OPERATING DATA DOCUMENTATION SYSTEM FOR A RESEARCH REACTOR

F.Kasparec, J.Hammer**
Atominstitut Vienna, Austria

1. History and Discussion of Requirements

The documentation of reactor operating data is an important task at research reactor stations. Though not directly neccessary for the safety of the reactor, it is an essential tool for maintenance and safeguarding purposes and therefore made obligatory by the licensing authorities.

Until 1985, the documentation system of the TRIGA reactor in Vienna comprised an automatic data logger with a separate hardcopy terminal. A textual report on all channel inputs was printed out each hour at normal conditions. When given thresholds were exceeded, warnings or alarms were issued together with an out-of-order log message, which conveyed information about the offending parameters.

In 1985 the development of the documentation system described in this paper was started. It was based on former experiences with the data logger which had led to a selection of representative data and useful data formats, but also made clear that the paper reports produced by the hardcopy terminal were not useful for further evaluation.

The documentation system was expected to be an extension of the logger-based system. Data had to be directed to the in-house computer center rather than to a terminal, though the terminal would be kept for reasons of diversity.

In particular, the documentation system had to meet the following requirements:

a) The central computer used for data archivation (a VAX-11/750) had not to be obstructed beyond neccessity. It was essential to avoid time-consuming high priority jobs.

b) Data security was neccessary, in the sense that VAX system crashes would not cause losses of acquired information.

c) A high degree of automatization, especially in the system crash and recovery phases of the VAX, had to be implemented in order to minimize human failures.

d) Upon request, data had to be available at any terminal of the VAX within seconds or minutes. (This was assumed to be particularly important in the case of warnings or alarms.)

e) Data management tasks (sorting into daily reports etc.) had to be performed

---

*) At present at Alcatel-Elin Research Center Vienna, Austria (F.Kasparec) and PSI Würenlingen, Switzerland (J.Hammer)

automatically, so the operator would only have to copy data to tape and clear up the VAX disk directory every week or month.

f) The transmission protocol used to transmit collected data to the VAX had to be portable, i.e. not relying on any specific hardware.

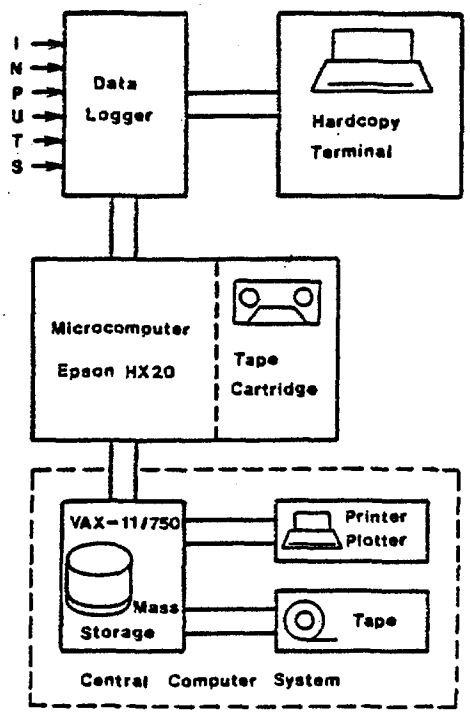

FIG.1: SYSTEM STRUCTURE

2. System Description

2.1. Microcomputer as Prebuffer

To meet all these requirements, special prebuffer hardware was provided to connect the data logger to the VAX. Because of its proven reliability an Epson-HX20 microcomputer was selected for this purpose.

Today a personal computer would also be a good choice, especially because of its large memory and built-in mass storage which is far more powerful than the microcassette drive of the HX20. However, the fact that the operating system of a PC is resident in dynamic RAM may cause reliability problems, which would have to be solved by special watchdog hardware (e.g. a MAX-691 chip on a prototype adapter card in a PC extension slot).

Fig.1 depicts the structure of the documentation system hardware obtained by introducing the microcomputer.

The microcomputer software performs five tasks:

a) **Input Processor:** This task processes the data-logger output in order to cut the data stream into records.
The logger output stream is a sequence of ASCII-coded text lines of 23 characters each, called frames, which contain information about the result of a physical measurement, a comment, or information about date and time. The latter case occurs whenever a new measurement cycle starts and is used by the input processor task to identify the begin of a new record (Fig.2). The number of frames per record is variable.
The input processor task also assumes the end of the current record whenever the stream pauses for more than 10 seconds. The frame following a pause always is a date/time frame.

