163 MVS update

April 2000

In this issue

Published by
Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 33598
From USA: 01144 1635 33598
E-mail: Jaimek@xephon.com

North American office
Xephon/QNA
PO Box 350100,
Westminister, CO 80035-0100
USA
Telephone: 303 410 9344

Contributions

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

Editor
Jaime Kaminski

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.

MVS Update on-line

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

Subscriptions and back-issues

A year's subscription to *MVS Update*, comprising twelve monthly issues, costs £340.00 in the UK; $505.00 in the USA and Canada; £346.00 in Europe; £352.00 in Australasia and Japan; and £350.00 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1992 issue, are available separately to subscribers for £29.00 ($43.00) each including postage.

Printed in England.

OS/390 Version 2 Release 9

INTRODUCTION

31 March 2000 saw the delivery of OS/390 Version 2 Release 9. Key features of the new release include Unix System Services file system support of HFS, better workload management through multisystem enclave support, cryptographic enhancements, text search support for XML documents and Unicode data, XES enhancements to Parallel Sysplex, Windows-compatible file and print serving, and an improved Language Environment.

SUPPORTING E-BUSINESS

IBM is backing its e-business initiatives with OS/390 Version 2 Release 9, by integrating business processes, extending the life of existing applications, and easing the task of porting Unix-based applications to the platform. Enhancements include:

- Porting C and C++ language applications is easier through new Language Environment and OS/390 Unix System Services support. The Language Environment and Unix System Services will now support 64-bit integers. These enhancements will make it easier for customers and solution developers to port Unix applications to the System/390 server platform.
- The WebSphere Application Server supports new industry standards for Java Server Pages and Servlets. New OS/390 Version 2 Release 9 enhancements include support for Websphere Studio Tooling and VisualAge for Java Tooling. Now e-business applications can be developed for any platform, including System/390, in a development environment.
- Other new OS/390 Version 2 Release 9 enhancements include access to DB2 data via the Java DataBase Connection standard protocol.

SYSTEMS MANAGEMENT

Systems management functionality has been enhancements include:

- File and Print server support for Windows workstation software in Release 9 will provide Windows networking compatibility on the OS/390 server. Systems administrators will be able to give users access to high-speed OS/390 attached printers. The Server Message Block (SMB) protocol will enable this access without having to manage multiple software programs on individual PCs.
- S/390's Parallel Sysplex clustering technology provides highly scalable capacity, workload balancing, resource sharing, and maximum availability. New Parallel Sysplex features include:
  - CFCC code level 9, which introduces extensions to the Coupling Facility list structure. OS/390 Version 2 Release 9 will use this enhanced structure to extend the scope of the OS/390 Workload Manager (WLM) to run work on different OS/390 images in a Parallel Sysplex cluster.
  - Enhanced Contention Analysis is designed to significantly improve multi-system availability and serviceability for both base and System/390 Parallel Sysplex environments.
  - Shared Hierarchical File Systems (HFS) support for the System 390 Parallel Sysplex environment adds support to the OS/390 Unix System Services file system for simultaneous read/write access of the same HFS running on different OS/390 images in a Parallel Sysplex cluster.

SECURITY

Security enhancements include:

- Support for PCI Cryptographic Coprocessor (PCICC), an optional feature of System/390 G5 and G6 Enterprise Servers. PCICC adds function and performance to the System/390 CMOS Cryptographic Coprocessors. Secure Web serving and the ability to keep e-business transaction details private requires the use of complex cryptographic operations. Performing these operations in software can severely restrict the number of users a company's Web server can support. It is essential to offload this processing to specialized cryptographic hardware to achieve realistic performance of secure Web serving.

- With PCICC, IBM has increased the hardware performance of cryptographic operations used by System/390 G5 and G6 servers for SSL (Secure Sockets Layer) connections by a factor of six. SSL is the commonly-accepted communications protocol for secure Web serving. Support for multiple PCICC features on a single G5 or G6 server allows customers to grow their cryptographic capacity with their e-business processing needs.

  IBM System/390 makes the US government's highest-rated commercial security products. The IBM System/390 CMOS Cryptographic Coprocessor and the IBM 4758 Model 1 PCI Cryptographic Coprocessors are the only security devices of their kind to be awarded the Federal Information Processing Standard (FIPS) 140-1 Level 4 validation by the US government. PCICC is based on the recently announced IBM 4758-2 PCI Cryptographic Coprocessor, which also meets FIPS 140-1 Level 4 requirements.

- Other security enhancements include the lifting of a special US government export restriction against OS/390 System SSL. System SSL Triple DES encryption can now be exported outside the USA and Canada (subject to the US government's general cryptographic export regulations). Also, a new callable interface to System SSL is available for programs running on OS/390 under high-speed dispatching known as SRB Mode.

NONDISTRUPTIVE UPGRADES

Non-disruptive upgrades, extended networking and I/O capabilites have been improved:

- IBM has enhanced the System/390 G5 and G6 Enterprise Server models, by improving the Capacity Upgrade on Demand function, other changes eliminate LPAR disruption during such upgrades.

- To extend high-speed networking capabilities and improve throughput, the System/390 platform extends support of its OSA-Express adapters and Queued Direct I/O architecture for the ATM 155 standard and Fast Ethernet protocols as well as high-speed message passing for Fast Ethernet LPAR-to-LPAR communications. In addition, increased G6 FICON channel support and increased G5 and G6 sub-channel support will provide customers with increased I/O bandwidth capabilities.

DISASTER RECOVERY

The disaster recovery functionality has also been improved:

- System/390's Geographically Dispersed Parallel Sysplex (GDPS), a multi-site continuous availability disaster recovery solution, is now enhanced to take advantage of the benefits of Capacity BackUp. Upon detection of a processor failure, site failure or planned disaster test, GDPS will dynamically add reserved Capacity BackUp processing power to the systems in the takeover site to restore capacity for the mission critical production applications. This simplifies the recovery process, removing the need for manual customer intervention and eliminating possible operator errors, and reduces the outage time of critical workloads from hours to minutes.
- Capacity BackUp's Automatic Password Authentication can also reduce the time required to activate emergency capacity from hours to minutes while minimizing the need for customer intervention and the potential for operator errors. In addition, enhancements have been made to eliminate LPAR disruption during activation of Capacity BackUp engines.

BENEFITS

OS/390 Release 9 has new functionality which confers many benefits. The new Unix System Services removes the old limitation of file systems being shared only across a Parallel Sysplex cluster in read-only mode. Also new are enhancements to the integrated WebSphere Application Server Version 1.2, which supports Java Server pages and servlets. It also uses OS/390's Workload Manager for prioritizing tasks. Release 9 supports the new cryptographic capabilities in System/390 G5 and G6 servers and there is additional support for digital certificates, which lets more users of a Web application access the application with RACF but with less administration.

There are also more management tools, including Web-based wizards to help configure the Parallel Sysplex environment and to calculate structure sizes for products that use the Coupling Facility. Finally, IBM stated that OS/390 Version 2 Release 9 is the last to include the following functions as part of the OS: LAN Server, VisualLift RTE, VisualLift ADE, High Speed UDP, and Softcopy Print.

IBM is heavily pushing the OS/390 platform for e-business applications. Underlying this IBM sees enterprises moving from an Online Transaction Processing (OLTP) model, the current standard for mission-critical business applications, to e-transaction processing (e-tp), a new model for mission-critical e-business applications.

The difference between the two can be summarized as the difference between a static and a dynamic environment, or between a structured and unstructured one. The e-transaction processing environment is similar to the OLTP environment but needs to support an unpredictable load. An e-transaction processing system must be capable of providing: 24 hour availability, scalability and usable capacity, the capability to manage commits across an entire e-transaction, robust security , high data throughput, transaction-level accounting, and the ability to display applications across heterogeneous platforms.

OS/390 is second to none in all of these areas, but to gain market share as a server platform, rather than retain current levels of market share, IBM needs to drill down into the SME (Small and Medium Enterprise) market. This market is key for the growth of the platform. To achieve this, it needs to actively demonstrate the practical application and benefits of OS/390 running on the System/390 platform for smaller users. This is unlikely to happen in the current environment, because smaller enterprises are often unable to afford the platform, may not have the required skills in-house, and may have been scared off by the highly-publicized problems with the low-end Multiprise 3000 boxes.

To alleviate this, IBM needs to create greater awareness of the platform outside the large enterprise. For example, a simple way of achieving this in the current e-economy would be to use the System/390 platform running OS/390 as the basis for an e-commerce hosting service. This would demonstrate availablity and power of the platform to the SME market, and as these companies grow they could then exploit the platform for themselves. Without such proactive initiatives from IBM the platform could become relegted to becoming a highly reliable data server while the e-business revolution passes by.

OS/390 is without parallel in the large enterprise market, but IBM needs to extend its e-business stratgey to include SMEs.

Manipulating WLM information in batch

INTRODUCTION

Workload Manager (WLM) is normally defined, installed, and activated using ISPF. It also has the ability to offload the dataset contents into a PDS and to reload it from a PDS. This is good enough to generate a transportable copy of the contents of the service policy. A good idea when managing multiple systems is to keep all the definitions in a central dataset (for example, on a test system) and then distribute relevant parts of it to other systems. Looking at the available process, these are the weaknesses if we try to achieve this:

- It requires manual intervention. To download and upload the information into and from the PDS requires the user to sign on to TSO and to do the copying via ISPF. This may not always be convenient because change management may forbid this sort of activity during daytime. So the systems programmer has to sign on at night to install and activate the changes.
- Unless the WLM definitions are manually edited once uploaded from the PDS, all systems will receive the full contents of the WLM definitions. There are times where this may not be desirable – for instance when different systems have different scheduling environment names. It could lead to a high level of confusion to see a Test scheduling environment on a production system. This would also apply to other system-specific parameters.

IBM has made available the following three macros that we can use in batch to overcome these problems:

- IWMDEXTR – this macro extracts the contents of the service definition and copies it into a storage area.
- IWMDINST – this macro copies the definitions from a storage area into WLM. It also does consistency checking and will fail with an applicable return code if an inconsistency is detected, eg a reference to a scheduling resource where the resource is not defined.

- IWMPACT – this macro can be used to activate the service definition.

Considering the above three macros, we found a way to make a handy batch utility to:

1 EXTRACT all the WLM contents into storage.

2 Copy the contents from storage into a dataset (which we send to other systems).

3 Selectively INSTALL parts of the contents from the dataset into other systems.

4 ACTIVATE the service definition with an option to delay it until after hours.

This will give us the ability to keep all WLM definitions in one place, strip out parts of it per system, INSTALL the definitions into another system's WLM and ACTIVATE the new service definition with a time delay. This entire process can be fully set up in JCL and run on a regular basis to distribute WLM definitions.

This article is about developing a program to do just that. Before we actually work through the program parameters and provide the source code, we will have a brief look at the way the data is extracted and installed from and into WLM. This will give you an understanding of what may be required should you decide to customize the program to suit your particular requirements.

WLM DATA FORMAT

The WLMDEXTR macro extracts the data into an area pre-allocated by the caller. The size of the area obviously depends on the number of definitions in the WLM dataset and return and reason codes are given if the area allocated is too small. Similarly, the IWMDINST macro reads the data from a provided storage area and writes it into the WLM dataset. The content is basically a contiguous string of data consisting of the different components. There is a header section (mapped by the SERVDHDR macro) consisting of pointers to the areas containing the different components. For instance, to get to the section containing the scheduling environment data, we extract the data and then map it with the following:

```
LA    R2,WLMTABLE          /*  Point to start of data     */
USING SERVDHDR,R2          /*  Addressability to header   */
A     R2,SERVD_SVSEA_OFF   /*  Offset to S/E definitions  */
USING SVSEAHDR,R2          /*  Addressability to S/E defs */
```

This same format is used when the data is INSTALLed. So, if we EXTRACT, we can then cut the data up into records (with a format of our choice), send/store it somewhere and then rebuild the data in storage from where we can INSTALL the definitions. We can also 'tamper' with the pointers and the definitions, provided that we do not cause any discrepancies. For instance, if we try to INSTALL a Service Definition that contains references to Scheduling Resources that gave not been defined, we will get an error code from IWMDINST. Although we do not get a description of where the discrepancy is, it does return a pointer containing the offset of the error. Here is a more specific example:

The Scheduling Environment section is pointed by SERVD_SVSEA_OFF and consists of three separate sub-sections:

- A section describing all the Service Definitions
- A section describing all the Scheduling Resources
- A section describing Service Definition/ Scheduling Resource pairs.

These sections are in adjacent storage areas and if there is an error, a non-zero return code and an offset is returned in a field called VALCHECK_OFFSET. By analysing the start address of each of the three sections, it can be determined where exactly the problem occurred. This is in fact a good feature and provides the same level of protection that is given when definitions are entered via ISPF.

THEPROGRAM

This is a copy of a working program to do what we have discussed so far. It receives the following parameters:

- FUNC – this can be EXTRACT or INSTALL. The intention is to run the EXTRACT on the repository system, send the extracted data set to the target system (with FTP or XMIT) and then run the (potentially partial) INSTALL there.

- SYSTEM – this is the name of the system for which we are INSTALLing data and must match the name of the system the INSTALL is run on. Because an error run could have serious effects on the performance of a system, this serves as a double check. If the INSTALL is destined for one system and accidentally runs on another the name will not match that of the system and the INSTALL will fail.
- ACTIVATE Y| N |hh.mm – this parameter indicates whether we should also ACTIVATE on the target system after an INSTALL. (It is ignored if FUNC=EXTRACT.) If we specify hh.mm, the ACTIVATE will be delayed until time hh.mm. If we specify ACTIVATE=Y activation occurs immediately following the INSTALL.
- POL – specifies the name of the policy to be ACTIVATEd if ACTIVATE=Y or ACTIVATE=hh.mm.

  COPY=ALL – if this is specified, the entire service definition is INSTALLed (and potentially ACTIVATEd) into the target system and all SYSIN control statements are ignored.

The following DDcards are used:

- WLMFILE – the work file. This is the file the EXTRACT function writes into and the INSTALL function reads from. It is the transportable data file of format LRECL=80 and RECFM=FB, and could be a PDS member.
- MSGLOG – this is used with the INSTALL function. Say there is a definition for scheduling environmens SE1 with scheduling resources SR1 and SR2. Inside the WLM data we will then have SE1/SR1and SE1/SR2 as paired definitions. If the user only specifies scheduling resource SR1, we will have to delete the SE1/SR2 combination when INSTALLing because it will cause an inconsistency. The program will do this, following which a warning message is written to MSGLOG. If this occurs then RC=4.
- SYSIN – this is a list of all the systems with their required scheduling environments and scheduling resource names. When we run the job with an INSTALL and SYSTEM=xxx parameter,

system xxx's required names are located from here and then INSTALLed from the WLM data contained in the WLMFILE. The control statements have to be coded in sets consisting of a SYSTEM= card followed by one or more SCHENV= cards, followed by one or more RESOURCE= cards. The format is free and one or more spaces can be kept between the names; names can also be spread over multiple cards. Here is an example – systems P01 and P02 both require scheduling environments PSCHED1 and PSCHED2 and scheduling resources PRODRUN1 and PRODRUN2. System T01 has a scheduling environment of TSCHED1 with scheduling resources TSTRUN1, TSTRUN2, and TSTRUN3:

```
//SYSIN DD *
SYSTEM= PØ1 PØ2
SCHENV=PSCHED1 PSCHED2
RESOURCE=PRODRUN1 PRODRUN2
SYSTEM=TØ1
SCHENV=TSCHED1
RESOURCE=TSTRUN1    TSTRUN2
RESOURCE=TSTRUN3
```

Note that the program was written to accept system names of three bytes; small alterations may be required if your site uses system names with a different length. The program also calls a general JCL-PARM scan utility called PARMSCAN, the source of which is supplied. This parameter scan utility can also be used elsewhere for other programs that accept parameters from JCL.

PARMSCAN CSECT

```
PARMSCAN CSECT
PARMSCAN AMODE 31
PARMSCAN RMODE ANY
*********************************************************************
* This routine locates a keyword parameter in a passed JCL parm. The
* name of the keyword parameter is passed at KEYWORD@, the length of
* the value allowed is in KEYWORDL. The caller specifies the address of
* where the keyword value must be put in VALUE@ and the maximum
* allowable length of the value is specified in VALUEL. If the keyword
* value is found and the length does not exceed the max, RC=ØØ.
* If the value is too long then RC=Ø4. If the keyword was not specified
* (found) then RC=Ø8 and VALUEL is set to Ø.
* Upon entry:
*             R1 = pointer to JCL card
*             R2 = pointer to DESCRIPT DSECT (describing our parms)
*********************************************************************
```

```
         BAKR  14,Ø
         LR    R12,R15
         USING PARMSCAN,R12
         LR    R4,R1               Preserve pointer to JCL parm
         LR    R5,R2               Preserve pointer descriptor
         USING DESCRIPT,R5
         LA    R3,STORSIZE         Our required work area
         STORAGE OBTAIN,LENGTH=(3),LOC=ANY
         LR    R2,R1               Point to getmained area
         XR    R9,R9
         MVCL  R2,R8               Propagate binary zeros
         LR    R13,R1
         USING STORAREA,R13
         BAS   R14,PROCPARM        Go process passed parms
         TM    FOUNDIT,YES         Did we find the parameter?
         BO    FREESTOR            Yes
         OC    RCODE,=F'8'         No, set return code to 8
         XC    VALUEL,VALUEL       Set return length of parm to Ø
FREESTOR LA    R3,STORSIZE         Size of area to free
         LR    R2,R13              Address of area to free
         L     R4,RCODE            Pick up the return code
         STORAGE RELEASE,LENGTH=(R3),ADDR=(R2)
         LR    R15,R4              Pick up the return code
         PR
*********************************************************************
*        This section processes the passed parm
*********************************************************************
PROCPARM BAKR  R14,Ø                Stack caller's status
         LH    R3,Ø(R4)            Length of passed parm
         ST    R3,PARMLENG
         LA    R4,2(R4)            Point to actual parameter data
         ST    R4,PARM@            Store the parm data address
         AR    R4,R3               Add length to start of parm...
         ST    R4,LASTCHAR         Parameter goes up to here
         LTR   R3,R3               Passed parm length must be > Ø
         BZ    PROCPARX            Nothing further to process
         L     R3,PARMLENG         Length of passed parm
         SH    R3,KEYWORDL         Subtract the length of the keyword
         BNP   PROCPARX            Get out if negative
         L     R4,PARM@            Address of passed parameter data
KEYWRDLP EQU   *
         LR    R6,R4
         L     R8,KEYWORD@         Point to start of keyword
         LH    R7,KEYWORDL         The length of the keyword to find
         LR    R9,R7               Copy length for CLCL
         CLCL  R6,R8               See if parm parameter was spec.
         BE    FNDKEYW             Yes, it was
         LA    R4,1(R4)            Bump up pointer
         BCT   R3,KEYWRDLP         Scan entire text
         B     PROCPARX
FNDKEYW  CLI   Ø(R6),C'='          Must be equal sign
         BNE   PROCPARX            Get out if not
         LA    R6,1(R6)            Point past equal sign
```

```
         ST    R6,START@           Where the parm starts
         XR    R7,R7               Counter to indicate length of parm
ENDLOOP  C     R6,LASTCHAR         Have we reached the end of parm?
         BH    GOTEND              Yes, get out of loop
         CLI   Ø(R6),C' '          Did we find a blank?
         BE    GOTEND              Yes, get out of loop
         CLI   Ø(R6),X'ØØ'         Did we find binary zeros?
         BE    GOTEND              Yes, get out of loop
         CLI   Ø(R6),C','          Did we find a comma?
         BE    GOTEND              Yes, get out of loop
         LA    R7,1(R7)            Bump up character counter
         LA    R6,1(R6)            Point to next character
         B     ENDLOOP             Repeat loop for all characters
GOTEND   LTR   R7,R7               Found a value for the keyword?
         BZ    PROCPARX            No, get out
         CH    R7,VALUEL           Does parm length exceed max length?
         BNH   MOVEPARM            No
         OC    RCODE,=F'4'         RC=4 if too long
         LH    R7,VALUEL           Reduce length to allowed maximum
MOVEPARM STH   R7,VALUEL           Return length to caller
         LR    R9,R7               Copy length for MVCL
         L     R6,START@           Last character for keyword
         L     R8,VALUE@           Where we want the parm
         MVCL  R8,R6               Move the value in
         OI    FOUNDIT,YES         Set flag to show we found parm
PROCPARX PR                        Return to caller
*********************************************************************
*        Constants
*********************************************************************
         LTORG
*********************************************************************
*        DSECTS
*********************************************************************
STORAREA DSECT
PARMLENG DS    F                    .Length of passed parameter
PARM@    DS    F                    .Address of passed parameter
START@   DS    F                    .Address of keyword value
LASTCHAR DS    F                    .Address of last parm character
RCODE    DS    F                    .Return code
FOUNDIT  DS    C                    .Flag
YES      EQU   X'8Ø'
STORSIZE EQU   *-STORAREA           .Length of our required work area
DESCRIPT DSECT                      .Our input parameters
KEYWORD@ DS    F                    .Address of keyword
KEYWORDL DS    H                    .Length of keyword
VALUEL   DS    H                    .Maximum/ actual length of value
VALUE@   DS    F                    .Where we want the result
RØ       EQU   Ø
R1       EQU   1
R2       EQU   2
```

