20 CB  £V£..f.......> .
12C0  4E 28 04 3E 44 18 06 CB  46 28 02 3E 4C F7 DF 69  N(.>D...F(.>L..i
12D0  D1 C1 CD 2B 13 DF 6A 18  C7 DF 77 C9 EF 0C 20 20  ...+..j...w...
12E0  20 20 20 20 20 20 20 20  20 44 69 73 6B 20 44 69           Disk Di
12F0  72 65 63 74 6F 72 79 20  46 69 6C 65 20 4C 6F 61  rectory File Loa
1300  64 0D 20 20 20 20 20 20  20 20 20 20 20 3D 3D 3D  d.           ===
1310  3D 3D 3D 3D 3D 3D 3D 3D  3D 3D 3D 3D 3D 3D 3D 3D  ================
1320  3D 3D 3D 3D 3D 0D 0D 0D  0D 00 C9 D5 C5 01 00 0A  =====...........
1330  1A 13 FE 20 F7 0C 78 FE  03 20 04 3E 2E F7 0C 10  ... ..x.. .>....
1340  EF 79 C1 D1 C9 3E 3A F7  3A 01 C0 C6 30 F7 C9 01  .y...>:.:...0...
1350  00 04 11 85 13 D5 E3 5E  23 56 23 E3 AF 3C ED 52  .......^£V£..<.R
1360  30 FB 19 3D 20 0D 0C 0D  20 09 78 3D 28 05 3E 20  0..= ... .x=(.>
1370  F7 18 04 0D C6 30 F7 10  DD D1 C9 DD 77 00 DD 23  .....0......w..£
```

NASCOM BASIC LISTING ..... PAGE :- 01

```
1000 REM Programme to read the Polydos  print
1010 REM of  disk directorys and  remove  the
1020 REM common  file  names  and  produce  a
1030 REM Print out of all files in disk order
1040 REM and also print a sorted list.
1050 REM
1060 REM                       M.J.R.GIBBS 2/11/82
1070 CLEAR 20000
1080 DIM ND$(50),DA(50,6)
1090 ND  = 1:REM NUMBER OF DISKS
1100 CLS
1110 SCREEN 12,16
1120 PRINT "DISK FILES INDEX PROGRAMME"
1130 SCREEN 12,1
1140 PRINT "=========================="
1150 BL$ = "                      "
1160 DIM AS$(1000)
1170 SETNEW (1),"INDXDATA"
1180 FOR I = 1 TO 1000
1190 SCREEN 12,8
1200 PRINT "Reading record no...:-";I
1210 SETINP (1),AS$( I)
1220 IF LEN(AS$( I)) = 0 GOTO 1210
1230 IF "EOF" = MID$(AS$( I),1,3) THEN 1680
1240 IF "Drive" = MID$(AS$( I),1,5) THEN 1540
1250 NN$ = MID$(AS$( I),23, 8)
1260 IF NN$ = "Exec     " THEN 1210
1270 IF NN$ = "Dfun     " THEN 1210
1280 IF NN$ = "Emsg     " THEN 1210
1290 IF NN$ = "Ecmd     " THEN 1210
1300 IF NN$ = "Edit     " THEN 1210
1310 IF NN$ = "Info     " THEN 1210
1320 IF NN$ = "BSfh     " THEN 1210
1330 IF NN$ = "Bsfh     " THEN 1210
1340 IF NN$ = "Bsut     " THEN 1210
1350 IF NN$ = "BSut     " THEN 1210
1360 IF NN$ = "BSdr     " THEN 1210
1370 IF NN$ = "FORMAT   " THEN 1210
1380 IF NN$ = "BACKUP   " THEN 1210
1390 IF NN$ = "SZAP     " THEN 1210
1400 IF NN$ = "PZAP     " THEN 1210
1410 IF NN$ = "Init     " THEN 1210
1420 IF NN$ = "PSfh     " THEN 1210
1430 IF NN$ = "Z2fh     " THEN 1210
1440 IF NN$ = "DPfh     " THEN 1210
1450 IF NN$ = "Info     " THEN 1210
1460 IF NN$ = "BASIC    " THEN 1210
1470 IF NN$ = "DISKPEN  " THEN 1210
1480 IF NN$ = "NASPAS   " THEN 1210
1490 EX$ = MID$(AS$( I),32,2)
1500 DS$ = MID$(AS$( I),21,1)
1510 AS$( I) = NN$+" "+EX$+" "+DS$+" "+DD$
1520 PRINT "                   ";AS$(I)
1530 GOTO 1670
1540 DD$ = MID$(AS$( I),28,2)
```

NASCOM BASIC LISTING ..... PAGE :- 02

```
1550 ND$(ND) = MID$(AS$( I),10,20)
1560 SETINP( 1),ZZ$
1570 DA(ND, 1) = VAL(MID$(ZZ$, 1,4))
1580 DA(ND, 2) = VAL(MID$(ZZ$,20,4))
