HP 3396 Integrator II Notes

Martin Hepperle, 2021

CONTENTS

The Integrator II is a special device for controlling gas chromatograph systems, for setting up analysis processes and for post-processing results. It can connect to external devices like computers or mass storage devices via various interface options (RS232C, analog, remote or, HP-IL/“INET”).

The device is equipped with four microprocessors including a Z-80 main processor and an 8051 processor for the HP-IL interface. The device has an internal ROM/RAM file system which stores some ready-to-run BASIC applications. Additionally, a BASIC system ROM is available. This ROM provides BASIC keywords and functions for programming the device and its I/O interfaces to a limited and specialized extent. These ROMs and RAMs are bank switched into the Z-80 address space. There seems to be no BASIC function to select a memory bank, though.

Opening the HP 3396

In order to open the device one has to remove three screws from the top cover, two screws under the transparent printer cover lid. The keyboard cover is hinged by three tabs at its front edge and is held in place by three hooks along its rear edge. To open one must push a pry tool down to unlatch these three hooks before the keyboard can be lifted up at the rear. Then it is possible to remove the three screws below the keyboard front edge. When lifting the cover up it is not necessary to unplug the flexprint

cable if the keyboard is carefully threaded through the opening. All this is described in the Installation and Service Manual (03396-90254).

Figure 1: Three latches are located at the upper edge of the keyboard plate. Push down with a proper pry tool of about 20-40 mm width to unlock them.

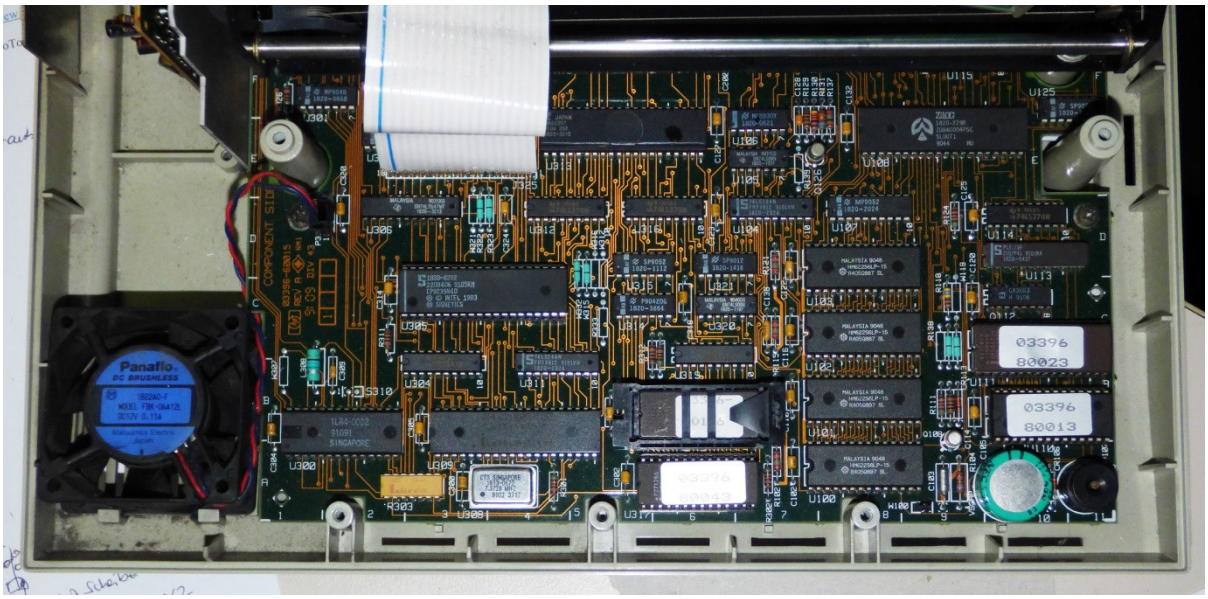


Figure 2: The main circuit board features the Z80 CPU, four ROMs resp. EPROMs, HP-IL chip and a 0.47 F storage capacitor. The three screws in the posts at the front edge can be accessed after lifting the keyboard out of the case. The two screws which go into the posts in the middle can be removed after unhooking the transparent printer cover. These two posts are bridged by an aluminum strip.

Paper

The Integrator II can be equipped with a paper holder which takes inkjet paper rolls with tractor perforation (HP Part Number: 5182-1219). Such paper is still available from HP/Agilent, but at 300 € for 4 rolls.

Serial Connection

The Integrator II comes with a female DB-15 connector offering a serial RS232C link.

A simple 3-wire connection is sufficient to link to a PC with an IBM-AT-style male DB-9 plug. This cable corresponds to the 03396-60530 DCE cable. XON/XOFF Handshaking or delays can be used to achieve stable communication, especially when sending long program lines to the device.

When pasting program lines from a terminal program without handshaking into the device, an inter-line delay of 500 ms was required, because the device takes some time to parse the line, delete any previous content of the line and insert it into the current program.

DB15 male	Signal	DB9 female
1	FRAME SHIELD	1
2	RXD	3
14	TXD	2
9	SIGNAL GROUND	5

A simple USB-Serial converter cable can be used.

Sample Number Connection

The Integrator II comes with a female DB-15 Sample connector for input of two BCD digits at TTL level indicating a bottle number from 0 to 99. The input can be read in BCD or binary format.

DB15 male	Signal	My Cable
1	BCD 20	violet
2	BCD 80	pink
3	BCD 40	gray
4	BCD 10	yellow
5	BCD 1	green
6	BCD 8	brown
7	BCD 4	white
8	BCD 2	blue
9	GND	black
10	HITR	red

Connect HITR to GND to invert default high = true BCD signal levels to low = true.

All input lines are pulled up to 5 V by an internal 1 KΩ pull-up resistor network R503 (1810-0083).

Analog Input Signal Connection

The Integrator II comes with a 3-pin male connector for input of an analogue voltage of up to 1 Volt. The pins have spacing of 2.54 mm.

3 pin plug	Signal	Location
1	shield	towards center line of device
2	signal -	
3	signal +	outer edge of device

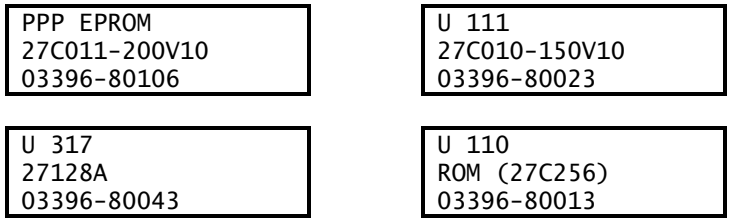
Starting

After switching on, the Integrator II takes about a minute to start and initialize and finally shows the following message on the serial connection.

```
~~HP3396,IBLI
```

Installed and Optional ROMs

My device came with the four ROMs installed, as depicted below. Tables in the manuals show that quite a few different ROMs existed at the time.



03396-80105	Application ROM, contains a file system with BASIC programs, first version
03396-80106	Application ROM, contains a file system with BASIC programs, B.02.00 to B.02.03
03396-80022	Z-80 Diagnostic ROM, first version
03396-80023	Z-80 Diagnostic ROM, revised version
03396-80042	INET ROM, first version
03396-80043	INET ROM, revised version
03396-80012	Z-80 Standard ROM, first version
03396-80013	Z-80 Standard ROM, B.02.00, 2/14/91
Optional replacements for 03396-80012, 03396-80013:	
03396-80032	Z-80 BASIC ROM, first version
03396-80033	Z-80 BASIC ROM, B.02.00, 2/14/91

Table 1: ROMs and EPROMs for the HP 3396.

The last two bytes of these ROMs carry a 16-bit CRC (polynomial = 0x8005, initial value = 0x0000) with LSB in the next to last and the MSB in the last byte. Replacing 03396-80013 (U110) with an EPROM copy of 03396-80033 added the BASIC support in my device.

Using BASIC

The optional BASIC system can be used in local mode directly from the keyboard or through an external terminal via the serial RS232C interface.

Local BASIC

`BA` [ENTER] on device keyboard enters local BASIC. The `>` prompt appears on the built-in printer.

External BASIC

`BX` [ENTER] on the device keyboard enters external BASIC. The `>` prompt appears on the terminal connected to the RS232C port and the KYBD LED blinks on 3396.

`E[XIT]` on the remote keyboard leaves the external BASIC interpreter and re-activates the keyboard of the 3396.

Programming

The device has 20128 bytes RAM for BASIC programs and data. The built-in BASIC has a good assortment of keywords, including `ADDR` as well as `PEEK` and `POKE`. Some limited access to HP-IL devices is also included.

