129

June 1997

MVS update

In this issue

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38030
From USA: 01144 1635 38030
E-mail: stevep_xephon@compuserve.com

North American office

Xephon
1301 West Highway 407, Suite 201-450
Lewisville, TX 75067, USA
Telephone: 940 455 7050

Australian office

Xephon/RSM
PO Box 6258, Halifax Street
Adelaide, SA 5000
Australia
Telephone: 08 223 1391

Contributions

If you have anything original to say about MVS, or any interesting experience to recount, why not spend an hour or two putting it on paper? The article need not be very long – two or three paragraphs could be sufficient. Not only will you be actively helping the free exchange of information, which benefits all MVS users, but you will also gain professional recognition for your expertise, and the expertise of your colleagues, as well as some material reward in the form of a publication fee – we pay at the rate of £170 ($250) per 1000 words for all original material published in *MVS Update*. If you would like to know a bit more before starting on an article, write to us at one of the above addresses, and we'll send you full details, without any obligation on your part.

Editor

Steve Piggott

Disclaimer

Readers are cautioned that, although the information in this journal is presented in good faith, neither Xephon nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither Xephon nor the contributing organizations or individuals accept any liability of any kind howsoever arising out of the use of such material. Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, EXECs, and other contents of this journal before making any use of it.

Subscriptions and back-issues

A year's subscription to *MVS Update*, comprising twelve monthly issues, costs £310.00 in the UK; $465.00 in the USA and Canada; £316.00 in Europe; £322.00 in Australasia and Japan; and £320.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1992 issue, are available separately to subscribers for £27.00 ($39.00) each including postage.

MVS Update on-line

Code from *MVS Update* can be downloaded from our Web site at http://www.xephon.com; you will need the user-id shown on your address label.

Year 2000 countdown – sending an alert

INTRODUCTION

This article describes an Assembler-written MPF user exit which I have called IEAVMY2K. Its aim is to raise the awareness among TSO users of the proximity to the year 2000 via a 'countdown' message, which is issued to any user logging on to TSO.

When the message IEF125I is sent to SYSLOG, MPF passes control to this routine which generates a SEND command conveying this message to the prospective user:

```
YEAR 2000 COUNTDOWN n YEARS nnn DAYS LEFT
```

During 1999, this becomes:

```
YEAR 2000 COUNTDOWN         nnn DAYS LEFT
```

until 31 December 1999, when it becomes:

```
YEAR 2000 COUNTDOWN   L.T.  nn HOURS
```

and after 23.59.59 on 31 December 1999, it becomes:

```
YEAR 2000 WELCOME TO THE NEW MILLENNIUM!
```

I have tested this routine under MVS/ESA 4.3 and TSO 2.4, and I would expect it to run with little or no change under other versions. Link-edit the program in a LINKLIST dataset and SET MPF to a member which associates message IEF125I with the exit as follows:

```
IEF125I,USEREXIT(IEAVMY2K)
```

IEAVMY2K SOURCE CODE

```
*********************************************************************
*  DESCRIPTIVE NAME      =  COMMUNICATIONS TASK USER EXIT FOR        *
*                            MSG IEF125I                            *
*  FUNCTIONS             =  ISSUES COMMAND "SEND" TO LOGGED ON USERID *
*                           WITH YEAR 2000 COUNTDOWN ALERT          *
*  ENTRY POINT           =  IEAVMY2K                                *
*  LINKAGE               =  BALR                                    *
*  INPUT DATA            =  REG 1 POINTS TO THE ADDRESS OF THE CTXT  *
*                           REG 13 ADDRESS OF STANDARD SAVE AREA    *
```

```
*                          R15 ENTRY POINT                              *
*  REGISTERS SAVED       =  RØ - R15                                     *
*  REGISTER USAGE        =  R2  - POINTER TO THE ADDRESS OF CTXT         *
*                           R3  - WORK REGISTER                          *
*                           R4  - WORK REGISTER                          *
*                           R5  - WORK REGISTER                          *
*                           R7  - BAL TO TIME ROUTINE                    *
*                           R11 - DYNAMIC GETMAINED STORAGE              *
*                           R12 - BASE REGISTER                          *
*                           R13 - STANDARD SAVE AREA CHAINING            *
*                           R14 - RETURN POINT                           *
*  REGISTERS RESTORED    =  RØ - R15                                     *
*  CONTROL BLOCKS        =                                               *
*  NAME     MAPPING MACRO    REASON USED                                 *
*  ----     -------------    -----------                                 *
*  CTXT     IEZVX1ØØ         USER EXIT PARAMETER LIST                    *
*  MGCR     IEZMGCR          SVC 34 PARAMETER LIST                       *
*  TABLES                =   NONE                                        *
*  MACROS                =   GETMAIN, FREEMAIN,  MGCR                    *
*********************************************************************
IEAVMY2K CSECT
IEAVMY2K AMODE 31                       31-BIT ADDRESSING MODE
IEAVMY2K RMODE ANY                      31-BIT RESIDENCE
SPINPRVT EQU   23Ø
*---STANDARD ENTRY LINKAGE-----------------------------------------
         STM   R14,R12,12(R13)          SAVE CALLER'S REGISTERS
         LR    R12,R15                  LOAD BASE
         USING IEAVMY2K,R12             USING THE BASE REGISTER
*---GET ADDRESSABILITY TO MESSAGE----------------------------------
         L     R2,Ø(R1)                 ESTABLISH ADDRESSABILITY
         USING CTXT,R2                  TO THE CTXT
*---GET A SAVEAREA-------------------------------------------------
GETSAVE  EQU   *
         GETMAIN RU,LV=DATAEND,SP=SPINPRVT,LOC=BELOW   OBTAIN          X
                                                 DYNAMIC STORAGE
         LR    R11,R1                   R1 --> ACQUIRED STORAGE
         USING DATAAREA,R11             ADDRESSABILITY TO DYNAMIC      X
                                        STORAGE
         ST    R13,SAVEAREA+4           STORE LSA ADDR IN NEW SA
         LA    R15,SAVEAREA             GET ADDRESS OF NEW SA
         ST    R15,8(R13)               STORE NEW SA ADDR IN LSA
         LR    R13,R15                  R13 --> NEW SA
*---PROCESS OUR MESSAGE-------------------------------------------
         L     R2,CTXTTXPJ              ESTABLISH ADDRESSABILITY
         DROP  R2                          TO THE
         USING CTXTATTR,R2                    INTERCEPTED MESSAGE
MSG125I  EQU   *
         LA    R2,CTXTTMSG              ADDRESS OF TEXT AREA
         DROP  R2                       DROP REGISTER
         USING IEF125ID,R2              IEF125I DSECT
```

```
         XC    MGCRPL(MGCRLTH),MGCRPL   CLEAR THE PARAMETER LIST
         MVC   MGCRTEXT(LSTRTMSG),STARTMSG  MOVE DEFAULT MESSAGE
         BAL   R7,COUNTDWN              HOW MUCH TIME IS LEFT ?
         LA    R3,M125UID               ADDR OF USERID (VAR. LENGTH)
         LA    R15,8                    MAX USERID LENGTH
*---MOVE VARIABLE LENGTH USERID-------------------------------------
UIDLOOP  EQU   *
         LA    R3,1(,R3)                INCREMENT TO NEXT BYTE
         CLI   Ø(R3),X'4Ø'              END OF USERID FIELD ?
         BE    UIDLPEND                 YES.
         BCT   R15,UIDLOOP              KEEP LOOPING.
         B     TERMINAT                 NEVER HAPPENS !
UIDLPEND EQU   *
         MVC   Ø(1,R3),=CL1')'          MOVE CLOSING BRACKET
         LA    R5,M125UID               ADDR OF USERID (VAR. LENGTH)
         SR    R3,R5                    USERID LENGTH
         LR    R4,R3                    SAVE USERID LENGTH
         MVC   UIDCMD(Ø),M125UID        EXECUTED MOVE
         EX    R3,*-6                   MOVE THE USERID
         AR    R2,R4                    ADD USERID LENGTH TO BASE
         LA    R1,(MGCRTEXT-MGCRPL)+LSENDCMD GET MGCRPL LENGTH
         AR    R1,R4                    ADD VARIABLE USERID LENGTH
         STC   R1,MGCRLGTH              SAVE LENGTH IN THE MGCRPL
         SR    RØ,RØ                    CLEAR REGISTER ZERO
         MGCR  MGCRPL                   ISSUE THE COMMAND
         B     TERMINAT                 ALL DONE
         DROP  R2                       DROP REGISTER
*---HOW MUCH TIME IS LEFT ?-----------------------------------------
COUNTDWN EQU   *
         TIME  DEC,#YYYYDDD,DATETYPE=YYYYDDD,LINKAGE=SYSTEM
*----->>>FIELD #YYYYDDD IS 16 BYTES WITH DATE @ +8 (ØY.YY.YD.DD)
         MVC   FULLW(4),#YYYYDDD+8
         L     1,FULLW                  R1 =    ØY.YY.YD.DD
         SLL   1,4                      R1 =    YY.YY.DD.DØ
         ST    1,FULLW                  FULLW = YY.YY.DD.DØ
         OI    FULLW+3,X'ØF'            FULLW = YY.YY.DD.DF (VALID SIGN)
         ZAP   PKWORK(2),=P'365'        ADD # OF DAYS IN A YEAR (NO LEAPS)
         SP    PKWORK(2),FULLW+2(2)     # OF DAYS LEFT IN THIS YEAR
         MVC   TEXTCMD+29(4),MASKDAYS   MAKE IT SUITABLE
         ED    TEXTCMD+29(4),PKWORK     FOR HUMANS
         TM    FULLW+1,B'11111111'      ALL ZEROES ?
         BZ    Y2K                      YES             = Y2K
         TM    FULLW+1,B'ØØØØ0111'      WHICH YEAR ?
         BZ    YEAR98                   1ØØØ ALL ZEROES = 98
         BM    YEAR99A                  1ØØ1 MIXED      = 99
*                                       Ø111 ALL ONES   = 97
         MVC   TEXTCMD+22(1),=C'2'      2 YEARS LEFT
         B     CDEND
YEAR98   MVC   TEXTCMD+22(1),=C'1'      1 YEAR  LEFT
         MVC   TEXTCMD+28(1),=C' '      SINGULAR
```

```
         B     CDEND
Y2K      MVC   TEXTCMD+10(36),=C': WELCOME TO THE NEW MILLENNIUM!    '
         B     CDEND
YEAR99A  EQU   *
         CLC   TEXTCMD+31(2),=C' 0'  31DEC1999 ? ONLY HOURS LEFT
         BNE   YEAR99B               NOT THE -LAST- DAY
         MVC   FULLW(4),#YYYYDDD+0   FULLW = HH.MM.SS.HS
         L     1,FULLW               R1 =    HH.MM.SS.HS
         SRL   1,20                  R1 =    00.00.0H.HM
         ST    1,FULLW               FULLW = 00.00.0H.HM
         OI    FULLW+3,X'0F'         FULLW = 00.00.0H.HF (VALID SIGN)
         ZAP   PKWORK(2),=P'24'      ADD # OF HOURS IN A DAY
         SP    PKWORK(2),FULLW+2(2)  # OF HOURS LEFT IN 31DEC1999
         MVC   TEXTCMD+22(16),=C'  L.T.     HOURS'
         MVC   TEXTCMD+28(4),MASKDAYS   MAKE IT SUITABLE
         ED    TEXTCMD+28(4),PKWORK         FOR HUMANS
         B     CDEND
YEAR99B  MVC   TEXTCMD+22(7),=CL7' '    1999: SHOW ONLY # DAYS
CDEND    BR    R7                      BACK TO CALLER
*---TERMINATION ROUTINE-----------------------------------------------
TERMINAT EQU   *
         L     R13,4(R13)              RESTORE REG 13
         FREEMAIN RU,LV=DATAEND,A=(R11),SP=SPINPRVT     FREE STORAGE
QUIT     LM    R14,R12,12(R13)         RESTORE CALLER'S REGISTERS
         BR    R14                     RETURN TO CALLER
*---SAMPLE COMMAND----------------------------------------------------
STARTMSG DC    CL5'SEND '
         DC    CL48''YEAR 2000 COUNTDOWN: 0 YEARS DDD DAYS LEFT ! '','
         DC    CL12'LOGON,USER=('
         DC    CL8' '                  VARIABLE LENGTH
         DC    CL1' '                  ')'VARIABLE OFFSET
         DC    CL1' '
LSTRTMSG EQU   *-STARTMSG
MASKDAYS DC    XL4'40202120'
PKWORK   DC    PL2'0'
*---STORAGE AND SAVE AREAS--------------------------------------------
DATAAREA DSECT
         DS    0F
SAVEAREA DS    18F                     STANDARD SAVE AREA
         DS    0F
MGCR     IEZMGCR DSECT=NO
         ORG   MGCRTEXT
SENDCMD  DS    CL4                     VALUE 'SEND'
         DS    CL1                     VALUE ' '
TEXTCMD  DS    CL48                    VALUE VARIABLE
         DS    CL12                    VALUE 'LOGON,USER=('
UIDCMD   DS    CL8                     VALUE USERID 1 TO 8 BYTES
         DS    CL1                     VALUE ')'
LSENDCMD EQU   *-SENDCMD
         ORG
```

```
#YYYYDDD DS    CL16                    OUTPUT AREA FOR MACRO TIME
FULLW    DS    F
DATAEND  EQU   *-DATAAREA
*---DSECT FOR MESSAGE IEF125I-----------------------------------------
IEF125ID DSECT
M125ID   DS    CL7                     MESSAGE ID (IEF125I)
         DS    CL1
M125UID  DS    CL8                     USERID
         DS    CL64
         IEZVX1ØØ
         END   IEAVMY2K
```

Rui-Jorge Pessoa
Information Specialist
Saudi Arabian Airlines (Saudi Arabia)

Using the SRST instruction

The SRST instruction is a recent addition to the ES/390 instruction set, introduced by APAR PN39397. The following code shows how it can be used to scan storage for a particular hexadecimal value. It's quite neat and beats using TRT hands down.

```
SET_SEARCH_VARS EQU   *
                LA    1,WORK_PARM         POINT TO WORK VAR
                L     3,=A(L'WORK_PARM)   LOAD WORK VAR LENGTH
                AR    3,1                 POINT TO END OR WORK VAR
                LA    3,1(,3)             MAKE UP FOR OFFSET Ø
                LA    Ø,C'*'              SET UP SEARCH CHAR
SCAN_PARM       EQU   *
                SRST  3,1                 LET'S DO SOME STRING SCANNING
                BC    1,SCAN_PARM         CPU INTERRUPTED INSTRUCTION
                BC    2,MATCH_FOUND       FOUND SEARCH CHAR
SCAN_PARM_END   EQU   *
NO_MATCH        EQU   *
                LA    5,8                 SET BAD RC
                B     ENDIT
MATCH_FOUND     EQU   *
                SR    5,5                 SET GOOD RC
*       AT THIS POINT:
*               GPRØ CONTAINS VALUE OF SEARCH CHAR
*               GPR1 POINTS TO ADDR OF SEARCH CHAR WITHIN WORK VAR
*               GPR3 POINTS TO ADDR OF END OF WORK VAR
```

```
*               GPR5 IS USED TO HOLD A USER RETURN CODE
ENDIT           EQU   *
WORK_PARM       DS    CL100              DECLARE STORAGE FOR VAR
```

Calum Reid
Systems Technician
Standard Life (UK)

Introducing Cartridge Unit Groups (CUGs) to MVS

THE PROBLEM

Let us consider the configuration shown in Figure 1 comprising two CPUs, each connected to two robotic Comparex 6388 Automated Cartridge Library (ACL) systems. Each robot serves 24 cartridge tape subsystems (3490s). In addition, there are four 3490 units and eight 3480s, which are outside the ACL range and are operated manually. We will refer to the cartridges belonging to one physical location (whether robot or manually-operated) as Cartridge Unit Groups (CUGs).

When a hardware error occurs on one of these devices, MVS/ESA will try to move the volume to another unit. This process is called swapping or system-initiated Dynamic Device Reconnect (DDR). MVS/ESA has no notion of CUGs – all similar devices are potential swap targets.

MVS will issue message IGF500I to indicate that it will try to swap the volume, and to propose a target unit address. The IGF500D message which follows immediately allows the operator to change the target unit.

During nightly unattended batch processing, we encountered the situation where MVS/ESA wanted to swap a cartridge unit to one that was not connected to the same robot, and so caused the job streams to halt until an operator responded to message IGF500D with an address in the range of the robot.

The problem remains the same wherever cartridges are used in different physical locations (eg two adjacent buildings).

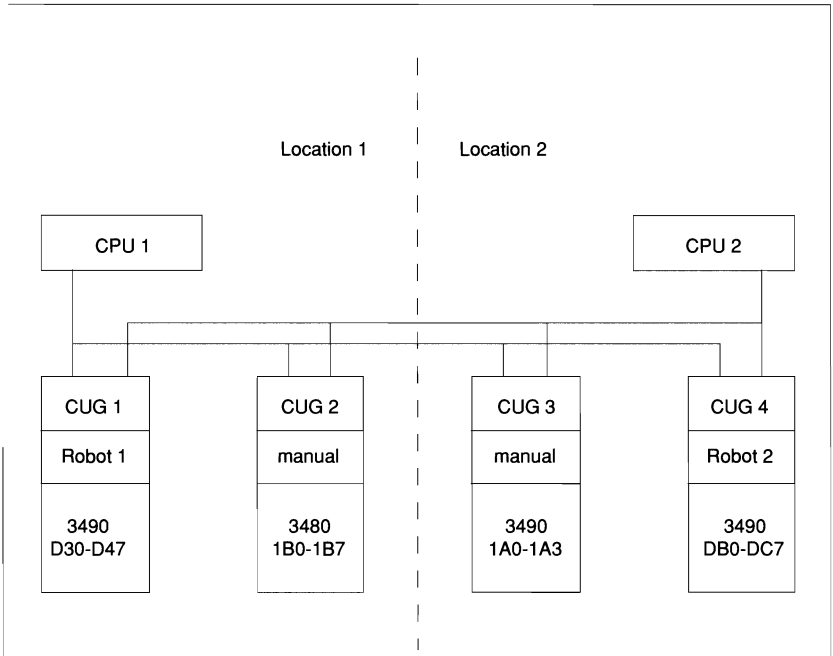

Figure 1: CUG configuration

THE SOLUTION

The following Message Processing Facility (MPF) exit solves the problem by responding to IGF500D with a unit address in the same CUG, so that processing can continue without manual intervention.

PREREQUISITES

MVS/ESA Version 4 is the base level to use the UCBLOOK macro. On pre-MVS/SP Version 4 systems, the more limited IOSLOOK macro can be coded. The availability of the OSLEVEL extension (needed for UCBLOOK logic) can be checked in the CVTDB byte. If the high-order bit (X'80' or CVTOSEXT) is set to on, UCBLOOK can be used.

INSTALLATION

The exit must be link-edited as reentrant with APF code (1) into a LINKLIST library. Add the following line to the MPFLST*xx* member of SYS1.PARMLIB to define the exit to MPF:

```
IGF500D,SUP(NO),USEREXIT(IGF500D) /* Exit has same name as message */
```

Finally, activate the code by issuing the system command SET MPF=*xx*.

TESTING

In order to activate a new version (say one link-edited in load library EXITLOAD), it is sufficient to issue a 'F LLA,DSN=EXITLOAD' command followed by a SET MPF command. We found the manual to be very unclear about the IEE677I message that the system responds with to a display MPF command.

After 'D MPF,CMD', IEE677I stated 'COMMAND USER EXITS NOT FOUND' although the exit was in the MPFLIST. A 'D MPF' command invokes a display of all messages eligible for MPF processing and their corresponding settings. The module name was listed after the IGD500D message without any ERR or NOTFOUND prefix, meaning that the exit was active.

The system can be forced to swap a volume by the swap command (G). We allocated a cartridge unit (1D0) with a huge copy job, displayed the unit status with the 'D U,,,1D0,24' command, and forced the swap to a free device (1D9) by issuing a 'G 1D0,1D9' command.

CUG DEFINITION

A table (contained in a copy book) is assembled in the exit. To keep the number of CUGs and the number of units in a CUG flexible, the program makes no assumption whatsoever but relies entirely on the information in this table.

Example table

```
* TABLES: POINTERS TO TABLES WITH ADDRESS RANGES FOR ROBOTS
*         EVERY ROBOT HAS AN ADDRESS RANGE IN A DIFFERENT TABLE
```

```
*         END OF TABLE IS INDICATED BY F'-1'
* WHEN ROBOTS ARE ADDED, INSERT A TABLE WITH THE RANGE OF
* CONNECTED UNITS (EG SIDE_A), ADD A POINTER TO THIS RANGE
* IN THE TABLE SIDE_A
A_TABLES DS    ØD
         DC    A(SIDE_A)               -> RANGE OF ADDRESSES FOR SIDE A
         DC    A(SIDE_B)               -> RANGE OF ADDRESSES FOR SIDE B
         DC    A(SIDE_X)               -> RANGE OF ADDRESSES FOR SIDE X
         DC    A(SIDE_Y)               -> RANGE OF ADDRESSES FOR SIDE Y
         DC    F'-1'
*
* SIDE A: ROBOT WITH CARTRIDGES STARTING FROM ØD3Ø - ØD47
*         END OF TABLE IS INDICATED BY F'-1'
*
SIDE_A   DS    ØF
         DC    CL4'ØD3Ø'
L_TABENT EQU   *-SIDE_A
         DC    CL4'ØD31'
         DC    CL4'ØD32'
         DC    CL4'ØD33'
         DC    CL4'ØD34'
         DC    CL4'ØD35'
         DC    CL4'ØD36'
         DC    CL4'ØD37'
         DC    CL4'ØD38'
         DC    CL4'ØD39'
         DC    CL4'ØD3A'
         DC    CL4'ØD3B'
         DC    CL4'ØD3C'
         DC    CL4'ØD3D'
         DC    CL4'ØD3E'
         DC    CL4'ØD3F'
         DC    CL4'ØD4Ø'
         DC    CL4'ØD41'
         DC    CL4'ØD42'
         DC    CL4'ØD43'
         DC    CL4'ØD44'
         DC    CL4'ØD45'
         DC    CL4'ØD46'
         DC    CL4'ØD47'
         DC    F'-1'                   END OF TABLE FOR SIDE_A
         EJECT
*
* SIDE_B: ROBOT WITH CARTRIDGES STARTING FROM ØDBØ - ØDC7
*         END OF TABLE IS INDICATED BY F'-1'
*
SIDE_B   DS    ØF
         DC    CL4'ØDBØ'
         DC    CL4'ØDB1'
         DC    CL4'ØDB2'
```

```
         DC    CL4'ØDB3'
         DC    CL4'ØDB4'
         DC    CL4'ØDB5'
         DC    CL4'ØDB6'
         DC    CL4'ØDB7'
         DC    CL4'ØDB8'
         DC    CL4'ØDB9'
         DC    CL4'ØDBA'
         DC    CL4'ØDBB'
         DC    CL4'ØDBC'
         DC    CL4'ØDBD'
         DC    CL4'ØDBE'
         DC    CL4'ØDBF'
         DC    CL4'ØDCØ'
         DC    CL4'ØDC1'
         DC    CL4'ØDC2'
         DC    CL4'ØDC3'
         DC    CL4'ØDC4'
         DC    CL4'ØDC5'
         DC    CL4'ØDC6'
         DC    CL4'ØDC7'
         DC    F'-1'                   END OF TABLE FOR SIDE_B
         EJECT
*
* SIDE_X: OUTSIDE ROBOT 349Ø CARTRIDGES
*         END OF TABLE IS INDICATED BY F'-1'
*
SIDE_X   DS    ØF
         DC    CL4'Ø1AØ'
         DC    CL4'Ø1A1'
         DC    CL4'Ø1A8'
         DC    CL4'Ø1A9'
         DC    F'-1'                   END OF TABLE FOR SIDE_X
         EJECT
*
* SIDE_Y: OUTSIDE ROBOT 348Ø CARTRIDGES
*         END OF TABLE IS INDICATED BY F'-1'
*
SIDE_Y   DS    ØF
         DC    CL4'Ø1B2'
         DC    CL4'Ø1B3'
         DC    CL4'Ø1BA'
         DC    CL4'Ø1BB'
         DC    CL4'Ø1FØ'
         DC    CL4'Ø1F1'
         DC    CL4'Ø1F8'
         DC    CL4'Ø1F9'
         DC    F'-1'                   END OF TABLE FOR SIDE_Y
```

PROGRAMMING NOTES

Some of the coding is redundant: the exit is only called by MVS/ESA if the message is written, so there is no real need to check the message-id. The checking of all control block eyecatchers could also be deleted. We incorporated this because of house standards (see *Assembler programming standards*, *MVS Update*, Issue 112, January 1996, pp 45-55) and because performance is not an issue in this case. If this coding is entered more then once a month, you really have a problem.

