BUSINESS BASIC PHASE II,
THE NEXT STEP TO A FRIENDLY COMMERCIAL PROGRAMMING LANGUAGE

Dr.Ulrich E. Fauser
WEIGANG MCS
Stöberlstr. 68
D-8000 München 21, W-GERMANY

Summary

Whereas the already powerful phase I released with T-Delta 2 and presented at the Washington conference was primarily aimed towards former BASIC/3000 users, the new phase II to be released mid-86 adds even more advanced statements so that a real flexible and rich language for commercial programming will be available.

This and the nevertheless comparable performance makes HPBB a real competitor to the other prevailing commercial languages COBOL and PASCAL.

This paper will give an overview of the new features from an OEM's view who has been also a beta-testsite from the beginning of the HPBB project.

Phase II will include features like:

softkey integration for userfriendly programflow control

cursor and screen I/O

a full fledged report writer facility in a 4th generation style

database search and sort with multiple threads

other database extensions

a new forms handling package working in character mode allowing fieldwise program interaction

All this and the existing conversion tools make HPBB not only a language with a high programmer productivity but also the perfect migration tool for BASIC/260 users as it is approx. 80 - 90% compatible to the 260 language after which of course HPBB was modeled.

Another feature making HPBB attractive is HP's commitment in making it available not only with the high end spectrum series but also including the low end 260 in its range of commercial computers through the JOIN/3000 program.

In this way an OEM will have a reasonable portable language throughout the range of the HP computer family.

Introduction

This paper will be presented in two different parts. The first will deal with the purely technical matters trying to give you an overview of the newly added statements and power to the phase II of HPBB like this language is officially named.

The second will bring you some of the experiences and problems that we as a major 250/260 OEM have met first as a beta testsite and then in converting our different manufacturing and commercial application packages to the 3000 environment, a still continuing effort.

Some History

The HP3000 Business Basic project was originally started in 1981 but it took a lot of pressure from some HP260 OEMs like us to really make it going finally in 83.

HPBB had several goals, first as the 250/260 OEMs and users slowly grew beyond the capabilities of their machine the demand for a growth path into a new hardware family was obvious but there was no suitable replacement with a similar friendliness and productivity within HPs range of commercial computers. Second BASIC/3000 users were not really pleased with their language so something had to be done to make it attractive again. And third many people were looking for an easy and friendly commercial language providing not only simple access to most of the necessary subsystems like IMAGE, VPLUS etc. but also increasing programmer's productivity by doing a lot of the housekeeping routines usally left to him, therefore leaving more time for better solutions.

Although most of HPBBs features were inherited from BASIC/260 one of the languages with the highest rating in user satisfaction most of BASIC/3000s features are incorporated too.

It was decided to release HPBB in two phases, phase I was primarily aimed towards former BASIC/3000 users or OEMs who wanted to start rewriting their applications with the still restricted language subset. This phase was released last year with T-Delta 2. Phase II which is scheduled to be released in mid 86 now provides a real upgrade path for former 260 users, especially after phase II was enhanced by a product made in HP Böblingen, called JOINFORM, giving the same screen handling facilities like on the 260. Thus the compatibility to BASIC/260 by not having to use VPLUS was drastically improved.

A short review of phase I

For those of you who haven't been in Washington and haven't read the proceedings here is a quick summary of the features of phase I.

Interpreter and compiler available

Full debugging and trace capability

Builtin editor with immediate syntax check and many edit functions and program development environment

Extensive ON LINE HELP

Always actice calculator and immediate command mode

All standard control structures
  GOTO, GOSUB, CALL, ON GOTO, ON GOSUB, function calls
  FOR-NEXT, IF-THEN-ELSE, WHILE-ENDWHILE, REPEAT-UNTIL,
  SELECT-CASE, LOOP-EXIT IF-ENDLOOP

Real subprograms and functions with parameters and local variables

Named and unnamed COMMON

Various OPTIONs to control interpreter and compiler

Variable and label names up to 63 characters

Seven datatypes INTEGER, REAL, DECIMAL (each SHORT and FULL precision)
  and Strings

Dynamic checks for ON ERROR, ON DBERROR, ON HALT, ON END

Statements for accessing all IMAGE intrinsics with easy handling

Rich output formatting features

Access to MPE commands (:SYSTEM) and control of son processes
  (:SYSTEMRUN)

Interface to routines written in SPL and PASCAL

More than necessary functions for stringhandling, numeric,
  mathematical, matrix, conversion, bithandling, logical, files

All possibilities to access the file system
  also special BDATA type file including type information with
  data
  KSAM only via intrinsics

and many more features of a modern language.

