5 MQ update

November 1999

In this issue

3 Programming MQSeries with Visual Basic

11 MQSeries naming standards

16 A system generator for MQSeries

29 Creating QM definition scripts from the BSDS

48 MQ news

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: +44 1635 550955
e-mail: HarryL@xephon.com

North American office

Xephon/QNA
1301 West Highway 407, Suite 201-405
Lewisville, TX 75077-2150
USA
Telephone: +1 940 455 7050

Contributions

Articles published in *MQ Update* are paid for at the rate of £170 ($250) per 1000 words and £90 ($140) per 100 lines of code. For more information about contributing an article, please check Xephon's Web site, where you can download *Notes for Contributors*.

MQ Update on-line

Code from *MQ Update* is available from Xephon's Web site at www.xephon.com/mqupdate.html (you'll need the user-id shown on your address label to access it). If you've a problem with your user-id or password call Xephon's subscription department on +44 1635 33886.

Editor

Harry Lewis

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, scripts, and other contents of this journal before making any use of it.

Subscriptions and back-issues

A year's subscription to *MQ Update*, comprising twelve monthly issues, costs £255.00 in the UK; $380.00 in the USA and Canada; £261.00 in Europe; £267.00 in Australasia and Japan; and £265.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the July 1999 issue, are available separately to subscribers for £22.50 ($33.50) each including postage.

Printed in England.

Programming MQSeries with Visual Basic

MQSeries provides a common API, or Message Queue Interface, to enable applications to communicate with queue managers. Using Visual Basic, there are two ways to access MQSeries' functionality:

- Through the procedural MQSeries API, which is implemented by Dynamic Link Libraries ('DLLs') on NT.
- Through ActiveX Automation Classes.

This article shows you how to enable your application to work with MQSeries using the procedural approach.

DECLARING A FUNCTION

In order to use functions from an MQSeries DLL, you must first declare the functions so that Visual Basic knows where to find them. Visual Basic also needs to know the parameters that the function expects to be passed and what (if anything) the function returns. Generally, the declaration follows the syntax shown below (note the use of the continuation character, '➤', to indicate that one line of code maps to more than one line of text).

```
DECLARE FUNCTION function LIB "dll" [ALIAS "alias name"]
➤  ([argument list]) As return type
```

So, in order to declare an MQSeries API function, we use the following syntax:

```
DECLARE FUNCTION MQCONN LIB "MQIC32.DLL"
➤  (By Val QMgrName As String, Hconn As Long, CompCode As Long,
➤  Reason As Long) As Integer
```

Here we declare the function *MQCONN* that's contained in the library *mqic32.dll*, along with the parameters that the function requires. You could specify a return type, if appropriate (*Integer*, in the example above).

PASSING ARGUMENTS

By default, Visual Basic passes all parameters by reference. This

means that, rather than passing the value of a variable, Visual Basic passes a pointer to the variable. As this is the default, you must explicitly declare that a variable is being passed by value using the keyword *By Val*, for example:

```
DECLARE FUNCTION MQCLOSE LIB "MQIC32.DLL" (By Val Hconn As Long,
➤  Hobj As Long, By Val As Long, CompCode As Long,
➤  Reason As Long) As Integer
```

Parameters that are input-only are passed by value. If you are not sure whether to pass the variable by value or by reference, it's safer to pass it by value. The worst thing that can happen is that an unexpected value is passed. Notice that the connection handle, *Hconn*, in the two examples above was passed by reference in the *MQCONN* function but by value in *MQCLOSE*. This is because the handle receives a unique identifier when a connection to the queue manager is established, but is just used as an input parameter to identify the thread to which a call belongs when other functions are called, such as a call to close a queue.

CREATING YOUR APPLICATION

All function declarations, constants, global variables, structures, and initializations are contained in a Visual Basic module or header file. MQSeries provides a header file called *cmqb.bas* that is part of the MQI definition.

Structures must be initialized to contain pre-defined values. If you don't initialize structures, your application will encounter 'invalid structure' errors. You can define subroutines in the header file that your application may invoke to initialize the MQI structures to their default values. These subroutines should be called before any MQI function calls are made.

The header file provided contains several subroutines with names of the form *MQxxx_DEFAULTS*, where *MQxxx* is the name of a structure. These are used in the following way:

```
MQMD_DEFAULTS (MQMD)          'Initialize message descriptor'
MQPMO_DEFAULTS (MQPMO)        'Initialize put-message options'
```

CONNECTING AND DISCONNECTING A QUEUE MANAGER

You can connect to a specific queue manager or the default one using the *MQCONN* call. Remember that the queue manager to which you connect must be local to the task – in other words, it must belong to the same system. The queue manager is an input parameter, so you must supply a value. Below is an example of how to connect to a queue manager:

```
Dim QMgrNm As Long            'queue manager name
Dim Hconn As Long             'connection handle
Dim CompCode As Long          'completion code
Dim Reason As Long            'return code qualifying CompCode

MQCONN QMgrNm, Hconn, CompCode, Reason
```

Connect to the default queue manager by passing a blank value as the queue manager name, or connect to a specific queue manager by specifying its name. The connection handle must be declared as a global variable, as it's used in all subsequent MQI calls.

After your application has finished interacting with the queue manager, you can break the connection using the *MQDISC* function. This requires the connection handle that you acquired when you issued the call to *MQCONN*. Below is an example of how to disconnect from a queue manager.

```
Dim Hconn As Long             'Connection handle
Dim CompCode As Long          'Completion code
Dim Reason As Long            'Reason code qualifying CompCode

MQDISC Hconn, CompCode, Reason
```

OPENING AND CLOSING OBJECTS

In order for your application either to send and receive messages or to set or inspect the attributes of an object, we must first open the object. We use the *MQOPEN* function to open an object, as shown in the example below of opening a queue.

```
Dim OpenOptions As Long      'Open option that controls the
                              'action of MQOPEN
Dim ObjDesc As MQOD          'Object descriptor structure
Dim CompCode As Long         'Completion code
```

```
Dim Reason As Long           'Reason code qualifying CompCode

MQOPEN Hconn, ObjDesc, OpenOptions, Hobj, CompCode, Reason
```

The input of the call comprises the connection handle returned by *MQCONN*, the description of the object to be opened using the *MQOD* structure, and one or more *OpenOptions*. An object handle (*Hobj*) is returned on successful completion. You can use this handle within the same thread as the call that returns it, and it is typically used in subsequent MQ calls.

When you are through using the object, you can close it using the function MQCLOSE (below is an example of closing a queue).

```
Dim CloseOptions As Long    'Close options that control the
                            'action of MQCLOSE
Dim CompCode As Long        'Completion code
Dim Reason As Long          'Reason code qualifying CompCode

MQCLOSE Hconn, Hobj, CloseOptions, CompCode, Reason
```

PUTTING AND GETTING A MESSAGE

You can place a message in a queue with a call to *MQPUT* (successive calls to this function put multiple messages on the same queue after an initial *MQOPEN* call). If you want to place one message and immediately close the queue, use the *MQPUT1* call.

Below is an example of putting a message on a queue.

```
Dim MsgDesc As MQMD          'Message descriptor
Dim PutMsgOpts As MQPMO      'Put message options that control
                             'the action of MQPUT
Dim BufferLength As Long     'Length of message to 'put'
Dim Buffer As String         'Message to 'put'
Dim CompCode As Long         'Completion code
Dim Reason As Long           'Reason code qualifying CompCode

MQPUT Hconn, Hobj, MsgDesc, PutMsgOpts, buflen, buffer, _
      CompCode, Reason
```

As input to this call, we provide the connection handle (*Hconn*), the object handle (*Hobj*), the description of the message to place using the *MQMD* structure, control information using the *MQPMO* structure, the length of the message, and the message itself. The length of the

buffer to be passed should be equal to the size of the buffer, which can be done by declaring *BufferLength = Len(Buffer)*.

When messages are in a queue, you can retrieve them using the *MQGET* call. As with *MQPUT*, you can get a single message with one call and multiple messages with repeated *MQGET*s. There are two ways to get a message from a queue: destructively (which removes the message from the queue) and non-destructively ('browse' the queue). You control the way you get a message (eg input, browse, or both) when you issue an *MQOPEN* call.

Below is an example of getting a message from a queue.

```
Dim Hconn As Long          'Connection handle'
Dim Hobj As Long           'Object handle'
Dim MsgDesc As MQMD        'Message descriptor'
Dim GetMsgOpts As MQGMO    'Options that control the
                           'action of MQGET'
Dim BufferLength As Long   'Length in bytes of the buffer area'
Dim Buffer As String       'Area to contain the message data'
Dim DataLength As Long     'Length of the message'
Dim CompCode As Long       'Completion code'
Dim Reason As Long         'Reason code qualifying CompCode'

MQGET Hconn, Hobj, MsgDesc, GetMsgOpts, BufferLength, Buffer, _
      DataLength, CompCode, Reason
```

MQGET takes the same parameters as *MQPUT* with the exception of the *MQPMO* structure, which is replaced by *MQGMO*, and an additional parameter that holds the length of the message that is received. On successful completion, *MQGET* returns the data and the length of the data.

The string that is passed to *MQGET* should be large enough to contain the data being received, and you can ensure this is the case by defining the string, as follows:

```
Dim Buffer As String * 100
```

Alternatively, set the length of the string explicitly at runtime:

```
Buffer = String (100, " ")
```

If you're not sure of the length of the data being returned, err on the side of caution when setting the length of the string. The string returned will not contain data if it's too small.

The functions *MQPUT* and *MQGET* require a definition for the data buffer. C defines this using a pointer to a character string. This allows the C programmer to pass a pointer to another type of data, for example an integer or a structure. In Visual Basic the definition of the buffer is:

```
By Val Buffer as String
```

This is required to force the conversion of Visual Basic strings into C-type strings (null-terminated strings). Note that Visual Basic doesn't support function overloading and requires type checking. Therefore, in order to pass another type of data, you should explicitly define it as such.

Below is a simple application written using Visual Basic 5.0. It shows the most commonly used MQI calls, such as connecting and disconnecting to a queue manager, opening and closing a queue, and putting a message on a queue. Be sure to create all the controls before using this code.