Editor's note: Insert register equates here.

```
         END
```

WLMEXTRT CSECT

```
WLMEXTRT CSECT
WLMEXTRT AMODE 31
WLMEXTRT RMODE 24
         BAKR  R14,Ø                .Save Caller's Status
         USING WLMEXTRT,R15
         LR    R11,R15
         L     R12,OFFSET
         DROP  R15
         USING WLMEXTRT,11,12
         B     STARTPT
OFFSET   DS    ØF
         DC    AL4(WLMEXTRT+4Ø96)
*********************************************************************
*        Main driver routine
*********************************************************************
STARTPT  DS    ØF
         L     R5,Ø(R1)             .Preserve passed parm pointer
STORAGE  LA    R3,GETMSIZE          .Size of storage to get and clear
         A     R3,TABSIZE           .Add size of buffer area
         STORAGE OBTAIN,LENGTH=(3),LOC=BELOW
         LR    R2,R1                .Point to getmained area
         XR    R9,R9
         XR    R8,R8
         MVCL  R2,R8                .Propagate binary zeros
         USING GETMAREA,R1
         ST    R13,SAVEAREA+4
         B     SETR13
         DROP  R1
         DS    ØF
SETR13   LR    R13,R1
         USING GETMAREA,R13         .Addressability to getmained area
         ST    R5,OURPARM@          .Store our parm address
         BAS   R14,GETPARM          .Analyse input parms
         CH    R15,=H'4'            .Successful?
         BH    RETURN               .No, get out
         TM    ACTION,FROMWLM       .From WLM into a dataset?
         BNO   STAGE                .No, from data set to WLM
DOWNLOAD BAS   R14,EXTRACT          .Go get the WLM data
         BAS   R14,TDATASET         .Go write the data into a dataset
         B     RETURN               .Get out
STAGE    BAS   R14,FDATASET         .Go get the WLM data from the ds.
         TM    COPYALL,YES          .Must we copy entire content?
         BO    GODOINST             .Yes, no further scanning required
         BAS   R14,GETSYSIN         .Obtain list of req'ed SE from parm
         LTR   R15,R15              .Any SE names specified?
         BNZ   RETURN               .No, get out
         BAS   R14,BLDNEWSE         .Replace SE table with req'ed names
         LTR   R15,R15              .Did we build a table?
         BNZ   RETURN               .No, get out
         BAS   R14,VERIFYSE         .Make sure all req'ed entries found
         BAS   R14,BLDNEWRE         .Reduce resource table as per parms
```

```
         LTR   R15,R15              .Did we build a table?
         BNZ   RETURN               .No, get out
         BAS   R14,VERIFYRE         .Make sure all req'ed entries found
         BAS   R14,XCHECKSE         .X check SE/SR table with SE list
         LTR   R15,R15              .Did we build a table?
         BNZ   RETURN               .No, get out
         BAS   R14,XCHECKRE         .X check SE/SR table with resources
         LTR   R15,R15              .Did we build a table?
         BNZ   RETURN               .No, get out
GODOINST BAS   R14,INSTALL          .Go write the data into WLM
         TM    ACTIVATE,YES         .Must we also ACTIVATE?
         BNO   RETURN               .No, get out
         TM    DELAY,YES            .Must we delay before activating?
         BNO   ACTIVNOW             .No
WTODELAY WTO   'WLMEXTRT(I): -WLM ACTIVATE will be delayed until hh:mm:X
               ØØ',ROUTCDE=11
         LA    R2,WAITPARM          .Point to GMT format wait parm
         STIMER WAIT,LT=(R2)        .Delay the ACTIVATE
ACTIVNOW BAS   R14,WLMACTIV         .Go ACTIVATE the policy
RETURN   TM    LOGOPEN,YES          .Is the LOG file open?
         BNO   LOADRC               .No
         CLOSE MSGLOG
         WTO   'WLMEXTRT(W): -Refer to warnings on MSGLOG',ROUTCDE=11
LOADRC   L     R4,RETCODE           .Pick up retrun code
         LR    R2,R13               .Pointer to storage area
         LA    R3,GETMSIZE          .Size of storage to free
         A     R3,TABSIZE           .Add size of table
         STORAGE RELEASE,LENGTH=(3),ADDR=(2)
         LR    R15,R4               .Reload return code
         PR                         .Back to our caller
*********************************************************************
*        This routine analyses the input parameter
*********************************************************************
GETPARM  BAKR  R14,Ø                .Preserve caller's registers
         LA    R1,KEYWORD           .The keyword we are looking for
         ST    R1,KEYWORD@          .Plug the address
         LA    R1,VALUE             .Where we want the parameter
         ST    R1,VALUE@            .Plug the address
GETFUNC  MVC   KEYWORD(8),=C'FUNC'
         MVC   KEYWORDL,=H'4'       .Length of keyword parameter
         MVC   VALUEL,=H'7'         .Length of INSTALL or EXTRACT
         LA    R2,PARMANLZ          .Parms for PARMSCAN
         L     R1,OURPARM@          .The JCL parm passed to us
         LINK  EP=PARMSCAN          .Call parm analyzer
         L     R1,RETCODE           .Load existing return code
         AR    R1,R15               .Add latest
         ST    R1,RETCODE           .Plug it back
         CH    R15,=H'4'            .Successful?
         BH    FUNCERR              .No, terminate
         LH    R1,VALUEL            .Did we get the parm?
         LTR   R1,R1                .See if it has a length > Ø
         BZ    FUNCERR              .No, error
```

```
GOTFUNC  CLC   VALUE(7),=C'INSTALL' .Requesting an INSTALL?
         BNE   CHKEXTRT            .No, go see if it is EXTRACT
         OI    ACTION,TOWLM        .Set flag
         B     SHOWFUNC            .Go give a WTO
CHKEXTRT CLC   VALUE(7),=C'EXTRACT' .Requesting an EXTRACT?
         BNE   FUNCERR             .No, get out
         OI    ACTION,FROMWLM      .Set flag
         B     SHOWFUNC            .Go give a WTO
FUNCERR  WTO   'WLMEXTRT(E): -FUNCTION must be specified as INSTALL or X
               EXTRACT',ROUTCDE=11
         LA    R15,12              .Set the return code to 12
         A     R15,RETCODE         .Add to existing return code
         ST    R15,RETCODE         .Plug it
         B     GETPARMX            .Get out
SHOWFUNC MVC   FUNCWTO+22(7),VALUE
FUNCWTO  WTO   'WLMEXTRT(I): -xxxxxxx function to be performed',       X
               ROUTCDE=11
         TM    ACTION,FROMWLM      .Are we extracting?
         BO    GETPARMX            .No further parms required
GETSYS   MVC   KEYWORD(6),=C'SYSTEM'
         MVC   KEYWORDL,=H'6'      .Length of keyword parameter
         MVC   VALUEL,=H'3'        .Length of INSTALL or EXTRACT
         LA    R2,PARMANLZ         .Parms for PARMSCAN
         L     R1,OURPARM@         .The JCL parm passed to us
         LINK  EP=PARMSCAN         .Call parm analyzer
         ST    R15,RETCODE         .Plug the return code
         LTR   R15,R15             .Successful?
         BNZ   SYSERR              .No, terminate
         CLC   VALUEL,=H'3'        .System name must be 3 bytes long
         BNE   SYSERR              .No, error
         MVC   SYSNAME,VALUE       .Move the system name
         B     ACTIVPRM            .Go get the ACTIVATE= keyword
SYSERR   WTO   'WLMEXTRT(E): -A system name of 3 bytes must be specifieX
               d',ROUTCDE=11
         LA    R15,12              .Set return code to 12..
         ST    R15,RETCODE         .Plug it
         B     GETPARMX            .Get out
ACTIVPRM MVC   KEYWORD(8),=C'ACTIVATE'
         MVC   KEYWORDL,=H'8'      .Length of keyword parameter
         MVC   VALUEL,=H'5'        .Length of INSTALL or EXTRACT
         LA    R2,PARMANLZ         .Parms for PARMSCAN
         L     R1,OURPARM@         .The JCL parm passed to us
         LINK  EP=PARMSCAN         .Call parm analyzer
         CH    R15,=H'4'           .Successful?
         BH    ACTIVDFL            .No, assume default value
CHKACTV  CLI   VALUE,C'Y'          .Must we activate?
         BNE   CHKNACTV            .No, see if NO was specified
         OI    ACTIVATE,YES        .Set the flag
         TM    ACTION,TOWLM        .Are we installing as well?
         BO    GETPOLNM            .Yes, go get the WLM policy name
         WTO   'WLMEXTRT(E): -ACTIVATE=YES conflicts with FUNC=EXTRACT'X
               ,ROUTCDE=11
         B     SETRC12             .Set the return code to 16
```

```
CHKNACTV CLI   VALUE,C'N'           .No activate?
         BE    ACTIVDFL            .Yes, get out
CHKDACTV CLC   VALUEL,=H'5'        .The length of HH.MM
         BNE   ACTIVERR            .No, error
         TM    ACTION,TOWLM        .Are we installing as well?
         BO    DELAYWTO            .No
         WTO   'WLMEXTRT(E): -Delayed ACTIVATE conflicts with FUNCTION=X
               EXTRACT',ROUTCDE=11
         B     SETRC12             .Set return code to 16
DELAYWTO OI    ACTIVATE,Yes        .ACTIVATE requested
         OI    DELAY,YES           .Request is for delayed activate
         BAS   R14,CALCDLAY        .Go calculate the delay
         LTR   R15,R15             .Acceptable format?
         BNZ   ACTIVERR            .No
         MVC   WTODELAY+57(5),VALUE .Move time into the WTO
         B     GETPOLNM            .Go get the WLM policy name
ACTIVERR WTO   'WLMEXTRT(E): -Invalid ACTIVATE value specified, must beX
                Y(es), N(o) or hh:mm',ROUTCDE=11
SETRC12  LA    R15,12              .Set the return code to 12
         A     R15,RETCODE         .Add to existing return code
         ST    R15,RETCODE         .Plug it
         B     GETPARMX            .Get out
GETPOLNM MVC   KEYWORD(3),=C'POL'
         MVC   KEYWORDL,=H'3'      .Length of keyword parameter
         MVC   VALUEL,=H'8'        .Length of policy name
         LA    R2,PARMANLZ         .Parms for PARMSCAN
         L     R1,OURPARM@         .The JCL parm passed to us
         LINK  EP=PARMSCAN         .Call parm analyzer
         LTR   R15,R15             .Successful?
         BNZ   POLERR              .No, error
CHKPOL   MVC   POLNAME,VALUE       .Move into polname variable
         OC    POLNAME,=8X'4Ø'     .Make uppercase
         LH    R1,VALUEL           .Length of policy name
         BCTR  R1,Ø                .Reduce by 1
         STC   R1,TRTPOL+1         .Update length of instruction
TRTPOL   TRT   POLNAME,VALCHARS    .Make sure valid format
         BZ    SCANCOPY            .Polname acceptable
POLERR   WTO   'WLMEXTRT(I): -POL= parameter not specified or contains X
               invalid format.',ROUTCDE=11
         B     SETRC12             .RC=12 and terminate
ACTIVDFL WTO   'WLMEXTRT(I): -ACTIVATE=NO taken from parm/ default',   X
               ROUTCDE=11
SCANCOPY MVC   KEYWORD(8),=C'COPY'
         MVC   KEYWORDL,=H'4'      .Length of keyword parameter
         MVC   VALUEL,=H'3'        .Length of INSTALL or EXTRACT
         LA    R2,PARMANLZ         .Parms for PARMSCAN
         L     R1,OURPARM@         .The JCL parm passed to us
         LINK  EP=PARMSCAN         .Call parm analyser
         LTR   R15,R15             .Specified?
         BNZ   GETPARMX            .No
         CLC   VALUE(3),=C'ALL'    .Must we copy all
         BNE   INVLALL             .Invalid parameter
```

```
         OI    COPYALL,YES          .Set flag
         WTO   'WLMEXTRT(I): -Entire Service Definition to be INSTALLedX
               ',ROUTCDE=11
         B     GETPARMX
INVLALL  WTO   'WLMEXTRT(W): -Invalid COPY= parameter specified, ignoreX
               d',ROUTCDE=11
         LA    R15,4
         ST    R15,RETCODE           .Plug it back
GETPARMX L     R15,RETCODE           .Pick up return code
         PR                         .Restore caller's registers
*********************************************************************
*        This routine extracts the data from WLM
*********************************************************************
EXTRACT  BAKR  R14,Ø
         LA    R2,WLMTABLE          .Where we want the data
         LA    R3,TABSIZE           .Size of output area
         LA    R4,QUERYLEN          .Returned length of data
         IWMDEXTR ANSAREA=(R2),ANSLEN=(R3),QUERYLEN=(R4),              X
               RSNCODE=RSNCODE      .Don't overlay RETCODE here!
         LTR   R15,R15              .Succeeded?
         BZ    EXTRACTX             .Yes
         ST    R15,RETCODE
         BAS   R14,CODEPRNT         .Go make RC and REASON codes prt.
         MVC   EXTRWTO+25(4),PrtRC
         MVC   EXTRWTO+38(4),PrtRSN
EXTRWTO  WTO   'WLMEXTRT(E): -RC=xxxx, REASON=xxxx from IWMDEXTR',     X
               ROUTCDE=11
         ABEND ØØØ1
EXTRACTX PR
*********************************************************************
*        This routine installs the data from the buffer into WLM
*********************************************************************
INSTALL  BAKR  R14,Ø                .Preserve caller's registers
         MVC   SVIDSSVP_NAME,=CL8'WLMEXTRT'
         STCK  SVIDSSVP_TIMESTAMP   .Time stamp the update
         LA    R2,WLMTABLE          .Where the WLM data has been loaded
         LA    R4,PRODID            .Our ID
         LA    R3,ERROFF            .Where we want the error offset
         IWMDINST COND=NO,SERVD_AREA=(R2),VALCHECK_OFFSET=(R3),        X
               PRODUCT_ID=(R4),VALCHECK_RSN=Reason,                    X
               RSNCODE=RSNCODE      .Don't overlay RETCODE here!
         LTR   R15,R15              .Succeeded?
         BZ    INSTALOK             .Yes
         ST    R15,RETCODE
         BAS   R14,CODEPRNT         .Go make RC and REASON codes prt.
         MVC   INSTWTO+25(4),PRTRC  .Move return code into WTO
         MVC   INSTWTO+38(4),PRTRSN .Move reason code into WTO
InstWTO  WTO   'WLMEXTRT(E): -RC=xxxx, REASON=xxxx from IWMDINST',     X
               ROUTCDE=11
         CLC   RSNCODE+2(2),=X'Ø83D' Do we have a validity error?
         BNE   ABINST                .No, other error
         MVC   VALWTO+58(8),PRTERROF
```

```
         MVC   VALWTO+43(4),PRTREASN
VALWTO   WTO   'WLMEXTRT(I): -Validity reason code xxxx at offset xxxxxX
               xxx',ROUTCDE=11
         LA    R1,WLMTABLE          .Point to start of table
         A     R1,ERROFF            .Add the offset
         C     R1,OLDSETB@          .Try to determine where error is
         BL    SHOWMEM              .Address too low to determine
CHKSE    C     R1,OLDSRTB@          .In SE Area?
         BNL   CHKSR                .No
         MVC   WHEREWTO+5Ø(32),=C'Scheduling Environment (SVSEASE)'
         B     WHEREWTO             .Go do the WTO
CHKSR    C     R1,OLDRETB@          .In SR Area?
         BNL   INRE                 .No, must be in resource area
         MVC   WHEREWTO+5Ø(32),=C'Sched Env/ Resource (SVSEASR)   '
         B     WHEREWTO             .Go do the WTO
INRE     MVC   WHEREWTO+5Ø(32),=C'Resource Definition (SVSEARE)   '
WHEREWTO WTO   'WLMEXTRT(I): -Error is in area describing              X
                                    ',ROUTCDE=11
SHOWMEM  LA    R1,WLMTABLE          .Point to start of table
         A     R1,ERROFF            .Add the offset
         LA    R2,31(R1)            .Show 32 bytes
         LINK  EP=SHOWMEM           .Show the data in error
         STIMER WAIT,DINTVL=WAITWTO .Make sure WTO's come out
ABINST   ABEND ØØØ2,DUMP
WAITWTO  DC    C'ØØØØØ2ØØ'          .Wait for 2 seconds
INSTALOK WTO   'WLMEXTRT(I): -WLM data has been INSTALLed',            X
               ROUTCDE=11
INSTALLX PR                         .Restore caller's registers
*********************************************************************
*        This routine gets applicable shecduling environment and
*        resource names from the SYSIN DDcard
*********************************************************************
GETSYSIN BAKR  R14,Ø                .Preserve caller's registers
         BAS   R14,BLANKTBS         .Go move spaces into work tables
SYSTEMNM BAS   R14,SYSINSYS         .Match system name with SYSIN
         TM    GOTSYSNM,YES         .Did we find our system name?
         BO    SCHDENVS             .Yes
         WTO   'WLMEXTRT(E): -This system''s name not specified with a X
               SYSTEM= parameter',ROUTCDE=11
         LA    R15,8                .Set return code to 8
         ST    R15,RETCODE
         B     GETSYSIX             .No, get out
SCHDENVS BAS   R14,SYSINSE          .Get sched envns
         TM    GOTENVS,YES          .Did we get sched envs?
         BNO   NOSCHENV             .No
         XR    R15,R15              .Clear return code
         B     RESOURCE
NOSCHENV WTO   'WLMEXTRT(E): -No SCHENV= specified for this system',   X
               ROUTCDE=11
         LA    R15,8                .Set return code to 8
         ST    R15,RETCODE
         B     GETSYSIX             .No, get out
RESOURCE BAS   R14,SYSINRE          .Get resources
```

```
         TM    GOTRESCS,YES         .Did we get resources?
         BNO   NORESCRC             .No
         XR    R15,R15              .Clear return code
         B     GETSYSIX             .Yes, got everything
NORESCRC WTO   'WLMEXTRT(E): -No RESOURCE= specified for this system', X
               ROUTCDE=11
         LA    R15,8                .Set return code to 8
         ST    R15,RETCODE
GETSYSIX PR                         .Restore caller's registers
*********************************************************************
*        This routine matches SYSTEM in parm with SYSTEM in SYSIN
*********************************************************************
SYSINSYS BAKR  R14,Ø                .Preserve caller's registers
         LA    R1,SYSINSYX          .Where we should go on EOF SYSIN
         ST    R1,EOFADDR           .Plug the address for EOF routine
         OPEN  SYSIN
GETSYSTM GET   SYSIN                .Get the NEXT SYSIN card
         LA    R2,72                .8Ø bytes - L'(SYSTEM=x)
SYSINLP  CLC   Ø(7,R1),=C'SYSTEM='
         BNE   BUMPUP
         LA    R1,1(R1)             .Point to next character
         BCTR  R2,Ø                 .reduce loop counter by 1
SYSLOOP  CLC   Ø(3,R1),SYSNAME      .Does it match our system name?
         BE    GOTSYST              .Yes
         LA    R1,1(R1)             .Bump up pointer
         BCT   R2,SYSLOOP           .Scan rest of card
         B     GETSYSTM             .Card contains SYSTEM= but not us
BUMPUP   LA    R1,1(R1)             .Bump up pointer
         BCT   R2,SYSINLP           .Scan entire card
         B     GETSYSTM             .Card does not contain SYSTEM=
GOTSYST  OI    GOTSYSNM,YES         .Set the flag
SYSINSYX PR                         .Restore caller's registers
*********************************************************************
*        This routine picks up SCHENV= parameters from SYSIN.
*********************************************************************
SYSINSE  BAKR  R14,Ø
         LA    R1,WORKBUFF+8Ø
         ST    R1,LASTCHAR          .Last usable input char
         LA    R1,CHKNUM1           .Where we should go on EOF SYSIN
         ST    R1,EOFADDR           .Plug the address for EOF routine
GETSCHNV GET   SYSIN
         MVC   WORKBUFF(8Ø),Ø(R1)   .Move record into our buffer
         CLC   WORKBUFF(7),=C'SCHENV='  .Our card?
         BNE   CHKNUM1              .No, get out
         LA    R3,WORKBUFF+7        .Point to first char after SCHENV=
DEBLANK1 CLI   Ø(R3),X'4Ø'          .Is there a blank?
         BE    BUMPPTR1             .Yes
CHKPTR1  C     R3,LASTCHAR          .Reached end-of card?
         BNL   GETSCHNV             .Yes, get next card
ISOLAT1  TRT   Ø(L'SVSEA_SE_SCHENV_NAME+1,R3),ValChars
         BNZ   CHKEOC1              .Name is not too long
SENAMELN WTO   'WLMEXTRT(E): -Sched env name longer than 16 bytes',    X
               ROUTCDE=11
```

```
         ABEND Ø0Ø3
CHKEOC1  C     R1,LASTCHAR          .Blank/ X'ØØ' after end-of-card?
         BL    MOVE1                .No
         L     R1,LASTCHAR          .Stop at last character
MOVE1    LR    R6,R3                .Move from here
         LR    R7,R1                .Address of blank/comma etc.
         SR    R7,R3                .Address of start of name
         B     COPYLEN1
COPYLEN1 LR    R9,R7                .Copy length for MVCL
         L     R8,PRM#SE            .Number of entries in table
         MH    R8,=AL2(L'SVSEA_SE_SCHENV_NAME)
         LA    R2,PRMSETBL          .Point to start of table
         AR    R8,R2
         A     R8,PRM#SE            .First byte of ea. entry is a flag
         LA    R8,1(R8)             .Skip over first byte of this entry
         MVCL  R8,R6                .Move name into table
         L     R9,PRM#SE            .Number of env's so far
         LA    R9,1(R9)             .Bump up counter
         ST    R9,PRM#SE            .Plug it back
         C     R9,=AL4(MAXSCHED)    .Within limits?
         BL    UPR8_1               .Yes
         WTO   'WLMEXTRT(E): -Max # sched envs allowed exceeded. Up MAXX
               SCHED and re-assemble',ROUTCDE=11
         ABEND Ø0Ø4
UPR8_1   LA    R8,L'SVSEA_SE_SCHENV_NAME(R8) Bump up pointer
         LR    R3,R1                .Point past end of name
         B     CHKLST1
BUMPPTR1 LA    R3,1(R3)             .Bump up pointer to card
CHKLST1  C     R3,LASTCHAR          .Have we reached end-of-card?
         BNH   DEBLANK1             .No, go remove blanks
CHKNUM1  L     R8,PRM#SE            .Number of entries in table
         LTR   R8,R8                .Did we find any?
         BZ    SYSINSX              .No, get out
         OI    GOTENVS,YES          .Turn found-envs flag on
SYSINSX  PR
*********************************************************************
*        This routine picks up RESOURCE= parameters from SYSIN.
*********************************************************************
SYSINRE  BAKR  R14,Ø
         LA    R1,CHKNUM2           .Where we should go on EOF SYSIN
         ST    R1,EOFADDR           .Plug the address for EOF routine
         B     COMPRS               .Our first card read by SYSINSE rtn
GETRESRC GET   SYSIN
         MVC   WORKBUFF(8Ø),Ø(R1)   .Move record into our buffer
COMPRS   CLC   WORKBUFF(9),=C'RESOURCE='  .Our card?
         BNE   CHKNUM2              .No, get out
         LA    R3,WORKBUFF+9        .First char after RESOURCE=
DEBLANK2 CLI   Ø(R3),X'4Ø'          .Is there a blank?
         BE    BUMPPTR2             .Yes
CHKPTR2  C     R3,LASTCHAR          .Reached end-of card?
         BNL   GETRESRC             .Yes, get next card
ISOLAT2  TRT   Ø(L'SVSEA_RE_RESOURCE_NAME+1,R3),ValChars
```

```
         BNZ   CHKEOC2              .Name not too long
RENAMELN WTO   'WLMEXTRT(E): -Resource name longer than 16 bytes',     X
               ROUTCDE=11
         ABEND ØØØ5
CHKEOC2  C     R1,LASTCHAR          .Blank/ X'ØØ' after end-of-card?
         BL    MOVE2                .No
         L     R1,LASTCHAR          .Stop at last character
MOVE2    LR    R6,R3                .Move from here
         LR    R7,R1                .Address of blank/comma etc.
         SR    R7,R3                .Address of start of name
         B     COPYLEN2
COPYLEN2 LR    R9,R7                .Copy length for MVCL
         L     R8,PRM#RE            .Number of entries in table
         MH    R8,=AL2(L'SVSEA_RE_RESOURCE_NAME)
         LA    R2,PRMRETBL          .Point to start of table
         AR    R8,R2
         A     R8,PRM#RE            .First byte of ea. entry is a flag
         LA    R8,1(R8)             .Skip over first byte of this entry
         MVCL  R8,R6                .Move name into table
         L     R9,PRM#RE            .Number of env's so far
         LA    R9,1(R9)             .Bump up counter
         ST    R9,PRM#RE            .Plug it back
         C     R9,=AL4(MAXRESRC)    .Within limits?
