UNIVERSITY OF MINNESOTA COMPUTER CENTER
Deadstart Systems Newsletter

8 January 1980 Vol. 6, No. 1

Send all comments, criticisms and contributions to the editor: T. W. Lanzatella
University Computer Center, 2520 Broadway Drive, Lauderdale, MN 55113.
The University of Minnesota is an equal opportunity educator and employer.

TABLE OF CONTENTS

NOTICE OF CHANGES TO THE SYSTEM

Kevin Matthews repaired an error in his overlayed MTR feature which caused the clock from being updated while a sysedit was in progress. Kevin also fixed a system-hanging problem in 1RI. The problem was that whenever a rollin file developed an error, 1RI hung. The error turned out to be the fault of a modset gleaned from the PSRs. Additionally, Kevin repaired a critical error in DELAY QUEUE code in UQM.

Don Mears installed the following changes.

1) Two places in TELEX suspected of causing dropped errors were fixed.

2) 1TD was repaired to not hang ports when an excessive number of ports are active when TELEX is brought up.

3) 1TO was repaired to not hang if TELEX requested too many pots to be dumped to the disk. 1TO was also repaired to not destroy a disk if a track limit is encountered.

4) The rotary check was finally converted from R3.

5) RUN command processing in 1TD was fixed so that if a FL beyond the TXOT service limit is computed, then the message OPERATOR DROP(?) is not issued. Instead, the users FL is set to the service limit. Similarly, if the FL calculated is greater than the users MFL but less than the service limit then the FL is reduced to the MFL.

6) Program 1CD was corrected so that account file messages are accurate when a page limit/punch limit is encountered. Previously, the messages were 4096 off when the limit was encountered.

7) CPM was repaired so that a MFL request causes the RFL value in the terminal table to be cleared.

Jeff Drummond contributed the following changes.

1) The ENQUIRE,N report was changed to include total logins and total ports.

2) The RELEASE macro in CPCOM was altered to produce an assembly error along with a directive to use DSP instead. Programs DFTERM, QREC, QMOVE, QLOAD, QDUMP, FNTLIST and PFLOAD were changed to use DSP.

3) MMF code in 1AJ was rearranged to save space.

4) Space saving changes were installed into DI5.

5) Program SET was altered to prevent the CMU from being turned off when turning off 1 cpu on a dual cpu machine.

Brad Blasing installed a new version of CCL. This version has no changes and was simply reassembled because of the TELEX subsystem changes.

Dean Nelson repaired a nagging problem in SUBMIT. The problem was that a /USER directive following a /NOSEQ directive was ignored.

Bill Sackett installed the following changes related to C720 default user number validations.

1) The default time sharing field length (TF) was increased to 61K.

2) CTLX is now on by default when creating a new UN.

3) CACC is now off by default when creating a new UN.

The abundant system time during the Christmas and New Year holidays this year was well used by Jeff Drummond and Don Mears. Between the two of them almost 30% of the code comprising NOS R5 was backed into our system. None of these changes constitute a feature. Ordinarily, changes like these are never mentioned here because they are part of the R5 base system. But, because they have been extracted and are being reinstalled as JPL modsets and because we are duty-bound to publicize all such changes we present here a somewhat pedantic and obscure list of changes.

Don installed the following modsets from NOS release 5.

KRA577$ - Prevent a write on a locked primary file.

KRA586$ - Prevent BATCHIO BKSPRU and SKIPRU commands from leaving print file positioned at the middle of a line.

KRA594$ - Repair an error in calculation of "first" pointer in 1TD.

KRA598$ - Fix TELEX wait completion processing.

KRA599$ - Fix TELEX DIAL command for short messages.

KRA603$ - Repair a test for transaction terminals in TELEX.

1RI4$ - Repair a problem where an S-key typed while a program is aborting causes a subsequent command to be ignored.

FNTLI1$ - Fix QALTER forms code processing.

Jeff installed the following modsets from NOS release 5.

RA539A$ - Fix error in 1MT where extending a multifile set sometimes causes a label missing error.

KRA588$ - A repair which ensures that a NORERUN job gets a dayfile if it was executing when a L3 deadstart was done.

KRA610$ - Ensure that a SKIPFB which encounters recovered parity errors on an EOF block actually works.

PCOM4B$ - Ensure that =1 macro in CPCOM will assemble correctly if base octal is in effect.

CPCOM7$ - Change FILEB macro to create 13D work FET when the LBL parameter is used.

CCPT1$ - COMCCPT was incorrectly assuming that if a 7700 table did not have the time in word 3, then the comment started there.

CECS2$ - COMCECS was using only 18 bit arithmetic on ECS addresses.

CECS3$ - COMCECS was not restoring -0 to A and B registers correctly.

