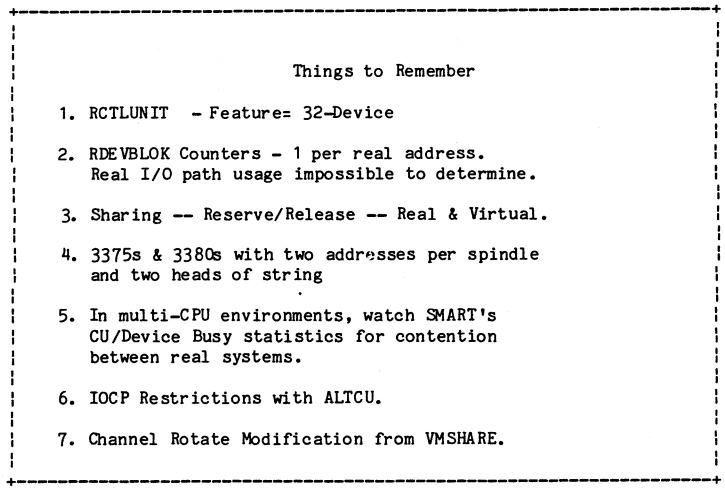
3375 and 3380 Device Construction

IBM 3375s and 3380s have multiple logical devices corresponding to unique device addresses on each physical unit. Both have two logical volumes per physical spindle. The 3375s can be given an second controller by acquiring a model D1. The 3380s can also have a second controller by acquiring a model AA4. With these second controllers, two data transfers on the string can occur simultaneously. However, if one logical volume on a physical spindle is busy, any attempt to access the other volume on the same spindle will also report back a busy. This is a subtle design feature (deficiency) with the 3375s. Although I would be happy for someone to correct me, I believe that the same feature is present with the 3380s. Cornell has addressed the problem , excuse the pun, by putting high-use data (like the CP nucleus, spool, and paging spaces,) on even numbered addresses. We reserve the odd numbered addresses for relatively low-use user data. After we gain some experience with the approach, I hope we will be able to report how well it performs.

Summary

The following chart trys to encapsulate the major concerns about an I/O configuration when you are using VM/370 in a complex environment.



SHARE

SHARE SESSION REPORT

61	B632	CMS IUCV		260
SHARE NO.	SESSION NO.	SESSION TITLE		ATTENDANCE
B/Systems CP Management			Stuart Bell	MTW
PROJECT			SESSION CHAIRMAN	INST. CODE

1820 Dolley Maddison Blvd., McLean, VA 22102 (703) 827-6366

SESSION CHAIRMAN'S COMPANY, ADDRESS, AND PHONE NUMBER

Using IUCV in CMS
SHARE 61

August 22, 1983

Samuel A. Thompson

IBM Corporation
PO Box 6
Endicott, NY 13760

ABSTRACT

The Inter-User Communication Vehicle (IUCV) was introduced in VM/SP Release 1 and has been enhanced in Releases 2 and 3 of VM/SP. IUCV is a general use communications vehicle which allows two programs executing in two different virtual machines to communicate with one another. The paper will give a general overview of IUCV and then concentrate on the changes which were made to IUCV in both the CP and CMS components of VM/SP Release 3.

1/E/LEJ/1

CONTENTS

IUCV OVERVIEW

The Inter-User-Communications-Vehicle (IUCV) was introduced in VM/SP Release 1. It is a general use communications medium which allows communications between two virtual machines and/or CP. It allows two programs running in two different virtual machines to communicate by sending messages back and forth internally in the same fashion that two people would communicate via the CP message command. Currently, IUCV supports three system services, which will be covered in detail later:

1. Console Communications

2. Message

3. DASD Block I/O

When two virtual machines are using IUCV to communicate, it appears to them as though they actually have a real communications link between the two virtual machines. In actuality, both virtual machines have a link with the IUCV code in CP. CP keeps track of which link from virtual machine 1 corresponds to the link from virtual machine 2. Because this is all transparent to the user, IUCV can be a well defined asynchronous communications medium.

