146

VM

October 1998

In this issue

update

VM Update

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

North American office
Xephon/QNA
1301 West Highway 407, Suite 201-405
Lewisville, TX 75077-2150
USA
Telephone: 940 455 7050

Editorial panel
Articles published in *VM Update* are reviewed by our panel of experts. Members of the panel include John Illingworth (UK), Reinhard Meyer (Germany), Philippe Taymans (Belgium), Romney White (USA), Martin Wicks (UK), and Jim Vincent (USA).

Subscriptions and back-issues
A year's subscription to *VM Update*, comprising twelve monthly issues, costs £175.00 in the UK; $265.00 in the USA and Canada; £181.00 in Europe; £187.00 in Australasia and Japan; and £185.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1990 issue, are available separately to subscribers for £14.50 ($22.50) each including postage.

Editor
Robert Burgess

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.

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

Contributions
Articles published in *VM Update* are paid for at the rate of £170 ($250) per 1000 words for original material. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our *Notes for Contributors*.

Printed in England.

VM/ESA Version 2 Release 3.0

OVERVIEW

VM/ESA Version 2 Release 3.0 became available at the end of March 1998. With an integrated TCP/IP communications suite, Network File System (NFS) access to VM files, Java Virtual Machine capability, and Message Queuing Interface (MQI), VM/ESA's role as a platform for business solution deployment in networked environments is strengthened further. This article is my personal view of the most interesting highlights from a technical standpoint. Formal details are contained in the IBM European Announcement Letter, ZP98-0203. Other useful information about the announcement can be found by visiting the VM home page on the World Wide Web at http://www.vm.ibm.com.

Building on the previous release, VM/ESA Version 2 Release 3.0 provides a Year 2000-ready base for applications. In addition to expanded network computing support, the new release also provides:

- Enhanced TCP/IP for VM/ESA FL310, including NFS server support for CMS mini-disk, Shared File System, and Byte File System files.
- Additional OpenEdition Unix programming interfaces.
- Java Virtual Machine, JDK 1.1.1, and NetREXX.
- Message Queuing client interface.
- Integration of CMS Pipelines PRPQ.
- Extensions to support for Year 2000.
- Full integrated IBM Language Environment (LE).
- OS/390 guest Parallel Sysplex Testing support.

The basic procedures for installing and servicing VM/ESA Version 2 Release 3.0 have not changed. However, the contents of the system DDR images have altered as follows:

- The VM/ESA GUI feature is now part of the CMS component of VM/ESA.
- The Language Environment (LE) product is pre-installed on the system DDR.
- The 3800 Printer Image Library Source is pre-installed on the System DDR.
- The NLS features, OSA/SF, and VM Online Library are installed on the System DDR and can be optionally loaded during installation.
- TCP/IP, the TCP/IP NFS feature, and RSCS are pre-installed in a disabled state as part of the base VM/ESA.
- Two new EXECs have been added to automate several manual steps required during installation.

The sections that follow expand on the major functional enhancement areas in this new release of VM/ESA. Much of this information comes from the VM/ESA Version 2 Release 3.0 Announcement Questions and Answers at http://www.vm.ibm.com/vm230/vmv23qas.html.

TCP/IP FOR VM/ESA FL310

TCP/IP has undergone major enhancements and repackaging in VM/ESA Version 2 Release 3.0. Prior to this new release, TCP/IP and the TCP/IP Network File System (NFS) feature were separately orderable through IBM program product number 5735-FAL. Now the TCP/IP base product and the TCP/IP NFS feature are automatically shipped and pre-installed in a disabled state, as optional priced features in the base product DDR. Before enabling the code, the customer must ensure that required feature licences are in place. Other optional features of TCP/IP (for example TCP/IP source code) provided with previous levels of TCP/IP for VM are still available as separately orderable priced features of VM/ESA.

The following section details some of the key enhancements in TCP/IP FL310.

SMTP

Performance of the Simple Mail Transfer Protocol (SMTP) server is greatly improved. Laboratory tests have shown an increase in throughput of up to 3.4-fold for sample configurations. Further TCP/IP performance improvements are described in the VM/ESA Version 2 Release 3.0 Performance Report which can be downloaded from the VM home page on the World Wide Web at http://www.vm.ibm.com/perf/docs/vm230prf.html.

Network File System

In TCP/IP FL310 the NFS server supports the Byte File System and the Shared File System, as well as CMS mini-disks. There are several ways to obtain immediate benefit from this capability – for example, by NFS mounting an SFS directory from a workstation client, one could use PC-based Web publishing tools to place HTML and other Web content into production for immediate delivery by a VM Web server. NFS access to VM data is controlled through a VM user-id and password supplied on the NFS MOUNT command. The user-id is used by the NFS server to determine authorization for SFS files and directories. The corresponding UID and GID are used to determine access permission for BFS files and directories. Depending on a supplied start-up parameter, the NFS server can also allow or deny anonymous access. If anonymous access is allowed, an NFS mount to an SFS directory does not require a user-id if PUBLIC has been granted authority to that directory. Similarly, an NFS mount to a BFS directory does not require a user-id, if the default UID (-1) has permission to access it.

Since all authorization and permission checking is done by the file pool server, the protection is the same as when the files are being accessed from CMS.

Options exist on the MOUNT command to enable an NFS client to request data in the format it needs. The LINES option governs the translation of files between CMS record formats and the byte stream format used by NFS, and can be automated conditionally – based on the file name. Thus, with a single mount, a workstation user can view PROGRAM.EXE without the insertion of line-end characters by the

NFS server, while the file README.TXT can be read with line-end characters inserted so that it displays correctly.

The TRANS option allows the client to control whether EBCDIC/ASCII translation should take place. This too can be made conditional based on the file name. The NLVALUE option allows the client to specify which character(s) should be used to represent line-end.

TN3270E

The TN3270E protocol (RFC 1647) support in TCP/IP FL310 provides for the creation of 3270 printer sessions as well as traditional display sessions. This allows workstation-attached printers to be accessed from VM over a Telnet connection. To the host application, the printer looks like a 3287 printer accepting 3270 data streams. TCP/IP defines a logical 3287 device to represent the printer, invokes an exit to attach the device to the appropriate service machine (normally RSCS), and then handles the data streams that flow between the server and the printer.

Long Fat Networks

TCP/IP extensions for high performance (RFC 1323) have been implemented to improve throughput over networks that have large bandwidth x delay values. A new DATABUFFERLIMITS statement in the TCP/IP configuration file specifies maximum send and receive window sizes that may be allocated for a TCP connection that is using window scaling. The statement has an effect only in networks containing other TCP/IP hosts with support for RFC 1323. Initiation of window scaling can be suppressed by inclusion of the NORFC1323 parameter on the ASSORTEDPARMS statement.

Multiple home addresses

TCP/IP FL310 supports RFC 1122, which allows multiple IP addresses to be associated with a single network link, as well as extensions for virtual hosting. The first home address specified for a link is its primary one.

Here is an example that shows how to code multiple home addresses

and how servers can handle a port on one virtual host or a port on all virtual hosts:

```
HOME
 10.0.0.1 SomeLink ; Primary home address
 10.0.0.2 SomeLink
 10.0.0.3 SomeLink
...
PORT 80 TCP HTTPD ; All virtual hosts
PORT 10.0.0.2 80 TCP HTTPD ; One virtual host
```

CHECKSUM Assist

The System/390 hardware CHECKSUM feature is used (where available on the processor) to compute packet checksums.

Performance, diagnostic, and tracing facilities

Extensive performance instrumentation has been added to allow the performance of TCP/IP applications to be measured (APPLMON CP monitor records).

The VM Control Program is now aware when a logical device is associated with TCP/IP (ie is being used to access the system via Telnet) and provides the Telnet server name and client IP address in appropriate operator messages, command responses, accounting records, and ACI parameter lists.

Trace facilities have been enhanced to allow just selected devices, users, or IP addresses to be traced.

NETSTAT command

The NETSTAT command has been enhanced to display more information and provide greater selectivity. Command operands concerned are:

- ARP – displays ARP cache entries.
- DELARP – deletes ARP cache entries (restricted use).
- LEVEL – displays processor type, VM/ESA, and TCP/IP levels.
- SELECT – limits response output based on specific values, such

as a user-id or client name.

- POOL – includes Xlate and IPRoute information.
- INTERVAL – enhanced to remove single panel display limitation.

New server virtual machines

The following new servers have been included:

- TFTP – Trivial File Transfer Protocol (BFS only)
- BOOTP – Boot Protocol
- DHCP – Dynamic Host Configuration Protocol
- UFT – Unsolicited File Transfer Protocol (RFC 1440).

The first three are intended primarily for use with the IBM Network Station. The latter supports receipt of files sent using the UFT protocol.

With the UFT or Sender-Initiated File Transfer (SIFT) protocol the receiver does not initiate the file transfer. UFT contrasts with other file transfer methods in that the sender of a file does not need an account or other form of registration on the target host system, and the receiving user may require fewer steps to retrieve the file(s) sent. Unlike traditional file transfer, UFT lends itself naturally to background or deferred operation, although it may be carried out immediately, or even interactively. This brings to IP-based networks the ability to send a file that is not a mail attachment to another user, thereby eliminating the need for the file to be processed by a mail user agent. In an NJE network, VM users have historically used SENDFILE as a simple, convenient way to transmit files to other users.

CMS NOTE and SENDFILE

The TCP/IP versions of the CMS NOTE and SENDFILE commands have been integrated into CMS to provide automatic support for TCP/IP electronic mail and file transfer.

Apart from minor additions to the syntax of the CMS NOTE command, several new options appear in the syntax of the CMS SENDFILE

command. Some that apply when the target node is an IP address are:

- SMTP – the file is sent using SMTP without any encoding or MIME headers.
- MIME – the file is sent using SMTP. The file is encoded in Base64 and sent as a single-part MIME data stream.
- UFTSYNC – the file is sent to the remote host's UFT server directly from the user's virtual machine. If no UFT server is available on the remote host, the file will be sent using SMTP instead. The file is sent as a two-part MIME data stream. The first MIME part is a meta-file containing all the UFT information associated with the file (file-id, format, record length, etc). The second MIME part is the file data encoded in Base64 (a MIME-defined encoding format).
- UFTASYNC – SENDFILE will send the file to another virtual machine identified in TCPIP DATA, from where it will be sent to the remote host's UFT server. This virtual machine acts like an agent and becomes the UFT client that connects to the remote UFT server.

Server configuration

Adding and replicating TCP/IP service machines has been simplified considerably through the use of a DTCPARMS file that contains tags (as in a CMS NAMES file) describing each of the TCP/IP servers. Here is a sample entry for an FTP server:

```
:nick.ftp          :type.class
                   :name.FTP Daemon
                   :command.SRVRFTP
                   :runtime.PASCAL
                   :diskwarn.YES
                   :anonymous.NO
                   :ESM_Enable.NO
                   :ESM_Validate.RPIVAL
                   :ESM_Racroute.RPIUCMS
```

ADDITIONAL UNIX INTERFACES

165 additional APIs help simplify the porting of Unix applications to VM. The APIs added in this release, a subset of the XPG4.2

(X/Open Portability Guide) specification, were chosen because of requirements submitted by customers who are porting Unix applications and tools to VM and through analysis of the functions most often needed by applications ported to other IBM platforms. The new APIs include interfaces for inter-process communications, mathematical functions, and extra signal support.

Many older Unix programs use the FORK() function, which has not been available in VM/ESA. VM/ESA Version 2 Release 1 provided the SPAWN() function as a replacement for the combined use of the FORK() and EXEC() functions to create a new process for running a new program. Programmers porting Unix applications to VM have found that replacing FORK() calls with the appropriate supported functions is mostly trivial. Nevertheless, they would also find it helpful to be able to try the application on VM unchanged before undertaking a conversion. The FORK() function in VM/ESA Version 2 Release 3.0 will allow most applications that use FORK() to run on VM while they are being ported. Applications can use the FORK() function on VM/ESA in cases where an EXEC() function invocation follows the FORK(). This accommodation of the FORK() function is a selectable option and enables a more rapid port of many Unix applications to VM/ESA. The CMS command OPENVM SET FORK ON must be issued before running any application that uses the FORK() function. Once the application is successfully ported, performance can be improved by converting the FORK() calls to SPAWN() or some other appropriate replacement.

JAVA VIRTUAL MACHINE, JDK 1.1.1, AND NETREXX

Java is the language of the Internet because of its object code portability. A Java program compiled on any system can execute unchanged on any other platform with a Java Virtual Machine.

It is interesting that the concept of a virtual machine, implemented in IBM's VM/ESA product, should become such a vital element of network computing enablement.