Now, what is new in phase II ?

Softkeys and typing aid keys

HPBB supports the 8 softkeys provided with most HP terminals in two ways: as typing aid keys and programatic keys. Typing aid keys are used for often needed charactersequences especially by programmers. GET KEY and SAVE key are used to save and store these. Definition and changes are made via standard terminal features.

More important are programatic keys or more often just called softkeys. These are used to allow the user control the flow of control for his program. In a program a softkey is defined using the ON KEY statement with a number (1 to 8), a priority, a keylabel and a softkeyaction which can be thought of as an interrupt routine. Only on the 3000 softkey actions are no longer real interrupts like on the 260 where an action could be invoked at the end of every statement or within INPUT statements and they

always returned to the point of invocation, except with GOTO.
On the 3000 there are no real softkey interrupts, they are handled like special character sequences and only work when the program is in an input state. When the user presses a softkey during an input the statement is terminated and the corresponding softkey action is executed. Unlike on the 260 the flow of control here goes to the next statement after the input (not with GOTO) and it is left to the programmer what to do. To do this HPBB provides a builtin function RESPONSE which tells you how an input was terminated (softkey, HALT, timeout, data etc). This is not as it used to be on the 260 but it works and the additional handling around the input statements is automatically inserted by the conversion program.

The following statements are available: (examples)

```
ON KEY 7 CALL Selection;LABEL="SELECT  DATA"
   action can be CALL, GOSUB or GOTO
```

OFF KEY or OFF KEY 6 deactivates all or specified keys

ENABLE, DISABLE allow or disallow keyinterrupts to occur

PRESS KEY n makes HPBB think a softkey was pressed by user

CURKEY function returns the number of the last key pressed

Waiting for softkey, no data input allowed

```
LOOP
  ACCEPT
END LOOP
```

Waiting for data or softkey, exit also when HALT is pressed

```
LOOP
  INPUT Var
  EXIT IF RESPONSE>0
END LOOP
```

Cursor and screen I/O

The CURSOR statement allows you to place the cursor anywhere on the screen and define any of the available video enhancements

E.g. CURSOR (12,5) positions the cursor in row 12, column 5

CURSOR (15,1),("BI",-3) positions the cursor to 15,1 and displays a blinking, inverse field 3 spaces long

The builtin functions CPOS and RPOS return the current column or row position of the cursor.

The ENTER and LENTER statements work just like INPUT and LINPUT except that the input is taken from the screen. LINPUT reads a whole line including commas whereas for ENTER commas mean data item separators.

HPFORMS (VPLUS) interface

For interfacing the most common VPLUS intrinsics HPBB provides statements releaving the programmer from all the usual hassles of providing the right parameters or defining his common area. For special problems however VPLUS intrinsics still have to be called directly.

Statements:

OPEN FORM with HOME/OVERLAY/APPEND/FREEZE options

CLOSE FORM with REMAIN option

READ FORM <variablelist> with TIMEOUT and SKIP option
including imbedded FOR loops

CLEAR FORM with DEFAULT option

WRITE FORM <expr.list> with CURSOR option
including imbedded FOR loops

JOINFORM Forms/260 interface

To provide a higher compatibility for 260 users and also as an alternative to VPLUS HP Böblingen (BGD) decided to launch an additional project called JOINFORM for HP260 FORMS conversion and emulation in HP Business Basic. This team has implemented a set of library subroutines performing the same functions as FORMS/260. Calling these routines is integrated into HPBB i.e. several statements (like on the 260) check if a 260 form is active and they then behave differently.

All 260 forms can be converted into JOINFORM formfiles which are different from VPLUS files (filecode BFORM). There will also be a new friendly FORMS editor for defining, modifying and printing these forms.

JOINFORM does not work in blockmode, but in formatmode.

The forms handling statements are the same as for VPLUS, they decide from the filecode which subsystem to call.

OPEN FORM

CLOSE FORM and CLOSE FORM REMAIN

CLEAR FORM

JOINFORM allows forms to have an inputfieldorder and an outputfieldorder independent from each other for the two fieldtypes provided. Fields can be arranged in any order on the screen.There is no more extra tabfieldorder like on the 260. This is now equal to the inputfieldorder like it was done mostly on the

260 anyhow. Internally still three pointers are maintained, an input- output- and cursorfieldpointer.