Additionally, a built in RAM disk MDISK (M:) offers 80640 bytes for storing programs (until power-off). A built-in ROM disk EDISK (E:) contains ready to run BASIC application programs. One of these programs is interesting as it allows for trading BASIC RAM for RAM disk space. Unfortunately these programs are protected against listing.

For entering a new program, the `AUTO` statement can be used, otherwise entering a line number followed by BASIC keywords stores the line in program memory.

```
>auto

FIRST NEW LINE NUMBER:10

STEP:10

(USE BREAK OR CONTROL-Y TO END)

>10 dim S$[255]
>20 for i=1 to 100
>30 print i
>40 next i
>50 end
>60
```

The program can then be run.

```
>RUN

STARTING JAN 1, 1901 00:03:24
 1
 2
...
 99
 100

DONE JAN 1, 1901 00:03:31
```

As expected, the `LIST` command lists the program.

```
>LIST
10 FOR I=1 TO 100
20 PRINT I
30 NEXT I
40 END
```

Changing an existing line can be achieved by simply overwriting it or by using the `MODIFY` command. The `MODIFY` command prints the current line and pressing [SPACE] moves the cursor to the right. At any position `d` or `i` commands can be used to `d`elete or `i`nsert characters. Pressing [ENTER] re-lists the changed line for more modifications and a second [ENTER] copies the final line to program memory.

```
>20 PRINT I;I^4
(DELETED OLD LINE 20)
```

Removing the last character of the `PRINT` statement and replacing it by `2`:

```
>MODIFY 20
20 PRINT I;I^4
              di2[ENTER]
20 PRINT I;I^2
[ENTER]
```

Running the current program.

```
>run

STARTING JAN 1, 1901 00:04:22
 1 1
 2 4
...
 99 9801
 100 10000

DONE JAN 1, 1901 00:04:35
```

Saving the current program.

```
>save M:TEST.BAS
```

Variables and program can be deleted with the `SCRATCH` keyword.

```
>scratch

KEEP PROGRAM IN WORKSPACE [Y*/N]:y
(NO CHANGE MADE)
>
```

The HELP Command

This keyword can be used with or without parameters. It displays the current memory status and help on the selected topic.

`HELP` presents memory usage and a list of options.

`HELP N` presents the selected topic 1 to 11.

```
>help

19996 + 46 BYTES STORAGE FREE OUT OF 20128
(VARIABLE STORAGE ASSIGNED)

MENU:

1 SIMPLE USER VARIABLES
2 USER ARRAY VARIABLES AND FUNCTIONS
3 USER STRING VARIABLES
4 USER LABELS
5 COMMANDS
6 COMMAND PARAMETERS
7 BREAK COMMANDS
8 KEYWORDS
9 SECONDARY KEYWORDS
10 BASIC FUNCTIONS
11 CHROMATOGRAPHIC FUNCTIONS
ENTER SECTION NUMBER:
```

Entering the desired section number (or entering `HELP sectionnumber`) produces the corresponding reports

```
1 SIMPLE USER VARIABLES
I
```

```
2 USER ARRAY VARIABLES AND FUNCTIONS
A
```

```
3 USER STRING VARIABLES
S$
```

```
4 USER LABELS
```

```
5 COMMANDS
ASSIGN AUTO_NUMBER BREAK COPY CREATE D DATE DEBUG DELETE DIRECTORY DQ
EXIT FORMAT GET HELP INET_CONFIGURATION JOIN L LIST LOAD MODIFY N
NEXT NOLIST P PACK PRINT PROMPT PURGE R RENAME RENUMBER RUN SAVE
SCRATCH SERIALIZE SET STEP SYSTEM TIME TRACE XADDRESS XLIST XMARGIN
```

```
6 COMMAND PARAMETERS
FIRST LAST NO OFF ON TO YES
```

```
7 BREAK COMMANDS
ABORT BREAK DEBUG HELP L LET LIST P PRINT RESUME SET STEP TRACE
```

```
8 KEYWORDS
ANALYZE AREA_PERCENT ASK ASSIGN BEEP BREAK CAUSE CFG_PATHS CHAIN
CLOSE CONTINUE COPY CREATE DATA DEBUG DEF DIM DIRECTORY DO ELSE END
ENDIF ENDSUB ERASE EXIT FOR GETCALIB GETMETH GETSEQ GO GOSUB GOTO
HPIL_IO IF IMAGE INC_SLICE_NUM INC_SLICE_TIME INET_IO INIT_ACCESS
INPUT INTEGER LET LINE LOOP NEXT ON OPEN OPTION PACK PEEK$ PEN PLOT
POKE POKE2 PRINT PRINTER PURGE RANDOMIZE READ RECONFIGURE RELEASE REM
RENAME REPORT RESTORE RETRY RETURN SAVEMETH SAVESEQ SET START STOP
TERMREAD TRACE USE WAIT WHEN
```

```
9 SECONDARY KEYWORDS
ALL AND AR_REJ ATT2 BASE CHT_SP DATUM DELAY DIV EXCEPTION EXT FONT IN
IS LENGTH LIFTYPE MARGIN MOD NAME NOT OFF ON OR PK_WD RECORD REST
RUNNUM RUN_LATER RUN_NOW SEQ_LATER SEQ_NOW STEP SUB TAB THEN THRSH
TIME TO UNTIL USING WHILE XOR ZERO
```

```
10 BASIC FUNCTIONS
ABS ADDR ANGLE ATN BINAND BINCMP BINEOR BINIOR BSTR$ BTL_BCD BTL_BIN
BVAL CHR$ COS DATE DATE$ DATE2$ DEVICE$ EPS EXLINE EXP EXTEXT$ EXTYPE
FLOAT FLOAT$ FONT FP INT INTRND IP KEY$ LCASE$ LEN LOG LTRIM$ MAX
MAXNUM MIN MOD NUM ORD PEEK PEEK2 PI POS REAL RND ROTATE ROUND RTRIM$
SERIAL_NUMBER$ SGN SHIFT SIN SQR STR$ TAN TIME TIME$ UCASE$ UND
USING$ VAL
```

```
11 CHROMATOGRAPHIC FUNCTIONS
AMT AMT_LBL$ AREA AR_REJ ATT2 BCD_BTL BIN_BTL CALAMT CALFIT$ CALNUM
CALRF CALRT CALTYPE$ CHT_SP EXT$ GROUP_NAME$ GROUP_SUM HEIGHT
IDENTIFIER$ INJTIME$ ISTDAMT ISTDNUM METHOD_NAME$ MULT NAME$ NUMCALB
NUMGRPS NUMLEV NUMPEAKS PEAKNUM PK_WD PROC$ PROCFILE$ REPORT_FILE$
REPORT_MEMO$ RF RT RUNFILE$ RUNNUM SAMPAMT SAMPNAME$ SAMPNUM SEPCODE$
SIGNAL_LEVEL SLICE_AREA SLICE_NUM SLICE_TIME SLICE_WIDTH THRSH TITLE$
UNCALRF WIDTH ZERO
```

SYSTEM

This command lists the devices found on the HP-IL loop. In this case a HP 9114A flexible disk drive was found and assigned HP-IL address 8. The associated drive letter is “A:”.

```
>system

            LOOP CONFIGURATION
ADDRESS   DEVICE ID   ACCESSORY ID & CLASS
-------   ---------   --------------------
   8      HP9114A     12H  Mass Storage

  1-7     (Reserved for HPIB devices)

DISC NAME  ADDRESS  DRIVE #  VOLUME #
---------  -------  -------  --------
    A         8        0         0

RS-232-C SWITCH SETTINGS
Baud                9600
Timeout             15 sec
Handshake Delay     Off
Hardware Handshake  Disabled

INTERNAL SERIAL NUMBER:  20691
```

DIR

Displays a directory listing of the default or the specified disk

```
>dir M:   (same as DIR without parameter)

VOLUME NAME: MDISK      DRIVE: M
DATE: JAN  1, 1901  00:02:02

   FILE NAME      LENGTH   CREATED/VERSION
SIG_BUFF.RAW        2048 01/01/01 00:00:00

              USED        FREE         MAX
FILES            1          77          78
BYTES         2048       76288       80640
```

The disk in drive A:, formatted for 128 files, can store 630784 bytes (616 Kbytes):

```
>dir a:

VOLUME NAME: HP3396     DRIVE: a
DATE: OCT 11, 2021  21:11:23

   FILE NAME      LENGTH   CREATED/VERSION

              USED        FREE         MAX
FILES            0         128         128
BYTES            0      626176      630784
```

RENUM

There seems to be no `RENUM` variant where one can directly supply parameters. The system always asks for the four parameters.

```
FIRST LINE OF SECTION TO RENUMBER:5
LAST LINE OF SECTION TO RENUMBER:1000
NEW FIRST LINE NUMBER FOR RENUMBERED SECTION:10
NEW STEP:10
```