1590 DA(ND, 3) = VAL(MID$(ZZ$,34,4))
1600 SETINP( 1),ZZ$
1610 DA(ND, 4) = VAL(MID$(ZZ$, 1,4))
1620 DA(ND, 5) = VAL(MID$(ZZ$,20,4))
1630 DA(ND, 6) = VAL(MID$(ZZ$,34,4))
1640 ND = ND + 1
1650 SETINP( 1),ZZ$
1660 GOTO 1210
1670 NEXT I
1680 REM         NOW PRINT IT ALL OUT AND SORT
1690 SETCLS(1)
1700 SCREEN 7,8
1710 CLS
1720 PRINT "Set Printer on press 'ENTER'..:-";
1730 INPUT AA$
1740 GOSUB 9000
1750 GOSUB 8000
1760 GOSUB 9000
1770 SETPRON
1780 GOSUB 20000
1790 PRINT
1800 PRINT SPC( 5);"DISK NAME            ";
1810 PRINT "------ FILES ----- . ";
1820 PRINT "---- SECTORS -----"
1830 PRINT SPC(25);
1840 PRINT "  USED    DEL  FREE . ";
1850 PRINT "  USED    DEL  FREE"
1860 ND = ND-1
1870 FOR I = 1 TO ND
1880 PRINT SPC( 5);ND$( I);
1890 PRINT RIGHT$("      "+STR$(DA( I, 1)),6);
1900 PRINT RIGHT$("      "+STR$(DA( I, 2)),6);
1910 PRINT RIGHT$("      "+STR$(DA( I, 3)),6);
1920 PRINT " . ";
1930 PRINT RIGHT$("      "+STR$(DA( I, 4)),6);
1940 PRINT RIGHT$("      "+STR$(DA( I, 5)),6);
1950 PRINT RIGHT$("      "+STR$(DA( I, 6)),6)
1960 NEXT I
1970 PRINT SPC(27);
1980 PRINT "----  ----  ---- .  ";
1990 PRINT "----  ----  ----"
2000 FOR I = 1 TO ND
2010 FOR J = 1 TO 6
2020 DA( 0, J) = DA( 0, J)+DA( I, J)
2030 NEXT J
2040 NEXT I
2050 PRINT SPC(25);
2060 PRINT RIGHT$("      "+STR$(DA( 0, 1)),6);
2070 PRINT RIGHT$("      "+STR$(DA( 0, 2)),6);
2080 PRINT RIGHT$("      "+STR$(DA( 0, 3)),6);
2090 PRINT " . ";
```

NASCOM BASIC LISTING ..... PAGE :- 03

```
2100 PRINT RIGHT$("      "+STR$(DA( 0, 4)),6);
2110 PRINT RIGHT$("      "+STR$(DA( 0, 5)),6);
2120 PRINT RIGHT$("      "+STR$(DA( 0, 6)),6)
2130 PRINT SPC(27);
2140 PRINT "====  ====  ==== .  ";
2150 PRINT "====  ====  ===="
2160 PRINT SPC( 5);"NUMBER OF DISKS ..:-";
2170 PRINT RIGHT$("      "+STR$(ND),6)
2180 CLS
2190 SETPROFF
2200 END
8000 REM SHELL SORT
8010 REM
8020 SCREEN 7, 8:PRINT "SHELL - METZNER SORT "
8030 REM
8040 SN = I-1
8050 SS = SN
8060 SS = INT(SS/2)
8070 IF SS = 0 THEN RETURN
8080 SCREEN 28, 8:PRINT "BLKSIZE :=";SS;"   ";
8090 SZ = SN-SS
8100 FOR SM = 1 TO SZ
8110 SI = SM
8120 SJ = SI+SS
8130 IF AS$(SI) <= AS$(SJ) THEN 8170
8140 SH$=AS$(SI):AS$(SI)=AS$(SJ):AS$(SJ)=SH$
8150 SI = SI-SS
8160 IF SI > 0 THEN 8120
8170 NEXT SM
8180 GOTO 8060
9000 SETPRON
9010 AS$( I) =""
9020 IN = INT((I-1)/200)+1
9030 FOR II = 1 TO IN
9040 IS = (II-1)*200+1
9050 IE = IS+200
9060 IF IE >= I-1 THEN IE = I+4
9070 GOSUB 10000
9080 NEXT II
9090 SETPROFF
9100 RETURN
10000 GOSUB 20000
10010 L  = INT((IE-IS)/4)
10020 IE = IS+4*L
10030 IF K > 200 THEN K = 200
10040 PRINT
10050 FOR J = IS TO IS+L-1
10060 PRINT AS$(J  );"  ";
10070 PRINT AS$(J + L);"  ";
10080 PRINT AS$(J+2*L);"  ";
10090 PRINT AS$(J+3*L)
10100 REM INPUT AA
10110 NEXT
10120 CLS
10130 RETURN
```

AMERSHAM COMPUTER CENTRE for MONITORS

COTRON SWORD

Cotron's progressive development of high technology TV monitors for professional users,has enabled them to produce computer monitors that few other manufacturers can match in both quality and price.