The Java Virtual Machine (JVM) defines a virtual processor architecture with its own instruction set, just as VM/ESA defines a virtual

System/390 processor with a System/390 instruction set. The main difference is that the JVM is completely platform neutral. Provision of a JVM for VM/ESA means that Java applications can execute unchanged on VM/ESA, thereby expanding significantly the portfolio of applications available to VM users.

Many programmers, whose background is solely in high-level procedural languages such as COBOL, FORTRAN, PL/I, etc, may find the object-oriented Java language difficult to learn. However, most VM programmers make heavy use of the REXX language to produce applications on CMS. The NetREXX language, a derivative of REXX, allows programmers to write Java applications in a REXX-like language. The NetREXX reference compiler generates Java source code for subsequent (automatic) compilation by the Java compiler into Java byte-codes. In fact, many existing REXX programs are themselves valid NetREXX programs.

VM/ESA V2 R3.0 Java support is at the Java Developer's Kit (JDK) Level 1.1.1. The support includes:

- Java compiler
- Debugger
- The Java Virtual Machine (Java byte-code interpreter)
- The JDK 1.1.1 class libraries.

The VM/ESA support for Java does not include a Just-In-Time (JIT) compiler or use of the Abstract Windowing Toolkit (AWT) and Java DataBase Connectivity (JDBC) classes for execution (you can, however, compile Java programs that use these classes on VM/ESA). One VM programmer has already reported successful use of Remote AWT (RAWT) components to enable a Java application using AWT methods that is executing on VM to drive a graphical windowing display on a programmable workstation. In practice, the absence of AWT run-time support is frequently immaterial because many Java applications have a client/server structure and AWT classes would only be used by the client component running on a PC or IBM Network Station for animation and graphical presentation. The Beta 2 level code for Java includes full use of the Java Native Interface (JNI).

An IBM International Technical Support Organization (ITSO) redbook about Java and NetREXX support in VM/ESA is currently undergoing final editing. It can be browsed in 'red pieces' format at http://www.redbooks.ibm.com.

MESSAGE QUEUING CLIENT INTERFACE

The Message Queuing (MQ) Interface is a popular network computing technology that enables applications on different systems and architectures to work together. The MQ application-to-application communication protocol is simple yet powerful, using a queue manager (MQ server) somewhere in the enterprise to allow applications to communicate asynchronously without requiring the requester (client) and server to be connected.

VM/ESA Version 2 Release 3.0 support for MQ enables client or server applications on the network to connect to applications on VM that have implemented the MQ communication protocol. Use of the MQ APIs on VM requires the deployment of an MQ server somewhere in the network.

Using MQSeries clients is an efficient way of implementing MQSeries messaging and queuing. An application can use the Message Queue Interface (MQI) running on one machine with the queue manager running on a different machine. In this model there is no need for a full MQSeries implementation on the client machine, system administration requirements are reduced, and an MQSeries application running on a client can connect to multiple queue managers on different systems.

The VM/ESA MQSeries client will connect to all nineteen IBM and non-IBM platforms that currently support an MQSeries Server installation. Programming language bindings for the MQSeries client include C, COBOL, PL/I, REXX, and Assembler.

For further information, visit the MQSeries product family home page at http://www.software.ibm.com/ts/mqseries.

INTEGRATION OF CMS PIPELINES PRPQ

VM/ESA Version 2 Release 3.0 now incorporates the functionally

richer Pipelines PRPQ as a standard part of the system. New support includes toleration for Pipelines program execution in a multi-tasking environment, TCP stages, and documentation of more Pipelines stages, including the popular Author's Help (AHELP).

Effectively, the VM/ESA ('Endicott') version of CMS Pipelines is merged with the Runtime Distribution ('Copenhagen') version, also known as CMS/TSO Pipelines.

Several message texts have been revised and over 200 new messages added. Return codes are the Copenhagen ones rather than those associated with the DMS messages. The DMS messages are deleted and a cross reference between the old DMS and new FPL message numbers is provided.

The Copenhagen *CMS/TSO Pipelines: Author's Edition* reference is now published as a VM manual, SL26-0018. Author's Help (AHELP) is also documented and supported.

A new CONFIGURE stage and a 'style' concept have been added to let a programmer control any differences between the two Pipelines versions. To determine how a pipeline should behave in other instances where there traditionally used to be a difference between the Copenhagen and the Endicott versions, defaults are now maintained, derived from variables set in the GLOBALV group FPL (by default).

Toleration support for CMS multi-tasking means that a pipeline now supports recursions on different processes and threads. However, a pipeline has no concept of the 'current thread' – pipeline sets run completely in parallel in their respective threads. How control passes between threads is of no concern to the pipeline.

A new stage named DATECONVERT will perform date conversion and validation.

The TCP stages from the Runtime Distribution version can be used to program TCP/IP sockets server applications. For example, TCPLISTEN listens on a TCP port. It listens for and accepts connection requests using the socket interface to TCP/IP. The socket representing the connection is then transferred to a TCPDATA stage, which exchanges data with the client.

EXTENSIONS TO YEAR 2000 SUPPORT

Besides the DATECONVERT Pipelines stage, additional Year 2000 conversion functions in VM/ESA Version 2 Release 3.0 include a utility to correct the century indicators for CMS files and 4-digit year support in the user NETLOG file.

DATECONVERT is easy to use and provides date conversion, validation, and windowing functions. It allows a programmer to rapidly convert and validate large amounts of date data. DATECONVERT supports all the REXX date formats, plus additional formats. Internally, it uses the DateTimeSubtract CSL routine, but provides a simpler interface for date format conversions.

The DateTimeSubtract routine (DMSDTS) has been updated in the new release of VM/ESA. See the 'CSL Considerations' section of the Runtime Library Distribution page at http://pucc.princeton.edu/~pipeline/ for notes on invoking the correct level of DMSDTS.

Two new commands are provided with VM/ESA Version 2 Release 3.0 that set the internal century information stored with mini-disk or SFS files. The commands are FIXCENT for mini-disks, and FILESERV FIXCENT for SFS file pools. These commands set the century according to a sliding window of '-50,+49' applied to the 2-digit year that is stored with the file information.

Setting the century information is not needed as part of a normal migration procedure, since all data that is created on a non-Year 2000-ready release of VM/ESA, when IPLed on a Year 2000-ready release of VM/ESA, will default to 19xx. Setting the century is only needed if, after testing, some files have 4-digit years that appear as 20xx but should be 19xx. The century will be reset according to the sliding window described above.

Although IBM provides this facility for migration and test, it does not recommend using production data and files when testing with a system date set in the Year 2000 or later, or sharing data and files between production systems and Year 2000 test systems. Data and files which are used for Year 2000 testing should be isolated from any production environment.

Beginning with VM/ESA Version 2 Release 3.0, when a new NETLOG file is created, entries written to the file will contain dates in ISODATE format (yyyy-mm-dd). NETLCNVT will convert the dates in an existing NETLOG file from 2-digit years to 4-digit years and *vice versa*.

INTEGRATED LANGUAGE ENVIRONMENT

In VM/ESA Version 2 Release 3.0, IBM has extended the 'standard' package to include the entire Language Environment. LE run-time libraries are needed by many IBM and non-IBM applications and subsystems; in the past any VM customer who wanted to use those products had to purchase LE as well. With the inclusion of LE in the base of VM, this additional purchase will no longer be necessary.

In addition to the Common Execution Library (CEL) support provided for OpenEdition VM/ESA in previous Version 2 releases, the run-time requirement for VM compilers such as IBM C for VM/ESA, IBM COBOL for MVS and VM, and IBM PL/I for MVS and VM is now satisfied.

OS/390 GUEST PARALLEL SYSPLEX TESTING SUPPORT

Many VM installations exploit VM as a hipervisor for pre-production testing of MVS and OS/390 systems. As MVS and OS/390 have moved into the coupled sysplex environment, the demand has grown to provide a similar capability under VM. With VM/ESA Version 2 Release 3.0, it is now possible to simulate coupled sysplexes within a single VM system image. This capability is dependent on IBM's Coupling Facility Control Code (CFCC), which is Licensed Internal Code (LIC) on the IBM Multiprise 2000 Server and the IBM Parallel Enterprise Server Generation 3 and Generation 4. These ranges of machines (at the proper engineering change levels) are supported for guest parallel sysplex testing under VM/ESA.

The CFCC runs in a special Coupling Facility Virtual Machine (CFVM). No other special hardware is required. Coupling links (CF channels) and external coupling facilities are neither required nor supported by VM.

CP simulates the message facility used in a real parallel sysplex. Virtual message devices are employed to transfer data and commands between the CF service machine(s) and the coupled guest virtual machines. They are defined in a SPECIAL Directory Control statement for a virtual machine participating in a simulated sysplex (ie with the CFUSER parameter on the OPTION Directory Control statement).

VM/ESA guest coupling simulation does not support or simulate sysplex timers (ETR). When running in a VM/ESA environment, all virtual machine TOD clocks are synchronized with the system TOD clock. This implicit synchronization allows all MVS images running in a sysplex on VM/ESA to have identically set TOD clocks without the use of an ETR.

A simulated sysplex environment can even run with a date and time that is different from the system date and time. The CP SET VTOD command makes it possible to perform Year 2000 testing of a parallel sysplex environment running second level on a VM/ESA production system without having to IPL the production system with a Year 2000 date.

VM allows the definition of a simulated coupled sysplex up to the architectural maximum of 32 coupled guests. Each coupled guest can be connected to a maximum of eight CFVMs.

OTHER CMS ENHANCEMENTS

Apart from CMS enhancements discussed earlier, there are other minor changes.

The EXECUTE macro has been rewritten as a compiled REXX program. This will provide performance improvements when EXECUTE is issued with the CMS FILELIST and RDRLIST commands. More EXECs have been moved into the CMSINST segment. The VM Graphical User Interface (VM GUI) now supports 32-bit workstation agents for Windows 95 and Windows NT. It also provides an option to eliminate the initial security prompt for authorized host user-ids by keeping the authorization list on the workstation.

SUMMARY

The focus of VM/ESA Version 2 Release 3.0 is chiefly on support of the network computing model and e-business enablement. With major enhancements to TCP/IP, extended NFS support, Java and NetREXX, OpenEdition APIs, and an MQSeries client, plus excellent Web serving solutions from IBM business partners, VM is well positioned to maintain its role as a flexible, efficient, functionally rich, high-performance server platform.

Cliff Laking
IT Technical Specialist
IBM United Kingdom Ltd (UK)