IUCV CONCEPTS

IUCV is implemented as an operation code of X'B2F0'. It is a software only implementation, meaning that when the hardware tries to execute the X'B2F0' instruction, a program check is generated, the CP program check handler receives control, recognizes that it is an IUCV instruction, and gives control to the CP IUCV code. There are 18 functions available which allow two communicators to establish communications, perform communications, and terminate communications, as well as some miscellaneous additional functions. IUCV supports multiple concurrent communications, and also multiple parallel concurrent communications. There are some directory controls available, so installations can control who is authorized to use it. An IUCV communication is really an asynchronous communication with CP. Virtual machines are notified that an IUCV event has occurred via an IUCV external interrupt. This gives IUCV its asynchronous nature. An assembler macro is provided to aid the user in coding the IUCV parameter list.

IUCV Messages

IUCV Information transfer is via a "message". To IUCV itself, the message is actually a control block which contains information about the source and target of the message and may or may not actually contain the message data itself. An IUCV message may be one or two way and may be a normal or a priority message. An IUCV message is created via the IUCV SEND function, and internally, the IUCV message is moved among IUCV queues.

IUCV Queues

There are 3 IUCV queues for each virtual machine. A send queue, which is where incoming message blocks are placed, a receive queue where message blocks are placed while a reply is pending, and a reply queue where message blocks are placed once the message has been replied to. The life of an IUCV message block is as follows. An IUCV SEND creates the message and the message block is placed on the target's send queue. The target of the message issues the IUCV RECEIVE function and the message block is moved to the target's receive queue. The target then issues the REPLY function, and the message block gets moved to the source's reply queue. Once the "incoming reply" external interrupt is reflected to the virtual machine, the message block ceases to exist. The IUCV queues are First-In-First-Out (FIFO) with priority.

IUCV Paths

A "path" is a communications capability between two IUCV communicators. It is the pipeline for communications, as messages move across a path. Multiple paths to different virtual machines are allowed, and multiple paths to one virtual machine are permitted also. A path is numbered and well defined at both ends. Each end of an IUCV path is assigned a number by IUCV when the path is established with an IUCV CONNECT or ACCEPT. Path numbers are assigned sequentially starting from zero. Since each path is well defined at both ends, either end of the path can SEVER or QUIESCE it.

IUCV External Interrupts

External interrupts are used to communicate asynchronously with virtual machines. Whenever an IUCV event has occurred for a virtual machine, CP reflects an IUCV external interrupt. The IUCV external interrupt type is X'4000', which is unique. When an external interrupt is reflected to a virtual machine, information about the IUCV event which has just occurred is stored in a buffer which is identified to IUCV by the virtual machine via the Declare Buffer function. There are some controls which allow a virtual machine to disable IUCV external interrupts:

- Bit 7 in the PSW can be turned off to disable all external interrupts.
- Bit 30 in control register 0 can be turned off to disable IUCV external interrupts.
- The IUCV SETMASK and SETCMASK functions can be used to disable specific IUCV external interrupt types.

IUCV external interrupts fall into two categories. Path related interrupts include connection pending, connection complete, path quiesced, path resumed, and path severed. Message related interrupts are incoming priority and non-priority messages and incoming priority and non-priority replies. Each individual IUCV external interrupt type can be masked off via the IUCV SETMASK and SETCMASK functions.

IUCV FUNCTIONS

IUCV functions fall into four categories:

- Establishing communications
- Message handling
- Some miscellaneous functions
- Terminating communications

The first catagory, establishing communications, contains four functions: Query, Declare Buffer, Connect, and Accept. The Query function returns the maximum number of paths which are allowed in a virtual machine and the size of the external interrupt buffer needed. The Declare Buffer function identifies the area of storage which the virtual machine wants IUCV to use as its external interrupt buffer. The Connect function initiates a communication with another virtual machine, CP or itself. The Accept function, completes a communications link initiated by another communicator.

The message handling category contains five IUCV functions: Send, Describe, Receive, Reply, and Test Completion. The Send function begins a communication and creates a message block in IUCV. The Describe function is used to interrogate IUCV to see if any pending messages are waiting. This function is used to avoid having CP reflect an incoming message external interrupt or is used in CP code where external interrupts cannot be generated. The Receive function is used to obtain the message data after an incoming message external interrupt has occurred. The Reply function is used to respond to a two-way message. The Test Completion function is used to interrogate IUCV to see if there are any incoming replies and avoid the external interrupt.