         BL    UPR8_2               .Yes
         WTO   'WLMEXTRT(E): -Max # resources allowed exceeded. Up MAXRX
               SRC and re-assemble',ROUTCDE=11
         ABEND ØØØ6
UPR8_2   LA    R8,L'SVSEA_SE_SCHENV_NAME(R8) Bump up pointer
         LR    R3,R1                .Point past end of name
         B     CHKLST2
BUMPPTR2 LA    R3,1(R3)             .Bump up pointer to card
CHKLST2  C     R3,LASTCHAR          .Have we reached end-of-card?
         BNH   DEBLANK2             .No, go remove blanks
CHKNUM2  L     R8,PRM#RE            .Number of entries in table
         LTR   R8,R8                .Did we find any?
         BZ    SYSINRX              .No, get out
         OI    GOTRESCS,YES         .Turn found-envs flag on
SYSINRX  PR
*********************************************************************
*        This routine closes the SYSIN file. This can occur from
*        either the SYSINSYS, SYSINSE or SYSINRE routines.
*********************************************************************
EOFSYSIN OI    EOFFLAG,YES          .Set flag to show EOF reached
         CLOSE SYSIN
         L     R14,EOFADDR          .Where we should branch
         BR    R14                  .Branch back into read routine
*********************************************************************
*        This routine writes data from the buffer into the data set
*********************************************************************
TDATASET BAKR  R14,Ø                .Preserve caller's registers
         OPEN  (OUTFILE,OUTPUT)
         LA    R2,WLMTABLE          .Point to where the data is
```

```
         L     R3,QUERYLEN
PUTLOOP  PUT   OUTFILE,(R2)         .Write an 8Ø-byte record
         LA    R2,8Ø(R2)            .Move "from" pointer 8Ø bytes
         SH    R3,=H'8Ø'            .Subtract 8Ø from the length
         BP    PUTLOOP              .Redo if bytes left
         CLOSE OUTFILE              .Close the output file
         WTO   'WLMEXTRT(I): -The EXTRACTed data has been written into X
               the output file',ROUTCDE=11
TDATASEX PR                         .Restore caller's registers
*********************************************************************
*        This routine selectively strips out scheduling environments
*********************************************************************
BLDNEWSE BAKR  R14,Ø                 .Preserve caller's registers
         USING SERVDHDR,WLMTABLE    .Addressability to SERVD
         LA    R2,WLMTABLE          .Point to start of table
         A     R2,SERVD_SVSEA_OFF   .Pointer to sched. env. defs.
         ST    R2,SCHDDEF@          .Remember this address
         L     R6,PRM#SE            .Max # sched. env. to copy
         MH    R6,SVSEASE_LEN       .Multiply by size of entry
         STORAGE OBTAIN,LOC=ANY,LENGTH=(6)
         ST    R1,NEWSETB@          .Store the address of getm'ed area
         ST    R6,NEWSETBL          .Store the length
CLEARIT1 LR    R2,R1                .Point to getmained area
         LR    R3,R6                .Load the length
         XR    R9,R9                .Set length to Ø
         XR    R8,R8                .Set dummy from address
         MVCL  R2,R8                .Propagate bin zeroes
         LR    R6,R1                .Point to getmained storage
         XR    R7,R7                .Clear matching sched. env. counter
         L     R2,SCHDDEF@          .Start of scheduling definitions
         USING SVSEAHDR,R2          .Addressability to sched. env. defs
         XR    R3,R3
         ICM   R3,3,SVSEA_NUMBER_SE .Pick up # of sched. env. defs.
         LR    R4,R2                .Start address of service defs
         A     R4,SVSEA_OFFSET_SE   .Offset of sched. env. section
         ST    R4,OLDSETB@          .Store this address
         USING SVSEASE,R4           .Addressability to sched. env.
* The entries in the table as built from disk are scanned against the
* list of entries supplied by the user, as for this system. Only the
* entries required are placed into a newly allocated table. The newly
* built SE table is then copied over the old SE table and the "number
* of scheduling environments" counter (SVSEA_EXT_NUM_SE) is also
* updated.
SCANLP1  LA    R8,PRMSETBL          .Point to first name passed in parm
         USING TBLDSECT,R8
         L     R9,PRM#SE            .Number passed in parm
SCANLP2  CLC   SVSEA_SE_SCHENV_NAME,TbSEName Matching tbl. entry?
         BNE   NXTTBENT             .Not matching No, skip
         MVC   Ø(SVSEASE_LEN,R6),SVSEA_SE_SCHENV_NAME Yes
         OI    TBENTFND,MATCHED     .Set flag to show entry found
         LA    R6,SVSEASE_LEN(R6)   .Bump up "to" pointer
         LA    R7,1(R7)             .Bump up "matched" counter
```

```
NXTTBENT LA    R8,1+L'SVSEA_SE_SCHENV_NAME(R8) Bump "from" pointer
         BCT   R9,SCANLP2           .Do for each parm entry
         LA    R4,SVSEASE_LEN(R4)   .Point to next sched env in table
         BCT   R3,SCANLP1           .Compare each with passed parm
         L     R4,OLDSETB@          .Start of (old) SE entries
         L     R6,NEWSETB@          .Start of (new) SE entries
         L     R2,SCHDDEF@          .Start of SE header
         LTR   R7,R7                .Did we find any of the SE names?
         BNZ   CHGNUMSE             .Yes, proceed
         WTO   'WLMEXTRT(E): -None of the specified SE names exist',   X
               ROUTCDE=11
         LA    R15,8                .Set the return code
         ST    R15,RETCODE          .Plug it
         B     BLDNEWSX             .Get out
CHGNUMSE STCM  R7,3,SVSEA_NUMBER_SE .Update number of SE entries
         MH    R7,=AL2(SVSEASE_LEN) .Number * length = total size
         BCTR  R7,Ø                 .Reduce length by 1 for MVCL
         LR    R5,R7                .Copy length for MVCL
         MVCL  R4,R6                .Overlay old data with new
BLDNEWSX L     R3,NEWSETBL          .Length of storage to free
         L     R2,NEWSETB@          .Address of storage to free
         STORAGE RELEASE,LENGTH=(3),ADDR=(2)
         PR                         .Restore caller's registers
*********************************************************************
*        This routine selectively strips out resource names
*********************************************************************
BLDNEWRE BAKR  R14,Ø                .Preserve caller's registers
         L     R2,SCHDDEF@          .Start of scheduling definitions
         L     R6,PRM#RE            .Max # sched. env. to copy
         MH    R6,SVSEARE_LEN       .Multiply by size of entry
         STORAGE OBTAIN,LOC=ANY,LENGTH=(6)
         ST    R1,NEWRETB@          .Store the address of getm'ed area
         ST    R6,NEWRETBL          .Store the length
CLEARIT2 LR    R2,R1                .Point to getmained area
         LR    R3,R6                .Get the length
         XR    R9,R9                .Set length to zero
         XR    R8,R8                .Set dummy from address
         MVCL  R2,R8                .Propagate binary zeros
         LR    R6,R1                .Point to getmained storage
         XR    R7,R7                .Count matching sched. env. entries
         L     R2,SCHDDEF@          .Start of scheduling definitions
         XR    R3,R3
         ICM   R3,3,SVSEA_NUMBER_RE .Pick up # of resource defs.
         LR    R4,R2                .Start address of resource defs
         A     R4,SVSEA_OFFSET_RE   .Offset of resource section
         ST    R4,OLDRETB@          .Store this address
         USING SVSEARE,R4           .Addressability to resource env.
* The entries in the table as built from disk are scanned against the
* list of entries supplied by the user, as for this system. Only the
* entries required are placed into a newly allocated table. The newly
* built RE table is then copied over the old RE table and the "number
* of resources" counter (SVSEA_EXT_NUM_RE) is also updated.
```

```
SCANLP3  LA    R8,PRMRETBL          .Point to first name passed in parm
         L     R9,PRM#RE            .Number passed in parm
SCANLP4  CLC   SVSEA_RE_RESOURCE_NAME,TbRERName Matching tbl. entry?
         BNE   NXTRESRC             .Not matching No, skip
         MVC   Ø(SVSEASE_LEN,R6),SVSEA_RE_RESOURCE_NAME Yes
         OI    TBENTFND,MATCHED     .Set flag to show entry found
         LA    R6,SVSEARE_LEN(R6)   .Bump up "to" pointer
         LA    R7,1(R7)             .Bump up "matched" counter
NXTRESRC LA    R8,1+L'SVSEA_RE_RESOURCE_NAME(R8) Bump "from" pointer
         BCT   R9,SCANLP4           .Do for each parm entry
         LA    R4,SVSEARE_LEN(R4)   .Point to next resource in table
         BCT   R3,SCANLP3           .Compare each with passed parm
         L     R4,OLDRETB@          .Start of (old) RE entries
         L     R6,NEWRETB@          .Start of (new) RE entries
         L     R2,SCHDDEF@          .Start of RE header
         LTR   R7,R7                .Did we find any of the RE names?
         BNZ   CHGNUMRE             .Yes, proceed
         WTO   'WLMEXTRT(E): -None of the specified RE names exist',   X
               ROUTCDE=11
         LA    R15,8                .Set the return code
         ST    R15,RETCODE          .Plug it
         B     BLDNEWRX             .Get out
CHGNUMRE STCM  R7,3,SVSEA_NUMBER_RE .Update number of SE entries
         MH    R7,=AL2(SVSEARE_LEN) .Number * length = total size
         BCTR  R7,Ø                 .Reduce length by 1 for MVCL
         LR    R5,R7                .Copy length for MVCL
         MVCL  R4,R6                .Overlay old data with new
BLDNEWRX L     R3,NEWRETBL          .Length of storage to free
         L     R2,NEWRETB@          .Address of storage to free
         STORAGE RELEASE,LENGTH=(3),ADDR=(2)
         PR
*********************************************************************
*        This routine removes SE/SR combinations that can no longer
*        exist because the SE definitions have been removed.
*********************************************************************
XCHECKSE BAKR  R14,Ø                .Preserve caller's registers
         L     R2,SCHDDEF@          .Start of scheduling definitions
         L     R6,PRM#SE            .Max # sched. env. to copy
         MH    R6,SVSEA_SIZE_SR     .Multiply with the size of 1 entry
         MH    R6,SVSEA_NUMBER_SR   .Maximum number of resources
         STORAGE OBTAIN,LOC=ANY,LENGTH=(6)
         ST    R1,NEWSRTB@          .Store the address of getm'ed area
         ST    R6,NEWSRTBL          .Store the length
CLEARIT3 LR    R2,R1                .Point to getmained area
         LR    R3,R6                .Get the length
         XR    R9,R9                .Set length to zero
         XR    R8,R8                .Set dummy from address
         MVCL  R2,R8                .Propagate binary zeroes
         LR    R6,R1                .Point to getmained storage
         XR    R7,R7                .Count matching SE/SR entries
         L     R2,SCHDDEF@          .Start of scheduling definitions
         XR    R3,R3
```

```
         ICM   R3,3,SVSEA_NUMBER_SR .Pick up # of SE/SR defs.
         LR    R4,R2                .Start address of service defs
         A     R4,SVSEA_OFFSET_SR   .Offset of SE/SR section
         ST    R4,OLDSRTB@          .Store this address
         USING SVSEASR,R4           .Addressability to SE/SR section
SCANLP5  LA    R8,PRMSETBL          .Point to first SR passed in parm
         L     R9,PRM#SE            .Number of SR passed in parm
SCANLP6  CLC   SVSEA_SR_SCHENV_NAME,TbSEName Matching tbl. entry?
         BNE   NXTTBLEN             .Not matching, skip
         MVC   Ø(SVSEASR_LEN,R6),SVSEA_SR_SCHENV_NAME Yes
         LA    R6,SVSEASR_LEN(R6)   .Bump up "to" pointer
         LA    R7,1(R7)             .Bump up "matched" counter
         B     NXTSESR1             .Get next SE/SR
NXTTBLEN LA    R8,1+L'SVSEA_SR_SCHENV_NAME(R8) Bump "from" pointer
         BCT   R9,SCANLP6           .Do for each parm entry
NXTSESR1 LA    R4,SVSEASR_LEN(R4)   .Point to next SE/SR in table
         BCT   R3,SCANLP5           .Compare each with passed parm
         L     R4,OLDSRTB@          .Start of (old) SR entries
         L     R6,NEWSRTB@          .Start of (new) SR entries
         L     R2,SCHDDEF@          .Start of sched env header
         STCM  R7,3,SVSEA_NUMBER_SR .Update number of SE entries
         LTR   R7,R7                .Did we find any of the SE names?
         BNZ   MOVESESR             .Yes, proceed
         WTO   'WLMEXTRT(W): -There are no Scheduling Resources matchinX
               g the requested Scheduling Environments',ROUTCDE=11
         OC    RETCODE,=F'4'        .Set the return code
         B     XCHECKSX             .Go free the table and return
MOVESESR MH    R7,=AL2(SVSEASR_LEN) .Number * length = total size
         BCTR  R7,Ø                 .Reduce length by 1 for MVCL
         LR    R5,R7                .Copy length for MVCL
         MVCL  R4,R6                .Overlay old data with new
XCHECKSX L     R3,NEWSRTBL          .Length of storage to free
         L     R2,NEWSRTB@          .Address of storage to free
         STORAGE RELEASE,LENGTH=(3),ADDR=(2)
         L     R15,RETCODE          .Load the return code
         PR                         .Restore caller's registers
*********************************************************************
*        This routine writes an entry to the log to indicate that an
*        SE/SR combination was deleted as the SR (resource) was not
*        selected on the input parms
*********************************************************************
NOMTCHLG BAKR  R14,Ø                .Preserve caller's registers
         LR    R4,R1                .Pointer to parm as passed
         TM    LOGOPEN,YES          .Has the LOG file been opened?
         BO    WRTLOG               .Yes, go write the entry
         TM    NOLOGDD,YES          .No log DD-card?
         BO    NOMTCHLX             .Yes, get out
         LA    R1,JFCBAREA          .Address of JFCB work area
         STCM  R1,7,JFCBPTR+1       .Plug it into JFCB pointer
         MVI   JFCBPTR,X'87'
         LA    R1,JFCBPTR
         STCM  R1,7,MSGLOG+37       .Plug JFCB pointer @ into DCB
```

```
         RDJFCB MSGLOG
         LTR   R15,R15             .Is the DD card present?
         BZ    OPENLOG             .Yes, go open it
         OI    NOLOGDD,YES         .Remember DD card not present
         B     NOMTCHLX            .Get out
OPENLOG  OPEN  (MSGLOG,OUTPUT)
         OI    LOGOPEN,YES         .Remember it is already open
WRTLOG   EQU   *
         MVC   DISCARD(16),SVSEA_SR_SCHENV_NAME
         MVI   DISCARD+16,C'/'
         MVC   DISCARD+17(16),SVSEA_SR_RESOURCE_NAME
         LA    R1,L'DISCARD        .Length of Discard
         LA    R2,DISCARD          .Start of SE/SR
         LA    R3,DISCARD          .Start of SE/SR
DEBLANK  CLI   Ø(R2),C' '          .Is it a blank?
         BE    NEXTCHAR            .Yes, skip it
         CLI   Ø(R2),X'ØØ'         .Is it binary zeroes?
         BE    NEXTCHAR            .Yes, skip it
         MVC   Ø(1,R3),Ø(R2)       .No, move the character
         LA    R3,1(R3)            .Bump up "to" pointer
NEXTCHAR LA    R2,1(R2)            .Bump up "from" pointer
         BCT   R1,DEBLANK          .Do for each character
*        LA    R3,1(R3)            .First not-used char
         LA    R4,LOGBUFFE         .End of buffer
         SR    R4,R3               .Length to clear
         STC   R4,CLEARREC+1
CLEARREC XC    Ø(Ø,R3),Ø(R3)       .Clear it
PUTREC   PUT   MSGLOG,LOGBUFFR
NOMTCHLX PR                       .Preserve caller's registers
LOGBUFFR DS    ØCL8Ø
MSGCONST DC    C'SE/SR REMOVED AS NO SR specified: '
DISCARD  DS    CL(8Ø-L'MSGCONST)
LOGBUFFE EQU   *
*********************************************************************
*        This routine removes SE/SR combinations that can no longer
*        exist because the resource definitions have been removed.
*********************************************************************
XCHECKRE BAKR  R14,Ø               .Preserve caller's registers
         L     R2,SCHDDEF@         .Start of scheduling definitions
         L     R6,PRM#SE           .Max # sched. env. to copy
         MH    R6,SVSEA_SIZE_SR    .Multiply with the size of 1 entry
         MH    R6,SVSEA_NUMBER_SR  .Maximum number of resources
         STORAGE OBTAIN,LOC=ANY,LENGTH=(6)
         ST    R1,NEWSRTB@         .Store the address of getm'ed area
         ST    R6,NEWSRTBL         .Store the length
CLEARIT4 LR    R2,R1               .Point to getmained area
         LR    R3,R6               .Get the length
         XR    R9,R9               .Set length to zero
         XR    R8,R8               .Set dummy from address
         MVCL  R2,R8               .Propagate binary zeroes
         LR    R6,R1               .Point to getmained storage
         XR    R7,R7               .Count matching SE/SR entries
```

```
         L     R2,SCHDDEF@          .Start of scheduling definitions
         XR    R3,R3
         ICM   R3,3,SVSEA_NUMBER_SR .Pick up # of sched. env. defs.
         LR    R4,R2                .Start address of service defs
         A     R4,SVSEA_OFFSET_SR   .Offset of sched. env. section
SCANLP7  LA    R8,PRMRETBL          .Point to first name passed in parm
         L     R9,PRM#RE            .Number passed in parm
SCANLP8  CLC   SVSEA_SR_RESOURCE_NAME,TbRERName Matching tbl. entry?
         BNE   NXTTABEN             .Not matching, skip
         MVC   Ø(SVSEASR_LEN,R6),SVSEA_SR_SCHENV_NAME Yes
         LA    R6,SVSEASR_LEN(R6)   .Bump up "to" pointer
         LA    R7,1(R7)             .Bump up "matched" counter
         B     NXTSESR2             .Go do the next SE/SR entry
NXTTABEN LA    R8,1+L'SVSEA_SR_RESOURCE_NAME(R8) Bump "from" pointer
         BCT   R9,SCANLP8           .Do for each parm entry
         LA    R1,SVSEA_SR_SCHENV_NAME No match found for this SE/SR
         BAS   R14,NOMTCHLG         .Go write an entry to the log
         OC    RETCODE,=F'4'        .Set RC=4
NXTSESR2 LA    R4,SVSEASR_LEN(R4)   .Point to next SE/SR in table
         BCT   R3,SCANLP7           .Compare each with passed parm
         L     R4,OLDSRTB@          .Start of (old) SR entries
         L     R6,NEWSRTB@          .Start of (new) SR entries
         L     R2,SCHDDEF@          .Start of sched env header
         STCM  R7,3,SVSEA_NUMBER_SR .Update number of SE entries
         LTR   R7,R7                .Did we find any of the resources?
         BNZ   MOVESRSE             .Yes, proceed
         WTO   'WLMEXTRT(W): -There are no Scheduling Resources matchinX
               g the requested resources',ROUTCDE=11
         OC    RETCODE,=F'4'        .Set the return code
         B     XCHECKRX             .Go free the table and return
MOVESRSE MH    R7,=AL2(SVSEASR_LEN) .Number * length = total size
         BCTR  R7,Ø                 .Reduce length by 1 for MVCL
         LR    R5,R7                .Copy length for MVCL
         MVCL  R4,R6                .Overlay old data with new
XCHECKRX L     R3,NEWSRTBL          .Length of storage to free
         L     R2,NEWSRTB@          .Address of storage to free
         STORAGE RELEASE,LENGTH=(3),ADDR=(2)
         L     R15,RETCODE          .Load the return code
         PR                         .Restore caller's registers
*********************************************************************
*        This routine verifies that all scheduling environments
*        requested per parm were in fact found in the table returned
*        by WLM.
*********************************************************************
VERIFYSE BAKR  R14,Ø                .Preserve caller's registers
         LA    R8,PRMSETBL          .Point to first name passed in parm
         L     R2,PRM#SE            .No of entries requested per parm
VERLOOP1 TM    TBENTFND,MATCHED     .Was this entry found?
         BO    NEXTSE               .Yes, go check the next entry
         MVC   NTFNDSE+57(16),TBSENAMe
NTFNDSE  WTO   'WLMEXTRT(W): -S.E. name requested but not found:       X
                                    ',ROUTCDE=11
```

```
         OC    RETCODE,=F'4'        .Set return code to 4
NEXTSE   LA    R8,L'SVSEA_SE_SCHENV_NAME+1(R8)
         BCT   R2,VERLOOP1          .Do for each entry
VERIFYSX PR                         .Restore caller's registers
*********************************************************************
*        This routine verifies that all resource names requested per
*        parm were in fact found in the table returned by WLM.
*********************************************************************
VERIFYRE BAKR  R14,Ø                .Preserve caller's registers
         LA    R8,PRMRETBL          .Point to first name passed in parm
         L     R2,PRM#RE            .No of entries requested per parm
VERLOOP2 TM    TBENTFND,MATCHED     .Was this entry found?
         BO    NEXTRE               .Yes, go check the next entry
         MVC   NTFNDRE+56(16),TBRENAMe
NTFNDRE  WTO   'WLMEXTRT(W): -RESOURCE requested but not found:        X
                                      ',ROUTCDE=11
         OC    RETCODE,=F'4'        .Set return code to 4
NEXTRE   LA    R8,L'SVSEA_RE_RESOURCE_NAME+1(R8)
         BCT   R2,VERLOOP2          .Do for each entry
VERIFYRX PR                         .Restore caller's registers
*********************************************************************
*        This routine ACTIVATEs the policy
*********************************************************************
WLMACTIV BAKR  R14,Ø                .Preserve caller's registers
         LA    R2,POLNAME           .Point to the policy name
         IWMPACT POLICY_NAME=(R2),                                     X
               RSNCODE=RSNCODE      .Don't overlay RETCODE here!
         LTR   R15,R15              .Success?
         BZ    ACTIVWTO
         ST    R15,RETCODE          .Now overlay old value
         BAS   R14,CODEPRNT         .Go make RC and REASON codes prt.
         MVC   ACTIVMSG+25(4),PRTRC .Move return code into WTO
         MVC   ACTIVMSG+38(4),PRTRSN .Move reason code into WTO
ACTIVMSG WTO   'WLMEXTRT(E): -RC=xxxx, REASON=xxxx from IWMPACT (ACTIVAX
               TE)',ROUTCDE=11
         CLC   PRTRSN,=C'Ø416'      .Most common error
         BNE   SETRCØ12             .No, other
         MVC   ACTIVFL+29(8),POLNAME
ACTIVFL  WTO   'WLMEXTRT(E): -Policy xxxxxxxx not found',              X
               ROUTCDE=13
SETRCØ12 LA    R15,12               .Activate failed
         ST    R15,RETCODE          .Plug the return code
         B     WLMACTIX             .Get out
ACTIVWTO MVC   ACTIVWTM+22(8),POLNAME
ACTIVWTM WTO   'WLMEXTRT(I): -xxxxxxxx has been activated',            X
               ROUTCDE=13
WLMACTIX PR                         .Restore caller's registers
*********************************************************************
*        This routine reads data from the dataset into the buffer
*********************************************************************
FDATASET BAKR  R14,Ø                .Preserve caller's registers
         OPEN  (INFILE,INPUT)
```

```
         LA    R2,WLMTABLE          .Where we want the data placed
GETLOOP  GET   INFILE,(R2)
         LA    R2,8Ø(R2)            .Bump up the "to" pointer
         B     GETLOOP              .Do for each of the records
ENDDATA  CLOSE INFILE               .Close the input file
FDATASEX PR                         .Restore caller's registers
*************************************************************************
*        This routine makes the RC, RSNCode and REASON printable