PRLI2$ - COMPRLI was not calculating negative relocation address correctly.

CIO4$ - Install CDC version of direct access file size limits. This replaces local mods DAFLIM and CIO003.

CIO5$ - CIO was not flushing the buffer on a WRITECW request where only EOF's occupy the buffer.

CIO6$ - Allow EVICT of execute - only file.

CIO8$ - Make CIO clear the WRITE bit in the FST on an open with rewind request.

CIO9$ - Make CIO return a file with no tracks on a close/return request.

CPM4$ - Repair a hang-situation in CPM caused by improper entry conditions for the error processing subroutine.

CPM5$ - Make CPM ignore SETQP requests from the SYSPROC job in order to prevent it from losing it's special priority.

DIS6$ - Correct scope-blanking problem caused by a timing/interlock problem between DIS and 1AJ.

DIS7$ - Make RESEX aborts under DIS appear more graceful.

DSD18$ - Fix overflow check for last command overlay in DSD.

LFM6$ - Correct device type returned by STATUS when equipment is not assigne

LFM7$ - Disallow unlock of execute - only file.

LFM9$ - Correct a situation where LFM returned tracks assigned to other files when attempting to make primary a file with no tracks.

0263$ - Install overflow checks into 026.

PFM2B$ - Repair a PFM error where the modification date on a file residing on a non-master device was not being updated on a write-mode attach.

PFM8$ - Change PFM to correctly allow master users to access files on subordinat user numbers.

PFM9$ - Correct PFM to not assume that the length of a FET is the largest possible.

PFM11$ - Make PFM return a proper error when a CATLIST function does not specify a file name.

PFM12$ - Fix REPLACE processing so that the file access count is updated.

QAC2$ - Correct a coding error in QAC which caused forms code not to be altered.

QFM9$ - Prevent QFM from generating garbage file names.

REC5$ - Correct an error in REC causing it to hang at deadstart attempting to display a message at the wrong control point.

RPU1$ - Correct an error which caused the extended-reprieve bit to remain set after clearing a reprieve.

SET1$ - SET was using the wrong default channel for the display console.

QDF3$ - MS validation was sometimes not being updated.

QRP1$ - The return-files event was not issued in all cases.

1AJ4A$ - Repair an error which caused garbage dayfile messages to be issued.

1AJ6$ - Fix a possible hang situation in 1AJ.

1DS2$ - Repair a minor coding error.

1MA2$ - 1MA was sometimes returning the wrong ECS MFL value on a MEM request.

1MT8$ - Inhibit 1MT from writing level numbers greater than 17B.

1MT9$ - Repair equipment mnemonics in some 1MT messages.

1MT10$ - Repair assorted errors in L format tape handling.

1MT11$ - Repair spurious NO EOP messages issued by 1MT.

1MT12$ - Correct a error in 1MT which caused L format write at 6250 CPI/200 ips to not work.

1MT13$ - Use ZERL rather than absolute zero for clearing core.

1MT14$ - 1MT could incorrectly diagnose an ATS conversion table load error.

1RO2$ - Eliminate an unnecessary call to QDF.

CHKPT3$ - CHKPT would try to checkpoint a file with name INPUT*.

CONTR1$ - Correct processing of out-of-range arguments in CONTROL.

CONTR2$ - Repair default argument processing in CONTROL.

CPUML3$ - Correct loading of CMU code in CPUMTR on a dual cpu 170.

PUMT23$ - Low core could be wrecked by a XJP request on a dual cpu system with CPU0 off.

PUMT25$ - Fix a problem in CPUMTR when the idle package aborts. An error flag is not set and control point zero is not advanced.

PUMT26$ - CPUMTR could sometimes incorrectly cause a wrong MOVE IN PROGRESS response on a TDAM request.

PUMT27$ - X3 was not being restored after an ECS error.

DFTER6$ - DFTERM was sometimes not clearing the FET.

GTR13$ - GTR was sometimes not building directories correctly.

IBED22$ - LIBEDIT will no longer return the new file if it is a tape (!).

IBED23$ - Properly process random addresses greater than 400000B.

LIBGE3$ - Correct LIBGEN to not accept duplicate parameters.

OD1F13$ - Enhance a MODIFY error message.

MSI3$ - Prevent processing of other devices in a family by MSI when the device to be initialized has not been recovered.

RESEX3$ - Prevent RESEX from building garbage E,P-display.

SYSED9$ - SYSEDIT could sometimes mode 1 when attempting to rebuild the system with a new LIBDECK.