WTOs scheduled by the program are formatted as follows:

CUGMS	Prefix.
xx	Number.
a	I(nformative) – commented out in the production version. A(ction) – requires attention.

CODING

```
 TITLE 'IGF5ØØD  -          COM TASK INSTALLATION EXIT FOR MESSAGE     X
               IGF5ØØD'
* MODULE NAME: IGF5ØØD
* DESCRIPTIVE NAME: MPF EXIT FOR MESSAGE IGF5ØØD
* REQUIREMENTS: MVS/ESA V4 IS THE BASE LEVEL FOR THE USE OF THE
*                UCBLOOK MACRO (FOR PRE-MVS/ESA V4 SYSTEMS, USE THE
*                MORE LIMITED IOSLOOK MACRO.
* FUNCTION: THE OPERATING SYSTEM ISSUES MESSAGE
*           IGF5ØØT FOR VERIFICATION OF A
*           SWAP COMMAND ENTERED BY THE OPERATOR OR A
*           REQUEST TO MOVE VOLUME AS A RESULT OF A
*           PERMANENT I/O ERROR ON A DEVICE.
* OPERATION:  MESSAGE IGF5ØØT THAT CONTAINS THE DEVICE
*             NUMBER OF THE DEVICE CAUSING THE ERROR IS
*             FOLLOWED BY MESSAGE IGF5ØØT WHERE A REPLY
*             CAN INCLUDE THE ADDRESS OF THE DEVICE
*             THAT SHOULD BE USED.
*             THIS EXIT REPLIES WITH A DEVICE UNIT ADDRESS
*             FROM A TABLE OF LIKE-DEVICES (E.G. CONNECTED
*             TO THE SAME ROBOT).
* ENTRY POINT: IGF5ØØD
* PURPOSE: IF A SWAP IS REQUESTED BY THE ROBOT,
*          ENSURE THAT THE ANSWERED DEVICE ADDRESS
*          CORRESPONDS TO A DEVICE NAME CONNECTED
*          TO THE ROBOT AND VICE VERSA.
* LINKAGE: BALR
```

```
* INPUT DATA: R1: POINTS TO THE ADDRESS OF THE CTXT
*             RD: ADDRESS OF A STANDARD SAVE AREA
*             RF: ENTRY POINT
* REGISTERS SAVED:    RØ - RF
* REGISTERS RESTORED: RØ - RF
* CONTROL BLOCKS:
* NAME     MAPPING MACRO   REASON USED                  USAGE
* ----     -------------   -----------                  -----
* CTXT     IEZVX1ØØ        USER EXIT PARAMETER LIST     R,W
* CVT      CVT             FIND DDRCOM POINTERS         R
* DDRCOM   IHADDR          DDR COMMUNICATION TABLE      R
* PSA      IHAPSA          FIND CVT                     R
* UCB      IEFUCBOB        FIND UNALLOCATED DEVICE      R
* MODULE TYPE: CSECT
* ATTRIBUTES: REENTRANT
*             REUSABLE
*             AMODE 31
*             RMODE ANY
* LINK: LINKLIST CONCATENATION
* ACTIVATION: PARMLIB MEMBER MPFLSTXX SHOULD CONTAIN:
*             'IGF5ØØD,SUP(NO),USEREXIT(IGF5ØØD)'
*             ACTIVATE BY COMMAND: 'SET MPF=XX'
*             AFTER VERSION UPDATE REFRESH THE LLA
* LOGIC: WHEN AN I/O ERROR OCCURS THE SYSTEM TRIES TO SWAP
*        TO ANOTHER DEVICE AND MESSAGE IGF5ØØI IS ISSUED,
*        FOLLOWED BY MESSAGE IGF5ØØT
*        WHEN MESSAGE IGF5ØØT IS ISSUED THE DDRCOM TABLES ARE SCANNED
*        FOR A DDRCOM AREA WITH OUTSTANDING IGF5ØØT MESSAGE
*        ONCE LOCATED, THE UCBADDRESS IS COMPARED TO THE UCB IN THE
*        PRE-DEFINED HARDCODED TABLE. IF THE UCBS ARE EQUAL,
*        THE EXIT REPLIES WITH AN ADDRESS FROM THE SAME TABLE IN THE
*        FOLLOWING ORDER:
*        - THE UCBS FROM THE CORRESPONDING TABLE ARE SCANNED TILL
*          ONE IS FOUND THAT IS ONLINE AND UNALLOCATED
*        - IF ALL UCBS IN THIS TABLE ARE ALLOCATED, THE FIRST
*          ONLINE DEVICE IN THE TABLE IS USED (OB WILL HAVE TO WAIT)
*        - IF ALL DEVICES IN THIS TABLE ARE OFFLINE, THE EXIT
*          ISSUES A MESSAGE AND LEAVES
* MESSAGES:  WTO STRUCTURE:
* CUGMS    : PREFIX
*      XX  : NUMBER
*        Y : I FOR INFORMATION
*            A FOR ACTION
*            I-TYPE MESSAGES ARE SUPPRESSED (OUTCOMMENTED)
*            A-TYPE MESSAGES ARE NOT SUPPRESSED AND SHOULD
*            BE PASSED TO THE SYSTEMS GROUP
IGF5ØØD  CSECT
IGF5ØØD  AMODE 31                    31-BIT ADDRESSING MODE
IGF5ØØD  RMODE ANY                   31-BIT RESIDENCY MODE
*********************************************************************
```

```
*        STANDARD ENTRY LINKAGE                                         *
************************************************************************
         STM   RE,RC,12(RD)            SAVE CALLER'S REGISTERS
         LR    RC,RF                   RC -> MODULE BASE
         USING IGF500D,RC              RC IS BASE ADDRESS
         L     R5,0(R1)                R5 -> CTXT
         USING CTXT,R5                 ADDRESS CTXT WITH R5
* IF NOT MESSAGE IGF500D RETURN TO OS
*        WTO   'CUGMS00I MPF EXIT ENTERED'
         L     R2,CTXTTXPJ             R2 -> MESSAGE ATTRIBUTES AREA
         USING CTXTATTR,R2             ESTABLISH ADDRESSABILITY
         LA    R4,CTXTTMSG             R4 -> TEXT AREA
         USING MSGTXT,R4               ADDRESS MESSAGE TEXT WITH R4
         CLC   MSGID,=CL8'IGF500D'     MESSAGE IGF500D?
         DROP  R2                      DROP TEXT AREA ADDRESSABILITY
         DROP  R4                      DROP MESSAGE TEXT ADDRESSABILITY
         BNE   THE_END                 NO -> EXIT IMMEDIATELY
* LOCATE AND CHECK DDRCOM CHAINED FROM CVTTRPOS
*        WTO   'CUGMS01I MPF EXIT ENTERED FOR MESSAGE IGF500D'
         XR    R3,R3                   R3 = 0
         USING PSA,R3                  R3 ADDRESSES THE PSA
         L     R3,FLCCVT               R3 -> CVT
         DROP  R3                      DROP PSA ADDRESSABILITY
         USING CVT,R3                  R3 ADDRESSES THE CVT
         L     R4,CVTTPUR              R4 -> DDRCOM AREAS PHASE 2
         LTR   R4,R4                   TABLE EXISTS?
         BZ    L0030                   NO --> GO CHECK FOR PHASE 1
         USING DDRCOM,R4               R4 ADDRESSES THE DDRCOM
L0000    DS    0H
*        WTO   'CUGMS02I MPF EXIT DDR COM PHASE 2'
         CLC   DDRID,=C'DDR '          CHECK EYECATCHER
         BE    L0010                   OK --> CONTINUE
         L     R6,CTXTFCNP             R6 = A(4 BYTE CONSOLE ID)
         L     R6,0(R6)                R6 = 4 BYTE CONSOLE ID
         WTO   'CUGMS03A MPF EXIT: DDR CONSISTENCY ERROR 1',           X
               CONSID=(R6)
         B     THE_END                 RETURN TO CALLER
L0010    DS    0H
         TM    DDRMSGCD,DDRM500D       IGF500D MESSAGE ISSUED?
         BNO   L0020                   NO --> GO CHECK NEXT ONE
         BAL   RA,L0070                YES -> GO LOOK UP IN TABLES
L0020    DS    0H
         L     R4,DDRNXT               NO --> R4 -> NEXT DDR COM
         LTR   R4,R4                   IS THERE A NEXT DDR COM?
         BNZ   L0000                   YES -> GO CHECK HIM
L0030    DS    0H
         L     R4,CVTTRPOS             R4 -> DDRCOM AREAS PHASE 1
         DROP  R3                      DROP CVT ADDRESSABILITY
L0040    DS    0H
         LTR   R4,R4                   TABLE EXISTS?
```

```
         BZ    THE_END                 NO --> NOTHING MORE TO DO
*        WTO   'CUGMSØ4I MPF EXIT DDR COM PHASE 1'
         CLC   DDRID,=C'DDR '          CHECK EYECATCHER
         BE    LØØ5Ø                   OK --> CONTINUE
         L     R6,CTXTFCNP             R6 = A(4 BYTE CONSOLE ID)
         L     R6,Ø(R6)                R6 = 4 BYTE CONSOLE ID
         WTO   'CUGMSØ5A MPF EXIT: DDR CONSISTENCY ERROR 2',          X
               CONSID=(R6)
         B     THE_END                 RETURN TO CALLER
LØØ5Ø    DS    ØH
         TM    DDRMSGCD,DDRM5ØØD       IGF5ØØD MESSAGE ISSUED?
         BNO   LØØ6Ø                   NO --> GO CHECK NEXT ONE
         BAL   RA,LØØ7Ø                YES -> GO LOOK UP IN TABLES
LØØ6Ø    DS    ØH
         L     R4,DDRNXT               NO --> R4 -> NEXT DDR COM
         B     LØØ4Ø                   GO CHECK NEXT DDR COM?
* ENTERED BY BAL RA,LØØ7Ø IF DDRCOM IS FOUND WHICH ISSUED
* THE IGF5ØØD MESSAGE
* RESERVED:
* R4 -> DDRCOM AREA
* R5 -> CTXT AREA
* RA -> RETURN ADDRESS IN DDRCOM SCAN LOOP
* RC -> BASE
LØØ7Ø    DS    ØH                      DDR COM AREA IS LOCATED
*        WTO   'CUGMSØ6I MPF EXIT DDR COM LOCATED'
         LA    R7,A_TABLES             R7 -> TABLE POINTERS
LØØ8Ø    DS    ØH
         CLC   =F'-1',Ø(R7)            END OF TABLE POINTERS REACHED?
         BER   RA                      YES -> RETURN TO DDRCOM SCAN
         L     R6,Ø(R7)                NO --> R6 -> TABLE
LØØ9Ø    DS    ØH
*                                      BELONGS TO TABLE?
         CLC   DDRFMNAM+1(DIGIT3),1(R6)
         BE    LØ1ØØ                   YES -> GO ISSUE REPLY
         LA    R6,L_TABENT(R6)         R6 -> NEXT TABLE ENTRY
         CLC   =F'-1',Ø(R6)            R6 -> END OF TABLE?
         BNE   LØØ9Ø                   NO --> GO CHECK NEXT ONE
         LA    R7,4(R7)                R7 -> NEXT TABLE ADDRESS
         B     LØØ8Ø                   GO CHECK NEXT TABLE
         DROP  R4                      DROP DDRCOM ADDRESSABILITY
* A DDRCOM AREA IS FOUND THAT MATCHES A TABLE ENTRY.
* GET SOME STORAGE TO WORK WITH
LØ1ØØ    DS    ØH
*        WTO   'CUGMSØ7I MPF EXIT MATCHING DDR COM - TABLE ENTRY'
         L     RØ,F_L_DYN              LENGTH OF DATA AREAS
         GETMAIN  RU,                  OBTAIN DYNAMIC STORAGE         X
               LV=(RØ),                #(BYTES) IS IN RØ              X
               SP=SPINPRVT             SUBPOOL NUMBER
         LR    RB,R1                   RB -> GETMAINED STORAGE
         USING DATAAREA,RB             RB ADDRESSES STORAGE
```

```
* ALGORITHM FOR FINDING THE REPLACE ADDRESS, KEEP THE POINTER TO
* THE UNIT ADDRESS IN CHARACTER FORMAT IN R6
* 1. TRY TO FIND A DEVICE THAT IS ON-LINE AND UNALLOCATED
* WE WILL NOT USE RA TO RETURN TO THE DDRCOM CB SCAN,
* RESERVED:
* R5 -> CTXT AREA
* R6 -> OLD (SWAPPED) UNIT ADDRESS
* RB -> DYNAMICALLY GETMAINED AREA
* RC -> BASE
* AT START:
* R7 -> A(TABLE FOR THIS ROBOT)
         L     R7,Ø(R7)                 R7 -> TABLE FOR THIS ROBOT
         LR    R4,R7                    R4 -> TABLE FOR THIS ROBOT
LØ11Ø    DS    ØH
         CR    R6,R7                    THIS THE SWAPPED ONE?
         BE    LØ14Ø                    YES -> SKIP PROCESSING
*                                       COPY UCBLOOK MACRO TO DYNAMIC
         MVC   UCBLK_D(UCBLKLEN),UCBLK_S
         UCBLOOK DEVNCHAR=(R7),         LOOK UP UCB POINTED TO BY R7    X
               UCBPTR=UCBPTR,           RETURN UCB ADDRESS IN UCBPTR    X
               NOPIN,                   DON'T PIN THE UCB               X
               DYNAMIC=YES,             INCLUDE DYNAMIC UCBS            X
               RANGE=3DIGIT,            3 BYTE DEVICE NUMBER (IGF5ØØD)  X
               RSNCODE=(R3),            PLACE REASON CODE IN R3         X
               MF=(E,UCBLK_DL)          EXECUTE FORMAT WITH DYN LIST
         LTR   RF,RF                    UCBLOOK OK?
         BZ    LØ12Ø                    YES -> CONTINUE
         L     R6,CTXTFCNP              R6 = A(4 BYTE CONSOLE ID)
         L     R6,Ø(R6)                 R6 = 4 BYTE CONSOLE ID
         WTO   'CUGMSØ8A MPF EXIT: UCBLOOK ERROR',                      X
               CONSID=(R6)
         B     LØ21Ø                    GO RETURN TO CALLER
LØ12Ø    DS    ØH
         L     RA,UCBPTR                RA -> UCB
         USING UCBCMSEG,RA              RA ADDRESSES UCB COMMON
         TM    UCBSTAT,UCBONLI          UCB ON-LINE?
         BNO   LØ14Ø                    NO --> CHECK NEXT ONE
LØ13Ø    DS    ØH
         TM    UCBSTAT,UCBALOC          UCB ALLOCATED?
         BNO   LØ18Ø                    NO --> USE FOR REPLY
         DROP  RA                       DROP UCB ADDRESSABILITY
LØ14Ø    DS    ØH
*        WTO   'CUGMSØ9I MPF EXIT UCB ALLOCATION REFUSED'
         LA    R7,L_TABENT(R7)          R7 -> NEXT TABLE ENTRY
         CLC   R7,=F'-1'                LAST ONE?
         BNE   LØ11Ø                    NO --> GO CHECK ALLOCATION
* ALGORITHM FOR FINDING THE REPLACE ADDRESS
* 2. TRY TO FIND A DEVICE THAT IS ON-LINE
* ALL UCBS IN THIS TABLE ARE ALLOCATED. SELECT THE FIRST
* ONLINE UNIT, THE JOB WILL HAVE TO WAIT.
```

```
* RESERVED:
* R4 -> START OF TABLE
* R5 -> CTXT AREA
* R6 -> OLD (SWAPPED) UNIT ADDRESS
* RB -> DYNAMICALLY GETMAINED AREA
* RC -> BASE
         LR    R7,R4                    YES -> R7 -> TABLE
LØ15Ø    DS    ØH
         CR    R6,R7                    THIS THE SWAPPED ONE?
         BE    LØ17Ø                    YES -> SKIP PROCESSING
*                                       COPY UCBLOOK MACRO TO DYNAMIC
         MVC   UCBLK_D(UCBLKLEN),UCBLK_S
         UCBLOOK DEVNCHAR=(R7),         LOOK UP UCB POINTED TO BY R7    X
               UCBPTR=UCBPTR,           RETURN UCB ADDRESS IN UCBPTR    X
               NOPIN,                   DON'T PIN THE UCB               X
               DYNAMIC=YES,             INCLUDE DYNAMIC UCBS            X
               RANGE=3DIGIT,            3-BYTE DEVICE NUMBER (IGF5ØØD)  X
               RSNCODE=(R3),            PLACE REASON CODE IN R3         X
               MF=(E,UCBLK_DL)          EXECUTE FORMAT WITH DYN LIST
         LTR   RF,RF                    UCBLOOK OK?
         BZ    LØ16Ø                    YES -> CONTINUE
         L     R6,CTXTFCNP              R6 = A(4 BYTE CONSOLE ID)
         L     R6,Ø(R6)                 R6 = 4 BYTE CONSOLE ID
         WTO   'CUGMS1ØA MPF EXIT: UCBLOOK ERROR',                      X
               CONSID=(R6)
         B     LØ21Ø                    GO RETURN TO CALLER
LØ16Ø    DS    ØH
         L     RA,UCBPTR                RA -> UCB
         USING UCBCMSEG,RA              RA ADDRESSES UCB COMMON
         TM    UCBSTAT,UCBONLI          UCB ON-LINE?
         BO    LØ18Ø                    NO --> CHECK NEXT ONE
         DROP  RA                       DROP UCB ADDRESSABILITY
LØ17Ø    DS    ØH
*        WTO   'CUGMS11I MPF EXIT UCB OFF-LINE REFUSED'
         LA    R7,L_TABENT(R7)          R7 -> NEXT TABLE ENTRY
         CLC   R7,=F'-1'                LAST ONE?
         BNE   LØ15Ø                    NO --> GO CHECK ALLOCATION
* NO ON-LINE DEVICE FOUND IN THIS TABLE, ISSUE WTO
         L     R6,CTXTFCNP              R6 = A(4 BYTE CONSOLE ID)
         L     R6,Ø(R6)                 R6 = 4 BYTE CONSOLE ID
         WTO   'CUGMS12A MPF EXIT: NO ON-LINE DEVICE TO SWAP TO',       X
               CONSID=(R6)
         B     LØ21Ø                    GO RETURN TO CALLER
* SUPPRESS THE IGF5ØØT MESSAGE AND ISSUE A REPLY
* RESERVED:
* R5 -> CTXT AREA
* R7 -> REPLY UNIT ADDRESS IN CHARACTER FORMAT
* RB -> DYNAMICALLY GETMAINED AREA
* RC -> BASE
LØ18Ø    DS    ØH
```

```
*        WTO   'CUGMS13I MPF EXIT UCB SELECTED'
         LR    R6,R7                    R6 -> REPLY ADDRESS
         OI    CTXTRFB2,CTXTRHCO        SUPPRESS MESSAGE IGF500T
* REPLY TO THE IGF500T MESSAGE
* - CONSTRUCT REPLY: R7 -> DYNAMICALLY BUILD REPLY ZONE
         LA    R7,DYNCMDR1              ADDRESS COMMAND REPLY AREA
* - CONSTRUCT REPLY: BLANK OUT DYNAMICALLY BUILD REPLY ZONE
         MVI   0(R7),BLANK              BLANK FIRST MESSAGE CHARACTER
         MVC   1(CMDRLENG-1,R7),0(R7)   BLANK MESSAGE FIELD
* - CONSTRUCT REPLY: INSERT COMMAND LENGTH
         MVC   0(L'H_L_CMD,R7),H_L_CMD  INITIALIZE COMMAND LENGTH
* - CONSTRUCT REPLY: INSERT 'REPLY '
         MVC   L'H_L_CMD(L'TXINS1,R7),TXINS1
* - CONSTRUCT REPLY: INSERT REPLY NUMBER
*                    CTXTRPYI CONTAINS THE EBCDIC REPLY ID VALUE.
*                    CTXTRPYL CONTAINS THE LENGTH OF THE REPLY ID.
         LH    R8,CTXTRPYL              R8 = L(REPLY ID)
         BCTR  R8,0                     --R8 FOR EXECUTE
         EX    R8,MOVERYID              MOVE REPLY ID VALUE INTO TEXT
* - CONSTRUCT REPLY: INSERT ','
         LA    R8,TXI1LENG+1(R8,R7)     POINT AFTER ID
         MVI   0(R8),C','               ADD ','
* - CONSTRUCT REPLY: INSERT NEW UNIT ADDRESS
         CLI   0(R6),C'0'               3 BYTE ADDRESS?
         BE    L0190                    YES -> REPLY 3 BYTE ADDRESS
         MVC   1(4,R8),0(R6)            UNIT ADDRESS TO REPLY
         B     L0200                    CONTINUE REPLY
L0190    DS    0H
         MVC   1(3,R8),1(R6)            UNIT ADDRESS TO REPLY
* - CONSTRUCT REPLY: INSERT STATIC MGCRE AREA INTO DYNAMIC AREA
L0200    DS    0H
         LA    R4,DYNMGCRE              ADDRESS MGCRE PARAMETER LIST
         MVC   0(REPLEN,R4),REPAREA     COPY MGCRE LIST TO DYNAMIC
* - REPLY
*        WTO   'CUGMS14I MPF EXIT REPLY READY'
         L     R6,CTXTFCNP              R6 = A(4 BYTE CONSOLE ID)
         L     R6,0(R6)                 R6 = 4 BYTE CONSOLE ID
         MGCRE TEXT=(R7),               R7 -> DYNAMIC REPLY AREA       X
               CONSID=(R6),             R6 = 0                         X
               MF=(E,(R4))              MACRO FORMAT = EXECUTE
         LTR   RF,RF                    SUCCESSFUL?
         BZ    L0210                    YES -> CONTINUE
         L     R6,CTXTFCNP              R6 = A(4 BYTE CONSOLE ID)
         L     R6,0(R6)                 R6 = 4 BYTE CONSOLE ID
         WTO   'CUGMS15A MPF EXIT: MGCRE ERROR',                       X
               CONSID=(R6)
         B     L0210                    GO RETURN TO CALLER
* FREEMAIN THE STORAGE
L0210    DS    0H
*        WTO   'CUGMS16I MPF EXIT --- EXIT'
```

```
         L     RØ,F_L_DYN              RØ = L(DYNAMIC DATA AREA)
         FREEMAIN  RU,                 RELEASE STORAGE                 X
               LV=(RØ),                RØ CONTAINS THE LENGTH          X
               A=(RB),                 RB CONTAINS THE ADDRESS         X
               SP=SPINPRVT             SUBPOOL NUMBER
* STANDARD EXIT LINKAGE, AND EXIT FROM THIS MODULE
THE_END  DS    ØH                      MY ONLY FRIEND, THE END
         LM    RE,RC,12(RD)            RESTORE CALLER'S REGISTERS
         BR    RE                      RETURN TO CALLER
* CHARACTER CONSTANTS
MIGF5ØØD DC    CL8'IGF5ØØD'            MESSAGE IGF5ØØD
F_L_DYN  DS    ØF                      LENGTH OF DYNAMIC AREA
         DC    AL4(DATAEND)            MUST REMAIN ON F-BOUNDARY
* LENGTH AND TEXT OF REPLY TO WTOR (MAXIMUM REPLY ID VALUE IS 9999)
CMDREPLY DS    ØF
H_L_CMD  DC    AL2(E_L_CMD)            LENGTH OF WTOR REPLY COMMAND
TXINS1   DC    CL6'REPLY '             FIRST STATIC INSERT
TXI1LENG EQU   *-CMDREPLY              OFFSET TO TXINS1
TXINS2   DC    CL4'XXXX'               PLACE REPLY ID HERE
TXINS3   DC    C','                    STATIC INSERT
TXINS4   DC    CL4'YYYY'               PLACE NEW UNIT ADDRESS HERE
E_L_CMD  EQU   *-TXINS1                COMMAND TEXT LENGTH
CMDRLENG EQU   *-CMDREPLY              COMMAND REPLY LENGTH
* LIST FORM OF MGCRE MACRO (STATIC)
USERREP  DS    ØF
REPAREA  MGCRE MF=L                    LIST FORM OF MACRO
REPLEN   EQU   *-USERREP               LENGTH OF MGCRE PARAMETER LIST
* LIST FORM OF UCBLOOK MACRO (STATIC)
+CBLK1   DS    ØF
UCBLK_S  UCBLOOK MF=(L,UCBLK_SL)       LIST FORM OF MACRO (STATIC)
UCBLKLEN EQU   *-UCBLK1                LENGTH OF UCBLOOK PARAM. LIST
* EXECUTE TARGETS
MOVERYID MVC   TXINS2-CMDREPLY(Ø,R7),CTXTRPYI  MOVE REPLY ID
* LITERAL POOL
         LTORG
* COPY BOOK: TABLES
         COPY TABLES
* EQUATES
BLANK    EQU   X'4Ø'                   BLANK CHARACTER
DIGIT3   EQU   3                       CHECK FOR 3 DIGITS
SPINPRVT EQU   23Ø                     SUBPOOL VALUE FOR GETMAIN
* DUMMY SECTIONS: 1. DYNAMIC AREA USED BY THIS PROGRAM
DATAAREA DSECT
UCBPTR   DS    F                       UCBLOOK WILL PUT A(UCB) HERE
DYNMGCRE DS    CL(REPLEN)              DYNAMIC MGCRE AREA
DYNCMDR1 DS    CL(CMDRLENG)            DYNAMIC COMMAND REPLY AREA
         DS    ØH
* LIST FORM OF UCBLOOK MACRO (DYNAMIC)
         DS    ØF
UCBLK_D  UCBLOOK MF=(L,UCBLK_DL)       LIST FORM OF MACRO (DYNAMIC)
```

```
DATAEND  EQU   *-DATAAREA
* DUMMY SECTIONS: 2. MESSAGE AREA
MSGTXT   DSECT
MESSAGE  DS    ØCL43
MSGID    DS    CL8                    MESSAGE ID
JOBDATA  DS    CL8                    JOB DATA (ID AND NAME)
* DUMMY SECTIONS: 3. MVS/ESA PROVIDED DSECTS
         CVT   DSECT=YES              COMMUNICATION VECTOR TABLE
         IEZVX1ØØ                     CTXT DSECT
         IHADDR                       DDR COM DSECT
         IHAPSA                       PREFIXED STORAGE AREA
         IEFUCBOB                     UNIT CONTROL BLOCK
         END   IGF5ØØD
```

