SPERRY UNIVAC

no. CS-61 rev. no. _____ date: 3/14/79

~~Owen Townsend~~

UNIVAC VANCOUVER

CHARLIE GIBBS

OS/3
COMPONENT PRODUCT SOFTWARE DESCRIPTION

name: CDI Library User Interface

component no.: ____________

author: M. B. Todd

hardware systems: □ SUL □ 90/25 □ 90/30 □ 90/40

□ ____________

ABSTRACT:

From Jerry GABLE — These Working Papers not Current but should be OK for our purposes Call if you have further questions

215-542-6916

APPROVALS:

section manager:	director
group manager:	
director:	

1.0 Introduction

1.1 Scope

This documents the CDI user interfaces to system libraries using library utilities. This one necessary to access or generate library modules and manipulate the library is contained here.

The MIRAM library modules supported for the current release (R7.0) are: R7.1

- o saved run library modules
- o screen format modules

The SAT library modules supported for the current release are:

- o source
- o proc

NO OBJECT
LOAD

2.0 Functions Available

The system library interface allows user programs to access directly any system library file. These functions are provided:

- o reading a library module
- o creating a library module
- o deleting a library module
- o interrogating a library directory

2.1 Library Interfaces

The system library [illegible]

2.1.1 Interface Macros

The CDI macros are the only ones used to interface with libraries (thru library utilities). Both imperative and declarative macros are used. The imperatives are:

- OPEN - data management file OPEN
- CLOSE - data management file CLOSE
- DMINP - retrieve a library record
- DMOUT - write a library record
- DMSEL - initialize a library operation (usually) ?

The declaratives are:

- CDIB - library operation control block
- RIB - used only at OPEN (as per data management)

2.1.2 Flow of Control

The CDIB (as defined by data management) is the basic controlling block used by library utilities. It is used to communicate between the user program and the utilities. It is specified on each imperative macro and will contain status on return.

The user starts by issuing an OPEN imperative to a CDIB which identifies the library file involved (by LFD). This is a standard data management requirement.

The next step depends upon what kind of operation the user wants to do. The operations available are described in Section 2.2 and assume the CDIB is already open. The flow of most operations is:

1. DMSEL - initialize (or identify) operation
2. DMINP/DMOUT - retrieve or present data
3. DMSEL - terminate operation

This is an outline of a typical operation. Step 2 will probably be executed more than once. Step 3 is not always required.

The user must issue a CLOSE imperative macro for each CDIB active before the program terminates.

2.2 Reading and Writing Library Modules

This interface allows the user to read or write library modules a record at a time. Records of a library-module are passed between library utilities and the user program as a result of a DMINP or DMOUT macro instruction.

2.2.1 Reading a Library Module

This sequence of imperative macros is required:

(0)

1. DMSEL cdib,LU TB,IN {~~ELEMENT=name, TYPE=type~~ / ,NEXT}

2. DMINP cdib, [(0) / workarea]

DMSEL initializes the operation by locating the module and returning the header record in the data buffer (IOAREA1). Two types of operation are permitted:

- o select element of specified name and type
- o select the next sequential element in file.

DMINP moves data records to the user buffer. The user issues as many of these as required. When the modules is exhausted, an end of data condition is set in the CDIB.

2.2.2 Writing a Library Module

This sequence of macros is required:

1. DMSEL cdib,LU,OUT,ELEMENT~~=name,TYPE=type~~ (0)

2. DMOUT cdib, [Ø / workarea]

3. DMSEL cdib,LU,ADD

DMSEL (out) initializes the operation.

DMOUT accepts user data records sequentially and adds them to the end of the module being generated. The user issues as many calls as required.

DMSEL (ADD) completes the operation by adding the module to the file. All required directory indices are created by library utilities on this call. If a module exists with the same name, it is deleted. When this call is omitted, this is the result:

- o the module is not added to the file
- o a module with the same name is not deleted
- o the integrity of the file is maintained

2.3 Deleting a Library Module

To delete a library, use the DMSEL macro:

```
DMSEL   filename, LU,(b) DEL, (o) ~~name, type~~
```

2.4 Interrogating a Library Directory

(To be described in a later revision).

