```
**********************************************************************************
**********************************************************************************
**
**
**
**    W   W   000   RRRR   K   K  EEEEE  RRRR
**    W   W  0   0  R   R  K  K   E      R   R
**    W W W  0   0  R   R  K K    E      R   R
**    W W W  0   0  RRRR   KK     EEEE   RRRR
**    W W W  0   0  R R    K K    E      R R
**    WW WW  C   0  R  R   K  K   E      R  R
**    W   W   000   R   R  K   K  EEEEE  R   R
**
**
**
**    U   U  PPPP    III   N   N  FFFFF   000     1      666         55555
**    U   U  P   P    I    NN  N  F      0   0   11     6            5
**    U   U  P   P    I    N N N  F      0   0    1     6            5555
**    U   U  PPPP     I    N N N  FFFF   0   0    1     6666            5
**    U   U  P        I    N N N  F      0   0    1     6   6           5
**    U   U  P        I    N  NN  F      0   0    1     6   6  ..    5   5
**     UUU   P       III   N   N  F       000    111    666   ..     555
**
**
**
**********************************************************************************
**********************************************************************************
```

```
*              UFD UPDATE INFORMATION FILE REV. 16.5
*
*
*       THIS UFD CONTAINS ALL SOFTWARE UPDATES GENERATED AFTER
*       THE LAST UPDATE DISK RELEASE WHICH WAS 16.4.
*       INFORMATION ABOUT ALL PREVIOUS UPDATE RELEASES SINCE
*       THE INITIAL RELEASE IS PRESENTED IN THIS FILE ALSO.
*       THE INITIAL REV. 16 RELEASE WAS 16.3.
*       TO UPDATE THOSE FILES
*       REQUIRED ON YOUR MASTER DISK, FUTIL COPY THE PROGRAM
*       REQUIRED TO THE UFD SPECIFIED IN THE TABLE UNDER THE -TO- COLUMN
*       AND USE UPXXX AS THE PROGRAM TO COPY AND THE NAME UNDER
*       THE NAME COLUMN AS THE NAME THE PROGRAM IS TO BE COPIED AS.
*
*       NOTE: ALL -TO- UFD'S MAY NOT EXIST ON YOUR
*       DISK IF YOU HAVE A 6 OR 12 MEG.BYTE DISK.
*
*
*        EXAMPLE:    UPDATE NO. NAME    TO
*                    ---------- ------  ------
*                    UP001      CPUT1   T&M
*
*                FUTIL
*                >FROM  'THIS UFD'      NOT NEEDED IF THIS IS HOME UFD
*                >TO T&M
*                >COPY UP001 CPUT1
*                >QU
*
*
*     NOTE:  >    EQUALS SUB-UFD IN -TO- COLUMN
*            NA   EQUALS NOT ASSIGNED
*
*       USED ON (UFDNAME)          DEFINITION
*       -----------------          ----------
*
*       8000        P8000          COBOL
*       8020        P8020          RJ2780
*       8060        P8060          RJCDC
*       8100        P8100          PRIMOS 4/5
*       8120        P8120          HASP300&400
*       8140        P8140          DBMS (DATABASE)
*       8150        P8150          RPG
*       8160        P8160          FORMS
*       8300        P8300          SPSS
*       8410        P8410          DPTX-DSC
*       8420        P8420          DPTX-TSF
*       8430        P8430          DPTX-TCF
*       8440        P8440          PRINET
*       8450        P8450          X.25
*       8520        P8520          BASICV
*
*
     SET TABS 12 21 46 58 66 75
UPDATE NO. NAME     TO                          SOURCE NO.  SCN NO. DATE    USED ON
```

*						
* REV. 16.4 APRIL 20,1979						
*						
UP001	DPTX-DSC	<M164B1>MFD (DIRECTORY)			042079	8410
UP002	DPTX-TSF	<M164B1>MFD (DIRECTORY)			042079	8420
UP003	DPTX-TCF	<M164B1>MFD (DIRECTORY)			042079	8430
UP004-UP014 SUPERCEEDED						
UP015	EDB	<M164A1>MFD (DIRECTORY)			042079	8100
UP016	EDB	<M164A1>CMDNC0 (RUN)			042079	8100
UP017-UP018 SUPERCEEDED						
UP019	MAGSR	<M164A1>MFD (DIRECTORY)			042079	8100
UP020	MAGRST	<M164A1>CMDNC0 (RUN)			042079	8100
UP021	MAGSAV	<M164A1>CMDNC0 (RUN)			042079	8100
UP022	MAGSAV	<M164B1>CMDNC0 (RUN)			042079	8100
UP023-UP040 SUPERCEEDED						
UP041	BASIC	<M164A1>MFD (DIRECTORY)			040278	8100
UP042	BASIC	<M164A1>CMDNC0 (RUN)			042079	8100
UP043	DBASIC	<M164A1>MFD (DIRECTORY)			042079	8100
UP044	DBASIC	<M164A1>CMDNC0 (RUN)			042079	8100
UP044-UP045 SUPERCEEDED						
UP047	PRINET	<M164B1>MFD (DIRECTORY)			042079	8440
UP048	X.25	<M164B1>MFD (DIRECTORY)			042079	8450
UP049	FIXRAT	<M164A1>MFD (DIRECTORY)			042079	8100
UP050	FIXRAT	<M164A1>CMDNC0 (RUN)			042079	8100
UP051	FIXRAT	<M164B1>CMDNC0 (RUN)			042079	8100
UP052-UP063 SUPERCEEDED						
UP064	ERRD.F	<M164A1>SYSCOM (SOURCE)			042079	8100
UP065	ERRD.P	<M164A1>SYSCOM (SOURCE)			042079	8100
UP066	SETSIZ	<M164A1>LIB7 (SOURCE)			042079	8100
UP067 SUPERCEEDED						
UP068	CPUT4	TMS400(SOURCE)	SRC1334.000	0217	042079	8100
UP069	C_CPUT4	TMS400 (COMMAND FILE)			042079	8100
UP070	CPUT4	T&M (RUN)			042079	8100
UP071-UP072 SUPERCEEDED						
UP073	PRMNT1	TMS400 (SOURCE)	SRC1326.003	0246	051079	8100
UP074	PRMNT1	T&M (RUN)			051079	8100
UP075	VTTYT1	TMS400 (SOURCE)	SRC1328.000	0208	042079	8100
UP076	VTTYT1	T&M (RUN)			042079	8100
UP077	C_VTTYT1	T&M (COMMAND FILE)			042079	8100
UP078-UP079 SUPERCEEDED						
UP080	P4WCST	TMS400 (SOURCE)	SCR1311.002	0216	042079	8100
UP081	P4WCST	T&M (RUN)			042079	8100
UP082	STLBT2	TMS400 (SOURCE)	SRC1313.004	0215	042079	8100
UP083	STLBT2	T&M (RUN)			042079	8100
UP084-UP085 SUPERCEEDED						
UP086	CRTT1	T&MSR1 (SOURCE)	SRC1324.002	0196	042079	8100
UP087	CRTT1	T&M (RUN)			042079	8100
UP088-UP089 SUPERCEEDED						
UP090	DISCT1	T&MSR1 (SOURCE)	SRC0787.011	0218	042079	8100
UP091	DISUFD	T&M (DIRECTORY)			042079	8100
UP092-UP096 SUPERCEEDED						
*						
*	REV. 16.5 JULY 24, 1979					

*						
UP097	MIDAS	<M165A1>MFD (DIRECTORY		254	072479	8100
UP098	KIDALB	<M165A1>LIB (BINARY)		254	072479	8100
UP099	KIDAFM	<M165A1>LIB (BINARY)		254	072479	8100
UP100	VKDALB	<M165A1>LIB (BINARY)		254	072479	8100
UP101	NVKDALB	<M164A1>LIB (BINARY)		254	072479	8100
UP102	K4000	<M165A1>SYSTEM (BINARY)		254	072479	8100
UP103	K2014A	<M165A1>SYSTEM (BINARY)		254	072479	8100
UP104	K2014B	<M165A1>SYSTEM (BINARY)		254	072479	8100
UP104A	IMIDAS	<M165A1>SYSTEM (BINARY)		254	072479	8100
UP105	CREATK	<M165A1>CMDNC0(RUN)		254	072479	8100
UP106	KBUILD	<M165A1>CMDNC0 (RUN)		254	072479	8100
UP107	KIDDEL	<M165A1>CMDNC0 (RUN)		254	072479	8100
UP108	REMAKE	<M165A1>CMDNC0 (RUN)		254	072479	8100
UP109	MCLUP	<M165A1>CMDNC0 (RUN)		254	072479	8100
UP110	C_MDLC1	TMS400 (COMMAND)		311	072479	8100
UP111	MDLCT1	TMS400 (SOURCE)	SRC1316.003	311	072479	8100
UP112	MDLCT1	T&M (RUN)		311	072479	8100
UP113	C_MDLC2	TMS400 (COMMAND)		259	072479	8100
UP114	MDLCT2	TMS400 (SOURCE)	SRC1317.002	259	072479	8100
UP115	MDLCT2	T&M (RUN)		259	072479	8100
UP116	C_MDLC3	TMS400 (COMMAND)		260	072479	8100
UP117	MDLCT3	TMS400 (SOURCE)	SRC1318.002	260	072479	8100
UP118	MDLCT3	T&M (RUN)		260	072479	8100
UP119	C_MDLC4	TMS400 (COMMAND)		261	072479	8100
UP120	MDLCT4	TMS400 (SOURCE)	SRC1319.002	261	072479	8100
UP121	MDLCT4	T&M (RUN)		261	072479	8100
UP122	C_MDLC5	TMS400 (COMMAND)		262	072479	8100
UP123	MDLCT5	TMS400 (SOURCE)	SRC1320.003	262	072479	8100
UP124	MDLCT5	T&M (RUN)		262	072479	8100
UP125	C_MDLC6	TMS400 (COMMAND)		314	072479	8100
UP126	MDLCT6	TMS400 (SOURCE)	SRC1321.004	314	072479	8100
UP127	MDLCT6	T&M (RUN)		314	072479	8100
UP128	C_MDLC7	TMS400 (COMMAND)		264	072479	8100
UP129	MDLCT7	TMS400 (SOURCE)	SRC1322.002	264	072479	8100
UP130	MDLCT7	T&M (RUN)		264	072479	8100
UP131	C_MDLC8	TMS400 (COMMAND)		315	072479	8100
UP132	MDLCT8	TMS400 (SOURCE)	SRC1323.004	315	072479	8100
UP133	MDLCT8	T&M (RUN)		315	072479	8100
UP134	C_MDLC9	TMS400 (COMMAND)		266	072479	8100
UP135	MDLCT9	TMS400 (SOURCE)	SRC1327.001	266	072479	8100
UP136	MDLCT9	T&M (RUN)		266	072479	8100
UP137	URCT1	T&MSR1 (SOURCE)	SRC0732.006	248	072479	8100
UP138	URCT1	T&M (RUN)		248	072479	8100
UP139	RTCT2	T&MSRC (SOURCE)	SRC0784.008	247	072479	8100
UP140	RTCT2	T&M (RUN)		247	072479	8100
UP141	HSSCT2	T&MSRC (SOURCE)	SRC0796.007	252	072479	8100
UP142	HSSCT2	T&M (RUN)		252	072479	8000
UP143	COBOL	<M165B1>MFD (DIRECTORY)			072479	8000
UP144	C4000	<M165A1>SYSTEM			072479	8000
UP145	C2014A	<M165A1>SYSTEM			072479	8000
UP146	C2014B	<M165A1>SYSTEM			072479	8000
UP146A	PXT1	<M166A1>TMS400 (SOURCE)	SRC1304.007	255	072479	8100
UP146B	PXT1	<M166A1>T&M (RUN)		255	072479	8100