The third category consists of seven miscellaneous functions: Reject, Purge, Quiesce, Resume, Set Mask, Set Control Mask, and Test Message. Reject simply rejects an incoming message and tells the sender you do not wish to receive it. Purge is used by the initiator of a message to remove it from IUCV. Quiesce disallows any incoming messages on a path and Resume returns the path to normal. Set Mask is used to prevent specific message related and all control IUCV external interrupts from occurring. Set Control Mask prevents individual control external interrupts from occurring. Test Message will place the issuer's virtual machine in a wait state if no incoming messages or replies are pending.

The final category, terminating communications has two functions: Sever and Retrieve Buffer. The Sever function is used to terminate a path and the Retrieve Buffer function is used to terminate the IUCV environment for the virtual machine. No IUCV functions are permitted after the Retrieve Buffer function has been issued.

TYPES OF IUCV MESSAGES

An IUCV message can be either a two-way or a one-way message. For a two-way message, the IUCV Send function creates the message and the message is placed on the target's send queue. The Target side receives an incoming message external interrupt and issues the Receive function to transfer the message data from the

source's virtual machine to the target's. Internally, IUCV then moves the message block to the Target's receive queue. The target then issues the Reply function to respond to the message, IUCV transfers the reply data to the source's virtual machine and moves the message block to the source's reply queue. Once the incoming reply external interrupt is reflected, the communication is complete.

For a one-way message, the IUCV Send function creates the message and the message is placed on the target's send queue. The target side receives an incoming message external interrupt and issues the Receive function to transfer the message data from the source's virtual machine to the target's. Since no reply is needed, IUCV moves the message block to the source's reply (or message complete) queue. Once the incoming reply (message complete) external interrupt is reflected, the communication is complete.

Parameter List Data Protocol

In VM/SP Release 3, a new "parameter list data" protocol has been established. With this protocol, message data is passed in the IUCV parameter list itself instead of in a remote buffer. This protocol is defined to IUCV via the PRMDATA=YES option on the IUCV Connect or Accept. A message using this protocol is identified via the DATA=PRMMSG and PRMMSG= options on an IUCV Send or Reply. The message data itself may be up to 8 bytes in length. Using this new protocol will cause a reduction in IUCV path length as IUCV will no longer have to move a potential large amount of data from one virtual machine to another. Also with this new protocol, the target of the message no longer has to issue an IUCV Receive function, since the message data is available in the external interrupt buffer when the interrupt is reflected.

TERMINATING COMMUNICATIONS

When a communicator wishes to terminate an communications path, he issues an IUCV Sever function on the path number which was given to him by IUCV when the path was established. All active messages are either purged or rejected, and when control returns from IUCV, the path is no longer there. On the other side of the path in the other virtual machine, a sever external interrupt is reflected. The program in that virtual machine must then issue an IUCV Sever for the corresponding path number that identifies the path in that virtual machine. Once IUCV returns control, the path is no longer there on his side either.

QUIESCING A PATH

When the IUCV Quiesce function is issued on a path, no incoming messages are permitted. All out going messages are still allowed. Each end of the IUCV path is independent, so one end can be in Quiesce mode while the other is not. This function can be used for clean-up or catch-up work. A service virtual machine can suspend any incoming requests by Quiescing a path until it is caught up, or

it is ready to receive more requests. Normal communication is re-established with the IUCV Resume function.

DIRECTORY CONTROLS

There are some directory controls which allow an installation to control how IUCV will be used in its environment and who is authorized to use it. The IUCV statement can be specified in a user's directory entry to control which other users can connect to him. On this statement an installation can identify whether priority messages will be allowed and also how many messages can be outstanding on a path at any particular time. The default message limit is 10 and the maximum is 255.