SAMPLE CODE FOR PUTTING A MESSAGE TO A QUEUE

```
Option Explicit
Dim gHconn As Long
Dim gHobj  As Long

Private Sub Form_Load()
    Dim CompCode As Long
    Dim Reason   As Long

    '=========================
    'Initialize the structures
    '=========================
    MQ_SETDEFAULTS

    '====================================
    'Connect to the default queue manager
    '====================================
    MQCONN "", gHconn, CompCode, Reason

    '============================================================
    'If there is a failure, report reason and end application
    '============================================================
    If CompCode = MQCC_FAILED Then
        If Reason = 2059 Then
```

```
            MsgBox "Cannot connect to queue manager.", vbOKOnly, _
                   "Reason(2059)  MQRC_Q_MGR_NOT_AVAILABLE"
        End If
    End If
End Sub

Private Sub cmdOpen_Click()
    Dim OpenOpts  As Long
    Dim ObjDesc   As MQOD
    Dim CompCode  As Long
    Dim Reason    As Long

    '=========================================================
    'If the connect was successful, open the queue specified
    'in the queue textbox
    '=========================================================
    If gHconn Then
        MQOD_DEFAULTS ObjDesc

        ObjDesc.ObjectName = UCase(QueueNm.Text) 'change to queue names
                                                 'to uppercase

        OpenOpts = MQOO_OUTPUT + MQOO_FAIL_IF_QUIESCING

        MQOPEN gHconn, ObjDesc, OpenOpts, gHobj, CompCode, Reason

        '=========================
        'If not successful, report the reason
        '=========================
        If CompCode <> MQCC_OK Then Beep
            If Reason = 2085 Then
               MsgBox "Unable to open the Queue. ", vbOKOnly, _
                      "ReasonCode = (2085) - MQRC_UNKNOWN_OBJECT_NAME"
            End If

            If Reason = 2087 Then Beep
                MsgBox "Unable to open the Queue.", vbOKOnly, _
                       "ReasonCode = (2087) - MQRC_REMOTE_Q_MGR"
            End If
        Else
            cmdOpen.Enabled = False
            cmdClose.Enabled = True
            cmdPut.Enabled = True
        End If
    End If
End Sub

Private Sub cmdPut_Click()
    Dim MsgDesc     As MQMD
    Dim PutMsgOpt   As MQPMO
```

```
    Dim buflen      As Long
    Dim buffer      As String
    Dim CompCode    As Long
    Dim Reason      As Long

    '================================================================
    'Initialize and setup MQMD and MQPMO to their respective values
    '================================================================

    MQMD_DEFAULTS MsgDesc
    MsgDesc.Format = MQFMT_STRING   'format message as string

    MQPMO_DEFAULTS PutMsgOpt

    '============================================================
    'Get the message from the text box to be put to the queue
    '============================================================
    buflen = Len(txtPut.Text)
    buffer = txtPut.Text
    MQPUT gHconn, gHobj, MsgDesc, PutMsgOpt, buflen, buffer, _
          CompCode, Reason

    '====================================================================
    'To see the messages you put, add the message to the Log listbox
    '====================================================================
    listLog.AddItem buffer
    listLog.ListIndex = listLog.ListCount - 1

    If CompCode <> MQCC_OK Then Beep
End Sub

Private Sub cmdClose_Click()
    Dim CloseOpts As Long
    Dim CompCode As Long
    Dim Reason As Long

    '======================================
    'Close the queue if opened successfully
    '======================================
    If gHobj Then
        CloseOpts = 0
        MQCLOSE gHconn, gHobj, CloseOpts, CompCode, Reason

        If CompCode <> MQCC_OK Then
            Beep
            MsgBox "MQCLOSE ended with Reason Code = " + _
                    Str(Reason), vbOKOnly, "Error on Close"
        Else
            cmdOpen.Enabled = True
            cmdClose.Enabled = False
```

```
            cmdPut.Enabled = False
        End If
    End If
End Sub

Private Sub Form_Unload(Cancel As Integer)
    Dim CompCode As Long      'completion code
    Dim Reason As Long        'reason code

    '=========================================================
    'If queue manager connected successfully - then disconnect
    '=========================================================

    If gHconn Then
        MQDISC gHconn, CompCode, Reason
        If CompCode <> MQCC_OK Then
            MsgBox "MQDISC ended with Reason Code = " + _
                    Str(Reason), vbOKOnly, "Error on Disconnect"
    End If
End Sub
```

Rommel Abdon
Senior Support Engineer
Client Server Technologies (The Philippines)

MQSeries naming standards

So, you're about to implement MQSeries in your enterprise, but just how ready are you? As I mentioned in a previous article published in *MQ Update* (*MQSeries and Windows NT Security*, published in the July 1999 issue), there are several important areas that have to be addressed before an MQSeries implementation: security, system resources available on the various platforms on which MQSeries is to run, network connectivity of these platforms, and naming standards for queue managers and related objects.

The last article dealt with security in a Windows NT environment. In this article, I discuss several naming conventions and the rationale behind them. The standard you choose may be instrumental in the

success of your implementation of MQSeries. As your MQ environment grows and you begin adding more heterogeneous platforms with more and more MQ objects, it is imperative to have a good naming standard not only in place but actively enforced. This standard can ease the burden of the MQSeries administrator by helping him or her to locate objects throughout the MQSeries configuration. For the purpose of this article I will discuss the three main components of MQSeries: queue managers, queues, and channels.

A good starting point in any discussion of MQSeries naming conventions is the MQSeries Application Programming Guide. I won't go into all of the rules contained in this manual, only those that are important to our discussion:

Rule 1 An MQSeries queue, process, 'namelist', and channel can all have the same name, as they are different types of object. An MQSeries object cannot have the same name as another object of the same type belonging to the same Queue Manager.

Rule 2 Names in MQSeries are case sensitive.

Rule 3 Names beginning with 'SYSTEM' are reserved.

Rule 4 The fully qualified name of a queue has two parts:

– The name of the owning queue manager

– The local name of the queue itself.

These rules describe what is allowed, not what is recommended, and I strongly suggest that you read all of the rules and understand them before you develop your own standard.

The first object to be named in any installation is the queue manager itself. As discussed later in this article, a good queue manager name is useful when it comes to selecting a naming standard for other objects. Most organizations choose one of two alternatives – they either use the system name as the queue manager name, adding a number or letter to distinguish between multiple queue managers on the same system (CSC1, CSC2, MVSA, MVSB) or they name them according to whether they're in production, test, or development

systems (TEST1, PROD1, DEV2). Some alternatives I have seen combine the two using the system name and its usage (WINAPROD, WINBTEST, CSCDEV1, CSCDEV2) or using the application as part of the name (PAYROLL1, ACCTREC). There are a few pitfalls to be aware of in choosing any naming scheme. First, queue manager names should NOT be duplicated within an enterprise. This can make troubleshooting problems difficult, and some third-party monitoring and administration software gets confused when trying to issue commands to the queue managers.

Another two pitfalls are specific to MVS. The first is that MVS queue manager names are limited to four characters. This is because queue managers are run as MVS subsystems, which use four character identifiers. For this reason, each byte should be meaningful. One suggested convention I have seen is as follows:

- Byte one and two identify the department:

  AR for 'Accounts Receivable'

  or are constant:

  QM 'Queue Manager', for all queue managers.

- Byte three identifies the MVS image:

  A for 'MVSA'.

- Byte four identifies the usage:

  P for 'Production'

  T for 'Test'

  D for 'Development'.

The full name may thus be *ARAP* or *QMBP*.

The second difficulty has to do with case sensitivity. Given that, under some circumstances, MVS accepts only uppercase letters, I recommended that you use only uppercase letters in the names of MQSeries objects when MVS is part of your MQ environment.

Whichever method you choose, you must use it consistently, or the topology of your MQSeries network will become more and more

confusing as your enterprise grows.

Queues come in various flavours. There are queues for NORMAL usage (that is, queues that are used by applications to ‘put’ or ‘get’ messages) and XMIT queues (those used for holding messages that are destined for a remote queue manager). For the sake of brevity, I’ll limit the discussion of NORMAL queues to LOCAL and REMOTE queues.

When it comes to naming NORMAL queues, there are several schools of thought. One is to include the queue type in the queue name:

- LOCAL.QUEUE1
- REMOTE.QUEUE2
- QL.ABCD.

Another is to name the queue after the queue manager:

- ARAP.QUEUE1
- QMAT.QUEUE3

Yet another is to name the queue with respect to the application that it services:

- PAYROLL.Q1
- AR.Q2.

The problem with the first two methods, as I see it, is that the application loses some of its portability. For example, say that application PAYROLL runs on system SYSA and deposits messages in queue LOCAL.QUEUE1. Now, suppose that application APPL1 reads the messages for further processing on the same system. If APPL1 needs to be moved to another system as a result of resource availability, you then need to redefine the local queue LOCAL.QUEUE1 as a remote queue – even though the queue name is still LOCAL.QUEUE1. You can see the confusion. The proponents of this naming convention point out that they can display all queues of the same type using wild cards – for instance:

```
DIS Q(LOCAL*) ALL
```

However, the same can be accomplished by specifying the TYPE subparameter:

```
DIS Q(*) TYPE(QL) ALL
```

Including the queue manager name in the queue name can lead to similar portability problems when applications have to be moved from one system to another. The last method does not cause portability problems and clearly identifies the application that uses the queue.

For XMIT queues, the standard I recommend is to give the queue the same name as the target queue manager. Again, whatever you decide on as the standard, the most important thing is to be consistent.

Next come the inbound RECEIVER/REQUESTER channels and outbound SENDER/SERVER channels (I intend to cover client connections and server connections in a future article). I find the best names for MQSeries channels are the recommended ones:

```
qm_name1.qm_name2
```

or:

```
qm_name1.to.qm_name2
```

where *qm_name1* and *qm_name2* are the sending and receiving queue managers respectively. This standard makes it easy to identify whether a channel is inbound or outbound and where it goes to or comes from. For example:

QMAP	*QMBP*
QMAP.QMBP	QMAP.QMBP
QMBP.QMPA	QMBP.QMAP

Following the suggested naming standard, it is easy to see that, from QMAP's point of view, QMAP.QMBP is an outbound channel from QMBP, while from QMBP's point of view it's an inbound channel. Functionality is just as clear for QMBP.QMPA. A third numeric qualifier might be used in the case of multiple channels between queue managers (for instance, QMAP.QMBP.1, QMAP.QMBP.2).

In conclusion, this article could arguably be titled 'MQSeries naming suggestions' as there is no hard-and-fast standard. Some conventions

work better in one shop than in another. If you are using a third-party administration tool, the better products include a search utility that simplifies the task of finding an object that needs attention. However, the use of duplicate names in the configuration can make these tools less effective. You should strive for a naming standard that does not diminish the portability of your applications and makes locating any given object as easy as possible. If there is only one thing that you gain from this article, it should be the importance of having a naming standard in place and enforced. Any naming standard is better than no standard at all.

Terrence House CDP
IBM Certified MQSeries Specialist

A system generator for MQSeries

We run MQSeries for MVS/ESA at our site on more than one LPAR, using it with a varied workload on our system. Supporting this set-up requires us to create many QManagers on all our different systems. In order to reduce the workload associated with this, we developed a tool that generates MQSeries systems very easily.

The tool is designed for MQSeries 1.1.4 for MVS/ESA, but it should be easy to change it for other MQSeries releases, and it should also be easy to customize it for your own environment.

