CLM-R202

CLM-R202

UKAEA

Report

INITIAL SCL USER INTERFACE FOR THE 2976

R. ENDSOR

CULHAM LABORATORY
Abingdon Oxfordshire

1980

C/13 Available from H. M. Stationery Office £1.50

CLM-R 202

INITIAL SCL USER INTERFACE FOR THE 2976

R Endsor

Culham Laboratory, Abingdon, Oxon, OX14 3DB, UK

(Euratom/UKAEA Fusion Association)

Abstract

This report describes work performed on an initial user interface for the Culham 2976. The interface consisted of SCL macros and procedures written in S3. The interface was designed to support batch jobs submitted from the twin PRIME 500 computer systems and to give a straightforward mapping from PRIME Culham commands onto 2900 SCL.

20 March 1980
dmg

ISBN 085311 086 7

CONTENTS

1. Description of Culham central computer configuration

2. The user interface on the 2900, a description of SCL

3. Culham commands on the PRIME

4. Design of the batch user interface

   4.1 The organisation image

   4.2 Filestore structure

   4.3 Facilities supported

   4.4 Usability and housekeeping features

5. Implementation of the batch user interface

6. Further enhancement of the 2976 user image

Acknowledgements

References

Appendix 1 Examples of use of the batch user interface

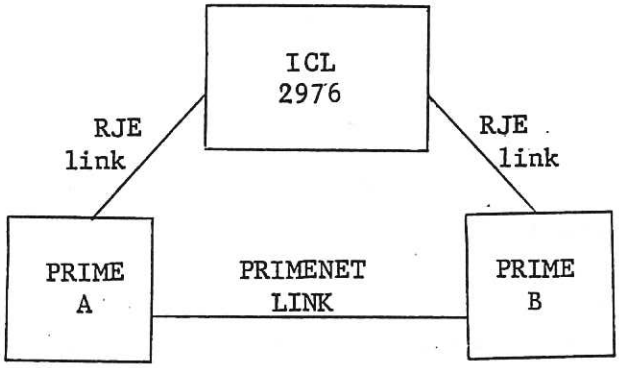
Appendix 2 Interface structure

* Appendix 3 Definition of the user interface

*Note: Appendix 3 is available by special request

1. Description of the Culham Central Computer Configuration

The central computer configuration at Culham consists of linked ICL 2900 and PRIME 500 computers. These form a Front End/back end system with the Front End rôle taken by the twin PRIME 500 systems. This is shown below:



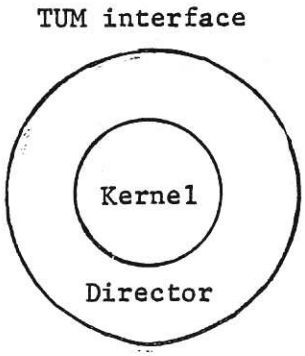
The two PRIME systems are linked together by a fast serial link known as PRIMENET. Each PRIME is linked to the 2976 by an RJE synchronous link running at 9600 baud.

The rôle of the PRIME systems is to handle terminal communications with user terminals distributed throughout the Laboratory. Interactive programs and program development can be performed on the PRIME systems. The 2976 is a background processor receiving jobs over the RJE links. In the initial configuration very little interactive work will be run on the 2976 although a few terminals will be directly connected for systems development.

A reconfiguration of the communications links between the 2976 and PRIME systems will take place soon. The synchronous RJE links will be replaced by much faster BSI parallel links (BS 4421). At this stage interactive working through the PRIME to the 2976 will be possible and a full 2900 user image will then have been developed.

2. The 2900 User Interface - A Description of SCL

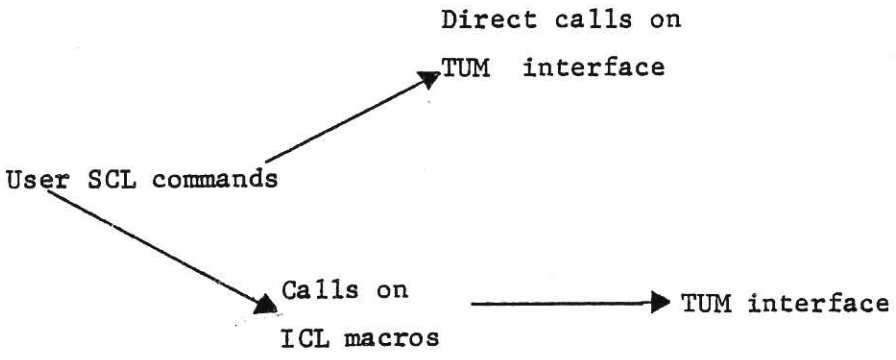
The 2976 at Culham runs the VME/B operating system produced by ICL. This operating system is structured to make use of the protection features built into the hardware. It is conceived as layers of software, the most trusted being the kernel which is fully protected from all other software. Surrounding the kernel is the director which performs VME/B supervisor functions. The outer layer of director forms an interface to user level software. This interface is known as the Total User Machine. The basic structure of VME/B is shown:



The user interface provided by ICL is thus built upon the underlying TUM interface which consists of VME/B procedures.

The kernel and director are mainly coded in the ICL software development language, S3. This has many features in common with ALGOL '68 and supports complex data structures and structured code. The structural design of the whole system is modular; the basic unit being known as a "holon" with well defined interfaces. This design concept was initiated for the construction of VME/B and was supported by a design database called CADES (see Ref. 1). The TUM interface is, therefore, a procedural interface of S3 routines and data types. It should be noticed that this interface is essentially open in the sense that the facilities of VME/B are available to the overlying user interface.

