68 1001 000 RvD 1968
Submitted to 4th European DECUS Seminar Edinburgh 1968

RECENT HARDWARE AND SOFTWARE DEVELOPMENTS FOR PDP-8

I.K.O. Bol *)
Institute for Nuclear Physics Research (IKO)
Amsterdam, Holland

ABSTRACT

Aspects and applications of:
PDP-8 binary program relocation, scatter-read magnetic tape, dynamic storage allocation, multi-job processing (time/memory sharing), asynchronous file handling, program tracing, autonomous display and a computer-computer conversational scheme as in use at IKO are indicated.

General Introduction.

At I.K.O. four PDP-8 computers have been installed for different tasks. The first of these, now in use for several years, has become part of the complex nuclear detection system "Bol", especially designed for coincidence experiments.

In this paper we will concentrate on work around this particular PDP-8 only as far as it has led to generally applicable results. In addition to being interfaced to the experimental set up, the PDP-8 has been coupled to:

- 2 Datamec 2020, 17 kc tape units
- an extended Nuclear Data M-unit for autonomous display with lightpen
- a medium size computer EL-X8

Although this PDP-8 has only 4 k memory it is able to control all its peripherals in a macroscopically simultaneous manner. This was achieved by developing a multi job supervisory system MONIKOR, and by reducing software processing-time as well as -space.

*) Identifies a group of people participating in the BOL-project (construction of a 64-fold nuclear detection system);
R. van Dantzig, J.E.J. Oberski, K. Mulder, L.A.Ch. Koerts
M.A.A. Sonnemans, J.L. Visschers, A.D. Ypenberg
W.C.M. Biekmann, P.G. van Engen, J.C.A. van Gessel, A. Mars
R.P. Meesters, F.A. van Hall, P.U. ten Kate, J. Kraus,
E. Kwakkel

principal author RvD

This system allows independent and semi-independent job programs to be run on a time- and memory sharing basis. The job-programs are task-oriented modular-programs protected and synchronized, using queuing-techniques. They can be written in PAL III by applying some additional rules. Job-programs are kept under supervision as far as dynamic storage allocation, activation and synchronozation are concerned. They are read into core during run time of other job-programs from either magnetic tape or paper tape. Job-programs are rejected whenever inactive and not protected, if the process runs short of space. Job-programs may occupy non-contiguous memory pages. A suitable pagewise allocation technique allows this.

Section 1 of this paper gives a brief discussion of the simple program relocation method we use.
Section 2 reviews some applications of useful hardware features of our magnetic tape system. Section 3 indicates how dynamic storage allocation is implemented into the PDP-8 system. The supervisory system MONIKOR is briefly discussed in Section 4. Section 5 contains a macroscopic description of an asynchronous file handling system while a helpful hardware circuit, enabling powerful program tracing is mentioned in Section 6. In Section 7 some aspects of the philosophy of our computer-computer interfacing are mentioned. Section 8, finally, contains some remarks on an autonomous display unit (with lightpen) coupled to the PDP-8.
The appendices contain some additional details.

1. Program Relocation.

The possibility for reading in a binary program in an ad hoc specified memory region is of importance for an efficient use of core memory.

Once such relocation possibilities are created, a Dynamic Storage Allocation technique (viz. 3) as e.g. required for our multi-job processing system MONIKOR (viz. 4) can be developed.

The relocation technique we have chosen allows for shifts of programs or program parts over in integral number of memory pages (200(8) addresses). "Keys" are used to link independently relocated programs, "relocators" to provide the additional information for relocation (viz. I). The technique is simple and has proved to be quite useful. The binary paper tape format we use (viz. I) is compatible with fixed and with relocatable

loading.
This compatibility implies:

- all programs relocatable or fixed are punched in paper tape in a common format called SHRIMP-format.
- all SHRIMP formatted papertapes can be loaded with a simpler loader, the SHRIMP loader (viz. II), which can store programs at a"fixed" predetermined memory region. The loader provides for sum checking and optional "load and go". Relocatable programs are loaded at their "nominal" location. The SHRIMP-loader occupies 25 (10) memory locations and in our system replaces both the RIM and BIN loader from DEC.
- for relocation of "normal" relocatable programs two loaders are present, the SHARKY-loader and the SHARK-loader. The SHARKY (viz. II.2) loader is the simpler one. It occupies 103 (10) locations and can be located at the last memory page together with the SHRIMP loader.