Some options are not (yet) in use, but I left them in the panel structure for future implementation.

The tool is menu-driven, and, before you can use it, the rest of your MVS system has to be ready to run MQSeries (for instance, the *SubSystemName* table entries, security attributes, SMS entries, VTAM definitions, and so on, must be ready).

The main menu can be called from a panel using the following statements:


```
+10_MQSeries SYSTEM GENERATOR

10,'PANEL(MQSBUIL)'
```

Note that, for brevity's sake, we've removed some blank lines from the ISPF panels below. The number of blank lines removed is shown in angle brackets ('<' and '>'). Please restore these lines to format the panels to fit the screen.

SYSTEM GENERATOR MAIN MENU

```
MQSBUIL ----------------  MQSeries SYSTEM GENERATOR   ---------------
OPTION  ===>

          1 -  RACF definitions ONLY SECADM (for future use)
          2 -  BootStrapDataSets and logcopy Datasets
          3 -  PageSets
          4 -  Procedures in the SYS1.PROCLIB.DB2
          5 -  Allocate and initiate SCSQPROC library
          6 -  Assemble CSQBDEFV (default QManager for connections)

<INSERT FOUR BLANK LINES HERE>

          11-  Extra functionality (for future use)


          I -  Information about this tool
```

The steps must all terminate with return code zero. If you encounter a problem, the best course of action is to delete the datasets just created, solve the problem, and start again at the beginning.

MQSERIES GENERATOR PANEL DEFINITIONS:

The following panels are invoked:

BGB000M0 (ON-LINE HELP)

```
)ATTR default(¬,%)
@ TYPE(OUTPUT) INTENS(HIGH) CAPS(OFF)  JUST(LEFT)
¬ TYPE(TEXT)   INTENS(LOW)  SKIP(ON)   COLOR(TURQ)
% TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(GREEN)
¢ TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(red)
~ TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(WHITE)
? TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(YELLOW) HILITE(REVERSE)
` TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(green)  HILITE(REVERSE)
```

```
_ TYPE(TEXT)   INTENS(high) JUST(ASIS) COLOR(blue)
+ TYPE(TEXT)   INTENS(LOW)             COLOR(white)
_ TYPE(INPUT)  INTENS(LOW)
)BODY
_------------------------`MQSERIES SYSTEM GENERATOR  _---------------
¬ This tool allows you to generate a complete MQSeries system.
  The system generated is set up using default values.

  Jobs should be started consecutively, and all have to end with
  return code zero.

  Before using this tool, make sure that other necessary components,
  such as MVS, RACF, VTAM, and SMS, are in place.

  Other actions should be carried out on the target LPAR
(eg the application configuration, parameter modules, etc).

<INSERT THREE BLANK LINES HERE>

                                                  (continued on next page)
)PROC
  &ZCONT=BGB000M1
)END
/* COPYRIGHT INTERPAY */
```

BGB000R0 (ON-LINE HELP)

```
)ATTR default(¬,%)
@ TYPE(OUTPUT) INTENS(HIGH) CAPS(OFF)  JUST(LEFT)
¬ TYPE(TEXT)   INTENS(LOW)  SKIP(ON)   COLOR(TURQ)
% TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(GREEN)
¢ TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(red)
~ TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(WHITE)
? TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(YELLOW) HILITE(REVERSE)
` TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(green)  HILITE(REVERSE)
_ TYPE(TEXT)   INTENS(high) JUST(ASIS) COLOR(blue)
+ TYPE(TEXT)   INTENS(LOW)             COLOR(white)
_ TYPE(INPUT)  INTENS(LOW)
)BODY
_------------------------`MQSERIES SYSTEM GENERATOR  _---------------

¬  This option is for future use.

<INSERT SIX BLANK LINES HERE>

                            (continued on next page)
)PROC
/* &ZCONT=BGC000R1*/
```

```
)END
/* COPYRIGHT INTERPAY */
```

BGB000M1 (ON-LINE HELP)

```
)ATTR default(¬,%)
@ TYPE(OUTPUT) INTENS(HIGH) CAPS(OFF)  JUST(LEFT)
¬ TYPE(TEXT)   INTENS(LOW)  SKIP(ON)   COLOR(TURQ)
% TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(GREEN)
¢ TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(red)
~ TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(WHITE)
? TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(YELLOW) HILITE(REVERSE)
` TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(green)  HILITE(REVERSE)
_ TYPE(TEXT)   INTENS(high) JUST(ASIS) COLOR(blue)
+ TYPE(TEXT)   INTENS(LOW)             COLOR(white)
_ TYPE(INPUT)  INTENS(LOW)
)BODY
_-------------------------`MQSERIES SYSTEM GENERATOR  _---------------
¬ The generator should be used only by authorized personnel.

  The security you need is ...........   (insert your own standard
                                          security ID here).

  Before you use this tool, make sure the other components are in
  place (MVS, RACF, VTAM, etc)

<INSERT FOUR BLANK LINES HERE>

)PROC
/* &ZCONT=BGB000M1 */
)END
/* COPYRIGHT INTERPAY */
```

BGM00000 (ON-LINE HELP START MENU)

```
)ATTR
@ TYPE(OUTPUT) INTENS(HIGH) CAPS(OFF)  JUST(LEFT)
$ TYPE(OUTPUT) INTENS(LOW)  CAPS(OFF)  JUST(ASIS)
% TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(GREEN)
¢ TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(TURQ)
~ TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(WHITE)
? TYPE(TEXT)   INTENS(LOW)  JUST(ASIS) COLOR(YELLOW)
# TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(YELLOW)
` TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(YELLOW) HILITE(REVERSE)
_ TYPE(TEXT)   INTENS(high) JUST(ASIS) COLOR(blue)
+ TYPE(TEXT)   INTENS(LOW)             COLOR(white)
_ TYPE(INPUT)  CAPS(ON)
)body
```

```
_-------------------------`MQSERIES SYSTEM GENERATOR  _---------------
%OPTION ===>_ZCMD                                                      +
%
+
   This tutorial gives on-line documentation for the MQSERIES SYSTEM
   GENERATOR


              %M+ MQSERIES SYSTEM GENERATOR

              %R+ RACF


)PROC
   &ZSEL = TRANS(&ZCMD
                 M,BGB000M0
                 R,BGB000R0
                 )
)END
/* COPYRIGHT INTERPAY */
```

MQSBUIL (GENERATOR MAIN MENU)

```
)ATTR
@ TYPE(OUTPUT) INTENS(HIGH) CAPS(OFF)  JUST(LEFT)
% TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(GREEN)
¢ TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(RED)
~ TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(WHITE)
? TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(YELLOW) HILITE(REVERSE)
! TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(YELLOW)
# TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(YELLOW)
` TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(GREEN) HILITE(REVERSE)
_ TYPE(TEXT)   INTENS(high) JUST(ASIS) COLOR(blue)
+ TYPE(TEXT)   INTENS(LOW)             COLOR(white)
_ TYPE(INPUT)  INTENS(LOW)
)BODY
_-------------------------`MQSERIES SYSTEM GENERATOR  _---------------
%OPTION  ===>_zcmd
+
          ~1 -  RACF definitions ?ONLY SECADM+ (for future use)
          ~2 -  BootStrapDataSets and logcopy Datasets
          ~3 -  PageSets
          ~4 -  Procedures in the SYS1.PROCLIB.DB2
          ~5 -  Allocate SCSQPROC library and initialize datasets
          ~6 -  Assemble CSQBDEFV (default QManager for connections)

<INSERT TWO BLANK LINE HERE>

+
          ~11-  Extra functionality (for future use)
```

```
+
+
          !I -  Information+
+
)INIT
  .HELP = TUTORPAN              /* insert name of the tutorial panel */
/*&zcmd=' '  */
/*)REINIT    */
/*  &zcmd=&z Refresh(zcmd)  */
)PROC
   &ZSEL=TRANS(TRUNC(&ZCMD,'.')
              1,'CMD(MQSRACF)'
              2,'CMD(MQSDEFB)'
              3,'CMD(MQSDEFP)'
              4,'CMD(MQSDEFS)'
              5,'CMD(MQSDEFA)'
              6,'CMD(MQSDEFC)'
              11,'PANEL(MQSEXTR)'
              I,'PGM(ISPTUTOR) PARM(BGM00000)'
              X,'EXIT'
              ' ',' '
              *,'?')
  &ZTRAIL = .TRAIL
  &PFKEY  = .PFKEY
)END
/* COPYRIGHT INTERPAY */
```

MQSDEFA

```
)ATTR
@ TYPE(OUTPUT) INTENS(HIGH) CAPS(OFF)  JUST(LEFT)
$ TYPE(INPUT)  INTENS(LOW)  PAD(_)
% TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(GREEN)
¢ TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(red)
~ TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(WHITE)
? TYPE(TEXT)   INTENS(LOW)  JUST(ASIS) COLOR(YELLOW)
` TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(green) HILITE(REVERSE)
_ TYPE(TEXT)   INTENS(high) JUST(ASIS) COLOR(blue)
+ TYPE(TEXT)   INTENS(LOW)             COLOR(white)
_ TYPE(INPUT)  INTENS(LOW)
)BODY
_-------------------------`MQSERIES SYSTEM GENERATOR  _---------------
%COMMAND ===>_ZCMD
+
+
+   Allocation SCSQPROC and initial members
+
+
+   MQSeries  system id   ....$z   +
```

```
+   LPAR                  ....$lpar+
+   Volume                ....$vol   +
+
+   Connects to CICS system ....$cicid+
+
+
+   The following members are added to the SCSQPROC library:
+   CSQ4INP1
+   CSQ4INP2
+   CSQ4DISX
+   CSQ4STGC
+   CSQINPX
+   CSQBDEFV
+
+   PF3 = Exit
+
)INIT
  .ZVARS = 'SYSID'
  .HELP = TUTORPAN           /* Insert name of the tutorial panel */
    &sysid=' '
    &lpar=' '
    &vol=' '
    &cicid=' '
    &pfkey=.pfkey
)PROC
    VER (&SYSID,NB,MSG=mqg001)
    VER (&LPAR,NB,MSG=mqg001)
    VER (&vol,NB,MSG=mqg001)
    VER (&cicid,NB,MSG=mqg001)
    &pfkey=.pfkey
)END
/* COPYRIGHT INTERPAY */
```

