THE OMEGA PROJECT: A CASE HISTORY IN THE DESIGN AND IMPLEMENTATION OF AN ON-LINE DATA ACQUISITION SYSTEM

R.D. Russell
CERN, Geneva, Switzerland

I. INTRODUCTION

In April 1967 a CERN working group was formed to study the feasibility of building a large magnet with wider solid-angle acceptance and larger useful volume within the magnetic field than any magnet previously built for a physics laboratory. With such a powerful tool, experiments having highly complex interactions could be performed with detection of more secondary particles than previously possibly. After considerable investigation, this group published a proposal in May 1968 for building at CERN a large aperture spectrometer magnet with associated detector and data handling systems that should become operational in 1972. This proposal formed the basis of the Omega Project. Within these specifications, the project is given a very broad scope. It is not restricted to any single experiment, but is to be a general facility, similar to a bubble chamber, which can be used by many experiments having a wide range of requirements. It is expected to have a useable lifetime in excess of ten years, and therefore had to be designed to anticipate developments in both physics and technology; frequent modifications were to be expected.

As originally proposed in 1968, the primary tool for detecting and measuring physics events was to be a set of optical spark chambers located in the magnetic field. Initially all events were to be recorded on film, but with an anticipated advance in technology, the importance of the film readout was expected to decline as more and more experiments made exclusive use of filmless detector systems. So rapid has been the advance of detector and readout technology that the film system was abandoned entirely more than a year before the first scheduled operations, long before the magnet itself was completely constructed.

This illustrates one of the primary characteristics of the Omega Project's short history: its definitions and requirements are always
changing. Many changes are minor, but some, such as the replacement of the film readout system with plumbicon cameras, require considerable reevaluation of the entire project, especially the data handling aspects. We therefore cannot talk about the Omega Project as a static set of definitions and proposals. The Omega Project of 1972 is very different from that of 1970, and will undoubtedly be completely different in 1976, when operation of the 300 GEV machine begins. Fortunately this rapid evolution was anticipated in the original proposal, and throughout the course of the project flexibility and generality have been the primary design principles.

This paper is divided into two parts. The first deals with the historical evolution of the Omega computing system, and concentrates mainly on the problems that occurred and the decisions that had to be made during the past few years. The second part will deal with some of the more interesting technical accomplishments of the Omega data handling system. Since this project was intended to produce an advanced physics facility, the computing system includes many advance ideas and techniques that should be of general interest. The author is a visiting scientist at CERN who joined the Omega Project in the summer of 1971. Since he was not involved in the planning or development of Omega prior to that time, it must be understood that all comments and criticisms reflect an attempt to objectively appraise the project's evolution from the point of view of one who has seen the results of the decisions without participating in the debates that surrounded them.
II. THE EVOLUTION OF OMEGA

A. The Beginning

The original report distinguished three main functions of the data handling system:

1) On-line data acquisition and control
2) Data reduction (pattern recognition)
3) Data analysis (geometry and kinematics).

Although these three categories remain more or less valid today as production uses of the system, they are incomplete, since they omit all uses of the system as a development tool. There are at least two additional phases, which experience has shown to be equally or even more important than those originally mentioned. These are:

4) Computer system development (hardware and software)
5) Experimental system development (hardware and software).

Recognition of these last two functions as distinctly different uses of the computing system caused several of the more basic changes to the original computer system specifications. Until June of this year development activities have been the sole occupation of both the computer group and the physicists, since there was no beam before June, and hence no data to acquire, reduce, and analyse. Since the beginning of the project, use of the computer system has been essential for its own development and for the development of the experimental systems. This use of the computer as a development tool will not stop with the start of production data taking. If anything, it may become more important now that feedback from the production activities is available, and now that changes to equipment configurations and software must be made rapidly yet accurately in order to avoid loss of precious beam time.
B. The Initial Data Handling System

The Omega computer system specified in the working group's proposal of May 1968 is shown in Figure 1. It consists of a main computer and a small control computer, both on-line to the Omega experiment, and an HPD film scanner with associated control computer that is connected to the main computer. The main computer was specified as the primary data acquisition and data reduction computer. During filmless experiments, it would acquire data directly from the experiment's electronics and would record it onto magnetic tape after some rudimentary filtering. It would also perform some calculations on a small fraction of this data in order to give real-time feedback to the physicist about the quality of the data and the course of the experiment. These calculations would be simple consistency checks that could be done without floating-point hardware. A CRT display connected to the main computer would be used to display histograms of simply calculated parameters, such as spark frequency distributions.

It was calculated that a medium-sized computer with fast fixed-point operations could perform the production pattern recognition tasks more economically than the larger central installation computers. Therefore, when not on-line to the filmless experiments, the main computer would be executing pattern recognition programs on data received from the on-line HPD film scanner, and would be engaged in pattern recognition of recorded events from both sources. The CRT display would enable a human operator with a light pen to interact with the pattern recognition programs in order to recover events that could not be processed automatically.

The use of the main computer during the central part of the project's lifetime was projected to be roughly the following: 17% of the time for on-line filmless data acquisition; 10% of the time for film scanning with the on-line HPD; 40% of the time for pattern recognition of about five million filmless events; and 13% of the time for recognition of the one million film events. The remaining 20% of the available time was allotted for general development work. The
later stages in the processing chain (geometric reconstruction, kinematics, and statistical analysis and curve fitting) would be done on the more powerful central installation at CERN and in the computing centres of the member states.

The inclusion of a small control computer that would also be on-line to the filmless equipment was clearly a secondary consideration in the original proposal, where it was 'suggested ... for reasons of economy and safety', rather than being required as the absolute necessity it quickly proved to be. Its primary functions were for data acquisition and control during film-mode data taking, and as an aid in setting up and testing equipment. During filmless experiments, it would be by-passed by the normal data flow to the main computer, and could be allocated to control functions. It would be used for on-line data taking only if the main computer should fail.

Although it is easy to criticize this original proposal after four years experience with the system, I would nevertheless like to point but a few problems with it. Basically it underestimates both the value of Omega as a physics facility and the need for computing power to utilize this facility effectively during non-production operations. Although the report includes proposals for several experiments and indicates possibilities for Omega's use in future experiments, it did not seem to envision the overwhelming response of the physics community to the potential of the Omega facility. Within two years more than a dozen experiments had been proposed for Omega. Since most of these experiments involve complex triggering systems that require many months to set up and test, and since beam time in the Omega area is both limited and expensive, it was soon obvious that tasks originally expected to occur sequentially would have to be performed simultaneously in order to make maximum utilization of the facility. This brings us to the second shortcoming of the original proposal: Its lack of sufficient emphasis on the non-production aspects of the facility. Production data taking and analysis constitute
only a small fraction of an experiment's life-time. The majority of the time is spent in designing, developing, and testing both hardware and software. The original proposal emphasized production while underestimating the value of the data handling system as a tool for setting up and testing experimental equipment. This resulted in considerable expansion of the data handling system when the increased demand for access to Omega made it necessary to perform these activities in parallel with production tasks. The specifications did however stress flexibility as an important design principal, and adherence to this goal made it possible to expand the data handling system easily as the need arose.

C. The 2 System

The Omega Working Group's proposal was published in May 1968. Within a year the first significant changes occurred in the specifications. Due to the extremely rapid development of direct readout technology, the original estimates on the use of film readout devices were revised downward from one million events per year over a period of ten years to about 3000,000 events per year for only the first two years of Omega operation. Since film detectors were to be used so little, and since they would be eliminated so soon in the project's lifetime, it was decided not to build a special HPD film scanner for connection to the Omega computer, but rather to modify the existing CERN HPD-1 so that it would accept the Omega pictures. This HPD would remain connected to the central computer installation at CERN, so that the Omega computer would no longer be required to perform any film scanning.

With the increased emphasis on direct readout detectors, the rôle of the on-line computer immediately became more important, and the development of the entire filmless data acquisition system became more urgent than in the timetable of the original proposal. As shown in Figure 2, the configuration of November 1969 has eliminated not only the HPD but also the direct connection between the main
Omega computer and the on-line equipment. This was due to the realization that during set up and testing phases, a new experiment would need the facilities of an on-line computer almost continually, and this type of on-line work would inevitably cause the computer to 'hang-up' fairly frequently, due to hardware and software bugs. Therefore, production pattern recognition programs, which were to be run on the main Omega computer when production data acquisition was not in progress, could not be executed without undue interference unless these two computer activities were isolated into two separate machines, as was specified in the original proposal. However, once equipment has been interfaced to the on-line computer, which may require several months of testing to demonstrate that it functions properly, it is clearly not desirable to switch the interface to another computer for production, since this would not only require retesting everything, but would require development of duplicate software for the two computers (as well as duplicate hardware interfaces). The triggering and detection systems are extremely complex so that interfacing to one computer is difficult enough. There is no reason to make it more complicated by changing the interface between production and testing. Therefore it was decided that all acquisition hardware would remain connected to the on-line computer during production as well as testing, rather than being switched to the main computer as originally specified.

An added factor in this decision was that if a sufficiently powerful on-line computer were selected, this machine would be able to perform all the data acquisition and recording onto magnetic tape for experiments having data rates below about 20,000 coordinates per burst. During such experiments, which were expected to consume about 20% of the real-time in a year, the main computer would be used only to perform sampling calculations on the data sent via the link from the on-line machine. For experiments with higher data rates, which were expected to occur only 10% of the real-time, the main computer would record the data onto magnetic tape as it arrived over the data
link from the on-line machine. During the remaining 70% of the year, the two computers would operate independently: The on-line machine being used for set up and testing, the main computer used for production pattern recognition and general development work.