The CURSOR statement was enhanced to position the cursor to these fields by adding the builtinfunctions IFLD, OFLD and CFLD

Thus CURSOR IFLD(5) sets input and cursorpointer to field #5

CURSOR OFLD(4) sets the outputpointer

CURSOR CFLD(3) sets the cursorfieldpointer

An additional builtin TFLD returns the fieldnumber of the last input field where return was pressed, this is the same as TFNUM on the 260.

All I/O interaction between program and form is done with normal I/O statements which internally behave differently when a form is active by calling the appropriate subroutines.

DISP and PRINT output to the actual outputfield thereby increasing the internal pointer.

INPUT and ACCEPT <variablelist> read from the current cursorfield for the benefit of the user it is only read what is really on the screen, so insert,delete and cursorkeys can be used, as the inputbuffer is simply discarded

ENTER <variablelist> reads starting from the current inputfield from the screen

Unlike VPLUS additional "normal" I/O can take place outside the form boundaries using LDISP and LINPUT/LENTER. The user doesn't have to distinguish between enter and return, all input is ended with return. This may cause some confusion if he uses his application together with VPLUS oriented utilities or other programs.

Database extensions

Using a socalled PACKFMT or an IN DATA SET statement allows DBGET, DBPUT and DBUPDATE to implicitly do packing and unpacking of the IMAGE buffer into/from program variables without extra PACK/UNPACK statements or program buffers.

Example:
```
Label: PACKFMT Partno$,Customer,Quantity
or
Label: IN DATASET "ORDERS" USE Partno$,Customer,quantity
```

DBGET Dbase$ USING Label;DATASET="ORDERS" will then immediately read databasevalues into program variables

The PREDICATE statement allows simple specifications of predicate for DBLOCK modes 5 and 6 without having to deal with wordcounts, entrylength and all that internal stuff.

```
PREDICATE P$ FROM "ORDERS" WITH Customer=1254;Quantity>100
```

Database SEARCH and SORT

Besides the report writer this is certainly the most powerful construct in HPBB. It allows programmers to select and sort IMAGE databases in an almost relational way, not only on single datasets but over multiple sets. HPBB supports the concept of workfile containing only pointers to selected/sorted records that is the database itself is not changed at all. This is available nowhere else on the 3000.

The DBASE IS Dbase$ statement only defines which database is actually to be searched and sorted

WORKFILE IS #1 defines the previously opened file #1 to be a special type for handling IMAGE pointers. Files with the WORKFILE option are treated specially and have some restrictions.

We have already presented the IN DATASET statement which is essential here too.

The THREAD IS statement defines which datasets are involved in a SEARCH or SORT. Up to 10 sets can be specified, no two consecutive sets of the same kind (master,detail) can be connected. If a master is linked to a detail with more than one path the PATH option allows to select which one to use. There is an additional contruct called a synthetical link when there is no real path from a detail to a master in the thread. (LINK Variable option)

The SEARCH statement searches all involved datasets according to the thread and evaluates its conditional expression. There are certain programming techniques keeping the number of records in the first dataset of the thread small because this is the only one to be read sequentially thus greatly influencing the performance. The result is stored as tuples of pointers in the workfile, one pointer for each dataset.

The SORT statement allows sorting of the pointers according to the related data in ascending or descending order with up to ten sortkeys. If a SEARCH preceded the SORT only the already selected records are sorted and vice versa.

Example (see also appendix)

```
Dset1: IN DATASET "ORDERS" USE Partno$,Customer
Dset2: IN DATASET "PARTMASTER" USE Partno$
Dset3: IN DATASET "PARTS" USE Partno$,SKIP 46,Storageloc$

ASSIGN "FILEA" to #1
WORKFILE IS #1

Thread1: THREAD IS Dset1,Dset2,Dset3

SEARCH USING Thread1;Customer=1254 AND Storageloc$="STORE_B"
```

This selects all records for customer 1254 whose parts are stored

in STORE_B.

SORT USING Thread1;Storageloc$,Partno$

Now sort them according to storagelocation and partnumber

Now a LOOP would read the workfile, access the database with DBGET in mode 4 and process the records in whatever way.

Syntax and semantic is somewhat different from what the 260 had, there it was much more flexible, but due to be compilable some restrictions had to be made.

The REPORT WRITER

This is probably the most powerful tool available to the programmer allowing him to easily define and produce reports. The definition of a report is done in a 4th generation style by telling the system how it should look like instead of programming every little single step. The report writer system takes care of all routine work like pageformat and numbering, headers, trailers, totals and averages. It also lets you define when logical breaks are to occur and what happens then.
The actual production of the report is controlled by simple output statements, the DETAIL LINE construct.