Administering multiple machines – part 2

This month we continue the collection of REXX procedures written to help administer and control multiple machines in an installation.

```
/* ================================================================== */
/* main: event wait loop                                              */
/* ================================================================== */
do forever

  /* Wait for an event                                                */
  event = Wait('Socket Read *','Cons','Mail all')
  if wrc¬=Ø then call error 'E', 36, 'WAIT() rc='wrc
  parse upper var event event rest
  select

    /* Leave program                                                  */
    when event='CONS' then do
      if rest='EXIT' then leave
      else say 'Enter "EXIT" to leave'
    end
    when event = 'MAIL'
      then
         call handle_reader_file rest  /* rest = spoolid */

    /* Accept connections from clients, receive and send messages     */
    when event='SOCKET' then do
```

```
      parse var rest keyword ts .

      /* Accept new connections from clients                              */
      if keyword='READ' & ts=s then do
        nsn = Socket('Accept',s)
        if src=Ø then do
          parse var nsn ns . np nia .
             say 'EventServer: Connected by' nia 'on port' np 'and
socket' ns
          parse value Socket('Recv',ns) with len event_string
          if nia /= vmhost then event_string=ac2ec(event_string)
          call Socket 'Close',ns
          call process_event
        end
      end
    end
    /* Unknown event (should not occur)                                 */
    otherwise nop
  end
end

/* Terminate and exit                                                   */
call Socket 'Terminate'
say 'EventServer: Terminated'
exit Ø

/* ================================================================== */
/* socket subroutines                                                 */
/* ================================================================== */
/* Calling the real SOCKET function                                   */
socket: procedure expose initialized src
  aØ = arg(1)
  a1 = arg(2)
  a2 = arg(3)
  a3 = arg(4)
  a4 = arg(5)
  a5 = arg(6)
  a6 = arg(7)
  a7 = arg(8)
  a8 = arg(9)
  a9 = arg(1Ø)
  parse value 'SOCKET'(aØ,a1,a2,a3,a4,a5,a6,a7,a8,a9) with src res
return res

/* Calling the real WAIT function                                     */
wait: procedure expose initialized wrc
  aØ = arg(1)
  a1 = arg(2)
  a2 = arg(3)
```

```
  a3 = arg(4)
  a4 = arg(5)
  a5 = arg(6)
  a6 = arg(7)
  a7 = arg(8)
  a8 = arg(9)
  a9 = arg(10)
  parse value 'WAIT'(a0,a1,a2,a3,a4,a5,a6,a7,a8,a9) with wrc res
return res

/* Calling the real SETVALUE function                                */
setvalue: procedure expose initialized wrc
  a0 = arg(1)
  parse value 'SETVALUE'(a0) with wrc res
return res

/* Calling the real QUERYVALUE function                              */
queryvalue: procedure expose initialized wrc
  a0 = arg(1)
  parse value 'QUERYVALUE'(a0) with wrc res
return res

/* Calling the real RESETVALUE function                              */
resetvalue: procedure expose initialized wrc
  a0 = arg(1)
  parse value 'RESETVALUE'(a0) with wrc res
return res

/* Syntax error routine                                               */
syntax:
  call error 'E', rc, '==> REXX Error No.' 20000+rc
return

/* Error message and exit routine                                     */
error: procedure expose ecpref ecname initialized
  type = arg(1)
  retc = arg(2)
  text = arg(3)
  ecretc = right(retc,3,'0')
  ectype = translate(type)
  ecfull = ecpref || ecname || ecretc || ectype
  address command 'EXECIO 1 EMSG (CASE M STRING' ecfull text
  if type¬='E' then return
  if initialized then do
    parse value Socket('SocketSetStatus') with . status severreason
    if status¬='Connected' then do
      say 'The status of the socket set is' status severreason
    end
    call Socket 'Terminate'
```

```
  end
exit retc

/* ================================================================== */
/* handle reader file event                                           */
/* ================================================================== */
handle_reader_file:
arg spoolid

/* say 'reading spoolfile' spoolid                                    */
'pipe reader file' spoolid '| stem inrecs.'
event_string=''
do ix02=2 to inrecs.0  /* ignore first record in reader file */
   event_string=event_string || strip(substr(inrecs.ix02,2))
end
call process_event
return

/* ================================================================== */
/* process_event                                                      */
/*                                                                    */
/* event_string contains the complete event string sent by client     */
/* ================================================================== */
process_event:

interpret es02(event_string)
say 'Event_received:' es.id es.origin es.timestamp

if es.id = 'RESETLOG'
then
   do
     'pipe <' logfile 'a | take first 1 | >' logfile 'a'
     say 'Logfile was reset'
     return
   end

/*  get last event number and increment it                            */
'pipe fileback' logfile '| take first 1 | var lastevent'
number=right(word(lastevent,1)+1,14,'0')

/* write log                                                          */
id=es.id
origin=es.origin
timestamp=es.timestamp
eventstring=event_string
logrec=left(number,15) || ,
       left(id,20) || ,
       left(origin,20) || ,
       left(timestamp,20) || ,
```

```
       left(eventstring,400)
'pipe var logrec | >>' logfile

/* attach number to event_string                                        */
dl=left(event_string,1) /* the delimiter */
event_string=event_string || 'NUMBER' number || dl
eventdef=eS17('eventdef.'id)

if eventdef = ''
then
   do
     say id 'is an undefined event id'
     return
   end

parse value translate(eventdef) with . 'NOTIFY(' notify_users ')' .
parse value translate(eventdef) with . 'ACTION(' call_procs ')' .

do ix05=1 to words(notify_users)
   uid=word(notify_users,ix05)
   'cp msg' uid 'Event' id 'from' origin 'at' timestamp ,
           'Number:'strip(number,'L','0')
end

do ix05=1 to words(call_procs)
   call_proc=word(call_procs,ix05)
   'pipe cms listfile' call_proc 'exec k | stem inx.'
   if inx.0 > 0
   then
      interpret 'call' call_proc  'event_string'
   else
      say 'Procedure' call_proc 'not found'
end
return
```

ES06 EXEC

```
#|/usr/bin/rexx
/* ================================================================== */
/* Name        :  ES06 EXEC                                           */
/* ================================================================== */
/* Application :  Event Services                                      */
/*                                                                    */
/* Usage       :  REXX/6000 Procedure                                 */
/*                                                                    */
/* Arguments   :  id, tag contents, tag contents, ...                 */
/*                                                                    */
/* Result      :  -                                                   */
```

```
/*                                                                    */
/* Function    :  Send an event  (AIX Version)                        */
/*                                                                    */
/* ================================================================== */
hostname='hostname'
if strip(hostname) = '' then hostname='WHOSIS?'
origin='ORIGIN' hostname
parse value date(sorted) with yyyy 5 mm 7 dd
isodate= yyyy'-'mm'-'dd
timestamp='TIMESTAMP' isodate'/'time()
dl='#'
id='ID' translate(arg(1))
es=dl || id || dl || origin || dl || timestamp || dl
do i=2 to arg()
   es=es || arg(i) || dl
end

'ESØ7 "'es'"'
return
```

ES07 EXEC

```
#|/usr/lpp/ssp/perl5/bin/perl
#*================================================================== */
#* Name        :  ESØ7 EXEC                                          */
#*================================================================== */
#* Application :  Event Services                                     */
#*                                                                   */
#* Usage       :  PERL Program for AIX                               */
#*                                                                   */
#* Arguments   :  event_string                                       */
#*                                                                   */
#* Result      :  -                                                  */
#*                                                                   */
#* Function    :  Send an event  (called by ESØ6)                    */
#*                                                                   */
#* This program sends the event string received as argument          */
#* to the event server in VM via a TCP/IP socket                     */
#*                                                                   */
#*================================================================== */

$eventdata=$ARGV[Ø];
require 'sys/socket.ph';
$AF_INET=2;
$SOCK_STREAM=1;
# get:      IP-Adresse of  VM Host
#           tpc protocol number
($name,$aliases,$type,$len,@addrlist)=gethostbyname('VMHOST');
$address=@addrlist[Ø];
```

```
$proto=getprotobyname('tcp');
$serverport=1952;
# create "packed network address"
$template='S n a4 x8';
$paddr=pack($template,$AF_INET,$serverport,$address);

socket(SOCKET,$AF_INET,$SOCK_STREAM,$proto)
      ||  die "socket fail: $|";
connect(SOCKET,$paddr) ||  die "connect fail: $|";
send(SOCKET,$eventdata,Øx ØØ)
      ||  die "send fail: $|";
close(SOCKET);
exit;
```

ES08 EXEC

```
/* ============================================================================ */
/* Name        :  ESØ8  EXEC                                                    */
/* ============================================================================ */
/* Application :  Event Services                                                */
/*                                                                              */
/* Usage       :  Function                                                      */
/*                                                                              */
/* Arguments   :  -                                                             */
/*                                                                              */
/* Result      :  RecDef of event logfile                                       */
/*                                                                              */
/* Function    :  Get RecDef of event logfile                                   */
/*                                                                              */
/* ============================================================================ */
return ,
'NUMBER,1,15 ID,16,2Ø ORIGIN,36,2Ø TIMESTAMP,56,2Ø EVENTSTRING,76,4ØØ'
```

ES09 EXEC

```
/* ============================================================================ */
/* Name        :  ESØ9  EXEC                                                    */
/* ============================================================================ */
/* Application :  Event Services                                                */
/*                                                                              */
/* Usage       :  Function                                                      */
/*                                                                              */
/* Arguments   :  -                                                             */
/*                                                                              */
/* Result      :  fmode  (on Errors: Nullstring)                                */
/*                                                                              */
/* Function    :  Link/access Event Services Data Disk                          */
/* ============================================================================ */
arg mode
```

```
machine=eS17('event.server')
disk='191'
Vaddr_mode = accdisk(machine,disk,'RR')
if words(vaddr_mode) < 2
then
   do
     sayer 'Link Error:' machine disk vaddr_mode
     return ''
   end

vaddr=word(vaddr_mode,1)
fmode=word(vaddr_mode,2)
return fmode
```

ES10 EXEC

```
/* ================================================================== */
/* Name        :  ES1Ø  EXEC                                          */
/* ================================================================== */
/* Application :  Event Services                                      */
/*                                                                    */
/* Usage       :  Procedure                                           */
/*                                                                    */
/* Arguments   :  EventNumber                                         */
/*                                                                    */
/* Result      :  Event Log Record                                    */
/*                                                                    */
/* Function    :  Get Event Log Record by Event Number                */
/*                                                                    */
/* ================================================================== */
arg number

number=right(strip(number),14,'Ø')
fm=esØ9()  /* access the disk */
logfile=eS17('event.logfile')
recdef=esØ8()
fromto='1-15'
'pipe <' logfile fm '| locate' fromto '/'number'/ | stem found.'
if  found.Ø > Ø
then
   retval=found.1
else
   retval=''

call diskrel fm
return retval
```

Editor's note: this article will be continued next month.

Mouse on the mainframe

Traditional 3270-based applications are usually designed to accept text which is typed into certain screen locations, and perhaps to respond when PF keys are pressed. Users require adequate keyboarding skills, along with a knack for remembering (often complex) command syntax.

The programs described here were designed to be manipulated with a workstation mouse and consequently demonstrate an alternative approach to 3270 application design. These 'pointer enabled tools' exploit a simple synergy between 3270 emulation software and host-based applications. The resulting interface is fast, intuitive, and highly functional. Users type less, make fewer errors, and as a result are more productive.

BACKGROUND – TO THE MAC AND BACK

In 1989 I swapped my monochrome green IBM 3178 terminal for a Macintosh IIfx and I suddenly understood the advantages of a mouse-clickable GUI interface. Within a year or so, I had abandoned VM/CMS as a personal computing platform, except for e-mail (PROFS and MailBook). I learned MS Excel, Aldus Persuasion, Superpaint – what wonderful tools!

However, around 1992 (when mainframes were dying, remember?) I realized that:

- It was taking me longer to complete text-only tasks.
- It was awkward to share such files with co-workers who used PCs, Unix workstations, and 3270 terminals.
- I could not work from home without borrowing a Mac (my home computer was a PS/2).
- I could not easily check on our MVS system.
- Apart from HyperCard, I had no programming tools and when my Mac was down (which admittedly was infrequently) I could not work at all (at least not on *my* files).

For many tasks, I preferred the Macintosh's software, but for a lot of routine chores, using the small system was simply inefficient. I found myself drifting back to CMS.

At the same time, despite the proliferation of microcomputers on campus, thousands of our students, faculty, and staff continued to use VM/CMS commands and the traditional 3270 interface for e-mail and other tools such as MINITAB, SAS, and FOCUS.

I wanted the best of both worlds. So I asked my staff why we couldn't have a more user-friendly mainframe.

Because my questions generally went unanswered – and because I wanted to learn how to program in REXX – I started playing around with some (in hindsight) rather awkward menuing programs. Menus led to XEDIT macros and to CMS windows and so on. Then, in 1996, I stumbled across the fact that some of what I had written could be manipulated by positioning the 3270 cursor with a workstation mouse and clicking with the right mouse button. Pure chance. The serendipitous convergence of:

- My interest in making VM/CMS easier to use.
- REXX code written to respond to the position of the cursor when the ENTER key is pressed.
- 3270 terminal emulation software which by default configured the right mouse button action in a certain way.
- Bumbling fingers which pressed the wrong mouse button!

With tongue in cheek, I started calling these programs 'Pointer Enabled Tools' or 'PETs' and the name stuck.

EXAMPLES OF MOUSE-CLICKABLE SOFTWARE

Excuse me? Did you say mouse-clickable software?