*************************************************************************
CODEPRNT BAKR  R14,Ø
         L     R1,RETCODE
         CVD   R1,DOUBLE
         UNPK  DOUBLE(4),DOUBLE+5(3)
         OI    DOUBLE+3,X'FØ'
         MVC   PRTRC,DOUBLE
         L     R2,REASON
         STCM  R2,3,DOUBLE
         STCM  R2,3,DOUBLE+2
         NC    DOUBLE(2),=X'FØFØ'   .Turn off right half of bytes
         NC    DOUBLE+2(2),=X'ØFØF' .Turn off left half of bytes
         TR    DOUBLE(2),LFTHALVE   .Make left half printable
         TR    DOUBLE+2(2),RGTHALVE .Make right half printable
         MVC   DOUBLE+4(1),DOUBLE+1 .Swap bytes 2 and 3
         MVC   DOUBLE+1(1),DOUBLE+2
         MVC   DOUBLE+2(1),DOUBLE+4 .Rsn code now printable
         MVC   PRTREASN,DOUBLE
         L     R2,ERROFF            .Error offset (if applicable)
         ST    R2,DOUBLE
         ST    R2,DOUBLE+4
         NC    DOUBLE(4),=4X'FØ'    .Turn off right half of bytes
         NC    DOUBLE+4(4),=4X'ØF'  .Turn off left half of bytes
         TR    DOUBLE(4),LFTHALVE   .Make left half printable
         TR    DOUBLE+4(4),RGTHALVE .Make right half printable
         LA    R1,4                 .Number of iterations to make
         LA    R2,DOUBLE            .Start of left half
         LA    R3,DOUBLE+4          .Start of right half
         LA    R4,PRTERROF          .Where we want the result
ERROFLP  MVC   Ø(1,R4),Ø(R2)
         LA    R4,1(R4)             .Bump up "to" pointer
         LA    R2,1(R2)             .Bump up "from" pointer
         MVC   Ø(1,R4),Ø(R3)
         LA    R4,1(R4)             .Bump up "to" pointer
         LA    R3,1(R3)             .Bump up "from" pointer
         BCT   R1,ERROFLP           .Do 8 characters
         L     R2,RSNCODE
         STCM  R2,3,DOUBLE
         STCM  R2,3,DOUBLE+2
         NC    DOUBLE(2),=X'FØFØ'   .Turn off right half of bytes
         NC    DOUBLE+2(2),=X'ØFØF' .Turn off left half of bytes
         TR    DOUBLE(2),LFTHALVE   .Make left half printable
         TR    DOUBLE+2(2),RGTHALVE .Make right half printable
         MVC   DOUBLE+4(1),DOUBLE+1 .Swap bytes 2 and 3
```

```
         MVC   DOUBLE+1(1),DOUBLE+2
         MVC   DOUBLE+2(1),DOUBLE+4 .Rsn code now printable
         MVC   PRTRSN,DOUBLE
         CLC   RSNCODE+2(2),=X'040A' Output area too small?
         BNE   CODEPRNX             .No, other error
         WTO   'WLMEXTRT(E): -Work buffer too small, contact System SofX
               tware',ROUTCDE=11
CODEPRNX PR
*********************************************************************
*        This routine calculates how long the task should be delayed
*        before the ACTIVATE command will be issued.
*********************************************************************
CALCDLAY BAKR  R14,0                .Preserve caller's registers
         CLI   VALUE+2,C':'         .Must have an ":" in the time
         BNE   INVLDLAY             .Error
         CLC   VALUE(2),=C'00'      .Less than zero?
         BL    INVLDLAY             .Yes
         CLC   VALUE(2),=C'23'      .> 23?
         BH    INVLDLAY             .Yes
         CLC   VALUE+3(2),=C'00'    .Less than zero?
         BL    INVLDLAY             .Yes
         CLC   VALUE+3(2),=C'59'    .> 59?
         BH    INVLDLAY             .Yes
         MVC   WAITPARM(2),VALUE    .Hours
         MVC   WAITPARM+2(2),VALUE+3 Minutes
         MVC   WAITPARM+4(4),=4C'0' .Seconds
         XR    R15,R15              .Clear return code
         B     CALCDLAX             .Get out
INVLDLAY LA    R15,12               .Return code
         B     CALCDLAX             .Get out
CALCDLAX PR                         .Restore caller's registers
*********************************************************************
*        This routine fills parameter work tables with blanks
*********************************************************************
BLANKTBS BAKR  R14,0                .Preserve caller's registers
         LA    R2,PRMSETBL          .Point to parameter SE table
         L     R3,=AL4(TABLENG1)    .Pick up the length of the table
         XR    R4,R4                .Dummy from address
         ICM   R5,15,=X'40000000'   .Length zero with blank prop. char
         MVCL  R2,R4                .Fill the table with blanks
         LA    R2,PRMRETBL          .Point to parameter SE table
         L     R3,=AL4(TABLENG2)    .Pick up the length of the table
         XR    R4,R4                .Dummy from address
         ICM   R5,15,=X'40000000'   .Length zero with blank prop. char
         MVCL  R2,R4                .Fill the table with blanks
         PR                         .Restore caller's registers
*********************************************************************
*        Constants follow
*********************************************************************
INFILE   DCB   DDNAME=WLMFILE,EODAD=EndData,DSORG=PS,LRECL=80,RECFM=FB,X
               MACRF=GM
OUTFILE  DCB   DDNAME=WLMFILE,DSORG=PS,LRECL=80,RECFM=FB,MACRF=PM
```

```
SYSIN    DCB   DDNAME=SYSIN,DSORG=PS,LRECL=8Ø,RECFM=FB,MACRF=GL,       X
               EODAD=EOFSYSIN
MSGLOG   DCB   EXLST=MSGLOG,DSORG=PS,DDNAME=MSGLOG,MACRF=PM,LRECL=8Ø,  X
               RECFM=FB
         DS    ØF
TABSIZE  DC    AL4(TABLENG)         .Table size set at Ø.5 meg
TABLENG  EQU   5ØØØØØ               .Size of work buffer
RGTHALVE DC    X'FØF1F2F3F4F5F6F7F8F9C1C2C3C4C5C6'
WORKBUFF DC    (8Ø+L'SVSEA_SE_SCHENV_NAME)X'ØØ'
LFTHALVE DS    ØCL24Ø
         DC    X'FØ',15X'ØØ',X'F1',15X'ØØ',X'F2',15X'ØØ',X'F3'
         DC    15X'ØØ',X'F4',15X'ØØ',X'F5',15X'ØØ',X'F6',15X'ØØ',X'F7'
         DC    15X'ØØ',X'C3',15X'ØØ',X'C4',15X'ØØ',X'C5',15X'ØØ',X'C6'
VALCHARS DS    ØCL256
         DC    8ØX'Ø1',X'ØØ',1ØX'Ø1',X'ØØ',16X'Ø1',2X'ØØ'
         DC    13X'Ø1',2X'ØØ'
         DC    68X'Ø1',9X'ØØ',7X'Ø1',9X'ØØ',8X'Ø1'
         DC    8X'ØØ',6X'Ø1',1ØX'ØØ',6X'Ø1'
         LTORG
GETMAREA DSECT
SAVEAREA DS    18F
QUERYLEN DS    F                    .Returned length
OURPARM@ DS    F                    .Address of parameter we receive
DOUBLE   DS    D                    .General workarea
WAITPARM DS    D                    .Time to do the ACTIVATE
POLNAME  DS    CL8                  .Name of policy to activate
EOFADDR DS     F                    .Address to branch to on EOF
LASTCHAR DS    F                    .Pointer to end of SYSIN card
RC       DS    F                    .Return code
RETCODE  DS    F                    .Return code
RSNCODE  DS    F                    .Reason code
NEWSETB@ DS    F                    .Address of new sched. env. table
NEWSETBL DS    F                    .Length of new sched. env. table
NEWRETB@ DS    F                    .Address of new resource table
NEWRETBL DS    F                    .Length of new resource table
NEWSRTB@ DS    F                    .Address of new SE/SR table
NEWSRTBL DS    F                    .Length of new SE/SR table
PRTRC    DS    F                    .Return code (printable)
PRTRSN   DS    F                    .Reason code (printable)
PRTERROF DS    CL8                  .Reason code (printable)
PRTREASN DS    F                    .Validity reason on IWMDINST
ERROFF   DS    F                    .Offset of error on IWMDINST
SYSNAME  DS    CL3                  .System name
ACTIVATE DS    C                    .Flag to indicate ACTIVATE as well
DELAY    DS    C                    .Flag to indicate DELAY ACTIVATE
YES      EQU   X'Ø1'
COPYALL  DS    C                    .Flag to indicate copy-all
EOFFLAG  DS    C                    .Flag to indicate end-of-SYSIN
GOTSYSNM DS    C                    .Flag to indicate found sysname
GOTENVS  DS    C                    .Flag to indicate found envs
GOTRESCS DS    C                    .Flag to indicate found resources
NOLOGDD  DS    C                    .Flag to indicate no MSGLOG dd-card
```

```
LOGOPEN  DS    C                    .Flag to indicate MSGLOG open
ACTION   DS    C
FROMWLM  EQU   X'Ø1'                .Flag to indicate EXTRACT
TOWLM    EQU   X'Ø2'                .Flag to indicate INSTALL
REASON   DS    F                    .Val_Check rsn returned by IWMDINST
PRODID   EQU   *                    .Storage area to describe our id
         IWMSVIDS LIST=YES,DSECT=NO
PRM#SE DS      F                    .Number of sched env to be copied
PRM#RE DS      F                    .Number of resources to be copied
PRMSETBL DS    CL(MAXSCHED*(1+L'SVSEA_SE_SCHENV_NAME))
TABLENG1 EQU   *-PRMSETBL
PRMRETBL DS    CL(MAXRESRC*(1+L'SVSEA_RE_RESOURCE_NAME))
TABLENG2 EQU   *-PRMRETBL
SCHDDEF@ DS    F                    .Start of SE area in WLM table
OLDSETB@ DS    F                    .Address of first SE in WLM table
OLDSRTB@ DS    F                    .Address of first SE/SR pair
OLDRETB@ DS    F                    .Address of first RE in WLM table
MATCHED  EQU   X'Ø1'                .Flag
MAXSCHED EQU   3Ø                   .Max number of sched env we handle
MAXRESRC EQU   3Ø                   .Max number of resources we handle
* Parms to call routine that analyses our input parameters
PARMANLZ EQU   *
KEYWORD@ DS    F                    .Address of keyword
KEYWORDL DS    H                    .Length of keyword
VALUEL   DS    H                    .Maximum/ actual length of value
VALUE@   DS    F                    .Where we want the result
KEYWORD  DS    CL12                 .Keyword accepted from JCL
VALUE    DS    CL12                 .Keyword value accepted from JCL
JFCBPTR  DS    F                    .Pointer to JFCB area
JFCBAREA DS    CL176                .JFCB work area
WLMTABLE DS    ØF
GETMSIZE EQU   *-GETMAREA
TBLDSECT DSECT                      .Describes parm passed data
TBENTFND DS    C                    .Flag to indicate entry found
TBSENAME DS    CL(L'SVSEA_SE_SCHENV_NAME) Sched. env. name of entry
         ORG   TBSENAME
TBRENAME DS    CL(L'SVSEA_RE_RESOURCE_NAME) Resource name of entry
         ICHPRCVT
         IWMYCON
         IWMSERVD DSECT=YES,LIST=YES
         IWMSVSEA
         CVT   DSECT=YES
RØ       EQU    Ø
R1       EQU    1
R2       EQU    2
R3       EQU    3
R4       EQU    4
R5       EQU    5
R6       EQU    6
R7       EQU    7
R8       EQU    8
R9       EQU    9
```

```
R1Ø      EQU   1Ø
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         END
         LA    R8,PrmRETbl          .Point to first name passed in parm
         L     R2,Prm#RE            .No of entries requested per parm

VerLoop2 TM    TbEntFnd,Matched     .Was this entry found?
         BO    NextRE               .Yes, go check the next entry

         MVC   NtFndRE+56(16),TbREName
NtFndRE  WTO   'TELWLMEX(W): -RESOURCE requested but not found:        X
                                    ',ROUTCDE=11
         OC    RetCode,=F'4'        .Set return code to 4

NextRE   LA    R8,L'SVSEA_RE_RESOURCE_NAME+1(R8)
         BCT   R2,VerLoop2          .Do for each entry

VerifyRX PR                         .Restore caller's registers

*********************************************************************
*        This routine ACTIVATEs the policy
*********************************************************************
WLMActiv BAKR  R14,Ø                .Preserve caller's registers

         LA    R2,PolName           .Point to the policy name
         IWMPACT POLICY_NAME=(R2),                                    X
               RSNCODE=RsnCode      .Don't overlay RETCODE here!
         LTR   R15,R15              .Success?
         BZ    ActivWTO

         ST    R15,RetCode          .Now overlay old value
         BAS   R14,CodePrnt         .Go make RC and REASON codes prt.
         MVC   ActivMsg+25(4),PrtRC .Move return code into WTO
         MVC   ActivMsg+38(4),PrtRSN .Move reason code into WTO

ActivMsg WTO   'TELWLMEX(E): -RC=xxxx, REASON=xxxx from IWMPACT (ACTIVAX
               TE)',ROUTCDE=11
         CLC   PrtRsn,=C'Ø416'      .Most common error
         BNE   SetRCØ12             .No, other

         MVC   ActivFl+29(8),PolName
ActivFl  WTO   'TELWLMEX(E): -Policy xxxxxxxx not found',             X
               ROUTCDE=13

SetRCØ12 LA    R15,12               .Activate failed
         ST    R15,RetCode          .Plug the return code
         B     WLMActiX             .Get out
```

```
ActivWTO MVC   ActivWTM+22(8),PolName
ActivWTM WTO   'TELWLMEX(I): -xxxxxxxx has been activated',           X
               ROUTCDE=13
WLMActiX PR                          .Restore caller's registers

*********************************************************************
*        This routine reads data from the dataset into the buffer
*********************************************************************
FDataset BAKR  R14,Ø                 .Preserve caller's registers
         OPEN  (InFile,INPUT)
         LA    R2,WLMTable           .Where we want the data placed

GetLoop  Get   InFile,(R2)
         LA    R2,8Ø(R2)             .Bump up the "to" pointer
         B     GetLoop               .Do for each of the records

EndData  CLOSE InFile                .Close the input file

FDataseX PR                          .Restore caller's registers

*********************************************************************
*        This routine makes the RC, RSNCode, and REASON printable
*********************************************************************
CodePrnt BAKR  R14,Ø
         L     R1,RetCode
         CVD   R1,Double
         UNPK  Double(4),Double+5(3)
         OI    Double+3,X'FØ'
         MVC   PrtRC,Double

         L     R2,Reason
         STCM  R2,3,Double
         STCM  R2,3,Double+2
         NC    Double(2),=X'FØFØ'    .Turn off right half of bytes
         NC    Double+2(2),=X'ØFØF'  .Turn off left half of bytes
         TR    Double(2),LftHalve    .Make left half printable
         TR    Double+2(2),RgtHalve  .Make right half printable
         MVC   Double+4(1),Double+1  .Swap bytes 2 and 3
         MVC   Double+1(1),Double+2
         MVC   Double+2(1),Double+4  .Rsn code now printable
         MVC   PrtReasn,Double

         L     R2,ErrOff             .Error offset (if applicable)
         ST    R2,Double
         ST    R2,Double+4
         NC    Double(4),=4X'FØ'     .Turn off right half of bytes
         NC    Double+4(4),=4X'ØF'   .Turn off left half of bytes
         TR    Double(4),LftHalve    .Make left half printable
         TR    Double+4(4),RgtHalve  .Make right half printable
         LA    R1,4                  .Number of iterations to make
         LA    R2,Double             .Start of left half
```

```
         LA    R3,Double+4          .Start of right half
         LA    R4,PrtErrOf          .Where we want the result
ErrOfLp  MVC   Ø(1,R4),Ø(R2)
         LA    R4,1(R4)             .Bump up "to" pointer
         LA    R2,1(R2)             .Bump up "from" pointer
         MVC   Ø(1,R4),Ø(R3)
         LA    R4,1(R4)             .Bump up "to" pointer
         LA    R3,1(R3)             .Bump up "from" pointer
         BCT   R1,ErrOfLp           .Do 8 characters

         L     R2,RsnCode
         STCM  R2,3,Double
         STCM  R2,3,Double+2
         NC    Double(2),=X'FØFØ'   .Turn off right half of bytes
         NC    Double+2(2),=X'ØFØF' .Turn off left half of bytes
         TR    Double(2),LftHalve   .Make left half printable
         TR    Double+2(2),RgtHalve .Make right half printable
         MVC   Double+4(1),Double+1 .Swap bytes 2 and 3
         MVC   Double+1(1),Double+2
         MVC   Double+2(1),Double+4 .Rsn code now printable
         MVC   PrtRsn,Double

         CLC   RsnCode+2(2),=X'Ø4ØA' Output area too small?
         BNE   CodePrnX             .No, other error

         WTO   'TELWLMEX(E): -Work buffer too small, contact System SofX
               tware',ROUTCDE=11

CodePrnX PR

*********************************************************************
*        This routine calculates how long the task should be delayed
*        before the ACTIVATE command will be issued.
*********************************************************************
CalcDlay BAKR  R14,Ø                .Preserve caller's registers

         CLI   Value+2,C':'          .Must have an ":" in the time
         BNE   InvlDlay              .Error

         CLC   Value(2),=C'ØØ'       .Less than zero?
         BL    InvlDlay              .Yes

         CLC   Value(2),=C'23'       .> 23?
         BH    InvlDlay              .Yes

         CLC   Value+3(2),=C'ØØ'     .Less than zero?
         BL    InvlDlay              .Yes

         CLC   Value+3(2),=C'59'     .> 59?
         BH    InvlDlay              .Yes

         MVC   WaitParm(2),Value     .Hours
```

```
         MVC   WaitParm+2(2),Value+3 Minutes
         MVC   WaitParm+4(4),=4C'Ø' .Seconds
         XR    R15,R15              .Clear return code
         B     CalcDlaX             .Get out

InvlDlay LA    R15,12               .Return code
         B     CalcDlax             .Get out

CalcDlaX PR                         .Restore caller's registers

*********************************************************************
*        This routine fills parameter work tables with blanks
*********************************************************************
BlankTbs BAKR  R14,Ø                .Preserve caller's registers
         LA    R2,PrmSETbl          .Point to parameter SE table
         L     R3,=AL4(TabLeng1)    .Pick up the length of the table
         XR    R4,R4                .Dummy from address
         ICM   R5,15,=X'40000000'   .Length zero with blank prop. char
         MVCL  R2,R4                .Fill the table with blanks

         LA    R2,PrmRETbl          .Point to parameter SE table
         L     R3,=AL4(TabLeng2)    .Pick up the length of the table
         XR    R4,R4                .Dummy from address
         ICM   R5,15,=X'40000000'   .Length zero with blank prop. char
         MVCL  R2,R4                .Fill the table with blanks

         PR                         .Restore caller's registers

*********************************************************************
*        Constants follow
*********************************************************************
InFile   DCB   DDNAME=WLMFILE,EODAD=EndData,DSORG=PS,LRECL=8Ø,RECFM=FB,X
               MACRF=GM

OutFile  DCB   DDNAME=WLMFILE,DSORG=PS,LRECL=8Ø,RECFM=FB,MACRF=PM

SYSIN    DCB   DDNAME=SYSIN,DSORG=PS,LRECL=8Ø,RECFM=FB,MACRF=GL,       X
               EODAD=EOFSYSIN

MSGLOG   DCB   EXLST=MSGLOG,DSORG=PS,DDNAME=MSGLOG,MACRF=PM,LRECL=8Ø,  X
               RECFM=FB

         DS    ØF
TabSize  DC    AL4(TabLeng)         .Table size set at Ø.5 meg
TabLeng  EQU   5ØØØØØ               .Size of work buffer

RgtHalve DC    X'FØF1F2F3F4F5F6F7F8F9C1C2C3C4C5C6'

WorkBuff DC    (8Ø+L'SVSEA_SE_SCHENV_NAME)X'ØØ'
LftHalve DS    ØCL24Ø
         DC    X'FØ',15X'ØØ',X'F1',15X'ØØ',X'F2',15X'ØØ',X'F3'
         DC    15X'ØØ',X'F4',15X'ØØ',X'F5',15X'ØØ',X'F6',15X'ØØ',X'F7'
```

```
         DC    15X'00',X'C3',15X'00',X'C4',15X'00',X'C5',15X'00',X'C6'

ValChars DS    0CL256
         DC    80X'01',X'00',10X'01',X'00',16X'01',2X'00'
         DC    13X'01',2X'00'
         DC    68X'01',9X'00',7X'01',9X'00',8X'01'
         DC    8X'00',6X'01',10X'00',6X'01'

         LTORG

GetmArea DSECT
SaveArea DS    18F
QueryLen DS    F                   .Returned length
OurParm@ DS    F                   .Address of parameter we receive
TempAddr DS    F                   .Temporary address store for debug
Double   DS    D                   .General workarea
WaitParm DS    D                   .Time to do the ACTIVATE
PolName  DS    CL8                 .Name of policy to activate
EOFAddr DS     F                   .Address to branch to on EOF
LastChar DS    F                   .Pointer to end of SYSIN card
RC       DS    F                   .Return code
RetCode  DS    F                   .Return code
RsnCode  DS    F                   .Reason code
NewSETb@ DS    F                   .Address of new sched. env. table
NewSETbL DS    F                   .Length of new sched. env. table
NewRETb@ DS    F                   .Address of new resource table
NewRETbL DS    F                   .Length of new resource table
NewSRTb@ DS    F                   .Address of new SE/SR table
NewSRTbL DS    F                   .Length of new SE/SR table
PrtRC    DS    F                   .Return code (printable)
PrtRSN   DS    F                   .Reason code (printable)
PrtErrOf DS    CL8                 .Reason code (printable)
PrtReasn DS    F                   .Validity reason on IWMDINST
ErrOff   DS    F                   .Offset of error on IWMDINST
SysName  DS    CL3                 .System name
Activate DS    C                   .Flag to indicate ACTIVATE as well
Delay    DS    C                   .Flag to indicate DELAY ACTIVATE
Yes      EQU   X'01'
CopyAll  DS    C                   .Flag to indicate copy-all
EOFFlag  DS    C                   .Flag to indicate end-of-SYSIN
GotSysNm DS    C                   .Flag to indicate found sysname
GotEnvs  DS    C                   .Flag to indicate found envs
GotRescs DS    C                   .Flag to indicate found resources
NoLogDD  DS    C                   .Flag to indicate no MSGLOG dd-card
LogOpen  DS    C                   .Flag to indicate MSGLOG open

Action   DS    C
FromWLM  EQU   X'01'               .Flag to indicate EXTRACT
ToWLM    EQU   X'02'               .Flag to indicate INSTALL

Reason   DS    F                   .Val_Check rsn returned by IWMDINST

ProdID   EQU   *                   .Storage area to describe our id
```

```
         IWMSVIDS LIST=YES,DSECT=NO
Prm#SE DS      F                    .Number of sched env to be copied
Prm#RE DS      F                    .Number of resources to be copied
PrmSETbl DS    CL(MaxSched*(1+L'SVSEA_SE_SCHENV_NAME))
TabLeng1 EQU   *-PrmSETbl
PrmRETbl DS    CL(MaxResrc*(1+L'SVSEA_RE_RESOURCE_NAME))
TabLeng2 EQU   *-PrmRETbl
SchdDef@ DS    F                    .Start of SE area in WLM table
OldSETb@ DS    F                    .Address of first SE in WLM table
OldSRTb@ DS    F                    .Address of first SE/SR pair
OldRETb@ DS    F                    .Address of first RE in WLM table
Matched  EQU   X'Ø1'                .Flag
MaxSched EQU   3Ø                   .Max number of sched env we handle
MaxResrc EQU   3Ø                   .Max number of resources we handle
* Parms to call routine that analyses our input parameters
ParmAnlz EQU   *
KeyWord@ DS    F                    .Address of keyword
KeyWordL DS    H                    .Length of keyword
ValueL   DS    H                    .Maximum/ actual length of value
Value@   DS    F                    .Where we want the result
KeyWord  DS    CL12                 .Keyword accepted from JCL
Value    DS    CL12                 .Keyword value accepted from JCL
JFCBPtr  DS    F                    .Pointer to JFCB area
JFCBAREA DS    CL176                .JFCB work area
WLMTable DS    ØF
GetMSize EQU   *-GetMarea
TblDSECT DSECT                      .Describes parm passed data
TbEntFnd DS    C                    .Flag to indicate entry found
TbSEName DS    CL(L'SVSEA_SE_SCHENV_NAME) Sched. env. name of entry
         ORG   TbSEName
TbREName DS    CL(L'SVSEA_RE_RESOURCE_NAME) Resource name of entry
         ICHPRCVT
         COPY  REGS                 .Register equates

         IWMYCON
         IWMSERVD DSECT=YES,LIST=YES
         IWMSVSEA
         CVT   DSECT=YES
         END
//
//RUN EXEC  PGM=TELWLMEX,
// PARM='FUNC=INSTALL,SYSTEM=&ZSYSNAME,ACTIVATE=Y,POL=PØØBASE'
//STEPLIB  DD  DSN=TPSCP.BKEYSER.LOAD,DISP=SHR
//WLMFILE  DD  DSN=TPSCP.BKEYSER.CNTL(WLM),DISP=SHR
//MSGLOG   DD  SYSOUT=*
//SYSIN    DD  *
SYSTEM= GØ1
SCHENV=   DEFAULT TEST GØ1 SAS
RESOURCE=SYSTEM_AVAILABLE PROD_BATCH_AVAIL SAS
```