The SHARK (viz. II.3) loader has additional facilities such as self-protection, printed documentation of occupied memory regions, and read-compare mode.
For "JOB"-programs suitable for parallel processing (viz. 4) a special loader (job) has been created, which allows programs with a length of several memory pages to be loaded in multi job processing environment, and to be scattered over the memory (viz. 3).

All loaders are relocatable themselves. Relocatable programs in SHRIMP format can be obtained using either a modified PAL III assembler or a special punch program (length: 103(10) locations).

The ideas of the SHRIMP-relocation technique have been used in a completely analogous fashion for loading programs from magnetic tape. A feature facilitating the application of this technique is the scatter-read facility for magnetic tapes as incorporated in our tape interfaces (viz. 2).

2. Scatter-read magnetic tape.

A magnetic tape control unit providing interrupted reading facilities has been designed. The facilities are useful for (relocatable) loading of binary programs in both single- and multi-job processing environment.

They amount to the possibility of reading in, tape records at non contiguous memory regions (viz. III).

This has proven to lead to short and relatively simple programs allowing for fast program monitoring and library processing.

This may be clear from the following:

-any program may be contained in one tape record. Programs occupying several disconnected contiguous memory regions of arbitrary length may be packed in a single tape record. Thus unnecessary interrecord gaps and associated waiting times are avoided.
When the program is relocatable, eventual key blocks as well as relocators (viz. 1) are all contained in a single (program) record.

-the length of a (program) record to be loaded from tape is in general unknown at the beginning of a search. Our tape format allows the tape programs to search for and to load a desired program in a single pass of the record containing that program. There is no need to stop, backspace and reread, nor is it necessary that a library description be stored at the beginning of the tape.

The automatic boot-strapping mechanism for magnetic tape may illustrate the foregoing:
Using the SHRIMP loader a short tape loader is read into core via the teletype. With the auto-start facility of the SHRIMP loader it is automatically started.
The tapeloader reads in from magnetic tape the library loader which again is automatically started. A package of programs, relocatable and fixed, may thereafter be loaded on either keyboard or papertape commands.

3. Dynamic storage allocation.

Memory sharing of asynchronous programs (viz. 4) requires a well-adapted dynamic storage allocation technique. In the technique we applied, the memory is divided into two regions: the free region and the occupied region. The regions are subdivided into pages of 200(8) memory locations. The boundary between the free- and the occupied region is dynamic and in fact the regions may pagewise penetrate each other. The administration consists mainly of a list of 32 page-status words,

one word for each page. Two functions embeded in the MONIKOR system (viz. 4) control the memory sharing administration: GETPA (get page) and PUTPA (put page).
In any legal job program the calls of these functions are paired off, so that every job finishes only after giving up all space reserved during runtime.
The problem that arises when any program requests space not available at that moment, is dealt with in the context of the "critical point" mechanism discussed in section 4.

Program allocation: whenever a job program is loaded under MONIKOR supervision the GETPA routine is used to obtain a sufficient number of pages. Since the loader program itself is also a job program, operating in parallel with other programs, successively allotted pages in general are not contiguous.
In order to use the dynamic storage algorithm for program allocation it is necessary that programs be relocatable by page rather than as a unit. By requiring that any direct transition from one page to another occurs via a jump, or subroutine call, the relocation method becomes trivial and the format can be kept identical to the SHRIMP format or the corresponding magnetic tape format.

4. Multi-job processing, time/memory sharing.

The MONIKOR supervisory-system allowing several independent or semi-independent job-programs to be run simultaneously has been mentioned in previous pages.