The QCMDS application presents a list of commands (actually, descriptions of commands) in a CMS window. This is shown in Figure 1. By using arrow keys, you can reposition the 3270 cursor on any menu item, press the ENTER key, and the corresponding command

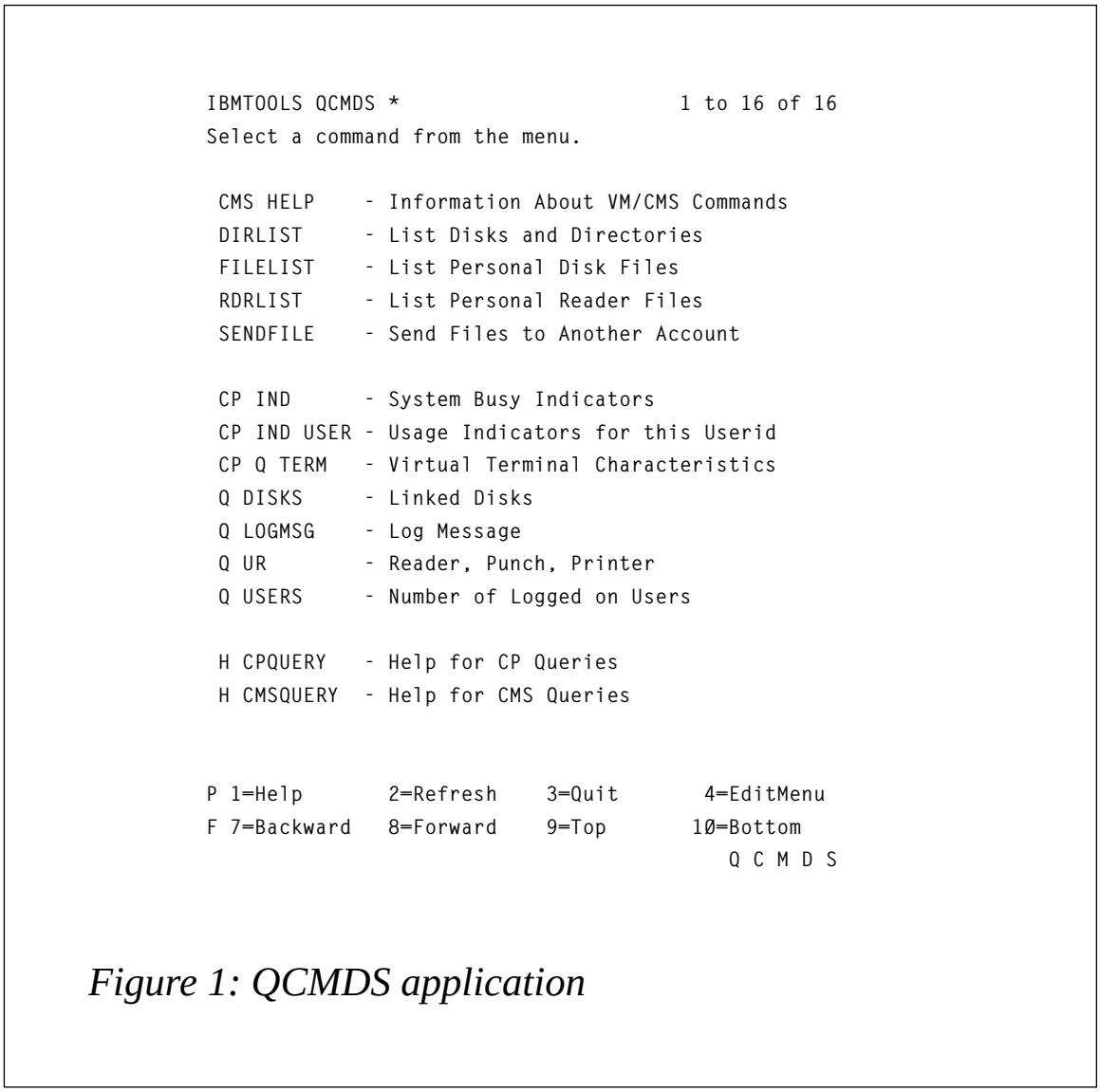

Figure 1: QCMDS application

is executed. Press a PF key, as labelled on the two lines at the bottom of the window, and QCMDS responds properly. Simple enough?

Now, if the mouse action associated with clicking your right mouse button is set properly, you can point to an item on the QCMDS menu with the workstation pointer and select that item by clicking with the right mouse button. In the case of QCMDS, you can also ‘point and click’ on the PF key labels to invoke the functions associated with those PF keys, and you can ‘point and click’ outside the QCMDS window to close that window.

The PFLIST application, which displays a list of disk files in its primary window, is a mouse-clickable alternative to FILELIST. This is shown in Figure 2.

```
Files: * EXEC A              4 Jun 1998 20:43:50              RGE AT UCONNVM
Select a file.                                                 1 to 19 of 19
    Filename Filetype FM Format Lrecl    Records     Blocks  Date      Time
    CHG2CLF  EXEC     A1 V         72         39          1  1/31/1998 10:55:48
    CHG2CMF  EXEC     A1 V         69         17          1 10/27/1997 12:54:18
    CISP     EXEC     A1 V         72         48          1 11/27/1996 17:25:35
    CLEANPRF EXEC     A1 V         73         65          1 10/09/1997 16:41:31
    COPY95   EXEC     A1 V         36        259          3 10/09/1997 16:41:39
    ERASE95  EXEC     A1 V         29        259          2 10/09/1997 16:41:39
    HELPDESK EXEC     A1 V         25          8          1  2/25/1998 11:42:27
    HERE     EXEC     A0 V         56         78          1 11/20/1997  8:22:03
    NOTELIST EXEC     A1 V         73        208          2  5/12/1997 14:57:14
    PCLICK   EXEC     A1 V         80         76          2  5/30/1998  8:59:16
    PGROUP   EXEC     A1 V         63         11          1  3/06/1998 16:49:15
    PROFILE  EXEC     A1 V         51         29          1  5/14/1998 17:10:53
    PRTX2PPR EXEC     A1 V         44         15          1 10/16/1997  5:36:07
    SHELP    EXEC     A1 V         51         21          1 12/31/1996 12:47:01
    SUPPM    EXEC     A1 V         71         43          1 11/07/1996 18:47:26
    TESTSQL  EXEC     A1 F         80         24          1 11/25/1996 13:30:35
    TIMEOFF  EXEC     A1 V         41          2          1  7/08/1996 13:52:48
    TRYC     EXEC     A1 F         80         25          1  9/25/1997 17:50:39
    Y2KDISK  EXEC     A2 V         71        138          2  4/25/1997 16:23:57


P 1=Help       2=Refresh    3=Quit       4=SortName   5=SortType   6=SortDate
F 7=Backward   8=Forward    9=Top       10=Bottom    11=View      12=CursHome
==>
Disks: * A B C D E F G H I J K L M O S Y                          P F L I S T
```

Figure 2: PFLIST application

Imagine the following:

- You can open a file to XEDIT by clicking on the filename.
- You can sort the list of files by clicking on any of the column headers (ie filename, filetype, fm format, etc).
- You can activate the functions associated with the PF keys by clicking on the PF key labels toward the bottom of the screen.
- You can view a list of files on another accessed disk (or directory)

by clicking on one of the active filemodes shown on the bottom line:

```
Disks: * A B C D E F G H I J K L M O S Y
```

Write the CMS code to respond to the position of the cursor when the ENTER key is pressed, configure your 3270 emulation software to 'set the 3270 cursor' and 'press ENTER', and there you have it – mouse-controllable software on CMS.

USE MY MOUSE ON THE MAINFRAME? IT CAN'T BE DONE!

Some might consider that using a PC or Unix workstation mouse to control CMS software is something close to heresy!

In presenting the ideas behind PETs, I generally run into two very different reactions. When I demonstrate the tools, people are surprised to learn that it can be done at all. They are pleased with the ease of use and the response time. I've heard the comment, "Wow!", on more than occasion.

But when I try to describe what I'm doing over the telephone, I generally run into what seems to be a psychological barrier. The words '3270' and 'mouse' not only don't go together in a sentence, they don't even belong in the same conceptual universe – until now that is.

At the beginning of this project, I realized that there were a couple of fundamental issues that we must deal with. Firstly, 3270 traditionalists have come to believe that pressing the ENTER key is usually a pretty harmless act – and often quite helpful. Press ENTER and the cursor returns to the command line, for example. With PETs, pressing the ENTER key invokes one of many possible actions, determined specifically by the location of the cursor when the deed is done. Pressing ENTER is *not* a 'soft reset' any more.

Secondly, we 3270 traditionalists place our hands on the keyboard, not on the mouse, when the green screen pops up. Long-standing habits make it difficult for us to 'go for the mouse' as the first option, the way we do when using MS Word or Netscape Navigator. To get the most from PETs, we have to reach for the mouse.

Thirdly, having just a few pointer-enabled tools won't do.

Psychologically it is awkward and frustrating to have to switch between a 'primarily typing' interface and a 'primarily mouse-controlled' interface. For PETs to catch on, a critical mass of useful software, which shares a common look and feel, would have to be developed. A critical mass of pointer-enabled software exists only when you can spend most of your day in mouse mode.

Finally, the design of these tools requires a rethink in the way in which users interact with the applications. Traditionally, 3270 screens accept typed data and/or commands; PF keys are often defined as shortcuts for a dozen or so functions. And that's about it. With PETs, any area, line, column, or even a single character on the screen becomes a potential control. Taking this to the extreme, a single (80 x 24) 3270 screen can contain 1,920 clickable control points!

A BLEND OF TWO INTERFACE STYLES

PETs are intended to serve two kinds of users:

- Those who like using a mouse (or other pointing device).
- Those who like using traditional 3270 keyboard functions (commands, PF keys, arrow keys, the ENTER key).

While some usability enhancements are obviously intended for use with a mouse, all functions are available as well on a dumb terminal. In a very real sense, the PETs style interface blends these two approaches together into a fast, non-graphical package that can enhance the productivity of all users.

ENABLING TECHNOLOGIES

PETs are written in REXX and make extensive use of CMS virtual screens and windows as well as CMS Pipelines and CP/CMS commands. The XEDIT macros, of course, make use of XEDIT subcommands and XEDIT internal variables. A few PETs are designed as 'front-ends' to other applications (eg IBM's OfficeVision or Script/VS, or L-Soft's Listserv), so naturally those other applications are required if the 'front-end' PETs are to be useful.

Some of the file management tools use the FULLDATE option of the

LISTFILE command, so CMS must be at a level to support this function (but since we're all getting ready for Year 2000, this should not be a problem for long!).

While PETs can be used as traditional 3270, keyboard-reliant applications from fixed-function 3270 terminals, the special advantages of PETs are unavailable if one does not use a 'PETs-friendly' 3270 emulator running on a PC, Macintosh, or Unix workstation. To be 'PETs-friendly', a 3270 emulator must allow some mouse action to set the 3270 cursor to any location on the screen and then to emulate pressing the ENTER key. I find it most convenient if a single click of one mouse button (the right mouse button on my PC) performs both these functions, but variations exist. Some emulators implement the required functions with two clicks of the left mouse button, by default. Other emulators require that the 3270 commands be configured into a macro, which is then assigned to the mouse action. Unfortunately, some emulators do not appear to provide a way to properly configure the mouse action at all.

GENERAL DESIGN CHARACTERISTICS

PETs programs derive from precursor attempts to make VM/CMS more 'user-friendly' and they do make working on the mainframe considerably easier. Programs incorporate a number of key ideas:

- Considerably less typing is required.
- Mouse-clicking for navigation and functional selection is enabled.
- CMS windows are used extensively.
- Menus are a key component.
- Menus and colours are easy to customize on-the-fly.
- All commands can be selected from menus or entered on command lines.
- Significant usability enhancements are added.
- The interface is fast.
- Colours and PF keys are used consistently.

- Windows have a consistent (and familiar) look and feel.
- Activity logs can record what you do so that you can do it again more quickly.
- Many functions are generic and can be re-used in new applications.
- Programs are completely compatible with other VM/CMS tools – PETs menus can include non-PETs applications.
- A critical mass of software is now available to enable users to do most of their CMS work from within PETs.

In general, I try to follow several usability rules:

- It is better to click than to type.
- It is better to click once than to click more than once.
- It is better to be simple than to be cryptic.
- It is better to rely on screen cues and human intuition than on documentation.

PETSHELL – A STARTING POINT FOR ALL CMS WORK

A GUI or non-GUI desktop offers a starting point for one's computing efforts. Icons and other screen cues invite certain actions which lead to programs and files. Human operators need to *recognize* what they see rather than *recall* commands and file names, and it is easier to do the former than to do the latter. So a visually helpful starting point is comforting to most people.