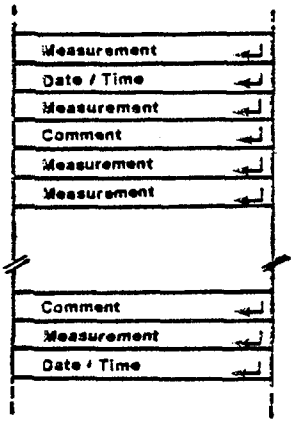

FIG.2: LOGGER OUTPUT

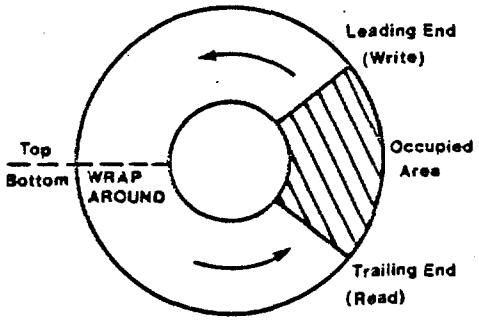

FIG.3: RING BUFFER ARCHITECTURE

b) **Buffer Management:** This task handles accesses to a 24kByte ring buffer, using a special algorithm which will be discussed in detail in chapter 3. All newly received data from the data logger are written to the leading end of the occupied memory area as if they were pushed on a stack, whilst data for transmission to the VAX are taken from the trailing end.
Since any buffer access beyond top or below bottom is wrapped around, repeated accesses make the occupied memory area wander through the ring (Fig.3).

c) **VAX Interface:** This task serves transmission requests issued by the VAX by sending entire data records to a VAX terminal port, which are then cleared out of the ring buffer.
As only whole records are handled, the time-out mechanism mentioned at the end of (a) makes sure that the most recent information becomes available as soon as the data logger stops sending. Without time-out, the input processor task would have to wait for the anticipated date/time frame (leader frame) of the next record to identify the end of the current record.
All communication runs through a serial 1.2kBaud link using 20mA current loop standard. A special protocol, which will be discussed in detail in chapter 3, traps line breaks and VAX system crashes, so no data are lost.

d) **Buffer Supervision and Tape Access Control:** This task periodically checks the ring buffer occupation. If the amount of occupied memory area exceeds an adjustable threshold higher than 16kByte, a 16kByte section of stored data is scanned for end-of-record marks, proceeding from the trailing buffer end in forward direction. The last end-of-record mark found identifies a memory block containing the largest number of entire records that fit into the 16kByte space (see Fig.4). This block is then saved out to tape in reverse order.
After successful completion of every transmission cycle to the VAX, which is assumed to have emptied the buffer except for partial records (when input from the data logger is in progress), this task checks for the existence of data blocks on tape. In case that some is found the block is read back and appended to the trailing buffer end, proceeding backward character by character. In case that none is found the VAX is informed that there are no outstanding data.

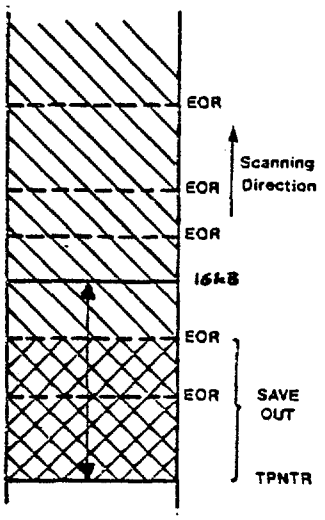

FIG.4: IDENTIFYING A SAVE-OUT BLOCK

VAX requests received during an already active save-out or read-back operation are rejected. Typically, the reason for the impending buffer overflow is a line break or VAX system crash, so requests need not be expected during a save-out operation.

e) **Tape Management / Operator Interface:** This task controls the HX20 micro-cassette drive as well as operator accesses to this drive, which are the only manual intervention possibly required. Together with six multicolor signal lamps the HX20 liquid crystal display informs the operator about the system state.
Tape management is stack based, i.e. the blocks exchanged with task (d) are stored first-in / last-out. Each of the MC90 tape cartridges used can contain four 16kByte blocks in distinct sections. The cartridges need no formatting; the sections are assumed at fixed offsets and accessed by first rewinding the tape to the begin and then winding forward to the desired location.
This task needs the most extensive functional modification to run on a PC.

