27
8-18-77
25 sep. to NTIS

UCID-17418

Lawrence Livermore Laboratory

A DATA BASE MANAGEMENT SYSTEM (ADBMS) INSTALLATION GUIDE

Edward W. Birss
Data Management Research Group

March 17, 1977

MASTER

This is an informal report intended primarily for internal or limited external distribution. The opinions and conclusions stated are those of the author and may or may not be those of the laboratory.

Prepared for U.S. Energy Research & Development Administration under contract No. W-7405-Eng-48.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

CONTENTS

	Page

Introduction	1
How To Install ADBMS	1
Contents of Tape	2
References	3
Appendix: Installation-Dependent Routines	4
Subroutine CPUSEC	5
Subroutine GETDAT	6
Subroutine GETTIM	7
Integer Function IAND	8
Integer Function IBITS	9
Integer Function IBYTE	10
Integer Function IBYTE1	11
Integer Function ICNC1	12
Integer Function ICSC1	13
Integer Function IHASH	14
Integer Function IOR	15
Integer Function ISCOMP	16
Integer Function ISHFTL	17
Integer Function ISHFTR	18
Subroutine ITOC	19
Integer Function LEFT	20
Subroutine LEFTI	21
Integer Function RIGHT	22
Subroutine RIGHTI	23
Subroutine SBITS	24
Subroutine SMOVE	25
Subroutine RANDCL	26
Subroutine RANDOP	27
Subroutine RANDRD	28
Subroutine RANDRW	29
Subroutine RANDWT	30

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Energy Research and Development Administration, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

28

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

ADBMS INSTALLATION GUIDE

INTRODUCTION

This manual provides rough guidelines to aid a programmer in installing ADBMS at a computer installation. ADBMS is a set of subroutines which together form a nucleus of a CODASYL-based data base management system. The use of this system is explained in detail in [1].

HOW TO INSTALL ADBMS

To install ADBMS at another installation, several steps must be followed.

1. Code the installation-dependent routines described in the Appendix.
2. Modify the variable NCW in the BLOCK DATA to correspond with the number of characters per word of your computer. The word length of your computer must be at least 24 bits to work without modification.
3. Modify the DDL Analyzer to correspond with the word size. Input formats 12A6 etc. should be changed to represent contiguous characters. For example, 12A6 should be changed to 20A4 for a 32-bit machine. A6 formats should be modified to A4,A2 again for a 32-bit machine. The CHAR6 and CHAR7 variables should be dimensioned if necessary.
4. Compile installation-dependent routines.
5. Modify the preprocessor which inserts common areas into the FORTRAN source.
6. Compile and test the preprocessor.
7. Compile the DDL Analyzer, the Data Base Initializer, and the Block Data Subprogram.
8. Compile the DBCS routines and create an object library of them.
9. Construct a load module of the Block Data, DDL Analyzer, the DBCS routines, and the installation-dependent routines.
10. Test out the DDL Analyzer.

11. Construct a load module of the Block Data, Data Base Initializer, DBCS routines, and installation-dependent routines.
12. Test out the Data Base Initializer.
13. Code a sample program, compile it, load it with the block data, the DBCS routines necessary, and the installation-dependent routines required.

CONTENTS OF TAPE

1. Preprocessor for inserting common areas.
2. Common declarations.
3. Routines used by DDL Analyzer and Data Base Initializer.
4. DDL Analyzer.
5. Data Base Initializer.
6. DBCS routines.

REFERENCES

1. Birss, Edward W. "ADBMS Users Guide", Lawrence Livermore Laboratory, February, 1977.
2. Hershey, Dissen, Foley "Implementation and Modification Instructions for PSA in Source Form", ISDOS Working Paper No. 111, University of Michigan, August, 1975.

APPENDIX: INSTALLATION-DEPENDENT ROUTINES

NOTE: The subroutine descriptions contained in this appendix are largely taken from [2].

Subroutine CPUSEC

Description:

Subroutine CPUSEC returns the CPU time used since the initialization call to this subroutine.

Definition:

CALL CPUSEC(IS)

Arguments:

IS (input and output) integer

Discussion:

This routine must originally be called with IS=0 for initialization. Subsequent calls made with IS not equal to zero result in IS being set to the number of CPU milliseconds since the initialization call to CPUSEC. It is possible to call CPUSEC with IS=0 several times during a run.

Subroutine GETDAT

Description:

Subroutine GETDAT returns the character format of the date.

Definition:

```
CALL GETDAT(MDY)
```

Arguments:

MDY (output) integer array

Discussion:

The date is returned in the array MDY. MDY must be able to store twelve characters, and the date is left-justified and in the form : MMM DD, 19YY.

Subroutine GETTIM

Description:

GETTIM returns the eight-character time of day.

Definition:

```
CALL GETTIM(HHMMSS)
```

Arguments:

HHMMSS (output) integer array

Discussion:

The eight-character time of day is returned in HHMMSS in the eight-character form HH:MM.SS .