D. Selecting a Computer

1. The requirements

Once the configuration shown in Figure 2 had been decided on, the choice of which computers to use had to be made. Without going into the details of the selection process (which lasted for many months), it is possible to delimit the major factors that were considered and their influence on the final decision.

Perhaps the biggest single factor was the data rates from the on-line experiments in filmless mode. The PS beam supplied to Omega consists of a 300 millisecond burst every two seconds. Although a maximum of 50-60 triggers per burst is possible with chambers having a five millisecond dead time, the highest average data rate was placed at about 30 triggers per burst. The maximum amount of data was estimated at 1500 16-bit parameters per trigger, which yields 45,000 coordinates per burst and an instantaneous transfer rate of 150,000 coordinates per second. Allowing for a buffer of about 25K of 32-bit words, a resident monitor, the acquisition programs, and the sample analysis program, a minimum memory size of 64K 32 or 36-bit words is seen to be necessary. The data rate also determines that tapes with a recording speed of at least 90 KC be attached to empty the buffer between bursts, but for reasons of future expansion, 120KC rates were preferable. It is interesting to note that at these rates, one tape would be filled every eight minutes, and the expected 50 million events per year would fill over 7000 tapes per year.

Since these figures were all estimates that depended on chamber dead time, amount of data acquired per trigger, and beam
characteristics, it was felt that all numbers (memory size, tape speed, etc.) should be exceeded by a comfortable margin, and that the system should be expandable to handle future increases in both size and speed.

Due to the high instantaneous data rates, it was specified that access to memory of both computers had to be one microsecond or less, in order to minimize read-out and transfer times. The main computer had to be capable of rapid fixed-point calculations in order to perform the on-line checking and the pattern recognition. Floating-point calculations were not expected to be necessary on-line, so that the floating-point hardware, while necessary, did not have to be exceedingly fast. The size of a memory word was placed at 32 or 36 bits, for numerical accuracy, but with operations available on the 16 or 18-bit half word, so that spark chamber coordinates (of 13 to 15 bits) could be handled without unpacking yet without wasting space. The availability of a large drum or disc, holding at least 20 million characters, was a necessity for system libraries, overlays, etc. Because this computer was to be heavily used for development purposes, especially during the initial period of Omega operations, it was essential that the machine be able to provide a facility for continuous, self-operated software development. This required that the manufacturer supply a general multi-programming supervisor as well as the standard software (FORTRAN, assembler, linking loader) and standard peripherals (card reader, line printer, card punch, console teletype).

The on-line computer was not required to be as powerful as the main machine, but did have to be capable of sustaining the high data rates needed by the readout equipment. A smaller word size (16 or 18 bits) and a smaller memory size (16K or 24K) were acceptable, although a fast access speed (one microsecond or less) was still felt to be essential, as were powerful, flexible I/O channels. It was important that the word size of the main computer
be a multiple of the on-line machine's word size in order to minimize problems with the link transfers and tape formats. The small machine also had to have a disc for program and monitor use, and 60 KC to 120 KC tape drives in order to record the data as it was acquired from experiments with slower data rates. The required software consisted of a FORTRAN compiler, an assembler, a linking loader, and a foreground/background supervisor.

2. The choice

Once the detailed specifications had been determined, evaluation of the proposals made by the manufacturers began. Some of the considerations used in making the final decision are summarized in Table 1. Benchmark tests, consisting of a number of FORTRAN programs similar to those expected to be used heavily in Omega, were run on all the proposed computers. This permitted a comparison of several important parameters: compilation speed, execution speed, size of the compiler, and size of the object code. In addition a number of subjective factors were obtained from these benchmarks, such as 'usability' of the system and general reliability of the hardware, software, and documentation. These qualitative considerations produced some unexpected discoveries. Many of the compilers contained bugs, so that legal FORTRAN programs did not work. Others ran but gave wrong answers due to hardware malfunctioning. One manufacturer never did succeed in getting the programs to run.

Besides the benchmark tests, the manufacturers' proposals were evaluated on the basis of 'how well' they seemed to meet the Omega specifications. Especially important was the expandability of the proposed configuration, since Omega was expected to grow and the first system was only a 'minimum' facility. Thus the ability to add additional core memory, disc memory, and I/O channels was of prime importance. A second important consideration was to determine what was already available and what was merely
proposed for some future time. Manufacturers often 'promise' software and hardware features which are only in the design or testing phase, so that any delay in their development would cause a corresponding delay in Omega. It was especially important that Omega not become the first user of a major piece of hardware or software, thereby becoming the one to find all the manufacturer's bugs.

Since the Omega computers were to be part of a general facility containing several critical components (such as beam-on time), it was important to have a reliable computing system. The hardware had to be dependable during production data taking that might go on 24 hours a day for several weeks, and the software had to be secure from interference between the production tasks and other users. Reliability of the proposed machines was determined from the benchmark tests and by making visits to other installations having a similar machine.

When the hardware does malfunction, as it inevitably must, it is extremely important to have competent technical people readily available to service the fault as quickly as possible. It was felt that maintenance personnel based in Geneva, if not on the CERN site, were essential for the Omega machine. Maintenance considerations also made it desirable to keep the number of non-standard peripherals (other than the experimental equipment) to a minimum. Tape drives or discs not supplied and supported by the computer manufacturer were felt to be potentially dangerous from the maintenance point of view, especially when these devices were so crucial to the production data taking.

Software maintenance is also an important consideration, and depends on the willingness of the company to correct any problems quickly and to make improvements to its software as dictated by Omega's needs. It was therefore important to have direct access to the responsible technical people within the company, rather than having to deal with public relations and marketing personnel
for solutions to technical problems. A great deal about a company's policy toward its users can be determined from its documentation: whether it exists at all, and if so, how accurate and up-to-date it is. Since it was anticipated that parts of the operating system of any computer decided upon would have to be rewritten to satisfy the unique needs of Omega, it was extremely important that the manufacturer make available to the Omega systems group all the source code for the operating system. Without this, necessary modifications to the system simply could not be made.

After all the machines were evaluated in detail, the final choice came down to two that met all criteria satisfactorily except the last one: price. One machine was significantly more expensive than the other, so the less expensive proposal was accepted. This resulted in the choice of the CII 10070, shown in Figure 3, as the main Omega computer. A similar selection and evaluation procedure resulted in the choice of the EMR 6130, shown in Figure 4, as the Omega on-line computer.

E. The 3 System

In early 1970 a decision was made to combine the Omega data handling system with that of another CERN project: the Split Field Magnet (SFM) Project. The split field magnet is a large magnet system located in intersection region 14 of the Intersecting Storage Rings (ISR) and designed such that the net effect of the magnetic field on the circulating proton beams is zero. Although development work had already begun on SFM by this time, it was not expected to become fully operational until the beginning of 1973. Both projects have a long expected lifetime and will generate a large amount of data that must be acquired by an on-line computer. Since SFM is also a general facility with data structured similar to the Omega filmless data, and since all the processing problems are expected to be similar, it was decided that the best approach was to combine the data handling
systems and to assign a single systems group the task of implementing the combined system.

The layout of the new system is shown in Figure 5. Since the main computer and the on-line computer for the Omega Project had already been selected and the orders placed, the simplest adequate solution was to order a second EMR 6130 to be on-line to the SFM facility, and to connect the main computer to both on-line EMR's via direct data links so that it could perform real-time sample analysis for either experiment separately, or both simultaneously.

The impact of this enlargement of the system was twofold: First, it made the EMR 6130's the critical bottleneck, since all data acquisition for two experimental facilities were now absolutely dependent on the capabilities of that computer: Second, it necessitated a more generalized link handling scheme in the main computer. Although the generalized link handler would have been designed anyway, the shift in emphasis from the CII 10070 to the EMR 6130 as the key computer for on-line operations was totally unanticipated by the original specifications. Whereas it was previously felt to be merely 'desirable' to allocate all on-line set up and testing to the EMR, it now became essential, since if the main computer were used for such tasks, malfunctioning of a test program at one facility would not only destroy a production analysis run (which is not too serious, since the data, safely on tape, could always be re-analysed), it would also destroy the sample analysis and/or data recording of an on-line experiment at the other facility. Clearly such interference with production data-taking must be eliminated, or at least held to an absolute minimum. As a result, the EMR 6130 became the sole computer to be used for set up and testing, as well as the most important machine during on-line acquisition (without it, an experiment could not proceed).
F. Designing the System

During most of 1970 the specifications for the software system were worked out in some detail. This included a detailed analysis of the dialogue for communicating via the computer to computer links, the integration of the link software into the computer operating systems, specification of a data acquisition and sample analysis system for a standalone EMR 6130, and specification of the CII data handling system and its expected use by both experimental facilities. Basic design principles were to make a flexible, modular system that could be easily modified or expanded in any direction, and to utilize existing software as much as possible, both that supplied by the manufacturer and that already available on other computers at CERN. This lead to the decisions that the CII 10070 was to operate with the SIRIS 7 system supplied by the manufacturer, and that graphics on both computers were to be done with the GD3 graphics package already developed on the CERN central installation.

It was also felt to be important to make the system as fail-safe as possible, both against equipment malfunctioning and against errors made by inexperienced operators. This was to be accomplished by building a library of well tested procedures for all basic tasks, such as link communication, display driving, and histogramming, and by building extensive monitoring and error recovery facilities into the software system. On the CII 10070 this required time-out monitors on all I/O activity, especially that involving the links, a revision to the existing FORTRAN library so that errors could be better detected and handled by the program, and a 'SPY' program for monitoring the progress of both user programs and the operating system.