DATE

Can be used for setting the date, which is lost when the device is powered down.

```
>DATE 11,7,2021
NOV 7, 2021 00:15:50
```

Querying the date:

```
>DATE
NOV 7, 2021 00:17:56
```

TIME

Can be used for setting the time, which is lost when the device is powered down.

```
>TIME 18:43:00
NOV 7, 2021 18:43:00
```

Querying the time

```
>TIME
18:43:26
```

Using `TIME` as a function returns the time in seconds.

```
10 T_Start=TIME
```

PEEK, PEEK2

These functions read and return a byte respectively a 16-bit integer value from the given address:

```
>print PEEK(19891)
-92
>print PEEK2(19891)
-23644
```

PEEK$

This keyword takes a string variable and an integer address. It reads bytes from the given address into the string buffer until it is full

```
10 DIM BUF$(256)
20 PEEK$(BUF$),32000
```

POKE, POKE2

These functions write a byte respectively a 16-bit integer value to the given address:

```
>10 POKE 19891,-92
>20 POKE 19891,-23644
```

ADDR

Obtaining the address of a variable (like VARPTR in MS-BASIC):

```
>print ADDR(I)
-12684
>print I
 8191
>print peek(ADDR(I))
-1
>print peek(ADDR(I)+1)
 31
```

8191_d = 1F FF = 31_d -1_d

INTEGERs are 2 bytes.

```
>list
10 DIM BUFFER$(10)
 20 BUFFER$="1234567890"
 30 A=ADDR(BUFFER$)
 33 L=PEEK(A)+256*PEEK(A+1)
 35 PRINT L
 40 FOR I=A+2 TO A+1+L
 50 PRINT PEEK(I);
 60 NEXT I
 65 PRINT
 70 FOR I=A+2 TO A+1+L
 80 PRINT CHR$(PEEK(I));
 90 NEXT I
100 PRINT
110 END

>run

STARTING NOV 7, 2021 19:44:37
 10
 49 50 51 52 53 54 55 56 57 48
1234567890

DONE NOV 7, 2021 19:44:39
```

STRINGs have a 2-byte length attribute followed by the characters.

```
 10   DIM BUFFER$(10)
 20   INTEGER K
 30   PRINT "String: 1234567890"
```

```
 40   BUFFER$="1234567890"
 50   A=ADDR(BUFFER$)
 60   L=PEEK(A)+256*PEEK(A+1)
 70   PRINT "Length =";L
 80   FOR I=A+2 TO A+1+L
 90     PRINT PEEK(I);
100   NEXT I
110   PRINT
120   FOR I=A+2 TO A+1+L
130     PRINT CHR$(PEEK(I));
140   NEXT I
150   PRINT
160   PRINT "Integer K=1234"
170   K=1234
180   A=ADDR(K)
190   FOR I=A TO A+1
200     PRINT PEEK(I);
210   NEXT I
220   PRINT
230   PRINT "Real R=1.234"
240   R=1.234
250   GOSUB 420
260   PRINT "Real R=-1.234"
270   R=-1.234
280   GOSUB 420
290   PRINT "Real R=12.34"
300   R=12.34
310   GOSUB 420
320   PRINT "Real R=123.4"
330   R=123.4
340   GOSUB 420
350   PRINT "Real R=10"
360   R=10
370   GOSUB 420
380   R=-12.34
390   PRINT "Real R=";R
400   GOSUB 420
410   END
420   A=ADDR(R)
430   FOR I=A TO A+4
440     PRINT PEEK(I);
450   NEXT I
460   PRINT
470   RETURN
480   END

String 1234567890:
Length = 10
 49  50  51  52  53  54  55  56  57  48
1234567890
Integer K=1234
-46  4                    = 4*256 + (256-46)
Real R=1.234              4 bytes
-36 -7  78  1        0x.01.4E.F9.DC
Real R=-1.234
 36  6 -79  1        0x.01.B1.06.24
Real R=12.34
 80 -72  98  4       0x04.62.B8.50
Real R=123.4
 100  102  123  7    0x07.7B.66.64
Real R=10
 10  0 -63  0        0x00.C1.00.0A
Real R=-12.34
-80  71 -99  4       0x04.9D.47.B0
```

Exploring Real numbers

```
 10   DIM LINE$(80),C$(2)
 20   FOR S=1 TO -1 STEP -2
 30     R=S/2^126
 40     FOR N=1 TO 10
 50       PRINT USING "Real #####.#####^^^^":R;
 55       R=REAL(R)
 60       GOSUB 190
```

```
 70       R=R*2
 80     NEXT N
 90   NEXT S
100   FOR S=1 TO -1 STEP -2
110     R=S*2^126
120     FOR N=1 TO 10
130       PRINT USING "Real #####.#####^^^^":R;
140       GOSUB 190
150       R=R/2
160     NEXT N
170   NEXT S
180   END
190   A=ADDR(R)
200   LINE$=""
210   FOR I=A TO A+3
220     LINE$=LINE$&USING$("####",PEEK(I))
230   NEXT I
240   LINE$=LINE$&"    =    "
250   FOR I=A TO A+3
260     C$=BSTR$(BINAND(PEEK(I),255),16)
270     IF LEN(C$)=1 THEN C$="0"&C$
280     LINE$=LINE$&C$&" "
290   NEXT I
300   PRINT LINE$
310   RETURN
320   END
```

Exponent and mantissa signs? Biased?

```
Real     0.00000E+00   0   0   0-128    =    00 00 00 80  expo=0x80: zero
                      lo mid  hi 2expo
Real     1.00000E+00   0   0  64   1    =    00 00 40 01  1/2 * 2^1
Real     2.00000E+00   0   0  64   2    =    00 00 40 02  1/2 * 2^2
Real     3.00000E+00   0   0  96   2    =    00 00 60 02  (1/2+1/4)*2^2 = 3
Real     4.00000E+00   0   0  64   3    =    00 00 40 03  1/2 * 2^3
Real     5.00000E+00   0   0  80   3    =    00 00 50 03
Real     6.00000E+00   0   0  96   3    =    00 00 60 03

Real     7.00000E+00   0   0 112   3    =    00 00 70 03  (1/2+1/4+1/8)*2^3 = 7
                           01110000
                           S       |       hi: sign?, 1/2 ... 1/128
                            1/2    |
                              1/4  |
                                1/8|
                                   1/128
Real     8.00000E+00   0   0  64   4    =    00 00 40 04  1/2 * 2^4
Real     9.00000E+00   0   0  72   4    =    00 00 48 04  (1/2+1/16)*2^4 = 9.0
                           01001000
                            1/2
                               1/16

1+1/2 + 1/256:
Real     1.00000E+00   0  64  64   1    =    00 00 40 01  (1/2+1/256)*2^1
                           01000000
                            1/256 |  64 = bit 6      mid: sign? 1/256 ... 1/16384
                             1/512|       bit 5
                                  1/16384 bit 0

1+1/32768
Real     1.00006E+00  -128  0  64   1    =    80 00 40 01
                       10000000
                       1/32768  bit 7               hi: 1/32768 ... 1/2^22

Real     0.00000E+00   0   0   0-128    =    00 00 00 80
negative mantissa: sign in bit7 of hi
Real    -1.00000E+00   0   0-128   0    =    00 00 80 00
Real    -2.00000E+00   0   0-128   1    =    00 00 80 01
Real    -3.00000E+00   0   0 -96   2    =    00 00 A0 02
Real    -4.00000E+00   0   0-128   2    =    00 00 80 02
Real    -5.00000E+00   0   0 -80   3    =    00 00 B0 03
Real    -6.00000E+00   0   0 -96   3    =    00 00 A0 03
Real    -7.00000E+00   0   0-112   3    =    00 00 90 03
Real    -8.00000E+00   0   0-128   3    =    00 00 80 03
Real    -9.00000E+00   0   0 -72   4    =    00 00 B8 04
```

SIGNAL_LEVEL

Reads the analog input value in millivolts from the analog connector.