MQSDEFB

```
)ATTR
@ TYPE(OUTPUT) INTENS(HIGH) CAPS(OFF)  JUST(LEFT)
$ TYPE(INPUT)  INTENS(LOW)  PAD(_)
% TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(GREEN)
¢ TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(red)
~ TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(WHITE)
? TYPE(TEXT)   INTENS(LOW)  JUST(ASIS) COLOR(YELLOW)
` TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(green) HILITE(REVERSE)
_ TYPE(TEXT)   INTENS(high) JUST(ASIS) COLOR(blue)
+ TYPE(TEXT)   INTENS(LOW)             COLOR(white)
_ TYPE(INPUT)  INTENS(LOW)
)BODY
_------------------------`MQSERIES SYSTEM GENERATOR  _---------------
%COMMAND ===>_ZCMD
```

```
+
+
+   Allocate BootStrapDataset and Logcopies
+
+
+   MQSeries  system id   ....$z   +
+   LPAR            ....$lpar+
+   BSDS1volume     ....$bsds1vol+
+   BSDS2volume     ....$bsds2vol+
+   logcopy1.1      ....$log11vol+    logcopy2.1    ....$log21vol+
+   logcopy1.2      ....$log12vol+    logcopy2.2    ....$log22vol+
+   logcopy1.3      ....$log13vol+    logcopy2.3    ....$log23vol+
+
+   Number of cylinders for the logcopy datasets ....$cyls+
+
+
+   PF3 = Exit
+
)INIT
  .ZVARS = 'SYSID'
  .HELP = TUTORPAN           /* Insert name of the tutorial panel */
    &sysid=' '
    &lpar=' '
    &bsds1vol=' '
    &bsds2vol=' '
    &catdvol=' '
    &log11vol=' '
    &log12vol=' '
    &log13vol=' '
    &log21vol=' '
    &log22vol=' '
    &log23vol=' '
    &cyls=' '
    &pfkey=.pfkey
)PROC
    VER (&SYSID,NB,MSG=mqg001)
    VER (&LPAR,NB,MSG=mqg001)
    &pfkey=.pfkey
)END
/* COPYRIGHT INTERPAY */
```

MQSDEFC

```
)ATTR
@ TYPE(OUTPUT) INTENS(HIGH) CAPS(OFF)  JUST(LEFT)
$ TYPE(INPUT)  INTENS(LOW)  PAD(_)
% TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(GREEN)
¢ TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(red)
~ TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(WHITE)
```

```
? TYPE(TEXT)   INTENS(LOW)  JUST(ASIS) COLOR(YELLOW)
` TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(green) HILITE(REVERSE)
_ TYPE(TEXT)   INTENS(high) JUST(ASIS) COLOR(blue)
+ TYPE(TEXT)   INTENS(LOW)             COLOR(white)
_ TYPE(INPUT)  INTENS(LOW)
)BODY
_-------------------------`MQSERIES SYSTEM GENERATOR  _---------------
%COMMAND ===>_ZCMD
+
+
+   Default batch/TSO adapter
+
+
+   MQSeries system id  ....$z   +
+   LPAR                ....$lpar+
+   Volume              ....$vol   +
+
+
+
+
+
+   PF3 = Exit
+
)INIT
  .ZVARS = 'SYSID'
  .HELP = TUTORPAN           /* Insert name of the tutorial panel */
    &sysid=' '
    &lpar=' '
    &vol=' '
    &pfkey=.pfkey
)PROC
    VER (&SYSID,NB,MSG=mqg001)
    VER (&LPAR,NB,MSG=mqg001)
    VER (&VOL,NB,MSG=mqg001)
    &pfkey=.pfkey
)END
/* COPYRIGHT INTERPAY */
```

MQSDEFP

```
)ATTR
@ TYPE(OUTPUT) INTENS(HIGH) CAPS(OFF)  JUST(LEFT)
$ TYPE(INPUT)  INTENS(LOW)  PAD(_)
% TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(GREEN)
¢ TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(red)
~ TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(WHITE)
? TYPE(TEXT)   INTENS(LOW)  JUST(ASIS) COLOR(YELLOW)
` TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(green) HILITE(REVERSE)
_ TYPE(TEXT)   INTENS(high) JUST(ASIS) COLOR(blue)
```

```
+ TYPE(TEXT)   INTENS(LOW)             COLOR(white)
_ TYPE(INPUT)  INTENS(LOW)
)BODY
_------------------------`MQSERIES SYSTEM GENERATOR  _---------------
%COMMAND ===>_ZCMD
+
+
+   Allocation Pagesets
+
+
+   MQSeries  system id   ....$z   +
+   LPAR            ....$lpar+
+   Pageset volume1 ....$vol1ps+
+   Pageset volume2 ....$vol2ps+
+
+   The pagesets are allocated on alternate volumes. (PS00 on
+   volume1, PS01 on volume2, PS02 on volume1, PS03 on volume2,
+   and so on).
+
+
+
+
+   PF3 = Exit
+
)INIT
  .ZVARS = 'SYSID'
  .HELP = TUTORPAN           /* Insert name of the tutorial panel */
    &sysid=' '
    &lpar=' '
    &vol1ps=' '
    &vol2ps=' '
    &pfkey=.pfkey
)PROC
    VER (&SYSID,NB,MSG=mqg001)
    VER (&LPAR,NB,MSG=mqg001)
    VER (&vol1ps,NB,MSG=mqg001)
    VER (&vol2ps,NB,MSG=mqg001)
    &pfkey=.pfkey
)END
/* COPYRIGHT INTERPAY */
```

MQSDEFS

```
)ATTR
@ TYPE(OUTPUT) INTENS(HIGH) CAPS(OFF)  JUST(LEFT)
$ TYPE(INPUT)  INTENS(LOW)  PAD(_)
% TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(GREEN)
¢ TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(red)
~ TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(WHITE)
```

```
? TYPE(TEXT)   INTENS(LOW)  JUST(ASIS) COLOR(YELLOW)
` TYPE(TEXT)   INTENS(HIGH) JUST(ASIS) COLOR(green) HILITE(REVERSE)
_ TYPE(TEXT)   INTENS(high) JUST(ASIS) COLOR(blue)
+ TYPE(TEXT)   INTENS(LOW)             COLOR(white)
_ TYPE(INPUT)  INTENS(LOW)
)BODY
_-------------------------`MQSERIES SYSTEM GENERATOR  _---------------
%COMMAND ===>_ZCMD
+
+
+   Procedure
+   (the procedures are stored in the SYS1.PROCLIB.DB2)
+
+   MQSeries system id  ....$z   +
+   LPAR                ....$lpar+
+
+
+
+
+
+
+   PF3 = Exit
+
)INIT
  .ZVARS = 'SYSID'
  .HELP = TUTORPAN           /* Insert name of the tutorial panel */
    &sysid=' '
    &lpar=' '
    &pfkey=.pfkey
)PROC
    VER (&SYSID,NB,MSG=mqg001)
    VER (&LPAR,NB,MSG=mqg001)
    &pfkey=.pfkey
)END
/* COPYRIGHT INTERPAY */
```

SUBMIT

```
)ATTR
   _ TYPE(INPUT) CAPS(OFF) INTENS(HIGH) FORMAT(&MIXED)

)BODY  WIDTH(&ZWIDTH) EXPAND(__)
%&CIVER EDIT ----------------------_-_-------------------------------+
%COMMAND ===>_ZCMD                  _ _              %SCROLL ===>_Z   %
+ **** TO SUBMIT THIS JOB NOW, TYPE 'SUBMIT' AND PRESS ENTER. **** %
)INIT
  .HELP  = ISR20000
  .ZVARS = 'ZSCED'
```

```
  &MIXED = MIX
  IF (&ZPDMIX = N)
    &MIXED = EBCDIC

)PROC

)END
/* COPYRIGHT INTERPAY */
```

Below are MQSeries Generator's 'exec' definitions. The following execs are invoked:

MQSDEFA

```
ADDRESS TSO
"ALLOC F(ISPFILE) DA('your.jcl.lib') SHR REUSE"
USERID=USERID()
ADDRESS ISPEXEC
"LIBDEF ISPSLIB DATASET ID('your.skel.lib')"
DO
"DISPLAY PANEL (MQSDEFA)"
'FTOPEN'
'FTINCL MQSDEFA'
'FTCLOSE NAME(MQSDEFA)'
"EDIT DATASET('your.jcl.lib(MQSDEFA)') PANEL(SUBMIT)"
END
```

MQSDEFB

```
ADDRESS TSO
"ALLOC F(ISPFILE) DA('your.jcl.lib') SHR REUSE"
USERID=USERID()
ADDRESS ISPEXEC
"LIBDEF ISPSLIB DATASET ID('your.skel.lib')"
DO
"DISPLAY PANEL (MQSDEFB)"
'FTOPEN'
'FTINCL MQSDEFB'
'FTCLOSE NAME(MQSDEFB)'
"EDIT DATASET('your.jcl.lib(MQSDEFB)')  PANEL(SUBMIT)"
END
```

MQSDEFC

```
ADDRESS TSO
"ALLOC F(ISPFILE) DA('your.jcl.lib') SHR REUSE"
USERID=USERID()
```

```
ADDRESS ISPEXEC
"LIBDEF ISPSLIB DATASET ID('your.skel.lib')"
DO
"DISPLAY PANEL (MQSDEFC)"
LOAD="&&LOADSET"
'FTOPEN'
'FTINCL MQSDEFC'
'FTCLOSE NAME(MQSDEFC)'
"EDIT DATASET('your.jcl.lib(MQSDEFC)')  PANEL(SUBMIT)"
END
```

MQSDEFP

```
ADDRESS TSO
"ALLOC F(ISPFILE) DA('your.jcl.lib') SHR REUSE"
USERID=USERID()
ADDRESS ISPEXEC
"LIBDEF ISPSLIB DATASET ID('your.skel.lib')"
DO
"DISPLAY PANEL (MQSDEFP)"
'FTOPEN'
'FTINCL MQSDEFP'
'FTCLOSE NAME(MQSDEFP)'
"EDIT DATASET('your.jcl.lib(MQSDEFP)')  PANEL(SUBMIT)"
END
```

MQSDEFS

```
ADDRESS TSO
"ALLOC F(ISPFILE) DA('your.jcl.lib') SHR REUSE"
USERID=USERID()
ADDRESS ISPEXEC
"LIBDEF ISPSLIB DATASET ID('your.skel.lib')"
DO
"DISPLAY PANEL (MQSDEFS)"
'FTOPEN'
'FTINCL MQSDEFS'
'FTCLOSE NAME(MQSDEFS)'
"EDIT DATASET('your.jcl.lib(MQSDEFS)')  PANEL(SUBMIT)"
END
```

This article concludes in next month's issue of *MQ Update*.

Paul Jansen
Systems Programmer
Interpay (The Netherlands)

Creating QM definition scripts from the BSDS

The need for this program arose during a disaster recovery exercise, when an operation to restore the BSDS failed. Instead of holding up other disaster recovery testing, we decided to build and initialize a new BSDS. However, we then found that the previous QM administrator had used the on-line CSQOREXX utility to carry out all maintenance. This meant that many definitions were lost.