A COBOL skeleton

THE PROBLEM

Every time you start a new COBOL program, you end up re-writing the same old code. Much of what goes into the average COBOL program is required by the language. Most COBOL programs have substantial code in common, especially if the programs are designed to accomplish similar tasks.

THE SOLUTION

One way of addressing this problem is to ‘kludge’ another program that does a similar task. Copy the old program, rip out the unnecessary code, then insert new code to run the new task. This is like creating an ‘instant skeleton’.

Another method is to have one or more skeleton programs residing in your ‘cupboard’ – in this case, your personal program files. You can then copy this skeleton into a new file, alter it to fit the requirements, then test and debug it for production. Being an itinerant consultant, I keep my ‘cupboard’ on my own computer and on a floppy disk that I can take with me when working in a client’s offices.

One of the most common repetitive tasks the average programmer encounters is to create a report. Usually, the report is required ‘yesterday’ (if the client wanted the report ‘today’, he would ask for it tomorrow). The following report outline should give you a fairly good idea of the type of code you might want to keep in your personal ‘bag of tricks’.

```
000100 ID DIVISION.
000200 PROGRAM-ID. OUTLINE.
000500*REMARKS.
000600*
000800* SKELETON COPYRIGHT 2000 BY ALLAN B. KALAR, USED BY PERMISSION.
000900*
001000********************** REVISION LOG ****************************
001100* ABK xx/xx/xxxx INITIAL VERSION.                                *
```

```
001200*****************************************************************
001300     EJECT
001400 ENVIRONMENT DIVISION.
001500 CONFIGURATION SECTION.
001600*SOURCE-COMPUTER. IBM370 WITH DEBUGGING MODE.
001700 SOURCE-COMPUTER. IBM370.
001800 OBJECT-COMPUTER. IBM370.
001900 INPUT-OUTPUT SECTION.
002000 FILE-CONTROL.
002100     SELECT PRINT-F              ASSIGN UT-S-REPORT1.
002200     SELECT FILE-IN-F            ASSIGN FILEIN
002300                                 FILE STATUS FILE-STATUS.
002400*
002500*
002600 DATA DIVISION.
002700 FILE SECTION.
002800 FD  PRINT-F
002900     BLOCK 0 RECORDS.
003000 01  PRINT-R                    PIC X(133).
003100
003200 FD  FILE-IN-F
003300     BLOCK 0.
003400 01  FILE1-R                    PIC X(80).
003500
003600     EJECT
003700 WORKING-STORAGE SECTION.
003800 01  WORK-S.
003900     05 FILLER            PIC X(19) VALUE '**WORKING-STORAGE**'.
004000 01  FILE-STATUS                PIC XX.
004100 01  EOF-FILE-IN-SW             PIC X     VALUE 'N'.
004200     88 EOF-FILE-IN                       VALUE 'E'.
004300*
004400*****************************************************************
004500*        RECORD LAYOUTS
004600*****************************************************************
004700 01  FILE-RECORD.
004800
004900*
005000*****************************************************************
005100*        DATABASE DCLGENS
005200*****************************************************************
005300     EXEC SQL.
005400       INCLUDE MACTB818
005500     END-EXEC.
005600
005700*
005800*
005900*****************************************************************
006000*        DATABASE CURSOR DECLARATIONS
006100*****************************************************************
```

```
006200     EXEC SQL
006300       DECLARE ABC-CSR CURSOR FOR
006400         SELECT
006500           FROM
006600           WHERE
006700           ORDER BY
006800     END-EXEC.
006900*
007000*
007100*****************************************************************
007200*              REPORT WORK AREAS
007300*****************************************************************
007400 01  TITLE-1.
007500     05 FILLER                    PIC X(115) VALUE
007600                        ' MISSOURI AUTOMATED CHILD SUPPORT SYSTEM'.
007700     05 FILLER                    PIC X(11)  VALUE 'PAGE'.
007800     05 T1-PAGE                   PIC ZZZ,ZZ9.
007900*
008000 01  TITLE-2.
008100     05 FILLER                    PIC X(115) VALUE
008200                         ' ELIMINATE DUPLICATE PSUEDO NUMBERS'.
008300     05 FILLER                    PIC X(10)  VALUE 'RUN DATE:'.
008400     05 T2-DATE.
008500       10 MM                      PIC XX.
008600       10 FILLER                  PIC X     VALUE '/'.
008700       10 DD                      PIC XX.
008800       10 FILLER                  PIC X     VALUE '/'.
008900       10 YY                      PIC XX.
009000*
009100 01  TITLE-3.
009200     05 FILLER                    PIC X(115) VALUE
009300                                 ' CONTROL REPORT - RECORD COUNTS'.
009400     05 FILLER                    PIC X(10)  VALUE 'RUN TIME:'.
009500     05 T3-TIME.
009600       10 HH                      PIC XX.
009700       10 FILLER                  PIC X     VALUE ':'.
009800       10 MM                      PIC XX.
009900       10 FILLER                  PIC X     VALUE ':'.
010000       10 SS                      PIC XX.
010100*
010200*
010300 01  PR-LINE                                VALUE SPACES.
010400     05 FILLER                    PIC X(25).
010500     05 PR-EXPLANATION            PIC X(20).
010600     05 PR-COUNT                  PIC ZZZ,ZZZ,ZZZ.
010700*
010800*
010900 01  PR-WORK-AREAS.
011000     05 PW-SYS-DATE.
011100       10 YY                      PIC XX.
```

```
011200       10 MM                      PIC XX.
011300       10 DD                      PIC XX.
011400     05 PW-RUN-TIME.
011500       10 HH                      PIC XX.
011600       10 MM                      PIC XX.
011700       10 SS                      PIC XX.
011800*
011900     05 LINE-LIMIT                PIC S99    VALUE +60     COMP.
012000     05 LINE-COUNT                PIC S99    VALUE +66     COMP.
012100     05 PAGE-NUMBER               PIC S9(5)  VALUE ZERO    COMP-3.
012200     05 SKIP                      PIC 99     VALUE 1.
012300
012400 01  RECORD-COUNTS.
012500     05 FILE-COUNT                PIC S9(8) VALUE ZERO COMP.
012600     EJECT
012700 PROCEDURE DIVISION.
012800 MAIN-LINE.
012900D    DISPLAY 'START OUTLINE'.
013000     ACCEPT PW-SYS-DATE          FROM DATE.
013100     ACCEPT PW-RUN-TIME          FROM TIME.
013200     MOVE CORRESPONDING PW-SYS-DATE TO T2-DATE.
013300     MOVE CORRESPONDING PW-RUN-TIME TO T3-TIME.
013400     OPEN INPUT FILE-IN-F
013500     OPEN OUTPUT PRINT-F.
013600
013700     EXEC SQL
013800       OPEN ABC-CSR
013900     END-EXEC.
014000
014100     PERFORM 9000-GET-FILE.
014200     PERFORM 9100-FETCH-ABC.
014300
014400D    DISPLAY 'START 1000'.
014500     PERFORM 1000-MAIN-LOOP UNTIL EOF-FILE-IN.
014600
014700D    DISPLAY 'CLOSE'
014800     CLOSE PRINT-F
014900           FILE-IN-F.
015000
015100     EXEC SQL
015200       CLOSE ABC-CSR
015300     END-EXEC.
015400
015500     GOBACK.
015600     EJECT
015700*****************************************************************
015800*                MAIN LOOP
015900*****************************************************************
016000 1000-MAIN-LOOP.
016100D    DISPLAY '1000 MAIN LOOP'.
```

```
016200     EJECT
016300*****************************************************************
016400*****************************************************************
016500*                     I/O SUBROUTINES                           *
016600*****************************************************************
016700*****************************************************************
016800 9000-GET-FILE.
016900D    DISPLAY '9000 GET FILE'.
017000     ADD 1                       TO FILE-COUNT.
017100     READ FILE-IN-F INTO FILE-RECORD
017200       AT END
017300         SUBTRACT 1              FROM FILE-COUNT
017400         SET EOF-FILE-IN         TO TRUE
017500         MOVE HIGH-VALUES        TO FILE-RECORD.
017600D    DISPLAY '    FILE-RECORD: ' FILE-RECORD.
017700
017800     EJECT
017900*****************************************************************
018000*            FETCH ABC CURSOR ROW
018100*****************************************************************
018200 9100-FETCH-ABC.
018300D    DISPLAY '9100 FETCH ABC'.
018400     EXEC SQL
018500       FETCH ABC-CURSOR
018600        INTO
018700     END-EXEC.
018800
018900     IF SQLCODE = +100
019000       SET ABC-END               TO TRUE
019100     ELSE
019200     IF SQLCODE NOT = ZERO
019300       *S*  ERROR ROUTINE ***
019400     END-IF
019500     END-IF.
019600
019700     EJECT
019800*****************************************************************
019900*            PRINT REPORT SUBROUTINES
020000*****************************************************************
020100 9800-PRINT.
020200D    DISPLAY '9800 PRINT'.
020300     IF LINE-COUNT > LINE-LIMIT
020400       PERFORM 9810-HEAD.
020500
020600     WRITE PRINT-R FROM PR-LINE AFTER SKIP.
020700     ADD SKIP                    TO LINE-COUNT.
020800     MOVE 1                      TO SKIP.
020900     MOVE SPACES                 TO PR-LINE.
021000*
021100 9810-HEAD.
021200D    DISPLAY '9810 HEAD'.
```

```
021300     ADD 1                        TO PAGE-NUMBER.
021400     MOVE PAGE-NUMBER             TO T1-PAGE.
021500
021600     WRITE PRINT-R  FROM TITLE-1 AFTER PAGE.
021700     WRITE PRINT-R  FROM TITLE-2 AFTER 1.
021800     WRITE PRINT-R  FROM TITLE-3 AFTER 1.
021900
022000     MOVE 4                       TO SKIP.
022100     MOVE 7                       TO LINE-COUNT.
```

NOTES

The following notes explain the significant elements of the above code.