The PETSHELL EXEC offers such a starting point (although there are others). PETSHELL displays information about the current session, a number of clickable, colour-coded 'hotspots' which invoke useful tools or display additional menus, and a command line which accepts any CP or CMS command or EXEC. A command 'logging space' in the centre of the window retains the commands which have been entered on the PETSHELL command line. In the example shown in Figure 3 the PETSHELL window displays the lines:

```
XEDIT PROFILE EXEC
Ready(Ø)
QUERY RGE
```

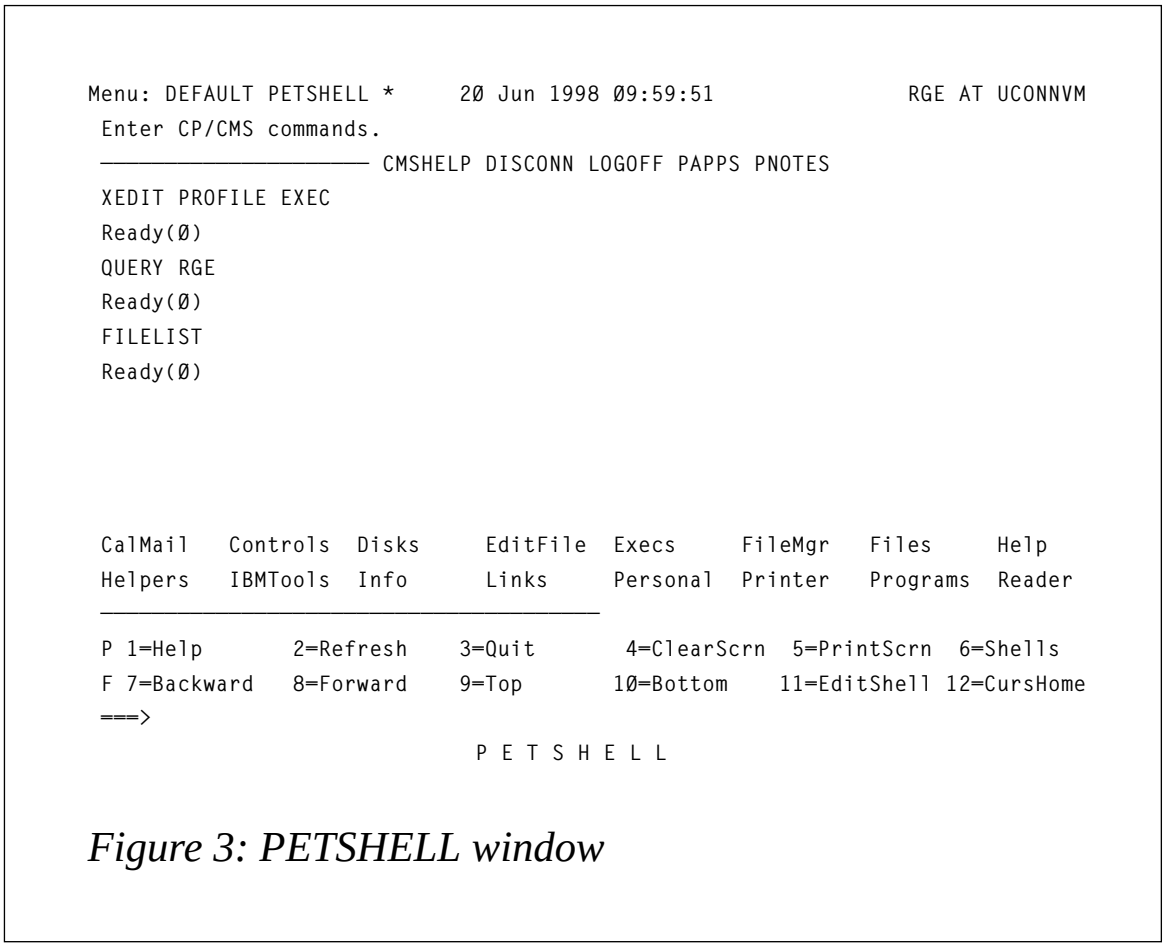

Figure 3: PETSHELL window

```
Ready(0)
FILELIST
Ready(0)
```

These represent three commands previously entered on the command line. One of these commands could be re-invoked exactly as before by clicking on the line containing the command. The text of the command can be altered before selection by typing over the previous command.

PETSHELL can be invoked by entering its name on the CMS (or XEDIT) command line or on any command line displayed by other PETs. For routine use, a CMS PF key can be defined to launch PETSHELL – and to use PETSHELL as the standard CMS desktop, you would simply invoke PETSHELL from your PROFILE EXEC as follows.

```
/* Profile Exec */
   .
   .
```

```
   .
'SET PFØ1 IMMED EXEC PETSHELL'
Push 'EXEC PETSHELL'
Exit
```

HOW FAR CAN PETS GO?

Obviously, the PETs interface is a text-only interface – but then a good deal of the world's business is conducted using text. While graphics are sometimes very important, most graphics seen today on the Web and elsewhere are merely icing on the content cake, and the cake consists of words.

For example, graphical calendars are appealing, but text-based calendars, like the one offered by the PCAL EXEC (shown in Figure 4) work quite nicely.

By clicking on PF4 or PF5 (4 = Previous, 5 = Next) the four month calendar is shifted forwards or backwards by one month. In other

```
+ ──────────────────────────────────────────────── +
|  Today: 20 Jun 1998     Select a date or PF key. |
|         June 1998               July 1998        |
|   Su Mo Tu We Th Fr Sa  |  Su Mo Tu We Th Fr Sa  |
|       1  2  3  4  5  6  |            1  2  3  4  |
|    7  8  9 10 11 12 13  |   5  6  7  8  9 10 11  |
|   14 15 16 17 18 19 20  |  12 13 14 15 16 17 18  |
|   21 22 23 24 25 26 27  |  19 20 21 22 23 24 25  |
|   28 29 30              |  26 27 28 29 30 31     |
|                         |                        |
|  ──────────────────────────────────────────────  |
|        August 1998      |    September 1998      |
|   Su Mo Tu We Th Fr Sa  |  Su Mo Tu We Th Fr Sa  |
|                      1  |         1  2  3  4  5  |
|    2  3  4  5  6  7  8  |   6  7  8  9 10 11 12  |
|    9 10 11 12 13 14 15  |  13 14 15 16 17 18 19  |
|   16 17 18 19 20 21 22  |  20 21 22 23 24 25 26  |
|   23 24 25 26 27 28 29  |  27 28 29 30           |
|   30 31                 |                        |
|  ──────────────────────────────────────────────  |
|  PF 1=Help  3=Quit  4=Previous  5=Next    C A L  |
+ ──────────────────────────────────────────────── +
```

Figure 4: PCAL calendar

applications, a user might select a date by clicking on it rather than by typing it.

The range of text-based applications that could benefit from the PETs style interface is very broad. We are more likely to be limited by our imaginations than by the basic technology, which is readily available.

ACKNOWLEDGEMENTS

I would like to express my appreciation for the work done to create the 3270 emulation software, *TCP3270 for Windows/NT Version 3.10* (Copyright McGill University, 1991-1996), whose thoughtful author(s) assigned to the right mouse button the mouse action required for PETs programs to work in the way they do.

Editor's note: in a future article, the author will focus on the technical aspects of PETs.

Richard G Ellis
Director, Computing and Information Systems
University of Connecticut (USA)

A full screen console interface – part 3

Editor's note: this month we continue the code for the full screen console interface for Disconnected Service Machines (DSM). This article is an extensive piece of work which will be published over several issues of VM Update*. It was felt that readers could benefit from the entire article and from the individual sections. Any comments or recommendations would be welcomed and should be addressed either to Xephon or directly to the author at fernando_duarte@vnet.ibm.com.*

```
         AIF   (T'&GARB EQ 'O').PROCESS
         MNOTE 8,'Invalid parameter &GARB'
         MEXIT
.*
.PROCESS ANOP
&COUNT   SETA  N'&OUT
.CHECK   AIF   (&COUNT GT Ø).CKØØØ
&COUNT   SETA  1
```

```
         AIF   ('&OUT' EQ '').CKEND
.CKØØØ   AIF   ('&OUT(&COUNT)' NE 'RC').CKØØ1
         AIF   (&RC).CKERR
&RC      SETB  1
         AGO   .CKEND
.CKØØ1   AIF   ('&OUT(&COUNT)' NE 'USER').CKØØ2
         AIF   (&USER).CKERR
&USER    SETB  1
         AGO   .CKEND
.CKØØ2   AIF   ('&OUT(&COUNT)' NE 'NOCMD').CKØØ3
         AIF   (&NOCMD).CKERR
&NOCMD   SETB  1
         AGO   .CKEND
.CKØØ3   AIF   ('&OUT(&COUNT)' NE 'NOALARM').CKØØ4
         AIF   (&NOALARM).CKERR
&NOALARM SETB  1
         AGO   .CKEND
.CKØØ4   AIF   ('&OUT(&COUNT)' NE 'SPACE').CKØØ5
         AIF   (&SPACE).CKERR
&SPACE   SETB  1
         AGO   .CKEND
.CKØØ5   AIF   ('&OUT(&COUNT)' NE 'CC').CKØØ6
         AIF   (&CC).CKERR
&CC      SETB  1
         AGO   .CKEND
.CKØØ6   MNOTE 8,'Invalid option ''&OUT(&COUNT)''.'
         MEXIT
.CKERR   MNOTE 4,'Repeated option ''&OUT(&COUNT)''. Ignored'
.CKEND   ANOP
&COUNT   SETA  &COUNT-1
         AIF   (&COUNT GT Ø).CKØØØ
.*
         AIF   (T'&MSGNUM NE 'O').VALØ1
         MNOTE 8,'Message number is missing.'
         MEXIT
.VALØ1   AIF   (NOT &RC).VALEND
         AIF   (NOT &USER AND NOT &NOCMD AND NOT &NOALARM).VALEND
         MNOTE 8,'RC cannot be used with any other option.'
         MEXIT
.VALEND  ANOP
.*
&OPTS    SETC  'Ø'
         AIF   (NOT &USER).SKPØ1
&OPTS    SETC  '&OPTS+MSGOPTUS'
.SKPØ1   AIF   (NOT &NOCMD).SKPØ2
&OPTS    SETC  '&OPTS+MSGOPTNC'
.SKPØ2   AIF   (NOT &NOALARM).SKPØ3
&OPTS    SETC  '&OPTS+MSGOPTNA'
.SKPØ3   AIF   (NOT &SPACE).SKPØ4
&OPTS    SETC  '&OPTS+MSGOPTSP'
```

```
.SKPØ4   AIF   ('&OPTS' EQ 'Ø').GEN
&OPTS    SETC  '&OPTS'(3,K'&OPTS)
.*
.GEN     ANOP
&MODULE  SETC  '&SYSECT'(4,3)
&LABEL   BAS   R1,*+1Ø                 Skip parameter list
         DC    H'&MSGNUM'              Message number
         DC    C'&MODULE'              Routine identification
         DC    AL1(&OPTS)              Message output
         AIF   (NOT &RC).NORC
         ST    R15,CSCRC               Store Return Code
.NORC    ANOP
         L     R15,@SCMSG
         A     R15,Ø(,R15)             Skip timestamp
         BAS   R14,2Ø(,R15)            Execute
         AIF   (NOT &CC).NOCC
         LA    R15,&MSGNUM             Store R15 with message number
.NOCC    ANOP
         MEND
```