[illegible]	[illegible]	<MI[illegible]AI>T&M	(SOURCE)	SRCI325.002	253	072479	8100
[illegible]	[illegible]	<MI[illegible]AI>T&M	(RUN)		253	072479	8100
[illegible]	[illegible]	<MI[illegible]AI>TMS400	(DIRECTORY)	SRCI253	256	072479	8100
[illegible]	[illegible]	<MI[illegible]AI>T&M	(RUN)		256	072479	8100
[illegible]	[illegible]	<MI[illegible]AI>TMS400	(SOURCE)	SRCI307.001	308	072479	8100
[illegible]	[illegible]	<MI[illegible]AI>T&M	(RUN)		308	072479	8100
[illegible]	CPUT4	<MI[illegible]AI>TMS400	(SOURCE)	SRCI334.001	309	072479	8100
[illegible]	CPUT4	<MI[illegible]AI>T&M	(RUN)		309	072479	8100
[illegible]	[illegible]	<MI[illegible]AI>TMS400	(SOURCE)	SRCI305.005	310	072479	8100
[illegible]	[illegible]	<MI[illegible]AI>T&M	(RUN)		310	072479	8100

* END OF UPDATE TABLE

* REASON FOR CHANGE

\- [illegible]TE 15 [illegible]

UP001 (CPTX-DSC) THIS IS A NEW PRODUCT FOR REV. 15.4 .

UP002 (CPTX-TSF) THIS IS A NEW PRODUCT FOR REV. 15.4 .

UP003 (CPTX-TCF) THIS IS A NEW PRODUCT FOR REV. 15.4 .

UP004 (BASICV) TO FIX LEADING SPACES IN "WRITE USING" FORMAT STRINGS. WHICH WERE PREVIOUSLY IGNORED.

UP007 (APPLIB) 15.4 FIXES A MINOR BUG FOUND IN [illegible] (MOVE SUBSTRING) WHICH CAUSED OVERLAPPING FIELDS TO BE MOVED INCORRECTLY.

UP008 (APPLIB) SEE UP007.

UP009 (VAPPLB) SEE UP007.

UP010 (RUNOFF) (1) TARS 25221 & 25222 INVOLVING PROBLEMS WITH [illegible]-
[illegible]ALIZATION COMMANDS PARTICULARLY ".DL"
(2) CAUSES ".SM" COMMAND TO TAKE EFFECT ON NEXT PAGE RATHER THAN WRITING ON EXTRA PAGE.
(3) NO LONGER SAVE A PLACE FOR THE PHANTOM HYPHENS IN THE TABLE OF CONTENTS.
(4) STACK FILE NAMES FOR ERROR MESSAGES CORRECTLY.

UP011 (RUNOFF) SEE UP010.

UP012 (SEG)(1) ALLOW TREE NAME IN QUOTES AS INPUT TO CMDSEG.
(2) REINITIALIZE DEFAULT MODE IN 64V.
(3) FLAG 64R WHEN IN "NEVER 64" MODE.
(4) SHARE A SPLIT MODULE LESS THAN 4000 (BASE 8) WORDS IN LENGTH AND DISPLAY STACK ADDRESS DURING THE LOAD.
(5) HANDLE AN INTEGER*4 COMMON BLOCK WITH THE DIMENSION [illegible].
NO TARS

UP013 (SEG) SEE UP012.

UP014 (SHARE4) SEE UP012.

UP015 (EDB)(1) FLAG SOURCE INPUT FILE AS A "BAD OBJECT FILE".
(2) GENET (OBSOLETE BUT STILL SUPPORTED) NOW WORKS.

*

UP016 (ECB) SEE UP015.

*

UP017 (LOAD)(1) TAR 25536 DEFERRED COMMOM ON A LIBRARY "COMMON" BLOCK BUG FIXED.
(2) ALLOWS LARGER COMMON REDEFINITION WHEN DEFERRED.

*

UP018 (LOAD) SEE UP017.

*

UP019 (MAGSR)

*

UP020 (MAGRST)(1) HANDLES THE CONDITION THAT "A NON DATA RECORD FOLLOWS A UFD TREE NAME RECORD".
(2) PRINT ERROR MESSAGE AND PAUSE WHEN A "DISC FULL" CONDITI OCCURS. (TAR 11969)
(3) PRINT PATHNAME OF THE FILE AT THE TIME AN "UNEXPECTED EO CONDITION OCCURS.
(4) SET READ/WRITE LOCK CORRECTLY. (TAR 10554)
(5) REMOVE "-LONG" FROM USAGE LINE. (TAR 22800)

*

UP021 (MAGSAV)(1) SAVE UFD WHICH HAS "READ ONLY" PERMISSION TO NON-OWNER AND FILES WITHIN THAT UFD WHICH PERMIT READ ACCESS TO NON-OWNER. PASSWORDS FOR THE SAVED UFD ARE SET TO NULL.
(2) WHEN PROGRAM ASKS FOR A NEW TAPE, PROGRAM CHECKS TO SEE IF THE NEW TAPE IS AT LOAD POINT. IF NOT, AND THE TAPE IS THE SECOND PHYSICAL REEL OF A LOGICAL TAPE, PROGRAM WILL QUERY USER TO SEE IF HE WANTS THE TAPE TO BE REWOUND. IF HIS ANSWER IS "YES", TAPE WILL BE REWOUND. IF THE ANSWER IS "NO", PROGRAM WILL ASK FOR A NEW TAPE UNIT.

*

UP022 (MAGSAV) SEE UP021.

*

UP023 (FTN) TAR 23673 GENERALIZED SUBSCRIPTS CAN GENERATE BAD CODE WHEN A VARIABLE IS SUBTRACTED FROM A CONSTANT.
TAR 25264 "LS" AND "RS" INTRINSICS GENERATE BAD CODE FOR NEGATIVE SHIFT COUNTS.
TAR 25561 THE COMPILER HANGS WHEN IN 64V MODE A STATEMENT FUNCTION IS PASSED AS AN OCTAL ARGUMENT.
WHEN A "$INSERT" FILE IS NOT FOUND, THE ERROR MESSAGE WILL NOT CONTAIN A SPURIOUS "T". THE "SHORTCALL" STATEMENT WORKS WITH LIBRARY CONVERSION FUNCTIONS. MINOR PROBLEMS IN PARSING ARRAY REFERENCES AND STATEMENT FUNCTIONS ARE FIXED. THE COMPILER USED TO GET THE EXCESS SUBSCRIPTS AND TOO FEW SUBSCRIPTS ERROR MESSAGES REVERSED.

*

UP024 (FTN) SEE INFO ON UP023.

*

UP025 (FTNOPT) ALL THE FIXES FOR "FTN" APPLY TO "FTNOPT" AS

WELL. OPTIMIZER PROBLEMS WHICH HAVE BEEN FIXED ARE:
-USE OF THE DO LOOP OPTIMIZER SOMETIMES PRODUCED LESS EFFICIENT CODE OUTSIDE LOOPS
-TEMPORARY VARIABLES INSIDE OPTIMIZED DO LOOPS WERE NOT ALWAYS FREED PROPERLY
-OPTIMIZED DO LOOPS OCCASIONALLY HAD BAD CODE FOR MIXED MODE ARITHMETIC.

*

UP026 (FTNOPT) SEE INFO ON UP025.

*

UP027 (COBOL)TO CORRECT TAR 25666. QUALIFIED DATA NAMES NOT OPERATING CORRECTLY.

*

UP028 (C4000) SEE UP027.

*

UP029 (C2014A) SEE UP027.

*

UP030 (C2014B) SEE UP027.

*

UP031 (FLIB6V) [F$IO]-FREE FORMAT COMPLEX INPUT DID NOT WORK FOR F$IO.

*

UP032 (VDSPK$) [TSRC$$]-"*>A" DID NOT WORKFOR TSRC$$.

*

UP033 (DOSPK$)
SEMLIB P300 CODE REMOVED. (TAR 81470)
TSRC$$ "*>A" DID NOT WORK.

*

UP034 (IFTNLB)
-P300 CODE REMOVED (TAR 81470)
-"*>A" DID NOT WRK

*

UP035 (PFTNLIB) SEE UP034.

*

UP036 (NPFTNLIB) SEE UP034.

*

UP037 (FTNLIB) SEE UP034.

*

UP038 (S4000) SEE UP034.

*

UP039 (S2014A) SEE UP034.

*

UP040 (S2014B) SEE UP034.