Fig. 5 illustrates the data flow among the five tasks.
Serial communication via the main data path (input from data logger and output to VAX) is fully interrupt-driven. Such a design was possible by using an asynchronous communication adaptor (ACIA) chip in an extension box attached to the HX20, whereas input from the VAX is polled.
There is also a possibility to output messages to the data logger for the purpose of dynamic reprogramming. If, for instance, task (d) anticipates an overflow condition and finds the tape cartridge to be full, the data logger could be automatically reprogrammed to drop useful, but redundant information and restrict its output stream to absolutely neccessary data. As this facility has not been implemented in the actual software version, the corresponding line in Fig.5 has been drawn dashed.
Since the five tasks partly run in parallel and there is no system programming language available for the HX20, the HX20-software has been written in assembler.

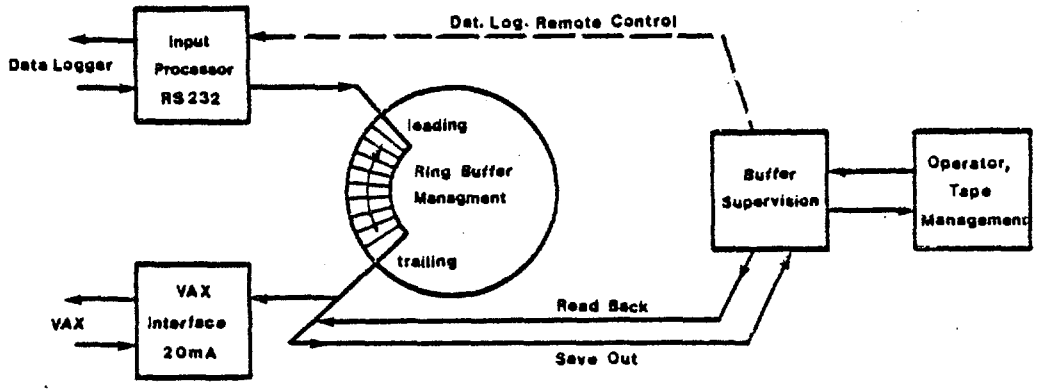

FIG.5: DATA FLOW AMONG PREBUFFER-RESIDENT TASKS

2.2. VAX Software

The VAX, on the other hand, also executes software, matching to the VAX Interface task on the HX20 (or PC). This software comprises three modules:

a) the **Controller Module**, which holds the code of an always resident controller task. This task is responsible for calling one or more acquisition tasks at fixed time intervals or upon request from the operating system.

b) the **Acquisition Module**, holding the code for all acquisition tasks, of which each serves an individual buffer subprocessor (HX20 or PC). As the present configuration contains only one HX20, only one acquisition task has been implemented.

c) the **Sorter Module**, which provides the code for all sorter tasks. Acquisition tasks and sorter tasks are organized in pairs, in a way that each acquisition task activates an individual sorter task on exit. As there is only one acquisition task, only one sorter task can be active in the present system configuration.
The activation does not take place unless the acquisition task has been informed by the assigned buffer subprocessor that there are no outstanding data on tape (also see buffer subprocessor task (d)).
The sorter task reads a workfile provided by the acquisition task, sorts the records contained there into a time-ordered sequence using their leader frames (date/time frames) as key, puts them into daily report files, and finally deletes the workfile.
If there were no accesses to tape, the workfile is time-ordered. On the other hand, Fig.6 gives an example of five data stream sections in the workfile of which two have been saved out to the tape stack. Since the section limits depend on the time when save-out or transmission to the VAX are started, the section definition is entirely random.
The workfile has to be preserved till the end of the last read-back and then sorted section by section or- as section lengths are undefined- record by record.
The Sorter Module uses the subroutine calls of the VAX/VMS Sort-Merge Utility, which exhibits a good behavior at this particular sorting problem.