Instead of a userid, two special keywords can be specified: ALLOW and ANY. ALLOW indicates that any userid can connect to this virtual machine. This keyword can be used for service virtual machines that everyone on the system might want to connect to. The ANY keyword indicates that this userid can connect to any virtual machine whether that virtual machine has an entry for it in its directory or not. This is useful for system programmer userids so they can connect to anyone.

The MAXCONN keyword on the option statement can be used to define the maximum number of paths allowed in this virtual machine. The default is 4 and the maximum is 65,535.

SYSTEM SERVICES

There are 3 system services currently supported in IUCV.

- Console Communications
- Message
- DASD Block I/O

CONSOLE COMMUNICATIONS

To connect to the Console Communications system service, *CCS should be specified as the userid in the IUCV connect parameter list. This system service is a specialized interface for passing screen data. The information passed is similar to the information returned from Diagnose 58. This system service was introduced in VM/SP Release 1 and is used by the VCNA service virtual machine.

MESSAGE

To connect to the Message system service, *MSG should be specified as the userid in the IUCV connect parameter list. This system service is used to intercept console output. Instead of having console output displayed on the screen, the output can be sent to the virtual machine via IUCV. The type of output sent or displayed, can be controlled via the CP SET command. This system service was introduced in VM/SP Release 2 and is used by the Programmable Operator.

DASD BLOCK I/O

To connect to the DASD Block I/O system service, *BLOCKIO should be specified as the userid in the IUCV connect parameter list. This system service provides device independent access to CMS disks formatted with a blocksize of 512, 1K, 2K, and 4K bytes. It provides the user with the ability to asynchronously read or write a full block of data to a CMS disk. This system service uses the new "parameter list data" protocol. This system service was introduced in VM/SP Release 3 and is used by SQL/DS.

CMS IUCV SUPPORT

WHY IS CMS IUCV SUPPORT NEEDED ?

IUCV treats a virtual machine as a single communications entity. Because of this, only one external interrupt buffer can be declared to IUCV for saving interrupt information. Also, in CMS, only one exit is available for handling external interrupts. The exit is defined by the HNDEXT macro, and it exit is responsible for handling all external interrupts, not just IUCV interrupts.

A problem occurs if two programs running in one virtual machine wish to use IUCV. Since only one external interrupt buffer and one general exit is permitted, both programs must coordinate their use of the buffer and the exit. This coordination is unlikely to occur when programs are developed in different parts of the world or if the developers never expected the two programs to be used together.

CMSIUCV SUPPORT

The CMS IUCV support introduced in VM/SP Release 3 solves these problems. With this support, CMS manages the external interrupt buffer and allows programs to define exits which will receive control only when an interrupt is intended for their program. One exit is permitted for each IUCV path, and an exit can also be defined to handle pending connect external interrupts which occur before a path is defined. The CMS external interrupt handler, DMSITE, has been modified to route IUCV external interrupts to the user defined exit addresses.

Two new functions are provided in the CMS IUCV support which allow programs to set up IUCV external interrupt addresses:

- HNDIUCV
- CMSIUCV

HNDIUCV function

The HNDIUCV function is a program identification function. It must be issued before any IUCV functions are performed to identify the program as an IUCV program. An exit address can be specified which will receive control whenever a pending connect external interrupt occurs for that program. A user fullword is available which is a fullword of storage that can be used to pass information to the exit routine. When the exit routine is driven, this fullword is passed to the exit in a register. A macro is provided in standard, list, and execute format to aid users in coding this function.

CMSIUCV function

The CMSIUCV function is a communication initiation and termination function. It should be issued whenever a program would normally have issued an IUCV Connect, Accept, or Sever. All other IUCV functions will be issued directly by a program. The CMSIUCV function issues the requested IUCV function, Connect, Accept, or Sever and then establishes a path specific exit which will receive control whenever an external interrupt occurs on the path. For the Sever function, the exit is terminated. A user word is available with the CMSIUCV macro also, and the macro is provided in standard, list, and execute forms.