Many other mundane details had to be worked out during this period, such as standardizing the tape formats for compatibility between all machines (including those at the central installation), and providing standard bookkeeping procedures for handling the documentation, tape storage, accounting, file editing, and program libraries.
On the hardware side, it was decided to use CAMAC for all interfacing between the Omega EMR 6130 and the counter and trigger systems. At this time, the spark chamber readout system was to be interfaced directly to the on-line machines, but in early 1971 it was decided to route this through the general CAMAC interface of the EMR 6130 as well. For the SFM experiment the chamber readout system still does not use CAMAC, although most other on-line hardware does. Tektronix T4002 display terminals were chosen for connection to the EMR 6130's, but since the CII 10070 was to drive at least two interactive display terminals having considerably greater power than the output-only displays on the EMR, more time was needed to study the options available. By early 1971 the decision was made to use two more Tektronix T4002 storage tube displays with keyboard and tracker-ball driven by a mini-computer linked to the CII 10070.

G. Expanding the System

By October 1970 the implications of the critical rôle played by the EMR 6130 in the on-line phase of all experiments at the Omega facility was fully realized. At this time, proposals for over a dozen experiments using Omega had been received, and due to the limited number of 14-day PS beam periods available to the Omega area (seven per year) and the length of time needed to set up and test the triggering systems for each experiment, it had become clear that in order to perform all these experiments between the time the facility would start receiving beam (June 1972) and the time Omega was scheduled to shut down operations in order to convert from optical to direct readout detectors, a new experiment would have to be ready to start production data taking every six weeks. Such a schedule could be met only if three experimental groups could have simultaneous access to the Omega facility: one to set up the triggering system; one to test it; and one to actually use it for production data taking. In this manner an experiment would be able to stay on the floor for a four-month period on the average. Since the EMR 6130 was an essential
part of each of these activities, it was felt that the three groups would have to have simultaneous access to this computer, especially during the beam-on time.

This represents a significant departure from the original specification of the EMR's functions as a single-user system, and in fact resembles more the requirements originally specified for the main CII computer. However, since one EMR had already arrived on site, and since the other was already ordered, it was not possible to consider acquiring a different machine for the on-line work.

In January the situation of the on-line computer became even more critical when it was decided to replace film entirely and begin Omega operations in 1972 with a new readout device called a plumbicon camera. This decision was prompted by experience with a prototype system at Rutherford Laboratory, and as the readout electronics for that system had been successfully interfaced to their computer via CAMAC, it was decided to interface the Omega plumbicon system to the EMR 6130 via CAMAC also. This in turn led to a decision that all on-line hardware used in Omega would be connected and controlled with CAMAC.

The plumbicon camera operates in a manner similar to a television camera: visual light from the spark chambers is focused onto a photosensitive surface where it is scanned by an electron beam in such a manner that each line in the raster scan corresponds to a single gap between planes of wires in the chambers. The position of any spark in that gap can be placed by the camera's logic with an accuracy better than 0.5 mm, and the digitized position coordinates are available for readout into the computer at the end of each scan line.

The Omega chamber layout would require eight of these cameras, six inside the magnet and two outside. Each camera would scan about 45 lines including the fiducials, and would be provided with eight scalers so that up to eight sparks could be digitized
per camera per line. Scanning a single line would require 68.3 microseconds, and reading the 64 scalers into the EMR would require at least 102 microseconds, so that the total time to scan and read out all 45 gaps would be 7.7 milliseconds.

Replacement of film cameras with plumbicons implied that Omega would have to be ready for on-line production data taking more than two years before the date originally specified for the first filmless experiment. In addition the initial load on the whole system, and on the on-line EMR 6130 in particular, would be much higher than estimated, and the need for the EMR to set up and test the new plumbicon system as well as the three independent trigger systems made this computer even more critical than before to the success of the entire Omega Project.

Many possible expansions of the data handling system to relieve the critical load on the on-line computer were discussed during early 1971, and it is worthwhile here to examine a few of these to see what the possibilities appeared to be, and what the real constraints proved to be. The trend clearly discernable in these discussions was 'downward' expansion of the system toward a symmetric arrangement of identical on-line 'satellites' that could be used independently of each other. The result of these discussions was the 4 system that is discussed in the next section.

The first possible direction for expansion was to design an EMR operating system that would be capable of supporting three users simultaneously. This was considered only briefly, and was quickly discarded because of the overwhelming technical problems involved. The system would have to be written entirely at CERN, and could not have been a multiprogramming system in any commonly understood sense of the term, simply because it appeared to be impossible to develop such a system on that computer, especially in the short time available before it was needed by the physics experiments. Enormous problems were involved if even a severely limited system were to be attempted, all due to the fact that a small computer such as the EMR 6130 was
not designed to function as a multiprogramming computer. It was
too small (24K of 16-bit memory), lacked adequate peripherals, such
as displays and teletypes that would be needed by each on-line user,
had discs that were too slow for any reasonable roll in/roll out, and
perhaps most importantly, lacked an adequate memory mapping or pro-
tection scheme that is essential in order to isolate simultaneous
users from one another.

The problem of interfacing the EMR to more than one set of
on-line equipment was even more formidable. Due to the channel
configuration of the EMR and the design of the CAMAC interface, at
most two CAMAC systems could be coupled to the computer at any one
time. This made it impossible to permit each user his own independent
CAMAC system, and required that a single CAMAC system with multi-user
access be designed. Such a system is virtually impossible to make
technically fail-safe from programming errors, spurious signals,
channel blocking, etc. The only possible approach was to dictate
rigid 'good practice' methods for attaching equipment and utilizing
the interface during testing. But this assumes that both the hardware
and the software are already at an advanced stage of testing, and makes
the unavoidable hang-ups during set up catastrophic for the production
data-taking experiment. Clearly the usefulness of the EMR for
equipment set up and debugging was minimal, if not non-existent in
such a system.

A second possibility that was also quickly discarded was the
purchase of a second EMR 6130 for on-line use in Omega. The first
EMR would be used exclusively by the single production experiment,
and would be connected via a link to the CII for the on-line sample
analysis, as was the case in earlier systems. The new EMR would be
devoted exclusively to the set up and test users.

This suggestion, shown in Figure 6, has several advantages
over the multi-programming approach. The production user is safely
isolated from any interference with the test users, and the second
EMR can be used as a backup should the first one fail during a run.
However, there are many technical problems, since a user would have to interface his CAMAC configuration to one machine for testing and to another, although of the same type, for production. Furthermore a 'multi-user' operating system for the second EMR would still have to be developed at CERN in order to handle more than one test user at a time. This system would be essentially the restricted one proposed previously and there would still be the problem of connecting two test users to a single CAMAC interface. Of more immediate concern was the fact that this approach would have been too expensive, since even a minimal configuration of only 16K memory, paper tape I/O and a disc (no card reader, line printer, tape units, or data link) exceeded the available Omega budget.

During this period discussions of a different though related nature were also going on in Omega concerning the interactive graphics facility for the main computer. By early 1971 it had become clear that a mini-computer connected via a data link to the CII 10070 could control a set of independent display consoles better than any special purpose display controller available on the market. Since this would be cheaper, more flexible, and offered much greater potential for future modification or expansion than a fixed controller, it was decided to purchase a mini-computer for the Omega graphics facility. Since a mini-computer offered such advantages for a graphics facility, it might also be useful for the on-line experiments during set up and testing, and this was the line of thought behind the next suggestion, shown in Figure 7.

This approach required that the system be expanded by the purchase of three mini-computers, one to drive the interactive displays, one for the test user and one for the set up user. One of the on-line machines was to be configured with 12K memory, paper tape input, no disc, and no FORTRAN capability, but connected to the CII 10070 via a data link to make up for its lack of power. The second on-line machine was to be more powerful, having 16K core, disc, and FORTRAN capabilities, so that it could be used independently of any other computer for advanced testing. The production user would still
remain on the EMR 6130. Each machine would be equipped with a CAMAC interface for the triggering system, which would include a Tektronix 611 storage tube display system. In addition, the EMR CAMAC would be connected to the plumbicon readout system and the beam parameter counters.

As described, this expansion suffered from two difficulties: the lack of a suitable FORTRAN on any existing small machine in the price range, and the difficulty of switching both hardware and software from the new machines to the EMR when going into production. A quick survey of existing small machines indicated that FORTRAN was just not an appropriate language for programming such computers. The compilers were usually slow, difficult to use (i.e. had poor diagnostics), and accepted only a rudimentary subset of the standard FORTRAN language. The object programs were usually large and slow, and quickly exceeded the capacity of the small computer's hardware. This meant that programming for set up and testing would have to be done in the assembly language, a tedious and error-prone process. Furthermore, everything would have to be reprogrammed for the EMR 6130 in order to go into production, a wasteful duplication of effort made more difficult by the scarcity of development time on the EMR 6130 due to its use in production data acquisition. A rapid and easy transfer from testing into production was further hampered by the necessity of reconnecting the on-line equipment to the EMR during the switch.

H. The 4 System

The last suggestion mentioned above was a step in the right direction, and after further discussions a concrete proposal was presented in April 1971 for the system shown in Figure 8. This consists of three identical on-line mini-computers without FORTRAN or disc, arranged so that any one of them could be linked to the EMR for production runs, while the other two could be linked simultaneously to the CII 10070 for set up and testing. These three mini-computers
would be the same type as the interactive display computer, in order to avoid introducing a fourth type of machine into the already complex system. Consideration of the requirements for the on-line configuration (8K memory, paper tape I/O, CAMAC interface, link to CII 10070) and the graphics facility (8K memory, disc, paper tape I/O, link to CII 10070) led to the choice of a PDP-11 as the Omega mini-computer.