- Line 200 – do a global change to the name of the program you are creating. In ISPF/PDF, enter the following on the command line: 'C OUTLINE yourprograme name'.
- Line 300 – put your name here and save it as part of your skeleton.
- Line 800 – leave this if you use this as-is. If you create your own, by all means insert your own copyright.
- Line 1100 – put in your own initials and the current date. If the shop you work in uses a different method of tracking revisions, by all means, follow that convention.
- Line 11900 – the maximum number of lines to a page.
- Line 12000 – the VALUE should be greater than LINE-LIMIT to force a new page and headers before the first line is written.
- Line 12200 – in the body of your program, set this variable to the number of lines you want to advance before printing. A value of zero will cause overprinting of the previous line, a value of '2' will skip one line before printing, etc. 'SKIP' will be reset to '1' after the line is written.
- Lines 1600 and 1700 – by swapping the asterisk in column seven, you can turn 'debugging' on and off easily (see *Using COBOL Debug* in *MVS Update*, Issue 156, September 1999).
- Lines 2000-2300 – make file changes here and in the 'FD' statements in the DATA DIVISION that follow.

- Line 3900 – this line makes it easier to find your WORKING-STORAGE data in a memory dump. You can carry this to ridiculous extremes if it suits you to do so.
- Lines 5000-6800 – this assumes that a DB2 or similar SQL database is being used. Delete it, if you do not use these databases.
- Lines 7600-12200 – the layout for the report. Alter it to the standard layout used in your shop in your version of the skeleton. Make custom changes for the particular report being created when using the skeleton.
- Line 12900 – this only compiles in 'debug mode' and provides a run-time trace of which paragraphs are entered. You will find these scattered throughout the program. If the program amends during a test run, you can get a pretty good idea of where the problem is by studying these clues in the CYST file.
- Lines 13700-13900 – more DB2 stuff here, in lines 15100-15300, and 18400-19500.
- Line 20100 – the main print routine. If you set-up the rest of the program up correctly, this is the only paragraph you have to perform to write a line to your report. It will decide when to eject a page and write new page heading.
- Line 22000 – decide how much space you want between the page heading and the first line, and change this line to fit.
- Line 22100 – this amount is a function of the number of lines taken up by the page heading and needs to be changed to fit.

Alan Kalar
Systems Programmer (USA)

The Initialization Parameter Area

INTRODUCTION

The Initialization Parameter Area (IPA) became available in OS/390 Version 1 Release 2. It is mapped by the IHAIPA macro in SYS1.MACLIB and contains initialization parameters defined in:

- The load parameter used to IPL
- The LOADxx member used to IPL
- All IEASYSxx members used to IPL.

