+---------------------------+
d	i	g	i	t	a	l
+---------------------------+

i n t e r o f f i c e

m e m o r a n d u m

To: Per Hjerppe
Peter Hurley
Alan Kotok
Walter Manter
Allan Titcomb

Date: 19 Jul 83
From: Mike Uhler
Dept: Large Systems A/D
DTN: (8-)231-6448
Loc/Mail stop: MR01-1/L26
Net mail: IO::UHLER

Subject: Impressions from a visit to Foonly

1.0 Introduction

On Tuesday, July 12, 1983, Alan Kotok and I spent the day talking to Dave Poole of Foonly. The purpose of the trip was to discuss the technical details of upgrading the Foonly F1 design to support extended addressing, with the possibility that Digital and Foonly could develop some sort of business relationship.

This memo gives my impressions of the existing F1 design, the proposal to upgrade the F1 to support extended addressing, and the possibilities of success.

Dave has written the outline of a proposal for this effort, which is attached to the end of this memo.

2.0 The existing Foonly F1 design

The existing F1 is a single-section, pipelined, PDP-10 processor built around 1978. It uses 10K ECL logic and has a cycle time of 90 ns. At present, the design does not support extended addressing and is therefore functionally equivalent to a KL10 model A, although it emulates a KA10.

The F1 uses a multi-stage pipeline with the following components:

- Prefetch unit. This piece of the design is dedicated to prefetching and holding instructions that may be executed. It has room for up to three instructions, but the three registers are not always cascaded. As a result, an instruction doesn't have to trickle through three stages of

registers if the pipe is empty. If the pipeline is nearly empty, the prefetch unit uses a high priority request to the MBOX; if it is nearly full, it uses a low priority request.

- IBOX. This piece of the design takes care of operand fetch for the instruction (both AC and E), and passes the operands to the EBOX for execution. It also holds (and passes?) the effective address around. The effective address carries several tag bits along with in including an indication that the address references the ACs.

  The IBOX control memory (i.e., the IBOX microcode) controls the IBOX processing for each instruction. One field of this memory is used to encode special operations for the more complex instructions (e.g., byte). In the case where the EBOX is performing a complex instruction, the IBOX synchronizes with the EBOX and assists in the execution of the instruction (e.g., the IBOX does the memory reads for BLT, while the EBOX does the stores).

  If the instruction is a jump, the IBOX starts a data reference to fetch the target of the jump. If the jump actually jumps, some sort of magic happens that causes the prefetch unit to flush and the data fetch to become an instruction fetch.

- EBOX. The EBOX is responsible for executing all instructions. It may also have some help from the IBOX for complex instructions. The initial EBOX microcode address for an instructions is based on the opcode, plus some qualifier bits that characterize each class of instruction. For example, the initial address for PUSHJ would be one of two locations depending on whether the stack pointer was local or global.

  The EBOX has special hardware to accelerate the performance-critical instructions. Dave was claiming 3 cycles for the byte instructions, 2 cycles for PUSHJ, and 3 cycles for POPJ. This is very impressive if it can be done.

  The EBOX also implements the memory management algorithms to perform pager refills, etc.

- MBOX. The MBOX contains a page table and a cache. I don't remember what the sizes of each is on the existing F1. His proposal called for a 4K, 1-way associative page table, and an 8K, 4-way associative data cache. The page table contains a process ID field to decrease the need for a pager sweep operation. We pointed out the problems with a 1-way associative page table, and he said that he'd consider making it 2-way associative. Cache hits return data in 1 cycle; cache misses return data in 4 cycles.

The MBOX also contains the memory controller and the interface to the I/O subsystem which he believes has next to infinite bandwidth.

In the short time that we spent talking about the design, I saw no obvious "gotcha's" that would cause the design not to work if extended addressing were added. The fact that the pipeline control logic already exists (and works) is a big plus.

3.0 Adding extended addressing to the F1

The Foonly proposal calls for upgrades to the F1 design to add the necessary support for extended addressing. He's already thought about some of the changes that are necessary and we talked about them briefly while we were there.

He believes that most of the changes are isolated to the address generation logic in the IBOX, which is probably true. as a result, most of the changes are probably limited to one section of the existing design, which decreases the risk of making the changes. In addition to the hardware changes, there are also significant microcode changes to add certain KL instructions, memory management, and PXCT.