This system seemed to resolve all the problems mentioned earlier. Since each test user had his own computer, with an independent CAMAC interface to his electronics, interference with other test users as well as with the production user was virtually eliminated. The transition from test stage into production was now considerably simplified since during production the triggering system and the readout programs would remain on the PDP-11 where they were tested and debugged. Only the plumbicon and beam parameter readout system was connected directly to the EMR, and since this facility was expected to change very little from experiment to experiment, a general program for the EMR could be written to read the chamber and beam data, merge it with the trigger data received from the PDP-11 that was connected at the time, and record it onto tape. Thus the experiment-independent tasks of acquiring chamber data and recording it onto tape are clearly separated from the experiment-dependent tasks of acquiring and checking trigger data.

The big advantage of this system is its symmetry, a factor that took on added significance at this stage in the development of Omega when it was not known which of the first three scheduled experiments would be ready for production data taking first. Therefore it was essential that all users have the same set of facilities available to them so that no one group would have an advantage over the others due to the fact that its equipment was attached to the EMR for testing and the others' was not, or because its test programs were written for the EMR and the others' were not.
The problem with this proposal was that it was too ambitious to have completed in its entirety by the fall of 1971, when first testing of on-line equipment was scheduled to begin. There were several aspects to this problem. First, there were three on-line PDP-ll computers that had not been anticipated a few months before, so that all the software for these machines had to be designed and implemented. To further complicate matters, the machines themselves would not arrive at CERN until mid-summer, and the CAMAC interface, to be built by the manufacturer, was not expected before fall.

Second, the number of links in the system had grown from one between a single EMR 6130 and the CII 10070, to six between the CII, two EMR's and four PDP-ll's. All the link hardware and software was being designed and built at CERN, with the first link (between the EMR and the CII) expected to be ready for testing in August, 1971. Now a new link interface to the PDP-ll had to be designed, new software written for the PDP-ll, and the entire link scheme re-evaluated for consistency, since it was considered highly desirable that all links be as nearly identical as possible in hardware and software. The task of producing six sets of link hardware in such a short time greatly exceeded the capacity of the CERN electronics shops, which after all are not set up for assembly-line production of such devices.

Third, initial program development for the mini-computer would be extremely difficult without the links to the CII 10070 or some alternative. In the final system programs to set up and test trigger systems could be largely written in FORTRAN and run on the CII 10070, so that only a minimal amount of assembly language coding would be required on the mini-computer. Without the links, everything had to be done on the mini's, and since they were to have only 8K memory and no disc, they could not support a FORTRAN compiler. Therefore all the initial programming would have to be done in assembly language unless a compiler for a better language could be written to run on the CII 10070 and produce code for the mini-computer.
Finally, the enlarged scope of this system required considerably more manpower than originally estimated, and competent people cannot be obtained and integrated into a development group overnight.

I. The 5 System

Because of the preceding considerations, it was decided by July that, although the 4 system was highly desirable and should be implemented as a long term objective, a simplified, interim system should be developed for the initial period of Omega testing and production. The system is shown in Figure 9. It consists of all the computers of the large system, but with links only between the CII and the two EMR 6130’s (the links specified a year earlier), and between the CII and the display PDP-11. Since all trigger data is read out by the PDP-11’s, but must be merged with chamber data and recorded onto tape by the EMR, a uni-directional CAMAC to CAMAC interface has been added between the EMR 6130 and the PDP-11 of the production user. This interface is simply a buffer into which the PDP-11 deposits the trigger data it has acquired from the experiment’s on-line equipment. The buffer is then emptied by the EMR 6130 after all the chamber data and beam parameters have been read out from the EMR’s CAMAC. Other than this, no communication between the two machines is possible, and there is no way for the PDP-11 to delay the EMR 6130. If the PDP-11 does not fill the buffer in time, an error condition is detected on the EMR, a message is printed to the operator, and the event is abandoned. Such a scheme is far from fail-safe, but offered the advantage of requiring almost no changes to the existing EMR data acquisition software, of creating no danger of destroying the chamber data or inhibiting the tape recording process, and of being an extremely simple method of interfacing the two computers within a short time.

With this configuration, all the programs for set up, testing, and production use of the trigger systems must be run on
the PDP-11, and since a FORTRAN language is not available on such small configurations, it became obvious that some sort of programming assistance would have to be given to the users. Therefore a library of standard procedures and an on-line user's monitor were specified and developed to handle the more routine data acquisition functions, and to provide drivers for the Tektronix 611 displays and the standard I/O peripherals. As a more direct aid to the programming difficulties of the user it was suggested that a FORTRAN compiler be written to run on the CII 10070 but to produce object code for execution on the PDP-11. This met several objections: first, that the compiler probably could not be completed before it was needed by test users in the fall, even if only a subset of FORTRAN were implemented; and second, that FORTRAN was not an appropriate language for programming the PDP-11, especially for on-line, real-time applications. Instead it was proposed to design and implement an intermediate-level language, called PL-11, which would combine the syntactic form and programming style of a high-level, ALGOL-like language with the run-time efficiency and closeness to the machine hardware of the assembler. Such a tool would make it possible to easily program the PDP-11's without a data link to the CII and without a FORTRAN compiler for the mini-computer.

J. The Past Year

The idea of implementing the 5 system as an interim version of the 4 system seems to have satisfied all the necessary requirements for Omega, and since its final approval in August 1971 no major revisions have occurred. The past year has been spent implementing the various aspects of the proposal.

The first PDP-11 and the Omega EMR 6130 were delivered in July 1971, the second PDP-11 in August. During that summer tests were begun on the plumbicon system connected via CAMAC to the EMR 6130. In the fall, work was begun on the software for the PDP-11/CAMAC interface, including a set of drivers for the 611 storage tube displays, and on a set of FORTRAN-callable histogramming routines
that would comprise a compatible histogramming package on all three
computers. By this time the first version of the PDP-11 on-line
monitor, which was begun in the Spring, became operational, and by
November the first version of the PL-11 compiler was available on
the CII 10070 for general use.

The CAMAC interface for the PDP-11, the first of its kind
built by the manufacturer, arrived in December, but it took several
months of debugging before it finally worked properly. Testing of
the first data link between the CII 10070 and the EMR 6130 began
in December, and was completed by March, when testing of the link
between the CII and the display PDP-11 began. By this time the
last two PDP-11's had arrived, and all were in use by experimental
groups for equipment set up and testing. By April the link between
the CII and PDP was reliable enough to permit use of the interactive
display terminals by the physicists. During this period a great
deal of development effort also went into the applications programs
on all three machines, particularly 'ROMEO' (Reconstruction of
OMega Events Off-line), which is a general program written entirely
in FORTRAN on the CII to perform pattern recognition and geometric
reconstruction of a large class of events expected in Omega.

In June the first beam appeared briefly in the Omega
facility and has been available for three and four-day periods every
several weeks since then. These periods have been utilized at one
time or another by all three experiments, both for testing and for
data taking with the PDP-EMR system. Although the CII is not on-line
to the acquisition hardware directly, it is still essential to the
on-line operations, since development and correction of all PDP-11
programs must be done with the PL-11 compiler on the CII 10070, and
data recorded onto tape by the EMR is analysed on the CII during the
run in order to get feedback on the quality of the data. In addition,
the direct link to the EMR has been used by one group in a preliminary
manner to transfer sample events to the CII where they were processed
by the pattern recognition programs and the results returned via the
link to the EMR for printing. By the end of September geometric reconstruction and simple kinematics programs will also be utilized on the CII in order to allow useful physics histograms to be accumulated in real-time on the sample of events that are selected and sent over the link from the user program in the EMR.

K. The Future

Now that data taking has begun, a great deal of development must still be done on the current system to bring it up to the specifications of the full system. Although tested and working, the data links that do exist have not really reached their full potential and are as yet little used. Work is already underway to integrate the CII/EMR link into a sophisticated sample analysis system with elaborate histogramming, display, and control facilities, and to integrate the CII/PDP link into a powerful debugging tool for use during set up and testing.

On the CII 10070 a roll in/roll out system is being developed to allow many real-time users to have access to the power of that machine without degrading its performance. At the same time a file handling subsystem is being implemented for use with the data links to give the on-line computers remote access to the full range of the CII facilities. On the EMR 6130, the first version of the data acquisition and sample analysis system is being upgraded to make it more flexible. This requires elimination of the many deficiencies in the manufacturer's software system, which lacks a good overlay facility, tape labelling, and user libraries.

L. An Evaluation

That completes the history of the Omega data handling system from its inception in 1968 to the present. Except for the past two months, it is a history of change and development, with no 'production' physics. The major revisions required in the original specifications demonstrate how carefully the 'development' phase of
such a large system must be considered, and prove the absolute
necessity of the computer as a development tool for on-line physics
experiments: it is as essential for setting up and testing equipment
as an oscilloscope. Our experience has shown that designers of
future on-line data handling systems must consider these two develop-
ment functions in addition to the production functions of their
systems if they are to avoid major revisions at a later stage. The
items listed in Table 2 comprise the functional aspects which have
evolved in Omega, and the most important consideration in the final
system is its method of handling all of these functions simultaneously
for a set of independent experiments.

The Omega data handling system was operational when the
first beam appeared, a remarkable achievement in view of its
complexities and the revisions which occurred. Furthermore, three
months of operations have demonstrated that it is a useful system,
although the rigorous test will not come until January 1973, when
construction of the Omega magnet will be completed and full production
data taking is scheduled to begin. This success was due to many
factors. Perhaps most important were the basic design principles of
generality and flexibility, but it must not be overlooked that all
the Omega decisions were tempered by a realistic appraisal of existing
options and constraints. The idea of starting with simple yet adequate
interim solutions instead of attempting the full final system straight-
off not only ensured successful attainment of a working system, it
also made the many redesigns easier to absorb. A simple first step
is often the most flexible method of finding out where the real design
problems lie without endangering the success of the project. Once the
simple base is made, more advanced stages are easily built upon it.