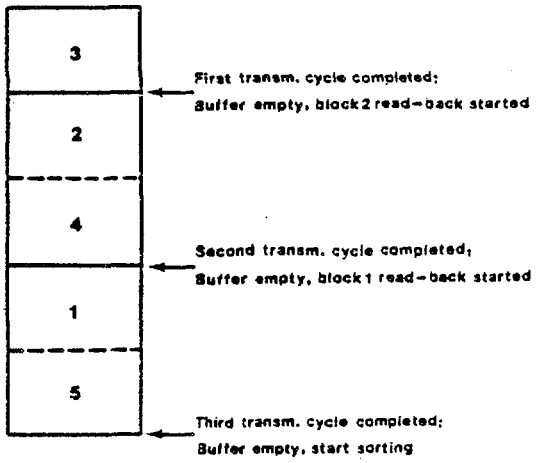

FIG.6: EXAMPLE OF TAPE-READ-BACK ORDER

Fig.7 gives a concise formal description of the VAX task interaction, using the real-time language Occam /1/. Although this language has been developed for concurrent programming on transputer systems, it is a good tool to describe software that extensively relies on VAX/VMS low level system calls.

An alternative description language could be seen in PEARL ('Process and Experiment Automation Real-Time Language'), as it is defined in the papers by the 'Kernforschungszentrum Karlsruhe' /2/.

Among the VAX/VMS system calls used, there are process hibernation, scheduled wakeup, wakeup enforced by a parallel process, as well as event flag services which are used for locking the acquisition task, in order to grant the operator exclusive access to the workfile for testing purposes.

Except for the Sorter Module, which has been written in Fortran, all VAX software has been programmed in VAX Macro-Assembler.

```
PAR
  WHILE TRUE
    SEQ
      PAR
        SEQ
          ... Acquisition Task 1
          IF
            no_outstanding_data
              ... Sorter Task 1
            TRUE
              SKIP
        ... Other Acquisition / Sorter Pairs (SEQs within PAR)
      ... Compute next_time (= schedule wakeup)
      ALT
        (TIMER ? AFTER next_time) & not_locked
          SKIP
        (forced_wakeup ? any) & not_locked
          SKIP
    -- End of infinite-loop body (outermost sequencial construct)
  ... Other jobs under VAX/VMS (write to channel forced_wakeup)
```

FIG.7: VAX TASK INTERACTION (FUNCTION OF THE CONTROLLER TASK)

In Fig.7 SEQ denotes a process comprising any number of subprocesses (next indentation level in the list) to be executed sequentially. In a similar way, PAR indicates a process consisting of two or more parallel (concurrent) subprocesses of which the slowest is responsible for the termination of the PAR.

An ALT construct specifies two or more alternative processes which can be triggered by individual timer, interprocess-communication or real-time events masked by static conditions ('&' operator). Only the alternative triggered first of all is executed; its termination is that of the entire ALT.

SKIP is a dummy process used in constructs where only communication or synchronisation is of interest.

3. Fault Tolerance Precautions

In the context of this paper the term 'Fault Tolerance' expresses that VAX system crashes and transmission errors are trapped. No means has been provided to meet hardware faults in the data logger or prebuffer system, because this could not be accomplished except by extensive hardware redundancy.

In transmitting data from the microcomputer (prebuffer) to the VAX, a record is the smallest unit that can be handled by the protocol. Whenever a line break occurs during transmission, both sides respond with a time-out and the last, only partially received record is discarded in the VAX.

Hence the buffer management task in the microcomputer must use a double-pointer algorithm. This algorithm distinguishes *character-level pointers* with names ending in 'PNTR' and *record-level pointers* with names ending in 'REC'.

A *transmission pointer* (TPNTR) always holds the address of the next byte to be sent to the line. This definition implies that TPNTR is incremented *after* every byte transmission, as is usual in system programming.

A *transmission record pointer* (TREC) is overwritten by the current TPNTR value whenever TPNTR points to the first byte of a record and the preceding end-of-record mark - which has been used to identify the record begin - has been succesfully transmitted. Hence TREC never can point at other locations than record begins and always is updated record by record.

When aborting the transmission, TPNTR is set back to the current TREC value, so the next communication cycle will start with the begin of the record which was affected by the error.