The system has grown as a consequence of a number of requirements dictated by the application of the PDP-8 as a coordinating element among several relatively autonomous peripherals including the experimental set up, two tape units, a display unit, another computer and an "operator/physicist". MONIKOR itself, however, is independent of the periphery (except the teletype).

Successively improved versions of the MONIKOR-system have been satisfactorily used in our institute for about one year.

The characteristics of the system can be formulated to include:

- possibility of programming on a high level.
- modular programs (so-called jobs) usually with minimal restrictions to size or hierarchic level.

- increased flexibility in combination of jobs and their mutual activation.
- jobs that run asynchronously on a macroscopic level.
- time divided between "active" job-programs only.
- switching times between jobs in the 10 micro-second range.
- if memory space is needed, space occupied by jobs which are not "busy" is used.
- jobs read in from paper tape or magnetic tape as needed.
- loaders (jobs themselves) that run simultaneously with the other programs.
- continuous access to the active processes by the "operator/physicist".
- standard functions (those within the system too) available for all jobs.
- waiting times minimized by using queuing-mechanisms.
- a free space of 20(10) pages for simultaneously active jobs (the system occupies 12(10) pages).

The system:

MONIKOR consists of a set of bookkeeping subroutines, a central administration and some standard job-programs such as paper tape/magnetic tape loader and a teletype job-program. The operator has almost continuously access to the system by using the command-job (COJOB), which enables him to "attach" new high level jobs (so-called protocols) to the running process and to examine and modify internal conditions or address contents.
A protocol-job, once started, automatically triggers a tree-structured process. Program-jobs synchronized at critical points are the building blocks of the process.
From this moment on, eventually not already attached jobs are either read in from magentic tape or requested via teletype, automatically.

Jobs: job-programming for the MONIKOR system requires some additional rules, because the mutual synchronization of different jobs and peripherals is realised by using private semaphores in the jobs themselves.

These semaphores are inserted at "critical points" which could be defined to be points where the program waits for the change of some internal or external condition.

5. Asynchronous information flows.

Within the framework of the WINDOW- processing method *) a general input-output file handling system has been designed and implemented in the MONIKOR system. This I/O system handles any information channel including the experimental set up, the tape units, the display unit and the EL-X8 computer. Even though the PDP-8 WINDOW-system is only a simple realization of the WINDOW processing ideas **) it appears to be useful.
The following main characteristics result:

- Interacting asynchronous input- and output programs "switch" information at the file level between one input channel and a number of output channels. Information is transferred between the input- and output-programs using the dynamic storage allocation mechanism as mentioned in section 3.
- The input- and output-programs are jobs, cooperating with the MONIKOR system.
- The flow of information chopped up into info-records, and headers ***) (length: 1 page or less) may be passed along "windows". These are non standard subroutines, to be specified at the hoghest (protocol) level, which may inspect and operate upon the information between input- and output. Window-routines may also prevent transfer of selected records to the output-side (skipping). Window-routines may be omitted; in that case a program of a few standard elements may specify a complete file process.
- The numbers of internal buffers (length: 1 page) is a variable parameter of the process.
- Buffer memory is reserved only when actually needed.
- Different I/O processes may run simultaneously.
- Output jobs wainting for (input) information do not take up processor time. Synchronization is accomplished using the semaphore-functions of the MONIKOR.

*) to be published

**) proceedings of the symposium on Nuclear Electronics, Versailles, 1968

***) Headers are special records acting as separators and specificators of nested macroscopic quanta of information.

6. Program tracing.

In this section a method is presented that allows programs to be "traced" and tested while completely under dynamic program supervision.
The method requires an extremely simple hardware facility, called: DELAYED SELF-INTERRUPT (DELSI).
DELSI consists of one single flag which can give a program interrupt.
The flag can be skipped upon, it can be cleared and set, the latter with a delay of about 5μs.
The delay time is chosen to permit exit from the interrupt sequence and to allow at most one selected instuction to be executed. At the completion of that instuction a program interrupt is again initiated *****). This means that the program can be interlaced in time by a tracer-interrupt-routine, thus operating at the background of the program.