The original report anticipated frequent change in the
definition of Omega, and experience has borne this out. Although there
were defects in the original proposals they were detected and corrected
at an early stage, indicating perhaps that flexibility is more impor-
tant than having correct initial designs. In a broader sense the
the evolution of Omega reflects a growing trend toward multiple computer systems, and in particular toward a system in which large numbers of mini-computers are used for all interfacing to the external world. The result is a powerful, usable on-line data handling facility.
III. TECHNICAL ASPECTS OF THE OMEGA DATA HANDLING SYSTEM

A. Introduction

The most obvious aspect of the history of the Omega data handling system is its evolution into an integrated hierarchy of computers: one medium-sized, two small, and four mini. As discussed in Part Two, this system appears to offer the best solution to many problems that confront the designers of an on-line data acquisition system.

There are many interesting technical problems involved in designing and implementing such a complex system, and this part of the paper will present some of these in detail. We feel that many of the features of Omega have general application and should be carefully considered in the design of future systems.

B. The Data Links

1. The concept

The most obvious first topic for discussion is the data links that join the individual computers. As shown in figure 8, the final system contains six links, and the interim system shown in figure 9 contains three. All the hardware and software for these links, and the interfaces to all three types of Omega computers, were designed and built at CERN. Since these links form such an important part of the data handling system, a great deal of thought and effort went into their design, implementation, and testing. A particularly important goal was to ensure that the operating system would be safe from errors in the users' FORTRAN programs, errors due to hardware failure, and errors in the computer at the other end of the link (such as its not being connected), and that any conflicts that might develop during the dialogue between machines would be automatically resolved. In addition, the link software was to be as close as possible to the basic functions of the link hardware in order to give a maximum of flexibility.
It would then be possible to build more complex systems, such as a file handling system, from the basic link primitives.

For purposes of flexibility, the link software was designed to permit any number of programs in one computer to communicate with any number of programs in the other computer over the same link. It was also planned to allow a program in one computer to communicate over different data links to several other computers, but because of implementation difficulties this is now permitted only on the CII. Finally, it was considered important that the user interface be as nearly identical as possible on all computers. This would make it easier for the user, since only one set of software routines need be learned, and also preserves the basic symmetry of the link hardware as seen from either computer. It does however require greater care in dealing with error conditions and transmission conflicts, such as occur when the two computers both wish to initiate a communication at the same time.

2. The user interface

The FORTRAN user/data link interface for the CII and EMR is shown in table 3, along with the equivalent PDP-11 interface. The reason for the differences in the parameters on the two machines is a simple but important consideration: multi-programming. On the CII, the general multiprogramming facilities provided by the operating system can be used to full advantage by the link system, whereas on the EMR no such facilities are available. The CII operating system is organized to handle multiple tasks, with all the necessary mechanisms to handle task activation, task blocking and task switching. It is therefore easy to have many simultaneous, independent link users. On the mono-programming system of the EMR, however, the concept of tasks and task switching does not exist, so that it is very difficult to share control of the processor between several users. To do so would essentially require the operating system to be rewritten for multi-programming. Therefore, although
the user routines are functionally equivalent on both computers, the actual details of their implementation and use must necessarily be different. Only the more general CII interface will be described here.

A program wishing to use a link must first call "connect link" (CNCTLK) in order to make itself known to the link interface. This sets up a buffer to receive transmissions over this link to this program, and a "call-cell" that will be used to identify the program at the other end of this link that is currently communicating with the program. A program can be simultaneously connected to any number of links, but at any time it can receive transmissions from only one program per link (i.e., the one identified in the call-cell). The "disconnect link" (DNTLK) routine is called to terminate a program's use of the link, thereby releasing the buffer space and the call-cell. Typically a user program will call CNCTLK when it first begins execution and DNTLK just before it terminates execution. Conceptually, CNCTLK is equivalent to the standard "open file" operation and DNTLK to the "close file" operation.

In order to initiate a communication with some program B in the other computer, the user program A must first call "write link" (WRITLK) with parameters to indicate the location and amount of data, and the name of the program (i.e., B) in the other computer. The receiving program's identification (B) is placed into program A's call-cell for this link, thereby establishing B as A's current partner on this link and ensuring that the only future transmissions to program A that will be "accepted" on this link by the interface will be those sent by program B. Similarly, this transmission will be refused by the interface in the receiving computer unless program B's call-cell for this link is either clear or contains program A's identification (as set by a previous transmission between A and B).
The data sent to program B can be interpreted either as actual data or as status information from which it can determine the nature of future communications desired by program A (such as "send more data", etc.). If program A expects a reply from program B, it must call "read link" (READLK) to inform the interface that it is waiting for a transmission on a particular link. Program A is put into the waiting state until a transmission to program A is received from program B, at which point the data area specified in the call is filled with the data received over the link and program A is reactivated. The data received can only be from program B (since B is the program name in A's call-cell as the result of the previous WRITLK), and should be the data requested by the previous WRITLK from program A. However, it is the task of communication partners to ensure this, which can be easily done by keeping user status information in the first few words of each block transmitted. The "read multiple" (RDMULK) operation is used identically to the READLK except that a transmission to program A over any link will be accepted by RDMULK.

The interface routine "idle link" (IDLELK) is called by program A when it has finished all the communication it expects over the links it is connected to, and is prepared to wait indefinitely until a new transmission is initiated from an unspecified program in another computer. The interface clears all the call-cells for program A (one for each link to which A is connected), and then enters program A into the waiting state with no specified time limit. On the CII, a future system will make it possible for programs in this state to be rolled out onto the disk, so that they will not occupy valuable memory space. Such a facility is required for general purpose file handling programs in the CII which may be requested at any time by programs in the remote computer and therefore must always be available for use, although they clearly should not occupy memory between uses. When a transmission to program A is received from some program C in another computer,
the system reactivates A, rolling it back into memory if necessary, and establishes C as A's communications partner on the appropriate link by putting the name C into A's call-cell for that link. At this point, program A must call READLK (or RDMULK) in order to receive the data sent from the other computer.

3. The link dialogue

The interface just described obviously requires a software package to perform the link transmissions, although the link users are never aware of the details of the actual dialogue.

Each link communication consists of three parts, as shown in figure 10. The first phase is simply a "wake-up" exchange of interrupts that inform the initiating computer whether or not the receiving computer is "alive", and that indicate to the responding computer that it is wanted in a communication by some other computer.

The second phase consists of a write by the initiating computer and a read by the responding computer in order to exchange a standard block of status information between the two software interfaces. This information includes the program identification of the sending and receiving programs, and the number of bytes to be transmitted. The end of block interrupt also includes the status of the transmission as seen by the initiating computer. The receiving computer must digest this information to determine whether or not it can accept the transmission. There are several possibilities: (1) the requested program is connected and the link buffer is empty, ok to proceed; (2) the program is connected but the link buffer is full, try again later; (3) the program is not connected, illegal operation; (4) data block too big to handle, illegal operation; (5) error on transmission of the status block, try again. This response is put into a status register by the requesting computer and a return interrupt is sent to the initiating computer, which
then reads the status and decides what to do next.

If the responding computer has indicated that it can accept the data, the initiating computer will then issue a write and the responding computer a read in order to transmit the user's data block. Again the end of block interrupt is sent from the initiating computer along with information on the status of its transmission. The responding computer will read this status and then send back an interrupt to the initiating computer along with status information to indicate whether or not the entire transmission was successful, and if not, whether to try again or break off.

The software has been designed so that the initiating computer will only write, never read, and the receiving computer will always attempt to read more bytes of data than it knows the initiating computer will be writing. In this manner, it is guaranteed that the write will always terminate and the read will never terminate unless there is a hardware failure. This permits the initiating computer to receive status information from the link hardware on the quality of the transmission (parity errors, etc.) and send this, along with instructions on what to do next, to the responding computer as part of the end of block interrupt. When the responding computer receives the end of block interrupt, it knows there are no more data words to be transmitted, and can terminate the "hanging" read. In this manner both computers are kept completely synchronized, and the state of each machine is well known at all times to the communications partner. This ensures an orderly dialogue, especially in the case of conflicts, errors, or delays caused by lack of buffer space.

While this dialogue may appear a bit complicated, it does enable the software interface to easily detect conflicts and all errors, thus guaranteeing the integrity of the link in all communications and removing this essential task from the user. It also has been designed to fit conveniently into the CII multiprogramming system, which can be used to buffer all transmissions received over
the links and to wake up the receiving program after the sequence has been completed. If necessary, the receiving program will be first rolled in off the disk. The interface design has made testing of both hardware and software quite straightforward, due to the frequent exchanges of status and the close synchronization of the two computers by means of interrupts. This should also simplify detection of hardware failures during normal operations.

4. The link hardware

Finally, a few words about the link hardware. The links are full duplex serial transmission channels capable of handling rates up to one million 8-bit characters per second. Each interface contains a small buffer for both data and status, so that high data rates can be maintained without loss of data when receiving a transmission over a 1600-meter cable. In practice channel operation is limited to 400,000 bytes/second, the maximum data rate that can be handled by the CII. The CII 10070 interface is in the form of a device controller attached to one of the standard multiplexed I/O processors (MIOP's). The EMR interface is connected to the standard telemetry data channel (TDC) which is capable of handling data rates of 1,320,000 bytes/second in units of 16-bit words.

