UNIVERSITY COMPUTER CENTER

NOTES AND COMMENTS

FEBRUARY, 1969 UNIVERSITY OF MINNESOTA
Volume 3, Number 5 MINNEAPOLIS, MINNESOTA

HYBRID COMPUTER LABORATORY

Dr. Stephen J. Kahne has been appointed director of the Hybrid Computer Laboratory effective December 15, 1968. Dr. Kahne is an Assistant Professor in the Department of Electrical Engineering and has been actively engaged in work involving the hybrid computer since it arrived on campus in 1967. The hybrid computer is now located on the first floor of the Space Science Center and employs a total of 11 people. It is run on an open shop basis and currently has projects underway from the Departments of Mechanical Engineering, Electrical Engineering, the Center for Control Science, The Space Science Center, Physiology, Chemical Engineering, Honeywell, and others. The machine is being operated on a two shift basis and is currently available on an open shop mode, free of charge to University users. Plans are underway for connection of the equipment to various laboratories throughout the University. A connection with the University Computer Center's 6600 is being planned.

Activity in the area of hybrid computation at the University is increasing at a fast rate due to the flexibility of the machine.

ELECTRON BEAM RECORDER

Film Labeling

Before beginning processing the input file, MF501 puts out tape information with account and bin numbers for film identificatior (See UCC Notes & Comments, Vol. 2, No. 6.) This film labeling facility can also be used by those who wish to produce their own EBR tapes without using MF501.

The calling procedure is as follows:

MF501(N,M)

where

N = empty file (i.e., your tape identification)
M = SNXXX
= TTXXXXX

If you desire the leader account number, site, bin number at the site, and date information to be the leading microfilm frames when tape M is run on the EBR, M must be in this form since MF501 generates its own internal tape request and the operator sees this request as "REQUEST M" (without comments). In addition, a tape request slip must be attached to the job with EBR SNXXX or EBR TTXXXXX in the tape name field.

After the return from MF501 the tape is positioned to receive information from the user's program.

Example:

This is a job deck to put EBR information on tape SN216:

```
JOBFILM,T200,CM60000.12345678,1
MF501(TAPE3,SN216)      (generates leader on tape SN216)
RUN(S)
LGO(SN216)              (generates EBR output on SN216)
7
8
9
      PROGRAM FILM(TAPE1,INPUT,OUTPUT)
      :
      :
```

6600 REMOTE FACILITIES

Biomedical

The University Computer Center has added a third IMPORT/EXPORT terminal. This terminal is the Biomedical Computing Facility and will run as a background job to the MASTER operating system. Persons interested in using this facility should contact Biomedical Computer Services for further information.

West Bank

1. Access to rooms: All West Bank rooms containing data processing equipment will be locked during the hours that the buildings are locked. Rooms 55, 86, and 90 will be locked at 11 PM and reopened at 7 AM the following morning.
2. Driver schedules: An additional pick-up/delivery stop is now made at the West Bank. Stops are now made at 10 AM, 2 PM and 6 PM.

6600 SYSTEM CHANGES

1. PP time is no longer incremented indefinitely until operator assignment of tapes on REQUEST cards.
2. PP time is no longer charged against the control point for ROLLOUT functions.

CHIPPEWA DECEMBER COMPILER

Use of the Compiler

Benefits will be harvested for jobs that fit available core if the DECEMBER compiler compile-to-core options are used. Memory is more efficiently used, because of field length compression, and better throughput is realized, because disk references to load object programs from disk are absent.

The DECEMBER compiler may be used by the following compiler control cards under SCOPE:

1. Compile-to-core, accelerated mode or A-Mode

   QXX(RUNDEC,A)

   or

   QXX(RUNDEC)

   System actions:

   a. If there are no compilation errors, object program is compiled to core by over-writing the compiler, unused memory is automatically returned to the system, and the object program is entered for execution. No printing of source program occurs.

   b. If Fortran compilation errors are encountered, error indications, source card in error, and program or subroutine identification will be printed and any further job processing will be terminated.

2. Compile-to-core, dump or print mode...D-mode

   QXX(RUNDEC,D)

   System actions:

   a. If there are no compilation errors, object program is compiled to core by over-writing the compiler, unused memory is automatically returned to the system, and the object program is entered for execution. Printing of source program occurs.

   b. If Fortran compilation errors are encountered, error indications and the entire source program will be printed. Any further job processing will be terminated.