CSCDATA MACRO

```
         MACRO
&LABEL   CSCDATA &PRINT
         LCLC  &DC,&A,&V
         AIF   ('&PRINT' NE 'PRINT').NOPRINT
         PUSH  PRINT
         PRINT GEN
.NOPRINT AIF   ('&SYSECT' EQ 'CSCSVP').CSECT
.*
&DC      SETC  'DS'
&A       SETC  'A '
&V       SETC  'V '
CSCDATA  DSECT
         AGO   .DATA
.*
.CSECT   ANOP
&DC      SETC  'DC'
&A       SETC  'A'
&V       SETC  'V'
         DS    ØD
*
* CSC Data area
*
*
CSCSVPD  EQU   *                       CSC Data area
.DATA    ANOP
         SPACE
*                                      Common routines
*
```

```
@CLOSE   &DC   &A.(CLOSE)             Close the shop
@ADD     &DC   &A.(ADD)               Add entry to user buffer
@DELETE  &DC   &A.(DELETE)            Delete entry from user buffer
@ADDTOFT &DC   &A.(ADDTOFT)           Add Tof-Of-File line (top)
@ADDTOFB &DC   &A.(ADDTOFB)                                (bottom)
@ADDEOFT &DC   &A.(ADDEOFT)               End-Of-File line (top)
@ADDEOFB &DC   &A.(ADDEOFB)                                (bottom)
@ADDBLKT &DC   &A.(ADDBLKT)               Blank line       (top)
@ADDBLKB &DC   &A.(ADDBLKB)                                (bottom)
@CLEAR   &DC   &A.(CLEAR)             Clear screen for CMS scrolling
@SEND    &DC   &A.(SEND)              IUCV SEND
@PREFIX  &DC   &A.(PREFIX)            Build record prefix
@MATCH   &DC   &A.(MATCH)             Process message table
@LOCATE  &DC   &A.(LOCATE)            Check message text against mask
@SELECT  &DC   &A.(SELECT)            Select message for processing
@OBTAIN  &DC   &A.(OBTAIN)            Allocate storage
@OBTAINP &DC   &A.(OBTAINP)           Allocate storage (page aligned)
@RELEASE &DC   &A.(RELEASE)           Release storage
@TRACE   &DC   &A.(TRACE)             Create Trace entry
@CSCIOX  &DC   &A.(CSCIOX)         *** Used by CSCMSG
         SPACE
@SCPARMC &DC   &A.(CSCPARMC)          IUCV Parmlist for CP
@SCPARMU &DC   &A.(CSCPARMU)          IUCV Parmlist for Users
@SCPARMA &DC   &A.(CSCPARMA)          IUCV Parmlist for APPC/VM
         SPACE
*                                     External routines
*
@SCCPW   &DC   &V.(CSCCPW)            Write Data File
@SCMSG   &DC   &V.(CSCMSG)            Display messages
@SCCFG   &DC   &V.(CSCCFG)            Setup and configuration
@SCRLS   &DC   &V.(CSCRLS)            Release allocated storage
@SCCLS   &DC   &V.(CSCCLS)            Terminate IUCV and Console trap
@SCSCN   &DC   &V.(CSCSCN)            Scan data, get next token
@SCSCNSC &DC   &V.(CSCSCNSC)          Search commands table after scan
@SCSCNVN &DC   &V.(CSCSCNVN)          Verify if numeric
@SCSEV   &DC   &V.(CSCSEV)            IUCV SEVER, terminate session
@SCOPC   &DC   &V.(CSCOPC)            Process Operator Commands (cons)
@SCUSC   &DC   &V.(CSCUSC)            Process User Commands (IUCV)
@SCUSCRB &DC   &V.(CSCUSCRB)            Rebuild User screen
@SCUSCRH &DC   &V.(CSCUSCRH)            Rebuild Header line
@SCUSCTL &DC   &V.(CSCUSCTL)            Move screen line to the top
@SCUSCBL &DC   &V.(CSCUSCBL)            Move screen line to the bottom
@SCUIN   &DC   &V.(CSCUIN)            Process <CSC>INI command
@SCUSB   &DC   &V.(CSCUSB)            Process Other User Commands
@SCWRP   &DC   &V.(CSCWRP)            Process WRAP messages
@SCWRPGS &DC   &V.(CSCWRPGS)            Get number of display columns
@SCWRPBT &DC   &V.(CSCWRPBT)            Adjust screen (Bottom line)
@SCWRPTP &DC   &V.(CSCWRPTP)            Adjust screen (Top line)
@SCWRPGT &DC   &V.(CSCWRPGT)            Locate top line on screen
@SCWRPGB &DC   &V.(CSCWRPGB)            Locate bottom line on screen
@SCBLD   &DC   &V.(CSCBLD)            Build user screen (327Ø DS)
```

```
@SCRDF   &DC   &V.(CSCRDF)          *** Force module to be linked
@SCRDFFT &DC   &V.(CSCRDFFT)           Read first record
@SCRDFLT &DC   &V.(CSCRDFLT)           Read last record
@SCRDFPR &DC   &V.(CSCRDFPR)           Read previous record
@SCRDFDP &DC   &V.(CSCRDFDP)           Read previous record from disk
@SCRDFNT &DC   &V.(CSCRDFNT)           Read next record
@SCRDFRS &DC   &V.(CSCRDFRS)           Locate last record written
@SCRDFGO &DC   &V.(CSCRDFGO)           Locate record by date and time
@SCTMR   &DC   &V.(CSCTMR)          Validate Event (TMR) entry
@SCRNL   &DC   &V.(CSCRNL)          Remote Node Links
@SCRNC   &DC   &V.(CSCRNC)          Remote Node Commands
@SCRNCTR &DC   &V.(CSCRNCTR)           Terminate all APPC/IUCV links
@SCRNCST &DC   &V.(CSCRNCST)           Start APPC/IUCV link
@SCRNCSP &DC   &V.(CSCRNCSP)           Stop APPC/IUCV link
@SCRNCSD &DC   &V.(CSCRNCSD)           Send data over APPC/VM link
@SCUSA   &DC   &V.(CSCUSA)          Process APPC/VM data received
@SCUSADP &DC   &V.(CSCUSADP)           Send data to user for display
@SCUSAPD &DC   &V.(CSCUSAPD)           Process pending request
@SCUSATC &DC   &V.(CSCUSATC)           Terminate connected sessions
@SCUSACL &DC   &V.(CSCUSACL)           Terminate affected sessions
@SCUSARL &DC   &V.(CSCUSARL)           Release UID block and buffers
         SPACE
CSCFLGØ1 DC    X'ØØ'                CSC Flag byte Ø1
HNDIOS   EQU   X'8Ø'                HNDIO SET executed
HNDIUCVS EQU   X'4Ø'                HNDIUCV SET executed
CMSIUCVC EQU   X'2Ø'                CMSIUCV CONNECT executed
CFGERROR EQU   X'1Ø'                Configuration error detected
MSGTIME  EQU   X'Ø8'                Prefix messages with time stamp
MSGPRINT EQU   X'Ø4'                Copy Console messages to Printer
CSCAPPC  EQU   X'Ø2'                Define APPC resources for remote
         SPACE
CSCFLGØ2 DC    X'ØØ'                CSC Flag byte Ø2
CSCWAIT  EQU   X'8Ø'                Waiting for work
WORKCP   EQU   X'4Ø'                Work to do (CP)
WORKIO   EQU   X'2Ø'                Work to do (Console IO)
WORKID   EQU   X'1Ø'                Work to do (User request on/off)
WORKMG   EQU   X'Ø8'                Work to do (User request message
WORKRM   EQU   X'Ø4'                Work to do (Remote user request)
WORKEND  EQU   X'Ø1'                END command entered
         SPACE
CSCFLGØ3 DC    X'ØØ'                CSC Flag byte Ø3 (CP messages)
CPFIRST  EQU   X'8Ø'                First time scheduled
CPCC     EQU   X'4Ø'                Connection completed
CPCCERR  EQU   X'2Ø'                Connection error
CPMSGERR EQU   X'1Ø'                Message error
CPTYPERR EQU   X'Ø8'                Message type error
         SPACE
*SCFLGØ4 DC    X'ØØ'                CSC Flag byte Ø4
         SPACE
*                                   Flags for CSCMSG option byte
MSGOPTUS EQU   X'8Ø'                Send message to user
```

```
MSGOPTNC EQU   X'4Ø'                    Do not redisplay input command
MSGOPTNA EQU   X'2Ø'                    Do not sound the alarm
MSGOPTSP EQU   X'1Ø'                    Do not compress spaces
         SPACE
         DS    ØD
*                                       User to Service Machine
*
COMMINI  DC    C'<CSC>INI'              Command to initiate the session
COMMCMD  DC    C'<CSC>CMD'              Other commands (data stream)
         SPACE
*                                       Service Machine to User
*
COMMRSK  DC    C'<CSC>RSK'              Reset keyboard
COMMACL  DC    C'<CSC>ACL'              Add command line
COMMALM  DC    C'<CSC>ALM'              Sound the alarm
COMMDCL  DC    C'<CSC>DCL'              Delete Command line
COMMHDR  DC    C'<CSC>HDR'              Overlay Header line
COMMMCL  DC    C'<CSC>MCL'              Move data to Command line
COMMMSG  DC    C'<CSC>MSG'              Move data to Message line
COMMSCR  DC    C'<CSC>SCR'              Rebuild User screen
COMMTTL  DC    C'<CSC>TTL'              Replace Title line
COMMPRT  DC    C'<CSC>PRT'              Create Print file (printer)
COMMWRT  DC    C'<CSC>WRT'              Create PrintLog file (disk)
         SPACE
CSCNODE  DS    CL8                      Local VM Node name
CSCMSGC  DS    CL8                      CP command to send messages
CSCRSCS  DS    CL8                      Local RSCS Service Machine name
CSCLOCAL DS    CL8                      Local APPC/VM Node name
CSCNAME  DC    C'CSC     '              CMS/IUCV name
CSCCPMSG DC    C'*MSG    '              *CPMSG Service name
CSCASTER DC    C'*       '              Match all (asterisk)
CSCCOMM  DS    CL12                     Last user command
CSCCOPT  DS    CL12                     Last user command option
CSCID    DC    C'CSC '                  Application ID
         SPACE
CSCLOCCH EQU   C'/'                     Default Locate character
CSCMATCH EQU   C'\'                     Default Match character
CSCPOST  EQU   X'4Ø'                    ECB Event completed bit
         SPACE
DFFILEW  FSCB  'CSC      DATA     A6',FORM=E,BUFFER=DFBUFF,BSIZE=128
DFFILER  FSCB  'CSC      DATA     A6',FORM=E,NOREC=32,BSIZE=4Ø96
         SPACE
MSGDFLVL EQU   4                        Default Message level
MSGLEVEL DS    F                        Message level
         SPACE
DFRBDFLT DC    F'16'                    Default Data File buffers (read)
DFRBMIN  DC    F'2'                        Minimum
DFRBMAX  DC    F'256'                      Maximum
DFSZDFLT DC    F'1Ø24'                  Default Data File size
DFSZMIN  DC    F'128'                      Minimum
DFRBUFFS DS    F                        Data File buffers for read
```

```
DFCURR   DS    F                       Current record pointer
DFEXPLIN DS    F                       Lines expanded
DFSSSLIN DS    F                       Lines processed
DFOLDTOT DS    F                       Old file size
DFNEWTOT DS    F                       New file size
         DS    ØD
DFBUFF   DS    ØCL128                  Buffer for Data File record
*                                   *** Layout MUST match CCHDFREC
DFDATE   DS    D                       Date
DFTIME   DS    D                       Time
DFUSER   DS    D                       User
DFOPTS   DS    X                       Options byte
DFOCYCLE EQU   X'8Ø'                   Cycle bit
DFOCONT1 EQU   X'4Ø'                   Continuation
DFOCONT2 EQU   X'2Ø'                   Continued
DFOHOLD  EQU   X'Ø2'                   Hold message
DFONODSP EQU   X'Ø1'                   Do not display
DFPNUM   DS    X                       Records on DF for previous cache
DFCNUM   DS    X                       Records on DF for current cache
DFRLEN   DS    X                       Message length
DFDATA   DS    CL1ØØ                   Message data
         SPACE
CACHESZ  DC    A(64*256/8)             Cache size (double words)
CACHEPTR DS    F                       Pointer to last entry updated
CACHE    DS    F                       Cache address
         SPACE
TRACESZ  DC    F'512'              *T* Trace Table size (double words)
TRACEPTR DS    F                   *T* Pointer to current entry
TRACEBEG DS    F                   *T* Begin of Trace Table
TRACEEND DS    F                   *T* End of Trace Table
         SPACE
UIDPTR   DS    F                       Pointer to first UIDSECT
SSSPTR   DS    F                       Pointer to list of sessions
RDFPTR   DS    F                       Pointer to list of DF buffers
PFXPTR   DS    F                       Pointer to PREFIX table
MSGPTR   DS    F                       Pointer to MSG table
HLDPTR   DS    F                       Pointer to HOLD table
HLDLAST  DS    F                       Pointer to last HOLD entry
RTEPTR   DS    F                       Pointer to RTE table
USRPTR   DS    F                       Pointer to USR table
RNDPTR   DS    F                       Pointer to RND table
TMRPTR   DS    F                       Pointer to TMR table
         SPACE
MSGSUB   DS    2F                      Area for MSG arbitrary character
ADDRCONS DS    F                       Console address
CSCBUFFE DS    F                       End address of IUCV message
CSCECB   DS    F                       ECB
CSCRC    DS    F                       CSC Return Code
CPMSGQ   DS    F                       CP messages queued
CSCBLEN  DC    AL2(L'CSCBUFF)          IUCV Buffer length
         SPACE
```

```
TOF      DC    C'*** *** *** Top of Data File *** *** ***'
EOF      DC    C'*** *** *** End of Data File *** *** ***'
RST      DC    C'*** *** * Restart of Data File * *** ***'
         SPACE
DIAG000C DS    4D                      Work area for DIAG 000C
IOXBK    DS    CL32                 *** INTBLOK (unable to make it work)
CSCBUFF  DS    CL256                   IUCV buffer (*MSG)
         SPACE
CSCUPP   DC    256AL1(*-CSCUPP)        Uppercase Translation table
         ORG   CSCUPP+C'a'
         DC    C'ABCDEFGHI'
         ORG   CSCUPP+C'j'
         DC    C'JKLMNOPQR'
         ORG   CSCUPP+C's'
         DC    C'STUVWXYZ'
         ORG
         SPACE
BLANKS   DC    CL16' '                 Clear, convert to uppercase
SCANUPP  DS    CL16                    Token in uppercase
SCANDEC  DS    D                       Work field to convert numerics
SCANLEN  DS    F                       Length of field
SCANMAX  DC    F'14'                   Maximum length of a token
SCANMAXN DC    F'8'                    Maximum length of numeric fields
         SPACE
ZERO     DC    F'0'                    Zero
ONE      DC    F'1'                    Zero plus one
TWO      DC    F'2'                    Zero plus one plus one
         SPACE
FSALL    DS    F                       Free storage allocations
FSALLDW  DS    F                            Allocations in double words
FSREL    DS    F                            Releases
FSRELDW  DS    F                            Releases in double words
FSINI    DS    F                            Allocations during init
FSINIDW  DS    F                            Allocations durint init dw
         SPACE
SCRHDRL  DS    F                       Length of Header line
SCRMCLL  DS    F                                 Command line data
SCRMSGL  DS    F                                 Message line
SCRTTLL  DS    F                                 Title Line
         SPACE
SCRHDR   DS    F     *** Max is 60 *** Address of Header line
SCRMCL   DS    F                72 ***            Command line data
SCRMSG   DS    F                74 ***            Message line
SCRTTL   DS    F                30 ***            Title Line
.*
         AIF   ('&PRINT' NE 'PRINT').END
         POP   PRINT
.END     ANOP
         MEND
```