The PDP-11 interface has been built to accept from the unibus commands to initialize internal registers for a direct memory access transfer of a block of data. This approach eliminates the programmed word by word transfer that is characteristic of most I/O on the PDP, thereby allowing the transmission to proceed independently of the central processor and at rates approaching the memory access speed.

An important design principle was to make the data link as symmetric as possible. This not only made the system conceptually cleaner and easier to program, it also made hardware debugging
simpler and enabled the designers to build a sophisticated remote
end simulator that could be used on-line and off-line to test link
drivers. In addition, rather simple simulators of the CII MIOP and
the EMR TDC were also constructed for preliminary testing purposes.

C. Omega Graphics

It is clear from the historical sequence of proposed Omega
configurations that CRT displays have always been a prominent component,
as they should be in any modern data handling system. In the current
Omega system, each computer is connected to at least one CRT display.

Each of the three on-line PDP-11's is connected via CAMAC
to a Tektronix 611 storage tube display. By making calls to a set of
software routines provided in the Omega PDP-11 on-line monitor (see
table 4), the programmer can conveniently perform the basic operations
of point plotting, character generation, vector drawing, and erasing
the storage screen.

Each EMR 6130 is interfaced directly to a T4002 storage
tube display. While not as elaborate as the facility provided to CII
10070 users, this device forms an integral part of the sample analysis
system on the EMR, and is heavily used in all phases of EMR operations.
Associated with each of the on-line PDP-11's, but not connected to
the PDP-11, is a second Tektronix 611 display which duplicates iden-
tically the picture displayed on the T4002 connected to the Omega EMR.
This provides the PDP-11 user with an inexpensive method of monitoring
the spark chambers and beam detectors, which are connected only to
the EMR 6130, since no hardware link is necessary between the two
computers.

A single large refreshed display with a 4K memory buffer
of its own is now being tested on a PDP-11 for installation in the
plumbicon CAMAC system of the EMR. This display will provide a con-
tinually updated picture of the spark coordinates superimposed on
the current chamber set up independently of any computer. This is
invaluable for testing and monitoring the performance of the chambers, the triggering systems, and the plumbicon cameras.

The fourth PDP-11 is devoted exclusively to graphics, and is connected through a special interface to several interactive display consoles, each having a Tektronix T4002 storage tube display with keyboard input, scratch-pad memory, and tracker-ball. Currently two display consoles have been installed, with two more expected in the near future. This system has been designed so that a user at one of the display consoles can interact directly with a program executing on the CII 10070. The PDP-11 is essentially "transparent" to him, since all his programming for the displays is done in FORTRAN on the CII 10070 using an interactive graphics package called GD3. This package was originally developed for the CERN central computer installation, but has been rewritten for the CII 10070 so that to the FORTRAN user it looks identical to the original GD3. This enables FORTRAN programs that utilize graphics to be transported easily between the Omega machine and the central computers, an important requirement for efficient development and production use of the pattern recognition and geometry programs.

The GD3 subroutines on the CII 10070 are written as if they were connected to a standard display controller, but instead of being routed via a data channel to a hardware device, the display commands generated by GD3 are sent via the data link to programs in the display PDP-11, where software tasks interpret the commands and generate the I/O instructions necessary to drive the T4002 displays.

The decision to drive the display consoles with a minicomputer and link system rather than a hardware display controller was made for several important reasons. Clearly a minicomputer is far more flexible than any conceivable hardwired controller, offering virtually unlimited potential for modification and expansion of the display facility. Of course, a computer can be used for other functions when not in use by the display facility. By performing the
actual picture generation and keyboard input on the PDP-11, a faster response can be given to user button pushes than seemed possible with software on the CII running in a multi-programming environment. Furthermore, a display controller connected directly to the CII would have been a new type of device for the CII software to handle, with all the attendant development effort, but since the link system was being installed for the EMR's and the on-line PDP-11's, no additional development was required in the CII, and in fact the display system provided a convenient vehicle for testing the link hardware itself. As if these factors were not sufficient, the minicomputer approach was also cheaper than any device controller of comparable power.

Although the display PDP-11 functions only as a very "smart" device controller to users of GD3, this machine has been provided with basic but fairly general multi-programming functions in order to facilitate the programming and debugging of the display package, and to provide a system that can be easily expanded, not only with more T4002 displays, but possibly also with other I/O units, such as teletypes or other types of display consoles. This system provides general mechanisms for reserving and releasing resources, for handling task switching, for managing memory buffers, and for queuing disc and link transfers. The multi-programming approach also made implementation of the link interface simpler, since the existing CII link software could be used as a working model.

D. Histogramming Facilities

One of the most heavily used features of the Omega software system is the histogramming package. Because graphic feedback is essential during all phases of on-line work, an on-line data handling system must provide comprehensive histogramming facilities that can be utilized easily by the programmers and by the on-line operators.
In Omega, the histogram package has been integrated with the display system and link systems in a very powerful manner. It can be used either on-line or off-line, and once again the design was guided by the principles of generality and flexibility coupled with the desire to provide compatible facilities on all three types of computers. Clearly it would be impossible to attempt to provide on the PDP-11 all the features that can be provided on the CII 10070. However, the same basic kernel is found in all three machines, and to the extent it could be implemented easily, the use of this kernel and the results it produces are identical on all machines. Table 5 lists the one-dimensional histogram routines which were available in the first version of the Omega software system.

In order to define a histogram the user program must first call HBOOK1. This is done typically at the start of an experimental run. The system will then reserve the necessary bin space, reset it to zero, and set up the appropriate information for accumulating the data and making it available for output. Each time a value is computed that must be entered into the histogram, HFILL1 is called. This is done typically once per event for each histogram being accumulated. The system will scale the value and update the corresponding bin by one. Underflow and overflow values are also detected. At any time, HSTDOL can be called to print the histogram or display it on the CRT, depending on the parameters. A specified histogram can be reset to zero by calling HZERO, and the routine HFREE cancels a histogram, thereby freeing the space reserved for it. HMENU produces a list of all the histograms that have been booked and not yet cancelled.

Two-dimensional equivalents of HBOOK1, HFILL1 and HSTDOL are also available to the user. In order to conserve buffer space it is possible for the user to define for two-dimensional histograms the maximum number of bits which should be used to hold the bin values. Although this requires considerably more work in the histogram package to update and display a histogram, it makes possible
the accumulation of fairly large two-dimensional histograms on a mini-computer. It is also possible to obtain a display of the two-dimensional histograms in either tabular or scatter-plot form.

In all computers histograms are represented in the same internal format in order to make it easy to accumulate a histogram in one computer, then send it over the link for display or printing on another computer. In the near future additional facilities will be provided to allow the user to have on-line control of the histogramming package without the need to recompile his program. System commands will enable the user to display, define and cancel histograms from the teletype while his program is running.

E. Consequences of the Multi-Computer Hierarchy

An important consideration in the Omega data handling system is the allocation of various tasks to the component computers. In many respects the hierarchy of computers in Omega represents an ideal situation; a basic model which future on-line systems would do well to follow. This is especially true of the relationship between the CII and PDP-11's. The CII is a fairly large machine, with an advanced multiprogramming system and a full range of file handling facilities. It was therefore decided that this machine should be used for all PDP-11 program development. The PDP-11 is designed to handle a wide variety of non-standard I/O devices in a very convenient manner. Therefore, all on-line equipment, including displays, should be attached to this machine. (Equipment is also attached to the EMR, but as a permanent experiment-independent facility that changes little, if ever, over time).

These were important decisions, and experience has shown them to be correct. First of all, since no program development is to be done on the PDP-11, a minimum configuration suited solely to the on-line requirements is perfectly adequate. Such a configuration does not include card reader, line printer or disc, all of which
are essential on a usable program development machine. 8K of memory, teletype, and high-speed paper tape reader plus display units comprise a system that is perfectly adequate for setting up and testing on-line equipment, as well as for providing an on-line monitor of the trigger system during production data taking. These systems are very cheap, which implies that there can be many of them. In Omega, each experimental group has its own PDP-11, which means that each group can be completely independent of the other groups while setting up and debugging its trigger system. This not only increases the efficiency of the testing effort, since all problems with mutual interference (at the hardware and personnel level) are eliminated, but also provides each group with equal facilities and access to those facilities that is scheduled according to the particular needs of the individual group. (That is, one group is not forced to come in at 2 a.m. in order to be able to use the computer to test its hardware).

Concentration of the mini-computer program development on a large central machine clearly provides the mini-computer programmer with a better software development tool than could be provided with a set of identical small or medium-sized computers. The large computer has bigger and faster discs, faster line printers and card readers, more tape units and more memory than could possibly be provided at comparable cost for a set of independent smaller machines. These features plus an advanced multiprogramming operating system make the CII a powerful development tool for many simultaneous users. Furthermore, the data processing nature of program development, such as file editing, compiling and link-editing, is very different from the nature of tasks that test on-line equipment, which implies that interference between the simultaneous users of the central machine can be essentially eliminated by a good operating system, a task that is virtually impossible on the on-line computers.

The software development system of the large machine is not only better but can also be achieved with less cost than it could be for a set of smaller machines. This is due largely to the fact
that a properly chosen large machine will be delivered from the
manufacturer with a large complement of working software. Thus
many of the development tools already exist, and these can be read-
ily used to develop additional development tools in a boot-strapping
process. In the Omega system, a set of macro-instructions within the
CII METASYM assembler produced a PAL-ll "assembler" for the CII in
a matter of weeks, and the PL-ll compiler for the CII was written in
FORTRAN in a matter of months. Both of these compilers produce relo-
catable object modules in the CII object language format, which
implies that all the library and link-editing facilities of the
existing CII operating system were immediately available to the PDP-ll
programmer "for free". Any improvements in the CII software would
automatically be improvements for the PDP-ll users as well, thereby
eliminating any wasteful duplication of effort by systems programmers
to develop similar facilities on different machines. (One should not
overlook the amount of effort necessary to develop adequate data
processing facilities, even in a physics laboratory with an ostensibly
complete software system provided by the manufacturer).