3. RUNDEC users should note that the UOFM subroutine library is unavailable when using this compiler.

4. If A or D mode is not used, binary is written on a file that has the same name as the main program name; consequently, the loading must be accomplished by naming the file ≠PROGRAM NAME≠

Examples:

```
JOB.......
QXX,RUNDEC,S.
QXX,WRF.
7
8
9
      PROGRAM WRF(INPUT,OUTPUT)
      :
      :
      END
7
8
9
DATA
6
7
8
9
```

Note that by using LGO as a program name, less confusion arises:

```
JOB......
QXX,RUNDEC,S.
QXX,LGO.
7
8
9
      PROGRAM LGO
      :
      :
      END
7
8
9
6
7
8
9
```

Fortran Control Card for RUNDEC

The Fortran compiler is called by the control card

```
QXX(RUNDEC,cm,fl,d,bl,if,of,fb)
```

where

cm = compiler mode option (if omitted, assume G)

- A compile-to-core, accelerated mode (no source listing)
- D compile-to-core, dump mode (print source program)
- G compile and execute
- S compile, no execute
- P compile and punch, no execute
- L compile and list, no execute
- C chain mode
- B batch mode
- M multiple mode-compiling is done as in batch mode, except octal versions of object programs, segments, and subroutines are produced for listing
- I incomplete mode

fl = object program field length (octal); if omitted, it is set equal to the field length at compile time, 35300 (octal) for compiler and constants.

d = object program common length (octal); if omitted, it is set equal to the amount of common storage required for the main program being compiled.

bl = object program I/O buffer lengths (octal); if omitted, assumed to be 1001 (octal).

if = file name for compiler input, if omitted, assumed to be INPUT.

of = file name for compiler output; if omitted, assumed to be OUTPUT.

fb = line-limit (octal) on the OUTPUT file of an object program. If not specified, it is set to 10000. If the line count exceeds the specified line limit, the job is terminated.

In if and of, the file name may be followed by an equal sign and an octal constant to indicate a new buffer length. The length is normally 1001_8 words. Minimum field length is 42000_8.

"Free Form" PARAMETERS FOR THE FUN CONTROL CARD

The parameters on the FUN card may be either in the fixed position form as in the past or in the new free form. The existing FUN call card has followed the RUN 2.x pattern which, with the addition of the rounded floating point parameter as the tenth parameter and the execute option as the eleventh parameter, is as follows:

FUN(Mode,OFL,Bl,I,O,B,NL,A,C,R,X)

If the R parameter is non-zero, rounded floating point operations are generated.

The new free form FUN card format is:

FUN(P1,P2,P3,P4,P5)

where the parameter list may contain any combination of the five following parameters:

1. Compile mode parameter

   This parameter may consist of any combination of the five following characters:

G
L
M
P

   ACRX

where G, L, M, P, and S are as defined for the RUN 2.x mode parameter (CDC 6600 Fortran Reference Manual, 60174900B, Appendix F) and,

- A selects the ASA code option
- C selects the cross reference table option
- R selects the rounded floating point option
- X selects the execute mode option. This gives the programmer control over how the remainder of his job will be processed after one or more fatal errors have occured during compilation. The FUN compiler has always terminated itself with an END call. Some user's would like the compiler to abort the job without calling the loader while others want it to go ahead and load and execute the binary file as is. If the compiler aborts on fatal errors, a secondary problem comes up. The programmer may have included an EXIT card followed by a DMP card which will dump all of core, presumably in case of error during execution. However, if the compiler terminates with an ABT, the programmer gets a dump of the compiler which promptly goes into the waste basket. To (hopefully) solve these problems, the X option has been added. This mode will cause the compiler to terminate normally even when fatal errors have occurred.

If any parts of this parameter are omitted, G, $\overline{A}$, $\overline{C}$, $\overline{R}$, and $\overline{X}$ are assumed.

2. Number of print lines parameter

   NLxxxxxxxx

   where xxxxxxxx is an octal number which specifies the maximum number of print lines. If omitted, NL10000 is assumed.

3. Source input file parameter

   I=fn

   where fn is the file name for the compiler input. If omitted, I=INPUT is assumed.

4. Listing output file parameter

   O=fn

   where fn is the file name for compiler listing output. If omitted, O=OUTPUT is assumed.

5. Binary output file parameter

   B=fn

   where fn is the file name for compiler binary output. If omitted, B=LGO is assumed.

As an example, the call card FUN(LRC,O=PRINTF) will cause the compiler to generate a full octal listing, generate rounded floating-point operations, generate a cross reference table, read its input from file INPUT, write its output on file PRINTF, and write its binary output on file LGO. If fatal errors occur during compilation, the compiler will abort the job.

The object field length and buffer length parameters were omitted from the free form because for the former, the UCC has incorporated automatic field length compression into the loader, and for the latter, buffer lengths can be specified individually for files on the PROGRAM card. In any case, if these parameters are desired, the fixed form can be used.

BLOCKED BINARY INPUT/OUTPUT

Job throughput is increased by blocking binary I/O files. Several logical records are transmitted to or from the file buffer with each access to an I/O device; and the total number of accesses depends upon the number of peripheral unit records (PRU's) necessary to contain the records instead of the number of records in a file.