PROPOSED CHANGES TO THE SYSTEM

MORE ABOUT CALLPRG AND THE GLOBAL LIBRARY SET - by M. Riviere et al

This article contains modifications and additions to the "Redefining the Global Library Set" proposal that I published in the December 11th issue of the DSN (Volume 5, Number 23). The first part of the modification consists of changing the error message "REVIEW GLOBAL LIBRARY SET", issued when the global library set can not be extended any longer, to a new message with more meaningful and explanatory text such as "GLOBAL LIBRARY OVERFLOW. SEE ENQUIRE,L." The second part of the modification consists in issuing dayfile messages stating what are the new user libraries that have been added to the global library set when a FETCH statement is executed. The test of the messages should be in the form "NEW USER LIBRARY: LIBNAME." One message should be issued for each new library. After naming all the libraries, a final line with the text "GLOBAL LIBRARY SET UPDATED" should be issued. The last line will be the only line printed for time-sharing terminals. To see the newly added libraries, a timesharing user will have to look at the dayfile.

The above changes were suggestions made by members of the System's Group. Additions or further enhancements of the proposal are a result of the library planning committee resolutions. I will present here a short discussion of our attempted future library definition scheme, the needed CALLPRG modifications to accomplish it and some examples that may help clarify the subject.

The scheduled date for the modifications is the end of the Winter Quarter. Perhaps, not all the modifications that I will describe here may be implemented in CALLPRG for that date. I will specify which of the features will, for sure, be available by then. The rest of the changes can be implemented later on. The second set of changes may still be reviewed and modified but I want to announce them here. I consider all the changes as being part of a planned project that should be compatible with whatever we implement first.

Beside the CALLPRG program and the CALLPRG index being modified to fit the new library scheme, it is also important to have clear user information about the subject, and to make that information available early in advance of the project's implementation. Michael Frisch is writing a descriptive article for the UCC Newsletter, and I believe that a short permanent writeup should also be created (WRITEUP,LIBSET?).

New let's get into the internal phase of the new libraries scheme. As we all know, we have at UCC several user libraries that, for the purpose of this new libraries set up, can be considered as composed of two parts. One part is purely a function processing section that is completely compiler independent. The other part contains features, such as input/output routines, that have an interface to a compiler where other libraries are used (MNF, FTN, PASCAL). Note that there may still be a third section in some of these libraries. This third section is the compiler's own functions as they exist on MNFCLIB and FORTRAN. This third section is the only part of each library that should remain included in the library as it is now. The first section can be used with binaries produced by any one of our compilers. The second section can only be used with each specific compiler. We are going to call the first section the common section and the second one the compiler dependent section. The third section will

remain with its original function, that is, to supply the most commonly needed external references, as designed by each compiler installation requirements.

The main idea of the new libraries scheme set up consists of splitting these libraries in a way that the common section will become a whole library by itself (i.e., the local compiler independent arithmetic routines of FORTRAN and MNFCLIB, will be included in a library called MINNLIB. MINNLIB will be available for use by FORTRAN and MNF programs and also by other compiler generated programs such as PASCAL). The compiler dependent section of MNFCLIB will be part of an MNF dependent library called MNFIOL. In the same way the compiler dependent section of FORTRAN will be part of an FTN dependent library called FTNIOL. This same process will be applied for all affected UCC libraries. Libraries such as MNFIOL and FTNIOL will contain the compiler dependent sections of all our actual libraries. We can look at this compiler dependent library as formed by several independent sections. Each one of the sections should be updated whenever the library to which it corresponds requires updates. The following chart helps describe the future library organization. I need to make the description clear in order to continue further with what CALLPRG modifications are concerned with.

FORTRAN	CDC section goes to FORTRAN (as a whole library) Local Common sections goes to MINLIB (as a whole library) Local compiler dependent section goes to FTNIOL (as a section)
MNFCLIB	MNF section goes to MNFCLIB (as a whole library) Local common section goes to MINNLIB (as a whole library) Local compiler dependent section "goes to" MNFIOL (as a section)
IMSL	Common section goes to IMSL (as a whole library) Compiler dep. section goes to (PASCAL compiler) PASIOL (as a section) Compiler dep. section goes to (FTN compiler) FTNIOL (as a section) Compiler dep. section goes to (MNF compiler) MNFIOL (as a section)

When a user needs to access a given library, lets say MINNLIB, he or she will need to establish, besides the library name itself, which compiler dependent library should be associated with it. The selection of the compiler associated library should be made by a qualifier added to the FETCH, LIBNAME statement for this purpose could be in the form of:

FETCH,MINNLIB/V=MNF

or

FETCH,MINNLIB/V=FTN

The CALLPRG index entry to supply the needed set of libraries should be of the form: MINNLIB,TY=FETCH,V=MNF...DA=MINNLIB+MNFIOL,UL=MINNLIB+MNFIOL. and MINNLIB,TY=FETCH,V=FTN...DA=MINNLIB+FTNIOL,UL=MINNLIB+FTNIOL. The same can also apply for PAST and FUTURE libraries.