It was clear to me that Dave has an appreciation of the changes necessary to add extended addressing. However, I believe that he needs my memo on extended addressing, plus some occasional consulting to get it right. Without this, there could be some problems.

4.0 The F1 I/O subsystem

The F1 I/O subsystem uses the Foonly FBUS I/O controllers to interface to the DMA ports in the MBOX. I/O reads check the cache for valid words and I/O writes invalidate any valid words in the cache.

Dave claimed that normal I/O could only use half the MBOX bandwidth, so there should be plenty of MBOX to service both I/O and the CPU.

The range of I/O controllers includes interfaces to SMD disks, Pertec tape drives, and other unit record peripherals. Foonly also has interfaces to a 10 MB ethernet and to the ARPAnet. Dave felt that the development of additional I/O adapters would be possible, but that wasn't included in the estimates that he gave.

Undoubtedly, we'd ultimately want interfaces to the corporate buses, especially the CI, but we shouldn't delay FCS to build one.

5.0 Software effort

The software effort necessary to make TOPS-20 run on the F1 is about the same as for any other processor. Dave claims that he'll have some version of TOPS-20 running on an F4 in the next few weeks. I'm not sure about this, but I believe that the version that he's bringing up is a Release 4, model A monitor.

In any event, he described his software person as a wizard and they're probably capable of doing the software work to support the current version of TOPS-20 on the F1. They'd need consulting from the TOPS-20 group, but I suspect that it would be limited to that.

6.0 Cabinetry, power supplies, boards, etc.

Dave has assumed that Digital would supply the engineering resources to design the cabinetry and power supplies to meet the FCC requirements. This could be a problem given the long lead times for this kind of thing within the company. The alternative is to have the work done by having Foonly hire outside consultants to do the work.

The F1 uses large wire wrapped panels on some sort of hinge mechanism to allow access to the boards. The panels are identical to those used by the S-1 project so that the CAD tools developed by that project can be used on the F1. Dave believes that the panels could be converted to PC or multiwire later on, but that the first machines should be wire wrapped to avoid any delay in FCS.

It's clear that wire wrap modules aren't particularly reliable (the pins seem to have an affinity for other pins and the wire gets cut on the pins), or suited to high-volume production. We'd have to do something about that as soon as possible.

7.0 Design automation

The SCALD system from Lawrence Livermore Labs is in the public domain, and Dave said that he intended to use that as the schematic entry vehicle. SCALD also includes a logic simulator and a timing verifier and he wanted to use both of those.

Note that the LLL SCALD was the basis for the version now sold by Valid, and is not what they sell now. There's some reason to believe that there may be some significant problems with the one from LLL.

8.0 Maintenance and debugging features

The F1 hardware includes extensive history memories which store the last 1K operations on the machine. In addition, the console computer is extremely powerful and has the ability to adjust clock skew, breakpoint the hardware, dump the state of the hardware, etc.

Dave's proposal includes a lengthy section on maintenance philosophy. We talked more about this during the day, and he seems to have an appreciation of how to diagnose and maintain hardware. His approach is somewhat revolutionary when compared to the traditional Digital maintenance philosophy, but it could be quite effective if he can do what he says.

9.0 Can the project be done?

The fact that a running machine exists to demonstrate the F1 design improves the chances of success. While the addition of extended addressing isn't trivial, it is certainly easier than designing an entirely new, pipelined machine. I believe that Dave has a reasonably good chance of pulling it off in the time frame that he's indicated.

I do believe that his success hinges on getting the consulting that he needs (e.g., on the right way to do extended addressing), without swamping him with red tape. Forcing him to adapt to the normal Digital hardware development process including phase reviews, etc., will effectively kill the project.

OVERALL PLAN for F1B

Design: Make minimal changes to F1.
KL10 compatibility is needed, but use Foonly i/o controllers.
Foonly will do logic, layout (using SCALD), firmware, and software, while DEC will do enclosures and metalwork design.

Manufacturing: Use S-1 methods, hardware, and vendors.
Lawrence Livermore is making all this available, and it is exactly adapted to the F1's needs. DEC could provide the plant.

ASSUMES SCAM AVAILABLE FULL SCM

Checkout: Use Foonly methods.
These are based on heavy use of the Console Computer debugging aids. Foonly will train suitably experienced personnel, provided (presumably) by DEC.