[Note: The maximum value seems to be 1083.3 mV and the minimum about -10 mV; according to the manual the resolution per step is 1/6 mV, which yields (1083. 3+10)/(1/6) = 6560 steps]

```
>print signal_level
 1028.33
```

BTL_BIN

Read the bottle number as a binary number from the Sample connector.

```
>print btl_bin
 153
```

BTL_BCD

Read the bottle number as a two digit BCD number from the Sample connector ($99_h = 153_d$).

```
>print btl_bcd
 99
```

INET_CONFIGURATION

```
LIST, EXIT, CHANGE, OR HELP [L/E*/C/H]: help

INET Instruments communicate over distinct "Data Paths". Each Data
Path carries one type of data. An Instrument may be a Producer "PROD"
or Consumer "CONS" of one or more Data Types. Some Data Types are:
 A0 Run Annotation
 C1 Chromatographic Signal
 I1 External Events Commands
 K0 Device Dependent Commands
 S0 Sample Number

Even though this integrator is single Channel "CH 0", it configures
with Multi-Channel Instruments by allowing only one Active Channel per
Instrument. When the loop is first connected, it auto-configures with
the largest possible set of Active Entries.

When changing Entry "STATUS", note that:
 - To activate an Entry, enter its number.
 - A "-" number idles an Entry
 - Any Data Path left without at least one Active Producer and one
 Active Consumer will be deactivated.
 - An Instrument may reject an Entry Activation for its own reason
 (e.g. an option not installed).

LIST, EXIT, CHANGE, OR HELP [L/E*/C/H]:
```

SYSTEM

```
>SYSTEM

 Loop is down

RS-232-C SWITCH SETTINGS
Baud 9600
Timeout 15 sec
Handshake Delay Off
Hardware Handshake Disabled

INTERNAL SERIAL NUMBER: 20691
```

Printing an ASCII Table

```
 10   COLS=4
```

```
 20   FOR I=0 TO 255 STEP COLS
 30     FOR C=0 TO COLS-1
 40       A=I+C
 50       IF A<32 THEN A=46
 60       REM   PRINT "   "&STR$(I+C)&": "&CHR$(A);
 70       PRINT USING 110:I+C,CHR$(A);
 80     NEXT C
 90     PRINT
100   NEXT I
110   IMAGE:#####: # #####: # #####: # #####: #
120   END
```

Reading an ASCII LIF file

```
 10   DIM LINE$(80)
 20   OPEN #1: NAME "M:TEST.BAA"
 30   ASK #1: LIFTYPE T
 40   PRINT "File Type=";T
 41   WHEN EXCEPTION IN
 50     READ #1:LINE$
 70     PRINT LINE$
 80     GOTO 50
 82   USE
 88   END WHEN
 90   CLOSE #1
100   PRINT "Done."

>save M:TEST.BAA

>run
```

A .BAA text file is a standard .LIF ASCII file. Therefore it consist of lines starting with a 16 bit line length word (high byte first), followed by the characters and a padding Null byte if the length is odd. The end of the file is indicated by a length word of 0xFFFF.

Listing the MDISK File Directory

Each directory entry in RAM and in the Application ROM has the following structure:

Element Name	Offset Bytes	Length Bytes	Example	Comments
File Name	0	10	BASDIALE4_	2 last characters: 4_=BASIC, 5_=BAA, 6_=RAW
LIF Type	10	2	FF D7	LIF type code for HP 3396 Integrator = D7FF
Start Record	12	2	02 00	2*256=200h
Length in Records	14	2	25 00	25h records * 256 Bytes = 2500h bytes
Date	16	3	91 10 23	date BCD (YY MM DD)
Time	19	3	10 39 56	time BCD (HH MM SS)
Length in Bytes	22	3	10 24 00	implementation bytes only for LIF type D7FF
TOTAL		**25**		

The 16-bit integer values are stored low byte first. The file name includes a file type extension of two characters and is blank filled to 10 characters.

The start location of the directory in RAM is unclear, I found it by experimentation. The system may buffer the directory in RAM, but accessing the real directory and the real files may require bank switching which is not available from BASIC.
The address ASTART seems to depend on the currently loaded program

```
 10   REM directory lister
 20   DIM N$(10),E$(3)
 30   DIM DMY$(8),HMS$(8)
 50   DIM SREC(32),LREC(32)
 60   DIM C$(1)
 70   ASTART=16991
```

```
 80   BLOCKS=16
 90   A=ASTART
100   PRINT "   N      NAME  EXT   LIF   BYTES  START LRECS       DATE      TIME"
110   FOR N=1 TO BLOCKS
120     PEEK$ (N$),A
130     A=A+10
140     N$=RTRIM$(N$)
150     E$=N$(LEN(N$)-1:LEN(N$))
160     N$=N$(1:LEN(N$)-2)
170     IF E$="6_" THEN E$="RAW"
180     IF E$="5_" THEN E$="BAA"
190     IF E$="4_" THEN E$="BAS"
200     TYPE=PEEK2(A)
210     A=A+2
220     SREC(N)=PEEK2(A)
230     A=A+2
240     LREC(N)=PEEK2(A)
250     A=A+2
260     YY=PEEK(A):MM=PEEK(A+1):DD=PEEK(A+2)
270     IF DD<16 THEN DMY$="0" ELSE DMY$=""
280     DMY$=DMY$&BSTR$(DD,16)&"."
290     IF MM<16 THEN DMY$=DMY$&"0"
300     DMY$=DMY$&BSTR$(MM,16)&"."
310     IF YY<16 THEN DMY$=DMY$&"0"
320     DMY$=DMY$&BSTR$(YY,16)
330     A=A+3
340     HR=PEEK(A):MI=PEEK(A+1):SE=PEEK(A+2)
350     IF HR<16 THEN HMS$="0" ELSE HMS$=""
360     HMS$=HMS$&BSTR$(HR,16)&":"
370     IF MI<16 THEN HMS$=HMS$&"0"
380     HMS$=HMS$&BSTR$(MI,16)&":"
390     IF SE<16 THEN HMS$=HMS$&"0"
400     HMS$=HMS$&BSTR$(SE,16)
410     A=A+3
430     FILEBYTES=256*(256*PEEK(A+2)+PEEK(A+1))+PEEK(A)
440     A=A+3
450     IF LREC(N)=0 THEN N$="[purged]":E$="   "
460     PRINT USING 610:
                  N,N$(1:8),E$,BSTR$(TYPE,16),FILEBYTES,SREC(N),LREC(N),DMY$,HMS$
470   NEXT N
480   PRINT "File Number ";      :    REM FILE DUMP DOES NOT WORK
490   INPUT N
500   PRINT "Start at offset ";SREC(N)
510   PRINT "Length ";LREC(N)*256;" bytes"
520   A=ASTART+SREC(N)*256
530   K=LREC(N)*256
540   FOR N=1 TO K
550     C=PEEK(A)
560     IF C<32 THEN C$="." ELSE C$=CHR$(C)
570     PRINT C$;
580     A=A+1
590   NEXT N
600   PRINT
610   IMAGE:>### >######## >### >#### >######  >#### >####  >######## ########
620   END

>run
   N      NAME  EXT   LIF   BYTES  START LRECS       DATE      TIME
   1  SIG_BUFF  RAW  D7FF       0      0     8   01.01.01  00:00:00
   2      TEST  BAA     1       0      8     2   01.01.01  00:03:39
   3      VARS  BAS  D7FF     758     16     4   10.11.21  20:25:01
   4     ASCII  BAS  D7FF     380     22     2   10.11.21  20:35:10
   5     ASCII  BAA     1       0     24     2   10.11.21  20:36:34
   6   REALVAR  BAS  D7FF     826     26     4   10.11.21  15:55:44
   7   DIRLIST  BAS  D7FF    2170     30     9   10.11.21  20:03:55
   8   READFIL  BAS  D7FF     286     20     2   10.11.21  20:34:44
   9      DUMP  BAS  D7FF     478     13     3   10.11.21  19:58:44
  10  [purged]          0       0      0     0   00.00.00  00:00:00
  11  CAPACITY  BAS  D7FF     586     10     3   10.11.21  19:47:47
  12  [purged]          0       0      0     0   00.00.00  00:00:00
  13  [purged]          0       0      0     0   00.00.00  00:00:00
  14  [purged]          0       0      0     0   00.00.00  00:00:00
  15  [purged]          0       0      0     0   00.00.00  00:00:00
  16  [purged]          0       0      0     0   00.00.00  00:00:00

>dir
```

```
VOLUME NAME: MDISK      DRIVE: M
DATE: OCT 11, 2021  20:04:49

   FILE NAME      LENGTH   CREATED/VERSION
SIG_BUFF.RAW        2048 01/01/01 00:00:00
TEST    .BAA         512 01/01/01 00:03:39
VARS    .BAS        1024 10/11/21 20:25:01
ASCII   .BAS         512 10/11/21 20:35:10
ASCII   .BAA         512 10/11/21 20:36:34
REALVAR .BAS        1024 10/11/21 15:55:44
DIRLIST .BAS        2304 10/11/21 20:03:55
READFIL .BAS         512 10/11/21 20:34:44
DUMP    .BAS         768 10/11/21 19:58:44
CAPACITY.BAS         768 10/11/21 19:47:47

              USED        FREE         MAX
FILES           10          68          78
BYTES         9984       68352       80640
```

Dumping Memory

```
 10   DIM C$(1),L$(32),H$(80),D$(7)
 20   PRINT "Start Address, Paragraphs";
 30   INPUT ASTART,BLOCKS
 40   FOR N=1 TO BLOCKS
 50     L$=""
 60     D$="000"&BSTR$(ASTART,16)
 70     DO WHILE LEN(D$)>4
 80       D$=D$(2:)
 90     LOOP
100     H$=D$&":"
110     FOR A=ASTART TO ASTART+15
120       C=PEEK(A)
130       IF C<32 THEN C$="." ELSE C$=CHR$(C)
140       L$=L$&C$
150       D$="0"&BSTR$(C,16)
160       IF LEN(D$)>2 THEN D$=D$(2:3)
170       H$=H$&" "&D$
180     NEXT A
190     PRINT H$&" "&L$
200     ASTART=ASTART+16
210   NEXT N
220   IMAGE: ##
230   END
```

Running the BYTE Magazine Sieve Benchmark

To speed things up, we only run one single iteration – the elapsed time has to be multiplied by 10 to obtain the final comparable result.