So far, things seem good, since the introduction of the V qualifier can neatly be added in CALLPRG (see footnote). But we are facing some problems.

1. Let us assume that the user types: FETCH (MINNLIB/V=MNF) and there is no entry in the index of the form: MINNLIB,TY=FETCH,V=MNF...
What do we retrieve in this case to the user? We can give a message "MINNLIB NOT FOUND, CURRENT, IF AVAILABLE, USED" set MINNLIB as a user library and let the user access the current systems version. Note that not all the libraries will necessarily have the current version in the system. That is why the "IF AVAILABLE" text should be added to the actual CALLPRG message. (The actual message: "NO FUTURE (FETCH OR PAST) AVAILABLE, CURRENT USED" is not true all the time). But what do we do about the I/O library? How do we proceed to set MINNLIB and MNFIOL as users libraries? We can easily define MINNLIB as a user library but to define MNFIOL as a user library is not possible since we do not know its name at that time.

The way to handle this case could be to define MINNLIB as a user's library and then issue a second message with the text "MNF ASSOCIATED LIBRARY UNDEFINED" where MNF will be transferred from the name used as the qualifier in the FETCH,MINNLIB/V=MNF of the user's statement. Though still another message ("MNFI/O LIBRARY HAS TO BE FETCHED"?) and also through an adequate writeup the user should know that if s/he is satisfied by using the current system version of MINNLIB, s/he still needs to complement the global library set by issuing another FETCH statement for the I/O library MNFIOL(FETCH,MNFIOL)

2. Let us assume a job requires the following library configuration.

```
PAST,MINNLIB/V=MNF

FETCH,IMSL/V=MNF
```

(we have fussy people around...don't we?) And let us also assume that the libraries in question are associated with different compiler dependent libraries. That is, MINNLIB is using a past version of the compiler dependent library and IMSL is using the current one. In this case the index entries for these statements are going to be:

```
MINNLIB,TY=FETCH,V=MNF,DA=MINNLIB+MNFIOLP,UL=MINNLIB+MNFIOLP.
IMSL,TY=FETCH,V=MNF,DA=IMSL+MNFIOLC,UL=MINNLIB+MNFIOLC.
```

(note the "P" and "C" suffixes added to the MNFIOL name). What do we do in this case? First. upon execution of the FETCH,MINNLIB statement we are printing the following dayfile messages:

```
NEW USER LIBRARY:  MINNLIB (dayfile)
NEW USER LIBRARY:  MNFIOLP (dayfile)
GLOBAL LIBRARY SET UPDATED (Timesharing terminal and dayfile)
```

Then, upon execution of the second FETCH statement we define IMSL as a user library and print the message "NEW USER LIBRARY : IMSL." But when we get to the step of defining MNFIOLC we can discover a possible library name conflict. We define MNFIOLC as a user library anyway, but we print the message:"POSSIBLE LIBRARY CONFLICT : MNFIOL". How do we detect a possible conflict? We do it be restricting the special compiler dependent library names to be six characters long. We then use the last character as a suffix to define which version of each library is contained on the given files. The suffix can be any character (or blank). Well, this is a little kludgy indeed, but I do not think that there is another way. (The possible library name conflict may also appear from previous library definition made by the user with a LIBRARY,MNFIOL statement.) If anyone has a better idea, I am open to suggestions.

3. Let us assume that the user defines two user libraries, that are somehow unrelated, through FETCH statements as follows:

```
FETCH,MINNLIB/V=MNF
FETCH,MINNLIB/V=FTN.
```

I consider these libraries to be unrelated since they are to be used by different compilers, and therefore no job step will need to be used at once. In this case, the global library set will contain the libraries MINNLIB, MNFIOL and FTNIOL. What happens if the user now loads a FTN program and gets some external references satisfied from MNFIOL instead of FTNIOL? This can happen since both libraries may contain records with identical names and MNFIOL is scanned before FTNIOL. We can't protect the user in this case. I've already discussed this possibility in my previous article, and there is not much we can do about it. We will have to instruct the users to be careful and clear the global library set and redefine it, if needed, as they switch from one compiler to another.

We could alleviate the problem by adding new libraries at the top of the LDSET block instead of at the bottom. The LDSET block rearrangement will direct the Loader to scan first the latest defined libraries. If we rearrange the LDSET block we may also create a sort of upside down library search sequence that can create worse invisible problems. Consider, for example loading a routine that resides in MNFIOL and is named identically to one in a user supplied library. We would also be interfering with a possible good sequence of library definitions created by the user to minimize circular library searchs.