*

UP041 (BASIC) TARS 12546 & 80852 "PRINT USING" JUXTAPOSED ITEMS WHEN THE FIRST NUMERIC ITEMS OVERFLOWED.
TAR 13717 ".NL." DID NOT RESET THE COLUMN COUNT IN ENTER STATEMENT.
TAR 24728 STATEMENT NUMBER "0" WAS NOT SENSED AS AN ERROR.
TAR 15819 "PRINT USING" ROUNDING IS NOT CONSISTENT. MACHINE FLOATING ACCURACY IS THE PROBLEM HERE, BUT

NOTE THAT THE ACTUAL COMPUTATION ACCURACY IS NOT AFFECTED BY THIS PROBLEM, WHICH IS DUE TO THE INPUT CONVERSION IF ASCII DIGITS TO FLOATING NUMBERS. A BETTER METHOD IS USED BY BASIC/VM AND FORTRAN, SO THESE PROBLEMS WILL NOT SHOW UP.

TAR'S 80236 & 80469 "HALT" 'S ARE ENCOUNTERED WHEN STRINGS ARE PASSED TO A FORTRAN PROGRAM. THE DOCUMENTATION IS WRONG AND INDEED STRINGS ARE NOT ALLOWED TO BE PASSED TO A FORTRAN PROGRAM.

TAR 22783 A "FOR-NEXT" UNMATCHING ERROR WAS GENERATED WHEN IN FACT NO MISMATCH EXISTED.

*

UP042 (BASIC) SEE INFO ON UP041.

*

UP043 (DBASIC) SEE INFO ON UP041.

*

UP044 (DBASIC) SEE INFO ON UP041.

*

UP045 (PRI400)

BUG FIXES AT REV. 16.4

COMINPUT COMMAND

THE FILE UNIT SPECIFIED WAS IGNORED IF SPECIFIED AFTER A -OPTION. E.G., IF THE COMMAND 'CO -CONTINUE 7' WAS GIVEN, FILE UNIT 6 WAS USED. (TAR 80697)

FILUNT COLD START PARAMETER

IF A FILUNT PARAMETER WAS USED IN THE COLD START FILE, SPURIOUS RESULTS WOULD OCCUR.

ASSIGNED AMLC LINES

OUTPUT CHARACTERS COULD BE LOST WHEN UNASSIGNING AMLC LINES. (TAR 23415)

WTLIN$

DATE-TIME MODIFIED NOT UPDATED WHEN FILE ACCESSED WITH CALL TO WTLIN$.

SHARE

IT WAS NOT POSSIBLE TO SHARE AN ENTIRE SEGMENT. I.E., RESTORE FILE WHOSE START ADDR = 0 AND END ADDR = 177777 OCTAL. (TAR 10555)

COMOUTPUT

DID NOT GIVE ERROR MESSAGE IF FILE SPECIFIED WAS A DIRECTORY. COMMAND OF FORM "COMO TREENAME -C" WOULD NOT WORK.

-DUE TO A CONFLICT WITH PREVIOUSLY DEFINED HARDWARE DEVICE ADDRESSES, THE DEVICE ADDRESS OF THE PRIMENET NODE CONTROLLER (PNC) HAS BEEN CHANGED FROM '61 TO '07.

*

UP046 (PRIRUN) SEE UP045.

FILE.

*

UP047 (PRINET) FAM FOR REV. 16.4, THE FOLLOWING BUGS HAVE BEEN FIXED:

-ACCESSING SEGMENT DIRECTORIES VIA PATHNAME NOW WORKS. (I.E., SEG REMOTE_UFD>SUBUFD>#PROG)

-DUPLICATE RECEIVED MESSAGE BUG IS PROBABLY FIXED.

-LONG WRITE LINES NOW WORK WITH > 255 TRAILING SPACES.

-GROSS FLAG IS NOW RESET IN FAMCYL, (COULD GET LOCKED SET IN 16.2).

-FAM NOW ACCEPTS CD$ CODES TO WORK WITH PRIMENET CIRCUIT CLEARING CAUSES.

-THE INTERNAL VERSION NUMBER AND RECEIVE BLOCK SIZE PASSING HAS BEEN UPDATED TO CONFORM WITH 17.0'S EXPECTATIONS.

*

UP048 (X.25) NETCFG HAS BEEN FIXED FOR HETEROGENEOUS COMBINATIONS OF PRIMENET AND X.25 SOFTWARE IN THE SAME NETWORK. IT IS NO LONGER A REQUIREMENT THAT IF ANY NODE HAS THE X.25 SOFTWARE, THEY ALL MUST HAVE IT. TO SUPPORT THIS FEATURE THERE HAVE BEEN SOME INTERNAL CHANGES TO THE FORMAT OF THE CONFIGURATION FILE 'NETCON'.

*

UP049 (FIXRAT) UFD COMPRESSION FAILED TO WORK CORRECTLY.

*

UP050 (FIXRAT) SEE UP049.

*

UP051 (FIXRAT) SEE UP049.

*

UP052 (MIDAS)

MIDAS REV. 16.4

ABSTRACT

+

NEW AT REV 16.4, MIDAS UTILITY *MPACK SORTS DATA RECORDS BY PRIMARY KEY AND RECOVERS SPACE OCCUPIED BY DATA RECORDS WHICH HAVE BEEN MARKED FOR DELETION.

FOR REV 16 MIDAS FILES, *MPACK SORTS DATA RECORDS BY PRIMARY KEY AND RECOVERS SPACE OCCUPIED BY DATA RECORDS WHICH HAVE BEEN MARKED FOR DELETION. INDEXES ARE ALSO RESTRUCTURED SO THAT THEY OCCUPY AS LITTLE DISK SPACE AS POSSIBLE. *MPACK IS USEFUL FOR APPLICATIONS IN WHICH 1) DISK SPACE IS VERY LIMITED, AND/OR 2) RECORDS ARE OFTEN INSERTED AND DELETED FROM A MIDAS FILE.

*MPACK IS BUILT BY COMMAND FILE C_MPACK IN UFD MIDAS>SOURCE. NOTE THAT *MPACK IS BUILT IN UFD MIDAS>SOURCE, NOT CMDNC0, AND EXECUTES IN R-MODE ONLY. *MPACK HAS BASICALLY TWO OPTIONS. A MIDAS FILE MAY SIMPLY BE RESTRUCTURED. IN THIS CASE THE EXISTING FILE IS OVERWRITTEN WITH THE

RESTRUCTURED DATA. THE SECOND OPTION CAUSES THE RESTRUCTURED DATA TO BE WRITTEN TO A SECOND FILE, THUS PRESERVING THE ORIGINAL FILE. FIGURE 1 ILLUSTRATES HOW TO USE *MPACK. COMMENTS ARE ENCLOSED IN PARENTHESES AND USER INPUT IS UNDERLINED.

```
OK, R *MPACK
GO
[MPACK REV 16.4]
ENTER MIDAS FILE NAME: ACCT>MASTER (PATH NAME OF FILE TO BE )
+                      ----------
                                   (RESTRUCTURED.)
OK TO OVERWRITE THE FILE? NO       (SEE NOTE 1. )
+                         --
ENTER NEW MIDAS FILE NAME: FILE1   (PATH NAME OF FILE TO CONTAIN THE)
+                          -----
                                   (RESTRUCTURED INFORMATION.)
FILE ALREADY EXISTS. OK TO OVERWRITE? NO (SEE NOTE 2.)
+                                     --
ENTER NEW MIDAS FILE NAME: FILE2         (SEE NOTE 3.)
+                          -----

BEGIN PROCESSING INDEX 0 AT 11:22:00
ENTRIES INDEXED:        250

BEGIN PROCESSING INDEX 1 AT 11:26:27
ENTRIES INDEXED:         92

RESTRUCTURE COMPLETED AT 11:28:26
```

FIGURE 1

NOTES

1. THE NO RESPONSE INDICATES THAT THE RESTRUCTURED DATA SHOULD BE WRITTEN TO ANOTHER FILE. THE FILE, MASTER, WAS NOT MODIFIED.

2. THE NO RESPONSE INDICATES THAT THE MIDAS FILE, FILE1, SHOULD NOT BE USED. *MPACK ALSO VERIFIES THAT THE FILE IS A VALID MIDAS FILE. IF NOT VALID, *MPACK NOTIFIES THE USER AND REQUESTS A NEW PATH NAME.

3. SINCE FILE2 DID NOT EXIST, *MPACK CREATED IT.

```
*
UP053       (KICALB) SEE UP052.
*
UP054       (KIDAFM) SEE UP052.
*
UP055       (VKDALB) SEE UP052.
*
UP056       (NVKDALB) SEE UP052.
*
```

UP057 (K4000) SEE UP052.

*

UP058 (K2014A) SEE UP052.

*

UP059 (K2014B) SEE UP052.

*

UP060 (CREATK) SEE UP052.

*

UP061 (KBUILD) SEE UP052.

*

UP062 (KIDDEL) SEE UP052.

*

UP063 (REMAKE) SEE UP052.

*

UP064 (ERRD.F) ERROR CODE FOR DPTX.

*

UP065 (ERRD.P) SEE INFO ON UP064.

*

UP066 (SETSIZ) SETSIZ SOMETIMES WENT INTO AN INFINITE LOOP UNDER PRIMOS 2

*

UP067 (DBMS) THE FOLLOWING IS A LIST OF BUGS FIXED IN REV. 16.3. EXCEPT WHERE NOTED, THE BUGS WERE FIXED BASED ON INTERNAL ERRORS OR ERRORS THAT WERE REPORTED BY CMSI OVER THE PHONE AND THERE ARE NO TAR NUMBERS.

1) THE FOLLOWING PATCHES HAVE BEEN MADE TO DMLCP.
   A. THE SIZE OF THE INTERNAL RECORD AREA HAS BEEN EXPANDED FROM 8KB TO 32 KB TAR 24722.
   B. THE OPEN COMMAND WILL NOW ONLY OPEN AREAS SPECIFIED ON THE OPEN COMMAND RATHER THAN ALL AREAS.
   C. THE CLEAR ERROR COMMAND HAS BEEN FIXED SO THE SYSTEM WILL NOT HANG.
   D. THE 710F ERROR IN THE ROUTINE SETLST HAS BEEN FIXED.
   E. THE ROUTINE PUTLST HAS BEEN PATCHED SO THAT DUPLICATES WILL BE INSERTED IN THE PROPER ORDER.
   F. AFTER IMAGE LOGGING HAS BEEN PATCHED TO ACCOMIDATE BUCKETS LARGER THAN ONE (1) PAGE.
   G. R4VAL HAS BEEN PATCHED TO ACCOMIDATE LONG RETRIEVAL TRANSACTIONS.
2) CLUP HAS BEEN PATCHED SO THAT CERTAIN ERRORS WILL BE DISPLAYED ON THEIR TERMINAL WHEN THEY OCCUR.
3) DBACP HAS BEEN FIXED SO THAT IT MAY INITIALIZE A FILE LARGER THAN 32,000 BLOCK PROPERLY.