CMS IUCV exits receive control as an extension of the CMS external interrupt handler. When driven, the significant registers contain:

- R0 UWORD
- R1 Savearea containing the registers and PSW
- R2 External Interrupt Buffer
- R13 User Savearea
- R14 Return address
- R15 Entry address

A COMPARISON OF IUCV AND CMS IUCV

- HNDIUCV SET replaces IUCV Query and Declare Buffer
- HNDIUCV CLR replaces IUCV Retrieve Buffer
- CMSIUCV replaces IUCV Connect, Accept and Sever
- "EXITS" replace the HNDEXT exit

LIMITATIONS OF THE CMS IUCV SUPPORT

Because IUCV treats the virtual machine as a single communications entity and CMS application programs run in supervisor state, there are some limitations of the CMS IUCV support. Programs using the CMS IUCV support are not entirely independent. Programs should not use IUCV functions which eliminate or bypass external interrupts. Since the external interrupt routing mechanism is the basis of the CMS IUCV support, if external interrupts are avoided, the CMS IUCV support will be useless. Also, many IUCV functions affect the entire virtual machine and if any of these functions are indeed issued by a program, other programs running in the virtual machine may be affected. CMS can not prevent a program from issuing such an IUCV function, because an IUCV function is issued directly and can't be intercepted by CMS.