Jan de Decker
JED:SP NV (Belgium)

Writing your own Sysplex service provider

INTRODUCTION

Many larger sites already run parallel Sysplexes and this has opened up a whole new world of both interesting problems to overcome as well as opportunities for the MVS systems programmer. The initial stages are often associated with all sorts of teething problem, particularly with regard to data sharing and Sysplex-wide deadlocks caused by incorrect RNL parameters or unforeseen GRS situations. Over time these problems are overcome and the vast benefits of running with Sysplexes become more evident. Preparing for and installing your first Sysplex is source for another interesting article (and also many hours of both scheduled and unscheduled after-hours work!) but this article is about making use of some of the many features that become available with Sysplex and XCF services.

PROBLEMS

Over time, more and more products will become Sysplex-compliant. In the meantime most third-party products are not and these are some of the problems that can be expected:

- Certain products just cannot be run on each member of a Sysplex at the same time. In a Sysplex, the intention is to share the master catalog, spool, and possibly even the RESVOL. The restriction with sharing the master catalog is that only one copy of any dataset can exist – that in itself can cause a problem sometimes. For example, unless your product that extracts print-outs from the spool file for network printing purposes is specifically written to cope with a shared spool environment, it may also cause unexpected results. Say it is running on each Sysplex member and it extracts job output and sends it to a network printer or to a job output repository on another system. In an ideal shared spool/Sysplex environment, a copy of this started task is to run on each system, that is per strict Sysplex definition. The problem is that a job's output will enter the spool and each copy of your product may extract a copy and send it off to the same printer – really bad if you care about trees.

- Most products and facilities are currently not geared to stay in synchronization across Sysplex members. One of them is LLA. Many sites have written their own version of a program to do LLA refreshes from a batch environment making use of the LLACOPY macro and these are often in use by application programmers etc. To keep LLA in synchronization across all members, this macro would have to be issued on all systems at the same time. If it is not done, different members of the Sysplex will have different copies of the same load module in LLA and the same job (or on-line system) running on different systems will produce different results.

- Few products are currently Automatic Restart Manager (ARM)-compliant. If they were they would be able to register with ARM and, upon failure, would automatically be started up either on the same system or elsewhere, as per ARM parameters. For non-compliant products we may have to find a way of notifying the other systems in the Sysplex in cases where we have a total system outage. Let's say we have a product and set it up to split the work by system. The copy running on system A services all 'type 1' requests, on system B all 'type 2' requests etc. Should we lose system A, we have to either start another copy of the product on system B or notify the product already running on system B to now do both 'type 1' and 'type 2' work. To start up a new copy of the

product to run on system B, we would have to somehow notify the scheduling package; if we go for the second option, we would have to notify the product itself. (This would depend on the product and could be via a MODIFY command or whatever the product may require.)

Looking at the above (there are many more examples), it would seem ideal if we could have similar copies of our own started task running on each system within the Sysplex and in full communication with each other. In this way we can detect changes in the environment and make sure that all systems are informed of what is going on. Going back to our spool-to-printer example, we could run one copy only and if we lose the member (system) it is running on for any reason, another member will know about it and it could start the software on its own system.

SOLVING THE PROBLEM WITH XCF

XCF group services are ideally designed for this purpose. Amongst other things, XCF provides the following:

- The ability for each member of the group to know which other member(s) are out there. These could be members running in the same step under another TCB, in another job on the same system, or in another job on another system. These are all 'linked' together via group services and each one is notified when a new member either joins or leaves the group.
- The ability to inquire about the status of any of the other members.
- The ability to communicate with any of the other members.

The XCF group services macros are easy to code and the only small complication is that XCF schedules all the user-written exits (group, message, and status) as SRBs. This puts some restrictions on the code that can be developed (eg no SVCs) but the idea is to let the exit routine do as little as possible. In most cases it should just POST the main task to notify it of the event and return control to MVS.

CODING EXAMPLE

Here is an example of an IXCJOIN macro coded to join a group:

```
LA     R2,MsgExit@          .Address of message exit routine
LA     R3,AnsArea           .Address of answer area
USING QUAMEM,AnsArea        .Addressability to answer area
ICM    R4,15,AnsAreaL       .Length of answer area
L      R5,GrpExit@          .Address of group exit routine
L      R6,Status@           .Address of status field in DREF
L      R7,StsExit@          .Address of status exit routine
LA     R8,UserData          .Address of user state data field
LA     R9,MemData           .Address of member data field
IXCJOIN GRPNAME=MyGroup,MSGEXIT=(2),                          x
     ANSAREA=(3),ANSLEN=(4),LASTING=YES,MEMNAME=MemName,      x
     GRPEXIT=(5),STATFLD=(6),STATEXIT=(7),INTERVAL=Interval,  x
     USTATE=(8),USLEN=32,MEMDATA=(9)
```

where:

- *MyGroup* is the name (8 bytes) of the group. If the group does not already exist, it is automatically created when the first member JOINS it.
- *MsgExit@* is the address of an SRB routine scheduled into this task by MVS if a message arrives for it. This exit routine has to issue an IXGMSGI macro to receive the message and should ideally move the data into a storage area pre-allocated by the mainline code – refer to the *MemData* field.
- *AnsArea* is a work area MVS uses to return group information, eg member names, status, etc.
- *MemName* is the name (16 bytes) of the member.
- *GrpExit@* is the address of an SRB routine scheduled into this task by MVS each time there is a change in the group status, eg a new member joins or a member leaves.
- *Status@* is the address of an 8-byte status field required in DREF storage.
- *StsExit@* is the address of an SRB routine scheduled in this task every *Interval* seconds if there has been no change in the status field.
- *UserData* is a storage area that contains the information XCF is to place in the member's user state field.
- *MemData* is a 64-bit storage area that gets passed to all the exit routines when scheduled. This can for instance be used to contain

the address of a storage area to move the message into as well as the address of an ECB to be POSTed by the message exit routine before returning control to MVS.

Once the new member has joined the group, it could have the mainline code or a subtask waiting on an ECB or an ECB list. At any given time another member could (IXC)LEAVE or (IXC)JOIN the group in which case it could be POSTed. (That is, assuming we need to know which other members are active at any given time. If we need to know this and do not use this convention of a group exit routine, the mainline code has to do an (IXC)QUERY to find out which others members are active from time to time.) The same principle applies for the status or message exits routines – it is best to wait on an ECB list and take the appropriate action once the event occurs.

We now have the basics for our service provider. It should start up, set up the ECB list, and wait for an external event to wake it up. It should also have an input message format convention, something like '*progname,parameter1,parameter2,...*'. That is, when it receives a message it will do some basic format checking and then call program *progname* and pass it the parameters *parameter1*, *parameter2*, etc. Any 'user' that wishes to have a service provided should build a message in this format, send it to our service provider (which could be running on another system), and optionally wait for a return message. The handshaking between user and provider can easily be agreed upon and set up in a commonly-assembled DSECT which is used to address the message traffic.

So let's get back to the LLACOPY example and see how we can make it work.

On each system we have a started task being started at IPL time (or maybe just another subtask attached from an existing 'systems programmer's task' that is already in use at your site). This task/subtask (IXC)JOINs a group and waits on a single ECB to be POSTed by a message exit routine. (All XCF macros require the invoker to be in supervisor state.) So if we have four members in the Sysplex we will have a group by the name of say 'LLAGROUP' with members 'LLAGROUPSYSTEM*xyz*', with *xyz* the name of each system. We now modify our (existing) LLACOPY program and add an additional parameter, say PARM='*mode*,...', and *mode* can be something like

'Sysplex-wide' or 'local'. This routine (say //S1 EXEC PGM=OURLLACP,PARM='*mode*,SYS2.LINKLIB') is executed and does the following:

- If its mode parameter specifies 'local', it issues the LLACOPY macro for the dataset and terminates.
- If its mode input parameter specifies 'sysplex-wide', it does the following:
  - (IXC)JOINs the LLAGROUP with member name LLAGROUPUSER (propagated with blanks).
  - Issues an IXCQUERY macro to find the names of all the other members and puts these into a table
  - Builds a parameter list of the format 'OURLLACP,LOCAL,SYS2.LINKLIB' and issues an IXCMSGO to send this parameter as a message to each of the service providers (either one at a time or at the same time using subtasking). A simple way to serialize events throughout a Sysplex is to do a system-wide exclusive ENQ. This can be used if the service provider can only do one activity at a time. Any new request for a similar service will thus be queued until this one has completed.

These identical service providers running on each of the systems will then each receive a message and individually call program OURLLACP with parameters 'local' and 'SYS2.LINKLIB'. In this way the LLACOPY is issued at each system. OURLLACP should also have the ability to receive back a message from the service provider indicating, for example, the return code received from the LLACOPY macro.

The service provider's parameter fields could easily be expanded to contain an option for type of action. In the LLA example, we simply need it to be able to do a call to a program, but we could let it issue commands (although these could be done from any member in the Sysplex with an RO command) or provide other services like putting messages into a queue. If, for instance, we have a started task that can only run on one of the Sysplex members, we can 'collect' all requests for work for it on all other members, and send them as messages to our service provider, which can then put it in a queue for the started task.

Just a brief warning on the message exit routine: it will normally do an IXCMSGI macro to actually obtain the message (IXCMSGI can only be done from this routine). Part of the convention between this routine and mainline code should be a check to make sure the mainline code has enough storage to accept the full message. The message routine gets the length of the message from the message header. If this check is not done, the message exit routine will get an 0C4 and, as it runs as an SRB, will simply disappear. This is very hard to detect as it leaves no traces – it simply does not work.

Experience over the last few years has shown that finding some non-standard Sysplex solutions often depends on the ingenuity of the systems programmer. New problems particular to any site come up from time to time and in many cases somewhat unconventional ways have to be found to make things work. The better and more flexible your service provider, the more use your site will get from it and, until such time that all vendors have got their Sysplex act together, you may have to depend on it to solve some interesting problems.

A A Keyser
Systems Programmer
Houghton Consulting Services Pty Ltd (Australia)

Cross-memory services – a working example

INTRODUCTION

The problem facing us was to pass a request from an OS/2 application to an APL program, to process the request, and then to return the result. The chosen way of moving the data between the different platforms was:

1 To use APPC services between the OS/2 program and an APPC/MVS program.

2 To use cross-memory services between the APPC/MVS program and a started task.

3 To use APL services between the started task and the APL program.

I was responsible for coding all programs except the OS/2 and APL programs. These were new challenges. I coded a batch program and a started task, which could communicate through cross-memory services, and a test driver on both sides. The OS/2 and the APL people became involved later on. Then the APL solution changed and my started task never came into production, but all my coding concerning cross-memory services was moved unchanged to the APL user, and has run in production without problems for a long period.

THE TEST DRIVER TO START THE PROCESS (CMSIFRUN)

This test driver was coded to simulate the APPC/MVS program, and was executed with the following JCL:

```
//        EXEC PGM=CMSIFRUN,PARM='APIxxxx,ttt,variable_data'
//LOGNING DD   SYSOUT=*
```

where *APIxxxxx* is the program that will process the request (in this case the test driver that simulate an APL program), *ttt* is the time-out value in seconds, and *variable_data* is the request and data given to the API.

This program links to the CMS_APPC_SERVER (program CMSIFPGM). The communication area between the two programs is GETMAINed in this program and is described in the DSECT CMS_PARAMETER.

THE CMS_APPC_SERVER (CMSIFPGM)

This batch program receives the *variable_data* from the test driver CMSIFRUN. Using cross memory services, the request and data are then passed to a communication area defined in the CMS_API_SERVER (the started task CMSIFSTC).

First the address space of the CMS_API_SERVER is found by picking out the task that has the same job name as the started task. Then the entry point address of the CMS_API_SERVER is loaded, and, by looking at the value at offset X'4' in that program, we can confirm that

we have found the correct task. This means that both the name of the started task and the value at offset X'4' of that task have to be fixed values that are known to this program. At offset X'C' of the started task we finally find the address of its communication area. This offset too has to be known, and fixed, in both programs.

The *variable_data*, an ECB in my task to be posted on return, and the address of the PARMLIST in my task (where to place the answer) are passed to the CMS_API_SERVER. Then a time-out request is set, an ECB in the CMS_API_SERVER is posted to wake up the started task, and my task goes into a wait. When the request has processed and an answer has been built, the CMS_API_SERVER places it in my PARMLIST, using cross-memory services, and posts my ECB.

The communication area between the CMS_APPC_SERVER and the CMS_API_SERVER is GETMAINed in the started task and is described in the DSECT STC_PARAMETER.

THE CMS_API_SERVER (CMSIFSTC) – A STARTED TASK

This program/started task receives the *variable_data* from the CMS_APPC_SERVER, passes the request and data to the API (program *APIxxxx*), and returns the answer to the CMS_APPC_SERVER. It is started at the console by typing the following:

```
S CMSIFSTC,PARM='nnn,APIxxxxx,ttt,APIxxxxx,ttt'
```

where *nnn* is the number of APIs this task will have to serve, and this value is followed by the name of the API and the time-out value in seconds for each of these APIs. All these APIs will be processed concurrently, and the same API can be written several times, allowing parallel processing of an API.

This started task will be driven by ECBs, and it will have one ECB to the console interface and three ECBs per API to serve (one for communication with the CMS_API_SERVER, one for communication with the API, and one to be posted if the API times out).

First the program will GETMAIN its working storage, consisting of an ECB list (described in DSECT ECB_PARAMETER), and one communication area per API to serve (described in DSECT

STC_PARAMETER), and the ECB list and the communication areas will be initialized. Then the console interface will be initialized, allowing one queued command, driven by an ECB. When the initialization is complete, offset X'C' of this task is set up to point to a parameter list, which again is pointing to the ECB list and to the communications areas. This address at offset X'C' is the same as used by program CMSIFPGM.

The started task will then start a loop, waiting on work to do, by waiting on the ECB list; the started task will be woken up when one or more of its ECBs is posted, and, when all post requests are processed, the started task will wait again.

If the started task is woken up by a post request from the CMS_APPC_SERVER, the requested API will be attached as a subtask. An ECB waiting on the subtask to complete and an ECB for the time-out of the subtask will be set.

If the started task is woken up by a post request from a timer request (a time-out of the subtask) the subtask is detached and a return code is set. This return code is returned to the CMS_APPC_SERVER.

If the started task is woken up by a post request from the subtask API, the time-out request will be cancelled. The answer from the API will be moved to the CMS_APPC_SERVER using cross-memory services, and an ECB in the CMS_APPC_SERVER is posted to wake up that task.

Finally, the started task can be posted by a modify or stop command from the console. The command F CMSIFSTC,STAT gives the number of outstanding requests, which is the number of requests from the CMS_APPC_SERVER minus the number of answers/time-outs from the APIs. The command 'F CMSIFSTC,VIEW' gives a view for each of the APIs to serve, in the form of an identifier, the name of the API, and the status of the API. The given identifier can be used in the command 'F CMSIFSTC, KILL *xxxx*' to kill any outstanding requests. All responses from the modify commands are in the form of WTOs.

The CMS_API_SERVER can be shut down using the stop command 'P CMSIFSTC' and, after a shut-down request, no incoming request will be received from the CMS_APPC_SERVER. If at shut-down time there are any outstanding requests, the task will wait on completion of

these: a second stop command however will immediately force down all outstanding requests.

THE API (API00001) – TEST DRIVER TO PROCESS THE REQUEST

This program is attached as a subtask from the CMS_API_SERVER. It is called with five parameters: a return code, the length of the request, the request (which is the *variable_data* from the program CMSIFRUN), the length of the answer, and the answer itself.

The functions in this test API are decided by the first four characters in the received request: ECHO will echo the received request, WTOR will return an answer received from the console through a write to operator with reply, WAIT will wait 60 seconds before returning an answer, ABND will abend the API, ERRC will return a non-zero return code, LOOP will wait for 25 seconds and then go into a loop, and any others will inform the user how many characters the API received.