Integer Function IAND

Description:

IAND returns the logical AND of two words.

Definition:

$$K = \text{IAND}(I, J)$$

Arguments:

I (input) integer
J (input) integer

Discussion:

IAND returns the logical bit-by-bit AND of I and J.

Integer Function IBITS

Description:

IBITS extracts bits from a word.

Definition:

$$J = \text{IBITS}(I, N, M)$$

Arguments:

I (input) integer
N (input) integer
M (input) integer

Discussion:

IBITS returns bits N through M of I, right-justified with leading zeros.
Bits are numbered starting with zero at the left.

Integer Function IByte

Description:

IByte returns a value equal to the character set representation of a character at the indicated position.

Definition:

$$I = \text{IByte}(\text{STRING}, \text{PTR})$$

Arguments:

STRING (input) integer array
PTR (input) integer

Discussion:

The value of the character set associated with the character at position PTR of STRING is returned. Characters are numbered from the left starting at one.

Example:

If string has the value of "ABCDEF" and that character string uses a six-bit representation and corresponds to the octal string 202122232425 then
IByte(STRING,1)=16; IByte(STRING,2)=17 etc.

Integer Function IBYTE1

Description:

IBYTE1 returns the value of the first character in STRING.

Definition:

I = IBYTE1(STRING)

Arguments:

STRING (input) integer array

Discussion:

IBYTE1 is equivalent to IBYTE(STRING,1).

Integer Function ICNC1

Description:

Find the first character that is not the one specified.

Definition:

$I = ICNC1(BUF, BPTR, CH, LEN)$

Arguments:

BUF (input) integer array containing character string
BPTR (input) integer
CH (input) single character e.g. "A" or 1HA
LEN (input) integer

Discussion:

ICNC1 looks at BUF starting at position BPTR up to LEN and returns the index into BUF of the first character that is not the same as CH. IF all characters are the same as CH, or if LEN is less than BPTR, ICNC1 returns 0. Characters are numbered from the left starting at one.

Example:

ICNC1("ABCDE",3,"C",5) would return 4 since the fourth character in the character string "ABCDE" is the first character (when you begin looking at position three) not equal to "C". ICNC1("abbb",2,"C",4) would return 0.

Integer Function ICSC1

Description:

Find the first character that equals the one provided.

Definition:

$I = ICSC1(BUF, BPTR, CH, LEN)$

Arguments:

BUF (input) integer array containing character string
BPTR (input) integer
CH (input) single character e.g. "A" or 1HA
LEN (input) integer

Discussion:

ICSC1 looks at BUF starting at position BPTR up to LEN and returns the index into BUF of the first character that is the same as CH. If all characters are not equal to CH, or if LEN is less than BPTR, ICSC1 returns 0. Characters are numbered from the left starting at one.

Example:

ICSC1 ("ABCD" , 1, "C", 4) would return 3 since the third character of the BUF is a "c".

Integer Function IHASH

Description:

Hash a character string

Definition:

$J = \text{IHASH}(\text{CHAR}, I, \text{LEN})$

Arguments:

CHAR (input) character string
I (input) integer
LEN (input) integer

Discussion:

IHASH returns an evenly distributed number between 1 and 256.

Integer Function IOR

Description:

Perform the logical OR of two words

Definition:

$$K = \text{IOR} (I, J)$$

Arguments:

I (input) integer
J (input) integer

Discussion:

The logical OR of I and J is returned.

Integer Function ISCOMP

Description:

Character string compare

Definition:

I = ISCOMP (STR1,S1,STR2,S2, LEN)

Arguments:

STR1 (input) character string
S1 (input) integer
STR2 (input) character string
S2 (input) integer
LEN (input) integer

Discussion:

ISCOMP compares the string STR1 starting at S1 with STR2 starting at S2, for a length LEN. ISCOMP returns a value denoting their equivalence or their relative alphabetic ordering.. The value returned is:

-1 if STR1 < STR2
0 if STR1 = STR2
+1 if STR1 > STR2
0 if LEN < 1

Example:

ISCOMP ("AAAA",1,"ZZZZ",1,4) returns +1
ISCOMP ("AAAA",2,"BAACCD",2,2) returns 0

Integer Function ISHFTL

Description:

Shift left logical

Definition:

$J = \text{ISHFTL}(I, N)$

Arguments:

I (input) integer

N (input) integer

Discussion:

ISHFTL shifts I left N bits. The result is padded on the right with zeros.

Integer Function ISHFTR

Description:

Shift right logical

Definition:

$$J = \text{ISHFTR} (I, N)$$

Arguments:

I (input) integer
N (input) integer

Discussion:

ISHFTR shifts I right N bits. The result is padded on the left with zeros.

Subroutine ITOC

Description:

Convert integer character

Definition:

```
CALL ITOC(NUM,BUF,PTR,LEN)
```

Arguments:

NUM (input) integer
BUF (output) integer array
PTR (input) integer
LEN (input) integer

Discussion:

ITOC converts the integer NUM into a right-justified hollerith of length LEN and puts the Hollerith into BUF starting at PTR.