*

UP068 (CPLT4) TO REDUCE THE NUMBER OF TEST PROGRAMS. P400T2 & P500T1 ARE COMBINED IN AND ARE REPLACED BY THIS NEW TEST.

*

UP069 (C_CPUT4) SEE UP068

*

UP070 (CPUT4) SEE UP068.

*

UP071 (RTCT2) TO ENABLE THE TEST TO RUN ON A VCP AS WELL AS A SOC.

*

UP072 (RTCT2) SEE UP071.

*

UP073 (PRMNT1) ADDED TESTS IN ORDER TO TEST PARTS OF THE HARDWARE THAT WEREN'T PREVIOUSLY TESTED. TO HAVE COMPATIBILITY BETWEEN THE WIRE WRAP AND ETCH VERSIONS SO THAT THEY CAN RUN ON THE SAME PROGRAM.
DEVICE ADDRESS OF PRIMENET NODE CONTROLLER IS BEING CHANGED FROM '61 OT '07.
A BUG WAS FOUND WHEN TRYING TO LOAD THE A REGISTER WITH THE DEVICE ADDRESS PRIOR TO RUNNING THE PROGRAM.

*

UP074 (PRMNT1) SEE UP073.

*

UP075 (VTTYT1) THIS DIAGNOSTIC CHECKS OUT THE SERIAL INTERFACE CAPABILITIES OF THE VCP V.I.A. PFO. THIS TEST OPERATED SIMILARLY TO TTYT2.

*

UP076 (VTTYT1) SEE UP075.

*

UP077 (URCT1) SUPPORT OF VRC / DECISION DATA CARD PROCESSOR.

*

UP078 (URCT1) SEE UP077.

*

UP079 (P4WCST) TEST FAILED IF THERE WERE LESS THAN 64K OF MEMORY.

*

UP080 (P4WCST) SEE UP079.

*

UP081 (STLBT2) TO ACCOMMODATE THE P750.

*

UP082 (STLBT2) SEE UP081.

*

UP083 (PXT1) TO FIX STRING PROBLEM.

*

UP084 (PXT1) SEE UP083.

*

UP085 (CRTT1) (1) TO ADD A ROUTINE TO CHECK THE ABILITY FO THE DEVICE TO TRANSMIT ON REQUEST OF THE HOST CPU AND CHECK THE INTEGRITY OF THE TERMINALS OWN MEMORY.
(2) TO CONDENSE THE WHOLD TEST INTO A SMALLER PACKAGE WHILE IMPROVING THE EFFECTIVENESS FO THE WHOLE TEST.
(3) TO REMOVE POSSIBLE BUG WHERE AMLC IS SHUTDOWN BEFORE IT HAS TIME TO CLEAR DEDICATED PELL.

*

UP086 (CRTT1) SEE UP085.

*

UP087 (AMLCT5) TO INCORPORATE TIMING CHANGES CAUSED BY THE VCP.

*

UP088 (AMLCT5) SEE UP087.

*

UP089 (DISCT1) TO INCORPORATE TIMING CHANGES CAUSED BY THE VCP.

*

*
*
UP092-UP096 (SPOOL) BETTER "QUEUE FULL" ERROR MESSAGE. (TAR 22414)
(2) HASP CONTROL ON SERIAL PRINTER. (TAR 23467)
*

MIDAS REV. 16.5

ABSTRACT

CONCURRENT PROCESS HANDLING AND THE DETECTION AND CORRECTION OF CONCURRENCY ERRORS ARE THE TWO MAJOR AREAS OF MODIFICATION IN MIDAS AT REV 16.5. DESIGNED TO PROVIDE A SUBSTANTIAL PERFORMANCE IMPROVEMENT, THE NEW CONCURRENT PROCESS HANDLING METHOD WILL REQUIRE MODIFICATION OF FORTRAN AND PMA MIDAS APPLICATION PROGRAMS. THE NEW METHOD IS AVAILABLE TO COBOL USERS AT THIS RELEASE, TO BASIC USERS AT REV 16.6, AND TO RPG II USERS AT REV 17.1. USERS MAY EASILY DISABLE THE NEW METHOD AND, AS A RESULT, EMPLOY THE CONCURRENT PROCESS HANDLING METHOD AVAILABLE IN PREVIOUS RELEASES. NOTE THAT USERS WITH APPLICATIONS WHICH ACCESS MIDAS FILES OVER PRIMENET MUST DISABLE THE NEW CONCURRENT PROCESS HANDLING METHOD.

THE SECOND CHANGE, INDEPENDENT OF THE FIRST, ALLOWS MIDAS IN MOST CASES TO DETECT AND CORRECT CONCURRENCY ERRORS.

SECTION 2 OF THE PE-T DISCUSSES THE NEW CONCURRENT PROCESS HANDLING METHOD AND ITS IMPACT ON USER APPLICATIONS AND OPERATIONS. SECTION 3 DESCRIBES HOW MIDAS DETECTS AND CORRECTS CONCURRENCY ERRORS. INSTALLATION METHODS AND CONSIDERATIONS ARE DISCUSSED IN SECTION 4.

TABLE OF CONTENTS

1 INTRODUCTION....3

2 HANDLING OF CONCURRENT MIDAS PROCESSES....5
2.1 OVERVIEW....5
2.2 IMPLEMENTATION METHOD....5
2.3 APPLICATION IMPLICATIONS....6
2.3.1 USER OPTIONS....6
2.3.2 APPLICATION PROGRAM MODIFICATIONS....7
2.3.2.1 NTFYM$....8
2.3.2.2 OPENM$....10
2.3.2.3 CLOSM$....11
2.3.3 EXAMPLES....12
2.3.3.1 USE OF NTFYM$....12
2.3.3.2 USE OF OPENM$ AND CLOSM$....13
2.3.4 ADMINISTRATION CHANGES....14
2.3.4.1 OVERVIEW....14
2.3.4.2 MIDAS INITIALIZATION -- IMIDAS....15
2.3.4.3 MIDAS CLEANUP UTILITY -- MCLUP....16

3 RECOVERY FROM CONCURRENCY ERRORS....17
3.1 OVERVIEW....17
3.2 IMPLEMENTATION OF CONCURRENCY ERROR DETECTION AND RECOVERY....17
3.2.1 COMMUNICATION ARRAY FORMAT....17
3.3 LIMITATIONS....18

4 INSTALLATION OF MIDAS....19
4.1 COMMAND FILES....19
4.2 MODIFYING THE SHARED LOCK AND SEMAPHORE VALUES....19
4.3 DISABLING THE NEW CONCURRENT PROCESS HANDLING METHOD....19
4.4 NETWORK USERS....20
4.5 MIDAS FILE READ/WRITE LOCKS....20
4.6 RELOADING APPLICATION PROGRAMS....20

1 INTRODUCTION

MIDAS AT REV 16.5 OFFERS FORTRAN AND PMA USERS TWO INDEPENDENT IMPROVEMENTS. FIRST, MANY USER APPLICATIONS MAY BE ABLE TO OPERATE SUBSTANTIALLY FASTER. TABLES 1.1 AND 1.2 SHOW SOME SAMPLE DATA. THE TEST PROGRAM PROCESSED A SINGLE MIDAS FILE CONTAINING 500 RECORDS. EACH RECORD WAS THE CONCATENATION OF FOUR ASCII TEN CHARACTER KEYS. FOR EACH RECORD, THE PROGRAM:

1) READ NEXT RECORD (OR FIRST) VIA PRIMARY KEY,
2) FOR EACH SECONDARY INDEX:
   2A) READ THE RECORD VIA THE SECONDARY KEY,
   2B) DELETED THE CURRENT KEY VALUE,
   2C) RE-INSERTED THE KEY VALUE.

THE PERFORMANCE DATA WERE OBTAINED ON A P-650 WITH 1024K BYTES OF MEMORY. MIDAS PROCESSES EXECUTED WITH THE FAM AND SPOOL PROCESSES AND A TERMINAL PROCESS. DATA IN TABLE 1.1 WERE OBTAINED FROM PROCESSES OPERATING CONCURRENTLY ON THE SAME MIDAS FILE. TABLE 1.2 SHOWS RESPONSE TIMES FOR CONCURRENT PROCESSES EXECUTING THE SAME TEST PROGRAM BUT OPERATING ON DIFFERENT COPIES OF THE SAME DATA.

NUMBER OF CONCURRENT PROCESSES	MIDAS RELEASE REV 16.4	REV 16.5
1	0.7	0.4
2	2.2	0.8
3	3.7	1.2
4	5.1	1.6
5	6.9	2.0
6	---	2.5
7	---	3.0

TABLE 1.1 -- AVERAGE RESPONSE TIME PER RECORD PROCESSED (SECONDS)
PROCESSES OPERATING ON THE SAME MIDAS FILE

NUMBER OF CONCURRENT PHANTOMS	MIDAS RELEASE REV16.4	REV16.5 TEST UNMODIFIED	REV16.5 TEST MODIFIED
1	0.7	---	0.4
2	1.8	2.0	1.0
3	3.2	3.6	1.9
4	4.8	5.3	2.9
5	5.7	7.7	3.9
6	7.5	9.5	5.8
7	9.0	13.8	8.0
8	9.3	19.6	10.3
9	12.5	---	11.8
10	15.5	---	13.4
11	21.0	---	14.8

TABLE 1.2 -- AVERAGE RESPONSE TIME PER RECORD PROCESSED (SECONDS)
PROCESSES OPERATING ON DIFFERENT FILES.

DATA FOR COLUMN TWO OF TABLE 1.1 AND COLUMN THREE OF TABLE 1.2 WAS OBTAINED BY MODIFYING THE TEST PROGRAM TO CALL THE NEW MIDAS USER INTERFACE ROUTINES, OPENM$ AND CLOSM$ RATHER THAN SRCH$$.