CMSIFRUN SOURCE

```
*************************************************************************
*  MODULE NAME :  CMSIFRUN
*  FUNCTION    :  THIS IS A TEST DRIVER
*                 - LINK TO CMS_APPC_SERVER
*  REQUIREMENT :  LINK WITH AMODE=31 AND RMODE=24
*************************************************************************
         TITLE 'CMSIFRUN : REGISTER USING'
R6       EQU   6           BASE FOR COMMAREA TO CMS_APPC_SERVER
BALREG   EQU   9           BAL-REGISTER
R11      EQU   11          BASE REGISTER
R13      EQU   13          ADDRESS OF SAVE AREA
R14      EQU   14          RETURN REGISTER
R15      EQU   15          ENTRY ADDRESS + RETURN CODE
         TITLE 'CMSIFRUN : INITIAL SET UP'
CMSIFRUN CSECT
         USING CMSIFRUN,R15       ESTABLISH TEMPORARY MODULE
         B     ENTRYLNK           BRANCH AROUND EYECATCHERS
         DC    CL8'CMSIFRUN'      EYECATCHERS FOR START OF MODULE
         DC    CL8'&SYSDATE'      .. AND DATE OF ASSEMBLY
         DC    CL8'&SYSTIME'      .. AND TIME OF ASSEMBLY
ENTRYLNK DS    ØH
         STM   R14,R12,12(R13)    SAVE CALLER'S REGISTERS
         LR    R11,R15            SET MODULE BASE REGISTER
         DROP  R15                DROP TEMPORARY MODULE ADDRESSABILITY
         USING CMSIFRUN,R11       ESTABLISH MODULE ADDRESSABILITY
         LA    R15,SAVEAREA       GET ADDRESS OF MY SAVE AREA
```

```
         ST    R15,8(R13)          SAVE FORWARD POINTER
         ST    R13,SAVEAREA+4      SAVE BACKWARD POINTER
         LA    R13,SAVEAREA        SET UP R13 TO POINT TO SAVE AREA IN
*                                  .. CASE THIS MODULE MAKES A CALL
         ST    R1,R1_SAVE          STORE REG1 AT ENTRY
OPEN_LOG DS    ØH
         OPEN  (LOGDCB,(OUTPUT)),MODE=31
         LTR   R15,R15             WAS OPEN OK ?
         BZ    OPEN_Ø1             .. YES
         DC    X'FFFF'             .. NO, ABEND THE PROGRAM
OPEN_Ø1  DS    ØH
         MVC   LOGLIN+1(L'MSGSTRT),MSGSTRT
         BAL   BALREG,WRITELOG      WRITE LOG-MESSAGE
         TITLE 'CMSIFRUN : GET DATA FROM THE INPUT PARAMETER'
GET_DATA DS    ØH
         LA    R2,CMS_LENGTH       COMMAREA LENGTH
         LA    R2,4ØØØ(R2)         TRANSACTION_DATA
         LA    R2,4ØØØ(R2)         ANSWER
         STORAGE OBTAIN,LOC=BELOW,                                    *
               LENGTH=(R2),ADDR=(R6)
         USING CMS_PARAMETER,R6    SET UP ADESSING OF COMMAREA
         LA    R3,CMS_LENGTH       COMMAREA LENGTH
         AR    R3,R6
         ST    R3,CMS_ADDR_TRNS    R3 -> TRANSACTION_DATA
         LA    R4,4ØØØ
         AR    R4,R3
         ST    R4,CMS_ADDR_ANSW    R4 -> ANSWER
         L     R1,R1_SAVE          LOAD PARMLIST_ADDRESS
         L     R1,Ø(R1)            R1 -> PARMLIST
         LH    R2,Ø(R1)            R2 = LENGTH OF PARM_STRING
         MVC   CMS_API_NAME,2(R1)
         PACK  DOB,11(3,R1)
         CVB   R4,DOB
         MH    R4,=H'1ØØ'
         ST    R4,CMS_TIMEOUT
         SH    R2,=H'13'
         STH   R2,CMS_LENG_TRNS
         BCTR  R2,Ø
         EX    R2,DATA_MVS
         B     DATA_9Ø
DATA_MVS MVC   Ø(Ø,R3),15(R1)
DATA_9Ø  DS    ØH
         LA    R2,LOGDCB
         ST    R2,CMS_LOG_DCBA
         TITLE 'CMSIFRUN : LINK TO CMS_APPC_SERVER'
LINK_PRG DS    ØH
         LA    R2,CMS_COMMAREA
         LINK  EP=CMSIFPGM,PARAM=((R2))
         MVC   LOGLIN+1(L'MSGANSW),MSGANSW
         UNPK  WORK9(5),CMS_LENG_ANSW(3)
         TR    WORK9(4),HEXTAB
```

```
         MVC   LOGLIN+16(4),WORK9
         L     R2,CMS_ADDR_ANSW
         MVC   LOGLIN+27(50),0(R2)
         BAL   BALREG,WRITELOG     WRITE LOG-MESSAGE
         MVC   LOGLIN+1(L'MSGRETN),MSGRETN
         MVC   LOGLIN+22(3),CMS_RETNCODE
         BAL   BALREG,WRITELOG     WRITE LOG-MESSAGE
         TITLE 'CMSIFRUN : EXIT THE PROGRAM'
END_PROG DS    0H
         MVC   LOGLIN+1(L'MSGENDP),MSGENDP
         BAL   BALREG,WRITELOG     WRITE LOG-MESSAGE
         CLOSE (LOGDCB),MODE=31
         L     R13,SAVEAREA+4      RETRIEVE BACKWARD POINTER
         LA    R15,0               GET DESIRED RETURN CODE
         RETURN (14,12),RC=(15)    RETURN CONTROL TO SYSTEM
         TITLE 'CMSIFRUN : ROUTINE TO WRITE A LOG-MESSAGE'
WRITELOG DS    0H
         L     R1,MODE24S
         BSM   R0,R1
MODE24S  DC    A(MODE24+X'00000000')
MODE24   DS    0H
         LA    R2,LOGLIN
         PUT   LOGDCB,(R2)
         L     R1,MODE31S
         BSM   R0,R1
MODE31S  DC    A(MODE31+X'80000000')
MODE31   DS    0H
         MVI   LOGLIN+1,C' '
         MVC   LOGLIN+2(131),LOGLIN+1
         BR    BALREG
         TITLE 'CMSIFRUN : DCB-S'
LOGDCB   DCB   DDNAME=LOGNING,                                          *
               DSORG=PS,                                                *
               MACRF=PM,                                                *
               RECFM=FBM,                                               *
               LRECL=133,                                               *
               BLKSIZE=1330
         TITLE 'CMSIFRUN : WORK AREAS'
SAVEAREA DS    18F            SAVE_AREA IN MY_MODULE
R1_SAVE  DS    F              REG1 AT ENTRY
RET_CODE DS    F              RETURN CODE FROM THIS TASK
HEXTAB   EQU   *-C'0'         WORK FIELDS
         DC    C'0123456789ABCDEF'
WORK9    DS    CL9
DOB      DS    D
LOGLIN   DC    X'09',CL132' '     LINE TO LOG
MSGSTRT  DC    C'CMSIFRUN : START OF APPLICATION'
MSGANSW  DC    C'CMSIFRUN : LGD=XXXX, ANSW=XXXXXXXXXXXXXXXXXXXXXXXXXXXXX*
               XXXXXXXXXXXXXXXXXXXXXX'
MSGRETN  DC    C'CMSIFRUN : RETN.CODE=XXX'
MSGENDP  DC    C'CMSIFRUN : END OF APPLICATION'
```

```
         LTORG
         TITLE 'CMSIFRUN : COMMUNICATION_AREA IN THE TESTDRIVER'
CMS_PARAMETER DSECT
CMS_COMMAREA  DS  ØF          COMMUNICATION AREA IN THE APPC_INTERFACE
CMS_RETNCODE  DS  CL3         .. RETURN_CODE
CMS_FILLER    DS  XL1         .. FILLER
CMS_LENG_TRNS DS  H           .. LENGTH OF TRANSACTION_DATA
CMS_LENG_ANSW DS  H           .. LENGTH OF ANSWER
CMS_ADDR_TRNS DS  F           .. ADDRESS OF TRANSACTION_DATA
CMS_ADDR_ANSW DS  F           .. ADDRESS OF ANSWER
CMS_API_NAME  DS  CL8         .. NAME OF API TO EXECUTE
CMS_TIMEOUT   DS  F           .. TIMEOUT VALUE IN THE CMS_APPC_SERVER
CMS_TIMESTMP  DS  CL16        .. TIMESTAMP SET BY THE CMS_APPC_SERVER
CMS_LOG_DCBA  DS  F           .. ADDRESS OF DCB FOR LOGGING
CMS_LENGTH    EQU *-CMS_COMMAREA
         END
```

CMSIFPGM SOURCE

```
*********************************************************************
*  MODULE NAME :  CMSIFPGM
*  FUNCTION    :  THIS PROGRAM IS LINKED TO BY THE APPC_INTERFACE
*                 ITS MAIN FUNCTIONS ARE
*                 .. TO RECEIVE DATA FROM THE APPC_INTERFACE
*                 .. TO PASS DATA THROUGH TO THE CMS_API_SERVER
*                 ..   AND POST AN ECB IN THIS ADDRESS SPACE
*                 .. TO WAIT ON THE ANSWER, BUILT BY THE CMS_API_SERVER
*                 ..   WHICH ALSO POSTS AN ECB IN MY ADDRESS SPACE
*                 .. AND TO SEND THE ANSWER BACK TO THE APPC_INTERFACE
*  REQUIREMENT :  LINK WITH AC(1)
*                 LINK WITH AMODE=31 AND RMODE=24
*                 EXECUTE FROM AUTHORIZED LIBRARY
*********************************************************************
         TITLE 'CMSIFPGM : REGISTER USING'
R6       EQU   6           BASE FOR COMMAREA FROM THE APPC_INTERFACE
BALREG   EQU   7           BAL-REGISTER
R8       EQU   8           WORK, ONLY USED IN SECONDARY ADDRESS SPACE
R9       EQU   9           WORK, ONLY USED IN SECONDARY ADDRESS SPACE
R1Ø      EQU   1Ø          WORK, ONLY USED IN SECONDARY ADDRESS SPACE
R11      EQU   11          BASE REGISTER 1
R12      EQU   12          BASE REGISTER 2
R13      EQU   13          ADDRESS OF SAVE AREA
R14      EQU   14          RETURN REGISTER
R15      EQU   15          ENTRY ADDRESS + RETURN CODE
         TITLE 'CMSIFPGM : INITIAL SET UP'
CMSIFPGM CSECT
         USING CMSIFPGM,R15       ESTABLISH TEMPORARY MODULE
         B     ENTRYLNK           BRANCH AROUND EYECATCHERS
         DC    CL8'CMSIFPGM'      EYECATCHERS FOR START OF MODULE
         DC    CL8'&SYSDATE'      .. AND DATE OF ASSEMBLY
```

```
         DC    CL8'&SYSTIME'       .. AND TIME OF ASSEMBLY
ENTRYLNK DS    ØH
         STM   R14,R12,12(R13)     SAVE CALLER'S REGISTERS
         LR    R11,R15             SET MODULE BASE REGISTER 1
         LR    R12,R15             ..
         LA    R12,4Ø95(R12)       ..
         LA    R12,1(R12)          .. AND BASE REGISTER 2
         DROP  R15                 DROP TEMPORARY MODULE ADDRESSABILITY
         USING CMSIFPGM,R11,R12    ESTABLISH MODULE ADDRESSABILITY
         LA    R15,SAVEAREA        GET ADDRESS OF MY SAVE AREA
         ST    R15,8(R13)          SAVE FORWARD POINTER
         ST    R13,SAVEAREA+4      SAVE BACKWARD POINTER
         LA    R13,SAVEAREA        SET UP R13 TO POINT TO SAVE AREA IN
*                                  .. CASE THIS MODULE MAKES A CALL
         L     R6,Ø(R1)            R6 -> COMMAREA FROM APPC_INTERFACE
         USING CMS_COMMAREA,R6     SET UP ADDRESSING FOR COMMAREA
         ESTAE ABNDEXIT            ESTABLISH ABEND HANDLING
         L     R4,PSAAOLD-PSA(Ø)   R4 -> ASCB ADDRESS OF OWN_TASK
         USING ASCB,R4             SET UP ADDRESSING FOR ASCB
         LH    R5,ASCBASID         R5 = ADDRESS SPACE NUMBER OWN_TASK
         ST    R4,CMS_ASCB         STORE ASCB ADDRESS OF OWN_TASK
         ST    R5,CMS_ASID         STORE ADDRESS SPACE NUMBER OF SAME
         DROP  R4
         TIME  DEC,CMS_TIMESTMP,   GET A TIMESTAMP                       *
               ZONE=LT,DATETYPE=DDMMYYYY,LINKAGE=SYSTEM
         TITLE 'CMSIFPGM : FIND ASCB ADDRESS OF THE CMS_API_SERVER'
FND_ASCB DS    ØH
         L     R4,CVTPTR           R4 -> CVT
         USING CVT,R4              SET UP ADDRESSING FOR CVT
         L     R4,CVTASVT          R4 -> ASVT
         USING ASVT,R4             SET UP ADDRESSING FOR ASVT
         LA    R2,ASVTFRST         R2 -> 1ST ENTRY IN ASVT
         L     R3,ASVTMAXU         R3 = MAX NUMBER OF ADDRESS SPACES
ASCB_Ø1  DS    ØH
         TM    Ø(R2),ASVTAVAI      IS ADDRESS SPACE NUMBER UNUSED ?
         BNO   ASCB_Ø2             .. NO
         C     R2,=A(X'8ØØØØØØØ')  IS THIS LAST ASVT ENTRY ?
         BE    ER_1Ø1_1            .. YES, ASID NOT FOUND
         B     ASCB_Ø4             .. NO, CHECK NEXT ASCB
ASCB_Ø2  DS    ØH
         L     R4,Ø(R2)            R4 -> ASCB
         USING ASCB,R4             SET UP ADDRESSING FOR ASCB
         L     R5,ASCBJBNS         R5 -> JOB NAME FIELD FOR START
         CLC   =CL8'INIT',Ø(R5)    IS THIS AN INITIATOR ?
         BNE   ASCB_Ø3             .. NO
         ICM   R5,15,ASCBJBNI      .. YES, R5 -> JOB NAME FIELD IN INIT
         BZ    ASCB_Ø4             IF Ø, IT IS AN UNUSED INITIATOR
ASCB_Ø3  DS    ØH
         CLC   STC_NAME,Ø(R5)      IS THIS THE CMS_API_SERVER ?
         BE    ASCB_Ø5             .. YES
ASCB_Ø4  DS    ØH
```

```
         LA    R2,L'ASVTENTY(R2)  R2 -> NEXT ENTRY IN ASVT
         BCT   R3,ASCB_Ø1         CHECK NEXT ASVT ENTRY
         B     ER_1Ø1_1           NO MORE, ADDRESS SPACE NOT FOUND
ASCB_Ø5  DS    ØH
         TM    ASCBRCTF,ASCBOUT   IS ADDRESS SPACE SWAPPED OUT ?
         BO    ER_1Ø1_2           .. YES, NOT USABLE
         LH    R5,ASCBASID        R5 = ADDRESS SPACE NO CMS_API_SERVER
         ST    R4,STC_ASCB        STORE ASCB ADDRESS OF CMS_API_SERVER
         ST    R5,STC_ASID        STORE ADDRESS SPACE NUMBER OF SAME
         DROP  R4
         TITLE 'CMSIFPGM : MOVE DATA TO THE SECONDARY ADDRESS SPACE AND*
               WAKE IT UP'
MOVE_SEC DS    ØH
         MODESET MODE=SUP         SET SUPERVISOR STATE
         SYSEVENT DONTSWAP        MAKE ADDRESS SPACE NON-SWAPPABLE
         MVI   SWAP_SET,DONTSWAP
         LA    RØ,1               GET AX
         AXSET AX=(RØ)            .. OF 1
         ST    RØ,CMS_AX          SAVE THE OLD AX
         L     R2,STC_ASID        GET ADDR SPACE NUMBER CMS_API_SERVER
         SSAR  R2                 SET AS SECONDARY ADDRESS SPACE
         LAM   R8,R1Ø,ALET1       LOAD ALET1 INTO AR 8 TO 1Ø
         SAC   512                SET AR MODE
         L     R8,STC_ASCB        LOAD ASCB ADDRESS OF CMS_API_SERVER
         USING ASCB,R8            SET UP ADDRESSING FOR ASCB
         L     R8,ASCBASXB        LOAD ASXB ADDRESS IN SECONDARY ASN
         USING ASXB,R8            SET UP ADDRESSING FOR ASXB
         L     R8,ASXBFTCB        LOAD 1ST TCB ADRESS IN SECONDARY ASN
         USING TCB,R8             SET UP ADDRESSING FOR TCB
MOVE_Ø1  DS    ØH
         ICM   R9,15,TCBJPQ       LOAD ADDR OF JOB-PACK-AREA QUEUE
         USING CDENTRY,R9         SET UP ADDRESSING FOR CDE
         BZ    MOVE_Ø3            NO ADDRESS, NO MODULES FOR TCB
MOVE_Ø2  DS    ØH
         CLC   STC_NAME,CDNAME    IS THIS THE TCB OF CMS_API_SERVER ?
         BE    MOVE_Ø5            .. YES
         ICM   R9,15,CDCHAIN      .. NO, LOAD ADDR. NEXT CDE ON JPAQ
         BNZ   MOVE_Ø2            IF ANY, GO AND PROCESS
MOVE_Ø3  DS    ØH
         ICM   R8,15,TCBTCB       LOAD ADDR. OF NEXT TCB ON QUEUE
         BNZ   MOVE_Ø1            IF ANY, GO AND PROCESS
         OI    SWITCH,GONE_STC    SORRY, CMS_API_SERVER HAS GONE
         B     MOVE_5Ø
MOVE_Ø5  DS    ØH
         L     R9,CDENTPT         LOAD EP_ADDRESS OF THE CMS_API_SERVER
         CLC   STC_IDNT,4(R9)     CAN WE IDENTIFY THE CMS_API_SERVER ?
         BE    MOVE_Ø6            .. YES
         OI    SWITCH,WRNG_STC    .. NO, IT IS NOT OUR CMS_API_SERVER
         B     MOVE_5Ø
         DROP  R8
         DROP  R9
```

```
MOVE_Ø6  DS    ØH
         L     R1Ø,12(Ø,R9)        LOAD ADDRESS OF COMMAREA_ADDRESS
         LTR   R1Ø,R1Ø             IS CMS_API_SERVER STARTING UP ?
         BZ    MOVE_Ø7             .. YES
         C     R1Ø,=F'-1'          IS CMS_API_SERVER CLOSING DOWN ?
         BE    MOVE_Ø8             .. YES
         B     MOVE_Ø9             ALL IS OK
MOVE_Ø7  DS    ØH
         OI    SWITCH,STRT_STC     CMS_API_SERVER IS STARTING
         B     MOVE_5Ø
MOVE_Ø8  DS    ØH
         OI    SWITCH,STOP_STC     CMS_API_SERVER IS STOPPING
         B     MOVE_5Ø
MOVE_Ø9  DS    ØH                  CMS_API_SERVER IS READY FOR USE
         OI    SWITCH,NFND_STC     MARK THAT ENTRY NOT FOUND YET
         L     R1Ø,Ø(Ø,R1Ø)        LOAD ADDR OF 1ST COMMUNICATION AREA
         USING STC_PARAMETER,R1Ø   SET UP ADDR COMMAREA CMS_API_SERVER
MOVE_1Ø  DS    ØH
         CLC   STC_API_NAME,CMS_API_NAME  FOUND COMMAREA FOR OUR API ?
         BNE   MOVE_11                    .. NO
         NI    SWITCH,EFND_STC            .. YES, MARK THAT ENTRY FOUND
         OI    SWITCH,BUSY_STC            MARK THAT ENTRIES ARE BUSY
         CLI   STC_STATUS,STC_STA_FREE    IS COMMAREA ENTRY FREE ?
         BNE   MOVE_11                    .. NO
         NI    SWITCH,FREE_STC            .. YES, MARK FREE ENTRY FOUND
         BE    MOVE_2Ø                    ALL IS OK
MOVE_11  DS    ØH
         L     R1Ø,STC_NEXT_ADR   LOAD ADDR OF NEXT COMMUNICATION AREA
         LTR   R1Ø,R1Ø            MORE COMMAREAS ?
         BNZ   MOVE_1Ø            .. YES
         B     MOVE_5Ø
MOVE_2Ø  DS    ØH
         MVI   STC_STATUS,STC_STA_APPC  MARK COMMAREA ENTRY IN_USE
*                                       MOVE TO SECONDARY ADDRESS SPACE
         MVC   STC_CMS_ASCB,CMS_ASCB        .. ASCB ADDR THIS TASK
         MVC   STC_CMS_ASID,CMS_ASID        .. ASID THIS TASK
         MVC   STC_CMS_ECBA,CMS_ECBA        .. ECB ADDR THIS TASK
         LA    R2,CMS_COMMAREA              .. ADDRESS OF PARMLIST IN
         ST    R2,STC_CMS_PARM              ..... THE CMS_APPC_SERVER
         MVC   STC_CMS_TIME,CMS_TIMESTMP    .. TIMESTAMP FOR CHECK
         MVC   STC_LENG_TRNS,CMS_LENG_TRNS  .. LENGTH OF DATA
         L     R8,STC_ADDR_TRNS             ..
         L     R2,CMS_ADDR_TRNS             ..
         LH    R9,CMS_LENG_TRNS             ..
         LH    R3,CMS_LENG_TRNS             ..
         MVCL  R8,R2                        .. TRANSACTION_DATA
*                                       GET FROM SECONDARY ADDR SPACE
         LA    R2,STC_ECB_APPC              ..
         ST    R2,STC_ECBA                  .. ECB ADDR SEC SPACE
         DROP  R1Ø
MOVE_5Ø  DS    ØH
```

```
         L     R2,CMS_ASID        RESTORE
         SSAR  R2                 .. ORIGINAL SASN
         SAC   Ø                  RESTORE ADDRESSING MODE
         L     RØ,CMS_AX          RESTORE
         AXSET AX=(RØ)            .. OLD AX
         CLI   SWITCH,X'ØØ'       IS EVERYTHING OK ?
         BE    MOVE_9Ø            .. YES
         TM    SWITCH,GONE_STC    HAS CMS_API_SERVER GONE ?
         BO    ER_1Ø1_3           .. YES
         TM    SWITCH,WRNG_STC    WAS IT WRONG CMS_API_SERVER ?
         BO    ER_1Ø1_4           .. YES
         TM    SWITCH,STRT_STC    WAS CMS_API_SERVER STARTING UP ?
         BO    ER_1Ø1_5           .. YES
         TM    SWITCH,STOP_STC    WAS CMS_API_SERVER CLOSING DOWN ?
         BO    ER_1Ø1_6           .. YES
         TM    SWITCH,NFND_STC    WAS ENTRY FOR API NOT FOUND ?
         BO    ER_1Ø2             .. NO
         TM    SWITCH,BUSY_STC    WAS CMS_API_SERVER BUSY ?
         BO    ER_1Ø3             .. YES
         B     ER_1Ø4             UNKWOWN REASON
MOVE_9Ø  DS    ØH
         L     R2,STC_ECBA        R2 = ECB ADDRESS IN CMS_API_SERVER
         L     R4,STC_ASCB        R4 = ASCB ADDRESS OF CMS_API_SERVER
         POST  (R2),ASCB=(R4),    POST THE ECB                          *
               LINKAGE=SYSTEM,    .. IN THE SECONDARY ADDRESS SPACE     *
               ERRET=ER_1Ø5,      IN CASE THE POST IS FAILING           *
               ECBKEY=Ø
         TITLE 'CMSIFPGM : WAIT ON RETURN FROM THE SECONDARY ADDRESS SP*
               ACE'
WAIT_SEC DS    ØH
         MVC   LOGLIN+1(L'MSGWAIT),MSGWAIT
         UNPK  WORK9,STC_ASCB(5)
         TR    WORK9(8),HEXTAB
         MVC   LOGLIN+34(8),WORK9
         UNPK  WORK9,STC_ECBA(5)
         TR    WORK9(8),HEXTAB
         MVC   LOGLIN+53(8),WORK9
         UNPK  WORK9,CMS_ECBA(5)
         TR    WORK9(8),HEXTAB
         MVC   LOGLIN+85(8),WORK9
         BAL   BALREG,WRITELOG    WRITE A LOG MESSAGE
         LA    R2,CMS_TIMEOUT     TIMEOUT_VALUE IN THE API_SERVER
*                                 SET A TIMER REQUEST
         STIMERM SET,                                                 *
               BINTVL=(2),                                            *
               EXIT=TIMEEXIT,                                         *
               ID=TIME_ID,                                            *
               ERRET=ER_1Ø6
         WAIT  ECBLIST=ECBLIST    WAIT ON CMS_API_SERVER OR TIME-OUT
         SYSEVENT OKSWAP          MAKE ADDRESS SPACE SWAPPABLE
         MVI   SWAP_SET,OKSWAP
```

```
         MODESET MODE=PROB        RESET TO PROBLEM STATE
         TM    TIME_ECB,X'40'       TIMEOUT_ECB POSTED ?
         BO    ER_100                .. YES
         STIMERM CANCEL,ID=TIME_ID  .. NO , CANCEL THE TIMER REQUEST
         LA    R15,0              SET RETURN CODE = 0
         ST    R15,RET_CODE       STORE IT FOR LATER USE
         B     END_PROG
         TITLE 'CMSIFPGM : ERROR HANDLING'
ER_100   DS    0H
         MVC   CMS_RETNCODE,=C'100'  SET RETURN_CODE 100
         MVC   WTOLABL+8(3),=C'100'
         MVC   WTOLABL+24(40),WTO100
         B     ER_900
ER_101_1 DS    0H
         MVC   WTOLABL+52(12),WTO101_1
         B     ER_101_9
ER_101_2 DS    0H
         MVC   WTOLABL+52(12),WTO101_2
         B     ER_101_9
ER_101_3 DS    0H
         MVC   WTOLABL+52(12),WTO101_3
         B     ER_101_9
ER_101_4 DS    0H
         MVC   WTOLABL+52(12),WTO101_4
         B     ER_101_9
ER_101_5 DS    0H
         MVC   WTOLABL+52(12),WTO101_5
         B     ER_101_9
ER_101_6 DS    0H
         MVC   WTOLABL+52(12),WTO101_6
         B     ER_101_9
ER_101_9 DS    0H
         MVC   CMS_RETNCODE,=C'101'  SET RETURN_CODE 101
         MVC   WTOLABL+8(3),=C'101'
         MVC   WTOLABL+24(28),WTO101
         B     ER_900
ER_102   DS    0H
         MVC   CMS_RETNCODE,=C'102'  SET RETURN_CODE 102
         MVC   WTOLABL+8(3),=C'102'
         MVC   WTOLABL+24(40),WTO102
         MVC   WTOLABL+56(8),CMS_API_NAME
         B     ER_900
ER_103   DS    0H
         MVC   CMS_RETNCODE,=C'103'  SET RETURN_CODE 103
         MVC   WTOLABL+8(3),=C'103'
         MVC   WTOLABL+24(40),WTO103
         MVC   WTOLABL+55(8),CMS_API_NAME
         B     ER_900
ER_104   DS    0H
         MVC   CMS_RETNCODE,=C'104'  SET RETURN_CODE 104
         MVC   WTOLABL+8(3),=C'104'
```

```
         MVC   WTOLABL+24(40),WTO104
         B     ER_900
ER_105   DS    0H
         MVC   CMS_RETNCODE,=C'105'  SET RETURN_CODE 105
         MVC   WTOLABL+8(3),=C'105'
         MVC   WTOLABL+24(40),WTO105
         ST    R15,R15_SAVE
         UNPK  WORK9,R15_SAVE(5)
         TR    WORK9(8),HEXTAB
         MVC   WTOLABL+55(8),WORK9
         B     ER_900
ER_106   DS    0H
         MVC   CMS_RETNCODE,=C'106'  SET RETURN_CODE 106
         MVC   WTOLABL+8(3),=C'106'
         MVC   WTOLABL+24(40),WTO106
         ST    R15,R15_SAVE
         UNPK  WORK9,R15_SAVE(5)
         TR    WORK9(8),HEXTAB
         MVC   WTOLABL+44(8),WORK9
         B     ER_900
ER_900   DS    0H
         WTO   MF=(E,WTOLABL)
         CLI   SWAP_SET,OKSWAP    IS TASK SWAPPABLE ?
         BE    ER_901             .. YES, ALL IS OK
         SYSEVENT OKSWAP          MAKE ADDRESS SPACE SWAPPABLE
         MVI   SWAP_SET,OKSWAP
         MODESET MODE=PROB        RESET TO PROBLEM STATE
ER_901   DS    0H
         LA    R15,12             SET RETURN CODE = 12
         ST    R15,RET_CODE       STORE IT FOR LATER USE
         B     END_PROG
         TITLE 'CMSIFPGM : EXIT THE PROGRAM'
END_PROG DS    0H
         L     R13,SAVEAREA+4     RETRIEVE BACKWARD POINTER
         L     R15,RET_CODE       GET DESIRED RETURN CODE
         RETURN (14,12),RC=(15)   RETURN CONTROL TO SYSTEM
         TITLE 'CMSIFPGM : ROUTINE TO LOG A MESSAGE'
WRITELOG DS    0H
         L     R1,MODE24S
         BSM   R0,R1
MODE24S  DC    A(MODE24+X'00000000')
MODE24   DS    0H
         LA    R2,LOGLIN
         L     R3,CMS_LOG_DCBA
         PUT   (R3),(R2)
         L     R1,MODE31S
         BSM   R0,R1
MODE31S  DC    A(MODE31+X'80000000')
MODE31   DS    0H
         MVI   LOGLIN+1,C' '
         MVC   LOGLIN+2(131),LOGLIN+1
```

```
         BR    BALREG
         TITLE 'CMSIFPGM : WORK AREAS'
SAVEAREA DS    18F           SAVE_AREA IN MY_MODULE
SAVEABND DS    18F           SAVE_AREA IN ABEND_EXIT
SAVETIME DS    18F           SAVE_AREA IN TIMEOUT_EXIT
ALET1    DC    16F'1'        IN ACCESS REGISTER MODE,
*                            .. AN ACCESS REGISTER THAT CONTAINS AN
*                            .. ALET OF X'00000001' CAUSES
*                            .. DAT TRANSLATION USING THE SECONDARY STD
ECBLIST  DS    0F            LIST OF ECB ADDRESSES
CMS_ECBA DC    AL4(CMS_ECB)              .. ADDR OF ECB WAITING
         DC    AL4(TIME_ECB+X'80000000')  .. ADDR OF TIMEOUT_ECB
STC_NAME DC    CL8'CMSIFSTC' NAME OF THE CMS_API_SERVER
STC_IDNT DC    CL8'CMS__STC' IDENTIFICATION OF THE CMS_API_SERVER
CMS_ASCB DS    F             ASCB ADDRESS OF THIS ADDRESS SPACE
CMS_ASID DS    F             ADDRESS SPACE NUMBER OF THIS ADDRESS SPACE
STC_ASCB DS    F             ASCB ADDRESS OF THE CMS_API_SERVER
STC_ASID DS    F             ADDRESS SPACE NUMBER OF THE CMS_API_SERVER
STC_ECBA DS    AL4           ADDRESS OF ECB IN CMS_API_SERVER
CMS_AX   DS    F             OLD AX
CMS_ECB  DC    F'0'          ECB, POSTED ON RETURN FROM CMS_API_SERVER
TIME_ECB DC    F'0'          ECB, POSTED AT TIME-OUT
TIME_ID  DS    F             TIMER REQUEST IDENTIFIER
ADR_SDWA DS    F             SDWA ADDRESS
R15_SAVE DS    F             REG15 AT POST FAILURE
RET_CODE DS    F             RETURN CODE FROM THIS TASK
SWITCH   DC    X'00'         SWITCH FOR STATUS OF THE CMS_API_SERVER
*                            .. CMS_API_SERVER IS NOT USABLE
GONE_STC EQU   X'80'         ..... IT HAS GONE
WRNG_STC EQU   X'40'         ..... IT IS NOT THE CORRECT TASK
STRT_STC EQU   X'20'         ..... IT IS STARTING
STOP_STC EQU   X'10'         ..... IT IS STOPPING
*                            .. CMS_API_SERVER IS USEABLE
NFND_STC EQU   X'08'         ..... ENTRY FOR API NOT FOUND
EFND_STC EQU   255-X'08'     ........ TO RESET PREVIOUS BIT
BUSY_STC EQU   X'04'         ..... ALL ENTRIES FOR API IS BUSY
FREE_STC EQU   255-X'04'     ........ TO RESET PREVIOUS BIT
SWAP_SET DC    X'00'         SWITCH FOR SWAP STATUS OF THIS TASK
OKSWAP   EQU   X'00'         .. TASK IS SWAPPABLE
DONTSWAP EQU   X'FF'         .. TASK IS NON-SWAPPABLE
HEXTAB   EQU   *-C'0'        WORK FIELDS
         DC    C'0123456789ABCDEF'
WORK9    DS    CL9
DOB      DS    D
WTOLABL  DS    0F
         WTO   'CMS____E  CMSIFPGM:                                    *
               ',ROUTCDE=2,DESC=(7),MF=L
WTO100   DC    CL40'CMS_APPC_SERVER TIMES OUT                '
WTO101   DC    CL28'CMS_APPC_SERVER NOT ACTIVE: '
WTO101_1 DC    CL12'NOT STARTED '
WTO101_2 DC    CL12'SWAPPED OUT '
```

```
WTO101_3 DC    CL12'HAS GONE    '
WTO101_4 DC    CL12'NOT CORRECT '
WTO101_5 DC    CL12'STARTING UP '
WTO101_6 DC    CL12'CLOSING DOWN'
WTO102   DC    CL40'CMS_API_SERVER DO NOT SERVE API XXXXXXXX'
WTO103   DC    CL40'CMS_API_SERVER IS BUSY FOR API XXXXXXXX '
WTO104   DC    CL40'CMS_API_SERVER IS NOT CONTACTED         '
WTO105   DC    CL40'POST CMS_API_SERVER FAILES, RC=XXXXXXXX '
WTO106   DC    CL40'STIMERM FAILES, RC=XXXXXXXX             '
WTO150   DC    CL40'CMS_APPC_SERVER HAS ABENDED             '
LOGLIN   DC    X'09',CL132' '     LOG
MSGWAIT  DC    C'CMSIFPGM : POST_OF_ECB: STC_ASCB=XXXXXXXX, STC_ECBA=XX*
               XXXXXX, WAIT_ON_ECB: CMS_ECBA=XXXXXXXX'
         LTORG
         TITLE 'CMSIFPGM : ABEND EXIT'
ABNDEXIT DS    0H
         STM   R14,R12,12(R13)    SAVE CALLER'S REGISTERS
         DROP  R11,R12
         USING ABNDEXIT,R15
         ICM   R11,B'1111',=AL4(CMSIFPGM)
         ICM   R12,B'1111',=AL4(CMSIFPGM+4096)
         DROP  R15
         USING CMSIFPGM,R11,R12   ESTABLISH MODULE ADDRESSABILITY
         LA    R15,SAVEABND       GET ADDRESS OF ABEND SAVE AREA
         ST    R15,8(R13)         SAVE FORWARD POINTER
         ST    R13,SAVEABND+4     SAVE BACKWARD POINTER
         LA    R13,SAVEABND       SET UP R13 TO POINT TO SAVE AREA IN
*                                 .. CASE THIS MODULE MAKES A CALL
         ST    R1,ADR_SDWA        SAVE SDWA ADDRESS
         MVC   WTOLABL+8(3),=C'150'
         MVC   WTOLABL+24(40),WTO150
         WTO   MF=(E,WTOLABL)
         L     R1,ADR_SDWA        LOAD SDWA ADDRESS
         USING SDWA,R1            ADDRESS SDWA USED BY SETRP MACRO
         L     R13,SAVEABND+4     RETRIEVE BACKWARD POINTER
*                                 CONTINUE ABEND
         SETRP RC=0,REGS=(14,12)
         LTORG
         TITLE 'CMSIFPGM : TIMEOUT EXIT'
TIMEEXIT DS    0H
         STM   R14,R12,12(R13)    SAVE CALLER'S REGISTERS
         DROP  R11,R12
         USING TIMEEXIT,R15
         ICM   R11,B'1111',=AL4(CMSIFPGM)
         ICM   R12,B'1111',=AL4(CMSIFPGM+4096)
         DROP  R15
         USING CMSIFPGM,R11,R12   ESTABLISH MODULE ADDRESSABILITY
         LA    R15,SAVETIME       GET ADDRESS OF ABEND SAVE AREA
         ST    R15,8(R13)         SAVE FORWARD POINTER
         ST    R13,SAVETIME+4     SAVE BACKWARD POINTER
         LA    R13,SAVETIME       SETUP R13 TO POINT TO SAVE AREA IN
```

```
*                                  .. CASE THIS MODULE MAKES A CALL
         POST  TIME_ECB            POST TIMEOUT_ECB
         L     R13,SAVETIME+4      RETRIEVE BACKWARD POINTER
         LM    R14,R12,12(R13)     RESTORE CALLER'S REGISTERS
         BR    R14                 RETURN TO THE SYSTEM
         LTORG
         TITLE 'CMSIFPGM : COMMUNICATION_AREA IN THE APPC_INTERFACE'
CMS_PARAMETER DSECT
CMS_COMMAREA  DS  ØF          COMMUNICATION AREA IN THE APPC_INTERFACE
CMS_RETNCODE  DS  CL3         .. RETURN_CODE
CMS_FILLER    DS  XL1         .. FILLER
CMS_LENG_TRNS DS  H           .. LENGTH OF TRANSACTION_DATA
CMS_LENG_ANSW DS  H           .. LENGTH OF ANSWER
CMS_ADDR_TRNS DS  F           .. ADDRESS OF TRANSACTION_DATA
CMS_ADDR_ANSW DS  F           .. ADDRESS OF ANSWER
CMS_API_NAME  DS  CL8         .. NAME OF API TO EXECUTE
CMS_TIMEOUT   DS  F           .. TIME OUT VALUE IN THE CMS_APPC_SERVER
CMS_TIMESTMP  DS  CL16        .. TIMESTAMP SET BY THE CMS_APPC_SERVER
CMS_LOG_DCBA  DS  F           .. ADDRESS OF DCB FOR LOGGING
         TITLE 'CMSIFPGM : COMMUNICATION_AREA IN THE CMS_API_SERVER'
STC_PARAMETER DSECT
STC_COMMAREA  DS  ØF          COMMUNICATION AREA FOR EACH API TO SERVE
STC_NEXT      EQU *           .. POSITIONING THIS COMMUNICATION AREA
STC_NEXT_ADR  DS  F           ..... ADDRESS OF NEXT COMMUNICATION AREA
STC_INFO      EQU *           .. GENERAL INFORMATION
STC_RETNCODE  DS  CL3         ..... RETURN_CODE
STC_STATUS    DS  XL1         ..... STATUS OF THIS COMMUNICATION AREA
STC_STA_FREE  EQU X'ØØ'       ........ READY_TO_RECEIVE
STC_STA_APPC  EQU X'11'       ........ IN_USE BY THE CMS_APPC_SERVER
STC_STA_API_  EQU X'22'       ........ IN_USE BY THE API
STC_STA_KILL  EQU X'FF'       ........ KILLED BY THE OPERATOR
STC_LENG_TRNS DS  H           ..... LENGTH OF TRANSACTION_DATA
STC_LENG_ANSW DS  H           ..... LENGTH OF ANSWER
STC_ADDR_TRNS DS  F           ..... ADDRESS OF TRANSACTION_DATA
STC_ADDR_ANSW DS  F           ..... ADDRESS OF ANSWER
STC_API       EQU *           .. INFORMATION OF THE API TO SERVE
STC_API_ID    DS  CL4         ..... ID-NUMBER OF THE API
STC_API_NAME  DS  CL8         ..... NAME OF THE API
STC_API_TIME  DS  F           ..... TIME-OUT VALUE FOR THE API
STC_ECB       EQU *           .. ECB'S TO WAIT ON
STC_ECB_APPC  DS  F           ..... WAITING ON THE CMS_APPC_SERVER
STC_ECB_API_  DS  F           ..... WAITING ON THE API
STC_ECB_TIME  DS  F           ..... TIMEOUT OF THE API
STC_CMS       EQU *           .. VALUES FROM THE CMS_APPC_SERVER
STC_CMS_ASCB  DS  AL4         ..... ASCB ADDRESS OF THIS TASK
STC_CMS_ASID  DS  AL4         ..... ADDRESS SPACE NUMBER OF THIS TASK
STC_CMS_ECBA  DS  AL4         ..... ECB ADDRESS TO POST ON RETURN
STC_CMS_PARM  DS  AL4         ..... PARMLIST ADDRESS IN CMS_APPC_SERVER
STC_CMS_TIME  DS  CL16        ..... TIMESTAMP
STC_STSK      EQU *           .. INFORMATION ABOUT THE SUBTASK
STC_STSK_TCB  DS  F           ..... ADDRESS OF SUBTASK TCB
```

```
STC_STSK_TRID DS  F          ..... TIMER REQUEST IDENTIFIER
STC_STSK_PARM DS  5F         ..... USER PARAMETER LIST TO ATTACH
STC_STSK_ATCH DS  ØF         ..... CONTROL PARAMETER LIST TO ATTACH
    ATTACH EP=XXXXXXXX,SF=L  ....... GET THE SPACE
STC_STSK_TSET DS  ØF         ..... PARAMETER LIST TO STIMERM_SET
    STIMERM SET,MF=L         ....... GET THE SPACE
STC_STSK_TCAN DS  ØF         ..... PARAMETER LIST TO STIMERM_CANCEL
    STIMERM CANCEL,MF=L      ....... GET THE SPACE
STC_DATA      EQU *          .. APPLICATION_DATA
STC_TRNS_DATA DS  CL4ØØØ     .... TRANSACTION_DATA
STC_ANSW_DATA DS  CL4ØØØ     .... ANSWER
STC_COMMEND   EQU *
         TITLE 'CMSIFPGM : DSECTS'
         CVT   DSECT=YES,LIST=NO  MAP CVT
         IHAASVT DSECT=YES        MAP ASVT
         IHAASCB DSECT=YES        MAP ASCB
         IHAASXB DSECT=YES        MAP ASXB
         IKJTCB DSECT=YES         MAP TCB
         IHACDE                   MAP CDE
         IHAPSA DSECT=YES         MAP PSA
         IHASDWA DSECT=YES        MAP SDWA
         END
```