Maintenance: Use Foonly remote maintenance equipment and methods.
Field personnel require only 1 week training. Foonly will train Maintenance Center personnel; both experienced medium-level hardware and system software people are needed.

Schedule: 12 months to delivery of first test site system.

Scheme: First, acquire F1 prototype and find (at DEC ?) the experienced people to be trained by Foonly for checkout and maintenance; training will begin at once and consist partly of participation in the F1B design and development process, which will use the prototype machine as a test bed and training tool. Microcoding and CC coding projects will start at once, and the SCALD system will be brought up under TOPS-20. DEC engineering will begin work on enclosure, metalwork, power distribution. Conservative orders of long lead time parts for the first few machines will be issued during the first few months.

Foonly will have the logic changes and layout done after 4 months, and two F1B pre-production units will be built during the next 2 months (all construction except wirewrapping will have been done in advance). After 2 months of checkout on these units, the first few production systems can be wrapped; production versions of mechanical subsystems are needed then. Two production units can be checked out by the end of the 12th month. The one-week field service training courses can be given at that time.

The rate of production can be 1 system/month per checkout person at first, rising to twice that rate. At first, of course, Foonly personnel will be heavily involved in supporting checkout and maintenance.

^L

DESIGN

There are three essential elements for the timely completion of the design phase of this project. The first is the acquisition of the prototype F1, now being offered for sale by III. This machine has been reliably emulating a KA10 (running TOPS-10) for five years, and it represents a proven starting point from which straightforward modifications can attain full KL10 functionality at 3 or more times KL performance. Having the F1 available from the beginning will be invaluable for training, microcode development, electrical engineering measurements, and the testing of small but subtle control logic design changes. The F1B will be in all respects so similar to this prototype that results obtained in these areas should be accurate.

The second important aid to the design effort is the avavilability (free !) from Lawrence Livermore Lab. of the "SCALD" system for CAD, as well as the complete design information for their S-1 (Mark II) supercomputer. Since the S-1 uses ECL 10K circuits, and since its packaging, interconnect, and power supply design was derived directly from the F1 prototype, all of this federal largesse is directly useable (even optimal) for the F1B. In particular, the custom ECL wirewrap panels, and tri-lead signal cables will be adopted without change. SCALD will take existing F1 drawings (from SUDS) as input. It has a logic simulator that really works (2 or 3 seconds per machine cycle for 10,000 ic's !), and also a "timing verifier" that examines the whole system analytically for logic paths that are too slow (using actual computed wire delays, after layout is done). SCALD also does the layout, thus saving an estimated 4 man-months of stultifying human effort.

Third, DEC's capabilities and expertise can ensure that the critical and time-consuming designs of metalwork, enclosures, and power systems are executed in conformance with regulatory and marketing requirements.

Additional important assets are Foonly's existing Console Computer system and programming language, our experience of (soon) running TOPS-20 on our F4 system, and our proven MOS memory and FBUS I/O controller designs. All of these will be directly utilized in the F1B.

DESIGN/DEVELOPMENT TASKS (and WHO DOES THEM)

Metalwork (DEC or professional consultants)

> The mounting of the wirewrap panels will follow the F1, with modifications to accomodate the slightly different S-1 panels; 2 panels will mount in each of 5 hinged pages. Power supply to the panels is by multi-layer bussbars (S-1 technology) integral with the page, thence via the same bolts which hold the panels. Power supply mounting can be varied to suit the desired enclosure; connection from supplies to bussbars is by welding cable.

^L

Enclosure (DEC or professional consultants)

Subject to the constraint that it enclose the 5 pages of logic panels (with room for them to swing when doors are open), the enclosure may be designed to satisfy the requirements of marketing and the FCC. It will have to be taller than a KL, but the footprint can probably be smaller.

Power Supplies (Foonly or DEC)

Foonly uses commercial switchers; DEC can supply or choose its own.

Logic, Firmware, Support Software (Foonly)

The F1 prototype will be used haevily in this devlopment. Among other things, the Wiring Tester (see under Manufacturing) will be used to verify that the logic drawings used as the basis for the devlopments below do in fact represent the running prototype.

KL Pager

This is part hardware and part microcode. Foonly has already implemented this on the F4; the hardware and microcode are both easily translatable to the F1. The pager hardware includes a translation table with 4k entries, used pseudo-associatively. A "context" field in each entry (together with context register) almost eliminates the time-gobbling "invalidate pager" operation.