TO OBTAIN THIS PERFORMANCE INCREASE, MIDAS NOW USES A DIFFERENT METHOD OF HANDLING CONCURRENT PROCESSES. THIS NEW METHOD, HOWEVER, WILL REQUIRE CHANGES IN FORTRAN AND PMA AND APPLICATION PROGRAMS IN ORDER FOR THE PROGRAMS TO OBTAIN THE PERFORMANCE INCREASE. COBOL PROGRAMS, HOWEVER, REQUIRE NO CHANGES. USER OPTIONS ARE DETAILED IN SECTION 2.3.1. NOTE THAT UNMODIFIED PROGRAMS WILL STILL OPERATE AND THAT PROGRAMS NEED NOT ALL BE MODIFIED AT THE SAME TIME. HOWEVER, ALL FORTRAN AND PMA PROGRAMS WHICH USE THE UNSHARED MIDAS LIBRARIES (KIDALB AND NVKDALB) MUST BE RELOADED WHETHER OR NOT THE PROGRAMS ARE MODIFIED. COBOL PROGRAMS WHICH USE THE UNSHARED COBOL AND/OR MIDAS LIBRARIES MUST ALSO BE RELOADED.

THE SECOND IMPROVEMENT IN MIDAS IS COMPLETELY INDEPENDENT OF THE FIRST AND REQUIRES NO CHANGES IN APPLICATION PROGRAMS. MIDAS WILL NOW DETECT AND CORRECT CONCURRENCY ERRORS. THESE ERRORS MAY OCCUR WHEN THE POSITION OF A PROCESS IN A MIDAS FILE IS MODIFIED BY THE ACTION OF A CONCURRENT PROCESS. THE ONLY CASE THAT APPLICATION PROGRAMS MUST BE ABLE TO HANDLE OCCURS WHEN A PROCESS ATTEMPTS TO OPERATE ON ITS 'CURRENT RECORD' (EG. UPDATE IT) AND A CONCURRENT PROCESS HAS DELETED THE RECORD. IN THIS SPECIAL CASE MIDAS WILL DETECT THE 'ERROR' AND RETURN A STATUS CODE OF 13, WHICH NOW HAS A DIFFERENT MEANING FOR ERROR RECOVERY THAN STATUS CODE 13 AT REV 16.4.

2 HANDLING OF CONCURRENT MIDAS PROCESSES

2.1 OVERVIEW

IN ORDER TO PROVIDE INCREASED PERFORMANCE, MIDAS NOW EMPLOYS A METHOD OF HANDLING CONCURRENT PROCESSES WHICH DIFFERS FROM PREVIOUS RELEASES. IN THE PAST MIDAS COORDINATED CONCURRENT PROCESSES BY GATING PROCESSES AT THE SEGMENT SUBFILE LEVEL (EG. A MIDAS FILE INDEX). THIS METHOD RELIED UPON FILE SYSTEM READ/WRITE LOCKS AND REQUIRED THAT SEGMENT SUBFILES BE OPENED AT THE START OF EACH MIDAS FILE OPERATION AND CLOSED UPON COMPLETION OF THE OPERATION. FOR EXAMPLE, TO RETRIEVE A RECORD, MIDAS OPENED THE INDEX SEGMENT SUBFILE(S) AND THE DATA SEGMENT SUBFILE. WHEN THE RETRIEVAL COMPLETED, MIDAS CLOSED THESE SEGMENT SUBFILES.

THE NEW CONCURRENT PROCESS HANDLING METHOD PROVIDES IMPROVED PERFORMANCE BY GREATLY REDUCING THE NUMBER OF FILE SYSTEM CALLS. THROUGH USE OF A SEMAPHORE AND A "LOCK" IN SHARED MEMORY, MIDAS SIMPLY ALLOWS ONLY ONE PROCESS AT A TIME TO EXECUTE A MIDAS FILE OPERATION. THEREFORE, MIDAS SEGMENT SUBFILES NEED NOT BE CLOSED AT THE END OF EACH OPERATION ONLY TO BE REOPENED AT THE START OF THE NEXT CALL. DETAILS OF THE NEW METHOD ARE DESCRIBED IN SECTION 2.2.

THE NEW METHOD OF HANDLING CONCURRENT PROCESSES REQUIRES THAT MIDAS BE NOTIFIED BOTH WHEN A PROCESS IS TO BEGIN USING A MIDAS FILE AND WHEN THE PROCESS HAS COMPLETED OPERATIONS ON THE FILE. FOR FORTRAN AND PMA USERS OF THE MIDAS CALL LEVEL INTERFACE, THIS REQUIREMENT MEANS THAT APPLICATION PROGRAMS MUST BE MODIFIED. SECTION 2.3 DESCRIBES METHODS OF MAKING THESE CHANGES. IMPORTANT INSTALLATION INSTRUCTIONS ARE DETAILED IN SECTION 4. IT SHOULD BE NOTED THAT PRIMENET USERS AND USERS WHO DO NOT WISH TO MAKE APPLICATION PROGRAM CHANGES MAY DISABLE THE NEW METHOD OF HANDLING CONCURRENT PROCESSES AND THUS RETURN TO THE METHOD EMPLOYED BY PREVIOUS MIDAS RELEASES. THE PROCEDURE FOR DISABLING THE NEW METHOD IS DESCRIBED IN SECTION 4.3.

2.2 IMPLEMENTATION METHOD

TO MAINTAIN FILE INTEGRITY, MIDAS MUST SYCHRONIZE CONCURRENT PROCESSES. IN PREVIOUS RELEASES OF MIDAS, THIS SYNCHRONIZATION WAS ACCOMPLISHED BY OPENING FILE SEGMENTS FOR READING AND WRITING. SINCE FILE READ/WRITE LOCKS WERE SET TO 2 (N READERS AND ONE WRITER), ONLY ONE PROCESS COULD ACCESS A FILE SEGMENT AT A TIME. A SECOND PROCESS WAS ONLY ABLE TO PROCEED WHEN THE FIRST PROCESS FINISHED ITS MIDAS OPERATION AND THE FILE SEGMENTS WERE CLOSED. THIS METHOD OF SYNCHRONIZATION REQUIRED MANY CALLS TO THE FILE SYSTEM ROUTINE SRCH$$ TO OPEN AND CLOSE FILE SEGMENTS AND THUS IMPOSED A SIGNIFICANT PERFORMANCE PENALTY.

IN THIS RELEASE MIDAS DOES NOT CLOSE FILE SEGMENTS BETWEEN MIDAS OPERATIONS. THIS, HOWEVER, REQUIRES THAT MIDAS FILE READ/WRITE LOCKS BE SET TO 3 (N READERS AND M WRITERS). OTHERWISE, CONCURRENT PROCESSES WOULD BE UNABLE TO OPEN A FILE SEGMENT WHICH HAD BEEN

ALREADY OPENED BY ANOTHER PROCESS. NOTE THAT IN ALL PAST AND PRESENT RELEASES, MIDAS MAY WRITE INTO A FILE ON BEHALF OF A USER-LEVEL READ REQUEST.

WITH FILE READ/WRITE LOCKS SET TO 3, FILE INTEGRITY COULD BE DESTROYED. THIS WOULD HAPPEN, FOR INSTANCE, IF TWO PROCESSES BOTH READ THE SAME RECORD AND THEN BOTH UPDATE THE RECORD. IN THIS CASE THE FIRST UPDATE WOULD BE LOST. TO PREVENT LOSS OF FILE INTEGRITY, MIDAS EMPLOYS A METHOD OF HANDLING CONCURRENT PROCESSES WHICH DOES NOT DEPEND ON OPENING AND CLOSING FILE UNITS.

IN THE NEW METHOD WHEN MIDAS IS CALLED, A CHECK IS DONE TO SEE IF ANY OTHER PROCESS IS USING MIDAS. TO DO THIS CHECK, MIDAS TESTS A "LOCK" LOCATED IN A SHARED MEMORY SEGMENT. A ZERO VALUE INDICATES THAT MIDAS IS AVAILABLE. IF NON-ZERO, THE LOW ORDER 15 BITS IS THE USER NUMBER OF THE PROCESS CURRENTLY ACCESSING MIDAS. (NOTE: BIT ONE IS ALWAYS SET WHEN MIDAS IS IN USE.) WHEN THE RESULT OF THE LOCK TEST IS ZERO, THE LOCK IS SET TO INDICATE THAT THE CURRENT PROCESS (DOING THE CHECK) NOW HAS SOLE ACCESS TO MIDAS. THIS "TEST AND SET" OPERATION IS NON-INTERRUPTIBLE. THEREFORE A PROCESS CANNOT MODIFY THE LOCK VALUE BETWEEN THE TIME THAT ANOTHER PROCESS HAS TESTED AND SET THE LOCK VALUE. IF THE TEST AND SET OPERATION IS SUCCESSFUL, THE PROCESS IS SAID TO HAVE "OBTAINED" THE LOCK.

IF WHEN TESTED, THE LOCK IS NON-ZERO, THE TESTING PROCESS MUST WAIT UNTIL MIDAS BECOMES AVAILABLE. TO ACCOMPLISH THIS, THE PROCESS IS SUSPENDED AND PUT ON A SEMAPHORE WAIT LIST. THE WAIT LIST FORMS A QUEUE OF PROCESSES WAITING TO BEGIN A MIDAS OPERATION. EACH TIME AN OPERATION COMPLETES, THE LOCK IS RELEASED, IE. THE LOCK VALUE IS SET TO ZERO. A PROCESS IS THEN REMOVED FROM THE WAIT LIST. THE RESTARTED PROCESS AGAIN MUST ATTEMPT TO OBTAIN THE LOCK.

2.3 APPLICATION IMPLICATIONS

2.3.1 USER OPTIONS

A USER HAS TWO BASIC OPTIONS WITH THE NEW MIDAS RELEASE.

1) THE USER MAY DISABLE THE NEW METHOD OF CONCURRENT PROCESS HANDLING AND MAKE NO APPLICATION PROGRAM CHANGES. ALTHOUGH THERE WOULD BE NO PERFORMANCE GAIN, THE DETECTION AND CORRECTION OF CONCURRENCY ERRORS WOULD STILL OCCUR. NOTE THAT THIS IS THE ONLY OPTION AVAILABLE TO PRIMENET USERS.