The user interface is provided by a System Command Language, known as SCL, together with a command language interpreter. The user, therefore, expresses the actions which are to be performed as SCL statements which are then interpreted and obeyed. Two main access paths to VME/B facilities are possible. In the first case SCL supports a subset of S3 data types and as there is a common procedural interface it is possible for SCL statements to interface directly to TUM procedures. Secondly ICL have provided a large set of SCL macros (which are essentially equivalent to procedures) which provide VME/B facilities. The situation is shown below:

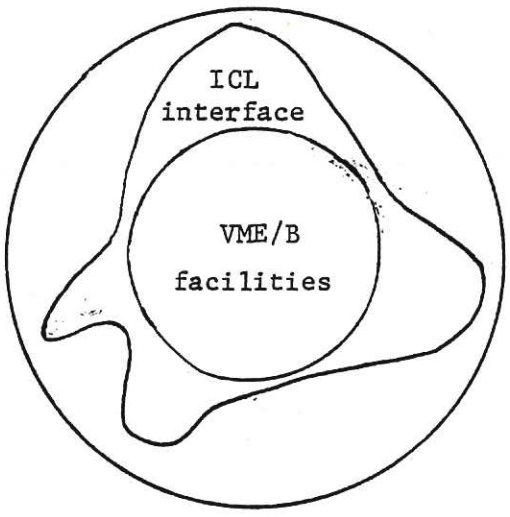


A third mode of access is available as it is possible for user produced S3 routines (which access the TUM interface) to be called from SCL commands. The user interface to VME/B is, therefore, very flexible and there are several approaches which an installation may adopt.

The simplest approach is for the user to work entirely with ICL macros (see Ref. 2) possibly supplemented by an approved set of customer macros known as GOODIES. These macros cover the full range of VME/B facilities and are very comprehensive. The user is, however, faced by the difficulty of understanding VME/B in all of its aspects. There are over 350 standard SCL macros some of which have over 30 parameters. Although the effort and time involved in understanding VME/B is acceptable to a professional programmer, this will not be the case with the scientist who wishes to use the computer as a tool. An installation would not usually offer a service to its users comprising only standard macros.

The alternative approach is for a set of installation defined macros to be produced. Each installation will wish to present to its users an interface which is suited to the way in which they work. The flexibility of SCL allows the installation to tailor VME/B facilities to the extent that the user interface can be radically different from the underlying software. It is possible for the interface to define objects, structures and semantics in a simple and self-consistent way. The user is presented with the installation defined system which is most acceptable. This is shown below:

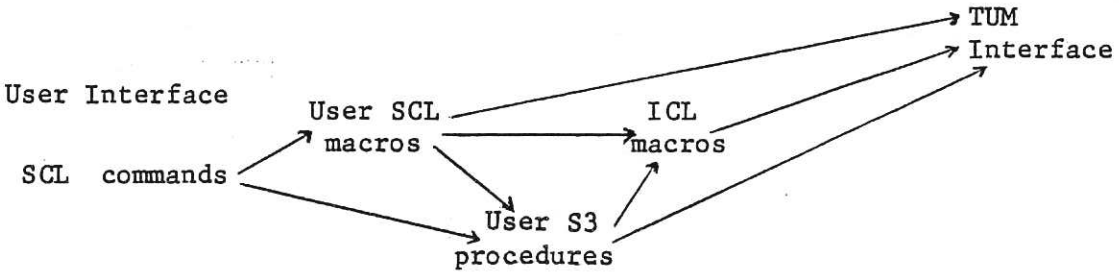
Installation interface



Communication between the user and the computing system is a two-way process. The installation interface defines both the commands which the user will give to the system and also messages which will be returned from the system to the user. This is particularly true of error messages where the prime aim of the message

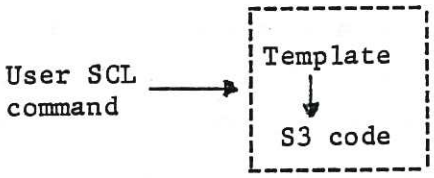
should be to explain the fault in terms which are meaningful to the user. Too often the error message relates to the internal structure of the operating system and requires expert interpretation. Again the flexibility of SCL will allow the user interface to have full control over messages which will be received by the user.

In producing a user interface an installation can use SCL only. There are, however, some disadvantages in this approach. The installation macros may be complex and as each macro must be interpreted as it runs the overheads may be quite considerable. In a later release of VME/B (5X36) software will be provided which will allow SCL procedures to be complied into an intermediate code to achieve faster execution. Another disadvantage is that the SCL language is fairly restricted in the data types which can be manipulated so that the best algorithms cannot always be used. Similarly, because SCL supports only a limited number of data types the mapping of SCL calls onto the TUM interface is restricted. These disadvantages can be avoided if the installation user interface is coded in S3. As S3 is fully compiled into code before execution it is much more efficient and also is fully compatible with the TUM interface which is similarly coded in S3. Although ICL had previously not allowed customer use of S3 this restriction has been removed and S3 is available as a special product. In practice the user interface will be a mixture of SCL macros and S3 procedures.



VME/B supports a common procedural interface to all its languages. The user interface could, therefore, also contain routines written in FORTRAN, COBOL or ALGOL 68. These languages are less useful than S3 because of the restricted data types which they support.