4. Let us assume that a job requests library IMSL with the statement:

```
FETCH,IMSL.
```

and we have in the CALLPRG index only entries for IMSL of the form:

```
IMSL,TY=FETCH,V=MNF...
IMSL,TY=FETCH,V=FTN...
```

No entry for a general IMSL library is compatible with all compilers. (Should we have a no-compiler associated version of IMSL we will be retrieving that one). Do we handle this case as case 1? That is, we make IMSL part of the global library set and issue the message:

"IMSL NOT FOUND, CURRENT, IF AVAILABLE, USED"?

Now with a few of the possible problems that I believe we will encounter when implementing this modification already described and with solutions for some of them, let me announce, in short terms what are the needed CALLPRG modifications:

1. Implement the V qualifier.
2. Recognize libraries with semi-identical names.

The implementation of these modifications, the installation of adequate CALLPRG index entries according with the group of libraries that we will be offering, a descriptive UCC Newsletter article and a permanent writeup should be the minimum work to be implemented for the end of the Winter Quarter.

Now let us go into more complex possibilities of extending the function of the FETCH statement. The extension should make the user's task of obtaining all his/her needed libraries easier. (see footnote.) This extension should process statements in the form: FETCH,IMSL,TEKLIB/V=MNF. The complexity of the code to handle the already described implementation of a single product FETCH statement multiplies geometrically when we want to expand it to several products. What do we do if there is a version of IMSL available for the V=MNF qualifier but not for TEKLIB? Do we still want to keep printing the messages "CURRENT USED" with reference to TEKLIB? How about if this library happens to have index entries such as:

```
IMSL,TY=FETCH,V=MNF,DA=IMSL+MNFIOLP,UL=...
TELKLIB,TY=FETCH,V=MNF,DA=IMSL+MNFIOLC,UL=...
```

The answer, of course, is to handle everything as for a single FETCH statement. But things become quite complicated when we need to split the scanning for a multiple FETCH statement between a user CALLPRG index and the system index. The code to be added in CALLPRG to handle cases when not all the products are found in the user's index is quite complicated and may delay us from our deadline. This extension could be considered in the future.

Footnote: Modifications similar to the ones proposed here were suggested by Dennis Lienke for CALLPRG in 1977 (Volume 2, Number 19). Part of the code to handle this extension of the FETCH statement was introduced in CALLPRG by James Mundstock at that time. The proposal went into several considerations and perhaps due to its complexity, was not accepted. The proposal and the code were "abandoned" since then. The code was never fully tested and not all the possibilities of the current implementation have been analyzed.

//////////

Library Reshuffling Proposal - by Mike Frisch

I propose that the following article be published in the February and March UCC Newsletters. (The section entitled "Background for these changes" would be part of the article.)

Attention All Fortran Subprogram Library Users

Beginning March 23, 1980, you must change Fortran jobs on the Cyber 74, 172 or 170-720 if they use any of the routines in the Minnesota Subprogram Library (see the list below).

If you use the FTN 4 compiler, add this statement in your deck after the USER statement:

```
FETCH(MINNLIB/FTN)
```

If you use the MNF compiler, add this statement in your deck after the USER statement:

```
FETCH(MINNLIB/MNF)
```

At the beginning of an MNFTS or FORTRAN timesharing session, enter this statement:

```
X,FETCH,MINNLIB/MNF
```

For example, if you run a Fortran program with the FTN 4 compiler which calls the routine MXLNEQ (one of the routines on the Minnesota Subprogram Library), the change would be:

Before March 23	March 23 and After
Job statement	Job statement
USER statement	USER statement
FTN	FETCH(MINNLIB/FTN)
LGO.	FTN.
7/8/9	LGO.
CALL MXLNEQ(...)	7/8/9
etc.	CALL MXLNEQ(...)
	etc.

The FETCH statement remains in effect for the entire job or timesharing session so it only need be entered once at the beginning. It must also be entered if you run later jobs or sessions in which only the relocatable code is executed.

If you do not include the FETCH(MINNLIB/compiler) statement starting March 23, you will get the loader error message "UNSATISFIED EXTERNALS," and your job will not execute correctly.

If you use any of the libraries IMSL, EISPACK, TEKLIB, HTEKLIB, FUNPACK, CALCOM, BESPACK, MEXPLOR, GPM, ALMAP, BSPLINE, YSMPLIB, or SIMPLX, you must also change your jobs.

If you use the FTN 4 compiler, change your FETCH(library) statement to FETCH(library/FTN). For example, FETCH(IMSL) becomes FETCH(IMSL/FTN).

If you use the MNF compiler, change your FETCH(library) statement to FETCH(library/MNF). For example, FETCH(EISPACK) becomes FETCH(EISPACK/MNF).

During an MNFTS or FORTRAN timesharing session, change your X,FETCH,library statement to an X,FETCH,library/MNF statement. For example, X,FETCH,TEKLIB becomes X,FETCH,TEKLIB/MNF.