In this interrupt-routine the program status may be investigated and dependent on certain criteria, to be selected and adjusted by the operator, a message to the operator may be given or tracing may be suspended.
Several tracer programs have been intensively used in the past year. Due to the simplicity of the method it is possible to obtain easily a "special purpose" tracer program. A very simple tracer may be as short as half a memory page.

The main advantage of the DELSI circuit is that programmatic awareness, except during execution of interrupt-routines, is constantly guaranteed without time- and space consuming interpretative instruction simulation. Also instruction modifications such as: DCA. + 1 do not disturb the tracer program.

A frequently used tracer program (CHASER), is under teletype control. It offers a number of standard facilities. For example, tracer messages are given when the program under control is about to: run outside/inside a specified memory region, to modify a specified address, to execute a specified instruction, etc.

*****) When a data break disturbs proper timing, the DELSI-flag is already up before the next test-instruction is executed. This is unimportant, it amounts to a dummy cycle. The IOF instruction has to be dealt with separately.

A tracer message consists of the octal printout of the current value of: the program counter, the address and the value of the last executed instructions, the content of accumulator and link or any subset of these.

Apart from "supervised runs", so called "blind runs" like the ODT program from DEC are possible. The latter may be used to test interrupt routines.

7. Computer-computer conversional scheme.

An interface with its associated software has been designed *) to couple a PDP-8 to a medium size computer of the type Electrologica X-8 **). The requirements which determined the adopted philosophy and subsequent design were dictated by the on-line function of the X-8 as a "hypervisor" of the complex nuclear experimental BOL-system. Direct supervision of the experiment and associated tasks such as recording, displaying and dumping of data are given to the PDP-8.
The primary tasks of the X-8 during an experiment can be summarized as 2-dimensional histogration on drumstore and checking with regard to the quality and stability of recorded data. Simultaneously the arrays resulting from histogration should be accessable for the physicists in order to get proper insight into the functioning of the experimental system.

The way in which the communication-game between different program levels in each computer and between the computers themselves has to be played, is called the conversational scheme.
The communication takes place at different levels, while both machines run their own programs in multi- or single run environment. All communication is devided into "conversations". In both machines, conversations are "opened" and "closed" by programs of the highest level.

A conversation is protected against "re-opening" using a queuing mechanism.

*) Cooperation of N.V. Electrologica is gratefully acknowledged.

**) Specifications: 32 k memory, 27 bits 1.5 μsec cycle time, independent I/O processor and data channel synchronizer, 54 bits floating point arithmetic, drum (500 k), line printer, 4 120 kc CDC tape units 3 teletypes, papertape reader/punch.

Once a conversation is opened both machines have mutually synchronized programs on each side among other asynchronous programs.

At this stage only "introductory" communication is possible. This introductory communication consists of the exchange of "introductory sentences", 120 bit messages. These are used for further program synchronization as well as for task delegation. Next stage in communication can be reached by the opening of "datafiles". Two files can be opened simultaneously, one for each direction. Once a datafile is opened, "data sentences" can be transferred in the direction associated with that file. Data sentences have arbitrary length up to a maximum specific for each data file and arbitrary location. Data files for the two directions may be interlaced almost arbitrarily. A few rules prevent the communication to get into a "dead end" due to simultaneous mutual dependence of high level programs on each side.

8. Autonomous display for nuclear data exploration.

A Nuclear Data-160 M unit has been interfaced to a PDP-8. The unit consists of an 18 bits, 4 k memory with control unit allowing automatic recycling of the memory for different types of point displays. The display unit has been extended with an oscilloscope interface providing several hardware facilities useful for data exploration and -modification. The system is in particular designed for spectrum analysis and nuclear particle-identification. The main facilities of the system are:

- 2 dimensional with adjustable viewing angle. The display can be changed continuously from a topview through an isomeric view into a side view.
- Selected lines or points can be displayed with increased light intensity.
- A computer controlled three bits intensity register allows 8 grades of intensity.
- Different lay-out configurations can be chosen: 1 x 4096, 16 x 256, 32 x 128 and 64 x 64.
- The PDP-8 program is able to write and read the display memory and to start and stop autonomous display runs.
- 12 bit of each 18 bit word are used for Y-deflection

the other 6 are function bits.