CSCDS MACRO

```
         MACRO
         CSCDS &SECT,&PRINT
         LCLA  &COUNT,&TOTAL
         LCLC  &KEY
.*
         AIF   ('&PRINT' NE 'PRINT').NOPRINT
         PUSH  PRINT
         PRINT GEN
.NOPRINT ANOP
.*
&COUNT   SETA  1
&TOTAL   SETA  N'&SECT
         AIF   (&TOTAL EQ Ø).ERROR1
.LOOP    ANOP
&KEY     SETC  '&SECT(&COUNT)'
         AIF   ('&KEY' NE 'APP').CCH
         SPACE
APPSECT  DSECT
APPLL    DS    H                       APPC Logical record length
         DS    H
APPLEN   DS    F                       Command length
APPCMD   DS    CL8                     Command name
APPORIG  DS    CL8                     Originating CSC Node
APPVMID  DS    CL8                     Originating VM User-id
APPDATA  DS    C
.* SIZE  EQU   (*-APPSECT+7)/8         Size in double words
.* SIZEB EQU   (*-APPSECT)             Size in bytes
         AGO   .NEXT
.CCH     AIF   ('&KEY' NE 'CCH').CMD
         SPACE
CCHSECT  DSECT
CCHFWD   DS    F                       Forward chain pointer
CCHBWD   DS    F                       Backward chain pointer
CCHRECNO DS    F                       Record number on Data File
CCHLINE1 DS    X                       First line on screen
CCHLINE2 DS    X                       Last line on screen
CCHPREF  DS    X                       Prefix
CCHATTR  DS    X                       Message attributes
CCHDFREC DS    ØCL24Ø                  Cache data record
*                                   *** Layout MUST match DFBUFF
CCHDATE  DS    D                       Date
CCHTIME  DS    D                       Time
CCHUSER  DS    D                       User
CCHOPTS  DS    X                       Options byte
CCHCYCLE EQU   X'8Ø'                   Cycle bit
CCHCONT1 EQU   X'4Ø'                   Continuation
CCHCONT2 EQU   X'2Ø'                   Continued
CCHHOLD  EQU   X'Ø2'                   Hold message
CCHNODSP EQU   X'Ø1'                   Do not display
```

```
CCHPNUM  DS    X                       Records on DF for previous cache
CCHCNUM  DS    X                       Records on DF for current cache
CCHRLEN  DS    X                       Message length
CCHDATA  DS    CL212                   Message data
CCHSIZE  EQU   (*-CCHSECT+7)/8         Size in double words
CCHSIZEB EQU   (*-CCHSECT)             Size in bytes
         AGO  .NEXT
.CMD     AIF   ('&KEY' NE 'CMD').MSG
         SPACE
CMDSECT  DSECT
CMDNAME  DS    CL14                    Command name
CMDTYPE  DS    X                       Type of entry (Int/Ext/Bit)
CMDMIN   DS    X                       Minimum abbreviation - 1
CMDCLASS DS    F                       Command class
CMDADDR  DS    F                       Processing routine address
.* SIZE  EQU   (*-CMDSECT+7)/8         Size in double words
CMDSIZEB EQU   *-CMDSECT               Size in bytes
         AGO  .NEXT
.MSG     AIF   ('&KEY' NE 'MSG').PFX
         SPACE
MSGSECT  DSECT
MSGFWD   DS    F     Must be first *** Forward chain pointer
MSGOPTS  DS    X                       MSG options
MSGORTE  EQU   X'80'                   Route message
MSGORLS  EQU   X'40'                   Release message
MSGOEXT  EQU   X'20'                   Exit option
MSGUNIQ  EQU   X'10'                   Hold Unique
MSGHOLD  EQU   X'08'                   Hold message
MSGCASE  EQU   X'04'                   Ignore case
MSGALARM EQU   X'02'                   Sound the alarm
MSGNODSP EQU   X'01'                   Do not display
MSGATTR  DS    X                       Message attributes
MSGARBCH DS    X                       Arbitrary character
MSGANYCH DS    X                       Any character
MSGUSER  DS    CL8                     Originating user
MSGNAME  DS    CL8                     Message name
MSGROUTE DS    CL8                     Message routing user or list
MSGRLSE  DS    CL8                     Message name to release
MSGEXIT  DS    CL8                     Message exit REXX EXEC
MSGMASKE DS    F                       End address of mask
MSGMASK  DS    CL36                    Message mask
MSGSIZE  EQU   (*-MSGSECT+7)/8         Size in double words
MSGSIZEB EQU   *-MSGSECT               Size in bytes
         AGO  .NEXT
.PFX     AIF   ('&KEY' NE 'PFX').RDF
         SPACE
PFXSECT  DSECT
PFXFWD   DS    F     Must be first *** Forward chain pointer
PFXPREF  DS    C                       Record prefix
PFXATTR  DS    X                       Default attributes
PFXCLASS DS    X                       Class (range 25-32)
```

```
PFXCLMIN EQU   25                      Class minimum is 25
PFXCLMAX EQU   32                      Class maximum is 32
         DS    X
PFXUSER  DS    CL8                     User-id
PFXSIZE  EQU   (*-PFXSECT+7)/8         Size in double words
PFXSIZEB EQU   *-PFXSECT               Size in bytes
         AGO   .NEXT
.RDF     AIF   ('&KEY' NE 'RDF').RND
         SPACE
RDFSECT  DSECT
RDFFWD   DS    F     Must be first *** Forward chain pointer
RDFBWD   DS    F                       Backward chain pointer
*DFTIME  DS    D                       TOD value
RDFREC   DS    F                       First record of block
RDFADDR  DS    F                       Address of buffer
RDFSIZE  EQU   (*-RDFSECT+7)/8         Size in double words
.* SIZEB EQU   (*-RDFSECT)             Size in bytes
         AGO   .NEXT
.RND     AIF   ('&KEY' NE 'RND').RTE
         SPACE
RNDSECT  DSECT
RNDFWD   DS    F     Must be first *** Forward chain pointer
RNDOPT1  DS    X                       Option byte 1
RNDOLCL  EQU   X'80'                   Entry for local node
RNDOLRS  EQU   X'40'                   Define node with LOCAL resource
RNDOTMP  EQU   X'20'                   Temporary entry
RNDOSND  EQU   X'10'                   Send link is active
RNDORCV  EQU   X'08'                   Receive link is active
RNDORQS  EQU   X'04'                   Pending request (send)
RNDORQR  EQU   X'02'                   Pending request (receive)
RNDORQP  EQU   X'01'                   Pending request (Pending conn)
RNDOPT2  DS    X                   *2*
RNDOCNC  EQU   X'80'                       Connection complete
RNDOCNS  EQU   X'40'                       Severed connection
RNDOCNP  EQU   X'20'                       Pending connection
RNDOSPG  EQU   X'10'                       Send in progress
RNDORPG  EQU   X'08'                       Receive in progress
RNDOMSG  EQU   X'04'                       Incoming message
RNDOFNC  EQU   X'02'                       Function complete
RNDOPND  EQU   X'01'                       Request pending
RNDMSGL  DS    H                   *3* Incoming message length
RNDBUFSZ EQU   512                     Send/Receive buffer size (dw)
RNDSBUFF DS    F                       Send buffer address
RNDRBUFF DS    F                       Receive buffer address
RNDPIDS  DS    F                       Sending IUCV PATHID+FLAG1+IPTYPE
RNDPIDR  DS    F                       Receiving IUCV path
RNDNODE  DS    CL8                     Node name
RNDRSRC  DS    CL8                     Resource name
RNDSIZE  EQU   (*-RNDSECT+7)/8         Size in double words
.* SIZEB EQU   (*-RNDSECT)             Size in bytes
         AGO   .NEXT
```

```
.RTE     AIF   ('&KEY' NE 'RTE').TMR
         SPACE
RTESECT  DSECT
RTEFWD   DS    F                       Forward chain pointer
RTECNT   DS    X                       Node/User entries used
RTEMAX   EQU   3                       Node/User entries defined
         DS    X
         DS    X
         DS    X
RTENAME  DS    CL8                     Route name
RTENODE  DS    CL8                     VM Node
RTEUSER  DS    CL8                     User-id
         DS    2CL16                   Space for two node/user entries
RTESIZE  EQU   (*-RTESECT+7)/8         Size in double words
.* SIZEB EQU   (*-RTESECT)             Size in bytes
         AGO  .NEXT
.TMR     AIF   ('&KEY' NE 'TMR').UID
         SPACE
TMRSECT  DSECT
TMRFWD   DS    F                           Forward chain pointer
TMRBWD   DS    F                           Backward chain pointer
TMROPT1  DS    X                   *1* Option byte 1
TMRONAME EQU   X'80'                       Name
TMROADDR EQU   X'40'                       Address
TMROCMD  EQU   X'20'                       Command
TMROPT2  DS    X                   *2* Option byte 2
TMRODATE EQU   X'80'                       Date
TMROFROM EQU   X'40'                       From
TMROTO   EQU   X'20'                       To
TMROTIME EQU   X'10'                       TIme
TMROFRST EQU   X'08'                       FIrst
TMROLAST EQU   X'04'                       Last
TMROINT  EQU   X'02'                       Interval
TMRODAYS EQU   X'01'                       DAYs
TMROPT3  DS    X                   *3* Option byte 3
TMRODSUN EQU   X'80'                       Sunday
TMRODMON EQU   X'40'                       Monday
TMRODTUE EQU   X'20'                       Tuesday
TMRODWED EQU   X'10'                       Wednesday
TMRODTHU EQU   X'08'                       Thursday
TMRODFRI EQU   X'04'                       Friday
TMRODSAT EQU   X'02'                       Saturday
TMROPT4  DS    X                   *4* Option byte 4
TMRADDR  DS    F                           Command address (internal call)
TMRNAME  DS    CL8                         Event name
TMRCC    DS    CL8                         Next date/time (TOD format)
TMRNDATE DS    0F                          Next date (yyyymmdd)
TMRNDYY  DS    H
TMRNDMM  DS    X
TMRNDDD  DS    X
TMRDATE  DS    0F                          Date (yyyymmdd)
```

```
TMRDTYY  DS    H
TMRDTMM  DS    X
TMRDTDD  DS    X
TMRFROM  DS    ØF                      From (yyyymmdd)
TMRFRYY  DS    H
TMRFRMM  DS    X
TMRFRDD  DS    X
TMRTO    DS    ØF                      To (yyyymmdd)
TMRTOYY  DS    H
TMRTOMM  DS    X
TMRTODD  DS    X
TMRNTIME DS    ØF                      Next time (hhmmss)
TMRNTHH  DS    H
TMRNTMM  DS    X
TMRNTSS  DS    X
TMRTIME  DS    ØF                      Time (hhmmss)
TMRTIHH  DS    H
TMRTIMM  DS    X
TMRTISS  DS    X
TMRFIRST DS    ØF                      First (hhmmss)
TMRFTHH  DS    H
TMRFTMM  DS    X
TMRFTSS  DS    X
TMRLAST  DS    ØF                      Last (hhmmss)
TMRLTHH  DS    H
TMRLTMM  DS    X
TMRLTSS  DS    X
TMRINT   DS    ØF                      Interval (hhmmss)
TMRITHH  DS    H
TMRITMM  DS    X
TMRITSS  DS    X
TMRCMNDL DS    H                       Command length
TMRCMND  DS    CL13Ø                   Command
TMRSIZE  EQU   (*-TMRSECT+7)/8         Size in double words
.* SIZEB EQU   (*-TMRSECT)             Size in bytes
         AGO   .NEXT
.UID     AIF   ('&KEY' NE 'UID').USR
         SPACE
UIDSECT  DSECT
UIDFWD   DS    F     Must be first *** Forward chain pointer
UIDOPTS  DS    ØF
UIDOPT1  DS    X                       Option byte 1
UIDREQ   EQU   X'8Ø'                   Request to be processed
UIDSEV   EQU   X'4Ø'                   Session severed
UIDSEND  EQU   X'2Ø'                   Send in progress
UIDPEND  EQU   X'1Ø'                   UICV SEND is pending
UIDFFREE EQU   X'Ø8'                   Record is in Free List
UIDRLSE  EQU   X'Ø4'                   Refresh screen after msg release
UIDCONN  EQU   X'Ø2'                   User connected to remote node
UIDRMTE  EQU   X'Ø1'                   Remote user
UIDOPT2  DS    X                       Option byte 2
```

```
UIDINIT  EQU   X'80'                  INIT received
UIDEDS   EQU   X'40'                  EDS supported
UIDAUTO  EQU   X'20'                  Auto refresh
UIDDATE  EQU   X'10'                  Date displayed on screen
UIDTIME  EQU   X'08'                  Time displayed on screen
UIDUSER  EQU   X'04'                  User displayed on screen
UIDINC   EQU   X'02'                  Include records
UIDEXC   EQU   X'01'                  Exclude records
UIDOPT3  DS    X                      Option byte 3
UIDCREQ  EQU   X'80'                  Remote connection requested
UIDCLEAR EQU   X'40'                  CMS Clear executed last
UIDFLTR  EQU   X'20'                  Record Filter is active
UIDWRAP  EQU   X'10'                  Wrap around messages
UIDCMS   EQU   X'08'                  CMS scrolling
UIDNODSP EQU   X'04'                  Filter NoDisplay records
UIDPPROG EQU   X'02'                  Print command in progress
UIDPAUTO EQU   X'01'                  Set AUTO option after Print
UIDOPT4  DS    X
UIDBSCR  EQU   X'80'                  Build screen data stream
UIDBACL  EQU   X'40'                  Add command line
UIDBALM  EQU   X'20'                  Sound the alarm
UIDBDCL  EQU   X'10'                  Delete Command line
UIDBHDR  EQU   X'08'                  Overlay Header line
UIDBMCL  EQU   X'04'                  Move data to Command line
UIDBMSG  EQU   X'02'                  Display new Message
UIDBTTL  EQU   X'01'                  Replace Title line
UIDOPT5  DS    X
UIDAPEND EQU   X'80'                  APPC/VM request pending
UIDA$SS  EQU   X'40'                     Start Session
UIDA$SC  EQU   X'20'                     Session Created
UIDA$SR  EQU   X'10'                     Session Rejected
UIDA$SE  EQU   X'08'                     Session Ended
UIDA$SD  EQU   X'04'                     Send Data to Connected node
UIDA$SU  EQU   X'02'                     Send Data to User
UIDA$TC  EQU   X'01'                     Terminate Connected Session
UIDOPT6  DS    X
UIDA$RN  EQU   X'02'                     Refresh CNN on user screen
UIDA$RU  EQU   X'01'                     Release UID block after Send
UIDCOL1  DS    X                      First display column
UIDSCRL  DS    X                      Detail lines on screen
UIDCMSTP DS    F                      Last top line for CMS scroll
UIDVMID  DS    CL8                    Partner userid
UIDSEL   DS    CL8                    Include/Exclude prefixes
UIDPID   DS    XL2                *1* IUCV Path Id
UIDFLAG1 DS    X                  *2* IUCV FLags1 byte
UIDTYPE  DS    X                  *3* IUCV IPTYPE / IPRCODE
UIDCLASS DS    F                      User classes
UIDORIG  DS    CL8                    Local APPC/VM Node name
UIDPIDCN DS    F                      Connect PATHID+FLAGS1+IPTYPE
UIDPIDRM DS    F                      Remote PATHID+FLAGS1+IPTYPE
UIDPWREM DS    F                      Remaining lines to Print / Write
```

```
         DS    F
UIDCSCSZ EQU   256/8                   CSCBUFF for APPC/VM
UIDCSC   DS    F                       Address of work CSCBUFF
UIDCSCL  DS    F                       Length of CSCBUFF data
UIDSCRNA DS    F                       Alternate screen address (APPC)
UIDSCRN  DS    F                       Screen address (4K)
UIDSCRNL DS    F                       Screen length (data stream)
UIDSCRSZ EQU   4Ø96/8                   Screen size in double words
UIDBUFF  DS    F                       Buffer address (12K)
UIDBUFSZ EQU   48*256/8                Buffer size in double words
UIDBUFF1 DS    F                       Address of first entry
UIDBUFF2 DS    F                       Address of last entry
UIDFREE1 DS    F                       Address of first free entry
UIDFREE2 DS    F                       Address of last free entry
UIDSIZE  EQU   (*-UIDSECT+7)/8         Size in double words
UIDSIZEB EQU   *-UIDSECT               Size in bytes
         AGO  .NEXT
.USR     AIF   ('&KEY' NE 'USR').ERRORØ2
         SPACE
USRSECT  DSECT
USRFWD   DS    F                       Forward chain pointer
USRCLASS DS    F                       User classes
USRNAME  DS    CL8                     User name
USRSIZE  EQU   (*-USRSECT+7)/8         Size in double words
.* SIZEB EQU   *-USRSECT               Size in bytes
         AGO  .NEXT
.ERROR1  MNOTE 8,'No DSECT selected.'
         AGO   .NEXT
.ERROR2  MNOTE 8,'Invalid DSECT selected ''&KEY''.'
.NEXT    ANOP
&COUNT   SETA  &COUNT+1
         AIF   (&COUNT LE &TOTAL).LOOP
.*
         AIF   ('&PRINT' NE 'PRINT').END
         POP   PRINT
.END     ANOP
.*
         MEND
```