An S3 procedure may have parameters associated with it. If, as part of the user interface, it is required that the user should be able to call that procedure from SCL then the user will need keywords for the parameters and default values will have to be supplied for optional parameters. This facility is added by a "template" defining parameter keywords and defaults being attached to the procedure.



Templates can also be attached to procedures written in other languages.

A user defined interface to the 2900 can either replace or exist alongside the standard ICL macro interface. In the latter case there is a restriction that macro names must be unique between the two interfaces. In some respects restricting usage to a single user interface is preferable as this reduces the support and documentation required by an installation. This is the approach which has been adopted at the South West Regional Computer Centre at Bath. The drawback is that all users are restricted to facilities provided by the installation interface. In a situation where, although most users will run scientific programs, commerical work also has to be supported, a single user defined interface may not be sufficient. The solution at Culham was to develop a simple user interface which would be attractive to scientific users alongside the standard ICL interface.

A basic feature of VME/B is that whatever macros or procedures are provided by an installation the user still has the facilities arising from the fact that SCL is a language and each user can produce macros which will be tailored to his requirements. Users providing applications programs as a general service can, therefore, develop the applications user image to their own specification.

To summarise the VME/B system has been designed to provide a flexible and extensible user interface. The basic building blocks, standard ICL macros, are comprehensive but use of these macros is really restricted to computing specialists. The aim of the initial user interface for the Culham 2976 has been to provide a simple and readily acceptable interface for scientific users.

3. Culham Commands on the PRIME

The PRIME front end computer systems have essentially taken on the Satellite rôle played by the CTL Modular One computer in the previous Culham configuration based on System 4-70 computers. One aspect of the CTL Satellite system was that it supported a General Command Language, known as GCL. The purpose of this language was to enable jobs, expressed by GCL statements, to be run on any mainframe attached to the Satellite. In practice this meant that jobs could be submitted to the System 4-70 or to the AERE IBM 168. GCL statements were processed by a macro processor, known as JOLT, into job control statements for the appropriate target systems and the job was then submitted via an RJE link.

One problem with GCL was that it was a separate subsystem on the CTL Satellite and so was not widely used. On developing the Satellite functions on the PRIME it was decided that the functions of GCL should be taken over into a general set of commands, known as Culham Commands. Having defined a target system (using an RJOB command) the user would then enter other commands describing the action to be taken by his job. Each Culham command would be expanded into the appropriate command language for the target system. This approach is more flexible in that Culham commands could form the basis for system independent interactive use of remote systems. (It is expected that Culham users will have interactive access to the 2976 from the PRIME later.)

On investigating the possibility of mapping PRIME Culham commands onto standard ICL macros it was found that this process was by no means straightforward and that each Culham command would have to generate very complicated SCL. The alternative approach, to build a user interface on the 2976 which corresponded to the Culham Commands on a one-to-one basis,was adopted. A positive advantage from this was that the user interface could also be used from terminals directly attached to the 2976. Error messages could also be handled by the user interface and this is currently not possible with Culham commands.

The initial aim of the SCL interface for the 2976 was, therefore, to provide a straightforward mapping for Culham commands. Not all the functions of the SCL interface were defined in this way. Some extra facilities were added which are not covered by the current implementation of Culham commands. However, it is true that the SCL interface has developed from the GCL and Culham command systems.

4. Design of a Batch User Interface

The batch user interface which is now supported on the 2976 has evolved from design criteria arising from the overall structure of the Culham computing configuration. A basic decision which was taken at an early stage was that users' source files would normally reside on the PRIME. Editing, which is essentially an interactive process, would be performed on the PRIME. User jobs sent to the 2976 would, therefore, contain any source which was to be compiled and no source would be permanently stored on the 2976. Compiled code would reside on the 2976 as would most data files. There would, however, be data files which would be transmitted from the PRIME and it was also decided that it should be possible to direct output files from the 2976 to nominated files on the PRIME. A second criterion was that user programs written in FORTRAN on the System 4-70 should be able to be run on the 2976 with minimum inconvenience. Fortunately a compiler, known as the FORTRAN(G) compiler, produced by the Edinburgh Regional Computer Centre, had been enhanced to compile any System 4 FORTRAN program.

The user interface, therefore, supported this compiler. A further aim was eventually to extend support to the ICL Optimising FORTRAN Compiler known as the OFC compiler. For this the difference in using the two compilers would be minimised. At present there are still incompatibilities in the handling of input/output between the two compilers and the standard ICL compiler will not be fully supported until these differences are resolved. The prime feature of the user interface is that it is designed for a remote batch work-load based on FORTRAN.

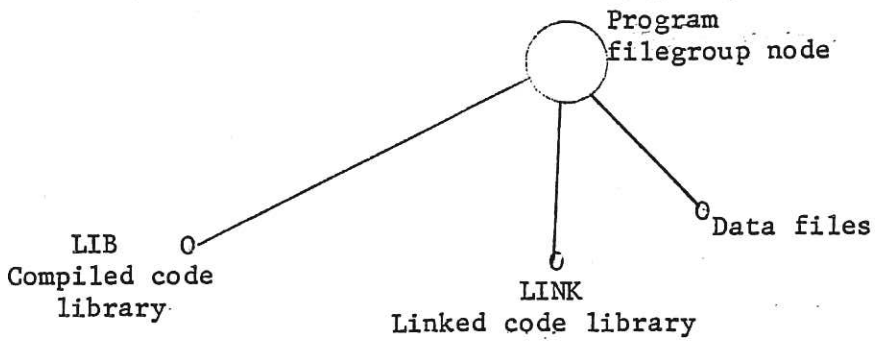
4.1 The Organisation Image - a VME/B Overstructure