The following REXX EXEC can be used to display the information in the IPA and also displays other system information. If executed from ISPF, the display will be put in a scrollable browse dataset.

```
/* REXX */
/* Trace ?r */
/*********************************************************************/
/* DISPLAY SYSTEM INFORMATION ON TERMINAL                            */
/*********************************************************************/
Numeric digits 10
Call RDATE TODAY                              /* call RDATE subroutine*/
DAY      = Word(RESULT,3)                     /* weekday from RDATE   */
DATE     = Substr(RESULT,1,10)                /* date as MM/DD/YYYY   */
JUL      = Substr(RESULT,7,8)                 /* date as YYYY.DDD     */
/*                                                                   */
CVT      = C2d(Storage(10,4))                 /* point to CVT         */
/*                                                                   */
JESCT    = C2d(Storage(D2x(CVT + 296),4))     /* point to JESCT       */
/*                                                                   */
STORSIZE = C2d(Storage(D2x(CVT + 856),4))     /* point to storage size*/
STORSIZE = STORSIZE/1024                      /* convert to Megabytes */
/*                                                                   */
RCE      = C2d(Storage(D2x(CVT + 1168),4))    /* point to RCE         */
ESTOR    = C2d(Storage(D2x(RCE + 160),4))     /* point to ESTOR frames*/
ESTOR    = ESTOR*4/1024                       /* convert to Megabytes */
/*                                                                   */
CVTGDA   = C2d(Storage(D2x(CVT + 560),4))     /* point to GDA         */
GDAPVTSZ = C2d(Storage(D2x(CVTGDA + 164),4)) /* point to MAX PVT<16M */
GDAPVTSZ = GDAPVTSZ/1024                      /* convert to Kbytes    */
GDAEPVTS = C2d(Storage(D2x(CVTGDA + 172),4)) /* point to MAX PVT>16M */
GDAEPVTS = GDAEPVTS/1024/1024                 /* convert to Mbytes    */
```

```
GDACSASZ = C2d(Storage(D2x(CVTGDA + 112),4)) /* point to CSA<16M     */
GDACSASZ = GDACSASZ/1Ø24                     /* convert to Kbytes    */
GDAECSAS = C2d(Storage(D2x(CVTGDA + 128),4)) /* point to CSA>16M     */
GDAECSAS = GDAECSAS/1Ø24                     /* convert to Kbytes    */
GDASQASZ = C2d(Storage(D2x(CVTGDA + 148),4)) /* point to SQA<16M     */
GDASQASZ = GDASQASZ/1Ø24                     /* convert to Kbytes    */
GDAESQAS = C2d(Storage(D2x(CVTGDA + 156),4)) /* point to SQA>16M     */
GDAESQAS = GDAESQAS/1Ø24                     /* convert to Kbytes    */
GDAVRSZ  = C2d(Storage(D2x(CVTGDA + 196),4)) /* point to V=R global  */
GDAVRSZ  = GDAVRSZ/1Ø24                      /* convert to Kbytes    */
GDAVREGS = C2d(Storage(D2x(CVTGDA + 2ØØ),4)) /* point to V=R default */
GDAVREGS = GDAVREGS/1Ø24                     /* convert to Kbytes    */
/*                                                                   */
CVTEXT2  = C2d(Storage(D2x(CVT + 328),4))    /* point to CVTEXT2     */
CVTATCVT = C2d(Storage(D2x(CVTEXT2 + 65),3)) /* point to VTAM AVT    */
ISTATCVT = C2d(Storage(D2x(CVTATCVT + Ø),4)) /* point to VTAM CVT    */
ATCVTLVL = Storage(D2x(ISTATCVT + Ø),8)      /* VTAM Rel Lvl VOVRP   */
VTAMVER  = Substr(ATCVTLVL,3,1)              /* VTAM Version   V     */
VTAMREL  = Substr(ATCVTLVL,4,1)              /* VTAM Release    R    */
VTAMMOD  = Substr(ATCVTLVL,5,1)              /* VTAM Mod Lvl     P   */
If VTAMMOD = ' ' then VTAMLEV = 'V'|| VTAMVER || 'R' || VTAMREL
   else VTAMLEV = 'V' || VTAMVER || 'R' || VTAMREL || 'M' || VTAMMOD
/*                                                                   */
AMCBS    = C2d(Storage(D2x(CVT + 256),4))    /* point to AMCBS       */
ACB      = C2d(Storage(D2x(AMCBS + 8),4))    /* point to ACB         */
CAXWA    = C2d(Storage(D2x(ACB + 64),4))     /* point to CAXWA       */
MCATDSN  = Storage(D2x(CAXWA + 52),44)       /* master catalog dsn   */
MCATDSN  = Strip(MCATDSN,T)                  /* remove trailing blnks*/
MCATUCB  = C2d(Storage(D2x(CAXWA + 28),4))   /* point to mcat UCB    */
MCATVOL  = Storage(D2x(MCATUCB + 28),6)      /* master catalog VOLSER*/
/*                                                                   */
SMCA     = Storage(D2x(CVT + 196),4)         /* point to SMCA        */
SMCA     = Bitand(SMCA,'7FFFFFFF'x)          /* zero high order bit  */
SMCA     = C2d(SMCA)                         /* convert to decimal   */
/*                                                                   */
/*********************************************************************/
/*  The IPL date is stored in packed decimal format - so to make     */
/*  the date printable, it needs to be converted back to hex and     */
/*  the packed sign needs to be removed.                             */
/*********************************************************************/
IPLTIME  = C2d(Storage(D2x(SMCA + 336),4))   /* IPL Time - binary    */
IPLDATE  = C2d(Storage(D2x(SMCA + 34Ø),4))   /* IPL Date - ØCYYDDDF  */
If IPLDATE  >= 16777231 then do              /*          is C = 1 ?  */
  IPLDATE  = D2x(IPLDATE)                    /* convert back to hex  */
  IPLDATE  = Substr(IPLDATE,2,5)             /* keep YYDDD           */
  IPLDATE  = '2Ø'IPLDATE                     /* use 21st century date*/
End
Else do
  IPLDATE  = D2x(IPLDATE)                    /* convert back to hex  */
  IPLDATE  = Left(IPLDATE,5)                 /* keep YYDDD           */
```

```
  IPLDATE  = '19'IPLDATE                      /* use 20th century date*/
End
IPLYYYY  = Substr(IPLDATE,1,4)                /* YYYY portion of date */
IPLDDD   = Substr(IPLDATE,5,3)                /* DDD  portion of date */
Call RDATE IPLYYYY IPLDDD                     /* call RDATE subroutine*/
IPLDAY   = Word(RESULT,3)                     /* weekday from RDATE   */
IPLDATE  = Substr(RESULT,1,10)                /* date as MM/DD/YYYY   */
IPLJUL   = Substr(RESULT,7,8)                 /* date as YYYY.DDD     */
IPLTIME  = IPLTIME / 100                      /* remove hundreths     */
HH       = IPLTIME % 3600                     /* IPL hour             */
MM       = (IPLTIME - (3600 * HH)) % 60       /* IPL minute           */
SS       = (IPLTIME - (3600 * HH)  - (60 * MM)) % 1  /* IPL seconds  */
HH       = Right(HH,2,'0')                    /* ensure 2 digit HH    */
MM       = Right(MM,2,'0')                    /* ensure 2 digit MM    */
SS       = Right(SS,2,'0')                    /* ensure 2 digit SS    */
IPLTIME  = HH':'MM':'SS                       /* time in HH:MM format */
/*                                                                     */
ASMVT    = C2d(Storage(D2x(CVT + 704),4))     /* point to ASMVT       */
CLPABYTE = Storage(D2x(ASMVT + 1),1)          /* point to CLPA byte   */
CHKCLPA  = Bitand(CLPABYTE,'8'x)              /* check for B'1000'    */
CHKCLPA  = C2d(CHKCLPA)                       /* convert to decimal   */
If CHKCLPA < 8 then IPLCLPA = '(with CLPA)'   /* bit off - CLPA       */
  Else IPLCLPA = '(without CLPA)'             /* bit on  - no CLPA    */
/*                                                                     */
SMFNAME  = Storage(D2x(SMCA + 16),4)          /* point to SMF name    */
/*                                                                     */
PRODNAME = Storage(D2x(CVT - 40),7)           /* point to mvs version */
FMIDNUM  = Storage(D2x(CVT - 32),7)           /* point to fmid        */
/*                                                                     */
GRSNAME  = Storage(D2x(CVT + 340),8)          /* point to system name */
GRSNAME  = Strip(GRSNAME,T)                   /* del trailing blanks  */
/*                                                                     */
RESUCB   = C2d(Storage(D2x(JESCT + 4),4))     /* point to SYSRES UCB  */
JESNAME  = Storage(D2x(JESCT + 28),4)         /* point to JESNAME     */
IPLVOL   = Storage(D2x(RESUCB + 28),6)        /* point to IPL volume  */
If Substr(PRODNAME,3,1) < 5 then ,
  IPLADDR  = Storage(D2x(RESUCB + 13),3)      /* point to IPL address */
Else do
  CVTSYSAD = C2d(Storage(D2x(CVT + 48),4))    /* point to UCB address */
  IPLADDR  = Storage(D2x(CVTSYSAD + 4),2)     /* point to IPL UCB     */
  IPLADDR  = C2x(IPLADDR)                     /* convert to EBCDIC    */
End
/*********************************************************************/
/*  The CPU model is stored in packed decimal format with no sign,   */
/*  so to make the model printable, it needs to be converted back    */
/*  to hex.                                                          */
/*********************************************************************/
MODEL    = C2d(Storage(D2x(CVT - 6),2))       /* point to cpu model   */
MODEL    = D2x(MODEL)                         /* convert back to hex  */
/*                                                                     */
```

```
CSD      = C2d(Storage(D2x(CVT + 660),4))    /* point to CSD         */
NUMCPU   = C2d(Storage(D2x(CSD + 10),2))     /* point to # of CPUS   */
/*                                                                   */
/*********************************************************************/
/*   Write  information to terminal.                                 */
/*********************************************************************/
Queue '*****************************************************' || ,
      '**********************'
Queue '************************  SYSTEM INFORMATION  **********' || ,
      '**********************'
Queue '*****************************************************' || ,
      '**********************'
Queue ' '
Queue 'Today is 'DAY DATE '('JUL').'
Queue 'The last IPL was 'IPLDAY IPLDATE '('IPLJUL')' ,
     'at 'IPLTIME IPLCLPA'.'
Queue 'The system IPL address was 'IPLADDR' ('IPLVOL').'
If Substr(PRODNAME,3,1) > 3 then do
  ECVT     = C2d(Storage(D2x(CVT + 140),4))  /* point to CVTECVT     */
  PLEXNM   = Storage(D2x(ECVT+8),8)          /* point to SYSPLEX name*/
  IPLPARM  = Storage(D2x(ECVT+160),8)        /* point to LOAD PARM   */
  IPLPARM  = Strip(IPLPARM,T)                /* del trailing blanks  */
  SEPPARM  = Substr(IPLPARM,1,4) Substr(IPLPARM,5,2),
             Substr(IPLPARM,7,1) Substr(IPLPARM,8,1)
  Queue 'The IPL LOAD PARM used was 'IPLPARM' ('SEPPARM').'
  Queue 'The SYSPLEX name is' PLEXNM
End
If Substr(PRODNAME,3,1) < 5 then do
  IOCON    = Storage(D2x(CVTEXT2 + 6),2)       /* HCD IODFxx or MVSCP*/
                                               /* IOCONFIG ID=xx     */
  Queue 'The currently active IOCONFIG or HCD IODF is 'IOCON'.'
End
Else do
  If Substr(FMIDNUM,4,4) >= 6602 then VOFF = 0
     else VOFF = 32
  CVTIXAVL = C2d(Storage(D2x(CVT+124),4))       /* point to IOCM     */
  IOCIOVTP = C2d(Storage(D2x(CVTIXAVL+208),4))  /* pt to IOS Vect Tbl*/
  IODF     = Storage(D2X(IOCIOVTP+288-VOFF),11) /* point to IODF name*/
  CONFIGID = Storage(D2X(IOCIOVTP+348-VOFF),8)  /* point to CONFIG   */
  EDT      = Storage(D2X(IOCIOVTP+360-VOFF),2)  /* point to EDT      */
  IOPROC   = Storage(D2X(IOCIOVTP+380-VOFF),8)  /* point to IODF Proc*/
  IODATE   = Storage(D2X(IOCIOVTP+412-VOFF),8)  /* point to IODF date*/
  IOTIME   = Storage(D2X(IOCIOVTP+420-VOFF),8)  /* point to IODF time*/
  IODESC   = Storage(D2X(IOCIOVTP+428-VOFF),16) /* point to IODF desc*/
  Queue 'The currently active IODF data set is 'IODF'.'
  Queue '  Configuration ID =' CONFIGID ' EDT ID =' EDT
  Queue '  TOKEN: Processor  Date      Time      Description'
  Queue '         'IOPROC'   'IODATE'  'IOTIME'  'IODESC
End
Queue 'The Master Catalog is 'MCATDSN' on 'MCATVOL'.'
```

```
Queue ' '
If Substr(PRODNAME,3,1) < 6 then
  Queue 'The MVS version is 'PRODNAME' - FMID 'FMIDNUM'.'
Else do
  PRODNAME = Storage(D2x(ECVT+496),16)       /* point to product name*/
  PRODNAME = Strip(PRODNAME,T)               /* del trailing blanks  */
  VER      = Storage(D2x(ECVT+512),2)        /* point to version     */
  REL      = Storage(D2x(ECVT+514),2)        /* point to release     */
  MOD      = Storage(D2x(ECVT+516),2)        /* point to mod level   */
  VRM      = VER'.'REL'.'MOD
  Queue 'The OS version is 'PRODNAME VRM' - FMID 'FMIDNUM'.'
End
Queue 'The VTAM Level is 'VTAMLEV'.'
Queue 'The primary job entry subsystem is 'JESNAME'.'
Queue 'The GRS system id is 'GRSNAME'. The SMF system id is 'SMFNAME'.'
Queue ' '
Queue 'The real storage size is 'Format(STORSIZE,,Ø)'M.'
If ESTOR > Ø then
  Queue 'The expanded storage size is 'ESTOR'M.'
Else
  Queue 'The system has no expanded storage.'
Queue 'The private area size <16M is 'GDAPVTSZ'K.'
Queue 'The private area size >16M is 'GDAEPVTS'M.'
Queue 'The CSA size <16M is 'GDACSASZ'K.'
Queue 'The CSA size >16M is 'GDAECSAS'K.'
Queue 'The SQA size <16M is 'GDASQASZ'K.'
Queue 'The SQA size >16M is 'GDAESQAS'K.'
Queue 'The maximum V=R region size is 'GDAVRSZ'K.'
Queue 'The default V=R region size is 'GDAVREGS'K.'
Queue 'The maximum V=V region size is 'GDAPVTSZ-2Ø'K.'
Queue ' '
Queue 'The CPU model number is 'MODEL'.'
Queue 'The number of online CPUs is 'NUMCPU'.'
/*                                                                  */
PCCAVT    = C2d(Storage(D2x(CVT + 764),4))   /* point to PCCA vect tb*/
/*                                                                  */
CPNUM     = Ø
FOUNDCPUS = Ø
Do until FOUNDCPUS = NUMCPU
PCCA = C2d(Storage(D2x(PCCAVT + CPNUM*4),4)) /* point to PCCA       */
  If PCCA <> Ø then do
    CPUID   = Storage(D2x(PCCA + 6),1Ø)      /* point to CPUID      */
    IDSHORT = Substr(CPUID,2,5)
    Queue 'The CPU serial number for CPU 'CPNUM' is ' || ,
     CPUID' ('IDSHORT').'
    FOUNDCPUS = FOUNDCPUS + 1
  End
CPNUM = CPNUM + 1
End  /* do until  */
/*************************************************/
```

```
/* Central Processing Complex Node Descriptor     */
/************************************************/
CVTHID   = C2d(Storage(D2x(CVT + 1068),4))   /* point to SHID        */
CPCND_FLAGS = Storage(D2x(CVTHID+22),1)      /* point to CPCND FLAGS */
If CPCND_FLAGS <> 0 then do                  /* Is there a CPC?      */
  CPCND_VALID = Bitand(CPCND_FLAGS,'E0'x)    /* Valid flags          */
  CPCND_INVALID = Bitand('40'x)              /* Invalid flag         */
  If CPCND_VALID <> CPCND_INVALID then do    /* Is it valid?         */
    CPCND_TYPE  = Storage(D2x(CVTHID+26),6)  /* Type                 */
    CPCND_MODEL = Storage(D2x(CVTHID+32),3)  /* Model                */
    CPCND_MAN   = Storage(D2x(CVTHID+35),3)  /* Manufacturer         */
    CPCND_PLANT = Storage(D2x(CVTHID+38),2)  /* Plant of manufact.   */
    CPCND_SEQNO = Storage(D2x(CVTHID+40),12) /* Sequence number      */
    CPC_ID      = C2x(Storage(D2x(CVTHID+55),1))  /* CPC ID          */
  End /* if CPCND_VALID <> CPCND_INVALID */
End  /* if CPCND_FLAGS <>0  */
Queue ' '
Queue 'Central Processing Complex (CPC) Node Descriptor:'
Queue '  CPC ND =',
  CPCND_TYPE'.'CPCND_MODEL'.'CPCND_MAN'.'CPCND_PLANT'.'CPCND_SEQNO
Queue '  CPC ID =' CPC_ID
Queue '  Type('CPCND_TYPE') Model('CPCND_MODEL')',
      'Manufacturer('CPCND_MAN') Plant('CPCND_PLANT')',
      'Seq Num('CPCND_SEQNO')'
Queue ' '
If Substr(FMIDNUM,4,4) >= 6602 then do
  /*************************/
  /* IPL parms from the IPA */
  /*************************/
  ECVTIPA  = C2d(Storage(D2x(ECVT + 392),4)) /* point to IPA         */
  IPALPARM = Storage(D2x(ECVTIPA + 16),8)    /* point to LOAD PARM   */
  IPALPDSN = Storage(D2x(ECVTIPA + 48),44)   /* load parm dsn name   */
  IPAHWNAM = Storage(D2x(ECVTIPA + 24),8)    /* point to HWNAME      */
  IPAHWNAM = Strip(IPAHWNAM,T)               /* del trailing blanks  */
  IPALPNAM = Storage(D2x(ECVTIPA + 32),8)    /* point to LPARNAME    */
  IPALPNAM = Strip(IPALPNAM,T)               /* del trailing blanks  */
  IPAVMNAM = Storage(D2x(ECVTIPA + 40),8)    /* point to VMUSERID    */
  /*************************/
  /* PARMS in LOADxx       */
  /*************************/
  IPANUCID = Storage(D2x(ECVTIPA + 23),1)    /* NUCLEUS ID           */
  IPAIODF  = Storage(D2x(ECVTIPA + 96),63)   /* IODF    card image   */
  IPASPARM = Storage(D2x(ECVTIPA + 160),63)  /* SYSPARM card image   */
  IPASCAT  = Storage(D2x(ECVTIPA + 224),63)  /* SYSCAT  card image   */
  IPASYM   = Storage(D2x(ECVTIPA + 288),63)  /* IEASYM  card image   */
  IPAPLEX  = Storage(D2x(ECVTIPA + 352),63)  /* SYSPLEX card image   */
  IPAPLNUM = Storage(D2x(ECVTIPA + 2148),2)  /* number of parmlibs   */
  IPAPLNUM = C2x(IPAPLNUM)                   /* convert to EBCDIC    */
  POFF = 0
  Do P = 1 to IPAPLNUM
```

```
    IPAPLIB.P = Storage(D2x(ECVTIPA+416+POFF),63) /* PARMLIB cards   */
    POFF = POFF + 64
  End
  IPANLID  = Storage(D2x(ECVTIPA + 2144),2)  /* NUCLSTxx member used */
  IPANUCW  = Storage(D2x(ECVTIPA + 2146),1)  /* load wait state char */
  Queue 'Initialization information from the IPA:'
  Queue '  IPLPARM =' IPALPARM
  Queue '  IPL load parameter data set name: 'IPALPDSN
  Queue '  HWNAME='IPAHWNAM '  LPARNAME='IPALPNAM ,
        '  VMUSERID='IPAVMNAM
  Queue '  LOADxx parameters (LOAD' || Substr(IPALPARM,5,2) || '):'
  If IPASYM   <> '' then queue '    IEASYM   'IPASYM
  If IPAIODF  <> '' then queue '    IODF     'IPAIODF
  If IPANUCID <> '' then queue '    NUCLEUS  'IPANUCID
  If IPANLID  <> '' then queue '    NUCLST   'IPANLID' 'IPANUCW
  Do P = 1 to IPAPLNUM
    Queue '    PARMLIB  'IPAPLIB.P
  End
  If IPASCAT  <> '' then queue '    SYSCAT   'IPASCAT
  If IPASPARM <> '' then queue '    SYSPARM  'IPASPARM
  If IPAPLEX  <> '' then queue '    SYSPLEX  'IPAPLEX
  /*************************/
  /* PARMS in IEASYSxx     */
  /*************************/
  Queue '  IEASYSxx parameters:'
  Call BUILD_IPAPDETB       /* Build table for init parms       */
  Do I = 1 to IPAPDETB.0
    Call EXTRACT_SYSPARMS IPAPDETB.I
  End
End
  /*************************/
  /* Virtual Storage Map   */
  /*************************/
If GDAVRSZ = 0 then do                      /* no v=r              */
  VRSTRT = 'N/A     '
  VREND  = 'N/A     '
  VVSTRT = '00005000'                       /* start of v=v        */
  VVEND  =  20480 + ((GDAPVTSZ-20)*1024) - 1 /* end of v=v         */
  VVEND  =  D2x(VVEND)                      /* display in hex      */
End
Else do
  VRSTRT = '00005000'                       /* start of v=r        */
  VREND  =  20480 + (GDAVRSZ*1024) - 1      /* end of v=r          */
  VREND  =  D2X(VREND)  /* display in hex      */
  VVSTRT = '00005000'                       /* start of v=v        */
  VVEND  =  20480 + ((GDAPVTSZ-20)*1024) - 1 /* end of v=v         */
  VVEND  =  D2x(VVEND)                      /* display in hex      */
End
GDACSA   = C2d(Storage(D2x(CVTGDA + 108),4)) /* start of CSA addr  */
GDACSAH  = D2x(GDACSA)                      /* display in hex      */
```

```
CSAEND   = (GDACSASZ*1Ø24) + GDACSA - 1      /* end of CSA           */
CSAEND   = D2x(CSAEND)                       /* display in hex       */
CVTSMEXT = C2d(Storage(D2x(CVT +1196),4))    /* point to stg map ext.*/
CVTMLPAS = C2d(Storage(D2x(CVTSMEXT+ 8),4))  /* start of MLPA addr   */
CVTMLPAS = D2x(CVTMLPAS)                     /* display in hex       */
If CVTMLPAS <> Ø then do
  CVTMLPAE = C2d(Storage(D2x(CVTSMEXT+12),4))  /* end of MLPA addr   */
  CVTMLPAE = D2x(CVTMLPAE)                     /* display in hex     */
  MLPASZ   = X2d(CVTMLPAE) - X2d(CVTMLPAS) + 1 /* size of MLPA       */
  MLPASZ   = MLPASZ/1Ø24                       /* convert to Kbytes  */
End
Else do /* no MLPA */
  CVTMLPAS = 'N/A     '
  CVTMLPAE = 'N/A     '
  MLPASZ   = Ø
End
CVTFLPAS = C2d(Storage(D2x(CVTSMEXT+16),4))  /* start of FLPA addr   */
CVTFLPAS = D2x(CVTFLPAS)                     /* display in hex       */
If CVTFLPAS <> Ø then do
  CVTFLPAE = C2d(Storage(D2x(CVTSMEXT+2Ø),4))  /* end of FLPA addr   */
  CVTFLPAE = D2x(CVTFLPAE)                     /* display in hex     */
  FLPASZ   = X2d(CVTFLPAE) - X2d(CVTFLPAS) + 1 /* size of FLPA       */
  FLPASZ   = FLPASZ/1Ø24                       /* convert to Kbytes  */
End
Else do /* no FLPA */
  CVTFLPAS = 'N/A     '
  CVTFLPAE = 'N/A     '
  FLPASZ   = Ø
End
CVTPLPAS = C2d(Storage(D2x(CVTSMEXT+24),4))  /* start of PLPA addr   */
CVTPLPAS = D2x(CVTPLPAS)                     /* display in hex       */
CVTPLPAE = C2d(Storage(D2x(CVTSMEXT+28),4))  /* end of PLPA addr     */
CVTPLPAE = D2x(CVTPLPAE)                     /* display in hex       */
PLPASZ   = X2d(CVTPLPAE) - X2d(CVTPLPAS) + 1 /* size of PLPA         */
PLPASZ   = PLPASZ/1Ø24                       /* convert to Kbytes    */
GDASQA   = C2d(Storage(D2x(CVTGDA + 144),4)) /* start of SQA addr    */
GDASQAH  = D2x(GDASQA)                       /* display in hex       */
SQAEND   = (GDASQASZ*1Ø24) + GDASQA - 1      /* end of SQA           */
SQAEND   = D2x(SQAEND)                       /* display in hex       */
CVTRWNS  = C2d(Storage(D2x(CVTSMEXT+32),4))  /* start of R/W nucleus */
CVTRWNS  = D2x(CVTRWNS)                      /* display in hex       */
CVTRWNE  = C2d(Storage(D2x(CVTSMEXT+36),4))  /* end of R/W nucleus   */
CVTRWNE  = D2x(CVTRWNE)                      /* display in hex       */
RWNUCSZ  = X2d(CVTRWNE)  - X2d(CVTRWNS)  + 1 /* size of R/W nucleus  */
RWNUCSZ  = Format(RWNUCSZ/1Ø24,,Ø)           /* convert to Kbytes    */
CVTRONS  = C2d(Storage(D2x(CVTSMEXT+4Ø),4))  /* start of R/O nucleus */
CVTRONS  = D2x(CVTRONS)                      /* display in hex       */
CVTRONE  = C2d(Storage(D2x(CVTSMEXT+44),4))  /* end of R/O nucleus   */
CVTRONE  = D2x(CVTRONE)                      /* display in hex       */
RONUCSZ  = X2d(CVTRONE)  - X2d(CVTRONS)  + 1 /* size of R/O nucleus  */
```

```
RONUCSZ  = Format(RONUCSZ/1Ø24,,Ø)           /* convert to Kbytes    */
RONUCSZB = X2d('FFFFFF') - X2d(CVTRONS) + 1  /* size of R/O nuc <16M */
RONUCSZB = Format(RONUCSZB/1Ø24,,Ø)          /* convert to Kbytes    */
RONUCSZA = X2d(CVTRONE) - X2d('1ØØØØØØ') + 1 /* size of R/O nuc >16M */
RONUCSZA = Format(RONUCSZA/1Ø24,,Ø)          /* convert to Kbytes    */
CVTERWNS = C2d(Storage(D2x(CVTSMEXT+48),4))  /* start of E-R/W nuc   */
CVTERWNS = D2x(CVTERWNS)                     /* display in hex       */
CVTERWNE = C2d(Storage(D2x(CVTSMEXT+52),4))  /* end of E-R/W nuc     */
CVTERWNE = D2x(CVTERWNE)                     /* display in hex       */
ERWNUCSZ = X2d(CVTERWNE) - X2d(CVTERWNS) + 1 /* size of E-R/W nuc    */
ERWNUCSZ = ERWNUCSZ/1Ø24                     /* convert to Kbytes    */
GDAESQA  = C2d(Storage(D2x(CVTGDA + 152),4)) /* start of ESQA addr   */
GDAESQAH = D2x(GDAESQA)                      /* display in hex       */
ESQAEND  = (GDAESQAS*1Ø24) + GDAESQA - 1     /* end of ESQA          */
ESQAEND  = D2x(ESQAEND)                      /* display in hex       */
CVTEPLPS = C2d(Storage(D2x(CVTSMEXT+56),4))  /* start of EPLPA addr  */
CVTEPLPS = D2x(CVTEPLPS)                     /* display in hex       */
CVTEPLPE = C2d(Storage(D2x(CVTSMEXT+6Ø),4))  /* end of EPLPA addr    */
CVTEPLPE = D2x(CVTEPLPE)                     /* display in hex       */
EPLPASZ  = X2d(CVTEPLPE) - X2d(CVTEPLPS) + 1 /* size of EPLPA        */
EPLPASZ  = EPLPASZ/1Ø24                      /* convert to Kbytes    */
CVTEFLPS = C2d(Storage(D2x(CVTSMEXT+64),4))  /* start of EFLPA addr  */
CVTEFLPS = D2x(CVTEFLPS)                     /* display in hex       */
If CVTEFLPS <> Ø then do
  CVTEFLPE = C2d(Storage(D2x(CVTSMEXT+68),4))  /* end of EFLPA addr  */
  CVTEFLPE = D2x(CVTEFLPE)                     /* display in hex     */
  EFLPASZ  = X2d(CVTEFLPE) - X2d(CVTEFLPS) + 1 /* size of EFLPA      */
  EFLPASZ  = EFLPASZ/1Ø24                      /* convert to Kbytes  */
End
Else do /* no EFLPA */
  CVTEFLPS = 'N/A     '
  CVTEFLPE = 'N/A     '
  EFLPASZ  = Ø
End
CVTEMLPS = C2d(Storage(D2x(CVTSMEXT+72),4))  /* start of EMLPA addr  */
CVTEMLPS = D2x(CVTEMLPS)                     /* display in hex       */
If CVTEMLPS <> Ø then do
  CVTEMLPE = C2d(Storage(D2x(CVTSMEXT+76),4))  /* end of EMLPA addr  */
  CVTEMLPE = D2x(CVTEMLPE)                     /* display in hex     */
  EMLPASZ  = X2d(CVTEMLPE) - X2d(CVTEMLPS) + 1 /* size of EMLPA      */
  EMLPASZ  = EMLPASZ/1Ø24                      /* convert to Kbytes  */
End
Else do /* no EMLPA */
  CVTEMLPS = 'N/A     '
  CVTEMLPE = 'N/A     '
  EMLPASZ  = Ø
End
GDAECSA  = C2d(Storage(D2x(CVTGDA + 124),4)) /* start of ECSA addr   */
GDAECSAH = D2x(GDAECSA)                      /* display in hex       */
ECSAEND  = (GDAECSAS*1Ø24) + GDAECSA - 1     /* end of ECSA          */
```

```
ECSAEND  = D2x(ECSAEND)                       /* display in hex       */
GDAEPVT  = C2d(Storage(D2x(CVTGDA + 168),4)) /* start of EPVT addr   */
GDAEPVTH = D2x(GDAEPVT)                       /* display in hex       */
EPVTEND  = (GDAEPVTS*1024*1024) + GDAEPVT - 1 /* end of EPVT         */
EPVTEND  = D2x(EPVTEND)                       /* display in hex       */
Queue '          '
Queue 'Virtual Storage Map:'
Queue '          '
Queue '     Storage Area     Start      End           Size'
Queue '          '
Queue '              PSA     00000000   00000FFF        4K'
Queue '           System     00001000   00004FFF       16K'
Queue '      Private V=R    '     Right(VRSTRT,8,'0') ' ' ,
   Right(VREND,8,'0')             Right(GDAVRSZ,8,' ')'K'
Queue '      Private V=V    '     Right(VVSTRT,8,'0') ' ' ,
   Right(VVEND,8,'0')             Right(GDAPVTSZ-20,8,' ')'K'
Queue '              CSA    '     Right(GDACSAH,8,'0') ' ' ,
   Right(CSAEND,8,'0')            Right(GDACSASZ,8,' ')'K'
Queue '             MLPA    '     Right(CVTMLPAS,8,'0') ' ' ,
   Right(CVTMLPAE,8,'0')          Right(MLPASZ,8,' ')'K'
Queue '             FLPA    '     Right(CVTFLPAS,8,'0') ' ' ,
   Right(CVTFLPAE,8,'0')          Right(FLPASZ,8,' ')'K'
Queue '             PLPA    '     Right(CVTPLPAS,8,'0') ' ' ,
   Right(CVTPLPAE,8,'0')          Right(PLPASZ,8,' ')'K'
Queue '              SQA    '     Right(GDASQAH,8,'0') ' ' ,
   Right(SQAEND,8,'0')            Right(GDASQASZ,8,' ')'K'
Queue '      R/W Nucleus    '     Right(CVTRWNS,8,'0') ' ' ,
   Right(CVTRWNE,8,'0')           Right(RWNUCSZ,8,' ')'K'
Queue '      R/O Nucleus    '     Right(CVTRONS,8,'0') ' ' ,
   Right('FFFFFF',8,'0')          Right(RONUCSZB,8,' ')'K',
   '(Spans 16M line)'
Queue '             16M line ———————————'
Queue ' Ext. R/O Nucleus    '     Right('1000000',8,'0') ' ' ,
   Right(CVTRONE,8,'0')           Right(RONUCSZA,8,' ')'K' ,
   '(Total' RONUCSZ'K)'
Queue ' Ext. R/W Nucleus    '     Right(CVTERWNS,8,'0') ' ' ,
   Right(CVTERWNE,8,'0')          Right(ERWNUCSZ,8,' ')'K'
Queue '         Ext. SQA    '     Right(GDAESQAH,8,'0') ' ' ,
   Right(ESQAEND,8,'0')           Right(GDAESQAS,8,' ')'K'
Queue '        Ext. PLPA    '     Right(CVTEPLPS,8,'0') ' ' ,
   Right(CVTEPLPE,8,'0')          Right(EPLPASZ,8,' ')'K'
Queue '        Ext. FLPA    '     Right(CVTEFLPS,8,'0') ' ' ,
   Right(CVTEFLPE,8,'0')          Right(EFLPASZ,8,' ')'K'
Queue '        Ext. MLPA    '     Right(CVTEMLPS,8,'0') ' ' ,
   Right(CVTEMLPE,8,'0')          Right(EMLPASZ,8,' ')'K'
Queue '         Ext. CSA    '     Right(GDAECSAH,8,'0') ' ' ,
   Right(ECSAEND,8,'0')           Right(GDAECSAS,8,' ')'K'
Queue '     Ext. Private    '     Right(GDAEPVTH,8,'0') ' ' ,
   Right(EPVTEND,8,'0')           Right(GDAEPVTS,8,' ')'M'
/*********************************************************************/
```

```
/* Done looking at all control blocks                                 */
/*********************************************************************/
Queue ''  /* null queue to end stack   */
/*********************************************************************/
/* If ISPF is active, browse output - otherwise write to the terminal*/
/*********************************************************************/
If Sysvar(SYSISPF)='ACTIVE' then do
  address ISPEXEC "CONTROL ERRORS RETURN"
  address TSO
  ddnm = 'DD'||random(1,99999)    /* choose random DDname  */
  junk = msg(off)
  "ALLOC FILE("||ddnm||") UNIT(SYSALLDA) NEW TRACKS SPACE(5,5) DELETE",
  " REUSE LRECL(8Ø) RECFM(F B) BLKSIZE(312Ø)"
  junk = msg(on)
  /*                                    */
  "EXECIO * DISKW" ddnm "(FINIS"
  address ISPEXEC "LMINIT DATAID(TEMP) DDNAME("||ddnm||")"
  address ISPEXEC "BROWSE DATAID("||temp")"
  address ISPEXEC "LMFREE DATAID("||temp")"
  junk = msg(off)
  "FREE FI("||ddnm||")"
End
Else do queued()
 Parse pull line
 Say line
End
Exit
/*********************************************************************/
/*  End of main IPLINFO code                                         */
/*********************************************************************/
/*  Start of sub-routines                                            */
/*********************************************************************/
EXTRACT_SYSPARMS:    /* Extract IEASYSxx values from the IPA         */
Arg IEASPARM
IEASPARM = Strip(IEASPARM,T)                /* remove trailing blnks*/
IPAOFF = ((I-1) * 8)                        /* offset to next entry */
IPASTOR = D2x(ECVTIPA + 2152 + IPAOFF)      /* point to PDE addr    */
IPAPDE  = C2x(Storage((IPASTOR),8))         /* point to PDE         */
If IPAPDE = Ø then return   /* parm not specified and has no default */
IPAADDR = Substr(IPAPDE,1,8)                /* PARM address         */
IPALEN  = X2d(Substr(IPAPDE,9,4))           /* PARM length          */
IPAPRM  = Storage((IPAADDR),IPALEN)         /* PARM                 */
/*********************************************************************/
/* CODE to split up page dataset parms  to multiple lines           */
/*********************************************************************/
If IEASPARM = 'NONVIO'   | IEASPARM = 'PAGE' | ,
   IEASPARM = 'PAGE-OPR' | IEASPARM = 'SWAP' then do
  MORE  = 'YES'
  FIRST = 'YES'
  SPLITPOS = 1
```

```
  Do until MORE = 'NO'
    SPLITPOS = Pos(',',IPAPRM)
    If SPLITPOS = Ø then do
      If FIRST = 'YES' then  queue '    'IEASPARM'='IPAPRM
        Else queue '          'IPAPRM
      MORE = 'NO'
    End
    Else do
      IPAPRM_SPLIT = Substr(IPAPRM,1,SPLITPOS)
      If FIRST = 'YES' then queue '    'IEASPARM'='IPAPRM_SPLIT
        Else queue '          'IPAPRM_SPLIT
      IPAPRM  = Substr(IPAPRM,SPLITPOS+1,IPALEN-SPLITPOS)
      FIRST = 'NO'
    End
  End  /* do until */
End
Else  Queue '    'IEASPARM'='IPAPRM  /* not a page ds */
Return

BUILD_IPAPDETB:      /* Build table for look-up for IPA values        */
NUM=1
IPAPDETB.NUM = 'ALLOC   ' ; NUM = NUM + 1
IPAPDETB.NUM = 'APF     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'APG     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'BLDL    ' ; NUM = NUM + 1
IPAPDETB.NUM = 'BLDLF   ' ; NUM = NUM + 1
IPAPDETB.NUM = 'CLOCK   ' ; NUM = NUM + 1
IPAPDETB.NUM = 'CLPA    ' ; NUM = NUM + 1
IPAPDETB.NUM = 'CMB     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'CMD     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'CON     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'CONT    ' ; NUM = NUM + 1
IPAPDETB.NUM = 'COUPLE  ' ; NUM = NUM + 1
IPAPDETB.NUM = 'CPQE    ' ; NUM = NUM + 1
IPAPDETB.NUM = 'CSA     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'CSCBLOC ' ; NUM = NUM + 1
IPAPDETB.NUM = 'CVIO    ' ; NUM = NUM + 1
IPAPDETB.NUM = 'DEVSUP  ' ; NUM = NUM + 1
IPAPDETB.NUM = 'DIAG    ' ; NUM = NUM + 1
IPAPDETB.NUM = 'DUMP    ' ; NUM = NUM + 1
IPAPDETB.NUM = 'DUPLEX  ' ; NUM = NUM + 1
IPAPDETB.NUM = 'EXIT    ' ; NUM = NUM + 1
IPAPDETB.NUM = 'FIX     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'GRS     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'GRSCNF  ' ; NUM = NUM + 1
IPAPDETB.NUM = 'GRSRNL  ' ; NUM = NUM + 1
IPAPDETB.NUM = 'ICS     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'IOS     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'IPS     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'LNK     ' ; NUM = NUM + 1
```

```
IPAPDETB.NUM = 'LNKAUTH ' ; NUM = NUM + 1
IPAPDETB.NUM = 'LOGCLS  ' ; NUM = NUM + 1
IPAPDETB.NUM = 'LOGLMT  ' ; NUM = NUM + 1
IPAPDETB.NUM = 'LOGREC  ' ; NUM = NUM + 1
IPAPDETB.NUM = 'LPA     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'MAXCAD  ' ; NUM = NUM + 1
IPAPDETB.NUM = 'MAXUSER ' ; NUM = NUM + 1
IPAPDETB.NUM = 'MLPA    ' ; NUM = NUM + 1
IPAPDETB.NUM = 'MSTRJCL ' ; NUM = NUM + 1
IPAPDETB.NUM = 'NONVIO  ' ; NUM = NUM + 1
IPAPDETB.NUM = 'NSYSLX  ' ; NUM = NUM + 1
IPAPDETB.NUM = 'NUCMAP  ' ; NUM = NUM + 1
If Substr(FMIDNUM,4,4) >= 66Ø3 then do
   IPAPDETB.NUM = 'OMVS    ' ; NUM = NUM + 1
End
Else do
   IPAPDETB.NUM = 'RESERVED' ; NUM = NUM + 1
End

IPAPDETB.NUM = 'OPI     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'OPT     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'PAGE-OPR' ; NUM = NUM + 1
IPAPDETB.NUM = 'PAGE    ' ; NUM = NUM + 1
IPAPDETB.NUM = 'PAGNUM  ' ; NUM = NUM + 1
IPAPDETB.NUM = 'PAGTOTL ' ; NUM = NUM + 1
IPAPDETB.NUM = 'PAK     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'PLEXCFG ' ; NUM = NUM + 1
IPAPDETB.NUM = 'PROD    ' ; NUM = NUM + 1
IPAPDETB.NUM = 'PROG    ' ; NUM = NUM + 1
IPAPDETB.NUM = 'PURGE   ' ; NUM = NUM + 1
IPAPDETB.NUM = 'RDE     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'REAL    ' ; NUM = NUM + 1
IPAPDETB.NUM = 'RER     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'RSU     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'RSVNONR ' ; NUM = NUM + 1
IPAPDETB.NUM = 'RSVSTRT ' ; NUM = NUM + 1
IPAPDETB.NUM = 'SCH     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'SMF     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'SMS     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'SQA     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'SSN     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'SVC     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'SWAP    ' ; NUM = NUM + 1
IPAPDETB.NUM = 'SYSNAME ' ; NUM = NUM + 1
IPAPDETB.NUM = 'SYSP    ' ; NUM = NUM + 1
IPAPDETB.NUM = 'VAL     ' ; NUM = NUM + 1
IPAPDETB.NUM = 'VIODSN  ' ; NUM = NUM + 1
IPAPDETB.NUM = 'VRREGN  ' ; NUM = NUM + 1
If Substr(FMIDNUM,4,4) >= 66Ø4 then do
   IPAPDETB.NUM = 'RTLS    ' ; NUM = NUM + 1
```

```
End
IPAPDETB.Ø = NUM-1
Return

RDATE:
/***********************************************/
/* Convert MM DD YYYY or YYYY DDD to standard    */
/* date output that includes the day of week and */
/* the century date. A parm of "TODAY" can also  */
/* be passed to the date conversion routine.     */
/* The output format is always as follows:       */
/*      MM/DD/YYYY.JJJ CCCCC WEEKDAY             */
/* The above value will be put in the special    */
/* REXX variable "RESULT"                        */
/* example: CALL RDATE TODAY                     */
/* example: CALL RDATE 1996 3ØØ                  */
/* example: CALL RDATE 1Ø 26 1996                */
/* result:  1Ø/26/1996.3ØØ 35363 Saturday        */
/***********************************************/
arg P1 P2 P3

JULTBL = '0000310590901201511812122432733Ø4334'
DAY.Ø = 'Sunday'
DAY.1 = 'Monday'
DAY.2 = 'Tuesday'
DAY.3 = 'Wednesday'
DAY.4 = 'Thursday'
DAY.5 = 'Friday'
DAY.6 = 'Saturday'

Select
  When P1 = 'TODAY' then do
    P1 = Substr(date('s'),5,2)
    P2 = Substr(date('s'),7,2)
    P3 = Substr(date('s'),1,4)
    call CONVERT_MDY
    call THE_END
  end
  When P3 = '' then do
    call CONVERT_JDATE
    call DOUBLE_CHECK
    call THE_END
  end
  otherwise do
    call CONVERT_MDY
    call DOUBLE_CHECK
    call THE_END
  end
end /* end select */
/* say RDATE_VAL  */
return RDATE_VAL
```

```
/*********************************************/
/*  E N D    O F   M A I N L I N E   C O D E  */
/*********************************************/

CONVERT_MDY:
if P1<1 | P1>12 then do
  say 'Invalid month passed to date routine'
  exit 12
end
if P2<1 | P2>31 then do
  say 'Invalid day passed to date routine'
  exit 12
end
if (P1=4 | P1=6 | P1=9 | P1=11) & P2>3Ø then do
  say 'Invalid day passed to date routine'
  exit 12
end
if P3<19ØØ | P3>2Ø99 then do
  say 'Invalid year passed to date routine'
  exit 12
end
BASE   = Substr(JULTBL,((P1-1)*3)+1,3)
if (P3//4=Ø & P3<>19ØØ) then LEAP= 1
  else LEAP = Ø
if P1 > 2 then BASE = BASE+LEAP
JJJ = BASE + P2

MM   = P1
DD   = P2
YYYY = P3
return

CONVERT_JDATE:
if P1<19ØØ | P1>2Ø99 then do
  say 'Invalid year passed to date routine'
  exit 12
end
if P2<1 | P2>366 then do
  say 'Invalid Julian date passed to date routine'
  exit 12
end
if (P1//4=Ø & P1<>19ØØ) then LEAP= 1
  else LEAP = Ø
ADJ1 = Ø
ADJ2 = Ø
Do MM = 1 to 11
   VAL1 = Substr(JULTBL,((MM-1)*3)+1,3)
   VAL2 = Substr(JULTBL,((MM-1)*3)+4,3)
   if MM >=2 then ADJ2 = LEAP
   if MM >=3 then ADJ1 = LEAP
   if P2 > VAL1+ADJ1 & P2 <= VAL2+ADJ2 then do
        DD = P2-VAL1-ADJ1
```

```
         MATCH = 'Y'
         leave
   end
end
if MATCH <> 'Y' then do
    MM = 12
    DD = P2-334-LEAP
end

YYYY = P1
JJJ  = P2
return

DOUBLE_CHECK:
if MM = 2 then do
   if DD > 28 & LEAP = Ø then do
     say 'Invalid day passed to date routine'
     exit 12
   end
   if DD > 29 & LEAP = 1 then do
     say 'Invalid day passed to date routine'
     exit 12
   end
end
if LEAP = Ø & JJJ > 365 then do
  say 'Invalid Julian date passed to date routine'
  exit 12
end
return

THE_END:
YRC = YYYY-19ØØ
CCCCC = (YRC*365) +(YRC+3)%4 + JJJ
if YYYY > 19ØØ then CCCCC = CCCCC-1
INDEX   = CCCCC//7  /* index to DAY stem */
WEEKDAY =  DAY.INDEX

DD      = Right(DD,2,'Ø')
MM      = Right(MM,2,'Ø')
CCCCC   = Right(CCCCC,5,'Ø')
JJJ     = Right(JJJ,3,'Ø')

RDATE_VAL = MM||'/'||DD||'/'||YYYY||'.'||JJJ||' '||CCCCC||' '||WEEKDAY
return
```