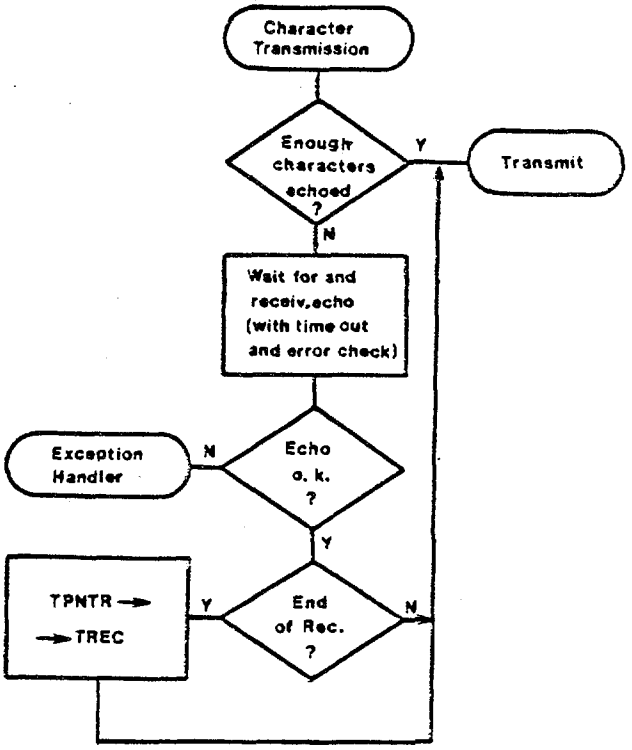

FIG.8: PREPARING TRANSMISSION OF ONE CHARACTER

There are two possible reasons for a communication abort, closely related to the error-checking mechanism used. This mechanism is based on the *echo* returned by the VAX terminal driver whenever a character is fetched by the VAX CPU. Such an echo-checking protocol requires a full-duplex line, but has the advantage that no software overhead in the VAX is caused as is by protocols like XModem or Kermit.

A communication abort by the microcomputer may be caused by an echo time-out or by an accumulation of echo errors. In case that an erroneous echo

arrives a limited number of retries is done, making the VAX discard the record torso by sending it a backspace character and then restarting at TREC position. If the retry limit is exceeded or the backspace is not echoed properly, communication is shut down.

As some time is required for a character to travel to the VAX and back, at least three unechoed characters must be on the way if the line shall not be slowed down. (See /3/ for a detailed description.) For this reason the VAX interface task in the microcomputer allows a credit of five characters, except if an end-of-record mark or the first character after an end-of-record or backspace (i.e. the leading character of a record) is to be sent.

The first exception ensures that the record has arrived properly before the VAX is informed to accept it definitely. The second exception does not allow to continue with a new record as long as it is not sure that the old one is on disk.

Taking this into account and regarding the TREC update condition explained above, TREC obviously must be set equal to TPNTR when an end-of-record echo is received (see flowchart in Fig. 8).

The double pointer philosophy is also used at the leading buffer end, which is controlled by the input processor task. Here the pointer pair is named RPNTR and RREC. RREC is overwritten by RPNTR whenever the last character written to the buffer is an end-of-record mark (Fig.9). Transmission to the VAX always stops at RREC, so records just being received by the data logger remain unaffected.

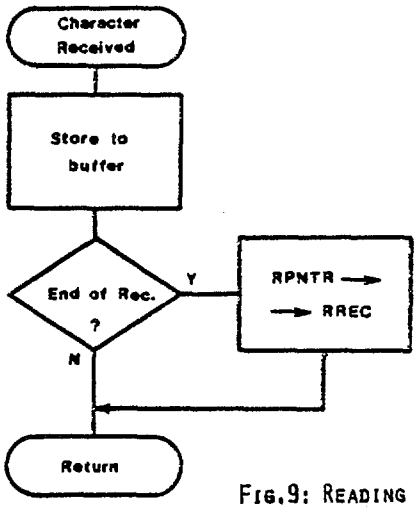

FIG.9: READING ONE CHARACTER

In the microcomputer-resident Buffer Supervision / Tape Access Control task (task (d) in chapter 2.1.) a pointer named SREC is used to cut off the save-out block, whereas SPNTR runs through the buffer during tape operations. However, save-out is done on a predecrement base whereas read-back operations keep to the post-increment convention.