First variant: we can use a STRING variable for the flags to minimize storage requirements.

```
 10 DIM S$(8191)
 20 PRINT "1 ITERATION"
 22 T0=TIME
 30 FOR M=1 TO 1
 40 C0=0
 50 FOR I=0 TO 8190
 60 S$(I:I)="-"
 70 NEXT I
 80 FOR I=0 TO 8190
 90 IF S$(I:I)#"-" THEN GOTO 170
100 P=I+I+3
110 K=I+P
120 IF K>8190 THEN 160
130 S$(K:K)="+"
140 K=K+P
150 GOTO 120
160 C0=C0+1
170 NEXT I
180 NEXT M
```

```
185 T1=TIME
190 PRINT C0,"PRIMES"
195 PRINT T1-T0
200 END

>help
19644 + 34 BYTES STORAGE FREE OUT OF 20128
(NO VARIABLE STORAGE ASSIGNED)

>run

STARTING NOV 7, 2021 19:14:50
1 ITERATION
 1899 PRIMES
130.547

DONE NOV 7, 2021 19:23:59
```

Second variant: using an INTEGER array instead increases storage requirements but also speed:

```
  5 OPTION BASE 0
 10 DIM S (8191)
 20 PRINT "1 ITERATION"
 22 T0=TIME
 30 FOR M=1 TO 1
 40 C0=0
 50 FOR I=0 TO 8190
 60 S(I)=0
 70 NEXT I
 80 FOR I=0 TO 8190
 90 IF S(I)#0 THEN GOTO 170
100 P=I+I+3
110 K=I+P
120 IF K>8190 THEN 160
130 S(K)=1
140 K=K+P
150 GOTO 120
160 C0=C0+1
170 NEXT I
180 NEXT M
185 T1=TIME
190 PRINT C0,"PRIMES"
195 PRINT T1-T0
200 END

>help
3210 + 50 BYTES STORAGE FREE OUT OF 20128
(VARIABLE STORAGE ASSIGNED)

>run

STARTING NOV 7, 2021 19:27:07
1 ITERATION
 1899 PRIMES
 85.3437

DONE NOV 7, 2021 19:28:34
```

The second variant running over the full 10 iterations:

```
10 ITERATIONS
 1899 PRIMES
 831.149
```

Thus we obtain final times of about 830 seconds. This places the device into the same league as other Z-80 Microsoft BASIC systems or an Apple II with BASIC. A HP-85 in assembler takes about 21 seconds, while the BASIC on the same machine requires about 2500-3000 seconds.

Using HP-IL Devices

The undocumented `HPIL_IO` command can be used to send arbitrary HP-IL frames and receive replies. The 3396 has an address of 0 and external devices receive addresses 1…31 (1…7 are reserved for HP-IB devices connected to a HP-IL/HP-IB converter).

Some research revealed that this command takes a sequence of double byte blocks to compose the frames sent out to the loop. The first byte is the data byte and the following byte indicates the frame type. The return string starts with a return status byte. You should have your HP-IL frame table at hand to select and define these bytes.

Note that if something goes wrong, the system may hang, but disconnecting the loop and waiting for a timeout usually keeps your program in memory.

Note that it is a bad idea to have a PIL-Box with the ILPer software in the loop. The integrator uses the SS/80 protocol to communicate with mass storage devices like the 9114 floppy disk drive or HP-IB hard disk drives. This protocol is not supported by ILper and interpreted incorrectly, resulting in an endless stream of bytes, blocking the HP-IL communication. You can use alternative HP-IL software blocks with the PIL-Box as long as you do not include a mass storage device (or if this simulation supports SS/80).

This following small program demonstrates how a digital voltmeter HP 3468A can be read. The program is written for demonstration purposes; for more elaborate programs a set of small subroutines to pre-compose the command bytes would be more efficient.

```
 10   REM ---------------
 20   REM Loop Configuration
 30   REM HP-IL Devices:
 40   REM addr=0: HP 3396
 50   REM addr=8: DVM 3468A
 60   REM addr=9: 9114B
 70   REM ---------------
 80   DIM REPLY$(80)
 90   DIM SEQ$(80)
100   DIM VOLTS$(16)
110   REM -----
120   REM Define Frame Types
130   DAB=1:DEB=2:CMD=4:RDY=5
140   REM -----
150   REM Define Frame Bytes
160   LAD=BVAL("20",16):REM LAD+addr
170   TAD=BVAL("40",16):REM TAD+addr
180   SDA=BVAL("60",16):REM Send Device Accessory ID
190   SDI=BVAL("62",16):REM Send Device ID
200   SAI=BVAL("63",16):REM Send Accessory ID
210   REN=BVAL("92",16):REM Remote ENable
220   TRG=BVAL("08",16):REM Group Execute Trigger
230   UNL=BVAL("3F",16):REM Unlisten
240   UNT=BVAL("5F",16):REM Untalk
250   REM -----
260   REM
270   REM Ask devices for their names
280   REM
290   PRINT "ADDR  DEVICE"
300   FOR N=1 TO 31
310     REM --- TADn, LAD0, SDI
320     WHEN EXCEPTION IN
330       SEQ$=CHR$(TAD+N)&CHR$(CMD)
340       SEQ$=SEQ$&CHR$(LAD+0)&CHR$(CMD)
350       SEQ$=SEQ$&CHR$(SDI)&CHR$(RDY)
360       HPIL_IO SEQ$,REPLY$
370       REM strip off leading byte and CR/LF
380       REPLY$=REPLY$(2:LEN(REPLY$)-2)
390       PRINT N;"   ";REPLY$
400     USE
```

```
410     END WHEN
420   NEXT N
430   PRINT
440   REM
450   REM -----
460   REM Set up DVM for software trigger T2
470   REM
480   REM --- TAD0, LAD8, REN
490   SEQ$=CHR$(LAD+8)&CHR$(CMD)
500   SEQ$=SEQ$&CHR$(TAD+0)&CHR$(CMD)
510   SEQ$=SEQ$&CHR$(REN)&CHR$(CMD)
520   HPIL_IO SEQ$,REPLY$
530   REM
540   REM --- TAD0, LAD8, "T2" (DAB, END)
550   SEQ$=CHR$(LAD+8)&CHR$(CMD)
560   SEQ$=SEQ$&CHR$(TAD+0)&CHR$(CMD)
570   SEQ$=SEQ$&"T"&CHR$(DAB)&"2"&CHR$(DEB)
580   HPIL_IO SEQ$,REPLY$
590   UMIN=0.8:UMAX=1.5
600   PRINT "How many samples to read? ";
610   INPUT NMAX
620   PRINT "  N     U [V]"
630   REM
640   FOR N=1 TO NMAX
650     REM Trigger DVM
660     SEQ$=CHR$(LAD+8)&CHR$(CMD)
670     SEQ$=SEQ$&CHR$(TAD+0)&CHR$(CMD)
680     SEQ$=SEQ$&CHR$(TRG)&CHR$(CMD)
690     HPIL_IO SEQ$,REPLY$
700     REM --- read voltage as text
710     REM --- TAD8, LAD0, SDA
720     SEQ$=CHR$(TAD+8)&CHR$(CMD)
730     SEQ$=SEQ$&CHR$(LAD+0)&CHR$(CMD)
740     SEQ$=SEQ$&CHR$(SDA)&CHR$(RDY)
750     HPIL_IO SEQ$,VOLTS$
760     REM strip off leading CHR$(4)
770     VOLTS$=VOLTS$(2:)
780     REM strip off trailing CR/LF
790     VOLTS$=VOLTS$(1:LEN(VOLTS$)-2)
800     U=VAL(VOLTS$)
810     PRINT USING "### ###.#######":N,U
820     IF U>UMAX THEN BEEP
830     IF U<UMIN THEN BEEP
840   NEXT N
850   REM
860   SEQ$=CHR$(UNL)&CHR$(CMD)
870   SEQ$=SEQ$&CHR$(UNT)&CHR$(CMD)
880   HPIL_IO SEQ$,REPLY$
890   END
```