If you do not change your FETCH(library) statement starting March 23, you will get the error message "library MUST HAVE /COMPILERNAME. /FTN ASSUMED." Thus, starting March 23, FETCH(FUNPACK) will assume the user meant FETCH(FUNPACK/FTN).

If you use both MNF and FTN 4 for separate programs in the same job, you must add a LIBRARY. statement before any FETCH(library/compiler) statement when switching compilers. However, you should not mix binary routines from MNF and FTN 4 in a single program.

The following is a list of the routines in the Minnesota Subprogram Library (MINNLIB):

AITKENF	ALOGAM	AMEAN	AMEAND	APPEND	ASSIGN	ATTACH
AXIS	AXISP	BANSOL	BESJ	BETAI	BINCOF	CDFN
CDFNI	CEINT1	CEINT2	CHEBY	CHECK	CHSQ	CHSQI
CINTEG	CMXCMBN	CMXLNEF	CMXLNEQ	CMXMOV	CMXMPLY	CMXPLY1
CMXTRP	CNTOUR	COMINT	CONST	CONVERT	CPRFIL	CVAL
CVECT	DEFINE	DMXLNEF	DMXLNEQ	DOTPRD	DOTPROD	DPLOT
DVAL	DXINT	EI	EIG3	ERFN	ERFNC	EXPD
FACTOR	FINV	FREQDSN	FRESNEL	FTEST	GAMMA	GAMMAF
GENSORT	GENSRT2	GETPF	GETRAN	GNREAD	GRIDIT	ICOUNT
ICPA	IRAN	ISERCH	IVLFREQ	LEGAL	LINE	LINT
LOCFET	LRSHFT	LSQORPY	LYNE	LYNE1	MAKEFET	MEANVAR
MERGE2	MERGE4	MXCMBN	MXEXTRM	MXLNEF	MXLNEQ	MXMOV
MXMPLY	MXMPLY1	MXTRIDI	MXTRP	MXTRP1	NEWPEN	NONLIN
NORMAL	NUMARG	NUMBER	ORTHON2	ORTHON3	PERMUTE	PLOT
PLOTS	PLOT3D	PLOT3S	PLROOT1	PLROOT2	PLROOT3	PLTSCL
POLPLOT	POLYGN	PRCON	PRNPLOT	PROCCIO	PROCCPM	PROCER
PROCERC	PROCFET	PROCLFM	PROCPFM	PROCQFM	PROCRTN	PROCSFM
PROCSYS	PROCTLX	PURGE	QLENTH	QRCMPLX	QRSYM	QSORT
QSORT1	Q92X8J	RANBIN	RANBIT	RANDOM	RANT	RAN2F
RAN3F	RCVECT	REPLACE	RESTFL	RETURN	RK	RKGILL
ROM1F	ROM2F	ROOT1	RPAUSE	RVAL	RVECT	SAVE
SCALE	SCLPLT	SETRAN	SICI	SIMPSON	SKALE	SKALE1
SNCNDN	SORT1	SORT2	SPACZRO	SQALE	SYMBOL	SYMINV
SYMPACK	SYMPLOT	SYMSOLU	SYMSOLV	SYMUPK	TINV	TINV1
TINV2	TTEST	TTEST1	TTEST2	WHERE	WHEREAS	XCEINT1
XCEINT2	XCEINT3	XINT	ZROSPAC			

Background for these changes

When we started using large-scale computers in 1961, 90% of our usage was Fortran compilation and execution. Since many Fortran programmers needed the same kinds of subprograms (for sorting, special functions, and matrix operations, for example), we developed a large library of efficient subprograms at UCC that the Fortran programmer could invoke easily. It was made part of the standard Fortran library; therefore, no extra control statements were needed to access the routines. The subprogram library was essentially "invisible" to the user, though each routine was well documented.

In the 1970's, we obtained a number of special subprogram libraries. The routines in these libraries (IMSL, EISPACK, TEKLIB, etc.) are high quality routines. They are accurate, well documented and portable (that is, they can be moved from site to site without a major reprogramming effort), and they execute rapidly.

We added these externally developed subprogram libraries to our operating system and required the FETCH control statement to access them. However, the "invisible" subprogram library developed at UCC was retained within the standard Fortran library. In this, UCC was unusual. Most major computer centers require that the user specify any locally developed or externally acquired subprogram library with a specific request for that library.

Recent developments made us re-evaluate the "invisible" library of UCC subprograms. Tests made by former staff member C. F. Schofield on small instructional and research Fortran jobs showed that of about 1.1 seconds total CP time, 0.2 seconds was spent in compilation, 0.2 seconds in execution, and 0.7 seconds in loading the relocatable binary. The 0.7 seconds loading time could be cut in half if the "invisible" library were made a separate library, a net 35% improvement for these jobs.