Mark Zelden
Systems Programmer (USA)

Maintaining a PROFILE in ISPF/PDF

A PDF's PROFILE has a lot to do with how the screen behaves when you work with a file.

Each file type has a separate PROFILE. The file type is usually determined from the last segment of the filename. For instance, if the filename is 'A.B.C.JCL', then the file type is 'JCL'. This can be overridden on the ISPF selection screen where you specified the file you wished to work with (see below).

```
+-----------------------------------------+
|                                         |
|  ---------------  EDIT - ENTRY PANEL  ---------------- |
|  COMMAND ===>                           |
|                                         |
|  ISPF LIBRARY:                          |
|                                         |
|     PROJECT ===> ISPFDEMO               |
|                                         |
|     GROUP   ===> MYLIB     ===> MASTER    ===>           ===>
|                                         |
|     TYPE    ===> PLI                    |
|                                         |
|     MEMBER  ===> _        (Blank or pattern for member selection list)
|                                                                  |
|  OTHER PARTITIONED OR SEQUENTIAL DATASET:                       |
|                                         |
|     DATA SET NAME  ===>                 |
|                                         |
|     VOLUME SERIAL  ===>          (If not cataloged)             |
|                                         |
|                                         |
|   DATASET PASSWORD ===>          (If password protected)        |
|                                         |
|                                         |
|  PROFILE NAME      ===>          (Blank defaults to data set type)
|                                         |
|  INITIAL MACRO     ===>       LMF LOCK   ===> YES   (YES, NO or NEVER)
|                                         |
|                                         |
|  FORMAT NAME       ===>       MIXED MODE ===> NO     (YES or NO)
|                                         |
+-----------------------------------------+
```

While in an application, such as EDIT, enter 'PROF' or 'PROFILE' on the command line. Something similar to the example shown in Figure 1 will appear under the command line:

```
File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------
EDIT       MLNABK.LIB.COBOL(MS175) - 01.07          Columns 00007 00078
Command ===>                                          Scroll ===> CSR
****** ***************************** Top of Data ******************************
=PROF> ....COBOL (FIXED - 80).....RECOVERY ON....NUMBER ON COB.....................
=PROF> .....CAPS ON.....HEX OFF.....NULLS ON STD.....TABS ON ;.....SETUNDO REC.....
=PROF> ....AUTOSAVE ON.....AUTONUM OFF.....AUTOLIST OFF.....STATS ON.....................
=PROF> ....PROFILE LOCK.....IMACRO NONE.....PACK OFF.....NOTE ON.....................
=TABS> -  *         *         *         *         *         *         *
=COLS> -1---+---2---+---3---+---4---+---5---+---6---+---7---+--
000100  IDENTIFICATION DIVISION.
000200  PROGRAM-ID.      MMS175IM
000300  DATE-COMPILED.  01/02/00
```

Figure 1: Example output

What does it mean? We will take the items one at a time and elaborate the more interesting ones:

- COBOL (FIXED - 80) – the last part of our dataset name was COBOL, so this profile defaulted to that 'FIXED - 80' is obvious.
- RECOVERY ON – if your session is interrupted by a network problem or mainframe malfunction, the next time you log on and try to get into this function (EDIT), a special screen will come up

asking if you want to resume editing your member. If you choose 'yes', the session will resume where you left off. It would not include any changes you made since the last time you pressed Enter or some other interrupt, such as a PF key, but it will have all your other changes (remember, the mainframe does not know what you are doing on the screen until you press Enter or a PF key). You can change the setting of RECOVERY by entering 'RECOVERY OFF' or 'RECOVERY ON' on the command line.

- NUMBER ON COB – COBOL numbering is on (notice the numbers to the left of the code below our profile). Possible settings (on the command line) are NUMBER ON COB, NUMBER ON STD (numbers in cols 73-80), NUMBER OFF, or UNNUM (remove numbers and set NUMBER to OFF).

```
=PROF> ....CAPS ON....HEX OFF....NULLS ON STD....TABS ON ;....SETUNDO REC......
```

- CAPS ON/OFF – if caps are 'ON', entered text will be changed to caps when you press Enter or a PF key. Existing lower-case text would not be changed. If caps are 'OFF', text will be recorded as entered (upper/lower case).
- HEX ON/OFF – see the discussion on'Hex' in the sidebar.
- NULLS ON STD – trailing blanks on a line, except for the first one, will be nulls (X 'ØØ') rather than spaces (X '4Ø'). Spaces take up space and get in the way of inserting characters, nulls do not. Items entered to the right of a null-filled line will left shift to the end of the existing line when Enter is pressed, space-filled lines will stay where they are put. If you press 'End' to erase to end of line, you always create nulls until Enter is pressed, then this setting tells the computer what to do with the deleted area – leave as nulls or convert to spaces. Using the 'Delete' key produces nulls to the right of the line as it left-shifts. If the field is entirely empty, it is written as all spaces.
- ON ALL – specifies that all trailing blanks and all-blank fields are written as nulls.
- OFF – specifies that trailing blanks in each data field are written as spaces.

- TABS ON – tabbing is on and the logical tab character is (a semi-colon). You can enclose the character in quotes (' or "), although this is not necessary unless a quote or a comma (,) is used as the tab character.
- TABS OFF – turns tabs mode off, which means that logical tabs cannot be used. Attribute bytes are deleted from all hardware tab position.
- TABS STD – activates all hardware tab positions (asterisks) that contain a blank or null character. The editor inserts attribute bytes, which cannot be typed over, at these positions. STD is the default operand.
- TABS ALL – causes an attribute byte to be inserted at all hardware tab positions. Characters occupying these positions are blanked out and the attribute bytes cannot be typed over. We will discuss TABS more completely in a subsequent article.
- SETUNDO REC – enable the 'UNDO' command by saving changes in the recovery file (REC or RECOVER) or memory (STG, STORE, STOR, or STO). Command line settings are SETUNDO REC, SETUNDO OFF, SETUNDO STO, etc. If RECOVERY is ON, SETUNDO OFF is the same as SETUNDO REC. If RECOVERY is OFF, it will be turned on by this command:

```
=PROF> ....AUTOSAVE ON....AUTONUM OFF....AUTOLIST OFF....STATS ON.......
```

- AUTOSAVE ON – automatically saves your file and changes when you exit the session entering END on the command line, pressing PF3, etc. Entering CAN on the command line will leave the session without saving your changes. If AUTOSAVE is OFF, you will have to enter SAVE on the command line before you exit the session.
- AUTONUM OFF – when you insert new lines they will be numbered between the existing lines until the computer runs out of numbers, then as many lines as necessary after the new work will be renumbered to accommodate the inserts. You will have to enter 'RENUM' on the command line to refresh the numbers. When this is ON, inserted lines will cause all following line

number to be re-sequenced using the default scheme (number by 100s in the case of COBOL numbering).

- AUTOLIST ON/OFF – this sends a source listing into the ISPF list dataset when you end the edit session (assuming you made changes and saved them). The disposition of the ISPF list dataset depends upon your settings. It will be printed, saved, or deleted when you log off from ISPF.
- STATS ON – update statistics will be generated for this file. This is the information you see when you list the contents of a PDS, such as 'Created' date, 'Changed' date, 'Size', etc
- PROFILE LOCK – when you issue this command, the profile attributes are locked. Any changes made after that will be forgotten when the session ends. Changes during subsequent sessions will also be forgotten when the session is over. If the profile is UNLOCKed, changes made to the profile's attributes will remain and be available the next time that particular profile is used:

```
=PROF> ....PROFILE LOCK....IMACRO NONE....PACK OFF....NOTE ON.........
```

- IMACRO NONE – the IMACRO primary command saves the name of an initial macro in the current edit profile. The editor runs an initial macro after it reads but before it displays data. The macro might initialize empty datasets, define program macros, or initialize PF keys. A complete discussion of initial macros is beyond the scope of this article.
- PACK OFF – the PACK primary command sets pack mode, which controls whether the data is to be stored in packed format.
- NOTE ON – the NOTES primary command sets note mode, which controls whether notes are displayed when a dialog development model is inserted into the data. This is used in conjunction with the MODEL command and is beyond the scope of this article.
- COLS – just what it looks like. Enter COLS in the line command area to get this line anywhere in the screen. It will stay there until cleared:

```
=COLS> -1---+---2---+---3---+---4---+---5---+---6---+---7---+--
```

Another useful command is HILITE. If you have a colour terminal (3270 emulation, etc) it will change the colour of keywords in your code. It will not work in PROCOMM or any other terminal that emulates a monochrome terminal. The HILITE primary edit command is used to change colour highlighting settings. HI and HILIGHT are valid synonyms. The commands are:

HILITE RESET – reset defaults (AUTO, ON, Find and Cursor on).

HILITE ON – set program colouring on (without logic highlighting).

HILITE OFF – set program colouring OFF.

HILITE AUTO – let ISPF determine the language.

HILITE <lang> – force the language. See Supported Languages.

HILITE LOGIC – turn on IF and DO logic matching. See Logic Highlighting.

HILITE IFLOGIC – turn on IF logic matching only.

HILITE DOLOGIC – turn on DO logic matching only.

HILITE NOLOGIC – turn off all logic matching.

HILITE FIND – toggle highlighting FIND strings.

HILITE CURSOR – toggle highlighting of the phrase with the cursor.

HILITE PAREN – toggle matching of parentheses.

HILITE SEARCH – finds the first unmatched END, ELSE, }, or) between the first line in the file, and the first line being displayed. For END, ELSE or } highlighting, you must have the LOGIC enabled. The search for mismatches occurs only for lines above the last displayed line, so you may need to scroll to the bottom of the file.

HILITE IFLOGIC – turn on IF logic matching only.

HILITE DOLOGIC – turn on DO logic matching only.

HILITE NOLOGIC – turn off all logic matching.

HILITE FIND – toggle highlighting FIND strings.

HILITE CURSOR – toggle highlighting of the phrase with the cursor.

HILITE DISABLE – disables all highlighting and removes the action bar.(Note: the DISABLE setting is not retained between edit sessions.)

HILITE – HILITE with no operands presents a dialog that allows you to change various colouring options.

In many cases, the ISPF editor can determine the language of the file you are editing. If you want to override the automatic language determination, specify the language you want on the HILITE command. Valid language names are:

```
AUTO    ASM  C       COBOL   DTL      IDL     JCL     PANEL
PASCAL  PLI  OTHER   REXX    BOOK     SKEL    DEFAULT
```

For example:

```
COMMAND ===> hi cobol
```

will turn on logical highlighting for COBOL program code.

OTHER is a pseudo-language similar to PL/I but with only very basic keywords (DO, END, SELECT, WHEN, IF, THEN, ELSE, etc). OTHER can be used on many languages such as CLIST. OTHER also does not support any compiler directives. DEFAULT is used when AUTO is specified, but no language can be determined.

You can use the edit PROFILE command to see the colouring status. If a language was explicitly selected, the language will be highlighted in RED. Otherwise it will be WHITE.

CLEANING UP THE SCREEN

To get rid of all profiles, tab lines, or column lines, enter RESET on the command line. Entering ‘D’ in an individual line command area will clear that line only.

The HELP command has a lot of information on Profile commands although they are sometimes a bit difficult to navigate through. Remember, in HELP you can make use of UP (PF7), DOWN (PF8), LEFT (PF10), and RIGHT (PF11), as well as Enter to navigate through screens. If a screen has +More, Enter will get the next screen. Enter will often navigate you through everything in a topic.

The following Hex message comes up:

```
-CAUTION- Data contains invalid (non-display) characters. Use command
  ===> FIND P'.' to position cursor to these
```

So enter ‘f p’.’ ’ in the command line and the browser (or whatever you’re using) positions you to the offending line in the list. Now, how to find out what’s really there.

Enter HEX on the command line and the listing will be converted to three lines and a blank line for each original line that was there (with a lot fewer lines per page). The lines will be: the original line in regular characters, followed by two lines of hex, each hex equivalent directly beneath the original character.

```
ABCD EFG 123
CCCC4CCC4FFF00000000 etc.
12340567012300000000
```

When you have finished, enter ‘HEX OFF’ and things will return to normal.

You can use Hex anytime you need to see the Hex equivalent of something. If you are in EDIT mode, you can edit the hex equivalent lines to produce characters not on your keyboard or to modify packed decimal or binary fields.

Alan Kalar
Systems Programmer (USA)

If you want to contribute an article to *MVS Update*, a copy of our *Notes for contributors* can be downloaded from our Web site. The URL is: www.xephon.com/contnote.html.

MacKinney Systems has announced JES Queue Client for Printers (JQP) and VTAM Virtual Printer (VVP). JQP, a VTAM-based application, prints reports from the JES output queue to network attached printers defined to a VTAM or TCP/IP LPD Daemon. It supports SNA, NON-SNA, and SCS VTAM printers. Reports in the JES output queue are selected based on their DESTID and printed on the pre-defined printer.

The software requires neither CICS nor TSO and printers can be added dynamically without the need to IPL or recycle VTAM or the JQP region. TCP/IP printers are supported using the standard Line Printer Daemon Protocol, RFC 1179.

Machine code and ASA control characters are supported, as is full FCB emulation using the FCB images from the SYS1.IMAGELIB library. A standard separator page is provided and an exit is available for customization.

There are commands available to display and manipulate printers and reports selected for printing. There is support for forms mounting and it has a number of security features.

For further information contact:

MacKinney Systems Inc, 2740 S Glenstone Avenue, 103 Springfield, MO 65804, USA.
Tel: (417) 882 8012
Fax: (417) 882 7569

http://www.mackinney.com

* * *

Metagon Technologies is to upgrade its DQbroker enterprise integration software with the addition of Cross Access's SERIESfour mainframe access software. The combined products, say the vendors, deliver the same functions for IBM legacy systems that were previously available only with open systems relational databases.

DQbroker provides access to most major relational databases, allowing mixed data to be accessed, joined, and managed as a single logical relational database.

Real-time data access and joins across relational and non-relational data will be presented in a global or unified view, regardless of platform.

Cross Access is the OEM of the IMS and VSAM access technology in IBM's Classic Connect.

For further information, contact:

Metagon Technologies, PO Box 2810, Matthews, NC 28106-2810, USA.
Tel: (704) 847 2390.
Fax: (704) 847 4875

http://www.metagon.com.

* * *