Each logical record with non-blocked binary I/O files requires at least one access to an I/O device. Since I/O is performed in terms of PRU's, the beginning of each logical record must correspond to the beginning of a PRU. Any space between the two is not used. Several logical records with blocked binary I/O files may be transmitted with one access to the I/O device. Records are packed into blocks and I/O is performed in terms of the PRU's comprising these blocks. There is no correspondence between records and blocks, and no space in a block is left unused. Each record immediately follows the preceding one and can begin anywhere in a block. A record can be continued from one block to the next or extend across several blocks depending on its length and position in the file.

Library Subroutine FTNBIN

In connection with the blocked binary I/O capability, the subroutine FTNBIN has been added to the Fortran Library to allow the user to use binary blocking. All files are assumed to be non-blocked unless declared otherwise by calling the following subroutine. The calling sequence is as follows:

```
CALL FTNBIN(i,n,array)
```

i indicates the mode for files specified by the parameters n and array. If i=0, the files are in non-blocked I/O mode. If i=1, the files are in blocked I/O mode.

n — n indicates the number of files to be processed in the mode specified by i. If n=0, all files are processed in the mode specified by i.

array — Integer array containing logical I/O unit numbers of all files to be processed. Subroutine FTNBIN will use n words of array to over-ride the installation options.

Examples:

1. CALL FTNBIN(1,0) -- All binary files are blocked

2. CALL FTNBIN(0,0) -- All binary files are non-blocked

3.
```
PROGRAM XYZ(INPUT,OUTPUT,TAPE1,TAPE2,TAPE3,TAPE4)
DIMENSION IBBFILE(2)
DATA IBBFILE/2,4/
CALL FTNBIN(1,2,IBBFILE)
```

The binary files on logical I/O units 2 and 4 are blocked. Logical I/O units 1 and 3 maintain a non-blocked status.

This routine may be called at any time with a Fortran program; however, without special precautions, a file written in blocked mode cannot be read accurately in non-blocked mode; neither can a file written in a non-blocked mode be read accurately in the blocked mode. A blocked file contains block control words which preface each block in a user's logical record; they would not be removed if the file were read in a non-blocked mode. In general, the routine should be called prior to any attempt to do I/O on a file for which blocking is intended. This option is <u>not</u> available in conjunction with FTN programs.

Example 3 above describes the changes which will be necessary to use the blocking feature. In addition to the above, several of the system I/O routines require modification to incorporate the blocking feature. The routines available on the running system do <u>not</u> reflect those changes and will not be changed until further checkout of the binary blocking feature is possible. Users may access the necessary binary by a reference to the common file NEWCODE. A typical deck might be

```
JOB.......
RUN(S)
COMMON NEWCODE.
R,NEWCODE.          (rewind the file)
CBF(NEWCODE,LGO)
RETURNU(NEWCODE)    (MUST be included to release the file)
LGO.
```

If there are any questions or problems with using this routine, please see William Franta (210 Exp. Eng.) or Dennis Lienke (212 Exp. Eng.) for assistance.

FORTRAN EXTENDED

UOFM library subroutines all expect RUN or FUN calling sequences. Since Fortran Extended (FTN) subroutine linkage (the method of passing argument addresses) is different from RUN and FUN, users should not use UOFM routines directly from the system library. See a consultant or call Mike Frisch (373-5907) to obtain a copy of programs which have been compiled for use with the FTN compiler.

LIBRARY CHANGES

Changes have been made to the following routines:

CHSQ	LOCF$	RANDEV	SCLPLT	TIME$
DESCRIB	LRSHFT	RAN2F	SYMSOLU	TTEST
LOCF	OR	RAN3F	SYMSOLV	

The following new routines have been added to the library:

ICOUNT - Function, to count the number of bits in a word.

ORSYM - Subroutine, similar to OR, which uses a packed input matrix.

If you have any questions concerning the changes or the new routines, call or see Mike Frisch, extension 3-5907, room 212 Exp. Eng.

UCC SUBROUTINE WRITEUPS

The UCC staff is in the process of revising all subprogram writeups to conform to 6600 usage. To aid people in using the old writeups until revisions are completed, we are listing below a few of the changes that must be noted when using the routines on the 6600:

1. All subroutines require a RETURN card to return control to the calling program.
2. 6600 subroutines use EXTERNAL cards instead of F-cards.
3. Floating point DO-loops must be changed to integer.
4. The OUTPUT statement is not available on the 6600.
5. Only one END card may be used.
6. Input/Output files must be used on PROGRAM cards.
7. The 6600 uses 14-digit constants.

If you have any questions concerning the available subroutines, and the writeup applies to the 1604 only, please see a consultant before attempting to use the routine on the 6600.