Therefore the report writer system has 3 distinct parts

The report definition part

Statements controlling the activation of the report

Functions giving various informations

The report writer definition

The REPORT HEADER defines

page format	PAGE LENGTH
margins	LEFT MARGIN
which GRAND TOTALS are to be taken	GRAND TOTALS
under which conditions a break occurs	BREAK n IF <cond>
	BREAK n WHEN var CHANGES

with up to 9 break levels
and also the appearance of a one time report header

The REPORT TRAILER defines a one time report ending trailer

The PAGE HEADER/TRAILER defines what is to be done when a new page starts or a pagebreak occurs

The HEADER/TRAILER level statements define what has to be done when a break for a certain level occurs, which local TOTALS are kept, under which conditions a line is printed at all, thereby protecting sensitive information.

A REPORT EXIT section defines what is to be done in case of

abnormal termination via a STOP REPORT command.

For an example see the appendix

REPORT WRITER executable statements

A report definition alone does nothing, it has to be activated before it can be used by a BEGIN REPORT Rptlabel statement No printing is done yet only internal initialization.

All actual printing is done triggered by the DETAIL LINE statement. It checks whether the report has started at all, which break occurs, printing appropriate headers and trailers, keeping local and grand totals and so on, all with one statement.

For rare cases when this is not enough special flow control can be activated be triggering some events like TRIGGER PAGE BREAK or TRIGGER BREAK level activating the corresponding sections in the report description.

END REPORT is the normal termination of a report, all remaining events are triggered, the report trailer is printed followed by a final page trailer. The report definition is deactivated.

STOP REPORT indicates an abnormal termination. No more normal processing is done, only the report exit section is activated if present usually giving some hints to the cause of the termination.

Report writer functions

They return various information about internal conditions and can be used to control the report even finer if necessary and to access numeric results.

AVG (level,i) average for the i-th expression of the TOTAL statement in level (0 being the GRAND TOTALS)

LASTBREAK level of last break condition

NUMBREAK (level) how often has a break for this level occured

NUMDETAIL (level) how many lines have been printed in level

OLDCV and OLDCV$ return the value of the last controlvariable for the BREAK WHEN statement, so even if you have read already a record which triggered the break the old value is still available

NUMLINE how many lines already on current page

PAGENUM current page number

RWINFO some more seldom used information

TOTAL (level,i) giving the i-th total for level

Missing features

For those of you familiar with BASIC/260 here is a list of features not available in HPBB. Programs using these statements

have to be rewritten.

AVAIL
BUFFER#
CATLINE
CATFILE
CHECKREAD ON/OFF
DET without parameter
DIRECT
DIRECT NOUPDATE
DOOR LOCK/UNLOCK
DUPTEST
EDIT
EDIT KEY
HOLE
INDIRECT
LINK
LIST KEY
LOAD / STORE BIN
ON / OFF DELAY
PRINT LABEL
READ LABEL
REQUEST
RELEASE
RES
SD / SI
SET DATE TO
SET TIME TO
SYSID$

All TASK Statements

All PERFORM Statements

All TIO Statements

All MEDIA Statements

<u>In IMAGE</u>

IN DATA SET DIM ALL
IN DATA SET USE ALL replace through IN DATA SET USE <list>
IN DATA SET IN COM
IN DATA SET FREE
DBCLOSE MODE 4
DBINFO Mode 4xx
DBCREATE
DBERASE
DBMAINT
DBPASS
DBPURGE
DBRESTORE
DBSTORE
READ / WRITE DBPASSWORD
XCOPY

There are no QUERY controlnumbers and no corresponding DBINFO

Performance hints

Like in phase I there is also an interpreter and compiler in phase II but unlike phase I all the new statements of phase II are compilable without exception.

The purpose of the interpreter is mainly for programmers use only or at most for one shot programs as its speed is too slow for everyday production programs. So its advantage is primarily for program development, testing and symbolic debugging in its fullest sense, because in the interpreter all source information is still available. You can not only look at or modify variables but also add, delete or modify the source program.

For enduser applications the compiled code is the only reasonable way to go, maybe with some programs needing noncompilable statements still interpreted. Remember you can call compiled programs from the interpreter but not viceversa.