We realized that we had uncovered a substantial source of exposure to the integrity of the system, as no documentation of dynamic definitions is available. The previous administrator told me that he had had to rebuild a BSDS before, and that this had been done manually by displaying each object and then defining it on the new QM.

To address this problem, I developed the program presented in this article, which carries out a *DISPLAY ALL* of all processes, storage groups, queues, and channels. The responses are then formatted to conform to the *CSQINPx* format.

Output is to two *DDNAME*s:

1 *SYSPRINT* to a *SYSOUT* dataset, which lists the replies from the *DISPLAY* command.

2 *MQDEFS*, which would normally be *PARMLIB* members. These then contain the correct inputs for the queue manager.

The program uses so-called ‘concept 14 macros’ (*IF-ELSE-ENDIF* and loop structures), which were distributed on SHARE tapes a few years ago. I repackaged them in one copybook, *PPFC14M0*, as well as adding a *SELECT-WHEN* structure, similar to the equivalent construct in REXX. Another copybook, *PPFGBLC0*, is a list of global macro variables used by *PPFC14M0*.

Note that the assembly is guaranteed to give ‘reentrant check’ warnings. This is because the entire working storage (from label *WRKSTOR* down) is relocated in a dynamic storage area. As far as the assembler knows, these labels are still part of the *RSECT*.

I use both relative and named ‘usings’, so a high-level assembler is required.

The linkage step must include a *DDNAME MQLIB*, which must refer to the MQSeries *SCSQLOAD* dataset.