2) THE USER MAY MODIFY SOME OR ALL APPLICATION PROGRAMS IN ORDER TO SELECTIVELY OBTAIN A PERFORMANCE IMPROVEMENT. UNMODIFIED PROGRAMS AUTOMATICALLY USE THE NEW METHOD OF HANDLING CONCURRENT PROCESSES BUT MAY SUFFER SOME PERFORMANCE DEGRADATION.

2.3.2 APPLICATION PROGRAM MODIFICATIONS

WHEN MIDAS IS INSTALLED, USERS MUST RELOAD ALL APPLICATION PROGRAMS WHICH USE AN UNSHARED MIDAS LIBRARY. IN ADDITION, TO OBTAIN THE POTENTIAL PREFORMANCE INCREASE, USERS MUST MODIFY FORTRAN AND PMA MIDAS APPLICATION PROGRAMS. THE MODIFICATIONS INVOLVE INSERTING SUBROUTINE CALLS TO NOTIFY MIDAS THAT FILE SEGMENTS ARE NOT TO BE CLOSED BETWEEN CALLS TO MIDAS. NOTE THAT NOT ALL APPLICATIONS NEED BE MODIFIED AT THE SAME TIME.

USERS MAY CHOOSE FROM TWO METHODS OF PROGRAM MODIFICATION. THE FIRST METHOD INVOLVES INSERTING CALLS TO SUBROUTINE NTFYM$. THE FIRST CALL SHOULD BE INSERTED FOLLOWING THE CALL TO OPEN THE MIDAS FILE BUT BEFORE THE FIRST MIDAS FILE OPERATION. THE OTHER CALL TO NTFYM$ SHOULD BE INSERTED JUST BEFORE THE CALL TO CLOSE THE MIDAS FILE. NTFYM$ NOTIFIES MIDAS THAT A MIDAS FILE HAS JUST BEEN OPENED OR IS ABOUT TO BE CLOSED. FOR FURTHER DETAILS REFER TO THE SECTION WHICH DESCRIBES SUBROUTINE NTFYM$.

THE SECOND METHOD IS TO REPLACE THE CALLS WHICH OPEN AND CLOSE A MIDAS FILE WITH CALLS TO OPENM$ AND CLOSM$ RESPECTIVELY. SUBROUTINE OPENM$ OPENS A MIDAS FILE AND THEN CALLS NTFYM$. CLOSM$ CALLS SUBROUTINE NTFYM$ AND THEN CLOSES A MIDAS FILE. DETAILS ARE PROVIDED IN THE SECTIONS WHICH DESCRIBE OPENM$ AND CLOSM$.

MIDAS SUPPORTS R MODE APPLICATIONS. HOWEVER, BECAUSE THE R MODE MIDAS LIBRARY ENTERS V MODE TO DO A PORTION OF THE CONCURRENT PROCESS HANDLING, MIDAS WILL NOT WORK ON A PRIME P-300.

2.3.2.1 NTFYM$

****
*   *
* NTFYM$ *
*   *
****

FUNCTION

NOTIFY MIDAS THAT A MIDAS FILE (SEGMENT DIRECTORY) HAS BEEN OPENED OR IS ABOUT TO BE CLOSED BY THE USER.

CALLING SEQUENCE

```
CALL NTFYM$ (KEY, UNIT, STATUS)
```

KEY -- (INPUT) SPECIFIES WHETHER THE FILE HAS BEEN OPENED OR IS ABOUT TO BE CLOSED.
1 - FILE HAS BEEN OPENED
2 - FILE IS ABOUT TO BE CLOSED

UNIT -- (INPUT) FILE UNIT ON WHICH THE FILE IS OPEN

STATUS -- (OUTPUT) ERROR STATUS
0 - NO ERROR
10001 - BAD PARAMETER
10002 - TOO MANY MIDAS FILES OPEN SIMULTANEOUSLY MAY OCCUR ONLY IF KEY IS 1. MAXIMUM NUMBER OF FILES IS 129. SEE PARAMETER MFILES IN FILE KPARAM.

DISCUSSION

1. A CALL TO NTFYM$ AFTER A MIDAS FILE HAS BEEN OPENED NOTIFIES MIDAS THAT IT SHOULD LEAVE OPEN BETWEEN MIDAS CALLS ANY OF THE SPECIFIED FILE'S SEGMENT SUBFILES WHICH IT OPENS DURING SUBSEQUENT FILE ACCESS.

2. A CALL TO NTFYM$ BEFORE A MIDAS FILE IS CLOSED NOTIFIES MIDAS THAT IT SHOULD CLOSE ANY OF THE FILE'S SEGMENT SUBFILES THAT IT HAS LEFT OPEN.

3. IF THE MIDAS LIBRARY HAS BEEN CUSTOMIZED TO DISABLE INTERNAL LOCKING, A CALL TO NTFYM$ HAS NO EFFECT.

4. NTFYM$ IS MOST USEFUL IN THOSE APPLICATIONS WHICH OPEN AND CLOSE ALL TYPES OF FILES VIA THE SAME CALLS TO THE FILE SYSTEM. IN THESE APPLICATIONS IT IS PROBABLY SIMPLEST TO INSERT CALLS TO NTFYM$ RATHER THAN GENERATE A SEPARATE FILE SYSTEM CALL FOR EACH TYPE OF FILE. (EG. SAM, DAM, MIDAS, ETC.)

5. NOTE THAT MIDAS DOES NOT VERIFY THAT THE FILE REFERENCED IN THE CALL TO NTFYM$ IS A MIDAS FILE. A FILE SYSTEM ERROR CODE MAY RESULT IF THE REFERENCED FILE IS NOT A MIDAS FILE.

2.3.2.2 OPENM$

```
**********
*        *
* OPENM$ *
*        *
**********
```

FUNCTION

OPENS A MIDAS FILE (SEGMENT DIRECTORY) AND, UNLESS THE MIDAS LIBRARY HAS BEEN CUSTOMIZED TO DISABLE INTERNAL LOCKING, CAUSES MIDAS TO LEAVE OPEN BETWEEN MIDAS CALLS ANY OF THE FILE'S SEGMENT SUBFILES WHICH IT OPENS DURING SUBSEQUENT FILE ACCESS. OPENM$ VERIFIES THAT THE SPECIFIED FILE EXISTS AND THAT IT IS OF THE APPROPRIATE TYPE, IE. SAM SEGMENT DIRECTORY.

CALLING SEQUENCE

```
CALL OPENM$ (KEY, TRENAM, NAMLEN, UNIT, STATUS)
```

KEY -- (INPUT) VALID SRCH$$ ACTION SUB-KEY (K$READ, K$WRIT, OR K$RDWR, OPTIONALLY TOGETHER WITH K$GETU)

TRENAM -- (INPUT) TREE NAME OF FILE TO BE OPENED

NAMLEN -- (INPUT) LENGTH OF TREE NAME IN CHARACTERS

UNIT -- (INPUT) IF K$GETU IS NOT SPECIFIED, THEN UNIT IS THE FILE UNIT ON WHICH THE FILE IS TO BE OPENED.
(OUTPUT) IF K$GETU IS SPECIFIED, UNIT IS THE FILE UNIT ON WHICH THE FILE WAS OPENED.

STATUS -- (OUTPUT) ERROR STATUS

0	- NO ERROR
< 10001	- FMS ERROR (SYSTEM DEFINED)
= 10001	- BAD KEY
= 10002	- TOO MANY MIDAS FILES OPEN THE LIMIT IS 129. SEE PARAMETER MFILES IN FILE KPARAM. SIMULTANEOUSLY
= 10003	- SPECIFIED FILE IS NOT A MIDAS SEGMENT DIRECTORY

2.3.2.3 CLOSM$

```
**********
*        *
* CLOSM$ *
*        *
**********
```

FUNCTION

CLOSES A MIDAS FILE (SEGMENT DIRECTORY) OPEN ON A SPECIFIED FILE UNIT AND, UNLESS THE MIDAS LIBRARY HAS BEEN CUSTOMIZED TO DISABLE INTERNAL LOCKING, CLOSES ANY OF THE FILE'S SEGMENT SUBFILES WHICH MIDAS HAS OPENED DURING THE COURSE OF FILE ACCESS.

CALLING SEQUENCE

```
CALL CLOSM$ (UNIT, CODE)
```

UNIT -- (INPUT) FILE UNIT ON WHICH THE MIDAS FILE IS OPEN

CODE -- (OUTPUT) ERROR STATUS
= 0 - NO ERROR
> 0 - FMS ERROR (SYSTEM DEFINED)

2.3.3 EXAMPLES

2.3.3.1 USE OF NTFYM$

IN THIS FORTRAN EXAMPLE THE PROGRAM OPENS FILE FNAME ON UNIT UNIT. VARIABLE TYPE HAS PREVIOUSLY BEEN SET TO A VALUE WHICH DESCRIBES THE TYPE OF FILE OPENED. IF THE FILE IS OF TYPE "MIDAS", THE PROGRAM CALLS NTFYM$ TO NOTIFY MIDAS THAT IT IS READY TO BEGIN OPERATIONS ON THE FILE. AFTER PROCESSING HAS BEEN COMPLETED, THE PROGRAM NOTIFIES MIDAS OF THE FACT AND THEN CLOSES THE FILE. NOTE THAT NTFYM$ IS USED HERE BECAUSE SEVERAL TYPES OF FILES MAY BE OPENED BY THE CALL TO SRCH$$. NTFYM$ SHOULD ONLY BE CALLED FOR MIDAS FILES.

```
C       OPEN THE FILE
        CALL SRCH$$(K$READ,FNAME,6,UNIT,FTYPE, CODE)
        IF (CODE .NE. 0) GO TO 9000
        IF (TYPE .NE. MIDAS) GO TO 200/* CHECK FILE TYPE
        CALL NTFYM$(1,UNIT,CODE) /* TELL MIDAS WE'RE READY
        IF (CODE .NE. 0) GO TO 9002
200     CONTINUE
        .
        .
        .
C       DO MIDAS FILE PROCESSING (EG. CALLS TO FIND$)
        .
        .
        .
        IF (TYPE .NE. MIDAS) GO TO 800
        CALL NTFYM$(2,UNIT,CODE)  /* TELL MIDAS PROCESSING IS D
ONE
800     CONTINUE
        CALL SRCH$$(K$CLOS,0,0,UNIT,TYPE,CODE) /* CLOSE FILE
        .
        .
        .
```