Extended Addressing

Hardware will handle this, except for some XCTP microcode. Foonly is now debugging an F4 implementation, which will perfect our understanding of the specifications. The required hardware is very straightforward on the F1 because of the high degree of separation of functions in that system; the address generation logic is dedicated to its task and the extended addressing modifications will not interact confusingly with other parts of the machine. No performance penalty is anticipated.

KL Instructions

These are mostly a matter of microcode, although some design changes will be made to support XCTP, the new byte pointers, and possibly the (shudder!) the business instruction set. The F1 already has KI style double precision arithmetic hardware and microcode, although the microcode is not fully debugged. Various KL instructions (eg, ADJBP, ADJSP) have been implemented on the F4, which will provide useful models.

^L

Internal Memory

The F1's memory interface will be modified to control Foonly's standard memory board, which uses 16k or 256k MOS dynamic RAMs; this will simplify the interface considerably (it now talks to DEC-like Ampex boxes). With 256k RAMs, 16 boards will provide 16MW of memory, which seems like a good amount. The usual ECC functions (by the word) will be provided, using our current algorithms.

The speed of the system will be substantially increased by this memory change, as the time lost on a cache miss (for a fetch) will be reduced from 12 to 4 cycles (about 320ns).

FBUS I/O

An extensive line of FBUS i/o controllers is currently manufactured by Foonly, so providing the F1B with an FBUS will immediately permit attachment of many devices, including:

- disk (SMD interface)
- tape ("Pertec" standard OEM interface)
- terminals (to 19.2kBd)
- line printers ("Data Products" standard OEM interface)
- electrostatic plotters (Versatec)
- ethernet (10MB)
- ARPANET
- Tymnet (actually a PDP-11 interface (DR-11C))
- bit-map display (Foovision)

Variations on these interfaces are usually easy to produce, but none are allowed for in the 12 month plan (unless additional design manpower is made available).

To provide plenty of bandwidth, 4 FBUS controllers will be included, each attached to the (internal) microcode-controlled EBUS and to one of the buffered DMA ports of the F1's I/O controller. The microcode needed to support the various FBUS controllers can be simply translated from the F4.

Expanded Microcode Memory

New ECL RAMs make it easy to increase the F1's 2k of microcode storage to 8k. The word length may be increased from its present 72 bits, if needed to support new features.

Expanded History Memories

History memories on both the F1 and F4 have proved to be of immense value. New ECL RAM's will be used to expand all of the F1's history memories to 1k entries.

^L

CC Interface

The Console Computer interface will be (trivially) modified for compatibility with our current microprocessor-based CC, which will be used unchanged in the F1B.

CC Software

The existing software will be translated into CCL, the special interpretive language developed for Foonly's current CC. Some additions will probably be made.

The CC and its software are the heart of Foonly's checkout and maintenance procedures, and in combination with our history memories and other debugging features they can revolutionize both tasks. Very careful attention will therefor be given to this software. Since CCL code is interactively modifiable (like LISP), debugging functions are often generated or improved during actual use. Thus it is to be expected that development will continue during the period of production and support of the F1B.

8K CACHE (optional)

The increases in density (and speed) of ECL RAMs since 1976 have been so great that quadrupling the F1's current cache size would be simple (and would save space !). This will be done if time permits and if investigation shows that it would be beneficial.

KL Compatible Accounting Hardware (?)

DEC I/O ?

IBM I/O ?

^L

MANUFACTURE

Buy S-1 wirewrap panels (Augat) and pre-fabricated tri-lead signal cables.
Find second sources for these items.

Install soldered ground clips on panels.

Have panels wrapped by the S-1's vendor.
They know how to handle them; but devlop alternate vendors.

Test wrapped panels on Wire Tester.

Test ICs with a complete functional tester.

Stuff IC's, resistor networks, and bypass capacitors.
Everything just plugs in; no soldering is done.

Test stuffed panel on Wire Tester.
This catches shorts caused by pins bent while stuffing.

Assemble metalwork, enclosures, and power system.

Install checked out panels and tested cables to form new system.

^L

CHECKOUT

New logic panels will be installed one at a time in an otherwise operational machine. Since the Wire Tester will have found wiring errors, the new panels should work if the stuffing is correct and the ICs have been tested (note that the IC's have not been through a soldering operation).