3.0 Design Trade-Offs and Product Objectives

The goals in providing library utilities through a CDI interface are:

1. To provide a standard interface consistant with other data management interfaces. We can eventually allow <u>users</u> to use library utilities (which we never have had before). The interface is easily learned as it is a derivative of the CDI system standard.

2. To provide device independence for sequential source INPUT/OUTPUT users. Some of the special commands (such as SELECT module name for input) should be optionally done automatically as part of OPEN. This would require more information on the job control device allocation set. It would allow users to use the same routine to read/write from either a sequential device or a library element.

4.0 Interface Description

4.1 User Interface

4.1.1 Declarative Macros

4.1.1.1 CDIB Macro

The CDIB is the controlling block for any library operation in progress. Its use is consistent with the data management interface. His format is:

```
   filename  CDIB
```

The format of the CDIB is defined in CS-14.

4.1.1.2 RIB Macro

NAME	OPERATION	OPERAND
label	RIB	[,ACCESS= EXC EXCR SHR]
		[,BFSZ= 256 n]
		[,INDA=symbol] *
		[,IOA1=symbol]
		[,IORG=(r)]
		[,NWAIT= NO YES]
		[,RCSZ= 256 n]
		[,WORK= NO YES]

* Used for MIRAM library files only.

All parameters are optional. Omitted parameters take on the underlined value.

The INDA and IOA1 parameters are not required, either. If they are omitted, the required space will be acquired from dynamic main storage. If they are specified, they are subject to the constraints:

- o For SAT library files, INDA is ignored and unused
- o For MIRAM library files, either specify both INDA and IOA1, or omit both. When you specify them, the INDA buffer must immediately precede and be contigious to the IOA1 buffer in main storage.

Parameters:

ACCESS=option — defines the disk file lock used. (See CS-14 for details).

BFSZ=$\left\{\begin{matrix}\underline{256}\\ n\end{matrix}\right\}$ — The disc buffer size (parameter IOA1). It must be a multiple of 256.

INDA=symbol — The symbolic address of the index processing buffer. It must be halfword aligned. Its length is 256 bytes.

IOA1=symbol — The symbolic address of the disk buffer. It must be halfword aligned.

IORG=$\left\{\begin{matrix}\underline{2}\\ 2\text{-}12\\ r\end{matrix}\right\}$ — r is the general register used to point to the current record when the user is not using a workarea. This parameter is ignored if work=yes is specified.

NWAIT=$\left\{\begin{matrix}\underline{NO}\\ YES\end{matrix}\right\}$ — Specifies whether to return on a file lock error. (See CS-14 for details).

RCSZ=$\left\{\begin{matrix}\underline{256}\\ n\end{matrix}\right\}$ — The size of the user's work area (where records are returned). It should be as large as the largest record expected.

WORK=$\left\{\begin{matrix}\underline{YES}\\ NO\end{matrix}\right\}$ — Specifies whether records are to be placed in the workarea. If NO is specified, records will be pointed to by IORG. (Note: Source-records will only be expanded when YES is specified.

4.1.2 Imperative Macros

4.1.2.1 DMSEL Macro Format

$$\text{DMSEL}\quad \text{cdib,LU,}\left\{\begin{array}{l}\text{ADD}\\ \text{DEL}\\ \text{IN}\\ \text{NEXT}\\ \text{OUT}\end{array}\right\}\left[\overset{(D)}{\text{,ELEMENT=}}\left\{\begin{array}{c}\text{'string'}\\ \text{TAG}\end{array}\right\}\right]\left[\text{,TYPE=}\left\{\begin{array}{c}\text{'string'}\\ \text{TAG}\end{array}\right\}\right]$$

ADD - add the current module being created to the file. This causes the required directory indices to be created. (Keywords ELEMENT, TYPE do not apply).

DEL - delete the named module from the file. This eliminates all directory indices for the module and marks the module header as inactive.

IN - initialize the input of the named module. (This call is required before DMINP macro calls are permitted.) The module header record is placed in the buffer area.

NEXT - initialize the input of the next module in the file. (This call is required before DMINP macro calls are permitted.) The module header is placed in the buffer area.