Some speed ratios HP260 vs HP3000/37

HPBB interpreter 2-5 times slower than BASIC/260

HPBB compiler 5-10 times faster than interpreter

-> on the /37 a compiled program is on the average 2 times faster than the interpreted program on the 260

That shows clearly that the /37 is no upgrade path for users who have outgrown their HP260

Comparison to other languages

So far only measurements with phase I have been made as phase II features are not available in most other languages and they indicate a slightly slower performance compared to PASCAL/COBOL, we talk about 10 to 20% for CPU bound programs. For I/O or IMAGE bound programs there is of course not much difference since most of their time is spent in the system intrinsics anyhow.

With the coming phase II extensions performance should favor HPBB because of the really powerful statements especially if you take into account a faster development cycle and better and more flexible programs.

Please note that all tests are preliminary as especially phase II is not yet performance tuned.

For ways of improving performance even more by switching off some off the features like dynamic error checking etc in the compiler see the appropriate sections in the HPBB manual.

Experiences with the conversion process

Those of you familiar with BASIC/260 can probably tell by now that converting existing 260 application packages still requires some manual work but also that there are no real big obstacles. Just how much effort do prospective OEMs still have to put in before their programs are running in HPBB and in an HP3000 environment.

Let me give you some of the results we have got in converting our first packages.

Before any conversion can be done some very timeconsuming preparations have to be made first, I mean preparing all your files and databases for transfer and the filetransfer itself. There are two ways two do it, a slow and cheap one using a RS232 connection or a faster, expensive one using INPs. We had only the slow connection available at 9600 Baud (HP recommends only 4800, but we didn't have any problems) and with the preparations on the 260 side and the filetransfer you can easily spend some 3-4 days or more, mostly depending on the size of your database which has to be unloaded, transfered and loaded set by set. Conversion of databases is straightforward as the 260 uses almost the same IMAGE subsystem.

The good thing is most preparation and transfer procedures can be automated using PERFORM on the 260 or batchfiles for the transfer, so this can easily run overnight. An example is shown in the appendix, without further comment.

The cheap transfer tool is TRNSFR a 260 utility provided with HPBB, the expensive one is DSN/DS and INP hardware on both sides.

The conversion itself is done by a supplied utility BBCT250 which takes care of all the syntax changes and some but not all of the semantic changes. After this is done the converted program is loaded into HPBB with the GET command and now its your time to do the remaining manual adaptation.

The worst problem we encountered was a program too big for HPBB to accept in one piece. As the 260 has a single 64 KB address space for one user it is possible to write programs up to 60 KB as a single subroutine or main program. Unfortunately this is not acceptable by HPBB, it can only accept so called subunits (main, subroutine or function) up to approx. 20-25 KB or about 400-500 lines of code but of course many of these subunits i.e. the whole program can get much larger but there is that damned restriction on the subunit size.

Well no problem, we said, we just segment the big ones into some smaller subroutines and a control program. Easier said than done, I have to admit because some of them just turned out to be ugly, unstructured monster programs which were much simpler to rewrite than to segment and convert. But we were lucky, these programs were less than 10% of all, nevertheless this subunitsize restriction can be a serious bottleneck for a successful conversion effort.

The small enough programs (90% or more) were relatively easy to convert the conversion program BBCT250 doing most of the work. However some minor manual changes still had to be made. (The following list is not complete)

Create a unique subprogram name, on the 260 all our application subprograms were called Pgm because they were all selectively loaded just before the CALL by the menu.

Change all references to the database statusarray to doubleword references, the 260 only had 16 bit pointers and capacities

Make file and formnames acceptable to the 3000, the 260 allowed upper and lowercase and also special characters

Change DBGET, DBPUT and DBUPDATE and insert the USING clause, the 260 did the variable assignment dynamically, here it is static

When using direct wordpointers for accessing files, check them because due to other storage requirements they may have changed

The biggest manual changes however requiring some knowledge about the programflow are adapting the IN DATASET, THREAD, SEARCH and SORT statements as their logic is somewhat different.

And some minor stuff, some of them could be done using the HPBB editor functions and by redirecting HPBB's inputfile to a commandfile.

All in all it took us about one hour per program for the conversion. I think an acceptable result as it offers a whole new dataprocessing environment keeping OEMs software investments and almost the same functionality in the applications. Thus a user having to move to a bigger machine due to size limitations of the 260 can immediately continue to work in a familiar way without a timeconsuming learning process.

Schedule

for the current phases

Phase I released in mid 85

Phase II and Joinform beta test in February 86
release mid 86

Preview into the future

Further development within HPBB (no HP commitment yet)

After the release of phase II in mid 86 there might be ongoing efforts to enhance further the useability, power and friendliness of HPBB.