SUMMARY OF THE CMS IUCV SUPPORT

With the addition of the CMS IUCV support, using IUCV from CMS is made easier. In particular, it allows multiple programs running in one virtual machine to use IUCV within certain guidelines. CMS manages the external interrupt buffer and routes external interrupts to user supplied exit addresses. In this manner, programs no longer have to handle all external interrupts, but just those IUCV interrupts which are intend for their programs. The CMS IUCV support does have it's minor limitations, however, the new support makes it much easier to use IUCV than what exists today, and it does provide multiple programs running in one virtual machine with the ability to use IUCV.

```
*****************************************************************  MSG00010
*                                                               *  MSG00020
*    This program demonstrates how one might use the new CMS IUCV *  MSG00030
*    support in VM/SP Release 3 to utilize the CP Message System  *  MSG00040
*    Service.  It establishes communications via IUCV and then    *  MSG00050
*    waits for incoming messages.  Each time a message comes in,  *  MSG00060
*    it is logged in a file named 'MSG FILE'.  When the user       *  MSG00070
*    wishes to have the program terminate, he sends himself a      *  MSG00080
*    message which contains 'STOP'.                                *  MSG00090
*                                                                  *  MSG00100
*    This program does work (I've tried it), however, it could use *  MSG00110
*    a lot of work as it virtually ignores any error conditions.   *  MSG00120
*    It is presented as a sample and should be viewed as such.     *  MSG00130
*                                                                  *  MSG00140
*    The program illustrates some of the new functions which will  *  MSG00150
*    be available in VM/SP Release 3.  The function invocations,   *  MSG00160
*    macro operands, parameter list formats, etc. are all subject  *  MSG00170
*    to change when VM/SP Release 3 is made available.             *  MSG00180
*                                                                  *  MSG00190
*                                                                  *  MSG00200
*****************************************************************  MSG00210
         SPACE 2                                                     MSG00220
MSG      CSECT                                                       MSG00230
         BALR R12,0               Establish                          MSG00240
         USING *,12               addressability                     MSG00250
         USING NUCON,R0                                              MSG00260
         SSM   DISABLE            Disable interrupts till we want em MSG00270
         LR    11,14              Save the return address in R11     MSG00280
*                                                                    MSG00290
*  Tell CMS that this program wishes to use IUCV, and will be        MSG00300
*  identified by the name of SAM. An address is given for pending    MSG00310
*  connects, however, this should never happen.                      MSG00320
*                                                                    MSG00330
         HNDIUCV SET,NAME=SAM,EXIT=CONPEND                           MSG00340
         LA    R2,IUCVPLST        Get the address of our IUCV parameter MSG00350
         USING IPARML,R2          list and establish addressability  MSG00360
*                                                                    MSG00370
*  Build the parameter list with the IUCV macro, then establish the  MSG00380
*  connection and set up the path's exit                             MSG00390
*                                                                    MSG00400
         IUCV CONNECT,PRMLIST=(R2),USERID=STARMSG,MF=L               MSG00410
         CMSIUCV CONNECT,NAME=SAM,PRMLIST=(R2),EXIT=MSGPATH          MSG00420
         MVC   PATHID(2),IPPATHID   Save the path id IUCV gives us   MSG00430
         DROP  R2                                                    MSG00440
*                                                                    MSG00450
*  Wait for *MSG to tell us our connection is complete. Our exit will MSG00460
*  post the CONCOMP ECB when the interrupt comes in.  Note that WAITECB MSG00470
*  internally will enable us for interrupts.                         MSG00480
*                                                                    MSG00490
         WAITECB ECB=CONCOMP,FORMAT=OS                               MSG00500
         LA    R1,SETMSG          Issue the SET MSG IUCV command to  MSG00510
         SVC   202                Tell CP we want messages via IUCV  MSG00520
         DC    AL4(1)                                                MSG00530
*                                                                    MSG00540
*  Wait here for messages.  The MSGIN ECB will be posted each time a MSG00550
```



```
*  message comes in. If the message is 'STOP', quit and cleanup,     MSG00560
*  otherwise, write the msg to a log file name 'MSG FILE'            MSG00570
*                                                                    MSG00580
WAITLOOP EQU  *                                                     MSG00590
         WAITECB ECB=MSGIN,FORMAT=OS                                MSG00600
         CLC   MSGAREA+8(4),=CL4'STOP'   If the message says to stop, MSG00610
         BE    CLEANUP                   Stop and cleanup           MSG00620
         FSWRITE 'MSG FILE A',FORM=E,BUFFER=MSGAREA,RECFM=F,BSIZE=140 MSG00630
         XC    MSGIN(4),MSGIN            Zero out the ECB           MSG00640
         B     WAITLOOP                  Wait for next message      MSG00650
*                                                                    MSG00660
*  Issue the CP SET MSG ON command to tell CP we no longer want      MSG00670
*  messages via IUCV.  Then issue the HNDIUCV CLR macro to tell CMS  MSG00680
*  we no longer wish to use IUCV and return to CMS.                  MSG00690
*                                                                    MSG00700
CLEANUP  EQU  *                                                     MSG00710
         LA    R1,SETON           Issue the SET MSG ON command to    MSG00720
         SVC   202                Tell CP we no longer want our      MSG00730
         DC    AL4(1)             messages via IUCV                  MSG00740
         HNDIUCV CLR,NAME=SAM                                       MSG00750
         BR    R11                return to CMS                      MSG00760
         DROP  R12                                                  MSG00770
         EJECT                                                      MSG00780