2.3.3.2 USE OF OPENM$ AND CLOSM$

THIS PROGRAM USES OPENM$ TO OPEN FILE FNAME ON UNIT UNIT AND AT THE SAME TIME NOTIFY MIDAS THAT PROCESSING IS ABOUT TO BEGIN. AFTER PROCESSING HAS BEEN COMPLETED, THE PROGRAM CALLS CLOSM$ TO NOTIFY MIDAS THAT PROCESSING HAS BEEN COMPLETED AND TO CLOSE THE FILE. THE USE OF OPENM$ AND CLOSM$ IS CONVENIENT WHEN ONE KNOWS THAT ONLY MIDAS TYPE FILES ARE BEING OPENED OR CLOSED.

```
C       OPEN THE FILE AND NOTIFY MIDAS THAT WE'RE READY
C       TO USE THE FILE.
        CALL OPENM$(K$READ,FNAME,6,UNIT,CODE)
        IF (CODE .NE. 0) GO TO 9000
        .
        .
        .
C       DO MIDAS FILE PROCESSING (EG. CALLS TO FIND$)
        .
        .
        .
        CALL CLOSM$(UNIT,CODE) /* TELL MIDAS WE'RE DONE
C                                  AND CLOSE THE FILE
        .
        .
        .
```

2.3.4 ADMINISTRATION CHANGES

2.3.4.1 OVERVIEW

USERS MUST PERFORM TWO TYPES OF MIDAS INITIALIZATION PROCEDURES. WHEN DOING A COLD START, THE SEGMENT CONTAINING THE LOCK MUST BE SHARED, THE LOCK VALUE MUST BE SET TO ZERO AND THE SEMAPHORE DRAINED. INITIALIZATION OF THE SEMAPHORE AND SHARED LOCK IS HANDLED BY MIDAS UTILITY IMIDAS. FOR DETAILS REFER TO SECTION 2.3.4.2.

THE SECOND TYPE OF INITIALIZATION IS NECESSARY IF AN APPLICATION PROGRAM ABNORMALLY TERMINATES AND AS A CONSEQUENCE FAILS TO RELEASE THE SHARED LOCK. IF THE LOCK IS NOT RELEASED, ALL MIDAS PROCESSES WILL BE BLOCKED. TO RELEASE THE LOCK, MCLUP SHOULD BE EXECUTED. NOTE THAT A BLOCKED CONDITION MIGHT NOT BE IMMEDIATELY RECOGNIZED BY USERS. IF THIS CONDITION IS SUSPECTED, MCLUP MAY BE EXECUTED SIMPLY TO DETERMINE WHICH PROCESS HOLDS THE LOCK. MCLUP IS DESCRIBED IN MORE DETAIL IN SECTION 2.3.4.3.

2.3.4.2 MIDAS INITIALIZATION -- IMIDAS

```
**********
*        *
* IMIDAS *
*        *
**********
```

FUNCTION

INITIALIZES THE MIDAS SEMAPHORE AND SHARED LOCK.

DISCUSSION

1. IMIDAS MUST BE RUN AS PART OF THE COLD START SEQUENCE. IF MIDAS APPLICATION PROGRAMS ARE RUNNING WHEN IMIDAS IS INVOKED, MIDAS FILES IN USE AT THE TIME MIGHT BE DAMAGED. COMMAND FILE C_MINIT MAY BE INSTALLED IN THE COLD START PROCEDURE TO SHARE THE SEGMENT CONTAINING THE LOCK AND TO EXECUTE IMIDAS.

2. IMIDAS HAS BEEN CODED AS A SUBROUTINE NAMED "MAIN" SO THAT IS CAN BE LOADED INTO SPLIT SEGMENT 4000. IMIDAS MAY THEN BE EXECUTED USING THE RESUME COMMAND.

3. COMMAND FILE C_IMIDAS IN UFD MIDAS>SOURCE MAY BE USED TO BUILD IMIDAS IN UFD MIDAS>CMDNC0.

4. IMIDAS MUST BE COMPILED WITH THE "-64V" AND "-BIG" FTN OPTIONS. DURING THE LOAD, THE COMMON BLOCK WITH THE NAME "LIST" MUST BE PLACED AT THE ADDRESS <0/1> WITH THE SEG COMMAND:

   ```
   SY LIST 0 1
   ```

2.3.4.3 MIDAS CLEANUP UTILITY -- MCLUP

```
**********
*        *
* MCLUP  *
*        *
**********
```

FUNCTION

AFTER ABNORMAL TERMINATION OF A MIDAS PROGRAM, MCLUP RE-INITIALIZES THE SHARED LOCK AND NOTIFIES THE SEMAPHORE TO AWAKEN ANY MIDAS PROCESS WAITING ON THE LOCK.

DISCUSSION

1. MCLUP IS NEEDED ONLY WHEN THE ABNORMAL TERMINATION OCCURS WITHIN THE MIDAS CODE. THIS SITUATION CAN ARISE IF THE USER TYPES 'BREAK' OR 'CONTROL-P', OR IF AN INTERNAL MIDAS BUG CAUSES AN ERROR SUCH AS AN ACCESS VIOLATION.

2. IF INVOKED WITH NO OPTIONS, MCLUP RE-INITIALIZES ONLY IF THE SHARED LOCK IS HELD BY THE TERMINAL USER, OTHERWISE MCLUP PRINTS THE USER NUMBER OF THE USER THAT HOLD THE LOCK. IF NO PROCESS HOLDS THE LOCK, THEN MCLUP DOES NOTHING.

3. IF INVOKED WITH AN OPTION OF THE FORM:

   -USER USERNUMBER

   THEN MCLUP WILL RE-INITIALIZE IF THE SHARED LOCK IS HELD BY THE SPECIFIED USER, OTHERWISE MCLUP PRINTS THE USER NUMBER OF THE USER THAT HOLDS THE LOCK. IF THE USER NUMBER OF AN ACTIVE MIDAS PROCESS IS SPECIFIED, DAMAGE MAY OCCUR TO MIDAS FILES IN USE BY THE PROCESS.

4. MCLUP MAY BE BUILT IN UFD CMDNC0 BY COMMAND FILE C_MCLUP IN UFD MIDAS.

5. MCLUP MUST BE COMPILED WITH THE "-64V" AND "-BIG" FTN OPTIONS. DURING THE LOAD, THE COMMON BLOCK WITH THE NAME "LIST" MUST BE PLACED AT THE ADDRESS <0/1> WITH THE SEG COMMAND

   SY LIST 0 1

3 RECOVERY FROM CONCURRENCY ERRORS

3.1 OVERVIEW

MIDAS NOW DETECTS AND CORRECTS MOST CONCURRENCY ERRORS. THESE ERRORS, ASSOCIATED WITH OPERATIONS INVOLVING THE CURRENT RECORD, OCCUR WHEN THE CURRENT INDEX ENTRY HAS BEEN DELETED OR PHYSICALLY MOVED SINCE THE TIME THE ENTRY BECAME CURRENT. IF MIDAS DISCOVERS THAT THE ENTRY HAS BEEN DELETED, THEN AN ERROR CODE OF 13 IS RETURNED. IN THE EVENT THAT THE ENTRY HAS BEEN MOVED, MIDAS AUTOMATICALLY LOCATES THE ENTRY AND CONTINUES NORMALLY.

3.2 IMPLEMENTATION OF CONCURRENCY ERROR DETECTION AND RECOVERY

AT THE FORTRAN CALL LEVEL INTERFACE, THE CONCEPT OF CURRENT RECORD AND CURRENT ENTRY IS IMPLEMENTED AS A FOURTEEN WORD COMMUNICATION ARRAY. THE COMMUNICATION ARRAY IS AN ARGUMENT IN MOST SUBROUTINE CALLS TO MIDAS. THE NEXT SECTION OUTLINES THE NEW COMMUNICATION ARRAY FORMAT.

3.2.1 COMMUNICATION ARRAY FORMAT

WORD 1 (INPUT) IF -1 THEN MIDAS ARRAY CONTENTS ARE NOT USED.
(OUTPUT) ERROR STATUS

WORDS 2-4 CURRENT INDEX ENTRY ADDRESS
- WORD 2 BITS 1-8 -- ENTRY NUMBER
- WORD 2 BITS 9-16 -- SEGMENT FILE NUMBER
- WORDS 3 & 4 (32 BITS) -- WORD OFFSET OF INDEX BLOCK

WORD 5 HASH VALUE (BASED ON CURRENT KEY VALUE)

WORDS 6-9 CURRENT KEY VALUE (OR 1ST 4 WORDS OF KEY)

WORDS 10-12 CURRENT RECORD ADDRESS
- WORD 10 BIT 1 -- RECORD LOCKED FLAG
- WORD 10 BITS 9-16 -- SEGMENT FILE NUMBER
- WORDS 11 & 12 -- WORD OFFSET OF RECORD

WORD 13 DATA CONTROL WORD
- BITS 1-8 -- FLAG BITS
- BITS 9-16 -- PRIMARY KEY SIZE (BITS)

WORD 14 DATA RECORD LENGTH (WORDS)

NOTE THAT WORDS 2 THROUGH 9 OF THE COMMUNICATION ARRAY SPECIFY A CURRENT INDEX ENTRY AND WORDS 10 THROUGH 12 SPECIFY A CURRENT RECORD.

DURING OPERATIONS INVOLVING THE CURRENT ENTRY (EG. GET NEXT RECORD) WORDS 2 THROUGH 4 ARE USED TO LOCATE THE EXPECTED POSITION OF THE

ENTRY. TO VERIFY THAT THE POSITION CONTAINS THE CORRECT ENTRY, MIDAS COMPARES THE DATA POINTER IN THE ENTRY WITH THE DATA POINTER IN WORDS 10 THROUGH 12 OF THE COMMUNICATION ARRAY. IF THE POINTERS DON'T MATCH, THE THE ENTRY IS THE WRONG ONE.