Running this program produces the following output:

```
HP3468A
HP9114B
  N     U
  1   1.8366808
  2   1.8363896
  3   1.8363896
  4   1.8360504
 ...  ...
```

HP-IL Command Frames

CMD [100]	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0 0000	NUL	GTL			SDC	PPD			GET							
1 0001		LL0			DCL	PPU			EAR							
2 0010	LAD0	LAD1	LAD2	LAD3	LAD4	LAD5	LAD6	LAD7	LAD8	LAD9	LAD10	LAD11	LAD12	LAD13	LAD14	LAD15
3 0011	LAD16	LAD17	LAD18	LAD19	LAD20	LAD21	LAD22	LAD23	LAD24	LAD25	LAD26	LAD27	LAD28	LAD29	LAD30	UNL
4 0100	TAD0	TAD1	TAD2	TAD3	TAD4	TAD5	TAD6	TAD7	TAD8	TAD9	TAD10	TAD11	TAD12	TAD13	TAD14	TAD15
5 0101	TAD16	TAD17	TAD18	TAD19	TAD20	TAD21	TAD22	TAD23	TAD24	TAD25	TAD26	TAD27	TAD28	TAD29	TAD30	UNT
6 0110	SAD0	SAD1	SAD2	SAD3	SAD4	SAD5	SAD6	SAD7	SAD8	SAD9	SAD10	SAD11	SAD12	SAD13	SAD14	SAD15
7 0111	SAD16	SAD17	SAD18	SAD19	SAD20	SAD21	SAD22	SAD23	SAD24	SAD25	SAD26	SAD27	SAD28	SAD29	SAD30	SAD31
8 1000	PPE0	PPE1	PPE2	PPE3	PPE4	PPE5	PPE6	PPE7	PPE8	PPE9	PPE10	PPE11	PPE12	PPE13	PPE14	PPE15
9 1001	IFC		REN	NRE							AAU	LPD				
A 1010	DDL0	DDL1	DDL2	DDL3	DDL4	DDL5	DDL6	DDL7	DDL8	DDL9	DDL10	DDL11	DDL12	DDL13	DDL14	DDL15
B 1011	DDL16	DDL17	DDL18	DDL19	DDL20	DDL21	DDL22	DDL23	DDL24	DDL25	DDL26	DDL27	DDL28	DDL29	DDL30	DDL31
C 1100	DDT0	DDT1	DDT2	DDT3	DDT4	DDT5	DDT6	DDT7	DDT8	DDT9	DDT10	DDT11	DDT12	DDT13	DDT14	DDT15
D 1101	DDT16	DDT17	DDT18	DDT19	DDT20	DDT21	DDT22	DDT23	DDT24	DDT25	DDT26	DDT27	DDT28	DDT29	DDT30	DDT31
E 1110																
F 1111																
→	second nibble (LSN)															
↓	first nibble (MSN)															

HP-IL Ready Frames

RDY [101]	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0 0000	RFC															
1 0001																
2 0010																
3 0011																
4 0100	ETO	ETE	NRD													
5 0101																
6 0110	SDA	SST	SDI	SAI	TCT											
7 0111																
8 1000	AAD0	AAD1	AAD2	AAD3	AAD4	AAD5	AAD6	AAD7	AAD8	AAD9	AAD10	AAD11	AAD12	AAD13	AAD14	AAD15
9 1001	AAD16	AAD17	AAD18	AAD19	AAD20	AAD21	AAD22	AAD23	AAD24	AAD25	AAD26	AAD27	AAD28	AAD29	AAD30	IAA
A 1010	AEP0	AEP1	AEP2	AEP3	AEP4	AEP5	AEP6	AEP7	AEP8	AEP9	AEP10	AEP11	AEP12	AEP13	AEP14	AEP15
B 1011	AEP16	AEP17	AEP18	AEP19	AEP20	AEP21	AEP22	AEP23	AEP24	AEP25	AEP26	AEP27	AEP28	AEP29	AEP30	IEP
C 1100	AES0	AES1	AES2	AES3	AES4	AES5	AES6	AES7	AES8	AES9	AES10	AES11	AES12	AES13	AES14	AES15
D 1101	AES16	AES17	AES18	AES19	AES20	AES21	AES22	AES23	AES24	AES25	AES26	AES27	AES28	AES29	AES30	IES
E 1110	AMP0	AMP1	AMP2	AMP3	AMP4	AMP5	AMP6	AMP7	AMP8	AMP9	AMP10	AMP11	AMP12	AMP13	AMP14	AMP15
F 1111	AMP16	AMP17	AMP18	AMP19	AMP20	AMP21	AMP22	AMP23	AMP24	AMP25	AMP26	AMP27	AMP28	AMP29	AMP30	IMP
	→ second nibble (LSN)															
	↓ first nibble (MSN)															

HP-IL Data Frames

DATA [000]	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0 0000	NUL	SOH	STI	ETX	EOT	ENQ	ACK	BEL	BS	HAT	LF	VT	FF	CR	SO	SI
1 0001	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2 0010		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3 0011	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4 0100	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5 0101	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6 0110	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7 0111	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
8 1000	€		‚	ƒ	„	…	†	‡	ˆ	‰	Š	‹	Œ		Ž	
9 1001		‘	’	“	”	•	–	—	˜	™	š	›	œ		ž	Ÿ
A 1010		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
B 1011	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C 1100	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D 1101	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E 1110	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F 1111	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ
	→ second nibble (LSN)															
	↓ first nibble (MSN)															

Using the Analog Input

The analog input connector accepts up to 1V. The resolution seems to be around 10 bit.

```
 10   REM Using the A/D Converter
 20   REM Capacity measurement of a used battery cell
 25   AMP=3 : REM Voltage divider: 3:1
 30   CAPACITY=0
 40   DT=10
 50   T=0
 60   FOR I=1 TO 36000.
 70     VOLTAGE=AMP*SIGNAL_LEVEL/1000
 80     CURRENT=VOLTAGE/12
 90     PRINT USING 140:T/3600,VOLTAGE,CAPACITY
100     WAIT DELAY DT
110     CAPACITY=CAPACITY+VOLTAGE*CURRENT*1000*DT/3600
120     T=T+DT
130   NEXT I
140   IMAGE:t =###.#### hrs, U =##.#### V, C =#####.### mAh
150   END
```

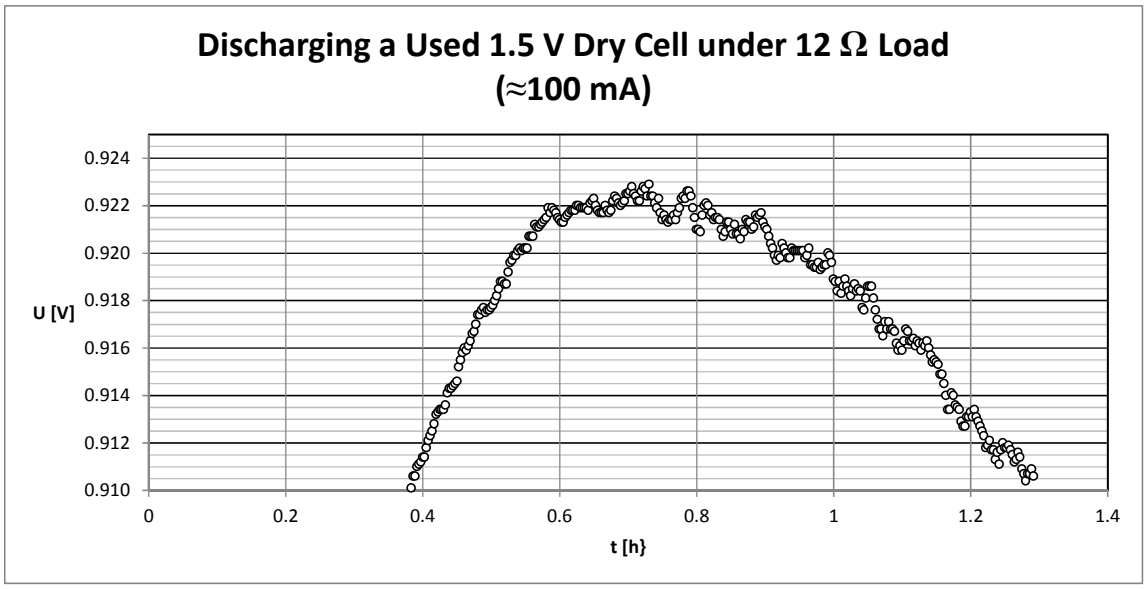


Figure 3: The sampled voltage of a well-used dry cell shows an initial increase when the chemistry starts to work until it starts to drop. Also some cyclic voltage variations can be observed.