Several topics have been discussed including:

File sort statements, thus getting rid of intrinsic calls

data structures, at least simple ones

printing into and reading from strings, like PL/1 GET/PUT STRING

support of fully qualified filenames for distributed systems

full screen editor with immediate syntax check within HPBB

and some more

All these points are still subject to discussion and no commitment has been made by HP so far.

Porting HPBB to SPECTRUM

This project started in Nov 85

According to Dave Elliot (HPBB manager phase I) about 10% of the existing code has to be changed to adapt HPBB to the new hardware but for performance reasons approximately 20% will be modified. HPs plan is to release HPBB on SPECTRUM at the same time as on the 3000, mid 86.

Concerning database access it is planned to provide a softwarelayer between HPBBs DB statements and the new database so that a programmer can continue using his well known IMAGE DB statements without changes.

In doing this he will of course not be able to exploit the full range of the features of the new database, therefore additional DB statements have to be integrated too.

Right now no direct interface is in discussion between HPBB and the SQL like DB language, this can be done at first using e.g. a PASCAL subprogram.

This is probably of not such a big concern to HPBB users because already todays THREAD, SEARCh and SORT commands provide relational like access to IMAGE.

Increasing compatibility downwards to the 260

As the 260 despite its limitations is still going strong especially in Europe some tentative projects are discussed (JOIN/260) enhancing the BASIC/260 language with constructs from HPBB and maybe even make some changes to already existing statements with the goal of being more compatible with the grownup members of the family.

With all these activities going on you can clearly see that Business Basic has a strong support within HP. HPBB is planned to be the commercial Basic for all future systems, from the 260 across the 3000 up to SPECTRUM.

Acknowledgements

I am very grateful to HP Böblingen and Cupertino for enabling me to stay with the research team directly in Cupertino for over 3 months. Also to my company for letting me go for such a long time to ensure the usefullness of the product for 260 OEMs.

My thanks also go to the HPBB team for the good cooperation despite my being sometimes a very uncomfortable demanding customer but I think it turned out to benefit both sides, HP and HPBB user.

Authors biography

Dr.Ulrich Fauser
has been with WEIGANG MCS since 1981.

He is R&D manager and system software specialist therefore responsible for the selection of future hard- and software for use within his company.

He also deals with the europewide system software and technical support for all used computersystems, right now exclusively HP products.

He studied computer science at the university of Stuttgart with emphasis on compiler construction and microprogramming. While he was teaching computer science there afterwards his interest shifted to distributed relational database systems of which a prototype was implemented by his research team. This has been also the subject of his doctoral thesis in 1980.

Before he started with his present company he spent a year at the University of California in Santa Cruz researching within Prof. F.Deremers Translator Writing Systems project.

His primary interest today are operating systems, databases and programming languages.

Privately his favorite pastimes are sailing, skiing and scuba diving.