CMSIFSTC SOURCE

```
*************************************************************************
*  MODUL NAVN   :  CMSIFSTC
*  FUNKTION     :  THIS PROGRAM IS A STARTED TASK
*                  ITS MAIN FUNCTIONS ARE
*                  .. TO RECEIVE DATA FROM THE CMS_APPC_SERVER
*                  .. TO PASS DATA THROUGH TO THE API
*                  .. AND TO SEND THE ANSWER ALL THE WAY BACK
*  ECB'S        :  THIS PROGRAM HAS ECBS TO WAIT ON
*                  . THE CONSOLE
*                  . ONE OR MORE CMS_APPC-SERVER(S)
*                  . ONE OR MORE API(S)
*                  . A TIME-OUT FOR EACH API
*  EXEC-PARM    :  PARM='NNN,AAAAAAAA,TTT,AAAAAAAA,TTT'
*                  . "NNN" IS A 3-CHARACTERS NUMBER, WHICH SPECIFIES HOW
*                      MANY SETS OF "AAAAAAAA,TTT" FOLLOW, WHICH ALSO
*                      IS THE NUMBER OF APIS THIS TASK WILL SERVE
*                  . "AAAAAAAA" IS AN 8-CHARACTER API, WHICH CAN BE
*                      ATTACHED FROM THIS TASK
*                  . "TTT" IS A 3-CHARACTER TIME-OUT VALUE IN SECONDS,
*                      WHICH IS USED FOR TIME-OUT IN THE PRECEDING API
*  REQUIREMENT  :  LINK WITH AC(1)
*                  LINK WITH AMODE=31 AND RMODE=24
*                  EXECUTE FROM AUTHORIZED LIBRARY
*************************************************************************
         TITLE 'CMSIFSTC : REGISTER USING'
```

```
R4       EQU   4           WORK + BASE FOR IEZCOM
R5       EQU   5           WORK + BASE FOR IEZCIB
R6       EQU   6           WORK + BASE FOR ECBLIST
R7       EQU   7           WORK + BASE FOR COMMUNICATION-AREA
R8       EQU   8           WORK + USED IN SECONDARY ADDRESS SPACE
R9       EQU   9           WORK + USED IN SECONDARY ADDRESS SPACE
BALREG   EQU   1Ø          BAL REGISTER
R11      EQU   11          BASE REGISTER 1
R12      EQU   12          BASE REGISTER 2
R13      EQU   13          ADDRESS OF SAVE AREA
R14      EQU   14          RETURN REGISTER
R15      EQU   15          ENTRY ADDRESS + RETURN CODE
         TITLE 'CMSIFSTC : INITIAL SET UP'
CMSIFSTC CSECT
         USING CMSIFSTC,R15       ESTABLISH TEMPORARY MODULE
         B     ENTRYLNK           BRANCH AROUND EYECATCHERS
*                                 DO NOT REMOVE/CHANGE NEXT STATEMENTS
*                                 .. USED FOR CROSS MEMORY SERVICES
         DC    CL8'CMS__STC'      .... IDENTIFICATION
COMM_ADR DC    AL4(Ø)             .... ADDRESS OF COMMAREA_ADDRESS
         DC    CL8'CMSIFSTC'      EYECATCHERS FOR START OF MODULE
         DC    CL8'&SYSDATE'      .. AND DATE OF ASSEMBLY
         DC    CL8'&SYSTIME'      .. AND TIME OF ASSEMBLY
ENTRYLNK DS    ØH
         STM   R14,R12,12(R13)    SAVE CALLER'S REGISTERS
         LR    R11,R15            SET MODULE BASE REGISTER 1
         LR    R12,R15            ..
         LA    R12,4Ø95(R12)      ..
         LA    R12,1(R12)         .. AND BASE REGISTER 2
         DROP  R15                DROP TEMPORARY MODULE ADDRESSABILITY
         USING CMSIFSTC,R11,R12   ESTABLISH MODULE ADDRESSABILITY
         LA    R15,SAVEAREA       GET ADDRESS OF MY SAVE AREA
         ST    R15,8(R13)         SAVE FORWARD POINTER
         ST    R13,SAVEAREA+4     SAVE BACKWARD POINTER
         LA    R13,SAVEAREA       SET UP R13 TO POINT TO SAVE AREA IN
*                                 .. CASE THIS MODULE MAKES A CALL
         ST    R1,R1_SAVE         STORE REG1 AT ENTRY
         ESTAE ABNDEXIT           ESTABLISH ABEND HANDLING
         BAL   BALREG,INIT_CAS    INITIATE COMMUNICATION AREAS
         BAL   BALREG,INIT_CON    INITIATE THE CONSOLE INTERFACE
         MODESET KEY=ZERO,MODE=SUP    SET KEY Ø AND SUPERVISOR STATE
         LA    R1,Ø
         SYSEVENT TRANSWAP            MAKE ADDRESS SPACE NON-SWAPPABLE
         MODESET KEY=NZERO,MODE=PROB  RESET TO PROBLEM KEY AND STATE
         LA    R2,PARM_COM        STORE ADDRESS OF COMMAREA_ADDRESS
         ST    R2,COMM_ADR        .. IN THE START OF THE PROGRAM
         L     R2,PSAAOLD-PSA(Ø)  R2 -> ASCB ADDRESS OF OWN TASK
         USING ASCB,R2            SET UP ADDRESSING FOR ASCB
         LH    R3,ASCBASID        R3 = ADDRESS SPACE NUMBER OWN_TASK
         DROP  R2                 DROP ASCB ADDRESSABILITY
         ST    R3,STC_ASID        STORE ADDRESS SPACE NUMBER OWN_TASK
```

```
         MVC   WTOLABL+4(10),WTO_0
         MVC   WTOLABL+14(50),WTO001
         WTO   MF=(E,WTOLABL)     INITIALIZATION IS OK
         TITLE 'CMSIFSTC : WAIT ON WORK_TO_DO'
WORKTODO DS    0H
         L     R4,CON_COMM        LOAD ADDR OF CONSOLE COMM AREA
         USING IEZCOM,R4          SET UP ADDRESSING FOR IEZCOM
         L     R2,COMECBPT        R2 -> ECB FOR STOP/MODIFY COMMAND
         TM    0(R2),X'40'        STOP/MODIFY ECB POSTED ?
         BO    CONSOLE            .. YES
         DROP  R4
         L     R7,PARM_COM        LOAD ADDRESS OF 1ST COMM AREA
         USING STC_COMMAREA,R7    SETUP ADDRESSING FOR COMM AREA
WORK_01  DS    0H
         LTR   R7,R7              MORE COMMUNICATION AREAS ?
         BZ    WORK_02            .. NO
         TM    STC_ECB_APPC,X'40' ECB FOR CMS_APPC_SERVER POSTED ?
         BO    APPC_IF            .. YES
         TM    STC_ECB_TIME,X'40' ECB FOR TIME-OUT POSTED ?
         BO    TIMEOUT            .. YES
         TM    STC_ECB_API_,X'40' ECB FOR API POSTED ?
         BO    API__IF            .. YES
         L     R7,STC_NEXT_ADR    LOAD ADDRESS OF NEXT COMM AREA
         B     WORK_01
         DROP  R7
WORK_02  DS    0H
         TM    SHUT_ECB,X'40'     SHUT-DOWN ECB POSTED ?
         BO    STOP_02            .. YES
         TM    SWITCH,SHUTDOWN    IS SHUT-DOWN REQUESTED ?
         BO    STOP_09            .. YES
WAIT     DS    0H
         L     R6,PARM_ECB        LOAD ADDRESS OF ECBLIST
         USING ECBLIST,R6         SET UP ADDRESSING FOR ECBLIST
         WAIT  ECBLIST=ECBLIST    WAIT ON ECBLIST
         DROP  R6
         B     WORKTODO           CHECK WHO HAS POSTED US
         TITLE 'CMSIFSTC : REQUEST FROM THE CMS_APPC_SERVER'
APPC_IF  DS    0H
         LR    R8,R7
         AH    R8,=H'4096'
         USING STC_COMMAREA,R7,R8  SET UP ADDRESSING FOR COMM AREA
         XC    STC_ECB_APPC,STC_ECB_APPC  CLEAR ECB
         L     R2,APPC_REQ        NUMBER OF REQUEST
         LA    R2,1(R2)           .. RECEIVED FROM THE CMS_APPC_SERVER
         ST    R2,APPC_REQ        .. IS COUNTED
         LA    R2,STC_API_NAME    R2 -> NAME OF THE API
         ST    R7,API_COMM        SAVE ADDRESS OF COMMUNICATION AREA
         ATTACH EPLOC=(R2),       SEND REQUEST TO THE API             *
               ECB=STC_ECB_API_,                                      *
               ESTAI=(ABNDSTSK,API_COMM),                             *
               LPMOD=1,                                               *
```

```
              PARAM=(STC_RETNCODE,STC_LENG_TRNS,STC_TRNS_DATA,        *
              STC_LENG_ANSW,STC_ANSW_DATA),VL=1,                      *
              MF=(E,STC_STSK_PARM),                                   *
              SF=(E,STC_STSK_ATCH)
         ST   R1,STC_STSK_TCB     SAVE ADDRESS OF TCB FOR SUBTASK
         LTR  R15,R15             WAS ATTACH SUCCESSFUL?
         BNZ  FAIL_203            .. NO
         STIMERM SET,             SET A TIMER REQUEST             *
              BINTVL=STC_API_TIME,                                    *
              EXIT=TIMEEXIT,                                          *
              ID=STC_STSK_TRID,                                       *
              PARM=API_COMM,                                          *
              ERRET=FAIL_204,                                         *
              MF=(E,STC_STSK_TSET)
*                                     MARK COMMUNICATION AREA AS
         MVI  STC_STATUS,STC_STA_API_  .. IN_USE BY THE API
         B    WORKTODO
         DROP R7
         DROP R8
         TITLE 'CMSIFSTC : EXECUTE IF THE API FAILS'
         USING STC_COMMAREA,R7    SET UP ADDRESSING FOR COMM AREA
FAIL_203 DS   0H
         MVC  STC_RETNCODE,=C'203'  SET RETURN_CODE 203
         MVC  WTOLABL+4(20),WTO_2
         MVC  WTOLABL+8(3),=C'203'
         MVC  WTOLABL+24(40),WTO203
         MVC  WTOLABL+35(8),STC_API_NAME
         ST   R15,R15_SAVE
         UNPK WORK9,R15_SAVE(5)
         TR   WORK9(8),HEXTAB
         MVC  WTOLABL+56(8),WORK9
         WTO  MF=(E,WTOLABL)
         B    API__01
FAIL_204 DS   0H
         DETACH STC_STSK_TCB              DETACH THE SUBTASK
         XC   STC_LENG_ANSW,STC_LENG_ANSW  ANSWER IS NOT USABLE
         MVC  STC_RETNCODE,=C'204'  SET RETURN_CODE 204
         MVC  WTOLABL+4(20),WTO_2
         MVC  WTOLABL+8(3),=C'204'
         MVC  WTOLABL+24(40),WTO204
         ST   R15,R15_SAVE
         UNPK WORK9,R15_SAVE(5)
         TR   WORK9(8),HEXTAB
         MVC  WTOLABL+43(8),WORK9
         WTO  MF=(E,WTOLABL)
         B    API__01
TIMEOUT  DS   0H
         DETACH STC_STSK_TCB              DETACH THE SUBTASK
         XC   STC_LENG_ANSW,STC_LENG_ANSW  ANSWER IS NOT USABLE
         MVC  STC_RETNCODE,=C'200'  SET RETURN_CODE 200
         MVC  WTOLABL+4(20),WTO_2
```

```
         MVC   WTOLABL+8(3),=C'200'
         MVC   WTOLABL+24(40),WTO200
         MVC   WTOLABL+29(8),STC_API_NAME
         WTO   MF=(E,WTOLABL)
         B     API__01
         DROP  R7
         TITLE 'CMSIFSTC : REQUEST FROM THE API'
API__IF  DS    0H
         USING STC_COMMAREA,R7    SET UP ADDRESSING FOR COMM AREA
         DETACH STC_STSK_TCB      DETACH THE SUBTASK
         STIMERM CANCEL,          CANCEL THE TIMER REQUEST            *
               ID=STC_STSK_TRID,                                      *
               MF=(E,STC_STSK_TCAN)
API__01  DS    0H
         XC    STC_ECB_API_,STC_ECB_API_  CLEAR ECB
         XC    STC_ECB_TIME,STC_ECB_TIME  CLEAR ECB
         L     R2,APPC_ANS        NUMBER OF ANSWER
         LA    R2,1(R2)           .. SEND BACK TO THE CMS_APPC_SERVER
         ST    R2,APPC_ANS        .. IS COUNTED
         CLI   STC_STATUS,STC_STA_KILL  API KILLED BY THE OPERATOR ?
         BNE   API__11                  .. NO
         MVC   STC_RETNCODE,=C'202'     .. YES, SET RETURN_CODE 202
         MVC   WTOLABL+4(20),WTO_2
         MVC   WTOLABL+8(3),=C'202'
         MVC   WTOLABL+24(40),WTO202
         MVC   WTOLABL+29(8),STC_API_NAME
         WTO   MF=(E,WTOLABL)
         XC    STC_LENG_ANSW,STC_LENG_ANSW  ANSWER IS NOT USABLE
API__11  DS    0H
         L     R4,CVTPTR          R4 -> CVT
         USING CVT,R4             SET UP ADDRESSING FOR CVT
         L     R4,CVTASVT         R4 -> ASVT
         USING ASVT,R4            SET UP ADDRESSING FOR ASVT
         LA    R2,ASVTFRST        R2 -> 1ST ENTRY IN ASVT
         L     R3,ASVTMAXU        R3 = MAX NUMBER OF ADDRESS SPACES
API__12  DS    0H
         TM    0(R2),ASVTAVAI     IS ADDRESS SPACE NUMBER UNUSED ?
         BNO   API__13            .. NO
         C     R2,=A(X'80000000') IS THIS LAST ASVT ENTRY ?
         BE    API__15            .. YES, ASID NOT FOUND
         B     API__14            .. NO , CHECK NEXT ASCB
API__13  DS    0H
         L     R4,0(R2)           R4 -> ASCB
         USING ASCB,R4            SET UP ADDRESSING FOR ASCB
         C     R4,STC_CMS_ASCB          IS IT OUR ADDRESS SPACE ?
         BNE   API__14                  .. NO
         CLC   ASCBASID,STC_CMS_ASID+2  SAME ADDRESS SPACE NUMBER ?
         BNE   API__15                  .. NO, CMS_APPC_SERVER HAS GONE
         TM    ASCBRCTF,ASCBOUT         ADDRESS SPACE SWAPPED OUT ?
         BO    API__15                  .. YES, NOT USABLE
         B     API__21                  FOUND OUR ADDRESS SPACE USABLE
```

```
API__14  DS    ØH
         LA    R2,L'ASVTENTY(R2)  R2 -> NEXT ENTRY IN ASVT
         BCT   R3,API__12         CHECK NEXT ASVT ENTRY
*                                 NO MORE, ADDRESS SPACE NOT FOUND
API__15  DS    ØH
         DROP  R4
         MVC   WTOLABL+4(2Ø),WTO_2
         MVC   WTOLABL+8(3),=C'26Ø'
         MVC   WTOLABL+24(4Ø),WTO26Ø
         WTO   MF=(E,WTOLABL)
         B     API__9Ø
API__21  DS    ØH                 RETURN THE ANSWER FROM THE API
*                                 .. TO THE CMS_APPC_SERVER
         MODESET MODE=SUP         SET SUPERVISOR STATE
         LA    RØ,1               GET AX
         AXSET AX=(RØ)            .. OF 1
         ST    RØ,STC_AX          SAVE THE OLD AX
         L     R2,STC_CMS_ASID    ADDR SPACE NUMBER OF CMS_APPC_SERVER
         SSAR  R2                 SET AS SECONDARY ADDRESS SPACE
         LAM   R8,R9,ALET1        LOAD ALET1 INTO AR 8 TO 9
         SAC   512                SET AR MODE
         L     R8,STC_CMS_PARM    LOAD ADDR COMMAREA IN CMS_APPC_SERVER
         USING CMS_COMMAREA,R8    SET UP ADDRESSING FOR CMS_COMMAREA
         NI    SWITCH,CORR_CMS            CLEAR BIT, MEANS CORRECT TASK
         CLC   STC_CMS_TIME,CMS_TIMESTMP  CAN WE IDENTIFY THIS SERVER ?
         BE    API__22                    .. YES, EVERYTHING OK
         OI    SWITCH,WRNG_CMS            .. NO, NOT OUR TASK
         B     API__23
API__22  DS    ØH                 MOVE TO SECONDARY ADDRESS SPACE
         MVC   CMS_RETNCODE,STC_RETNCODE    .. RETURN_CODE
         CLC   STC_LENG_ANSW,=H'Ø'          ..
         BE    API__23                      .. AND IF ANSWER RETURNED
         MVC   CMS_LENG_ANSW,STC_LENG_ANSW  .... LENGTH OF DATA
         L     R8,CMS_ADDR_ANSW             ....
         L     R2,STC_ADDR_ANSW             ....
         LH    R9,STC_LENG_ANSW             ....
         LH    R3,STC_LENG_ANSW             ....
         MVCL  R8,R2                        .... TRANSACTION_DATA
         DROP  R8
API__23  DS    ØH
         L     R2,STC_ASID        RESTORE
         SSAR  R2                 .. ORIGINAL SASN
         SAC   Ø                  RESTORE ADDRESSING MODE
         L     RØ,STC_AX          RESTORE
         AXSET AX=(RØ)            .. OLD AX
         TM    SWITCH,WRNG_CMS    WAS IT CORRECT INTERFACE PROGRAM ?
         BO    API__24            .. NO, BYPASS POST
         L     R2,STC_CMS_ECBA    R2 = ECB  ADDRESS IN CMS_APPC_SERVER
         L     R3,STC_CMS_ASCB    R3 = ASCB ADDRESS IN CMS_APPC_SERVER
         POST  (R2),ASCB=(R3),    POST THE ECB                          *
               LINKAGE=SYSTEM,    .. IN THE SECONDARY ADDRESS SPACE     *
```

```
              ERRET=API__25,     IN CASE THE POST IS FAILING          *
              ECBKEY=0
API__24  DS   0H
         MODESET MODE=PROB       RESET TO PROBLEM STATE
         TM   SWITCH,WRNG_CMS    WAS IT CORRECT INTERFACE PROGRAM ?
         BZ   API__90            .. YES
         MVC  WTOLABL+4(20),WTO_2
         MVC  WTOLABL+8(3),=C'261'
         MVC  WTOLABL+24(40),WTO261
         WTO  MF=(E,WTOLABL)
         B    API__90
API__25  DS   0H
         MVC  WTOLABL+4(20),WTO_2
         MVC  WTOLABL+8(3),=C'262'
         MVC  WTOLABL+24(40),WTO262
         ST   R15,R15_SAVE
         UNPK WORK9,R15_SAVE(5)
         TR   WORK9(8),HEXTAB
         MVC  WTOLABL+54(8),WORK9
         WTO  MF=(E,WTOLABL)
         B    API__90
API__90  DS   0H
         XC   STC_RETNCODE,STC_RETNCODE
         XC   STC_LENG_ANSW,STC_LENG_ANSW
*                                      MARK COMMUNICATION AREA AS
         MVI  STC_STATUS,STC_STA_FREE  .. FREE = READY_TO_RECEIVE
         B    WORKTODO
         DROP R7
         TITLE 'CMSIFSTC : REQUEST FROM THE CONSOLE'
CONSOLE  DS   0H
         USING IEZCOM,R4         SET UP ADDRESSING FOR IEZCOM
         L    R5,COMCIBPT        R5 -> COMMAND INPUT BUFFER
         USING IEZCIB,R5         SET UP ADDRESSING FOR IEZCIB
         CLI  CIBVERB,CIBMODFY   MODIFY COMMAND ?
         BE   MODIFY             .. YES
         CLI  CIBVERB,CIBSTOP    STOP COMMAND ?
         BE   STOP               .. YES
         B    MODIFY_9           IGNORE UNKNOWN COMMAND
MODIFY   DS   0H
         CLC  =C'STAT',CIBDATA   IS STATUS WANTED ?
         BE   MODIFY_1           .. YES
         CLC  =C'VIEW',CIBDATA   IS API_VIEW WANTED ?
         BE   MODIFY_2           .. YES
         CLC  =C'KILL',CIBDATA   IS KILLING WANTED ?
         BE   MODIFY_3           .. YES
         MVC  WTOLABL+4(10),WTO_0
         MVC  WTOLABL+14(50),WTO100
         WTO  MF=(E,WTOLABL)     GIVE INFO ABOUT COMMANDS
         B    MODIFY_9
MODIFY_1 DS   0H
         MVC  WTOLABL+4(10),WTO_0
```

```
         MVC   WTOLABL+14(50),WTO101
         L     R2,APPC_REQ         NO OF REQUESTS - NO OF ANSWERS
         S     R2,APPC_ANS         GIVES NO OF OUTSTANDING REQUESTS
         CVD   R2,DOB
         MVC   WTOLABL+14(4),=X'40202120'
         ED    WTOLABL+14(4),DOB+6
         WTO   MF=(E,WTOLABL)      TELL ABOUT OUTSTANDING REQUESTS
         B     MODIFY_9
MODIFY_2 DS    0H
         L     R7,PARM_COM         LOAD ADDRESS OF COMMUNICATION AREAS
         USING STC_COMMAREA,R7     SET UP ADDRESSING FOR COMM AREA
MODI_2_1 DS    0H
         CLI   STC_STATUS,STC_STA_API_  WAIT ON THE API ?
         BNE   MODI_2_2
         MVC   WTOLABL+4(10),WTO_0
         MVC   WTOLABL+14(50),WTO102_1
         MVC   WTOLABL+18(4),STC_API_ID
         MVC   WTOLABL+23(8),STC_API_NAME
         WTO   MF=(E,WTOLABL)           .. YES, API IS OUTSTANDING
         B     MODI_2_3
MODI_2_2 DS    0H
         MVC   WTOLABL+4(10),WTO_0
         MVC   WTOLABL+14(50),WTO102_2
         MVC   WTOLABL+18(4),STC_API_ID
         MVC   WTOLABL+23(8),STC_API_NAME
         WTO   MF=(E,WTOLABL)           .. NO,  API IS NOT OUTSTANDING
MODI_2_3 DS    0H
         L     R7,STC_NEXT_ADR
         LTR   R7,R7               IF MORE COMMAREAS
         BNZ   MODI_2_1            .. CHECK THEM
         DROP  R7
         B     MODIFY_9
MODIFY_3 DS    0H
         L     R7,PARM_COM         LOAD ADDRESS OF COMMUNICATION AREAS
         USING STC_COMMAREA,R7     SET UP ADDRESSING FOR COMM AREA
MODI_3_1 DS    0H
         CLC   STC_API_ID,CIBDATA+5     OUR API ?
         BNE   MODI_3_2                 .. NO
         CLI   STC_STATUS,STC_STA_API_  WAIT ON THE API ?
         BNE   MODI_3_2                 .. NO
*                                        MARK COMMUNICATION AREA AS
         MVI   STC_STATUS,STC_STA_KILL  .. KILLED BY THE OPERATOR
         LA    R2,STC_ECB_API_          GET ADDRESS OF API_ECB
         POST  (R2)                     POST IT
         MVC   WTOLABL+4(10),WTO_0
         MVC   WTOLABL+14(50),WTO103_1
         MVC   WTOLABL+18(4),STC_API_ID
         MVC   WTOLABL+23(8),STC_API_NAME
         WTO   MF=(E,WTOLABL)           TELL ABOUT KILLED API
         B     MODIFY_9
MODI_3_2 DS    0H
```

```
         L     R7,STC_NEXT_ADR
         LTR   R7,R7               IF MORE COMMAREAS
         BNZ   MODI_3_1            .. CHECK THEM
         MVC   WTOLABL+4(10),WTO_0
         MVC   WTOLABL+14(50),WTO103_2
         MVC   WTOLABL+18(4),CIBDATA+5
         WTO   MF=(E,WTOLABL)      TELL THAT API NOT FOUND/OUTSTANDING
         DROP  R7
         B     MODIFY_9
MODIFY_9 DS    0H
         QEDIT ORIGIN=COMCIBPT,BLOCK=(R5)       FREE PROCESSED CIB
         B     WORKTODO
STOP     DS    0H                  SET ADDR OF COMMAREA_ADDR TO STOP
         MVC   COMM_ADR,=F'-1'     .. REQUEST FROM THE CMS_APPC_SERVER
         MVC   WTOLABL+4(10),WTO_0
         MVC   WTOLABL+14(50),WTO003
         WTO   MF=(E,WTOLABL)      TELL THAT STOP COMMAND IS RECEIVED
         QEDIT ORIGIN=COMCIBPT,BLOCK=(R5)       FREE PROCESSED CIB
         L     R2,APPC_REQ         NO OF REQUESTS - NO OF ANSWERS
         S     R2,APPC_ANS         GIVES NO OF OUTSTANDING REQUESTS
         LTR   R2,R2
         BZ    STOP_09             NO OUTSTANDING REQUEST
         TM    SWITCH,STOP_CMD     DO WE HAVE AN OUTSTANDING STOP?
         BO    STOP_01             .. YES, KILL ALL SUBTASKS
         OI    SWITCH,STOP_CMD     .. NO,  SAY WE NOW HAVE
         STIMERM SET,              SET A TIMER REQUEST                 *
               BINTVL=MAX_TIME,                                        *
               EXIT=TIMESHUT,                                          *
               ID=SHUT_ID
         MVC   WTOLABL+4(10),WTO_0
         MVC   WTOLABL+14(50),WTO004
         WTO   MF=(E,WTOLABL)      SAY WE WILL WAIT ON OUTSTANDING
         B     WORKTODO            CHECK IF OUTSTANDING HAS COMPLETED
STOP_01  DS    0H
         STIMERM CANCEL,           CANCEL THE TIMER REQUEST            *
               ID=SHUT_ID
STOP_02  DS    0H
         L     R7,PARM_COM         LOAD ADDRESS OF COMMUNICATION AREAS
         USING STC_COMMAREA,R7     SET UP ADDRESSING FOR COMM AREA
STOP_2_1 DS    0H
         CLI   STC_STATUS,STC_STA_API_  WAIT ON THE API ?
         BNE   STOP_2_2                 .. NO
*                                       MARK COMMUNICATION AREA AS
         MVI   STC_STATUS,STC_STA_KILL  .. KILLED BY THE OPERATOR
         LA    R2,STC_ECB_API_          GET ADDRESS OF API_ECB
         POST  (R2)                     POST IT
         MVC   WTOLABL+4(10),WTO_0
         MVC   WTOLABL+14(50),WTO005
         MVC   WTOLABL+18(4),STC_API_ID
         MVC   WTOLABL+23(8),STC_API_NAME
         WTO   MF=(E,WTOLABL)      TELL ABOUT API FORCED DOWN
```

```
STOP_2_2 DS    ØH
         L     R7,STC_NEXT_ADR