It should also be mentioned that every transmission burst (from the prebuffer to the VAX) is preceded by a password exchange, so it is difficult to interrupt the line and extract data from the prebuffer by simulating a VAX request.

The VAX always begins the exchange by sending a password to the microcomputer. The microcomputer then answers with another password, which is not echoed by the VAX, and starts the transmission protocol.

However, since it is easy to monitor a serial line, this is a facility for avoiding accidental data losses rather than a protection against 'hackers'. In endangered areas glass fiber cables should be used, as they are hard to tap. Assemblies with optical transmitters and receivers integrated in RS232 connector cases, which are available from various manufactures, allow data rates up to 100kBaud over distances of 40m and more with excellent reliability at a cheap price.

4. Experiences and Conclusion

By now, the Documentation System has been operational for about ten months, collecting a daily average of 30kByte of data, and overcome a number of VAX system crashes, maintenance days and VAX/VMS updates.

During this time various software extensions have been developed for processing the daily reports. Among these extensions there are programs for convenient tape backup, procedures for formatted printer output, and other.

It is especially helpful to use commercial personal computer software for further data processing, in particular when PCs are connected in a network with the VAX or, on the other hand, emulate VAX terminals. There is a vast number of spread-sheet or database programs with excellent computational and graphic support available for PCs. Most of them allow data import, though format conversion programs will be neccessary in most cases.

Fig.10 gives an example where nuclear reactor power and radiation level at the reactor pool surface (the latter measured by an aerosol detector) are set against each other graphically, using the program MS-Chart by Microsoft. The measurement time extended over ten days, including a week-end. The picture is based on data collected by the Documentation System.

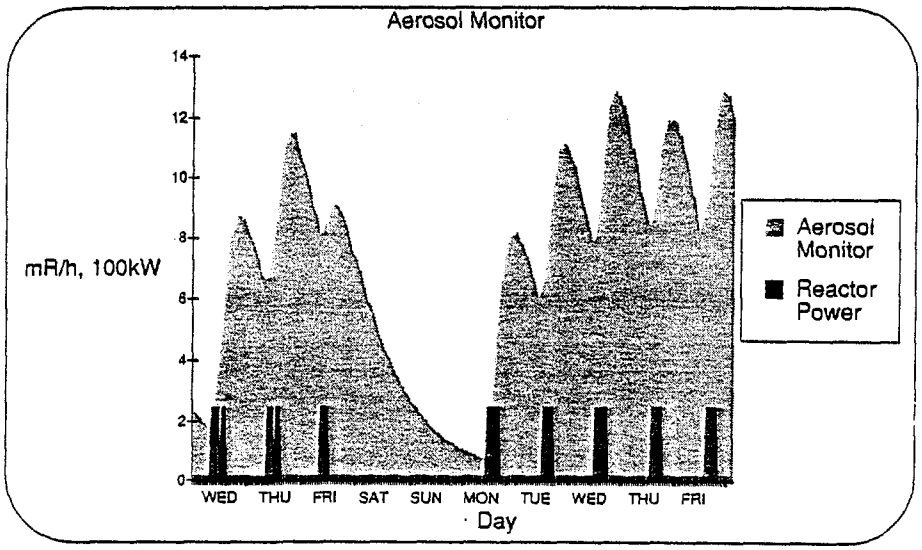

FIG.10: AEROSOL ACTIVITY AT POOL SURFACE (WED: JUN 15, 1988 - FRI: JUN 24, 1988)

References:

/1/ 'OCCAM Reference Manual', INMOS Ltd.

/2/ 'Full PEARL Language Description', Ges. für Kernforschung m.b.H. Karlsruhe, PDV-Report KFK-PDV 130

/3/ F.Kasparec, 'Betriebsdaten-Dokumentationssystem für Forschungsreaktoren' (Diploma Thesis, Tech. Univ. Vienna)

/4/ J.Hammer, 'Computer Controlled Area Surveillance System for the TRIGA Mark II Reactor Vienna', Acta Physica Hungarica 59 (1985)