PMQGETB0

```
PUNCH ' MODE AMODE(31),RMODE(ANY) '
         PUNCH ' SETOPT PARM(REUS(RENT),XREF,MAP) '
         PUNCH ' INCLUDE MQLIB(CSQBSTUB) '
         PUNCH ' ORDER PMQGETB0 '
         PUNCH ' ENTRY PMQGETB0 '
*-------------------------------------------------------------------*
*                                                                   *
* PROGRAM: PMQGETB0 - GET QM DEFINITIONS OFF THE BSDS AND BUILD     *
*          PARMLIB DEFINITIONS.                                     *
*                                                                   *
* MACROS:                                                           *
*          CONCEPT-14 IN COPYBOOK PPFC14M0:                         *
*               IF-ELSE-ENDIF                                       *
*               DO-DOEXIT-ENDDO                                     *
*               SELECT-WHEN-ENDSEL                                  *
*               STRTSRCH-EXITIF-ORELSE-ENDLOOP-ENDSRCH              *
*                                                                   *
*          DATA MANAGEMENT:                                         *
*               OPEN                                                *
*               CLOSE                                               *
*               PUT                                                 *
*               DCB     - DCB DEFINITION                            *
*               DCBD    - DCB DSECT                                 *
*                                                                   *
*          MQSERIES:                                                *
*               CMQODA  - OBJECT DESCRIPTOR                         *
*               CMQMDA  - MESSAGE DESCRIPTOR                        *
*               CMQPMOA - PUT MESSAGE OPTIONS                       *
*               CMQGMOA - GET MESSAGE OPTIONS                       *
*               CMQA    - MQ EQUATES                                *
*                                                                   *
*          MISCELANEOUS:                                            *
*               CALL                                                *
*               REGEQU                                              *
*               STORAGE                                             *
*               WTO                                                 *
*                                                                   *
* SUBROUTINES:                                                      *
*          MQSERIES (ALL INCLUDED IN CSQBSTUB)                      *
*               MQCONN                                              *
```

```
*               MQOPEN                                                *
*               MQPUT                                                 *
*               MQGET                                                 *
*               MQCLOSE                                               *
*               MQDISC                                                *
*                                                                     *
*---------------------------------------------------------------------*
         PRINT OFF,NOPRINT
* LKDPARM=RENT
* STDUSE=NO
         COPY  PPFC14M0               .CONCEPT-14 MACROS
         PRINT ON,NOGEN,NOPRINT
PMQGETBO RSECT
PMQGETBO AMODE 31
PMQGETBO RMODE ANY
         USING *,R15
         B     CODE_START
         DC    AL1(L'EP_LITERAL)
EP_LITERAL     DC  C'PMQGETBO..DATE=&SYSDATC..TIME=&SYSTIME..GENERATE Q+
               UEUE MANAGER''S DEFINITIONS FROM DISPLAY OUTPUT'
CODE_START     DS  0H
         BAKR  R14,0                       .ESA-STYLE SAVE
         DROP  R15
         LR    R12,R15
         USING PMQGETBO,R12,R11
         LA    R11,2048(,R12)
         LA    R11,2048(,R11)
*---------------------------------------------------------------------*
* THE CODE BELOW RELOCATES THE ENTIRE WORKING STORAGE, THUS           *
* MAKING THE CODE REENTRANT, EVEN THOUGH THE ASSEMBLER WILL GIVE      *
* REENTRANT CHECK WARNINGS.                                           *
* THE WORKING STORAGE IS DEFINED ON A PAGE BOUNDARY, MAKING IT EASY   *
* TO DEBUG, SINCE THE LAST THREE NIBBLES OF THE ADDRESS IS THE OFFSET *
* FROM THE START OF WORKING STORAGE.                                  *
*---------------------------------------------------------------------*
         STORAGE OBTAIN,LENGTH=WSLEN,LOC=BELOW,BNDRY=PAGE
         LR    R13,R1
         LR    R14,R1
         LR    R15,R0
         LR    R1,R0
         L     R0,=A(WRKSTOR)
         MVCL  R14,R0
         MVC   4(4,R13),=C'F1SA'           .INDICATE BAKR FORMAT SAVE
         USING WRKSTOR,R13
*---------------------------------------------------------------------*
* END OF RELOCATION TRICK                                             *
*---------------------------------------------------------------------*
         EJECT
PRT      USING IHADCB,SYSPRINT
DEFS     USING IHADCB,MQDEFS
```

```
         EREG  R1,R1
         XR    R15,R15
         BAS   R14,INIT
         IF    LTR,R15,R15,Z
            LA    R9,CMDLIST
            DO    WHILE=(CLI,0(R9),NE,0)
               MVC   CMDLINE,0(R9)
               BAS   R14,GET_ALL_OBJ
            DOEXIT LTR,R15,R15,NZ
               LA    R9,L'CMDLINE(,R9)
            ENDDO
         ENDIF
         ST    R15,MAXRC
         BAS   R14,CLEANUP
RETURN   DS    0H
         L     R4,MAXRC
         LR    R1,R13
         STORAGE RELEASE,LENGTH=WSLEN,ADDR=(1)
         LR    R15,R4
         PR
         EJECT
INIT     DS    0H
         BAKR  R14,0
         BAS   R14,CHECK_PARMS
         LTR   R15,R15
         BNZ   INIT_DONE
         BAS   R14,CONNECT_MQ
         LTR   R15,R15
         BNZ   INIT_DONE
         BAS   R14,OPEN_REPLY
         LTR   R15,R15
         BNZ   INIT_DONE
         BAS   R14,OPEN_REQ
         LTR   R15,R15
         BNZ   INIT_DONE
         OPEN  (SYSPRINT,(OUTPUT)),MODE=31,MF=(E,OPEN)
         LA    R15,8
         TM    PRT.DCBOFLGS,DCBOFOPN
         BZ    INIT_DONE
         OPEN  (MQDEFS,(OUTPUT)),MODE=31,MF=(E,OPEN)
         XR    R15,R15
         TM    DEFS.DCBOFLGS,DCBOFOPN
         BO    INIT_DONE
         LA    R15,8
INIT_DONE      DS  0H
         PR
         EJECT
CHECK_PARMS    DS  0H
         BAKR  R14,0
         L     R10,0(,R1)
```

```
         SELECT
         WHEN  CLC,=H'0',EQ,0(R10)
            WTO   'ERROR: QUEUE MANAGER NAME REQUIRED'
            LA    R15,8
         WHEN  CLC,0(2,R10),GT,=H'4'
            WTO   'ERROR: ILLEGAL QUEUE MANAGER NAME'
            LA    R15,8
         WHEN  NONE
           LH    R1,0(,R10)
           BCTR  R1,0
           EX    R1,COPY_QMGR_NAME
         ENDSEL
         PR
COPY_QMGR_NAME MVC MQ_NAME(0),2(R10)
         EJECT
CONNECT_MQ     DS  0H
         BAKR  R14,0
         CALL  MQCONN,(MQ_NAME,MQ_HANDLE,MQ_RETCODE,MQ_REASON),VL,     +
               MF=(E,CALL_PARMS)
         IF    ICM,R15,15,MQ_RETCODE,NZ
            CVD   R15,DUB
            OI    DUB+7,X'0F'
            UNPK  WTO1+33(3),DUB+6(2)
            L     R0,MQ_REASON
            CVD   R0,DUB
            OI    DUB+7,X'0F'
            UNPK  WTO1+45(5),DUB+5(3)
            WTO   MF=(E,WTO1)
            LA    R15,8
         ENDIF
         PR
         EJECT
OPEN_REPLY     DS 0H
*--------------------------------------------------------------------*
* OPEN A DYNAMIC QUEUE NAMED SYSTEM.PMQGETB0.*, BASED ON             *
* SYSTEM.COMMAND.REPLY.MODEL; MQ WILL GENERATE A UNIQUE VALUE IN     *
* PLACE OF THE '*'                                                   *
*--------------------------------------------------------------------*
         BAKR  R14,0
         CALL  MQOPEN,(MQ_HANDLE,REPLYQ,=A(MQOO_INPUT_SHARED),         +
               REPLY_HANDLE,MQ_RETCODE,MQ_REASON),                     +
               VL,MF=(E,CALL_PARMS)
         IF    ICM,R15,15,MQ_RETCODE,NZ
            CVD   R15,DUB
            OI    DUB+7,X'0F'
            UNPK  WTO3+42(3),DUB+6(2)
            L     R0,MQ_REASON
            CVD   R0,DUB
            OI    DUB+7,X'0F'
            UNPK  WTO3+54(5),DUB+5(3)
```

```
            WTO   MF=(E,WTO3)
            LA    R15,8
         ENDIF
         PR
         EJECT
OPEN_REQ       DS OH
*-------------------------------------------------------------------*
* OPEN QUEUE SYSTEM.COMMAND.INPUT                                   *
*-------------------------------------------------------------------*
         BAKR  R14,0
         CALL  MQOPEN,(MQ_HANDLE,REQUESTQ,=A(MQOO_OUTPUT),          +
               REQUEST_HANDLE,MQ_RETCODE,MQ_REASON),                +
               VL,MF=(E,CALL_PARMS)
         IF    ICM,R15,15,MQ_RETCODE,NZ
            CVD   R15,DUB
            OI    DUB+7,X'0F'
            UNPK  WTO4+44(3),DUB+6(2)
            L     R0,MQ_REASON
            CVD   R0,DUB
            OI    DUB+7,X'0F'
            UNPK  WTO4+56(5),DUB+5(3)
            WTO   MF=(E,WTO4)
            LA    R15,8
         ENDIF
         PR
         EJECT
GET_ALL_OBJ    DS  OH
*-------------------------------------------------------------------*
* ISSUE THE DISPLAY COMMAND, PROCESS THE RESPONSES                  *
* SEE LABEL CMDLIST FOR THE LIST OF OBJECTS DISPLAYED               *
*-------------------------------------------------------------------*
         BAKR  R14,0
         BAS   R14,SEND_REQ
         SELECT
         IF    LTR,R15,R15,NZ
            PR
         ENDIF
         BAS   R14,READ_RESPONSE
         IF    LTR,R15,R15,Z
            BAS   R14,PRINT_REPLY
            BAS   R14,PROCESS_CARD1         .1ST RSP. SHOWS COUNT
            IF    ICM,R5,15,RESPONSE_COUNT,NZ
               BAS   R14,PROC_RESPONSES
            ENDIF
            XR    R15,R15
         ENDIF
         PR
         EJECT
PROC_RESPONSES DS  OH
         BAKR  R14,0
```

```
         DO    FROM=(R5)
            BAS   R14,READ_RESPONSE
         DOEXIT LTR,R15,R15,NZ
            BAS   R14,PRINT_REPLY
            IF    CLC,=C'CSQ9',NE,BUFFER,AND,                          +
               CLC,=C'CSQMDRTS',NE,BUFFER+15
               MVC   BUFFER(8),=CL8'DEFINE'
               BAS   R14,GEN_PARMLIB
               XR    R15,R15
            ENDIF
         ENDDO
         PR
         EJECT
PRINT_REPLY    DS  0H
*-------------------------------------------------------------------*
* FOR DEBUGGING: PRINT THE RESPONSE FROM MQ **AS IS** TO DDNAME      *
* SYSPRINT                                                          *
*-------------------------------------------------------------------*
         BAKR  R14,0
         L     R1,REPLY_LENGTH
         LA    R1,4(,R1)
         STH   R1,BUFFER_LENGTH
         PUT   SYSPRINT,BUFFER_LENGTH
         XR    R15,R15
         PR
         EJECT
PROCESS_CARD1  DS  0H
         BAKR  R14,0
         LA    R10,BUFFER
         XC    RESPONSE_COUNT,RESPONSE_COUNT
         IF    CLC,=C'CSQN205I',EQ,BUFFER
            PACK  DUB,BUFFER+17(8)
            CVB   R1,DUB
            BCTR  R1,0
            ST    R1,RESPONSE_COUNT
            PACK  DUB,BUFFER+34(8)
            CVB   R1,DUB
            ST    R1,MAXRC
         ENDIF
         PR
         EJECT
READ_RESPONSE  DS  0H
         BAKR  R14,0
*-------------------------------------------------------------------*
* CLEAR THE RESPONSE BUFFER                                         *
*-------------------------------------------------------------------*
         LA    R0,BUFFER
         L     R1,=A(32*1024)
         XR    R15,R15
         MVCL  R0,R14
```

```
*--------------------------------------------------------------------*
* LET THE SYSTEM DEFAULT IN CODEDCHARSETID, FORMAT, MSGID, CORRELID   *
* AND PRIORITY                                                       *
*--------------------------------------------------------------------*