- Flicker is reduced by two hardware provisions, which can be switched on and off by the program:
  - interline
  - chaining

  In interline mode alternately only odd and only even addresses are displayed; in chaining mode a "jumpfunction bit" indicates whether the next display cycle should be at the next address (jumpbit = 0) or at the address contained in the (last) memory word with jumpbit = 1. The display chain is to short-circuit uninteresting regions in the display memory, thus increasing the refresh-frequency.
- A lightpen can be used in two different modes. In the "mark mode" lightpen indication is hardware recorded into the "lightpen function bit", of the corresponding display word. In "signal mode" lightpen indication is directly carried over to the program without modification of the display memory.

APPENDIX I

Format description for papertape.

A SHRIMP format papertape has the following form +):

```
 < leader trailer >
⌘ < program-group identifier >
⌘ < program identifier >
 < ASCII program name >
⌘< program version identifier >
⌘ < nominal origin setting >
  < word >
  < word >
      :          relocatable or
      :          fixed program-block
  < word >
⌘ < nominal origin setting >
  < word >
  < word >       fixed program-
      :          or key-blocks
      :
⌘   7777
⌘ < relocator >
⌘ < relocator > relocator
      :          list
      :
⌘ < relocator >
⌘   ØØØØ        end of tape indication
⌘ < checksum >
⌘   7777
  <   terminal instruction >
```

Remarks:

- The identifiers and program name at the head of the program are used for automatic program searching and checking.
- The version identifier indicates also whether the tape is relocatable or not.
- Relocatable programs may have only one "relocatable" program block. All following blocks are in that case by definition "Key-block".

+) ⌘ has in SHRIMP format the same representation as the origin setting in BIN format.

- A key-block is a program block for which the actual origin setting is definitely equal to the nominal origin-setting while the content of the origin-address will be modified. This modification is such that if the origin-address would contain a pointer (or key) to any location in the nominally located program, it will do so after relocation. In general a key-block contains only one word, the "key", which actually may serve to link different relocated programs via a fixed address (preferentially being taken at page Ø). When a key-block contains more than one word, the remaining words are located without modification. A set of subroutines may thus be made relocatable, keys ensuring that from any place in the memory the subroutines may be called wherever they have been loaded.
- The relocator list may be empty.
- The relocators are the nominal addresses in the program where internal address pointers are placed, i.e. locations of which the addresses and the content are being modified in case of relocation. They have to be specified by the programmer although for symbolic programs automatic detection of these relocators could be done by double translation.
- The checksum has been defined slightly different than for the BIN format.
- The terminal instruction may be any instruction to be executed after loading the program with the SHRIMP loader. In most cases it is a HLT (74Ø2).
- Addresses Ø and 7777 are excluded from loading.

APPENDIX II

1. SHRIMP loader

The SHRIMP loader has been developed from an idea of N. CHASE *). Though slightly longer than the DEC-RIM loader, it seems competitive with the BIN-loader; while short enough to be toggled in using the switch register.

The octal representation of the SHRIMP loader as normally used at the top of the memory (starting address = 7753) is:

*) DECUSCOPE 5/7 (1966) p. 6

≠ 7747; 3346; 7ØØ4; 1377; 1ØØØ; 3ØØØ; 6Ø32; 536Ø; 6Ø36;
7166; 7146; 6Ø31; 5361; 751Ø; 5356; 7ØØ6; 6Ø34; 3377; 1377;
743Ø; 5347; 3746; 2346; 535Ø; 1ØØØ.

The SHRIMPloader uses the addresses Ø and 7777.
Features: load and go. The SHRIMP loader has a load and go facility which gives the following possibilities:

- program auto-start: when the program is loaded it may automatically be started.
- loading interlude: during loading a program part which just has been read into core may be executed while at the completion of this interlude program loading may be automatically resumed.

  12 bits checksum held in the accumulator at the completion of loading.