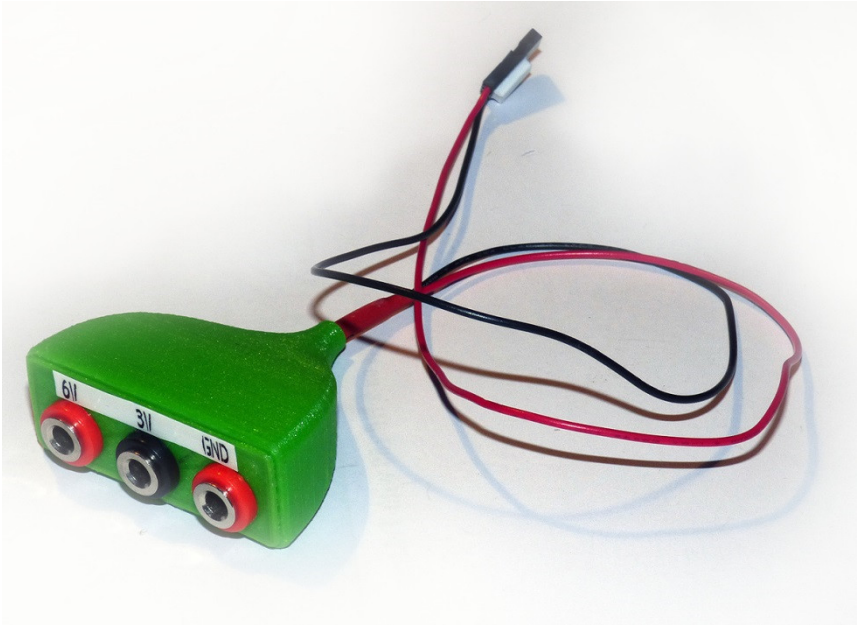


Figure 4: A simple adapter box with voltage divider providing ranges of 3 V and 6V.

HP 49G RPL Voltage Divider Program

This little program calculates the resistor values needed for a given set of voltages. The desired voltages are hardcoded as an ordered list $\{U_1\ U_2\ ...\ U_n\}$. In a typical application, the first (lowest) value U_1 would be the output voltage (the list is the first object in the program, the current is also hard coded).

EVALuating the program produces two lists:

- the list of voltages $\{U_1\ U_2\ ...\ U_n\}$
- the list of resistors $\{R_1\ R_2\ ...\ R_n\}$

General Circuit

```
            INPUT VOLTAGES
   0V             U2      ...       Un
   |              |        |         |
   |              |        |         |
   +-[R1]-+-[R2]-+-[R3] ... [Rn] -+
   |      |
   |      |
   0V    U1
OUTPUT VOLTAGE
```

Example Circuit

```
0V             3V      6V
|              |       |
|--R1--+--R2--+--R3--+
|      |
0V     1V
```

```
<< {1. 3. 6.}                    # input: voltages
   DUP SIZE 0.0001 0.            # L length, I current, S cumulative resistance
   -> L I S
   << 1. L FOR N
         DUP N GET I / S -       # next resistor value
         DUP S + 'S' STO SWAP    # update resistance
      NEXT
      'N' STO                    # save voltage  list
      L ->LIST                   # make resistor list
      N SWAP                     # restore voltage list
   >>
>>

EVAL
{1. 3. 6.}
{10000. 20000. 30000.}
```

I = 0.1 mA, R1 = 10K → 11K, R2 = 20K → 22K, R3 = 30K →33K

Remote Control through RS232C

The Integrator can be controlled through its serial interface. HP offered the PEAK96 software to control the device from a PC. Unfortunately, the corresponding programming manual was not available, so a short exploration of the communication protocol was performed. The communication was examined by adding a HP 4951 Protocol Analyzer to the connection cable.

As the protocol seems to be rather cryptic, no further analysis was performed.

Some observations:

- Blocks sent from the **PC** always end with 0D0A (CR/LF).
- The reply of the integrator (abbreviated below as **IT**) is always terminated by a DC3 (0x13, CTRL+S) character.
- `CEVRQ`: the Integrator sends its name and a status byte(?).
- `COTPR`: the PC sends of a string of ASCII codes to the Integrator for printing.
- `CRKPR`: the PC sends keyboard commands in form of ASCII HEX codes to the Integrator.
- `KRPR`: the Integrator sends result in form of ASCII HEX codes to the PC.

  ```
  IT:        KRPR 0D 0A 23  20    20    [DC3]
                  CR LF 19d SPACE SPACE
  ```

  This format is also used when a directory list is asked for.

Starting PEAK96 establishes the communication, sends date and time commands and prints a welcome message:

```
PC:       [ESC]CNTA[DC3]A[ESC]CEP[ESC]CM-[ESC]CRFRQ[CR][LF]
IT:       RFEN00[DC3]

PC:       [ESC]CARRQ[CR][LF]
IT:       ARCO[DC3]

PC:       [ESC]CARPR2[CR][LF]
IT:       AREN[DC3]

PC:       [ESC]CRKRQ[CR][LF]
IT:       RKCO[DC3]
                    D A T E 1 1 / 2 3 / 2 1CR                 set date
PC:       [ESC]CRKPR44415445313132F32332F32313OD[CR][LF]      hex codes
IT:       RKEN[DC3]

PC:       [ESC]CRKRQ[CR][LF]
IT:       RKCO[DC3]

                    T I M E 2 1 : 1 0 : 3 6 CR              set the time
PC:       [ESC]CRKPR54494D4532313A31303A33360D[CR][LF]      hex codes
IT:       RKEN[DC3]

PC:       [ESC]COTRQ[CR][LF]
IT:       OTCO[DC3]

PC:       [ESC]COTPR11/23/21[CR][LF]                        print message in Integrator
IT:       OTCO[DC3]

PC:       [ESC]COTPRHost available.[CR][LF]                 print message in Integrator
IT:       OTCO[DC3]
```

PEAK96 issues a Status request:

```
PC:       [ESC]CRFRQ[CR][LF]
IT:       RFEN00[DC3]

PC:       [ESC]CEVRQ[CR][LF]
IT:       EVPRHP3396,IBLI,24[DC3]                           the 24 encodes status bits

PC:       [ESC]CEVCO[CR][LF]
IT:       EVPR[DC3]

PC:       [ESC]CEVCO[CR][LF]
IT:       EVPR[DC3]

PC:       [ESC]CEVCO[CR][LF]
IT:       EVEN[DC3]

PC:       [ESC]CNTA[DC3]A[ESC]CEP[ESC]CM-
IT:       no reply, o.k.
```

This exchange switches the Integrator to Remote Keyboard mode and back (when ESC is pressed):

```
PC:       [ESC]CRFRQ[CR][LF]
IT:       RFEN00[DC3]

PC:       [ESC]CEVRQ[CR][LF]
IT:       EVPRHP3396,IBLI,24[DC3]

PC:       [ESC]CEVCO[CR][LF]
IT:       EVPR[DC3]

PC:       [ESC]CEVCO[CR][LF]
IT:       EVPR[DC3]

PC:       [ESC]CEVCO[CR][LF]
IT:       EVEN[DC3]
```

```
PC:        [ESC]CNTA[DC3]A[ESC]CEP[ESC]CM-[ESC]CARRQ[CR][LF]
IT:        ARCO[DC3]

PC:        [ESC]CARPR4[CR][LF]
IT:        KRRQ[DC3]

PC:        [ESC]CKRCO[CR][LF]
IT:        KRPROD0A232020[DC3]

PC:        [ESC]CKRCO[CR][LF]
IT:        AREN[DC3]

PC:        [ESC]CLURQ[CR][LF]
IT:        LUCO[DC3]

PC:        [ESC]CLUPR1[CR][LF]
IT:        LUEN[DC3]

PC:        [ESC]CARRQ[CR][LF]
IT:        ARCO[DC3]

PC:        [ESC]CARPR0[CR][LF]
IT:        KREN[DC3]

PC:        [ESC]CEVRQ[CR][LF]
IT:        EVPRHP3396,IBLI,20[DC3]

PC:        [ESC]CEVAB[CR][LF]
IT:        AREN[DC3]
```