         XC    REPLY_MD_CODEDCHARSETID,REPLY_MD_CODEDCHARSETID
         MVC   REPLY_MD_FORMAT,=CL133' '
         XC    REPLY_MD_MSGID,REPLY_MD_MSGID
         XC    REPLY_MD_CORRELID,REPLY_MD_CORRELID
         MVC   REPLY_MD_PRIORITY,=F'-1'
         MVC   REPLY_MD_PERSISTENCE,=F'2'
         CALL  MQGET,(MQ_HANDLE,REPLY_HANDLE,                          +
               REPLY_MD,MQGMO,=A(16000),BUFFER,REPLY_LENGTH,           +
               MQ_RETCODE,MQ_REASON),                                  +
               VL,MF=(E,CALL_PARMS)
         IF    ICM,R15,15,MQ_RETCODE,NZ
            CVD   R15,DUB
            OI    DUB+7,X'0F'
            UNPK  WTO8+28(3),DUB+6(2)
            L     R0,MQ_REASON
            CVD   R0,DUB
            OI    DUB+7,X'0F'
            UNPK  WTO8+40(5),DUB+5(3)
            WTO   MF=(E,WTO8)
            LA    R15,8
         ENDIF
         PR
         EJECT
SEND_REQ       DS  0H
         BAKR  R14,0
*--------------------------------------------------------------------*
* COPY THE GENERATED QUEUE NAME TO THE REQUEST-REPLY-TO QUEUE NAME    *
*--------------------------------------------------------------------*
         MVC   REQ_MD_REPLYTOQ,REPLYQ_OBJECTNAME
         CALL  MQPUT,(MQ_HANDLE,REQUEST_HANDLE,REQ_MD,                 +
               MQPMO,=A(133),CMDLINE,                                  +
               MQ_RETCODE,MQ_REASON),VL,MF=(E,CALL_PARMS)
         IF    ICM,R15,15,MQ_RETCODE,NZ
            CVD   R15,DUB
            OI    DUB+7,X'0F'
            UNPK  WTO7+35(3),DUB+6(2)
            L     R0,MQ_REASON
            CVD   R0,DUB
            OI    DUB+7,X'0F'
            UNPK  WTO7+47(5),DUB+5(3)
            WTO   MF=(E,WTO7)
            LA    R15,8
         ENDIF
         PR
         EJECT
GEN_PARMLIB    DS  0H
```

```
         BAKR  R14,0
         MVC   PRTLINE,=CL80'DEFINE '
         LA    R9,BUFFER_LENGTH
         AH    R9,BUFFER_LENGTH
         LA    R10,BUFFER+7
         DO    WHILE=(CR,R10,LT,R9)
            BAS   R14,GET_NEXT_PARM
         DOEXIT LTR,R10,R1,Z
            BAS   R14,FIND_PARM_VALUE
         DOEXIT LTR,R8,R1,Z
            BAS   R14,GEN_NXT_LINE
            LR    R10,R1
            IF    CLI,NXTLINE,NE,X'40'
               MVI   PRTLINE+70,C'+'
               PUT   MQDEFS,PRTLINE
               IF    CLC,=C'DESCR(',NE,NXTLINE
                  MVC   PRTLINE,NXTLINE
               ELSE
                  MVC   PRTLINE,=CL80'DESCR('''
                  MVC   PRTLINE+7(73),NXTLINE+6
                  TRT   PRTLINE,TRT_CLOSE
                  MVC   0(2,R1),=C''')'
               ENDIF
               MVC   NXTLINE,=CL133' '
            ENDIF
         ENDDO
         PUT   MQDEFS,PRTLINE
         MVC   PRTLINE,=CL133' '
         PUT   MQDEFS,PRTLINE
         PR
FIND_NONBLANK  TRT 0(0,R10),TRT_NONBLANK
         EJECT
GET_NEXT_PARM  DS  0H
         BAKR  R14,0
         XR    R1,R1
         DO    WHILE=(CR,R10,LT,R9)
            LR    R3,R9
            SR    R3,R10
            IF    C,R3,GT,=F'255'
               LA    R3,255
            ENDIF
         DOEXIT EX,R3,FIND_NONBLANK,NZ
            LA    R10,1(R3,R10)
         ENDDO
         PR
         EJECT
GEN_NXT_LINE   DS  0H
         BAKR  R14,0
         MVC   NXTLINE,=CL133' '
         IF    CLC,=C'QUEUE',EQ,0(R10)
```

```
            MVC   NXTLINE(L'NXTLINE-4),CMDLINE+4
            TRT   NXTLINE(L'NXTLINE-4),TRT_OPEN
            LA    R6,1(,R1)
            LA    R10,1(,R8)
            BAS   R14,COPY_PARMVAL
         ELSE
            LR    R15,R8
            SR    R15,R10
            EX    R15,COPY_PARMNAME
            LA    R10,1(,R8)
            IF    CLI,0(R8),EQ,C'('
               LA    R6,NXTLINE+1(R15)
               BAS   R14,COPY_PARMVAL
               IF    CLI,0(R6),EQ,C')',                          +
               OR,CLC,=C'TYPE(Q',EQ,NXTLINE,                     +
               OR,CLC,=C'CRDATE',EQ,NXTLINE,                     +
               OR,CLC,=C'CRTIME',EQ,NXTLINE
                  MVC   NXTLINE,=CL133' '
               ENDIF
            ENDIF
         ENDIF
         PR
COPY_PARMNAME  MVC NXTLINE(0),0(R10)
         EJECT
COPY_PARMVAL   DS  0H
         BAKR  R14,0
         STRTSRCH WHILE=(CR,R10,LT,R9)
            LR    R15,R9
            SR    R15,R10
            LR    R14,R10
            L     R1,=A(X'40000000')
            CLCL  R14,R0
         EXITIF CLI,0(R14),EQ,C')'
            MVI   0(R6),C')'
            LA    R1,1(,R14)
         ORELSE
            LR    R3,R9
            SR    R3,R14
            IF    C,R3,GT,=F'255'
               LA    R3,255
            ENDIF
            EX    R3,FIND_DELIMITER
            LR    R3,R1
            SR    R3,R10
            EX    R3,COPY_DATA
            LA    R6,1(R3,R6)
         EXITIF CLI,0(R1),EQ,C')'
            LA    R1,1(,R1)
         ORELSE
            LA    R10,1(,R1)
```

```
         ENDLOOP
            LR    R1,R9
         ENDSRCH
         PR
FIND_DELIMITER TRT 0(0,R14),TRT_DELIMITER
COPY_DATA      MVC 0(0,R6),0(R10)
FIND_PARM_VALUE    DS  0H
         BAKR  R14,0
         XR    R1,R1
         DO    WHILE=(CR,R10,LT,R9)
            LR    R3,R9
            SR    R3,R10
         DOEXIT EX,R3,FIND_OPEN2,NZ
            LA    R10,1(R3,R10)
         ENDDO
         PR
FIND_OPEN2     TRT 0(0,R10),TRT_DELIMITER2
         EJECT
         EJECT
CLEANUP  DS    0H
         BAKR  R14,0
         SELECT EVERY
         WHEN  TM,PRT.DCBOFLGS,DCBOFOPN,0
            CLOSE SYSPRINT,MF=(E,CLOSE),MODE=31
         WHEN  TM,DEFS.DCBOFLGS,DCBOFOPN,0
            CLOSE MQDEFS,MF=(E,CLOSE),MODE=31
         WHEN  OC,REQUEST_HANDLE,REQUEST_HANDLE,NZ
            BAS   R14,CLOSE_REQUEST
         WHEN  OC,REPLY_HANDLE,REPLY_HANDLE,NZ
            BAS   R14,CLOSE_REPLY
         WHEN  OC,MQ_HANDLE,MQ_HANDLE,NZ
            BAS   R14,DISCONNECT
         ENDSEL
         PR
         EJECT
CLOSE_REQUEST  DS  0H
         BAKR  R14,0
         CALL  MQCLOSE,(MQ_HANDLE,REQUEST_HANDLE,                    +
               =A(MQCO_NONE),MQ_RETCODE,MQ_REASON),                  +
               VL,MF=(E,CALL_PARMS)
         IF    ICM,R15,15,MQ_RETCODE,NZ
            CVD   R15,DUB
            OI    DUB+7,X'0F'
            UNPK  WTO5+45(3),DUB+6(2)
            L     R0,MQ_REASON
            CVD   R0,DUB
            OI    DUB+7,X'0F'
            UNPK  WTO5+57(5),DUB+5(3)
            WTO   MF=(E,WTO5)
         ENDIF
```

```
         PR
         EJECT
CLOSE_REPLY    DS  0H
         BAKR  R14,0
         CALL  MQCLOSE,(MQ_HANDLE,REPLY_HANDLE,                          +
               =A(MQCO_DELETE_PURGE),MQ_RETCODE,MQ_REASON),              +
               VL,MF=(E,CALL_PARMS)
         IF    ICM,R15,15,MQ_RETCODE,NZ
            CVD   R15,DUB
            OI    DUB+7,X'0F'
            UNPK  WTO6+43(3),DUB+6(2)
            L     R0,MQ_REASON
            CVD   R0,DUB
            OI    DUB+7,X'0F'
            UNPK  WTO6+55(5),DUB+5(3)
            WTO   MF=(E,WTO6)
         ENDIF
         PR
         EJECT
DISCONNECT     DS  0H
         BAKR  R14,0
         CALL  MQDISC,(MQ_HANDLE,MQ_RETCODE,MQ_REASON),VL,               +
               MF=(E,CALL_PARMS)
         IF    ICM,R15,15,MQ_RETCODE,NZ
            CVD   R15,DUB
            OI    DUB+7,X'0F'
            UNPK  WTO2+36(3),DUB+6(2)
            L     R0,MQ_REASON
            CVD   R0,DUB
            OI    DUB+7,X'0F'
            UNPK  WTO2+48(5),DUB+5(3)
            WTO   MF=(E,WTO2)
         ENDIF
         PR
         EJECT
         LTORG
HEX      DC    C'0123456789ABCDEF'
TRT_OPEN DC    XL256'00'
         ORG   TRT_OPEN+C'('
         DC    X'FF'
         ORG
TRT_CLOSE DC   XL256'00'
         ORG   TRT_CLOSE+C')'
         DC    X'FF'
         ORG
TRT_NONBLANK   DC  X'00',255X'FF'
         ORG   TRT_NONBLANK+X'40'
         DC    X'00'
         ORG
TRT_DELIMITER  DC  256X'00'
```

```
         ORG   TRT_DELIMITER+X'40'
         DC    X'FF'
         ORG   TRT_DELIMITER+C')'
         DC    X'FF'
         ORG
TRT_DELIMITER2 DC  256X'00'
         ORG   TRT_DELIMITER2+X'40'
         DC    X'FF'
         ORG   TRT_DELIMITER2+C'('
         DC    X'FF'
         ORG
CMDLIST  DS    0F
         DC    CL(L'CMDLINE)'DIS STGCLASS(*) ALL'
         DC    CL(L'CMDLINE)'DIS CHANNEL(*) ALL'
         DC    CL(L'CMDLINE)'DIS QLOCAL(*) ALL'
         DC    CL(L'CMDLINE)'DIS QREMOTE(*) ALL'
         DC    CL(L'CMDLINE)'DIS QALIAS(*) ALL'
         DC    CL(L'CMDLINE)'DIS QMODEL(*) ALL'
         DC    CL(L'CMDLINE)'DIS PROCESS(*) ALL'
         DC    X'00'
         EJECT
CODE_SIZE      EQU *-PMQGETB0
PAD_SIZE EQU   4096-(CODE_SIZE-(CODE_SIZE/4096*4096))
PAD      DC    (PAD_SIZE)C'P'
WRKSTOR  DS    0D
MYSAVE   DC    18A(X'FEFEFEFE')
DUB      DS    D
OPEN     OPEN  0,MODE=31,MF=L
CLOSE    CLOSE 0,MODE=31,MF=L
MQ_HANDLE      DS  F
MQ_RETCODE     DS  F
MQ_REASON      DS  F
REQUEST_HANDLE DS  F
REPLY_HANDLE   DS  F
REPLY_LENGTH   DS  F
RESPONSE_COUNT DS  F
CMD_BUF_SIZE   DS  F
CMD_BUF_ADDR   DS  A
MAXRC          DS  F
CALL_PARMS     DS  20F
MQ_NAME  DC    CL96' '
SYSPRINT DCB   DDNAME=SYSPRINT,DSORG=PS,MACRF=PM,                      +
               LRECL=27994,RECFM=VB,BLKSIZE=27998
MQDEFS   DCB   DDNAME=MQDEFS,DSORG=PS,MACRF=PM,                        +
               LRECL=80,RECFM=FB,BLKSIZE=27920
*              LRECL=27994,RECFM=VB,BLKSIZE=27998
PRTLINE  DC    CL80' '
NXTLINE  DC    CL80' '
CMDLINE  DS    CL133
WTO_HANDLE     WTO 'CONNECT: HANDLE = X''00000000''',MF=L
```

```
WTO_HANDLE2    WTO 'DISCONNECT: HANDLE = X''00000000''',MF=L
WTO1     WTO   'ERROR: MQ_CONNECT FAILED, RC=XXX, REASON=XXXXX',MF=L
WTO2     WTO   'ERROR: MQ_DISCONNECT FAILED, RC=XXX, REASON=XXXXX',MF=L
WTO3     WTO   'ERROR: OPEN OF REPLY QUEUE FAILED, RC=XXX, REASON=XXXXX+
               ',MF=L
WTO4     WTO   'ERROR: OPEN OF REQUEST QUEUE FAILED, RC=XXX, REASON=XXX+
               XX',MF=L
WTO5     WTO   'ERROR: CLOSE OF REQUEST QUEUE FAILED, RC=XXX, REASON=XX+
               XXX',MF=L
WTO6     WTO   'ERROR: CLOSE OF REPLY QUEUE FAILED, RC=XXX, REASON=XXXX+
               X',MF=L
WTO7     WTO   'ERROR: SEND REQUEST FAILED, RC=XXX, REASON=XXXXX',MF=L
WTO8     WTO   'ERROR: MQGET FAILED, RC=XXX, REASON=XXXXX',MF=L
         PRINT GEN,NOPRINT
MQ_OBJECT_Q    DS  0D
REPLYQ   CMQODA    DSECT=NO,LIST=YES,                                  +
               OBJECTTYPE=MQOT_Q,                                      +
               OBJECTNAME=SYSTEM.COMMAND.REPLY.MODEL,                  +
               DYNAMICQNAME=SYSTEM.PMQGETB0.*
REQUESTQ CMQODA DSECT=NO,LIST=YES,                                     +
               MSGTYPE=MQMT_REQUEST,OBJECTTYPE=MQOT_Q,                 +
               OBJECTNAME=SYSTEM.COMMAND.INPUT
REQ_MD   CMQMDA DSECT=NO,LIST=YES,MSGTYPE=MQMT_REQUEST,                +
               REPLYTOQ=
REPLY_MD CMQMDA DSECT=NO,LIST=YES,MSGTYPE=MQMT_REPLY
         CMQPMOA DSECT=NO,LIST=YES,                                    +
               OPTIONS=MQPMO_NO_SYNCPOINT
         CMQGMOA DSECT=NO,LIST=YES,                                    +
               OPTIONS=MQGMO_WAIT+MQGMO_NO_SYNCPOINT,                  +
               WAITINTERVAL=30000
TRTAB    DS    XL256
BUFFER_RDW     DS  0D
BUFFER_LENGTH  DS  H
RDW_ZERO       DC  H'0'
BUFFER   DS    16CL1024
CODE_SIZE1     EQU *-WRKSTOR
PAD_SIZE1      EQU   4096-(CODE_SIZE1-(CODE_SIZE1/8*8))
PAD1     DC    (PAD_SIZE1)C'P'
WSLEN    EQU   *-WRKSTOR
         REGEQU
         DCBD  DSORG=PS,DEVD=DA
         CMQA
         END
//PMQGETJ0 EXEC PGM=PMQGETB0,PARM=CSQ2
//STEPLIB  DD  DSN=P99999.SDCD0.LOAD,DISP=SHR
//         DD  DSN=SYS1.@MQ12000.SCSQLOAD,DISP=SHR
//         DD  DSN=SYS1.@MQ12000.SCSQAUTH,DISP=SHR
//SYSTRACE DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//MQDEFS   DD  SYSOUT=*
```

```
//*MQDEFS   DD  DSN=P99999.MQDEFS(CSQ4),DISP=SHR
//SYSUDUMP DD  SYSOUT=*
```