(Note: if a BOG or EOG is the next file item, it will be returned).

OUT - initialize the output of the named module. (This call is required before DMOUT macro calls are permitted.)

Where a "named module" is required, it is specified a twelve byte field whose address is in Reggie Ø ~~with the keywords ELEMENT and TYPE~~.

- ELEMENT - the 8 character library module/group name
- TYPE - the four character library module/group name ?

They are ~~can be specified as~~ a contiguous 12 byte string ~~(the actual name in quotes) or the tag of the area containing them may be~~ specified.

4.1.2.2 DM INP/DMOUT Imperative Macros

The DMINP macro makes a record available for processing The DMOUT macro adds a new record to the library module being created. Both macros are generalized imperatives that can be used to request records from all data management processors.

Their format is:

NAME	OPERATION	OPERAND
[label]	$\left\{\begin{matrix}\text{DMINP}\\ \text{DMOUT}\end{matrix}\right\}$	$\left\{\begin{matrix}\text{(1)}\\ \text{1}\end{matrix}\right\}\left\{,\begin{matrix}\text{workarea}\\ (\emptyset)\\ \emptyset\end{matrix}\right\}$

<u>Parameters</u>:

cdib — the symbolic address of the CDIB that corresponds to the similarly named file assigned through job control.

workarea — specifies a user defined work area that will contain the record transferred.

For DMINP, a retrieved record is placed in the workarea if WORK=YES is specified. Otherwise the record remains in the IOA1 buffer and is pointed to by register IORB.

For DMOUT, the user places records to be output in the workarea.

4.2 Operator Interfaces

N/A

4.3 Data Bases

The proposed structure of the MIRAM library file will be presented in an appendix.

The definition of the SAT library file structure exceeds the scope of this document.

5.0 Environment Characteristics

5.1 Hardware Required

Unknown.

5.2 Restrictions :

The MIRAM libraries (and library utilities) exist only in the CDI environment. MIRAM libraries will in inaccessable in the DTF only system.

5.3 Compatibility

N/A

6.0 ARM

To be supplied.

7.0 Performance

To be supplied.

8.0 Standards

To be supplied.

9.0 Standards Deviations

N/A

10.0 Documentation

N/A

11.0 Support

N/A

Appendix A Proposed Structure for MIRAM Libraries:

This format for MIRAM library files is based on the Nailos - Holt - Synder letters (10/17/78 - 11/20/78).

Data Partition

The data partition will consist of 256 byte fixed length non-keyed records. Neither deleted records nor mixed keyed/non keyed records will be used. It is felt that neither feature is required and their inclusion would require a one byte record control block (RCB) in each record. The RCB would break otherwise contiguous text data (for block modules) and unnecessarily complicate the design.

A module consists of a set of contiguous blocks. The first block is the header block, which occupies one full block and contains the number of blocks in the module. The next block (or blocks) contains the module data. All pointers within the module which point to blocks within the module are relative to the module itself. Copying the module from one file to another can be block for block without modification.

The first module will be written starting in the first block of the data partition, followed by an ENDLIB block. When a second module is created, its header will be written in to the same block as the ENDLIB and a new one ENDLIB written at the end of the module. There should only be one EDNLIB block in every file.

The header block format is the same for all module types as well as BOG/EOG records and contains these fields:

- Module (or group) name
- Module type
- Number of blocks occupied by this module
- Flags:
  - status (deleted/active)
  - index item (yes/no)
  - uniqueness (yes/no)
- Number of blocks in module
- Block (module relative) displacement of data/ text blocks.
- Number of text bytes

Directory Partition

The directory is a set of MIRAM indices maintained by MIRAM and manipulated through index-only operations by library utilities. Indices are maintained lexically ordered. Since none of the data records are keyed, library utilities will be aware of the directory entries necessary for each module and add each using an index-only write.

BOG/EOG records occupy an entire block and have the same format as a header record (roughly) with these exceptions:

- Group Name is the module name
- 'BOG' or 'EOG' is the module type
- Number of module blocks is one
- The unique flag is <u>not</u> set (allowing multiple groups of the same name)