         LTR   R7,R7               IF MORE COMMAREAS
         BNZ   STOP_2_1            .. CHECK THEM
         DROP  R7
         XC    SHUT_ECB,SHUT_ECB   CLEAR ECB
         OI    SWITCH,SHUTDOWN     DO A SHUT-DOWN
         B     WORKTODO            CHECK IF OUTSTANDING HAS COMPLETED
STOP_Ø9  DS    ØH
         MVC   WTOLABL+4(1Ø),WTO_Ø
         MVC   WTOLABL+14(5Ø),WTOØØ2
         WTO   MF=(E,WTOLABL)      TELL THAT APPLICATION HAS ENDED
         LA    R15,Ø               SET RETURN CODE = Ø
         ST    R15,RET_CODE        STORE IT FOR LATER USE
         B     END_PROG
         DROP  R4
         DROP  R5
         TITLE 'CMSIFSTC : EXIT THE PROGRAM'
END_PROG DS    ØH
         L     R13,SAVEAREA+4      RETRIEVE BACKWARD POINTER
         L     R15,RET_CODE        GET DESIRED RETURN CODE
         RETURN (14,12),RC=(15)    RETURN CONTROL TO SYSTEM
         TITLE 'CMSIFSTC : ROUTINE TO INITIATE THE COMMUNICATION AREAS'
INIT_CAS DS    ØH
         ST    BALREG,BALRSAVE
         L     R1,R1_SAVE          LOAD REG1 AT ENTRY
         L     R1,Ø(R1)            ADDRESS OF PARM= STRING TO R1
         LA    R2,2(R1)            R2 -> PARM_STRING
         LH    R3,Ø(R1)            R3 = LENGTH OF PARM_STRING
         CH    R3,=H'16'
         BL    INIT_SØ1            ERROR, "NNN/AAAAAAAA/TTT" MISSING
         MVC   WORK9(3),=C'ØØØ'
         MVN   WORK9(3),Ø(R2)
         CLC   WORK9(3),Ø(R2)
         BNE   INIT_SØ2            ERROR, "NNN" NOT NUMERIC
         PACK  DOB,Ø(3,R2)
         CVB   R4,DOB
         STH   R4,API_NUMB         SAVE NUMBER OF APIS TO SERVE
         MH    R4,=H'13'
         AH    R4,=H'3'
         CR    R3,R4
         BNE   INIT_SØ3            ERROR, "AAAAAAAA/TTT" NOT MATCH "NNN"
         LH    R4,API_NUMB
         LA    R5,13(R2)
INIT_Ø1  DS    ØH
         MVC   WORK9(3),=C'ØØØ'
         MVN   WORK9(3),Ø(R5)
         CLC   WORK9(3),Ø(R5)
         BNE   INIT_SØ4            ERROR, "TTT" NOT NUMERIC
         LA    R5,13(R5)
         BCT   R4,INIT_Ø1
```

```
         LA    R4,12               .. LENGTH OF 3 ECB-ADDRESSES
         MH    R4,API_NUMB         ..... FOR EACH API TO SERVE
         LA    R4,8(R4)            .. PLUS ECB ADDRESS CONSOLE/SHUT-DOWN
*                                  LENGTH OF ECBLIST IN R4
         LH    R5,CAS_LENG         .. LENGTH OF EACH COMMUNICATION AREA
         MH    R5,API_NUMB         LENGTH OF COMMUNICATION AREAS IN R5
         AR    R5,R4               GETMAIN STORAGE TO THE PARMLIST
         STORAGE OBTAIN,LOC=BELOW,                                      *
               LENGTH=(R5),ADDR=(R6)
         LTR   R15,R15
         BNZ   INIT_S11            ERROR, NO STORAGE OBTAINED
         ST    R6,PARM_ECB         STORE ADDRESS OF ECBLIST
         LR    R7,R6
         AR    R7,R4
         ST    R7,PARM_COM         STORE ADDRESS OF COMMUNICATION AREAS
         USING ECBLIST,R6          SET UP ADDRESSING FOR ECBLIST
         LA    R5,SHUT_ECB         ADDRESS OF ECB FOR SHUTDOWN
         ST    R5,ECB_SHUT         .. TO THE ECBLIST
         LA    R6,8(R6)            BYPASS ADDR OF CONSOLE/SHUT-DOWN ECB
         USING ECBLIST+8,R6        SET UP ADDRESSING FOR ECBLIST + 8
         USING STC_COMMAREA,R7     SET UP ADDRESSING FOR COMM AREA
         LA    R2,4(R2)            R2 -> 1ST API IN THE EXEC PARM
         LH    R3,API_NUMB         NUMBER OF APIS TO SERVE IN R3
         ZAP   COUNTER,=P'1'
         B     INIT_12             INITIALIZE ECBLIST AND COMMAREAS
INIT_11  DS    ØH
         LA    R6,12(R6)           ADDRESS NEXT SET OF ECBS
         LH    R5,CAS_LENG
         AR    R7,R5               ADDRESS NEXT COMMUNICATION AREA
         LA    R2,13(R2)           ADDRESS NEXT API/TIME-OUT PARM
INIT_12  DS    ØH
         LA    R5,STC_ECB_APPC     ADDRESS OF ECB FOR CMS_APPC_SERVER
         ST    R5,ECB_APPC         .. TO THE ECBLIST
         LA    R5,STC_ECB_API_     ADDRESS OF ECB FOR THE API
         ST    R5,ECB_API_         .. TO THE ECBLIST
         LA    R5,STC_ECB_TIME     ADDRESS OF TIMEOUT_ECB
         ST    R5,ECB_TIME         .. TO THE ECBLIST
*                                  BUILD THE COMMUNICATION AREA
         XC    STC_COMMAREA(STC_DATA-STC_COMMAREA),STC_COMMAREA
         MVC   STC_STSK_ATCH(ATCH_LEN),ATCH_LAB
         MVC   STC_STSK_TSET(TSET_LEN),TSET_LAB
         MVC   STC_STSK_TCAN(TCAN_LEN),TCAN_LAB
         LH    R5,OFS_TRNS         GET OFFSET TO TRANSACTION_DATA
         AR    R5,R7               .. ADD ADDRESS OF COMMUNICATION AREA
         ST    R5,STC_ADDR_TRNS    .. AND STORE ADDRESS OF TRANS_DATA
         LH    R5,OFS_ANSW         GET OFFSET TO ANSWER
         AR    R5,R7               .. ADD ADDRESS OF COMMUNICATION AREA
         ST    R5,STC_ADDR_ANSW    .. AND STORE ADDRESS OF ANSWER
         LH    R5,CAS_LENG         GET LENGTH OF EACH COMMAREA
         AR    R5,R7               .. ADD ADDRESS OF COMMUNICATION AREA
         ST    R5,STC_NEXT_ADR     .. AND STORE ADDRESS OF NEXT COMMAREA
```

```
         MVC   WORK9(6),=X'402120202020'
         ED    WORK9(6),COUNTER
         MVC   STC_API_ID,WORK9+2 STORE IDENTIFICATION OF THE API
         AP    COUNTER,=P'1'
         MVC   STC_API_NAME,0(R2) STORE NAME OF THE API
         PACK  DOB,9(3,R2)
         CVB   R5,DOB
         MH    R5,=H'100'
         ST    R5,STC_API_TIME    STORE TIME-OUT IN VALUE OF 0.01 SECS
         C     R5,MAX_TIME
         BNH   INIT_21
         ST    R5,MAX_TIME        GET HIGHEST TIME-OUT VALUE
INIT_21  DS    0H
         BCT   R3,INIT_11         IF MORE ENTRIES, INITIATE NEXT
*                                 IF LAST ENTRY, INITIATE LAST
         OC    ECB_TIME,=X'80000000'      .. MARK AS LAST ECB IN LIST
         XC    STC_NEXT_ADR,STC_NEXT_ADR  .. CLEAR POINTER NEXT
         DROP  R6
         DROP  R7
         L     BALREG,BALRSAVE
         BR    BALREG
INIT_S01 LA    R15,4              SET RETURN CODE = 4
         B     INIT_S09
INIT_S02 LA    R15,8              SET RETURN CODE = 8
         B     INIT_S09
INIT_S03 LA    R15,12             SET RETURN CODE = 12
         B     INIT_S09
INIT_S04 LA    R15,16             SET RETURN CODE = 16
INIT_S09 ST    R15,RET_CODE       STORE IT FOR LATER USE
         MVC   WTOLABL+4(20),WTO_2
         MVC   WTOLABL+8(3),=C'251'
         MVC   WTOLABL+24(40),WTO251
         WTO   MF=(E,WTOLABL)
         B     END_PROG
INIT_S11 ST    R15,RET_CODE       STORE R15 FROM STORAGE_OBTAIN MACRO
         MVC   WTOLABL+4(20),WTO_2
         MVC   WTOLABL+8(3),=C'252'
         MVC   WTOLABL+24(40),WTO252
         WTO   MF=(E,WTOLABL)
         B     END_PROG
         TITLE 'CMSIFSTC : ROUTINE TO INITIATE THE CONSOLE INTERFACE'
INIT_CON DS    0H
         ST    BALREG,BALRSAVE
         MODESET KEY=ZERO,MODE=SUP    SET KEY 0 AND SUPERVISOR STATE
         LA    R4,CON_COMM            WHERE TO PLACE THE ANSWER
*                                     GET ADDR OF COMMAND SCHEDULER
         EXTRACT (R4),FIELDS=COMM     ............ COMMUNICATION LIST
         L     R4,CON_COMM            LOAD ADDR OF CONSOLE COMM AREA
         USING IEZCOM,R4              SET UP ADDRESSING FOR IEZCOM
         MODESET KEY=NZERO,MODE=PROB  RESET TO PROBLEM KEY AND STATE
         L     R5,COMCIBPT            R5 -> COMMAND INPUT BUFFER
```

```
         USING IEZCIB,R5              SET UP ADDRESSING FOR IEZCIB
         CLI   CIBVERB,CIBSTART       CIB FOR START COMMAND ?
         BNE   INIT_51                .. NO
*                                     .. YES, FREE START CIB
         QEDIT ORIGIN=COMCIBPT,BLOCK=(R5)
INIT_51  DS    ØH
*                                     SET LIMIT = 1 ON QUEUED COMMANDS
         QEDIT ORIGIN=COMCIBPT,CIBCTR=1
         L     R6,PARM_ECB        LOAD ADDRESS OF ECBLIST
         USING ECBLIST,R6         SET UP ADDRESSING FOR ECBLIST
         MVC   ECB_CONS,COMECBPT  MOVE ADDRESS OF ECB TO ECB_LIST
         DROP  R4
         DROP  R5
         DROP  R6
         L     BALREG,BALRSAVE
         BR    BALREG
         TITLE 'CMSIFSTC : WORK AREAS'
PARMLIST DS    ØF            PARMLIST FOR THE STARTED TASK
PARM_ECB DS    F             .. ADDRESS OF ECBLIST
PARM_COM DS    F             .. ADDRESS OF COMMUNICATION AREAS
SAVEAREA DS    18F           SAVE_AREA IN MY_MODULE
SAVEABND DS    18F           SAVE_AREA IN ABEND_EXIT FOR MOTHER TCB
SAVESTSK DS    18F           SAVE_AREA IN ABEND_EXIT FOR SUBTASK TCB
SAVETIME DS    18F           SAVE_AREA IN TIMEOUT_EXIT FOR API'S
SAVESHUT DS    18F           SAVE_AREA IN TIMEOUT_EXIT AT SHUTDOWN
ALET1    DC    16F'1'        IN ACCESS REGISTER MODE,
*                            .. AN ACCESS REGISTER THAT CONTAINS AN
*                            .. ALET OF X'00000001' CAUSES
*                            .. DAT TRANSLATION USING THE SECONDARY STD
RET_CODE DS    F             RETURN CODE FROM THIS TASK
STC_ASID DS    F             ADDRESS SPACE NUMBER OF THIS TASK
STC_AX   DS    F             OLD AX
BALRSAVE DS    F             SAVE OF BALREG IN BALR ROUTINES
CON_COMM DS    F             ADDR CONSOLE COMMUNICATION_PARAMETER_LIST
API_COMM DS    F             ADDR COMMUNICATION AREA FOR API
ADR_SDWA DS    F             SDWA ADDRESS
R1_SAVE  DS    F             REG1 AT ENTRY
R15_SAVE DS    F             REG15 AT POST OR ATTACH FAILURE
APPC_REQ DC    F'Ø'          COUNTER, REQUEST FROM CMS_APPC_SERVER
APPC_ANS DC    F'Ø'          COUNTER, ANSWER BACK TO CMS_APPC_SERVER
MAX_TIME DC    F'Ø'          MAX TIME-OUT VALUE FOR APIS
SHUT_ECB DC    F'Ø'          ECB, WAITING AT OUTSTANDING AT SHUT-DOWN
SHUT_ID  DS    F             TIMER REQUEST IDENTIFIER AT SHUT-DOWN
API_NUMB DS    H             NUMBER OF APIS TO SERVE
CAS_LENG DC    AL2(STC_COMMEND-STC_COMMAREA)   LENGTH OF EACH COMMAREA
OFS_TRNS DC    AL2(STC_TRNS_DATA-STC_COMMAREA) OFFSET TO TRANS_DATA
OFS_ANSW DC    AL2(STC_ANSW_DATA-STC_COMMAREA) OFFSET TO ANSWER
SWITCH   DC    X'ØØ'         SWITCH
WRNG_CMS EQU   X'8Ø'          .. THE CMS_APPC_SERVER IS INCORRECT
CORR_CMS EQU   255-X'8Ø'      .... TO RESET PREVIOUS BIT
STOP_CMD EQU   X'Ø8'          .. STOP COMMAND IS OUTSTANDING
```

```
SHUTDOWN EQU   X'Ø1'          .. SHUT DOWN THE TASK
HEXTAB   EQU   *-C'Ø'        WORK FIELDS
         DC    C'Ø123456789ABCDEF'
WORK9    DS    CL9
COUNTER  DS    PL3
DOB      DS    D
ATCH_LAB DS    ØF            CONTROL PARMLIST FOR ATTACH
         ATTACH EP=XXXXXXXX,SF=L
ATCH_LEN EQU   *-ATCH_LAB
TSET_LAB DS    ØF            CONTROL PARMLIST FOR STIMERM_SET
         STIMERM SET,MF=L
TSET_LEN EQU   *-TSET_LAB
TCAN_LAB DS    ØF            CONTROL PARMLIST FOR STIMERM_CANCEL
         STIMERM CANCEL,MF=L
TCAN_LEN EQU   *-TCAN_LAB
WTOLABL  DS    ØF
         WTO   '                                                    *
                    ',ROUTCDE=2,DESC=(7),MF=L
WTO_Ø    DC    CL10'CMSIFSTC: '
WTOØØ1   DC    CL50'INITIALIZATION HAS COMPLETED                    '
WTOØØ2   DC    CL50'APPLICATION HAS ENDED                           '
WTOØØ3   DC    CL50'STOP COMMAND RECEIVED                           '
WTOØØ4   DC    CL50'WAIT ON OUTSTANDING REQUESTS                    '
WTOØØ5   DC    CL50'API XXXX/XXXXXXXX IS FORCED DOWN                '
WTO1ØØ   DC    CL50'COMMANDS ARE "STATUS", "VIEW" AND "KILL XXXX"   '
WTO1Ø1   DC    CL50'XXXX OUTSTANDING REQUEST                        '
WTO1Ø2_1 DC    CL50'API XXXX/XXXXXXXX IS OUTSTANDING                '
WTO1Ø2_2 DC    CL50'API XXXX/XXXXXXXX IS NOT OUTSTANDING            '
WTO1Ø3_1 DC    CL50'API XXXX/XXXXXXXX IS KILLED                     '
WTO1Ø3_2 DC    CL50'API XXXX IS NOT OUTSTANDING                     '
WTO_2    DC    CL20'CMS____E  CMSIFSTC: '
WTO2ØØ   DC    CL40'API "XXXXXXXX" TIMES OUT                '
WTO2Ø1   DC    CL40'API "XXXXXXXX" HAS ABENDED              '
WTO2Ø2   DC    CL40'API "XXXXXXXX" KILLED BY THE OPERATOR   '
WTO2Ø3   DC    CL40'ATTACH OF "XXXXXXXX" FAILED, RC=XXXXXXXX'
WTO2Ø4   DC    CL40'STIMERM FAILED, RC=XXXXXXXX             '
WTO25Ø   DC    CL40'CMS_API_SERVER HAS ABENDED              '
WTO251   DC    CL40'ERROR IN EXECUTE PARAMETER              '
WTO252   DC    CL40'ERROR IN STORAGE_OBTAIN                 '
WTO26Ø   DC    CL40'CMS_APPC_SERVER HAS GONE                '
WTO261   DC    CL40'CMS_APPC_SERVER HAS INCORRECT TIMESTAMP '
WTO262   DC    CL40'POST OF RETURN_ECB FAILED, RC=XXXXXXXX  '
         LTORG
         TITLE 'CMSIFSTC : ABEND EXIT FOR THIS TASK'
ABNDEXIT DS    ØH
         STM   R14,R12,12(R13)    SAVE CALLER'S REGISTERS
         DROP  R11,R12
         USING ABNDEXIT,R15
         ICM   R11,B'1111',=AL4(CMSIFSTC)
         ICM   R12,B'1111',=AL4(CMSIFSTC+4Ø96)
         DROP  R15
```

```
         USING CMSIFSTC,R11,R12   ESTABLISH MODULE ADDRESSABILITY
         LA    R15,SAVEABND       GET ADDRESS OF ABEND SAVE AREA
         ST    R15,8(R13)         SAVE FORWARD POINTER
         ST    R13,SAVEABND+4     SAVE BACKWARD POINTER
         LA    R13,SAVEABND       SET UP R13 TO POINT TO SAVE AREA IN
*                                 .. CASE THIS MODULE MAKES A CALL
         ST    R1,ADR_SDWA        SAVE SDWA ADDRESS
         MVC   WTOLABL+4(20),WTO_2
         MVC   WTOLABL+8(3),=C'250'
         MVC   WTOLABL+24(40),WTO250
         WTO   MF=(E,WTOLABL)
         L     R1,ADR_SDWA        LOAD SDWA ADDRESS
         USING SDWA,R1            ADDRESS SDWA USED BY SETRP MACRO
         L     R13,SAVEABND+4     RETRIEVE BACKWARD POINTER
*                                 CONTINUE ABEND
         SETRP RC=0,REGS=(14,12)
         LTORG
         TITLE 'CMSIFSTC : ABEND EXIT AT SUBTASK (API) EXECUTING'
ABNDSTSK DS    0H
         STM   R14,R12,12(R13)    SAVE CALLER'S REGISTERS
         DROP  R11,R12
         USING ABNDSTSK,R15
         ICM   R11,B'1111',=AL4(CMSIFSTC)
         ICM   R12,B'1111',=AL4(CMSIFSTC+4096)
         DROP  R15
         USING CMSIFSTC,R11,R12   ESTABLISH MODULE ADDRESSABILITY
         LA    R15,SAVESTSK       GET ADDRESS OF ABEND SAVE AREA
         ST    R15,8(R13)         SAVE FORWARD POINTER
         ST    R13,SAVESTSK+4     SAVE BACKWARD POINTER
         LA    R13,SAVESTSK       SET UP R13 TO POINT TO SAVE AREA IN
*                                 .. CASE THIS MODULE MAKES A CALL
         ST    R1,ADR_SDWA        SAVE SDWA ADDRESS
         USING SDWA,R1            ADDRESS SDWA
         L     R7,SDWAPARM        ADDRESS OF PARMLIST
         L     R7,0(R7)           ADDRESS OF COMMAREA
         USING STC_COMMAREA,R7    SETUP ADDRESSING FOR COMM AREA
         XC    STC_LENG_ANSW,STC_LENG_ANSW  ANSWER IS NOT USEABLE
         MVC   STC_RETNCODE,=C'201'  SET RETURN_CODE 201
         MVC   WTOLABL+4(20),WTO_2
         MVC   WTOLABL+8(3),=C'201'
         MVC   WTOLABL+24(40),WTO201
         MVC   WTOLABL+29(8),STC_API_NAME
         DROP  R7
         WTO   MF=(E,WTOLABL)
         L     R1,ADR_SDWA        LOAD SDWA ADDRESS
         USING SDWA,R1            ADDRESS SDWA USED BY SETRP MACRO
         L     R13,SAVESTSK+4     RETRIEVE BACKWARD POINTER
*                                 CONTINUE ABEND
         SETRP RC=0,REGS=(14,12)
         LTORG
         TITLE 'CMSIFSTC : TIMEOUT EXIT FOR APIS'
```

```
TIMEEXIT DS    ØH
         STM   R14,R12,12(R13)    SAVE CALLER'S REGISTERS
         DROP  R11,R12
         USING TIMEEXIT,R15
         ICM   R11,B'1111',=AL4(CMSIFSTC)
         ICM   R12,B'1111',=AL4(CMSIFSTC+4Ø96)
         DROP  R15
         USING CMSIFSTC,R11,R12   ESTABLISH MODULE ADDRESSABILITY
         LA    R15,SAVETIME       GET ADDRESS OF ABEND SAVE AREA
         ST    R15,8(R13)         SAVE FORWARD POINTER
         ST    R13,SAVETIME+4     SAVE BACKWARD POINTER
         LA    R13,SAVETIME       SET UP R13 TO POINT TO SAVE AREA IN
*                                 .. CASE THIS MODULE MAKES A CALL
         L     R7,4(R1)           LOAD ADDRESS AF COMMUNICATION AREA
         USING STC_COMMAREA,R7    SET UP ADDRESSING FOR COMM AREA
         LA    R2,STC_ECB_TIME    GET ADDRESS OF TIMEOUT ECB
         POST  (R2)               POST IT
         DROP  R7
         L     R13,SAVETIME+4     RETRIEVE BACKWARD POINTER
         LM    R14,R12,12(R13)    RESTORE CALLER'S REGISTERS
         BR    R14                RETURN TO THE SYSTEM
         LTORG
         TITLE 'CMSIFSTC : TIME-OUT EXIT AT SHUT-DOWN'
TIMESHUT DS    ØH
         STM   R14,R12,12(R13)    SAVE CALLER'S REGISTERS
         DROP  R11,R12
         USING TIMESHUT,R15
         ICM   R11,B'1111',=AL4(CMSIFSTC)
         ICM   R12,B'1111',=AL4(CMSIFSTC+4Ø96)
         DROP  R15
         USING CMSIFSTC,R11,R12   ESTABLISH MODULE ADDRESSABILITY
         LA    R15,SAVESHUT       GET ADDRESS OF ABEND SAVE AREA
         ST    R15,8(R13)         SAVE FORWARD POINTER
         ST    R13,SAVESHUT+4     SAVE BACKWARD POINTER
         LA    R13,SAVESHUT       SET UP R13 TO POINT TO SAVE AREA IN
*                                 .. CASE THIS MODULE MAKES A CALL
         POST  SHUT_ECB           POST SHUT-DOWN ECB
         L     R13,SAVESHUT+4     RETRIEVE BACKWARD POINTER
         LM    R14,R12,12(R13)    RESTORE CALLER'S REGISTERS
         BR    R14                RETURN TO THE SYSTEM
         LTORG
         TITLE 'CMSIFSTC : ECBLIST IN THE CMS_API_SERVER'
ECB_PARAMETER DSECT
ECBLIST       DS    ØF       LIST OF ECB ADDRESSES
ECB_CONS      DS    F        .. ADDR OF ECB FOR THE CONSOLE
ECB_SHUT      DS    F        .. ADDR OF ECB FOR SHUT-DOWN
*                            .. ADDR OF ECBS FOR INTERFACES AGAINST
ECB_APPC      DS    F        ..... THE CMS_APPC_SERVER
ECB_API_      DS    F        ..... THE API
ECB_TIME      DS    F        ..... TIME-OUT OF THE API
         TITLE 'CMSIFSTC : COMMUNICATION_AREA IN THE CMS_API_SERVER'
```

```
STC_PARAMETER DSECT
STC_COMMAREA  DS  ØF         COMMUNICATION AREA FOR EACH API TO SERVE
STC_NEXT      EQU *          .. POSITIONING THIS COMMUNICATION AREA
STC_NEXT_ADR  DS  F          ..... ADDRESS OF NEXT COMMUNICATION AREA
STC_INFO      EQU *          .. GENERAL INFORMATION
STC_RETNCODE  DS  CL3        ..... RETURN_CODE
STC_STATUS    DS  XL1        ..... STATUS OF THIS COMMUNICATION AREA
STC_STA_FREE  EQU X'ØØ'      ........ READY_TO_RECEIVE
STC_STA_APPC  EQU X'11'      ........ IN_USE BY THE CMS_APPC_SERVER
STC_STA_API_  EQU X'22'      ........ IN_USE BY THE API
STC_STA_KILL  EQU X'FF'      ........ KILLED BY THE OPERATOR
STC_LENG_TRNS DS  H          ..... LENGTH OF TRANSACTION_DATA
STC_LENG_ANSW DS  H          ..... LENGTH OF ANSWER
STC_ADDR_TRNS DS  F          ..... ADDRESS OF TRANSACTION_DATA
STC_ADDR_ANSW DS  F          ..... ADDRESS OF ANSWER
STC_API       EQU *          .. INFORMATION OF THE API TO SERVE
STC_API_ID    DS  CL4        ..... ID-NUMBER OF THE API
STC_API_NAME  DS  CL8        ..... NAME OF THE API
STC_API_TIME  DS  F          ..... TIME-OUT VALUE FOR THE API
STC_ECB       EQU *          .. ECBS TO WAIT ON
STC_ECB_APPC  DS  F          ..... WAITING ON THE CMS_APPC_SERVER
STC_ECB_API_  DS  F          ..... WAITING ON THE API
STC_ECB_TIME  DS  F          ..... TIME-OUT OF THE API
STC_CMS       EQU *          .. VALUES FROM THE CMS_APPC_SERVER
STC_CMS_ASCB  DS  AL4        ..... ASCB ADDRESS OF THIS TASK
STC_CMS_ASID  DS  AL4        ..... ADDRESS SPACE NUMBER OF THIS TASK
STC_CMS_ECBA  DS  AL4        ..... ECB ADDRESS TO POST ON RETURN
STC_CMS_PARM  DS  AL4        ..... PARMLIST ADDRESS IN CMS_APPC_SERVER
STC_CMS_TIME  DS  CL16       ..... TIMESTAMP
STC_STSK      EQU *          .. INFORMATION ABOUT THE SUBTASK
STC_STSK_TCB  DS  F          ..... ADDRESS OF SUBTASK TCB
STC_STSK_TRID DS  F          ..... TIMER REQUEST IDENTIFIER
STC_STSK_PARM DS  5F         ..... USER PARAMETER LIST TO ATTACH
STC_STSK_ATCH DS  ØF         ..... CONTROL PARAMETER LIST TO ATTACH
    ATTACH EP=XXXXXXXX,SF=L  ....... GET THE SPACE
STC_STSK_TSET DS  ØF         ..... PARAMETER LIST TO STIMERM_SET
    STIMERM SET,MF=L         ....... GET THE SPACE
STC_STSK_TCAN DS  ØF         ..... PARAMETER LIST TO STIMERM_CANCEL
    STIMERM CANCEL,MF=L      ....... GET THE SPACE
STC_DATA      EQU *          .. APPLICATION_DATA
STC_TRNS_DATA DS  CL4ØØØ     .... TRANSACTION_DATA
STC_ANSW_DATA DS  CL4ØØØ     .... ANSWER
STC_COMMEND   EQU *
        TITLE 'CMSIFSTC : COMMUNICATION_AREA IN THE APPC_INTERFACE'
CMS_PARAMETER DSECT
CMS_COMMAREA  DS  ØF         COMMUNICATION AREA IN THE APPC_INTERFACE
CMS_RETNCODE  DS  CL3        .. RETURN_CODE
CMS_FILLER    DS  XL1        .. FILLER
CMS_LENG_TRNS DS  H          .. LENGTH OF TRANSACTION_DATA
CMS_LENG_ANSW DS  H          .. LENGTH OF ANSWER
CMS_ADDR_TRNS DS  F          .. ADDRESS OF TRANSACTION_DATA
```

```
CMS_ADDR_ANSW DS  F          .. ADDRESS OF ANSWER
CMS_API_NAME  DS  CL8        .. NAME OF API TO EXECUTE
CMS_TIMEOUT   DS  F          .. TIMEOUT VALUE IN THE CMS_APPC_SERVER
CMS_TIMESTMP  DS  CL16       .. TIMESTAMP SET BY THE CMS_APPC_SERVER
         TITLE 'CMSIFSTC : DSECTS'
IEZCOM   DSECT
         IEZCOM                    MAP COMMUNICATIONS_PARAMETER_LIST
IEZCIB   DSECT
         IEZCIB                    MAP COMMAND_INPUT_BUFFER
         CVT   DSECT=YES,LIST=NO   MAP CVT
         IHAASVT DSECT=YES         MAP ASVT
         IHAASCB DSECT=YES         MAP ASCB
         IHAPSA DSECT=YES          MAP PSA
         IHASDWA DSECT=YES         MAP SDWA
         END
```