*****************************************************************  MSG00790
*                                                               *  MSG00800
*  This is our external interrupt exit for the path we've set up *  MSG00810
*  between *MSG and ourself. It works as follows:                *  MSG00820
*                                                               *  MSG00830
*    Connection Complete : Post the CONCOMP ECB and return      *  MSG00840
*                                                               *  MSG00850
*    Incoming message : Receive the message data into our message *  MSG00860
*                       buffer, post the MSGIN ECB and return.  *  MSG00870
*                                                               *  MSG00880
*****************************************************************  MSG00890
         SPACE 2                                                    MSG00900
MSGPATH  EQU  *                                                     MSG00910
         BALR  R12,0              Establish                         MSG00920
         USING *,R12              addressability                    MSG00930
         USING IPARML,R2          R2 points to ext. int. buffer     MSG00940
         CLI   IPTYPE,X'02'       Is it a connection complete ??    MSG00950
         BE    CONNCOMP           If so, it's a special case        MSG00960
         XC    IUCVPLST(40),IUCVPLST  Zero out the IUCV parameter list MSG00970
         XC    MSGAREA(140),MSGAREA   and the message buffer       MSG00980
         MVC   MSGCLASS,IPTRGCLS  Save the class                    MSG00990
         MVC   MSGID,IPMSGID      and the id of the message         MSG01000
         LA    R2,IUCVPLST        Let R2 point to the PRMLIST       MSG01010
*                                                                    MSG01020
*   Get the message which has been sent to you by issuing an         MSG01030
*   IUCV RECEIVE.  Put the message in MSGAREA.                       MSG01040
*                                                                    MSG01050
         IUCV  RECEIVE,PATHID=PATHID,PRMLIST=(R2),MSGID=MSGID,TRGCLS=MS*MSG01060
               GCLASS,BUFFER=MSGAREA,BUFLEN=BUFERLEN                MSG01070
*                                                                    MSG01080
         OI    MSGIN,X'40'        Post the MSGIN ECB                MSG01090
         BR    R14                and return to CMS                 MSG01100
```

```
FILE: MSG      ASSEMBLE A              VM/SP CONVERSATIONAL MONITOR SYSTEM

*                                                                      MSG01110
*  Come here if it's a connection complete interrupt (IPTYPE = X'02')  MSG01120
*                                                                      MSG01130
CONNCOMP EQU   *                                                       MSG01140
         OI    CONCOMP,X'40'      Post the CONCOMP ECB                 MSG01150
         BR    R14                AND RETURN                           MSG01160
*                                                                      MSG01170
*  This is our connect pending exit.  It should never be driven.       MSG01180
*  If it is, we'll just ignore it and return.                          MSG01190
*                                                                      MSG01200
CONPEND  EQU   *                                                       MSG01210
         BR    R14                                                     MSG01220
         EJECT                                                         MSG01230
*********************************************************************** MSG01240
*                                                                     * MSG01250
*               E Q U A T E S and C O N S T A N T S                   * MSG01260
*                                                                     * MSG01270
*********************************************************************** MSG01280
         SPACE 2                                                       MSG01290
         DS    0D                                                      MSG01300
SETMSG   DC    CL8'CP'            Parameter list to tell CP we         MSG01310
         DC    CL8'SET'              want our message via IUCV         MSG01320
         DC    CL8'MSG'                                               MSG01330
         DC    CL8'IUCV'                                              MSG01340
         DC    8X'FF'                                                 MSG01350
SETON    DC    CL8'CP'            Parameter list to tell CP we         MSG01360
         DC    CL8'SET'              want our message set back         MSG01370
         DC    CL8'MSG'              to 'ON'.                         MSG01380
         DC    CL8'ON'                                                MSG01390
         DC    8X'FF'                                                 MSG01400
IUCVPLST DC    40X'00'            Used for the IUCV parameter list     MSG01410
MSGID    DS    F                  Holds the IUCV message id            MSG01420
MSGCLASS DS    F                  Holds the IUCV message class         MSG01430
STARMSG  DC    CL8'*MSG'          The name of the IUCV Message Service MSG01440
SAM      DC    CL8'SAM'           Name which CMS will know us by       MSG01450
CONCOMP  DC    F'0'               Connection Complete ECB              MSG01460
MSGIN    DC    F'0'               Message has arrived ECB              MSG01470
PATHID   DC    H'0'               Holds the IUCV path id               MSG01480
BUFERLEN DC    H'140'             Says our buffer will be 140 chars    MSG01490
MSGAREA  DC    CL140'             Message buffer area                  MSG01500
DISABLE  DC    X'00'              Byte to disable us for interrupts    MSG01510
         REGEQU                                                        MSG01520
         NUCON                                                         MSG01530
         COPY  IPARML                                                  MSG01540
         END                                                           MSG01550
```


SESSION REPORT

SHARE

61	B638	CMS ARCHITECTURE AND INTERACTIVE COMPUTING	300
SHARE NO.	SESSION NO.	SESSION TITLE	ATTENDANCE

CMS	LARRY GRAZIOSE	BAM
PROJECT	SESSION CHAIRMAN	INST. CODE

BANK OF AMERICA, 1455 Market Street, San Francisco, CA 94103 (415) 622-1881

SESSION CHAIRMAN'S COMPANY, ADDRESS, and PHONE NUMBER

CMS ARCHITECTURE AND INTERACTIVE COMPUTING

Charles Daney
Senior Scientist
Tymshare, Inc.
20705 Valley Green Drive
Cupertino, California
Installation Code: TYM

August 22, 1983
CMS Project
Session B638

This paper contains the personal views of the author and not necessarily those of Tymshare, Inc.

1/c/pa1/1