DEBUGGING AIDS

The Console Computer (CC) is the central element, of course. It uses a display terminal to present information in dynamically updated displays. Because many screenfuls of data are accessible, the user is allowed to define screen formats containing particular displays, call them up by name, and modify them at any time.

CC controls power, senses voltages and temperatures.

CC can adjust clock speed and skew delays.

CC can load and examine registers, memories, and data paths.

CC can test for cables being plugged in correctly.
One wire in each signal cable is devoted to this function.

CC controls the logic analyzer.
In conjunction with the CCTALK program on the support system (see description under "Maintenance"), this greatly improves the utility and ease of operation of this excellent tool. Condition setups can be stored and edited, and displays of data are more flexible, have symbolic lables, and can be stored.

History Memories record the last 1k machine cycles.
Items recorded included macro and micro PCs, MBOX addresses, and dozens of signals indicating the type of operation performed and the result (eg, Instruction Fetch with Cache Hit) by each of the functional units of the machine.

Hardware Breakpoints are available for macro and micro addreses.

The system can be single stepped.

Functional units can be separately disabled.

Certain major functions can be disabled to simplify machine operation.
For example, the cache can be turned off. Also, the overall piplining can be inhibited, so that only one functional unit at a time is operating; this not only helps isolate bugs in the (rather complex) pipeline control logic, but also often allows the machine to keep running in the presence of bugs.

^L

MAINTENANCE

A new level of responsiveness and effectiveness will be attained in dealing with interruptions of system availability, whether caused by hardware failure or system software problems. The improvement over traditional methods will result from our new Instant Expert Service (IES); it works as follows.

Upon detection of any abnormality in system operation, the customer notifies the Support Center (SC) (by telephone or network). SC personnel immediately connect to the customer's system via the dedicated modem in the Console Computer of the F1. An expert in the diagnosing of malfunctioning systems then determines the general nature of the problem (operator error, user confusion, system software, probable hardware failure). This expert is a specialist in this task, knows how to use Exec DDT and basic hardware debugging aids, and has available an extensive set of databases (online in the SC computer system) containing up-to-date information on the exact configurations, version levels, and problem history of the customer's system. No time will be lost, therefore, before the SC has an accurate idea of the difficulty.

Experience shows that 90% of all problems can be solved (or at least system operation restored) at this stage by expert intervention. Operator errors and user confusions can be cleared up; known system problems can be corrected by immediate patches or special intervention; and software distribution problems can often be fixed by transmission of updates via the maintenance connection. Most interruptions of service can thus be remedied with little delay.

If the Diagnostic Expert concludes that a hardware problem is likely, a Support Engineer takes over the investigation. This engineer's qualifications are very much higher than that of a field service representative, and furthermore he begins his examination with the benefit of a detailed report on the observed symptoms, obtained from the Diagnostic Expert. Using the unprecedented maintenance aids of the F1 and its Console Computer, as well as the SC database (which includes on line all hardware drawings and wire lists), the Support Engineer localizes the failure. If, as sometimes happens, he would like to do experiments with the help of an operating system expert, one is near at hand. Meanwhile a field service representative can be en route to the customer's site with spare parts and test equipment. By the time he arrives, the engineer will probably have a good understanding of the failure, and replacement of failed parts proceeds under the engineer's careful supervision, which nearly eliminates the common problem of secondary trouble induced by inexpert field service.

This scenario is admittedly idealized, but it illustrates the fact that makes the IES system so effective in the real, non-ideal world: every stage of the response to each problem is conducted by an expert. The advantages of this are clear, but they can be realized fully only by a carefully integrated system of exceptional debugging aids,

^L

precisely controlled databases and procedures, and (most important) the commitment of personnel with sufficient expertise. We believe that our systems can achieve these goals more fully than ever before.

IMPLEMENTATION of MAINTENANCE

When correctly implemented, Foonly's remote maintenance methods not only deliver improved service to the customer, but also simplify the tasks of the maintainers. Software and hardware fixers can get more done per unit of time from the remote Support Center than they could if they were on site. This is largely due to the instant on-line availability of documents and records, but several other advantages are importatant. These include the presence of a wide range of experts for consultation, the presence of working systems with which to experiment and check results of observations, and improved facilities for using the logic analyzer.