Furthermore, there is now a proliferation of Fortran compilers: MNF, M77, FTN 4 and (soon) FTN 5, plus a Pascal compiler that has a capability for allowing calls to Fortran library subprograms. When a Fortran subprogram that contains no I/O statements is compiled with FTN 4, it can be used by programs compiled with any of the above compilers. However, since each compiler translates I/O statements in a different manner, the compiled code of subprograms containing I/O statements is different for each compiler.

This led us to build all our Fortran libraries by compiling all routines that contained no I/O statements with FTN 4 and putting the routines that contained I/O into the standard library for each compiler, thereby increasing the size and loading time for all Fortran users.

An additional problem was that CDC's loader design puts a limit on the number of global libraries that can be used simultaneously in a given job step. (Global libraries are defined by the LIBRARY and FETCH statements; local libraries are defined by the LDSET statement but there are practically no limits on local libraries.) The global library limits are either two user libraries plus two system libraries, one user library plus thirteen system libraries, or no user library plus twentyfour system libraries. Each FETCH statement was taking up one user library and this caused some users with their own user libraries to have to add LDSET statements. Because LDSET only defines local libraries, each load requires its own LDSET statement and it must occur just before the load. This can be annoying when there are several loads in a single job or timesharing session.

Considering the Schofield results, the changes caused by additional compilers and the global library limit problem, we looked at three alternatives. The first was to do nothing further. That was rejected because it meant that most users were paying for unneeded load time. The second alternative involved a change to all compilers to generate nonstandard LDSET entries for the relevant libraries in the relocatable binary code. This was rejected because it would have required that all users' relocatable binary code be recompiled when UCC made this change.

The third alternative was to separate the "invisible" library from the standard Fortran library. It would be split into two parts: a part called MINNLIB that had routines which contained no I/O statements, compiled with FTN 4, and a part that had routines which contained I/O statements, compiled into separate versions for each compiler: MNFIOL, FN4IOL, etc.

The FETCH statement would also be changed to associate each library with its corresponding compiler. Thus, FETCH(MINNLIB/MNF) would access the MINNLIB library compiled with FTN 4 and the MNFIOL library compiled with MNF. Libraries such as EISPACK which contained no I/O statement would only access a single library: EISPACK in this example.

To solve the global library limit problem, the most important UCC libraries would be made system libraries. In that case, virtually all library users would have no more than two global system libraries and could have up to two of their own user libraries without needing LDSET statements. If the user referenced more than one Fortran compiler in a single job, LIBRARY. statements would have to be added to clear out the old global library list before the FETCH statements were used. Note that the ENQUIRE(OP=L) control statement prints out the global library list.

The advantages of the third alternative were:

Loading time would be reduced for most users, those who access only the standard Fortran library.

We could ensure that jobs are more portable by requiring users to explicitly FETCH all necessary libraries. UCC can supply Fortran versions of virtually all the routines in MINNLIB (and UCC-written routines in the IOL libraries) if the programs calling them are taken elsewhere by users.

We could now collect accurate usage statistics for each library. This was difficult with a single Fortran library.

A single copy of the routines in MINNLIB would be easier for us to manage than separate "invisible" copies inside each standard Fortran library.

The Fortran compilers and their standard libraries could be installed on our systems without any changes or additions and all UCC Cyber computers would have the same library structure.

The sole disadvantage was:

Users of our libraries (other than the standard Fortran ones) would have to add or change some control statements for each job and our documentation would have to be changed accordingly.

We accepted the third alternative because it produced the best results and affected the fewest users. Its advantages outweighed the disadvantages. This was our "environmental impact statement" for this UCC decision.

//////////

PROPOSAL FOR A NEW DSP - by K. C. Matthews

I have been playing on and off for several months with a re-written version of PP program DSP. DSP puts files in the queue in response to the ROUTE control statement. DSP also places files in the input queue for the SUBMIT statement. DSP is used by the subsystems (BATCHIO,SUPIO) to place jobs in the input queue. Finally, DSP is used by the queue dumping and loading utilities to place files in the queues.

The parameter block used when DSP is called is a very messy looking thing. I am not planning on changing the DSP call block, because that would involve changing all the programs (including, perhaps, user programs) which call

DSP. Instead, DSP will be reorganized internally, leaving the external interface unchanged.

The original CDC version of DSP had one main program; one overlay existed for error processing. It was changed by Brian Hanson when we went to NOS Release 3 to move some of the main program code into another overlay. This was necessary to make room for some of our multi-mainframe code. The changed DSP works, but it looks bad and is not organized in any easy to follow fashion. There is not any room left in DSP to add in any new features - like auto-divert.