Before describing the structure of the user image a short digression on the support of an organisation image by VME/B is necessary. The organisation image may be implemented by relationships within the VME/B catalogue above the user node level. At the highest level in the catalogue is an installation node. A hierarchy of divisional/group and "real-user" nodes can then be set up representing the organisation structure. The importance of this mapping is that it allows computer resource allocation, budgeting and privacy to reflect the existing organisation structure. At Culham, for example, resources are allocated by projects and this is reflected within the Culham VME/B catalogue by the allocation of file space to projects. Whenever a user runs a job the job will run for a particular project, and will use the resources owned by that project. However, this function of VME/B is not directly part of the user image which is primarily concerned with the computing facilities made available to the user rather than the organisational structure within which work is performed.

4.2 The User Filestore Structure

The user image has to be based within a firm context which defines the objects which are to be manipulated and the relationships between them. The first object which was selected was the program. This was envisaged as the code which was needed, associated with data files used when the code was executed. In general each user would have a number of programs which he would wish to keep separate. The user filestore was, therefore, divided into a number of programs. In VME/B terms a program was mapped onto a file group. Within each program there was a requirement for storing object code and the standard VME/B method of storing object code directly in a library was chosen. At this point the decision was taken that each program would only need one library for storing compiled code and a standard name, LIB, was chosen. The FORTRAN compilation macros would assume that this library would be used though provision was made for an alternative library to be nominated. Although there is no strict requirement for compiled modules to be collected before a program is

executed it was further decided that users would wish to use the VME/B Collector to produce "linked" modules. To avoid problems with common entry names to compiled and linked modules the linked modules would be kept in a second library, known as LINK. The overall structure for a program was as follows:



Within libraries the following conventions were observed:

a) Compiled code was kept with each routine in a separate library file module. Only the latest generation of each library file module was kept.

b) When a program was collected all routines in the LIB library were collected into a single linked module, known as CCM (for Culham Collected Module) in the LINK library. More than one generation of linked module could be kept.

The formality of the conventions is an advantage in constructing the user image. In the first instance the adoption of standard naming conventions decreases the parameters which need to be specified by the user. Also the formality should help to establish a mode of working for new users which will be beneficial to the supporting staff. The possible disadvantage is that flexibility is reduced, and users, as they develop, will be forced to use the more complex standard ICL macros. In the current implementation of the user image some care has been taken to ensure that flexibility is preserved and the possibility of users requiring some standard ICL macros on special occasions has been envisaged.

Besides forming part of the file structure the LIB and LINK libraries form an integral part of the "library list". This concept is fundamental to the way in which VME/B is structured. Associated with each virtual machine is a hierarchic library list containing all the routines available within that machine. At the lowest level is the standard ICL library constituting the TUM interface. Above that level are libraries of compiler support routines and standard subroutines such as the NAG library. At the top of the list are the libraries containing the program code which is about to be executed. In standard SCL extra commands have to be given to extend the library list to include the user program libraries. In the Culham user image this is performed automatically for the LIB and LINK libraries.

4.2.1 Filetypes Supported in the User Image

The filetypes supported by the user image are currently restricted by the limited batch usage which is planned for the introductory service. The full user image will encompass many more of the structures which are available under VME/B but this will not be available until interactive access to the 2976 is implemented.

The first filetype is compiled code in OMF (Object Module Format) which is held in the LIB library. These files are known as modules. Secondly compiled code for a program is collected into a single file (also in OMF) held in the LINK library. These files are known as linked modules. The two libraries, LIB and LINK, are set up for each program. Finally, program data files are held as individual files (ie not within libraries) and may be of two types. Serial data files are held in variable length spanned record format. This means that the maximum record size for these files is a user option. Direct access data files with a record size up to 2046 bytes are supported. This limit is a function of the uniform pre-formatting of allocated file space into 2 Kb blocks. This is an installation decision and VME/B will allow allocated file space to be set up with larger blocks. If the need arises for a larger block size the user image will be extended to support this. Multiple generations of both serial and direct access data files are supported.

The above filetypes are the only permanent filetypes supported by the user image. FORTRAN source text and text data files with a maximum record size of 80 bytes are transported to the 2976 as 'in-line' data with the SCL commands (in VME/B terms this is known as "alien data"). Such files only exist for the duration of the user job. Similarly output files containing line printer output may be created. These 'spooled' files are also temporary and disappear after printing. These temporary filetypes exist within a generally available filespace owned by the system.

Finally a spool file type associated with a special data transport is supported for copying serial text files with a maximum record size of 132 bytes from the 2976 to the PRIME.

4.3 Facilities of the User Image

Two basic modes of running FORTRAN programs are supported by the user image. The first mode, known as test mode, is primarily designed for small programs which are to be run once before further modification. The FORTRAN text is compiled directly into virtual store, rather than a library module.

When the program is run the virtual store code is executed. The user cannot collect modules in test mode. Files associated with test mode programs are catalogued immediately below the user node. In the current implementation these files are permanent. A future modification will be to set up these files as temporary files so that a run of a program in test mode will have no permanent effect on the user filestore.