API00001 SOURCE

```
*********************************************************************
*  MODULE NAME :  API00001
*  FUNCTION    :  LINKED TO FROM THE API_SERVER
*                 WHATEVER YOU WANT
*                 RETURN ANSWER TO THE API_SERVER
*  REQUIREMENT :  LINK WITH AMODE=31 AND RMODE=24
*********************************************************************
         TITLE 'API00001 : REGISTER USING'
R3       EQU   3           BASE FOR RETURN_CODE
R4       EQU   4           BASE FOR LENGTH_OF_TRANSACTION_DATA
R5       EQU   5           BASE FOR TRANSACTION_DATA
R6       EQU   6           BASE FOR LENGTH_OF_ANSWER
R7       EQU   7           BASE FOR ANSWER
R8       EQU   8           WORK
R12      EQU   12          BASE REGISTER
R13      EQU   13          ADDRESS OF SAVE AREA
R14      EQU   14          RETURN REGISTER
R15      EQU   15          ENTRY ADDRESS + RETURN CODE
         TITLE 'API00001 : PARAMETERS FROM THE API_SERVER'
STC_PARM1     DSECT
STC_RETNCODE  DS  CL3        .. RETURN_CODE
              USING STC_PARM1,R3
STC_PARM2     DSECT
STC_LENG_TRNS DS  H          .. LENGTH OF TRANSACTION_DATA
              USING STC_PARM2,R4
STC_PARM3     DSECT
STC_TRNS_DATA DS  CL4000     .. TRANSACTION_DATA
              USING STC_PARM3,R5
STC_PARM4     DSECT
STC_LENG_ANSW DS  H          .. LENGTH OF ANSWER
              USING STC_PARM4,R6
STC_PARM5     DSECT
```

```
STC_ANSW_DATA DS  CL4000          .. ANSWER
              USING STC_PARM5,R7
         TITLE 'API00001 : INITIAL SET-UP'
API00001 CSECT
         USING API00001,R15       ESTABLISH TEMPORARY MODULE
         B     ENTRYLNK           BRANCH AROUND EYECATCHERS
         DC    CL8'API00001'      EYECATCHERS FOR START OF MODULE
         DC    CL8'&SYSDATE'      .. AND DATE OF ASSEMBLY
         DC    CL8'&SYSTIME'      .. AND TIME OF ASSEMBLY
ENTRYLNK DS    0H
         STM   R14,R12,12(R13)    SAVE CALLER'S REGISTERS
         LR    R12,R15            SET MODULE BASE REGISTER
         DROP  R15                DROP TEMPORARY MODULE ADDRESSABILITY
         USING API00001,R12       ESTABLISH MODULE ADDRESSABILITY
         LA    R15,SAVEAREA       GET ADDRESS OF MY SAVE AREA
         ST    R15,8(R13)         SAVE FORWARD POINTER
         ST    R13,SAVEAREA+4     SAVE BACKWARD POINTER
         LA    R13,SAVEAREA       SET UP R13 TO POINT TO SAVE AREA IN
*                                 .. CASE THIS MODULE MAKES A CALL
         LM    R3,R7,0(R1)        LOAD PARM ADRESSES FROM API_SERVER
         TITLE 'API00001 : DO WHATEVER THE RECEIVED DATA TELLS YOU'
CHK_DATA DS    0H
         CLC   =C'ECHO',STC_TRNS_DATA  ECHO DATA RECEIVED ?
         BE    DO_ECHO            .. YES
         CLC   =C'WTOR',STC_TRNS_DATA  ANSWER VIA THE CONSOLE ?
         BE    DO_WTOR            .. YES
         CLC   =C'WAIT',STC_TRNS_DATA  WAIT FOR A LONG TIME ?
         BE    DO_WAIT            .. YES
         CLC   =C'ABND',STC_TRNS_DATA  ABEND THE TASK ?
         BE    DO_ABND            .. YES
         CLC   =C'ERRC',STC_TRNS_DATA  RETURN ERROR_CODE ?
         BE    DO_ERRC            .. YES
         CLC   =C'LOOP',STC_TRNS_DATA  MAKE A LOOP ?
         BE    DO_LOOP            .. YES
         B     DO_ELSE
DO_ECHO  DS    0H
         LA    R8,STC_TRNS_DATA+4      R8  -> MOVE_FROM FIELD
*                                      ...    (BYPASS TEXT "ECHO")
         LA    R10,STC_ANSW_DATA       R10 -> MOVE_TO FIELD
         LH    R9,STC_LENG_TRNS
         SH    R9,=H'4'
         LR    R11,R9                  R9/R11 = LENGTH OF DATA TO MOVE
         STH   R11,STC_LENG_ANSW                BUILD ANSVAR (LENGTH)
         MVCL  R10,R8                           BUILD THE ANSVAR (DATA)
         MVC   STC_RETNCODE,=C'000'    RETURN_CODE SAYS OK
         B     END_PROG
DO_WTOR  DS    0H
         LH    R9,STC_LENG_TRNS        R9 = LENGTH OF RECEIVED DATA
         SH    R9,=H'4'                .. LESS 4 (BYPASS TEXT "WTOR")
         CH    R9,=H'60'               MORE THAN 60 CHARACTERS ?
         BNH   WTOR_01                 .. NO
```

```
         LA    R9,60                    .. YES, ONLY WRITE 60 CHARACTERS
WTOR_01  DS    0H
         BCTR  R9,0                     LESS 1 FOR EXECUTE
         EX    R9,WTOR_MVC              MOVE RECEIVED DATA TO THE WTOR
         B     WTOR_02
WTOR_MVC MVC   WTO_TEXT+20(0),STC_TRNS_DATA+4
WTOR_02  DS    0H
         XC    WTO_ECB,WTO_ECB          CLEAR ECB FOR WTOR
         WTOR  TEXT=(WTO_MSG,WTO_RPL,WTO_LNG,WTO_ECB),                  *
               MCSFLAG=REPLY,                                           *
               ROUTCDE=2,DESC=(7)
         WAIT  ECB=WTO_ECB              WAIT ON REPLY
         LA    R9,WTO_LNG
         STH   R9,STC_LENG_ANSW                 BUILD ANSVAR (LENGTH)
         MVC   STC_ANSW_DATA(WTO_LNG),WTO_RPL  BUILD THE ANSWER (DATA)
         MVC   STC_RETNCODE,=C'000'     RETURN_CODE SAYS OK
         B     END_PROG
DO_WAIT  DS    0H
         STIMER WAIT,BINTVL=WAITTIME
         LA    R9,WAITLNG
         STH   R9,STC_LENG_ANSW                 BUILD ANSVAR (LENGTH)
         MVC   STC_ANSW_DATA(WAITLNG),WAITRPL  BUILD THE ANSWER (DATA)
         MVC   STC_RETNCODE,=C'000'     RETURN_CODE SAYS OK
         B     END_PROG
DO_ABND  DS    0H
         LA    R9,ABNDLNG
         STH   R9,STC_LENG_ANSW                 BUILD ANSVAR (LENGTH)
         MVC   STC_ANSW_DATA(ABNDLNG),ABNDRPL  BUILD THE ANSWER (DATA)
         MVC   STC_RETNCODE,=C'000'     RETURN_CODE SAYS OK
         DC    X'FFFFFFFF'              SUBTASK ABENDS
DO_ERRC  DS    0H
         MVC   STC_RETNCODE,=C'600'     RETURN_CODE SAYS ERROR
         B     END_PROG
DO_LOOP  DS    0H
         STIMER WAIT,BINTVL=LOOPTIME
DO_LOOP1 DS    0H
         LA    R9,LOOPLNG
         STH   R9,STC_LENG_ANSW                 BUILD ANSVAR (LENGTH)
         MVC   STC_ANSW_DATA(LOOPLNG),LOOPRPL  BUILD THE ANSWER (DATA)
         MVC   STC_RETNCODE,=C'000'     RETURN_CODE SAYS OK
         B     DO_LOOP1
DO_ELSE  DS    0H
         LA    R9,ELSELNG
         STH   R9,STC_LENG_ANSW                 BUILD ANSVAR (LENGTH)
         MVC   STC_ANSW_DATA(ELSELNG),ELSERPL  BUILD THE ANSWER (DATA)
         LH    R9,STC_LENG_TRNS
         CVD   R9,DOB
         MVC   STC_ANSW_DATA+21(6),=X'402020202120'
         ED    STC_ANSW_DATA+21(6),DOB+5
         MVC   STC_RETNCODE,=C'000'     RETURN_CODE SAYS OK
         B     END_PROG
```

```
         TITLE 'API00001 : EXIT THE PROGRAM'
END_PROG DS    0H
         L     R13,SAVEAREA+4      RETRIEVE BACKWARD POINTER
         LA    R15,0               RETURN CODE = 0
         RETURN (14,12),RC=(15)    RETURN CONTROL TO SYSTEM
         TITLE 'API00001 : WORK AREAS'
SAVEAREA DS    18F                 SAVE_AREA IN MY_MODULE
WTO_ECB  DS    F                   WTOR ECB USED BY THE SYSTEM
WTO_MSG  DS    0H                  WTOR MESSAGE_FIELD
         DC    AL2(L'WTO_TEXT)             .. LENGTH OF MESSAGE
WTO_TEXT DC    CL80'API_000I  API00001: '  .. MESSAGE
WTO_RPL  DS    CL60                WTOR REPLY_FIELD
WTO_LNG  EQU   L'WTO_RPL           LENGTH OF REPLY_FIELD
WAITTIME DC    F'6000'             WAIT IN 60 SECONDS
WAITRPL  DC    C'API00001 HAS WAITED IN 60 SECONDS'
WAITLNG  EQU   L'WAITRPL
ABNDRPL  DC    C'API00001 HAS ABENDED'
ABNDLNG  EQU   L'ABNDRPL
LOOPTIME DC    F'2500'             WAIT 25 SECONDS
LOOPRPL  DC    C'API00001 HAS WAITED 25 SECONDS'
LOOPLNG  EQU   L'LOOPRPL
ELSERPL  DC    C'API00001 HAS RECEIVED XXXXX CHARACTERS'
ELSELNG  EQU   L'ELSERPL
DOB      DS    D                   WORK FIELD
         LTORG
         END
```