The SHRIMP loader loads RIM format tapes normally.

2. SHARKY loader.

Specifications:
purpose: loading of relocatable and fixed SHRIMP formatted programs
lenght: 1Ø3 (1Ø) locations
nominal starting address: 76ØØ
use: program to be loaded in reader

The SHARKY loader when started, reads and skips ASCII characters. It examines the version identifier to detect whether the offered program is relocatable or not (viz. I). When the program is not relocatable it is loaded normally. When the program is relocatable, the loader halts, displaying in the accumulator the relative page address of the initial origin-setting. The operator may now define, using the switch register, the page where the actual origin-setting should be located. After having pressed the continue-switch the program is loaded at the specified memory region.

3. SHARK loader.

Specifications:
purpose: loading of relocations and fixed SHRIMP formatted programs
length: 256(1Ø)locations

nominal starting address: 2ØØ
use: similar to SHARKY loader

Keeping track of the location of different relocated programs is easy using this loader, due to the printout of program identifier and the occupied memory regions. The operator thus may be constantly aware of where his programs are. The loader protects itself against overwriting. It further has a read-compare facility. By putting the switch register equal to zero during loading each word being stored is first compared to the present value. When the new and old value are different the storage address with the old and the new value are printed. This feature may be of help in detecting various types of errors.

APPENDIX III

Facilities scatter-read magnetic tape.

The interface allows the following types of commands to be accepted (TA = tape address, TW = word count):

a. READ BLOCK "NORMAL MODE" maximally TW WORDS to begin with TA .
b. WRITE BLOCK "NORMAL MODE", maximally TW WORDS to begin with TA .
c. READ BLOCK "SCATTER MODE", the first TW WORDS to begin with TA .
d. READ BLOCK "WORD MODE", (MAX)TW WORDS to begin with TA .
e. The commands: REWIND, UNLOAD, SPACE FORWARD, SPACE BACKWARD, WRITE BLANK 7", SET BCD/BIN PARITY.

at a. and b.:
"NORMAL MODE" reading and writing takes place using the data break memory-access, an interrupt flag being raised at the completion of the transfer. On inspection of the tape-status register the program may check on the correct execution of the command. When during reading the actual tape record contains more words than specified, transfer to the memory is stopped if the word-count is exhausted, Apart from this, the command will be normally completed.

at c:
For a record read in SCATTER MODE, the word-count reaching the value zero, will not simply terminate the data transfer.

It will initiate an interrupt which if properly handled by the program may lead to a continuation of reading in a newly specified region in memory. Since the interrupt has to be handled before the next tape-word assembled in the data-word register is ready for transfer, this interrupt should be handled with utmost priority. The interface, however, should not rely on program timing. Therefore, in the interface investigation is made whether the interrupt handling and eventual reloading of registers has occured in time. If not, a timing error flag is raised, memory transfer is stopped. The program upon detecting the error will reread the previous record.

at d.:
During reading in WORD MODE, at the completion of each data breakcycle a flag is raised signalling to the program that a new data word has been stored. This mode is particularly useful. Address advance may be suppressed. With this option, part of a record or an entire record may be examined using a memory buffer-region of a single word. Important is the possibility of switching between this mode and the other modes (program searching and loading, viz.2).

It will initiate an interrupt which if properly handled by the program may lead to a continuation of reading in a newly specified region in memory. Since the interrupt has to be handled before the next tape-word assembled in the data-word register is ready for transfer, this interrupt should be handled with utmost priority. The interface, however, should not rely on program timing. Therefore, in the interface investigation is made whether the interrupt handling and eventual reloading of registers has occured in time. If not, a timing error flag is raised, memory transfer is stopped. The program upon detecting the error will reread the previous record.

at d.:
During reading in WORD MODE, at the completion of each data breakcycle a flag is raised signalling to the program that a new data word has been stored. This mode is particularly useful. Address advance may be suppressed. With this option, part of a record or an entire record may be examined using a memory buffer-region of a single word. Important is the possibility of switching between this mode and the other modes (program searching and loading, viz.2).