The preceding considerations are valid whether or not the
two computers are physically connected. They need only be in physical
proximity to one another to be effective. However, once a reliable
data link does exist between the machines, the whole becomes truly
more than the sum of its parts. The central machine has gained a
very powerful yet relatively cheap peripheral, and the mini-machine
is suddenly endowed with direct access to processing power far beyond
its normal capacity.

Consider first the role of the PDP-ll as a CII peripheral.
Its importance as a controller, tester and read-out box for the on-
line trigger systems has already been discussed and is beyond question.
We have also mentioned the importance of the mini-computer in the
interactive graphics facility. The use of a mini-computer to drive
a set of displays gave more flexibility and better user response at
lower cost than was possible with any of the special purpose display
controllers considered. And of course, the mini-computer can be used for other tasks when the displays are not in use.

But perhaps the most interesting aspect of the mini-computer as a peripheral to the large machine is one that is now being developed in Omega: the mini-computer as a remote I/O station for the large machine. In particular, a system is being designed for Omega whereby a user at the PDP-11 teletype will be able to manipulate and edit files on the CII disc, submit jobs for execution on the CII and receive the results at his PDP-11 terminal. This benefits both the PDP-11 users and the CII users simultaneously.

From the point of view of the PDP-11, a link to the CII has many attractions. As mentioned in the last point above, direct access text-editing and file manipulation facilities on the CII will greatly enhance the already existing PDP-11 program development features on that machine (PL-11 compiler, link-editor, etc.). In addition, direct loading of the PDP-11 memory from the CII disc eliminates the need for paper tape to transfer programs, not only reducing the time the PDP-11 is idle during program modifications, but also enabling the PDP-11 to be located at a considerably greater distance from the CII than would be possible in the current setup. The link of course allows any PDP-11 program to communicate directly with CII programs, thereby making it possible to write large, sophisticated analysis programs in FORTRAN for the CII that obtain raw data directly from the PDP and return the results for display on the PDP. This may be of great use in the advanced testing stages of complex trigger systems. In addition, programs executing on the PDP-11 can be overlayed from the CII disc via the link, eliminating the need for a disc on the PDP-11 to perform this function. The discs on the CII are already being used for storage of PDP-11 program libraries, but the links now make it possible for programs in separate PDP-11's to communicate dynamically via the CII discs. The possibilities seem to be endless.
F. The Mini-Computers

1. Introduction

The final topic for discussion is the on-line mini-computers themselves. There are two aspects of this topic to be considered: real-time monitors and programming languages. The fact that the Omega on-line mini-computers have a very small configuration influenced both of these topics to a large degree.

2. The on-line monitor

The Omega PDP-11 on-line monitor is not a true operating system as that term is commonly understood. On such a small computer, it would be impossible to even consider a complete operating system. Rather, the monitor consists of a collection of modules that include I/O drivers for the teletype, paper tape reader and CAMAC interface, commonly used conversion routines, routines for using the Tektronix 611 display (see table 4), an interrupt handler, an on-line controller, and a teletype command interpreter. This interpreter accepts simple commands for loading absolute binary paper tapes, for starting and stopping program execution, for simulating event and burst interrupts, and for interrogating and/or modifying any register or memory cell dynamically. The monitor is simple but provides a basic tool which can be easily used and understood by a wide range of users.

The on-line controller provides a primitive foreground/background type of operation. It requires the user program to be structured into three parts: an initialization sequence (main program), an event procedure, and a general procedure. The initialization sequence is activated once, when the execute command is entered on the teletype, typically at the beginning of a run. This sequence sets up data values, histograms, CAMAC addresses, etc. For the run, informs the on-line controller of the location of the event and general procedures, and then relinquishes control
to the on-line controller. The controller can respond to an event trigger in one of two ways: it can call a system routine to read the counter values into memory using a table of CAMAC addresses set up by the initialization sequence, then write them into the CAMAC buffer between the PDP-11 and the EMR 6130; or it can transfer control directly to the user's event procedure where the data is read under programmer control, and any other type of calculation performed. Typically, this involves the histogramming of experimental parameters such as counter values, hodoscope patterns, etc. The user's general procedure can be activated when a system flag bit is set by the event procedure, at regular intervals by using the monitor clock, or when the operator issues a teletype request to execute it. Status information in a monitor communication cell permits the user to determine what caused his routine to be activated and to take appropriate action. Typically this action will consist of displaying a histogram or set of histograms, typing out parameter values, changing parameter settings, etc. Each user is free to include as many general features in this procedure as he wishes.

3. Programming languages

At the time the idea of purchasing mini-computers for each experiment was being considered, a very important part of this discussion concerned the problem of what language would be used to program these machines. Although the debate centered on the choice between FORTRAN, a "High-Level" language that is widely used on many computers, and the PDP-11 assembly language, a "Low-Level" language, it was resolved by the introduction and acceptance of a third alternative, PL-11, an "Intermediate-Level" machine-oriented language. This language was designed and implemented by Omega with the intention of developing a better programming tool for PDP-11 computers than either FORTRAN or assembly language. The idea behind PL-11 is simply to keep the high-level
syntax of a FORTRAN-like language, thereby making it easy to learn and program, but to design the language constructs with the architecture and organization of the PDP-11 in mind, thereby making it efficient and close to the PDP-11 hardware.

In order to accomplish this goal, one must consider the advantages and disadvantages of high-level languages (see table 6) and machine-level languages (see table 7). The advantages of high-level languages all derive from the fact that they are "people-structured"; that is, constructs in the language, such as arithmetic expressions, iteration statements, conditional expressions, etc, correspond to the way people think about a particular algorithm. In some sense these represent a natural notation for expressing formal algorithms. Although it can hardly be argued that people think in terms of array declarations and read and write commands, at least these are expressed in a notation that is easy for people to use and understand. The net effect is that programs written easily and reasonably quickly in a high-level language are also easy for both the original programmer and other people to understand and modify.

The problem with high-level languages is that their constructs may bear little if any relationship to the organization of the machine hardware. Too often computer hardware is designed without regard for how it can be used by people, so that the mapping between high-level language constructs and the machine instructions is very bad. The result is that object programs are both large and slow, and for real-time mini-computers, this is unacceptable. Consequently, assembly languages are too often used in order to gain run-time efficiency at the expense of programmer efficiency. A second major reason for the use of assembly languages is that many hardware "features", especially custom-built on-line equipment, may not be accessible in a high-level language.

There is little doubt that an assembly language is an
extremely primitive tool for programming a computer, even a mini-
computer, because it is "machine structured" rather than "people
structured". It requires enormous attention to petty detail that
is irrelevant to the user's task, so that programming becomes a
tedious and error-prone process that dissuades many people from
even attempting it. Thus a great deal of the power of a computer
is not utilized, simply because it is too difficult for people to
write and understand assembly language programs. Large assembly
language programs are virtually unmodifiable, since even the
original programmer will be loath to attempt a change once he
has the whole thing finally in some sort of working order. To
expect a non-professional programmer to use assembly language is
to expect him not to use the computer.

The goal of PL-11 was therefore to be as "high-level" as
possible without sacrificing run-time efficiency. The result is a
language with a syntax based on ALGOL 60 (see the example in
figure 11). It is free-field, with statements and declarations
separated by semi-colons and comments allowed anywhere. The lan-
guage includes declarations, procedures, and control statements,
such as loops, and branching statements, that are very ALGOL-like
in their syntactic form. Arithmetic expressions are more primitive
than ALGOL expressions, because they must be evaluated in a linear
fashion on the PDP-11, but they do retain the well-known arithmetic
operator notation (+ for add, - for subtract, etc.). All register
and storage allocation is under programmer control, and subscripting
is restricted to constants or constants plus a register, simply
because this is the only type of indexing which can be handled
directly by the hardware. All eight of the PDP-11 addressing modes,
including indirect addressing and stacking operations, are directly
available to the programmer. A heavily used feature of PL-11 is
the ability to give meaningful symbolic names of any length to
almost any object of interest in the computer: fixed memory cells,
compile time constants, elements of an array, registers, I/O
devices, CAMAC registers, etc. This plus the ALGOL-like structure of the language elements encourages a very readable programming style with programs that can be easily understood and modified. In addition, many of the more common assembly language errors are avoided completely due to the discipline imposed by the language structure. This means that programmers can get PL-11 programs running and debugged much faster than equivalent assembly language codes.

Furthermore, PL-11 is efficient (see an example of the code generated in figure 12). All instructions in the computer can be represented by language constructs, so that all hardware features are available to the programmer. More importantly, the code generated by the compiler for each language construct is part of the specification of the language itself, thereby enabling the programmer to be as conscious as he wishes of the number of instructions he generates. However, experience has shown that a readable programming style with good global organization that can be easily understood and modified results in far more efficient programs than local nit-picking tricks can ever hope to attain.