The rewritten DSP gets around these problems and provides a few extra features. The main feature is that two (or perhaps three) dayfile messages appear in the user dayfile for each file placed in the queue. If, for example, I ROUTE file OUTPUT to the Experimental Engineering print queue, the following two messages are issued:

```
OUTPUT IN PRINT QUEUE FOR EA(74).
216 PRUS IN QUEUE FILE ALTQABX.
```

The first message gives the original lfn for the file, which queue it went to (print, punch, plot), the TID for the queue files (EA in this case) and the machine ID of the machine which will finally process the file. The second message gives the size of the queue file in sectors (decimal) and the queue file name of that file. A third message can be issued if the file was diverted. For example, if a long print file was diverted from EA (Experimental Engineering) to BC (Lauderdale) the message could say:

```
DIVERTED FROM EA TO BC, PICKUP AT EA.
```

I am not sure when this message should be issued. The two reasonable choices are:

1. Whenever a file is diverted from one TID to another, or:
2. Whenever a file is diverted and the "PICKUP AT" terminal id is not the same as the "DIVERTED FROM" terminal id.

Some other messages are issued for other routines. These will replace the ROUTE COMPLETE message that always comes out now. If a file called XYZ is deferred routed, the message:

```
XYZ - DEFERRED ROUTE SET.
```

is issued.

If a deferred route is rescinded, (DC=SC) for XYZ, the message

```
ROUTE RESCINDED FOR XYZ.
```

is issued.

If no DC is specified on an immediate route, DC=SC is assumed. This is true currently. If this is tried on a file that was not previously deferred routed, the warning message:

```
NO ROUTE SPECIFIED FOR XYZ.
```

is issued.

This will be much more helpful than the current "ROUTE COMPLETE" message.

Thus, one effect of the rewritten DSP is a set of reasonable (I hope) dayfile messages. A second feature is that DSP will process all files at end-of-job. Currently, 1CJ is called at end-of-job to complete the job. All queue files are placed in the proper queue, an OUTPUT file is created (if needed), and the job dayfile is copied to the end of the OUTPUT file for the job. All other files are returned. Under my proposal, DSP will be called at end-of-job to process all these file functions. This means that the queue file messages listed above will be issued for all queue files. (Old timers will remember that this was the case in MOMS.) This is good, because all queue file processing will be done in one routine - DSP. When we change things, hopefully only DSP itself will have to be changed.

The advantages of a rewritten DSP are.

1. At least one UCC person (KCM) will understand the organization of DSP.

2. DSP will be readable.

3. A decent set of messages will be issued for queue file operations.

4. All queue file insertion will be in one place.

5. There is room in DSP to add changes.

The disadvantages are:

1. There will probably be some initial errors. Although I plan to test DSP extensively, there are probably some obscure calls to DSP that might not work. I blame this on the lack of organization and definition inherrent in the DSP call block. I hope that any such errors should be easy to correct.

2. We will be farther away (again) from CDC in the queue file area. This will cause problems (again) whenever we upgrade operating systems.

SYSTEM MAINTENANCE: People and Procedures

Cyber Deadstart Dump Analysis from Wednesday, 26 December through Sunday, 6 January

Wednesday, 26 December

12:27 (DD2020) Cyber 74
The system hung when a job was being cleared from the EXPORT control point. This is done by a memory change command. The command was wrong, and the changed file looked like a rollout file to the system. 1RI hangs the system on a bad rollout file. This bug in 1RI has been fixed.

Monday, 31 December

14:02 Cyber 172
1IO and 1CD seemed hung up at the BATCHIO control point. The deadstart dump was unsuccessful, so the cause cannot be determined.

Tuesday, 2 January

08:06 Cyber 172
An omission during the re-cabling of some disk drives went badly when no drive was available for pack SHA. An emergency fix was created, but SHA was then not set up as a shared queue device. DSP hung due to the lack of a shared queue device.

23:08 (DD2005) Cyber 74
1AJ hung. The problem was due to a delay queue/large job problem. The problem was corrected on 3 January.

Wednesday, 3 January

16:21 (DD2004) Cyber 172
1AJ hung again. This is the same problem as that of January 2 on the 74.

Friday, January 4

16:16 (DD2011) Cyber 172
1CJ hung dropping the tracks on an input job from the 74. The tracks must have been dropped, probably on the 74. We cannot determine the cause.

24:00 (DD2021) Cyber 172
TELEX hung when it was stopped for some hardware work. The cause of the hang was a bad file created during the recovery deadstart at 16:16. This is a software problem which will be corrected soon.

//////////

Cyber 170-720 Deadstart Dump Analysis (12/26 - 1/6) - by R. A. Williams

Date	Description	Tape
800102	TELEX aborted with a GQE (get queue entry) abnormal error. Don Mears has found and fixed a bug relating to monitor mode that was responsible for this abort.	Fixed