The second mode of working is program development mode where the LIB and LINK libraries are available within a specified program file group. These two libraries are placed at the top of the user's library list so that programs will be loaded from them when a call is made. The user has the option of executing a program directly from compiled modules which will be cascade loaded from the LIB library or by setting up a single linked module which will be loaded from the LINK library. In both cases further libraries may be specified and routines will be loaded as required,

The various macros available in the initial user image will now be briefly described. A specification of each macro is given in Appendix 3. The macros are grouped by functions.

a) Virtual Machine handling

JOB

The JOB macro contains the username for the user running the job and the project under which the job is to be run. The users job is then connected to the filestore associated with that project. The virtual machine profile for the job may also be selected

ENDJOB

This macro terminates the control statements for a job.

b) Program handling

PROGRAM

This macro is used to select either test mode or program development mode. In the latter case a program file group name must be specified. For a new program file group the option NEW=YES must be specified. This macro ensures that the LIB and LINK libraries are present. If not these libraries will be created. The libraries are then put on the library list. The file and library contexts are extended to the program file group node.

If the user wishes to handle several program file groups in the same job then the statements applying to each file group should be enclosed in BEGIN and END statements.

DELETE_PROGRAM

All files and libraries associated with the specified program file group will be deleted.

CHANGE_PROGRAM_ACCESS

All files and libraries within the program file group may either be accessible to all other users for read and execute access or will be inaccessible.

c) Compilation, collection and execution

FORTRAN

The FORTRAN source, which usually follows the macro as alien data, will be compiled either into the LIB library or virtual store. This macro supports the System 4 and IBM H dialects of the FORTRAN language supplied by the FORTRAN(G) compiler and the non-optimised and optimised versions of F1 FORTRAN supplied by the F1 and OFC compilers. In the initial release of software at Culham the F1 compilers will not be available. There still exist some incompatabilities in input/output handling between FORTRAN(G) and F1. When these are resolved the compilers should be fully compatible in the sense that compiled modules can be freely mixed.

LINK

This macro collects the modules in the LIB library into a single linked module in the LINK library. Other nominated libraries may also be searched and specific modules may also be included. Several generations of the linked module may be kept to allow fall back if a new version of a program does not work.

CALL

This macro starts execution of a program. The program is identified by the specification of an entry point which will be in a library (normally LIB or LINK) on the user's library list. The module, or linked module, satisfying the entry point is loaded and then any other routines required will be cascade loaded. There is an option allowing further libraries to be added to the library list for scanning, and the scanning procedure can also be specified. Other options set diagnostic and trace mechanisms.

d) File specification

DEFINE

This macro is the basic building block for data file types. Whenever other macros set up data files (such as OUTPUT) a call is made to DEFINE.

Serial and direct access data files are supported. One feature of the implementation is that DEFINE checks that a new generation of an existing file has the same filetype and characteristics as the existing file. This is a usability feature to ensure that files are uniform and helps with support of the interface.

A file to be directed to a specific file on the PRIME may also be set up. This is a spool file which will be directed to the PRIME data transport.

Compiler and collector listings can also be directed to nominated files using this macro. It is expected that this will usually be used to send such listings to the PRIME.

DATA

Input data for FORTRAN programs is assigned to a specific FORTRAN dataset by this macro. The data may be contained in a nominated file or follow the macro as alien data (the alien data is copied into a temporary file for subsequent access). If the FORTRAN dataset is not specified the file will be attached to dataset 5 or dataset 97.

INOUT

An input/output file may be set up and attached to a specific FORTRAN dataset. A file set up by this macro is not spooled for printing.

OUTPUT

This macro assigns an output file to a specific FORTRAN dataset. The file may be spooled to a line printer. If the FORTRAN dataset is not specified the file will be attached to dataset 6 or dataset 99.

e) File handling

COPY

A nominated VME/B file, or alien data following the macro, is copied to a nominated VME/B file or to a file on the PRIME. The underlying file support provided by DEFINE is used to ensure that a new generation of an output file has the same filetype and characteristics as previous generations.

DELETE

The specified VME/B file is deleted. An option exists to allow the latest generation and all previous generations to be deleted.

PRINT

The specified file is copied into a spool file for printing. The file may then be reused or deleted. This method of printing is similar to that used on the PRIME. The standard ICL macros place on a spool queue a request for the file to be printed without copying.

COPY_MODULE COPY_LINKED_MODULE

The macros, COPY_MODULE and COPY_LINKED_MODULE, copy modules into the LIB and LINK libraries for the current program. These macros can only be used in named program context.

DELETE_MODULE DELETE_LINKED_MODULE

These two macros delete modules from the current LIB and LINK libraries and can only be used in named program context.

f) File listing

LIST_FILES

This macro lists the names of all the files belonging to the user. An option is available to allow names of files within libraries to be listed.

LIST_FILES_OF_ PROGRAM

This macro lists the names of files within the current program file group or belonging to the named program file group. An option is available to allow names of files within libraries to be listed.

4.4 Usability and Housekeeping Features

Part of the purpose of the user image is to improve usability for non-computer specialists over the standard ICL macros. The various improvements are described below:

Removal of special names

To use the ICL macros the user needs to know several special names, such as ICL9CEMAIN. This is made more difficult by the fact that these names are different for F1 FORTRAN and FORTRAN(G). The user image supports these names implicitly and the user need not be aware of their existence.

Support of standard libraries

The PROGRAM macro automatically sets up the LIB and LINK libraries and will ensure that these libraries are present. The libraries are then placed on the library list.