References

HPBB Reference Manual , 32115-90001

HPBB Quick Reference Guide , 32115-90002

HPBB Programmers Guide , 32115-90003

BASIC/3000 to HPBB Conversion Guide , 32115-90004

BASIC/250 to HPBB Conversion Guide , 32115-90005

Phase II External Reference Specification

JOINFORM External Reference Specification

no manuals for phase II are available yet

```
 1     :! Perform "PERF01:M"    ! COMMAND file
 2     :! Files, beginning with FBH (FIBU) and of type
 3     :! PROG, DATA or FORM , are read per CATLINE from
 4     :! tape and copied to :M, PROG files are saved as
 5     :! ASCII-Files
 6     :! In a parallel process an EDITOR file is created,
 7     :! later to be used as a batchfile for the TRNSFR program
 8     :! The program WART1 only waits for the parallel task to
 8.1   :! be in INPUT state
 9     :! ************************************************
10     REQUEST#7
11     SEND CONTROL HALT#7
12     SEND COMMAND#7,"RUN "&CHR$(34)&"EDITOR:M"&CHR$(34)
13     RUN"WART1:M"
14     SEND INPUT#7,"S LENGTH=160"
15     RUN"WART1:M"
16     SEND INPUT#7,"A"
17     RUN"WART1:M"
18     :DIM PARM(8)
19     :SET PARM(1) TO 0
20     :LOOP
21     :SET PARM(1) TO PARM(1)+1
22     CATLINE PARM(1) ON ":K" ,A$
23     :EXIT IF A$[1,3]="EOD"
24     :IF A$[1,2]="FB"  THEN
25     :IF A$[14;4]="PROG" OR A$[14;4]="DATA" OR A$[14;4]="FORM" THEN
26     :SET PARM(5) TO NUM(A$[4;1])
27     :SET PARM(6) TO NUM(A$[5;1])
28     :SET PARM(7) TO NUM(A$[6;1])
29     :IF A$[14;4]="PROG" THEN
30     LOAD A$[1,6]&":K"
31     B$="FBP"&CHR$(PARM(5))&CHR$(PARM(6))&CHR$(PARM(7))&":M"
32     SAVE B$
33     SEND INPUT#7,"DATA_FILE_TRANSFER "&B$&" TO FBH"&B$[4,6]
34     RUN"WART1:M"
35     :ELSE
36     :IF A$[14;4]="DATA" THEN
37     B$="FBD"&CHR$(PARM(5))&CHR$(PARM(6))&CHR$(PARM(7))&":M"
38     :END IF
39     :IF A$[14;4]="FORM" THEN
40     B$="FBF"&CHR$(PARM(5))&CHR$(PARM(6))&CHR$(PARM(7))&":M"
41     :END IF
42     SEND INPUT#7,"ARCHIVE_TRANSFER "&B$&" TO FBH"&B$[4,6]
43     C$="FBH"&B$[4,6]&":K"
44     COPY C$ TO B$
45     RUN"WART1:M"
46     :END IF
47     :END IF
48     :END IF
49     :END LOOP
50     SEND INPUT#7,"//"
51     RUN"WART1:M"
52     SEND INPUT#7,"K"&CHR$(34)&"TRATES:M"&CHR$(34)&",UNN"
53     RUN"WART1:M"
54     SEND INPUT#7,"E"
55     :END IF
56     :END
```

```
 1     :! SAMPLE OUTPUT OF PERFORM COMMAND FILE
 1.1
 2     DATA_FILE_TRANSFER FBP044:M TO FBH044
 3     DATA_FILE_TRANSFER FBP045:M TO FBH045
 4     DATA_FILE_TRANSFER FBP046:M TO FBH046
 5     DATA_FILE_TRANSFER FBP047:M TO FBH047
 6     DATA_FILE_TRANSFER FBP048:M TO FBH048
 7     DATA_FILE_TRANSFER FBP049:M TO FBH049
 8     DATA_FILE_TRANSFER FBP051:M TO FBH051
 9     DATA_FILE_TRANSFER FBP052:M TO FBH052
10     DATA_FILE_TRANSFER FBP054:M TO FBH054
11     DATA_FILE_TRANSFER FBP065:M TO FBH065
12     DATA_FILE_TRANSFER FBP057:M TO FBH057
13     DATA_FILE_TRANSFER FBP055:M TO FBH055
14     DATA_FILE_TRANSFER FBP053:M TO FBH053
15     DATA_FILE_TRANSFER FBP056:M TO FBH056
16     DATA_FILE_TRANSFER FBP058:M TO FBH058
17     DATA_FILE_TRANSFER FBP059:M TO FBH059
18     ARCHIVE_TRANSFER FBF301:M TO FBH301
19     ARCHIVE_TRANSFER FBF302:M TO FBH302
20     ARCHIVE_TRANSFER FBF303:M TO FBH303
21     ARCHIVE_TRANSFER FBF304:M TO FBH304
22     ARCHIVE_TRANSFER FBF305:M TO FBH305
23     ARCHIVE_TRANSFER FBF306:M TO FBH306
24     ARCHIVE_TRANSFER FBF307:M TO FBH307
25     ARCHIVE_TRANSFER FBF308:M TO FBH308
26     ARCHIVE_TRANSFER FBF309:M TO FBH309
27     ARCHIVE_TRANSFER FBF310:M TO FBH310