In conclusion, the Omega experience indicates that the best tool for programming a mini-computer is neither FORTRAN nor assembly language, but a machine-oriented intermediate language. This tool can be used by systems programmers and applications programmers alike. PL-11 has been used extensively by all the on-line physics groups with the result that the PDP-11 is programmed easily and efficiently by physicists for a wide range of tasks, many of which might not have been attempted with either FORTRAN or assembly language.
Table 1: Criteria for selecting a computer

<table>
<thead>
<tr>
<th>Benchmark tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compiler - size, speed, reliability</td>
</tr>
<tr>
<td>Object code - size, speed</td>
</tr>
<tr>
<td>Operating system - usability</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Meeting the specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expandability</td>
</tr>
<tr>
<td>Availability of components</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware</td>
</tr>
<tr>
<td>Software</td>
</tr>
<tr>
<td>Documentation</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Maintenance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
</tr>
<tr>
<td>Quality</td>
</tr>
<tr>
<td>Hardware, software, peripherals</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Existence</td>
</tr>
<tr>
<td>Accuracy</td>
</tr>
<tr>
<td>Availability of source code for system software</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Price</th>
</tr>
</thead>
</table>
Table 2: Functions that an on-line physics data handling system must perform simultaneously

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Development of data handling system</td>
</tr>
<tr>
<td></td>
<td>(online hardware and software, offline software)</td>
</tr>
<tr>
<td>2</td>
<td>Development of on-line physics systems</td>
</tr>
<tr>
<td></td>
<td>(hardware and software)</td>
</tr>
<tr>
<td>3</td>
<td>Real-time data acquisition, control, and sample analysis</td>
</tr>
<tr>
<td>4</td>
<td>Data reduction (pattern recognition)</td>
</tr>
<tr>
<td>5</td>
<td>Data analysis (geometry and kinematics)</td>
</tr>
</tbody>
</table>
Table 3: The data-link routines

<table>
<thead>
<tr>
<th>Function</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>CII</td>
<td>(FORTRAN USER LEVEL)</td>
</tr>
<tr>
<td>CNLTLK</td>
<td>(LINKNO, PROGRAMID, NBYTES, KIND)</td>
</tr>
<tr>
<td>DCNTLK</td>
<td>(LINKNO)</td>
</tr>
<tr>
<td>WRITLK</td>
<td>(LINKNO, BUFFER, NBYTES, STATUS, IDOPRECEIVER)</td>
</tr>
<tr>
<td>READLK</td>
<td>(LINKNO, BUFFER, NBYTESEXPECTED, NBYTESRECEIVED, STATUS)</td>
</tr>
<tr>
<td>RDMULK</td>
<td>(LINKNO, BUFFER, NBYTESEXPECTED, NBYTESRECEIVED, STATUS)</td>
</tr>
<tr>
<td>IDLELK</td>
<td></td>
</tr>
<tr>
<td>EMR</td>
<td>(FORTRAN USER LEVEL)</td>
</tr>
<tr>
<td>CNCTLK</td>
<td>(EMRPROGRAMID, CIIPROGRAMID)</td>
</tr>
<tr>
<td>DCNTLK</td>
<td>(EMRPROGRAMID)</td>
</tr>
<tr>
<td>WRITLK</td>
<td>(BUFFER, NBYTESSEND, STATUS)</td>
</tr>
<tr>
<td>READLK</td>
<td>(BUFFER, NBYTESRECEIVED, STATUS)</td>
</tr>
<tr>
<td>PDP</td>
<td></td>
</tr>
<tr>
<td>CNCTLK</td>
<td>(PDPROGRAMID)</td>
</tr>
<tr>
<td>DCNTLK</td>
<td>(PDPROGRAMID)</td>
</tr>
<tr>
<td>WRITLK</td>
<td>(BUFFER, NBYTESSEND, STATUS, IDOPRECEIVER)</td>
</tr>
<tr>
<td>READLK</td>
<td>(BUFFER, NBYTESRECEIVED, STATUS, IDOSENDER)</td>
</tr>
</tbody>
</table>
Table 4: PDP-11 on-line monitor routines for using the Tektronix 611 storage tube display

- DYERAS <erase the storage screen>
- DYPOIN <plot a table of N points>
- DYVECT <plot N vectors from a table of N+1 endpoints>
- DYCHAR <plot a string of N ASCII characters>
- DSREGN <return coordinates of specified region of screen>

Table 5: 1-Dimensional histogram routines on all Omega computers

- HBOOK1 (HISTOGRAMNUMBER, NWORDSINTITLE, TITLE, NBINS, LOWEREDGE; BINWIDTH, STATUS)
  <reserve space for 1-D histogram>
- HFILL1 (HISTOGRAMNUMBER, VALUE)
  <increment bin in 1-D histogram>
- HSTDOL (HISTOGRAMNUMBER, OUTPUTUNIT, POSITION)
  <display 1-D histogram on output unit>
- HZERO (HISTOGRAMNUMBER)
  <reset all bins of this histogram to zero>
- HFREE (HISTOGRAMNUMBER)
  <release space occupied by this histogram>
- HMENU (OUTPUTUNIT)
  <display list of all current histograms>
**Table 6**: Advantages and disadvantages of high level languages

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Easy to learn, use, read, modify</td>
<td>1. Inapplicable</td>
</tr>
<tr>
<td>2. Self-documenting</td>
<td>2. Cannot utilize hardware features</td>
</tr>
<tr>
<td>3. Transportable</td>
<td>3. Inefficient object code</td>
</tr>
<tr>
<td>4. Fast to program and debug</td>
<td>4. Not really transportable</td>
</tr>
<tr>
<td>5. Usable without knowledge of the machine</td>
<td></td>
</tr>
<tr>
<td>6. Less error prone</td>
<td></td>
</tr>
<tr>
<td>7. Designed to encourage good program structure</td>
<td></td>
</tr>
</tbody>
</table>
Table 7: Disadvantages and advantages of machine level languages

<table>
<thead>
<tr>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Require a detailed knowledge of the machine</td>
</tr>
<tr>
<td>2. Prone to &quot;tricks&quot;</td>
</tr>
<tr>
<td>3. Difficult to learn, use, read, modify</td>
</tr>
<tr>
<td>4. Not self-documenting</td>
</tr>
<tr>
<td>5. Not transportable</td>
</tr>
<tr>
<td>6. Tedious to program and debug</td>
</tr>
<tr>
<td>7. Highly error prone</td>
</tr>
<tr>
<td>8. completely unstructured</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Advantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Code produced is one-for-one</td>
</tr>
<tr>
<td>2. All hardware features are usable</td>
</tr>
<tr>
<td>3. Efficiency is a function of the programmer.</td>
</tr>
</tbody>
</table>
Fig. 1 The initial system
Fig. 2 The Omega system
Fig. 3: The CII 10070 configuration
Fig. 4. The EMR 6130 configuration.
Fig. 6 A possible expansion of the 3 system
Fig. 7  A second possible expansion of the 3 system
Fig. 8  The 4 system
Fig. 10  The data link dialogue
PROCEDURE MAGIC SQUARE;
BEGIN
COMMENT ALGORITHM 118 COMM. ACM 5 (AUG, 1962) - MAGIC SQUARES;
INTEGER NSQR, N SYN R0, I SYN R1, J SYN R2, IJ SYN R3, K SYN R4;

N => J => I + 1 SHA;
FOR K FROM 1 STEP 1 UPTO N => NSQR*N DO
BEGIN
I => IJ SHA 4 + J SLA;
IF X(IJ) /= 0 THEN
BEGIN
 IF I=1 < 1 THEN  I += N;
 IF J=2 < 1 THEN  J += N;
 I => IJ SHA 4 + J SLA;
 END;
 K => X(IJ);
 IF I+1 > N THEN  I += N;
 IF J+1 > N THEN  J += N;
 END;
END;

Fig. 11 A sample PL-1l program
BEGIN
COMMENT THIS PROGRAM REPRESENTS THE PL-11 CODE FOR THE EXAMPLE FOUND ON PAGE 26 OF THE DEC PL-11 HANDBOOK. AS WRITTEN, IT WILL GENERATE EXACTLY THE SAME NUMBER OF INSTRUCTION WORDS AS THAT EXAMPLE. THE PROGRAM GENERATES A HISTOGRAM OF THE FREQUENCY OF OCCURRENCE OF ALL VALUES IN THE RANGE 1-100 IN 'ITABLE';

ARRAY 100 INTEGER 'ITABLE'; ARRAY 1000 INTEGER 'ITABLE';

COMMENT NEXT CLEAR THE OUTPUT TABLE TO 0. NOTE THAT IN PL-11, THIS TABLE WOULD BE AUTOMATICALLY INITIALIZED TO 0 WHEN THE PROGRAM IS LOADED. HENCE, THIS CLEARING IS NECESSARY ONLY IF THE TABLE HAS BEEN USED PREVIOUSLY;

REF('ITABLE') => RO; FOR R1 FROM 1-100 STEP 1 UPTO 0 DO 0 => P8P(RO);

COMMENT HISTOGRAM ALL THE ITEMS IN ITABLE WITH VALUE 0 <= VALUE <= 100;

REF('ITABLE') => RO; 100 => R2; COMMENT PUT LIMIT IN R2 FOR SPEED;

FOR R1 FROM 1-1000 STEP 1 UPTO 0 DO
  IF P8P(RO) => R4 > 0 & R4 <= R2 THEN
    COMMENT VALUE WITHIN RANGE, ADD IT TO THE HISTOGRAM BIN;
    BEGIN R4 SLA; 'ITABLE(R4) + 1;
  END;

IEND.

004234 012700
004236 000004+BASE
004240 012701
004242 177634
004244 005020
004246 005201
004250 003775

004252 012700
004254 000314+BASE
004256 012702
004260 003144

004262 012701
004264 176030

004266 012004
004270 003405
004272 020402
004274 030003

004276 006304
004300 005264
004302 000004+BASE
004304 005201
004306 003767

004310 002007

Fig. 12 A PL-11 program with generated code