CSCCMMD MACRO

```
         MACRO
&LABEL   CMMD
         LCLA  &IX
         LCLB  &ER
         LCLC  &TP,&CL,&AB,&NM,&AD,&HEX(33)
&HEX(Ø1) SETC  '0000000000'
&HEX(Ø2) SETC  '8000000001'
&HEX(Ø3) SETC  '4000000002'
&HEX(Ø4) SETC  '2000000003'
```

```
&HEX(Ø5) SETC  '1000000004'
&HEX(Ø6) SETC  '0800000005'
&HEX(Ø7) SETC  '0400000006'
&HEX(Ø8) SETC  '0200000007'
&HEX(Ø9) SETC  '0100000008'
&HEX(1Ø) SETC  '0080000009'
&HEX(11) SETC  '004000000A'
&HEX(12) SETC  '002000000B'
&HEX(13) SETC  '001000000C'
&HEX(14) SETC  '000800000D'
&HEX(15) SETC  '000400000E'
&HEX(16) SETC  '000200000F'
&HEX(17) SETC  '0001000010'
&HEX(18) SETC  '0000800011'
&HEX(19) SETC  '0000400012'
&HEX(2Ø) SETC  '0000200013'
&HEX(21) SETC  '0000100014'
&HEX(22) SETC  '0000080015'
&HEX(23) SETC  '0000040016'
&HEX(24) SETC  '0000020017'
&HEX(25) SETC  '0000010018'
&HEX(26) SETC  '0000008019'
&HEX(27) SETC  '000000401A'
&HEX(28) SETC  '000000201B'
&HEX(29) SETC  '000000101C'
&HEX(3Ø) SETC  '000000081D'
&HEX(31) SETC  '000000041E'
&HEX(32) SETC  '000000021F'
&HEX(33) SETC  '0000000120'
.*
         AIF   (N'&SYSLIST GT Ø).VØØ
         MNOTE 8,'Missing operand(s).'
         MEXIT
.*
.VØØ     ANOP
&IX      SETA  &IX+1
.*
.VØ1     AIF   (N'&SYSLIST(&IX) EQ 5).VØ2
         MNOTE 8,'Entry number &IX is invalid.'
&ER      SETB  1
         AGO   .V99
.VØ2     AIF   ('&SYSLIST(&IX,1)' EQ 'I' OR '&SYSLIST(&IX,1)' EQ 'E' OR*
                '&SYSLIST(&IX,1)' EQ 'B').VØ3
         MNOTE 8,'Invalid type &SYSLIST(&IX,1)..'
&ER      SETB  1
         AGO   .V99
.VØ3     AIF   (T'&SYSLIST(&IX,2) EQ 'N').VØ4
         MNOTE 8,'Invalid class &SYSLIST(&IX,2)..'
&ER      SETB  1
         AGO   .V99
.VØ4     AIF   (&SYSLIST(&IX,2) LT 33).VØ5
```

```
         MNOTE 8,'Class must be in the range Ø-32.'
&ER      SETB  1
         AGO   .V99
.VØ5     AIF   (T'&SYSLIST(&IX,3) EQ 'N').VØ6
         MNOTE 8,'Invalid abbreviation &SYSLIST(&IX,3)..'
&ER      SETB  1
         AGO   .V99
.VØ6     AIF   (&SYSLIST(&IX,3) LE K'&SYSLIST(&IX,4) AND              *
                &SYSLIST(&IX,3) GT Ø).VØ7
         MNOTE 8,'Invalid abbreviation for name &SYSLIST(&IX,4)..'
&ER      SETB  1
         AGO   .V99
.VØ7     AIF   (K'&SYSLIST(&IX,5) LE 16).V99
         MNOTE 8,'Address &SYSLIST(&IX,5) too long.'
&ER      SETB  1
         AGO   .V99
.V99     AIF   (&IX LT N'&SYSLIST).VØØ
         AIF   (NOT &ER).GEN
         MEXIT
.*
.GEN     ANOP
&LABEL   DS    ØD
&IX      SETA  Ø
.*
.GØØ     ANOP
&IX      SETA  &IX+1
&CL      SETC  '&HEX(&SYSLIST(&IX,2)+1)'(1,8)
&TP      SETC  '&SYSLIST(&IX,1)'
&AB      SETC  '&HEX(&SYSLIST(&IX,3))'(9,2)
&NM      SETC  '&SYSLIST(&IX,4)'
         AIF   ('&NM'(1,1) NE '''').GØ1
&NM      SETC  '&NM'(2,K'&NM-2)
.GØ1     ANOP
&NM      SETC  '&NM             '(1,14)
&AD      SETC  'A(&SYSLIST(&IX,5))'
         AIF   ('&TP' NE 'E').GØ2
&AD      SETC  'V(&SYSLIST(&IX,5))'
.GØ2     ANOP
         DC    C'&NM',C'&TP',X'&AB',X'&CL',&AD
         AIF   (&IX LT N'&SYSLIST).GØØ
         DC    X'FFFFFFFFFFFFFFFF'
         MEND
```

You can now create the CSC macro library.

Editor's note: this article will be continued next month.

Fernando Duarte
Analyst (Canada)

VM news

IBM has announced Version 2 Release 3 of its VM server platform, formerly VM Productivity System (VMPS). This includes a maintenance upgrade of VM/ESA 2.3.0 to 2.4.0 and OfficeVision 1.3.0 to 1.4.0, plus new products including COBOL for MVS and VM, and Language Environment for OS/390 and VM. DisplayWrite/370, BookManager Build, BookManager Read, and Bulletin/VM are no longer included in the suite.

VM/ESA improvements include a Java Virtual Machine for creating and porting Java applications, an MQ client interface for application-to-application communication across different systems and architectures, integration of CMS Pipelines PRPQ, TCP/IP performance enhancements, and added support for the year 2000.

OfficeVision/VM Release 4.0 is now Year 2000 ready. The default panels shipped with OV Release 4 will be those supporting Shared PROFS.

For further information contact your local IBM representative.

* * *

Sterling Software has launched a new VM skills service that includes training and education, software implementation and consulting, and assessments for Year 2000 compliance.

The courses cover topics such as VM/ESA technology, Internet and Web technology, systems management, and the company's own VM products.

For further information contact:
Sterling Software, 1800 Alexander Bell Drive, Reston, VA 22091, USA.
Tel: (703) 264 8000.
Sterling Software, 64 London Road, Reading, Berkshire, RG1 5AS, UK.
Tel: (0118) 975 0055.
URL: http://www.vm.sterling.com.

* * *

IBM has announced High Level Assembler Release 3 for VM, MVS, and VSE, to replace all previous System/370 and System/390 Assemblers. This includes tools for developing and maintaining all Assembler applications, converting symbolic source statements to machine language object code with extensive checks for common coding errors. There is support for ESA/390 IEEE binary floating-point instructions, data, and additional floating point registers, plus new diagnostic and cross-reference features and a range of language enhancements.

The USING statement has been enhanced to allow a limit to the resolution range to be specified, and a new ACONTROL statement allows fine-grained controls over certain options within the source program. Its status may be saved and restored with the PUSH and POP instructions. A new INFO option provides details of the current service status of the user's Assembler, and a cross reference of general register usage has tags to indicate usage types.

For further information contact your local IBM representative.