With the introduction of the Sword range, a new dimension is offered to micro users. All the Sword monitors incorporate 14" 'Black Glass' tubes, producing very high contrast with bright clean colours.

All Sword monitors will accept both TTL and analog input signals, via a 25 pin 'D' type connector.

We think you will agree that Cotron's Sword range is British technology at it's best.

FEATURES INCLUDE:
14" RGB MONITOR * TTL & ANALOGUE * 18MHz BANDWIDTH * INFINITE COLOUR PALETTE * PRESTIGE CASE * BRITISH DESIGN & MANUFACTURE * OPTIONAL TILT & SWIVEL BASE * STANDARD OR LONG PERSISTANCE

S A B R E
The Sabre is a medium resolution monitor that has a horizontal resolution of 650 pixels and a dot pitch of .40mm, bandwidth is 18MHz.

SABRE-1S	£ 455.00
SABRE-1L	£ 480.00
SABRE-2S	£ 542.00
SABRE-2L	£ 570.00

R A P I E R
The Rapier is a high resolution monitor with a horizontal resolution of 850 pixels and a dot pitch of .31mm, bandwidth is 18MHz.

RAPIER-1S	£ 550.00
RAPIER-1L	£ 575.00
RAPIER-2S	£ 628.00
RAPIER-2L	£ 650.00

C A B L E S

Cable - PLUTO mini pallette	£ 32.00
Cable - PLUTO TTL/RGB	£ 32.00

K E Y
(1) No Stand
(2) With tilt & swivel stand
(S) Standard persistance
(L) Long persistance

M I C R O V I T E C

C U B 4 5 2
The CUB-452 is the standard resolution option from the popular Microvitec CUB range of 14" colour monitors. This model offers 452 x 585 addressable pixels.

This monitor has an RGB type input avalable as either TTL or linear.
A special version (MZ) is also avalable for the Sinclair Spectrum computer.

The (AP) version of this monitor is designed for use with a video recorder as well as a computer and as such is equipped with both PAL and AUDIO inputs.

1431MS	(Metal Case)	£ 199.00
1431LS	(Structural Foam)	£ 249.00
1431MZ	(TTL + Spectrum)	£ 225.00
1431AP	(TTL + PAL + Audio)	£ 225.00

C U B 6 5 3
The CUB-653 is the medium resolution option in the Microvitec 14" range. This model offers 653 x 585 addressable pixels.

Avalable in both TTL and Linear versions the CUB-653 provides the ideal specification for the majority of micro's with high resolution colour and 80 column displays.

1451MS (Metal Case)	£ 299.00
1456LI (IBM-PC)	£ 395.00
1451AP (TTL + PAL + Audio)	£ 340.00
1451DQ3 (Sinclair QL)	£ 239.13

C U B 8 9 5
The CUB-895 is the high resolution option in the Microvitec 14" range. This model offers 895 x 585 addressable pixels and is ideal for those applications requiring very high clarity and precise colour reproduction.

1441MS (Metal Case)	£ 440.00
1441LS (Structural Foam)	£ 450.00
1446LI (IBM/Programmed Rom)	£ 495.00

P H I L I P S

C T 2 0 0 7 T V / M O N
The Philips CT2007 is a 14" colour TV reciver/monitor with inputs for both R.G.B. and C.V.B.S. as well as audio. Bandwidth is 20MHz and the display is standard resolution.

CT2007	£ 229.00

S A N Y O

C R T 7 0
The Sanyo CRT70 is a 14" high resolution monitor in an attractive silver alloy finish. Resolution is 800 pixels and the input is R.G.B.

CRT70	£ 499.00