Support of default input/output channels

A user program which reads from channel 5 (or 97) and outputs to channel 6 (or 99) does not need specific assignment of these channels. Also if channels 6 or 99 are not used for output the spool file prepared for output is automatically deleted so that the user does not receive a blank output file.

Uniform file type maintenance

As mentioned previously a check is performed that successive generations of a file have the same file type and characteristics.

Simpler and more user directed error messages

The user image goes some way to trapping the more complex error messages which may arise for the underlying VME/B software and producing its own, user directed error messages. In contrast to the standard ICL messages which are upper case only, both upper and lower case are used. This feature has implications for the support team as user image messages can be immediately distinguished from (possibly unexpected) VME/B messages.

5. Implementation of the Batch User Interface

Implementation of the batch user interface was accomplished by the design of a structure containing S3 procedures and SCL macros. These could be arranged into five subsystems described by Holon trees. The main categories were:

Holon number	Holon name
GA1000	GENERAL_SERVICES
GA2000	ASSIGN_FILE_SERVICES
GA3000	PROGRAM_SERVICES
GA5000	FILESTORE_SERVICES
GA6000	LANGUAGES

A complete description of these subsystems, itemised at component level, is given in Appendix 2. The function of the many service routines is to give comprehensive and uniform support for the underlying objects chosen for the interface. This has meant that VME/B facilities are obtained via these support procedures so that there is control throughout the interface. In some cases this meant that a support procedure would map closely onto a standard procedure. The existence of the support procedure would, however, allow modification or enhancement at a later stage in a uniform manner. The function of each subsystem will be briefly described.

GENERAL_SERVICES

These routines can be divided into four major categories. The fundamental basis of the user image is the approach to file handling and routines were provided to give the characteristics of objects held in the VME/B catalogue. The aim was always to be able to validate user-supplied objects within the context of the user image. This was seen as essential

if the user was to be guaranteed meaningful error messages. A particular example is the proper handling of protection error messages where standard ICL procedures may produce incomprehensible messages.

A second category of routines handled the creation and deletion of spool files. These were used for line-printer output and as a means of transporting files to the PRIME. Deletion of a spool file was necessary where a default output file had been created and not then used by a program.

One routine provided an interface for parameterised error messages. It is useful to include the name of an object which has been incorrectly specified in the text of an error message.

There were some routines of general utility for conversion and validation of literals and integers.

ASSIGN_SERVICES

These routines were based upon a global table containing file assignments to channels. The channels corresponded to FORTRAN datasets and were represented by integers apart from the READ, PRINT and PUNCH datasets of F1 FORTRAN. When a program was run the necessary job space variables would be created and assigned to the appropriate currencies. Though the existence of the table was global the currencies could be invalidated by the use of SCL block structure so that these had to be checked before assignment.

Entries in the table were made by calls from the DATA, OUTPUT and INOUT macros. Besides assignment these macros handled file creation using the underlying support procedures of DEFINE.

PROGRAM_SERVICES

These routines divided into three categories. Procedures underlying CALL handled the loading of a user program. Extra libraries could be added during the loading procedure and the loader searching algorithm could be specified. A feature of the implementation was that a default program entrypoint could be selected if a linked module existed in the LINK library. The CALL procedures were also invoked at the finish of a program run to remove unused default output files.

The LINK procedures provided a simple interface to VME/B collector. All modules in the program LIB library were collected into a single module, known as CCM, in the LINK library. Further modules and libraries could

be included in the collection. The implementation ensured that if an entryname in an additional module clashed with an entryname in the LIB library the latter would be superseded.

SCL macros handled the copying of modules in the LIB and LINK libraries using the standard utility COPY_LIBRARY_TO_LIBRARY.

FILESTORE_SERVICES

The DEFINE procedure and its underlying support procedures formed a basis for the user's file specification. The DEFINE procedure set up a new generation of an existing file provided that the characteristics of the file matched those of previous generations (if these existed). This was done by calling the procedure SELECT_PREVIOUS_GENERATION and then calling MATCH_DESCRIPTIONS. The check performed was that the filetypes (ie SERIAL/DIRECT) were the same and that the record size for DIRECT files was the same.

The PROGRAM procedure supported the underlying file group structure and set up the basic program environment including access to the LIB and LINK libraries. These two libraries would also be created if the specified file group node existed but they were absent.

Procedure PRINT copied a 2900 file to a spool file for listing on the line printer. Copying was at the record level to enable spanned record format to be changed to standard variable length format. The filename of the copied file was placed in the spool queue entry for inclusion on the banner page of the listing. A specified number of copies of the listing could be produced. Optionally the file to be printed could be deleted after the spool file copy had been made.

Other filestore services were provided by SCL macros. COPY could be used to copy VME/B files. The output file was set up using DEFINE procedures to ensure uniformity of successive file generations. Serial VME/B files could also be copied to spool files for transfer to nominated PRIME files. Deletion was based on standard ICL macros DELETEFILE and DELETE_GROUP. A check was required that the user did not attempt to delete a current program file group. A simple filestore protection facility was based on CHANGE_GROUP_ACCESS. Finally two macros gave file listing facilities. These were based upon LIST_FILES_OF_USER and LIST_FILES_IN_GROUP.

LANGUAGES

Only one programming language, FORTRAN, was supported by the initial system. Both the System-4 and IBM extended H dialects of FORTRAN could be compiled. The definition of the macro allowed eventual extension to support of ICL F1 FORTRAN. In program context the FORTRAN macro would produce OMF files in the LIB library. In test mode the compiled code was set up on virtual store.