28     ARCHIVE_TRANSFER FBF311:M TO FBH311
29     ARCHIVE_TRANSFER FBF312:M TO FBH312
30     ARCHIVE_TRANSFER FBF313:M TO FBH313
31     ARCHIVE_TRANSFER FBF314:M TO FBH314
32     ARCHIVE_TRANSFER FBF315:M TO FBH315
33     ARCHIVE_TRANSFER FBF316:M TO FBH316
34     ARCHIVE_TRANSFER FBF317:M TO FBH317
35     ARCHIVE_TRANSFER FBF318:M TO FBH318
36     ARCHIVE_TRANSFER FBF319:M TO FBH319
37     ARCHIVE_TRANSFER FBF320:M TO FBH320
38     ARCHIVE_TRANSFER FBF321:M TO FBH321
39     ARCHIVE_TRANSFER FBF322:M TO FBH322
```

```
! simple HPBB example program demontrating database extensions
! and report writer
! not every detail shown
! (line numbers have been ommitted)
! YES I know it can be done in a different and easier way

SUB Orderreport
  COM Dbas$,Limit  ! lets assume database opened excusive in MENU

! Variable definitions only partially

  DIM Partno$[16],Storageloc$[6]
  INTEGER Customer,Orderno
  DECIMAL Quantity


Dset1: IN DATASET "ORDERS" USE Partno$,Customer,Orderno,Quantity,
       Salesarea,Orderdate$
Dset2: IN DATASET "PARTSMASTER" USE Partno$
Dset3: IN DATASET "PARTS" USE Partno$,Partdesc$,SKIP 46,Storageloc$

  Prt=FNSelectprinter      ! ask user which printer to use
  SYSTEM "FILE LP;DEV="+VAL$(Prt)
  DBASE IS Dbas$
  ASSIGN #1 TO "WORK"
Input: !
  OFF KEY
  CALL Clearscreen
  CALL Ask_user("Report for which salesarea",Area$)
  CALL Ask_user("for which month",Month$)
  WORKFILE IS #1
  POSITION #1;RESET
Thread1: THREAD IS Dset1,Dset2,Dset3
  SEARCH USING Thread1;Salesarea=VAL(Area$) AND VAL(Orderdate$[3;2])
      =VAL(Month$)
  IF NUMREC(#1)=0 THEN
    CALL Msg("No orders for this salesarea and month")
    ON KEY 8 GOTO Final;LABEL="EXIT"
    ON KEY 2 GOTO Input;LABEL="NEW INPUT"
    LOOP
      ACCEPT
    END LOOP
  END IF
  IF NUMREC(#1)>1 THEN SORT USING Thread1;Customer,Partno$
  ON HALT GOTO Exit
  SEND OUTPUT TO "*LP"
  BEGIN REPORT Rpt1
  FOR I=1 TO NUMREC(#1)
    READ #1;Orderptr,Dummy,Partptr
    DBGET Dbas$ USING Dset1;DATASET="ORDERS",MODE=4,KEY=Orderptr
    DBGET Dbas$ USING Dset3;DATASET="PARTS",MODE=4,KEY=Partptr
    DETAIL LINE 1 USING Dt1;Orderno,Quantity
  NEXT I
  END REPORT
Exit: STOP REPORT
Final: SEND OUTPUT TO DISPLAY
  SYSTEM "RESET LP"
SUBEXIT           ! return to menu
```

```
! Report layout for monthly order printout

Rpt1: REPORT HEADER WITH 3 LINES
    GRAND TOTALS ON Quantity*Price
    PAGE LENGTH 72,2,2
    LEFT MARGIN 10
    BREAK 1 WHEN Customer CHANGES
    BREAK 2 WHEN Partno$ CHANGES
    PRINT USING Hd1;Title$,Area$
  PAGE HEADER WITH 2 LINES
    PRINT USING Ph1;DATE$(1),TIME$
  PAGE TRAILER WITH 0 LINES          ! can also be omitted
  HEADER 1 WITH 2 LINES
    TOTALS ON Quantity*Price
    GOSUB Readcustomername
    PRINT USING Hd1;"Orders for "+Custname$
    HEADER 2 WITH 2 LINES
      TOTALS ON Quantity,Quantity*Price
      PRINT USING Hd2;"Orders for "+Partno$,Partdesc$,Storageloc$
    TRAILER 2 WITH 2 LINES
      PRINT USING Tr2;"Quantity for this part ",TOTAL(2,1)
      PRINT USING Tr2_1;"Ordervalue ",TOTAL(2,2)
  TRAILER 1 WITH 2 LINES
    PRINT USING Tr1;"Ordervalue for this customer ",TOTAL(1,1)
  REPORT TRAILER WITH 3 LINES
    PRINT USING Rtr1;"Total ordervalue for salesarea",TOTAL(0,1)
    IF TOTAL(0,1)<Limit THEN PRINT "Orderlimit not reached"
  REPORT EXIT WITH 3 LINES
    PRINT USING Re1;"TERMINATION DUE TO USERREQUEST"
  END REPORT DESCRIPTION

! Output format specifications would be here

SUBEND

! Other subroutines like Ask_user, FNSelectprinter etc would follow
```