Leaving PEAK96:

```
PC:        [ESC]CEVRQ[CR][LF]
IT:        EVPRHP3396,IBLI,24[DC3]

PC:        [ESC]CEVAB[CR][LF][ESQ]CARRQ[CR][LF]
IT:        ARCO[DC3]

PC:        [ESC]CARPR2[CR][LF]
IT:        AREN[DC3]

PC:        [ESC]COTRQ[CR][LF]
IT:        OTCO[DC3]

PC:        [ESC]COTPRHost unavailable.[CR][LF]           print message in Integrator
IT:        OTCO[DC3]

PC:        [ESC]COTEN[CR][LF][ESC]CARRQ[CR][LF]
IT:        ARCO[DC3]

PC:        [ESC]CARPR0[CR][LF]
IT:        AREN[DC3]

PC:        [ESC]CM+[CR][LF]
IT:        no further response
```

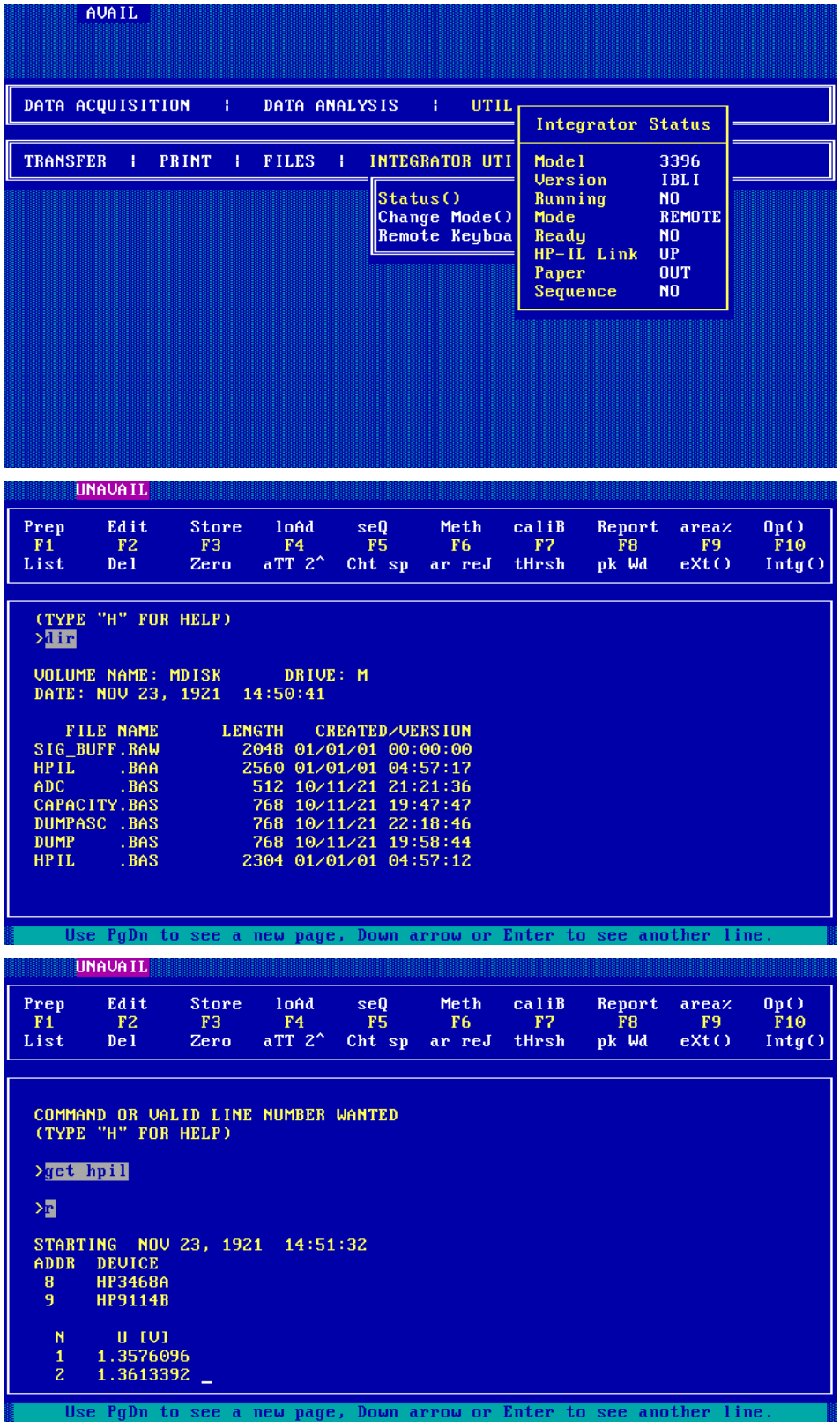


Figure 5: The PEAK96 Software can control the Integrator, up- and download data and act as a virtual keyboard.

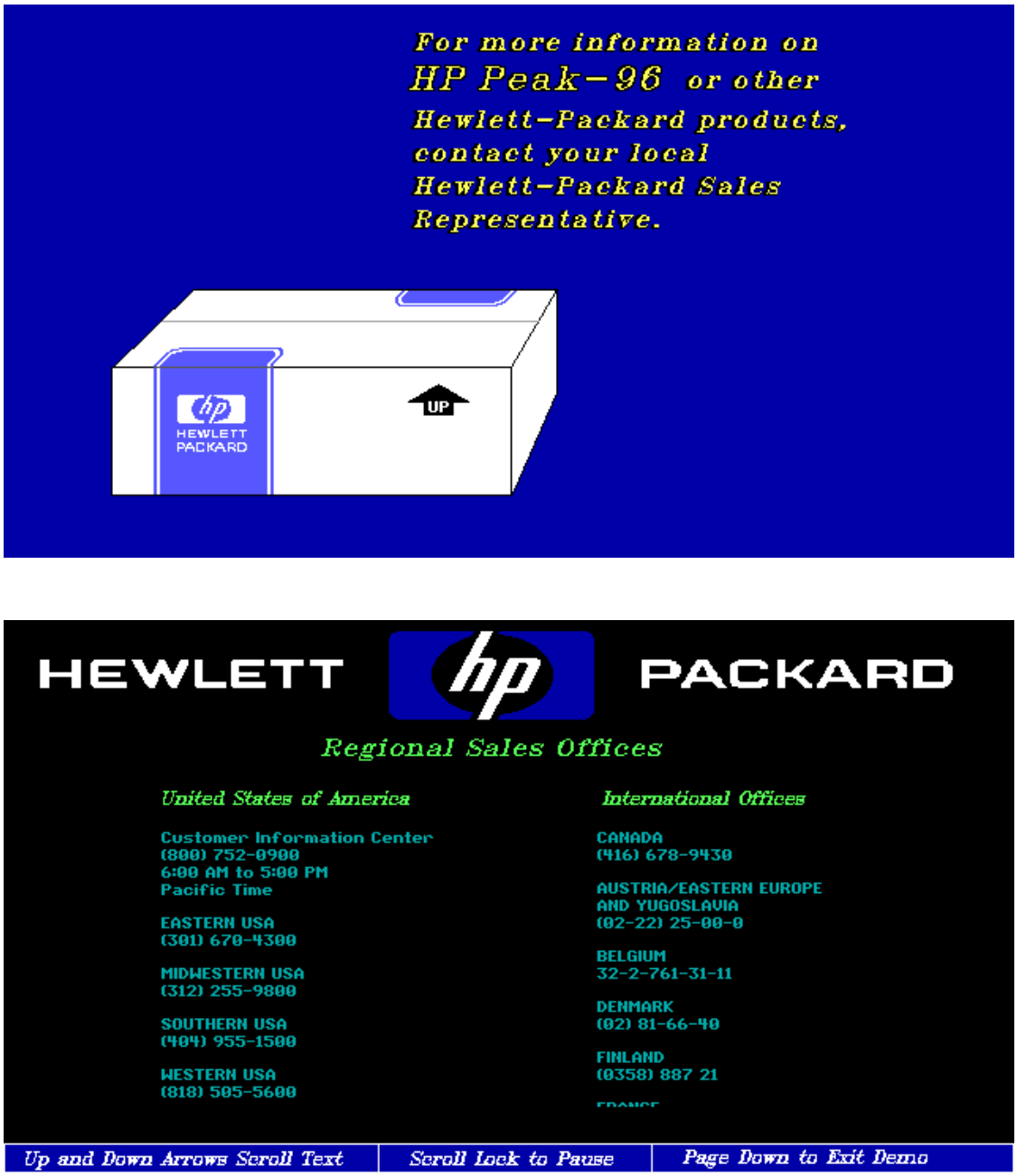


Figure 6: The DEMO version of the PEAK96 Software ends with a small sales pitch.