The DEFINE procedure could be used to set up a listing file for compiler output. This was detected in the FORTRAN macro by the existence of a job space variable containing the file currency.

6. Further Enhancement of the 2976 User Image

The interface described in this report was prepared for the initial batch usage of the Culham 2976. Developments of this interface will continue over the next two years. The main areas which have to be covered are the inclusion of the scientific and management aspects of the laboratory workload to be serviced by the 2976 in its role as part of the new computing system. This includes:

a) Interactive use of the 2976 for engineering design application

b) Management program development and production based on COBOL and IDMS

c) Scientific database development

d) Further enhancement for batch processing of the base workload

e) 'Real time' processing of experimental results.

Other computing systems at Culham provide flexible interactive facilities, communications and word processing.

The guideline for future developments will be the same attention to simplicity and consistency which has been built into the initial interface. To illustrate this examples of use of the initial interface are shown in Appendix 1.

Acknowledgements

The initial batch user interface has been developed as a joint project involving the Culham Computer Systems and Services Group and the on-site ICL Project Team. Mr R Thomas (ICL) was responsible for the design of the holon structure shown in Appendix 2 and a substantial proportion of the implementation. The initial interface is closely linked to the design of PRIME Culham commands initiated by Dr R Dakin (CSIRO, Commonwealth of Australia) and continued by Mr D Fox. Finally the author wishes to acknowledge the work on a VME/B user interface produced at the South West Regional Computer Centre under the guidance of Mr M Thomas which has substantially influenced the design in its usability aspects.

References

1. CADES - Support for the development of complex software; G D Pratten and R A Snowdon. Eurocomp 76.

2. VME/B SCL Vocabulary, ICL publication TP6500

APPENDIX 1

Examples of use of Culham SCL

1. A test run of a single module program. The program is compiled and then run using input for FORTRAN Dataset 5. Output on Dataset 6 is sent to the 2976 printer. A compiler listing and job journal will also be printed.

```
JOB(:KTCEXM.EXAMPLE1)
PROGRAM
FORTRAN
----
    REAL*4 A,B,C(10)
     .
     . .
     .
     :
     :
    END
++++
CALL
----
    1.0   1.35   1.03   2.65
++++
ENDJOB
****
```

2. A job which compiles three routines into a program library. The program is called ENERGY_BALANCE and is a new program (ie the filestore structure for that program does not already exist)

```
JOB(:KTCEXM.EXAMPLE2)
PROGRAM(ENERGY_BALANCE,NEW=YES)
FORTRAN
----
    SUBROUTINE SUB1
      .
      .
      .
    END
    SUBROUTINE SUB2
      .
      .
      .
    END
    SUBROUTINE  SUB3
      .
      .
      .
    END
++++
ENDJOB
****
```

3. A job to compile a main routine in program ENERGY_BALANCE and to run the job returning the results to a PRIME file.

```
JOB(:KTCEXM.EXAMPLES3)
PROGRAM(ENERGY_BALANCE)
FORTRAN
----
      INTEGER L(10)
       .
       .
       .
      END
++++
DATA(CHANNEL=16)
----
INITIAL RUN1
1.7   5.9   16.7
++++
OUTPUT(KTCEXM>ENER>RESULTS1,TARGET_SYSTEM=PRIME)
CALL
ENDJOB
****
```

4. A job to set up a linked module for program TORUS_STABILITY and then run the program using a Direct Access Data file for FORTRAN Dataset 12. A second program GRAPHS is then run to read the Direct Access file. The programs are separated into separate SCL blocks using BEGIN and END so that the library list extensions and contexts set up for the first program are completely removed before the second program starts.

```
JOB(:KTCEXM.EXAMPLE4)
BEGIN
PROGRAM(TORUS_STABILITY)
LINK
DEFINE(INDEX,TYPE=DIRECT,RECORD_SIZE=80,MAX_SIZE=10)
INOUT(INDEX,CHANNEL=12)
CALL
END
@ Run second program using INDEX as data @
BEGIN
PROGRAM(GRAPHS)
DATA(TORUS_STABILITY.INDEX,CHANNEL=11)
OUTPUT(CHANNEL=9)
CALL
END
ENDJOB
****
```

5. A job to perform some housekeeping for program ENERGY_BALANCE. The program is made generally available, a file is deleted, another file is printed and a file is copied to a PRIME file. Finally all files belonging to the user are listed in the job journal.

```
JOB(:KTCEXM.EXAMPLE5)
PROGRAM(ENERGY_BALANCE)
CHANGE_PROGRAM_ACCESS(ACCESS=READ)
DELETE(RESULTS1,ALL=YES)
PRINT(DATA2)
COPY(RESULTS2,KTCEXM>ENER>RES2,TARGET_SYSTEM=PRIME)
LFS(HIGH)
ENDJOB
****
```

APPENDIX 2

The holon trees indicate the type of object supporting the interface. In particular:

above IL : A conceptual holon which controls a number of real, implemented holons.

SCL : An SCL macro.

SIM : A Simple Procedure written in S3 and declared in the controlling holon.

GSP : A Global Static Procedure written in S3.