Example:

```
I = 5  
CALL ITOC (I,BUF,2,5 )  
would make the second through sixth characters of BUF become "bbbb5" where b  
= 1 blank.
```

Integer Function LEFT

Description:

Get the left half of a word

Definition:

$I = \text{LEFT}(W)$

Arguments:

W (input) integer

Discussion:

LEFT returns the left half of W as if it were an integer.

Subroutine LEFTI

Description:

Set a value in the left half of a word

Definition:

CALL LEFTI (W1,W2)

Arguments:

W1 (output) integer
W2 (input) integer

Discussion:

LEFTI sets the left halfword of W1 equal to the value of the right halfword of W2. W2 and the right halfword of W1 remain unchanged.

Integer Function RIGHT

Description:

Get the right halfword of a word

Definition:

$$I = \text{RIGHT}(W)$$

Arguments:

W (input) integer

Discussion:

RIGHT returns the right halfword of W as if it were an integer.

Subroutine RIGHTI

Description:

Set the right halfword

Definition:

CALL RIGHTI (W1,W2)

Arguments:

W1 (output) integer
W2 (input) integer

Discussion:

RIGHTI sets the right halfword of W1 equal to the right halfword of W2. The left halfword of W1 and W2 remain unchanged.

Subroutine SBITS

Description:

Set bits within a word

Definition:

CALL SBITS (I,N,M,J)

Arguments:

I (output)	integer
N (input)	integer
M (input)	integer
J (input)	integer

Discussion:

SBITS replaces bits N through M of I with the N-M+1 rightmost bits of J.
Bits are numbered from the left starting at 0.

Subroutine SMOVE

Description:

Move character strings

Definition:

CALL SMOVE (TO,TPOS,FROM,FPOS,LEN)

Arguments:

TO (output) character string e.g. integer array
TPOS (input) integer
FROM (input) character string e.g. integer array
FPOS (input) integer
LEN (input) integer

Discussion:

SMOVE replaces LEN number of characters in TO, starting with the character at TPOS. Characters are moved from FROM starting at FPOS.

Subroutine RANDCL

Description:

Close a file previously opened for random I/O

Definition:

```
CALL RANDCL(FDESG )
```

Arguments:

FDESG (input) integer

Discussion:

This routine closes a file previously opened for random I/O. The file to be closed is identified by FDESG.

Subroutine RANDOP

Description:

Open a file for random I/O.

Definition:

```
CALL RANDOP (LIONUM,FDESG )
```

Arguments:

LIONUM (input) integer
FDESG (output) integer

Discussion:

RANDOP opens the file associated with the logical unit number LIONUM, and returns an identifier FDESG. FDESG uniquely identifies the file, and is used by the other random I/O routines.

Subroutine RANDRD

Description:

Random read of a page from a random file

Definition:

```
CALL RANDRD(FDESG,PAGENO,BUFFER )
```

Arguments:

FDESG (input) integer
PAGENO (input) integer
PAGENO (input) integer
BUFFER (output) integer array

Discussion:

RANDRD reads a page from the random file identified by FDESG and puts the page in BUFFER.

Subroutine RANDRW

Description:

Rewrite a page to a random file

Definition:

CALL RANDRW (FDESG, PAGENO, BUFFER)

Arguments:

FDESG (input) integer
PAGENO (input) integer
BUFFER (output) integer array

Discussion:

This routine rewrites a page identified by PAGENO into the file identified by FDESG from BUFFER.

Subroutine RANDWT

Description:

Random write of a page

Definition:

Subroutine RANDWT(FDESG,PAGENO,BUFFER)

Arguments:

FDESG (input) integer
PAGENO (input) integer
BUFFER (input) integer array

Discussion:

This routine writes page PAGENO onto the file identified by FDESG from BUFFER.

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Energy Research & Development Administration, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately-owned rights.

NOTICE

Reference to a company or product name does not imply approval or recommendation of the product by the University of California or the U.S. Energy Research & Development Administration to the exclusion of others that may be suitable.

Printed in the United States of America
Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161
Price: Printed Copy \$; Microfiche \$3.00

Page Range	Domestic Price	Page Range	Domestic Price
001-025	\$ 3.50	326-350	10.00
026-050	4.00	351-375	10.50
051-075	4.50	376-400	10.75
076-100	5.00	401-425	11.00
101-125	5.50	426-450	11.75
126-150	6.00	451-475	12.00
151-175	6.75	476-500	12.50
176-200	7.50	501-525	12.75
201-225	7.75	526-550	13.00
226-250	8.00	551-575	13.50
251-275	9.00	576-600	13.75
276-300	9.25	601-up	*
301-325	9.75		

* Add \$2.50 for each additional 100 page increment from 601 to 1,000 pages;
add \$4.50 for each additional 100 page increment over 1,000 pages.

WC/crs