EVEN IF THE POINTERS DO MATCH, MIDAS COMPARES THE KEY VALUE IN THE INDEX ENTRY TO THE KEY VALUE IN THE COMMUNICATION ARRAY. IF THEY DON'T MATCH, THEN THE ENTRY IS THE WRONG ONE. WHEN A WRONG ENTRY IS DETECTED, MIDAS SEARCHES FOR THE CORRECT ENTRY. IF NOT FOUND, MIDAS RETURNS AN ERROR CODE OF 13. NOTE THAT REV 16 VERSIONS EARLIER THAN REV 16.5 RETURNED AN ERROR CODE OF 13 WHEN A CONCURRENCY ERROR WAS DETECTED. USERS OF THESE EARLIER RELEASES MAY HAVE MODIFIED THEIR APPLICATIONS TO ATTEMPT TO RECOVER FROM AN ERROR 13. AN ERROR 13 INDICATES THAT THE CURRENT INDEX ENTRY HAS BEEN DELETED, EXISTING APPLICATION ATTEMPTS TO HANDLE AN ERROR 13 MAY NEED MODIFICATION.

3.3 LIMITATIONS

FOR INDEXES WITH KEYS WHICH ARE LONGER THAN 8 BYTES, MIDAS MAY FAIL TO DETECT A CONCURRENCY ERROR. TO UNDERSTAND HOW THIS MAY OCCUR, NOTICE THAT IN THE COMMUNICATION ARRAY, AT MOST EIGHT BYTES OF A KEY MAY BE STORED. FOR KEYS LONGER THAN EIGHT BYTES, MIDAS STORES A HASH VALUE IN WORD 5 OF THE ARRAY. THE HASH VALUE IS BASED ON THE PORTION OF THE KEY BEYOND THE EIGHTH BYTE. NOW MIDAS WILL FAIL TO DETECT A CONCURRENCY ERROR IF:

A) THE DATA POINTERS MATCH (IE. THE 2 INDEX ENTRIES POINT TO THE SAME DATA RECORD),
B) THE KEY IS LONGER THAN 8 BYTES,
C) THE FIRST 8 BYTES OF THE KEY MATCH THE 8 BYTES STORED IN THE COMMUNICATION ARRAY, AND
D) THE HASH CODE, BASED ON THE REMAINING BYTES, IS THE SAME AS THE HASH CODE IN THE ARRAY.

OR IF:

A) THE DATA POINTERS MATCH,
B) THE KEYS ARE LESS THAN OR EQUAL TO 8 BYTES, AND
C) THE KEYS MATCH.

4 INSTALLATION OF MIDAS

4.1 COMMAND FILES

SEVERAL NEW COMMAND FILES HAVE BEEN ADDED.

C_MIDAS -- BUILDS MIDAS LIBRARIES AND UTILITIES.

C_VKDALB -- BUILDS THE SHARED V MODE LIBRARY, VKDALB.
VKDALB IS PUT IN LIB. K4000, K2014A,
AND K2014B ARE PLACED IN UFD SYSTEM.

C_NVKDALB -- BUILDS THE UNSHARED V MODE LIBRARY NVKDALB
IN UFD LIB.

C_KIDALB -- BUILDS THE R MODE LIBRARY IN UFD LIB.

C_IMIDAS -- BUILDS UTILTY IMIDAS IN UFD SYSTEM.

C_MCLUP -- BUILDS UTILTY MCLUP IN UFD CMDNC0.

C_CREATK -- BUILDS CREATK IN CMDNC0.

C_KBUILD -- BUILDS KBUILD IN CMDNC0.

C_KIDDEL -- BUILDS KIDDEL IN CMDNC0.

4.2 MODIFYING THE SHARED LOCK AND SEMAPHORE VALUES

AS SUPPLIED, MIDAS USES SEMAPHORE NUMBER 64 AND WORD :177777 OF SEGMENT 2020 AS THE SHARED LOCK. THESE VALUES, DEFINED IN FILE KPARAM, MAY BE MODIFED BY USERS.

THE PARAMETERS ARE:

MSEMA1 -- SEMAPHORE NUMBER
SLSEG -- SEGMENT NUMEER OF THE SHARED LOCK
SLWORD -- WORD NUMBER OF THE SHARED LOCK

IF ANY OF THESE VALUES IS MODIFIED, THE USER MUST FOLLOW THE PROCEDURE DESCRIBED IN PARTS 2 AND 3 OF SECTION 4.3. MIDAS UTILITIES MCLUP AND IMIDAS MUST BE REBUILT AND INSTALLED. IN ADDITION, COMMAND FILE C_MINIT AND THE COLD START PROCEDURE MUST BE MODIFIED SO THAT THE CORRECT SEGMENT GETS SHARED.

4.3 DISABLING THE NEW CONCURRENT PROCESS HANDLING METHOD

USERS MAY DISABLE THE CONCURRENCY CONTROL METHOD AND THEREBY RETURN TO THE METHOD USED IN PREVIOUS RELEASES. NOTE THAT PROGRAMS WHICH USE NTFYM$, OPENM$, AND CLOSM$ WILL STILL WORK CORRECTLY.

PROCEDURE:

1) IN FILE KPARAM, CHANGE THE VALUE OF PARAMETER SHDSEG FROM .TRUE. TO .FALSE.,

2) FOR THE UNSHARED MIDAS LIBRARIES, KIDALB AND NVKDALB,

   A) COMPLILE SUBROUTINE LDPOOL. FOR V MODE LIBRARY NVKDALB USE FILE LONGPL. FOR THE R MODE LIBRARY KIDALB USE FILE LDPOOL.

   B) USE THE BINARY EDITOR, EDB, TO REPLACE THE OLD VERSION OF ROUTINE LDPOOL WITH THE NEW VERSION.

   C) RELOAD APPLICATION PROGRAMS WHICH USE THE UNSHARED LIBRARIES.

3) FOR THE SHARED V MODE LIBRARY VKDALB, REBUILD AND RE-INSTALL THE LIBRARY. APPLICATION PROGRAMS WHICH USE THE SHARED LIBRARY DO NOT NEED TO BE RE-LOADED.

4.4 NETWORK USERS

FOR NETWORK APPLICATIONS IN WHICH PROCESSES ACCESS REMOTE MIDAS FILES, THE CONCURRENT PROCESS HANDLING METHOD MUST BE DISABLED BY THE USER TO PREVENT LOSS OF FILE INTEGRITY.

4.5 MIDAS FILE READ/WRITE LOCKS

WHEN MIDAS IS INSTALLED, THE READ/WRITE LOCK FOR EACH MIDAS FILE WHICH IS TO BE ACCESSED CONCURRENTLY, MUST BE SET BY THE USER TO 3. (N READERS AND M WRITERS)

4.6 RELOADING APPLICATION PROGRAMS

WHEN INSTALLING MIDAS, ALL APPLICATION PROGRAMS WHICH USE AN UNSHARED MIDAS LIBRARY MUST BE RELOADED.

UUP110-UP112 (MDLC1) RELEASE OF BASIC DIAGNOSTIC FOR THE 5600 (MDLC) SERIES SYNCHRONOUS CONTROLLERS

**

UUP113-UP115 (MDLC2) RELEASE OF BISYNC MICROCODE DIAGNOSTIC FOR THE 5600 (MDLC) SERIES OF SYNCHRONOUS CONTROLLERS

**

UUP116-UP118 (MDLC3) RELEASE OF PACKET MICROCODE DIAGNOSTIC FOR THE 5600 (MDLC) SERIES OF SYNCHRONOUS CONTROLLERS

**

UUP119-UP121 (MDLC4) RELEASE OF DIAGNOSTIC FOR THE ICL7020-UT200 UNIVAC 1004 MICROCODE FOR THE 5600 (MDLC) SERIES OF SYNCHRONOUS CONTROLLERS

**

UUP122-UP124 (MDLC5) RELEASE OF DIAGNOSTIC FOR HDLC MICROCODE FOR THE 5600 (MDLC) SERIES OF SYNCHRONOUS CONTROLLERS

**

UUP125-UP127 (MDLC6) RELEASE OF DIAGNOSTIC FOR BISYNC + ANY OTHER PROTOCOL ON THE 5600 (MDLC) SERIES OF SYNCHRONOUS CONTROLLERS

**

UUP128-UP130 (MDLC7) RELEASE OF DIAGNOSTIC FOR PACKET + ANY OTHER PROTOCOL ON THE 5600 (MDLC) SERIES OF SYNCHRONOUS CONTROLLERS

**

UUP131-UP133 (MDLC8) RELEASE OF DIAGNOSTIC FOR THE HDLC + ANY OTHER PROTOCOL ON THE 5600 (MDLC) SERIES OF SYNCHRONOUS CONTROLLERS

**

UUP134-UP136 (MDLC8) RELEASE

**

UUP137-UP138 (URCT1) TO ADD TEST FOR NEW ELECTRONIC VERTICAL FORMAT UNIT OPTION ON 1000 LPM DATA PRINTER LINE PRINTER

**

UUP139-UP140 (RTCT2) TO FIX PIO TIMING CHARACTERISTICS PERTINENT TO VCP OPERATION

**

UUP141-UP142 (HSSCT2) FAILED OCCASIONALLY ON PRIME 200'S

**

UUP143-UP146 (COBOL) SEE MIDAS 16.5. COBOL HAS BEEN CHANGED TO WORK CORRECTLY WITH MIDAS 16.5.

**

UUP146A-UP146B (PXT1) TO ALLOW THE VCP TO OPERATE WITH THE DIAGNOSTIC AS THE TEST USED TO USE THE SOC'S DIAGNOSTIC MODE CAPABILITY WHICH ARE NOT PRESENT ON THE VCP.

**

UUP146C-UP146D (AMLCT5) TWO SMALL CHANGES WERE MADE. ONE WAS A BUG FIX AND THE OTHER IS AN ADDED FEATURE.

**

UUP146E-UP146F (FLT750) NEW TEST PROGRAM FOR P750 FLOATING POINT HARDWARE

**

UUP146G-UP146H (P500T2) TO ACCOMMODATE THE P750 CPU.

UUP146I-UP146J (CPUT4) TO ACCOMMODATE CHANGES MADE TO THE 750.

**
UUP146K-UP146L (XACHE1) TO ACCOMODATE THE P750 CPU.