Support Center resources:

The SC needs a fairly large system, and a backup system; these machines do not have to be totally dedicated to the SC, but they must be always available. They will have modems with auto-dial, graphics displays (for SUDS and remote logic analyzer displays), lots of good Emacs terminals, and plenty of disk storage.

All communication with customer systems is via the CCTALK program, which dials out to the remote Console Computer and talks to it with checksummed data packets. CCTALK also implements certain protocols for record keeping during maintenance, handles the remote logic analyzer, and (very importantly) keeps a log file of all activity. Each work station in the SC will need a (display) terminal for CCTALK, an Emacs terminal for accessing the database, and a color graphics display (Foovision) for drawings and analyzer displays.

The database at the SC must have all software and firmware sources and assembly listings (all versions), all logic drawings and wirelists, all hardware, software, and diagnostic documentation, and the latest news and notes regarding current software and hardware features and fixes. In addition, for each customer system the database will have maintenance history, current status (hardware and software !), and the CCTALK logging files.

^L

Support Center "System Diagnosis Experts"

These people are the keystone of the maintenance structure; it is they who first look at every reported problem. They must master the subtle art, well known among the "wizards" found in highly computer-oriented organizations, of examining a malfunctioning system and determining the probable category of the trouble.

They are not "system programmers", still less "engineers", and absolutely not "operators" of any degree. They have various backgrounds (none are novices), but all are excited by the challenge (and power) involved in doctoring sick systems. Probably several such people exist within DEC; more can be found at universities and timesharing companies. Only a few are needed. They will be attracted by the flashy debugging tools available at the SC and in the F1B, and also by the surrounding community of wizards of all kinds.

These "diagnosers" will know what to look at when a system stops, using EDDT, history memories, and the CC's record of recent output on the CTY. They will know the meaning of common error messages and halts, and be able to use breakpoints at certain informative places in the monitor. They will be able to judge when it is safe to continue running, and when to take a dump of core (sic). The signs of the more common hardware troubles will be familiar to them, as will the effects of the current system bugs. The more common forms of confusion among operators and users, which generally account for very many problems, will be easily detected.

Most of the diagnosers will probably want to be involved in new development projects while waiting for trouble reports, and they would make very useful members of such projects, being as closely connected as they are to the Real World.

It is difficult to overestimate the benefits of having the right people in this position; no effort should be spared in finding and supporting them.

Support Center hardware fixers:

These can be some of the same people who do checkout, since the same tools and methods will be used. In this case they will depend on on-site field service to replace parts and attach logic analyzer probes (and wire tester probes !). Foonly experience has shown that a hardware fixer will choose to operate from the SC even if the sick machine is in the next building.

^L

When a fixer starts to look at system, he will have a meaningful, believeable report on the symptoms and their history. If others have worked on the problem, he can find out from the CCTALK logs exactly what they did. If the fixer starts to suspect that the problem is software after all, or if he wants to do some experiment involving the operational software, he has software experts near at hand.

If the trouble is really a bad IC (a relatively rare occurence), the fixer will make a couple of guesses and have the on-site helper replace a few DIPs (in groups of 4 or 5 -- DIPs are cheap). The fixer, not the helper, controls the powering down and up of the system for this operation. If the initial DIP replacement is not successful, the logic analyzer will be used. If a fault in the wiring (a cut-through or bent pin, for instance) is suspected, the relevant parts of the circuitry can be checked with the CC's built-in (resistance measuring type) wire checker.

Field Service personnel:

Really, they need only one week of training (except for i/o equipment, of course). They must be able to type well enough to communicate with the remote fixer, and they should learn the protocols developed to make such interaction go smoothly -- standard phraseology for requests and reports of compliance are the main aspect of this.

If the field service types on site actually always operate under direction from the SC, huge amounts of confusion, misdirected effort, and induced problems will be avoided.

Maintenance of I/O Devices:

With current Foonly controllers, the F1B can use Pertec tape drives, which DEC knows how to maintain. Any disk drive with the SMD interface can be used; it would also be simple to handle some other straightforward disk drive interface. So, we can probably use drives already known to DEC maintenance. Similarly, it should be easy to use some line printer that DEC already maintains.

Even for the repair of i/o devices the supervision of a Support Engineer at the SC will prove valuable, both for accurate record keeping and for running of diagnostics.