Holon Tree for UKC9GB1000 SERVICES

```
GB1000
SERVICES   (above IL)
           GC1100
           CHECK_FILE_NAME        (GSP)
           GC1200
           CREATE_SPOOL_FILE      (GSP)
           GC1300
           DESTROY_SPOOL_FILE     (GSP)
           GC1400
           GIVE_FILE_WRITTEN_SIZE (GSP)
           GC1500
           REPORT     (GSP)
           GC1600
           CONVERT_INT_TO_CHAR        (GSP)
           GC1700
           VALIDATE_LITERAL       (GSP)
           GC1800
           SELECT_OBJECT          (GSP)
           GC1900
           EXTEND_LIBRARY_LIST        (GSP)
           GC1A00
           READ_OBJECT_LIBRARY_CURRENCY   (GSP)
           GC1B00
           CHECK_FILE_TYPE        (GSP)
                    GD1B10
                    REPORT_INVALID_TYPE   (SIM)
           GC1C00
           GIVE_FILE_CHARACTISTICS    (GSP)
```

Holon Tree for UKC9GA2000 ASSIGN_FILE_SERVICES

GA2000
ASSIGN_FILE_SERVICES (above IL)

GB2000
FILE_TABLE_SERVICES (above IL)

GC2100
ASSIGN_DEFAULT_CHANNELS (GSP)

GD2110
SEARCH_FILE_TABLE (SIM)

GD2120
DEFAULT_OUTPUT_CHANNEL (SIM)

GD2130
DEFAULT_INPUT_CHANNEL (SIM)

GC2200
ASSIGN_FILE (GSP)

GC2300
RESET_FILE_TABLE (GSP)

GC2400
WRITE_LOCAL_NAMES (GSP)

GD2410
CONSTRUCT_LOCAL_NAMES (SIM)

GD2420
PROPOGATE_CHANNEL (SIM)

GC2500
TIDY_SPOOL_FILES (GSP)

GC2600
ASSIGN_OUTPUT_FILE (GSP)

GB2100
DATA (SCL)

GB2200
OUTPUT (SCL)

GB2300
INOUT (SCL)

GB2400
APPEND (SCL)

Holon Tree for UKC9GA3000 PROGRAM_SERVICES

GA3000
PROGRAM_SERVICES (above IL)

GB3000
CALL (GSP)

GC3100
GCL_LOAD (SIM)

GC3200
LOAD_DEFAULT_PROGRAM (SIM)

GC3300
LOAD_DEFAULT_TEST_PROGRAM (SIM)

GC3400
CONSTRUCT_TRACE_DATA (SIM)

GC3500 (SCL)
RUN

GB3100
LINK (GSP)

GC3110
TEST_LINK (SIM)

GC3120
PROGRAM_LINK (SIM)

GD3121
ADD_COLLECTOR_DIRECTIVE (SIM)

GD3122
CONSTRUCT_COLLECTOR_DIRECTIVE (SIM)

GD3123
PROCESS_INPUT_MODULE (SIM)

GC3124
PROCESS_SCAN_LIBRARY (SIM)

GB3200
COPY_MODULE (SCL)

GB3300
COPY_LINKED_MODULE (SCL)

GB3400
DELETE_LINKED_MODULE (SCL)

GB3500
DELETE_MODULE (SCL)

Holon Tree for UKC9GA5000 FILESTORE_SERVICES

```
GA5000
FILESTORE_SERVICES  (above IL)
        GB5000
        PRINT       (GSP)
        GB5100
        PROGRAM     (GSP)
            GC5110
            CREATE_TEST_ENVIRONMENT  (SIM)
            GC5120
            CREATE_PROGRAM_ENVIRONMENT  (SIM)
                    GD5120
                    CHECK_PARAMETER_NOT_NULL  (SIM)
                    GD5121
                    PROCESS_FILE_GROUP_NODE  (SIM)
                    GD5122
                    PROCESS_OBJECT_LIBRARY  (SIM)


        GB5200
        DEFINE    (GSP)
            GC5210
            DESCRIBE_FILE  (GSP)
            GC5220
            SELECT_PREVIOUS_GENERATION  (GSP)
            GC5230
            CHANGEFILE_ACCESS  (GSP)
            GC5240
            MATCH_DESCRIPTIONS  (GSP)
            GC5250
            CREATE_FILE  (GSP)
```

GB5300
COPY (GSP)

GB5400
DELETE (SCL)

GB5500
DELETE_PROGRAM (SCL)

GB5600
CHANGE_PROGRAM_ACCESS (SCL)

GB5700
LIST_FILES (SCL)

GB5800
LIST_FILES_IN_PROGRAM (SCL)

Holon Tree for UKC9GA6000 LANGUAGES

GA6000
LANGUAGES
    GB6000
    FORTRAN (SCL)

The definition of the user interface is available on request as a separate document.

The following macros are defined.

CALL
CHANGE_PROGRAM_ACCESS
COPY
COPY_LINKED_MODULE
COPY_MODULE
DATA
DEFINE
DELETE
DELETE_LINKED_MODULE
DELETE_MODULE
DELETE_PROGRAM
ENDJOB
FORTRAN
INOUT
JOB
LINK
LIST_FILES
LIST_FILES_OF_PROGRAM
OUTPUT
PRINT
PROGRAM

HER MAJESTY'S STATIONERY OFFICE

Government Bookshops

49 High Holborn, London WC1V 6HB
13a Castle Street, Edinburgh EH2 3AR
41 The Hayes, Cardiff CF1 1JW
Brazennose Street, Manchester M60 8AS
Wine Street, Bristol BS1 2BQ
258 Broad Street, Birmingham B1 2HE
80 Chichester Street, Belfast BT1 4JY

Government publications are also available through booksellers