Klem Thomsen
Systems Programmer
Nykredit (Denmark)

DFSMS and the Catalog Address Space

The DFSMS product integrates and expands many of the functions previously available in the MVS/DFP product and the DFHSM and DFDSS products. All of the functionality available in these program products and major new functions are contained in DFSMS, the functional components of which are:

DFSMSdfp — Storage, data, program, and device management functions.

DFSMSdss — Data movement, copy, back-up, and space management functions.

DFSMShsm — Back-up, recovery, migration, and space management functions.

DFSMSrmm — Management functions for removable media such as optical volumes and tape/cartridge media.

With DFSMS, one area to take into consideration is the catalog configuration. IBM maintains that performance is not the main consideration in this area, but that the design of the system to allow the easiest method of recovery of damaged catalogs and minimizing system disruption is. DFSMS does provide a number of options for improving performance as well.

One of the main factors to improve catalog performance is caching. I/O to a catalog and the time taken to satisfy the request can be reduced by using special catalog caching. For example, if the System Master Catalog only contains system datasets, catalogs, and alias entries it will be read into storage when the system is IPLed.

Also, IBM recommends the removal of all JOBCAT and STEPCAT JCL statements. The searching performed when these statement are coded degrades performance and can also produce unpredictable results if the number of catalogs specified on the statements exceeds the maximum open catalog limit. The limit is set to 50 by default. Heavy usage of JOBCAT and STEPCAT can affect overall system performance.

The simplest method of improving performance is to utilize cache to maintain catalog records within main storage or a dataspace. This method of operation reduces the number of direct reads to DASD. The In-Storage Catalog facility (ISC) residing in the Catalog Address Space (CAS) is one type of cache and uses the system's main storage. Catalog Dataspace Cache (CDSC) is a separate cache in an ESA dataspace. The two types of cache are optional and can be stopped and started without the need for an IPL.

Both types of cache can be used together but they cannot contain the same catalog entries. The Master Catalog must reside in ISC. The following rules apply:

1 For the Master Catalog, all records accessed sequentially or by key are cached except for alias records, which are kept in a main storage table.

2 Only records accessed by key are cached for user catalogs.

3 Records are cached in ISC and CDSC depending on the specification that you set up.

The ISC is the default catalog cache and each catalog is allocated a fixed amount of space, the Least Recently Used algorithm being used to manage the code. Catalogs updated infrequently utilize ISC efficiently. Performance is affected, however, if catalogs are shared between systems. Unlike user catalogs, the Master Catalog can use as much ISC as it needs so care must be taken.

CDSC resides in an MVS dataspace defined in the COFVLF*xx* member of SYS1.PARMLIB. Virtual Lookaside Facility (VLF), which is started at IPL time normally using the START VLF command, is used to manage the CDSC. By updating COFVLF as shown later, the VLF can add or remove catalogs from CDSC. CDSC does not restrict storage usage and cache records are created until no space is left in the dataspace cache. Once full, the cache will take no more records and individual records are only removed from the cache if the record is actually deleted from the catalog. The MODIFY CATALOG commands can be used to manage the CDSC.

If a catalog has SHAREOPTIONS (3 4) coded when defined using IDCAMS and resides on shared DASD, it is considered shared even if it is only used on a single system. ISC and CDSC undergo special checking to allow the catalogs to retain their integrity in this circumstance at the expense of performance.

If ISC is used and a record is updated, the entire system's ISC is released and a new ISC is created. If CDSC is used, individual updates can be identified and the record can be updated without the CDSC being released. Shared catalogs, therefore, should always be added to CDSC unless of course they are the Master Catalog. To make a catalog non-shareable, code SHAREOPTIONS (3 3). If this is coded then the catalog should be placed on a non-shared volume to avoid possible corruption.

To set up the CDSC you must edit the COFVLF*xx* member of SYS1.PARMLIB. The following class statement defines two catalogs to the CDSC and specifies a maximum virtual storage of 2 megabytes:

```
CLASS     NAME(IGCCAS)
          EMAJ(CATALOG.ICF1)
          EMAJ(CATALOG.ICF2)
          MAXVIRT(512)
```

The syntax of the CLASS statement is:

NAME(IGCCAS) Specifies the class name for the catalog dataspace cache.

EMAJ(*name*) Specifies the dataset name of the catalog to add to CDSC.

MAXVIRT(*value*) The decimal value multiplied by 4K of storage to be used as a ceiling. The default value is 256 and the maximum value is 524288.

The MODIFY CATALOG,REPORT,VLF command can be used to monitor CDSC performance. An example of the command and its associated output looks like this.

```
F CATALOG,REPORT,VLF

IEC351I        CATALOG ADDRESS SPACE MODIFY COMMAND ACTIVE
IEC359I        CATALOG REPORT OUTPUT 686
**CAS*******************************  CATALOG DATA SPACE CACHE    ******
**      HIT%   RECORDS   SEARCHES    HITS    DELETES   SHARING  INVALID  *
*************************************************************************
** CATALOG.ICF1                                                        **
**      072%   00000981   000040C3 00002EE2 00000017   00000000 0000000 **
** CATALOG.ICF2                                                        **
**      091%   00001ACA   000040C3 00002EC2 00000019   00000000 0000000 **
*************************************************************************
IEC352I CATALOG ADDRESS SPACE MODIFY COMMAND COMPLETED
```

The CAS performs catalog functions for the system. Most of the catalog modules and related control blocks are located in this address space above the 16-megabyte line. This improves virtual storage in a user's private area when catalog functions are performed.

The CAS contains tables that record the user catalog names, aliases, and associated volume serial numbers that are built when MVS is started. Service tasks specified in the SYSCAT*xx* member are also created in the address space and a table known as the CRT is created to keep track of these catalog service tasks. Any changes made in the Master Catalog are automatically reflected in the CAS. On systems where catalogs are shared, the CAS is updated on each system to

maintain the integrity of the system. When a catalog function is requested by a user the CAS will assign a service task to that function. The CAS is not restricted by the number of tasks specified in the SYSCAT*xx* member. The default maximum limit is 180: if fewer tasks than this are created, the CAS will create new tasks as needed up to the maximum. When a task completes, the service task is freed until the lower limit specified in SYSCAT*xx* is reached.

The CAS system has four special tasks for its own use. These are the mother task, allocate task, analysis task, and modify task. The mother task is used to track the CAS service tasks and other functions. If the mother task is abended for any reason, all service tasks will be stopped and the CAS will restart. The allocate task performs VVDS and catalog allocation. The analysis task is dedicated to checking the CAS for errors. The modify task is used by the MODIFY CATALOG operator command. Only one modify command can be handled at a time so if two commands are entered together the second is always rejected.

The whole process of controlling CAS is via operator MODIFY commands. Some examples of how it can be used follow. If you need to recover a volume containing a catalog or VVDS dataset then that volume must be varied off-line to the system. The vary command may not work if a VVDS or catalog on the volume is open as they will be allocated to the CAS. You can issue the MODIFY CATALOG,OPEN command to determine which catalogs are open. You can unallocate the catalog using the MODIFY CATALOG,UNALLOCATE command and the VVDS can be unallocated using the MODIFY CATALOG, VUNALLOCATE command. Before issuing these commands you should issue the MVS VARY command to take the device off-line.

After the volume has been recovered, the VVDS's location may have changed. Because the CAS will have control blocks for every VVDS in storage, the change of location will mean that VSAM and SMS-managed datasets will be inaccessible for the volume where the VVDS has been moved. The unallocate will cause the control blocks to be rebuilt after a request for the volume is processed by catalog management. If, for some reason, this does not occur, then issue MODIFY CATALOG,VCLOSE. Remember that you must still VARY the device off-line as the UNALLOCATION of the catalog and VVDS

still means they can be accessed. If the VARY off-line is not performed then errors could occur.

You must also take care when applying PTFs to a system where SMS is running. Such systems mean that the JOBCAT and STEPCAT DD statements cannot be utilized to point to another system's Master Catalog. In order to apply fixes you can use the SYS% conversion facility. This can be activated on a system at IPL time or by issuing the MODIFY CATALOG,SYS%ON command. It can then be deactivated if required by issuing the MODIFY CATALOG,SYS%OFF command. The current setting for the SYS% can be determined by issuing the MODIFY CATALOG,REPORT command.

Other catalog commands that can be useful are:

DUMP	Causes dynamic dumping of the catalog address space during diagnostic testing.
ENTRY (message IEC349I)	Provides the storage address and PTF level of a catalog management load module, so that Serviceability Level Indication Processing (SLIP) trap(s) can be set.
LIST (message IEC347I)	Provides the task identification and address of CAS tasks, so that the ID can be used in other MODIFY CATALOG commands. The listing also supplies information on job names, elapsed time of the job, and other selected information.
OPEN (message IEC348I)	Provides information about catalogs allocated and the number of jobs using the catalog. This information can be used to close catalogs on volumes which must be recovered.
REPORT (message IEC359I)	Provides general information.

You can also use the MODIFY CATALOG command to fix problems with control blocks in the CAS. If these become damaged it can be assumed that some problem exists with the physical catalog. It is best to attempt a rebuild of the control blocks before attempting catalog recovery. The following commands can be useful in fixing temporary problems.

ABEND	To end the CAS task abnormally. This should only be used after you have unsuccessfully tried END, or when you are ending the CAS allocate, analysis, or modify tasks.
CLOSE	To release all CAS storage for the specified catalog. The catalog is not locked, and is re-opened to CAS by the next catalog request that accesses the catalog. The catalog is not unallocated from CAS, and the Common Services Area storage used by the catalog is not freed.
END	To end the CAS task. You can choose to redrive the request, or simply end it. This is the preferred method of ending a CAS task, and should be used before attempting ABEND.
RESTART	To abnormally end the CAS mother task and restart it in a new address space. This option should only be used when your only other option is to IPL the system. If the RESTART fails, you must IPL.
UNALLOCATE	To unallocate and close a catalog from CAS without releasing CAS storage.
VCLOSE	To close the VVDS which resides on the specified volume. The VVDS is opened by the next request which tries to access it.
VUNALLOCATE	To unallocate all VVDSs from the catalog address space.

The full syntax of the command is as follows:

```
MODIFY   CATALOG,  ABEND(TASK)
                   ALIASLEVEL(N)
                   ALLOCATE(CATNAME),NOISC|NOVLF
                   CATMAX(NN)
                   CLOSE(CATNAME)
                   DUMPON|DUMPOFF
                   END(ID),REDRIVE|NOREDRIVE
                   ENTRY(CSECTNAME)
                   ISC|NOISC(CATNAME)
                   LIST(TASK)
                   OPEN(VOLSER)
                   REPORT,VLF(CATNAME)|DUMP
                   RESTART
                   ROTATE|NOROTATE
                   SYS%ON|SYS%OFF
                   TASKMAX(NN)
                   UNALLOCATE(CATNAME
                   VCLOSE(VOLSER)
                   VLF|NOVLF(CATNAME)
                   VUNALLOCATE|NOVUNALLOCATE
```

John Bradley
Systems Programmer (UK)

A mini editor revisited

The original MEDIT code published in the article *A mini editor* in the April 1997 issue of *MVS Update* fails when more than 32,000 bytes of data are to be processed. This problem is corrected when the length of the dynamic area is truncated to 4096 bytes. The corrected BuildDisplay routine follows (only the LENGTH line is actually new).

```
BuildDisplay:
  /* Create DynamicArea for display */
  darea = '' /* clear area */
  DO i = i0 TO imax
    darea = darea||buf.i
    IF LENGTH(darea) > 4096 THEN LEAVE /* maximum display
size */
  END
RETURN
```

Anthony Rudd (Germany)

MVS news

Version 1.9.5 of BETA 42, its job scheduling package, has been unveiled by Beta Systems Software GmbH, adding new variables, enhanced processing criteria, and interfaces to other BETA products. BETA 42 offers a new Process Predecessor and a Variable Predecessor, which allow users to group together related tasks into processes. The execution, delay in execution, or non-execution of tasks can be controlled by the success, delay, or failure of jobs within a process or by the values of variables. New in version 1.9.5 are a newly-refined, panel-driven ISPF interface; the ability to clone job commands and definitions; and a batch load utility for importing task definitions when performing mass updates or converting from other job schedulers.

For further information contact:
Beta Systems Software GmbH, Kurfürstendamm 182, D-1000 Berlin 15, Germany
Tel: +49 30 884 3060 or
Beta Systems Software Inc, One Securities Center, 3490 Piedmont Road, Suite 1100, Atlanta, GA 30305, USA
Tel: (404) 812 1556/(800) 777 9864
Fax: (404) 812 1563 or
Beta Systems Software, Unit 3, Heron Industrial Estate, Basingstoke Road, Spencers Wood, Reading, Berks, RG7 1PJ, UK
Tel: (01734) 885175
Fax: (01734) 884899.

Sterling Software Inc has released Version 3.2 of SAMS:Vantage, an automated storage management tool. Up to 50 MVS images can be monitored and automatically analysed with the resulting information displayed on a single screen containing critical desired-state indicators for all systems. Colour-coded alert bars enable administrators to see, at a glance, the health of the whole MVS storage system. More detailed information can be obtained by pointing and clicking on alert bars to zoom directly in on the problem on the system in question. SAMS:Vantage also boasts I/O Plus, a feature that allows it to operate across multiple RAID subsystems **and** conventional DASD technologies. I/O Plus analyses I/O performance, identifies bottlenecks, and recommends device-specific solutions.

For further information contact:
Sterling Software Inc, 11050 White Rock Road, Suite 100, Rancho Cordova, CA 95670-6095, USA
Tel: (916) 635 5535
Fax: (916) 635 5604 or
Sterling Software Ltd, 1 Longwalk Road, Stockley Park, Uxbridge, Middx, UB11 1DB, UK
Tel: (0181) 867 8000
Fax: (0181) 867 8001.

xephon