PPFC14M0 – COPYBOOK WITH MACROS USED IN PROGRAM

```
         MACRO
         GETCC &COND
         GBLA  &PF_CCVAL
         LCLC  &LWK1
         AIF   ('&COND'(1,1) LT '0' OR '&COND'(1,1) GT '9').NOTNUM
&PF_CCVAL      SETA  &COND
         MEXIT
.NOTNUM  AIF   (K'&COND NE 1).TWOCHAR
&LWK1    SETC  '&COND'
         AGO   .CALCC
.TWOCHAR AIF   (K'&COND NE 2).INVCOND
         AIF   ('&COND'(1,1) NE 'N').OTHERMN
&LWK1    SETC  '&COND'(2,1)
         AGO   .CALCC
.OTHERMN AIF   ('&COND' EQ 'EQ').BC8
         AIF   ('&COND' EQ 'LT').BC4
         AIF   ('&COND' NE 'LE').TRYGT
&PF_CCVAL      SETA  13
         MEXIT
.TRYGT   AIF   ('&COND' EQ 'GT').BC2
         AIF   ('&COND' NE 'GE').INVCOND
&PF_CCVAL SETA 11
         MEXIT
.CALCC   AIF   ('&LWK1' NE '0').TRYH
&PF_CCVAL      SETA  1
         AGO   .TSTN
.TRYH    AIF   ('&LWK1' EQ 'P' OR '&LWK1' EQ 'H').BC2
         AIF   ('&LWK1' EQ 'L' OR '&LWK1' EQ 'M').BC4
         AIF   ('&LWK1' EQ 'E' OR '&LWK1' EQ 'Z').BC8
         AGO   .INVCOND
.BC8     ANOP
&PF_CCVAL      SETA  8
         AGO   .TSTN
.BC4     ANOP
&PF_CCVAL      SETA  4
         AGO   .TSTN
.BC2     ANOP
&PF_CCVAL      SETA  2
.TSTN    AIF   ('&COND'(1,1) NE 'N').DONE
&PF_CCVAL      SETA  15-&PF_CCVAL
.DONE    MEXIT
.INVCOND ANOP
&PF_CCVAL      SETA  15
         MNOTE 8,'INVALID CONDITION MNEMONIC. NOP GENERATED'   @BA25155
```

```
         MEND
*************************************************************************
         MACRO
         POPINS &P
         COPY  PPFGBLC0
         LCLA   &W
&W       SETA   &P
         AGO    .TEST
.UNSTACK ANOP
         AIF   ('&PF_IIND3(&W)' EQ '').ONEOP
         AIF   ('&PF_IIND4(&W)' NE '').THREEOP
&PF_IIND5(&W)  &PF_IIND1(&W) &PF_IIND2(&W),&PF_IIND3(&W)
         AGO   .INCTR
.THREEOP ANOP
&PF_IIND5(&W)  &PF_IIND1(&W) &PF_IIND2(&W),&PF_IIND3(&W),&PF_IIND4(&W)
         AGO   .INCTR
.ONEOP   ANOP
&PF_IIND5(&W)  &PF_IIND1(&W) &PF_IIND2(&W)
.INCTR   ANOP
&W       SETA  &W+1
.TEST    AIF    (&W LE &PF_II).UNSTACK
&PF_II   SETA  &P-1
         AIF   ('&PF_NEST(&PF_NI)'(3,1) NE ' ' OR '&PF_NEST(&PF_NI)'(4,+
               1) EQ ' ').NEQ
&PF_IIND5(&PF_II) &PF_IIND1(&PF_II) &PF_IIND2(&PF_II)
.NEQ     AIF   (&PF_II GT 0 OR (&PF_II EQ 0 AND '&PF_NEST(&PF_NI)'(5,4)+
                EQ 'IF')).END
         MNOTE 8,'NEGATIVE INSTRUCTION STACK PTR. EXPANSION INVALID.'
.END     MEND
*************************************************************************
         MACRO
         POPNEST &P1
         COPY  PPFGBLC0
         LCLC  &SUFFIX
&SUFFIX  SETC  '&PF_NEST(&PF_NI)'(5,4)
         AIF   ('&PF_NEST(&PF_NI)'(5,4) EQ '&P1').GOOD
         MNOTE 8,'&SUFFIX MACRO AT SAME LEVEL AS &P1 TERMINATOR.'
.GOOD    ANOP
&PF_NI   SETA  &PF_NI-1
         AIF   (&PF_NI GE 0).OK
         MNOTE 8,'NEGATIVE NEST STACK POINTER. CHECK NUMBER OF ENDS.'
.OK      MEND
*************************************************************************
         MACRO
         STKINS &P1,&P2,&P3,&P4,&P5,&P6
         COPY  PPFGBLC0
         AIF    ('&P1(2)' EQ '').NOTSUBL
         AIF    ('&P1(6)' EQ '' OR '&P1(6)' EQ '&PF_LIND(&PF_LI)').OKSU+
               BL
         MNOTE 12,'TOO MANY OPERANDS INSIDE PARENTHESES'
```

```
         MEXIT
.OKSUBL  PUSHINS     (&P1(1),&P1(2),&P1(3),&P1(4),&P1(5),&P1(6))
         MEXIT
.NOTSUBL AIF   ('&P2' EQ '' OR '&P2' EQ 'OR' OR '&P2' EQ 'AND' OR '&P2'+
               EQ 'ORIF' OR '&P2' EQ 'ANDIF').SGLOPR
         AIF   ('&P5' EQ 'OR' OR '&P5' EQ 'AND' OR '&P5' EQ 'ORIF' OR  +
               '&P5' EQ 'ANDIF').TWOPER2
         PUSHINS     (&P1,&P2,&P3,&P4,&P5,&P6)
&PF_CTR  SETA  &PF_CTR+4
         MEXIT
.TWOPER2 PUSHINS     (&P1,&P2,&P3,&P4,,&P6)
&PF_CTR  SETA  &PF_CTR+3
         MEXIT
.SGLOPR  GETCC  &P1(1)
         MEND
*********************************************************************
         MACRO
         PUSHINS     &PAM
         COPY PPFGBLCO
         LCLA &WK,&I,&J,&K
&I       SETA  3
&J       SETA  4
&K       SETA  4
         AIF   ('&PAM(1)'(1,1) EQ 'B' OR '&PAM(1)' EQ 'EQU').BCH
         AIF   ('&PAM(5)' EQ '').TWOPERS
         AIF   ('&PAM(1)' EQ 'CS').CNSWAP
         AIF   ('&PAM(1)' EQ 'CDS').CNSWAP
         AIF   ('&PAM(1)'(1,1) EQ 'C').SETK
.CNSWAP  ANOP
&J       SETA  5
         AGO   .GETCOND
.TWOPERS AIF   ('&PAM(1)'(1,1) NE 'C').TSTIAC
         AIF   ('&PAM(1)' EQ 'CLCL').CLCL
&I       SETA  4
&J       SETA  3
         AGO   .SETK
.CLCL    ANOP
&I       SETA  3
&J       SETA  4
&K       SETA  3
         AGO   .SETK
.TSTIAC  ANOP
         AIF   ('&PAM(1)' NE 'IAC').SETK
&I       SETA  4
&J       SETA  3
&K       SETA  4
         AGO   .GETCOND
.SETK    ANOP
&K       SETA  5
.GETCOND GETCC &PAM(&J)
```

```
.BCH     AIF   (&PF_II GE 100).OVERI
&PF_II   SETA  &PF_II+1
&PF_IIND1(&PF_II)    SETC  '&PAM(1)'
&PF_IIND2(&PF_II)    SETC  '&PAM(2)'
         AIF   ('&PAM(&I)' NE '').LD31
&PF_IIND3(&PF_II)    SETC  ''
         AGO   .PAM4
.LD31    ANOP
&PF_IIND3(&PF_II)   SETC  '&PAM(&I)'
.PAM4    ANOP
         AIF   ('&PAM(&K)' NE '').LD41
&PF_IIND4(&PF_II)   SETC  ''
         AGO   .PAM5
.LD41    ANOP
&PF_IIND4(&PF_II)   SETC  '&PAM(&K)'
.PAM5    AIF   ('&PAM(6)' EQ '').BLKOUT5
         AIF   ('&PAM(6)'(1,10) NE 'PF_C14LBL_').BLKOUT5
&PF_IIND5(&PF_II)   SETC  '&PAM(6)'
         MEXIT
.BLKOUT5 ANOP
&PF_IIND5(&PF_II)   SETC  ''
         MEXIT
.OVERI   MNOTE 8,'INSTRN STK SIZE EXCEEDED. FURTHER EXPANSIONS INVALID'
         MEND
*************************************************************************
         MACRO
         PUSHNEST &P1
         COPY  PPFGBLC0
&PF_NI   SETA  &PF_NI+1
         AIF   (&PF_NI GE 50).OVER
&PF_NEST(&PF_NI)     SETC  '    '.'&P1'
         MEXIT
.OVER    MNOTE 8,'NEST STACK SIZE EXCEEDED. FURTHER EXPANSIONS INVALID'
         MEND
*************************************************************************
         MACRO
         PUSHLAB
         COPY  PPFGBLC0
         AIF   (&PF_LI GE 100).OVER
&PF_SEQ  SETA  &PF_SEQ+1
&PF_LI   SETA  &PF_LI+1
&PF_LIND(&PF_LI)     SETC  'PF_C14LBL_&PF_SEQ'
         MEXIT
.OVER    MNOTE 8,' LABEL STK SIZE EXCEEDED. FURTHER EXPANSIONS INVALID'
         MEND
*************************************************************************
         MACRO
         IFPROC
         COPY  PPFGBLC0
         LCLB  &ANDIND,&ORIND
```

```
         PUSHLAB
&PF_CTR  SETA  2
&PF_ST(&PF_NI+1)     SETA  &PF_II+1
&PF_NEST(&PF_NI)     SETC  '  R'.'&PF_NEST(&PF_NI)'(4,5)
         AIF   (T'&SYSLIST(1) EQ 'O').LOOP
         AIF   (&SYSLIST(1) LE 0 OR &SYSLIST(1) GE 15).INVALCC
&PF_CCVAL      SETA  &SYSLIST(1)
         AIF   ('&SYSLIST(2)' EQ '').ENDBOOL
         MNOTE 4,'CC KEYWORD USED. OTHER PARAMETERS IGNORED'
         AGO   .ENDBOOL
.INVALCC MNOTE 4,'CC OUTSIDE VALID RANGE OF 1 TO 14. NOP GENERATED'
&PF_CCVAL SETA 15
         AGO   .ENDBOOL
.LOOP    STKINS    &SYSLIST(&PF_CTR),                                  +
               &SYSLIST(&PF_CTR+1),                                    +
               &SYSLIST(&PF_CTR+2),                                    +
               &SYSLIST(&PF_CTR+3),                                    +
               &SYSLIST(&PF_CTR+4)
         AIF   ('&SYSLIST(&PF_CTR+1)' EQ 'AND').ANDPROC
         AIF   ('&SYSLIST(&PF_CTR+1)' NE 'ANDIF').TESTOR
.ANDPROC PUSHINS   (BC,15-&PF_CCVAL,&PF_LIND(&PF_LI-1))
&ANDIND  SETB 1
         AIF   ('&SYSLIST(&PF_CTR+1)' NE 'ANDIF' OR NOT &ORIND).TESTLP
         POPINS    &PF_ST(&PF_NI+1)
&PF_LIND(&PF_LI)   EQU   *
&ORIND   SETB  0
&PF_LI   SETA  &PF_LI-1
         PUSHLAB
         AGO   .TESTLP
.TESTOR  AIF   ('&SYSLIST(&PF_CTR+1)' EQ 'OR').ORPROC
         AIF   ('&SYSLIST(&PF_CTR+1)' NE 'ORIF').TESTLP
.ORPROC  PUSHINS   (BC,&PF_CCVAL,&PF_LIND(&PF_LI))
&ORIND   SETB  1
         AIF   ('&SYSLIST(&PF_CTR+1)' NE 'ORIF' OR NOT &ANDIND).TESTLP
         PUSHINS (EQU,*,,,,&PF_LIND(&PF_LI-1))
&ANDIND  SETB 0
         PUSHLAB
&PF_LI   SETA &PF_LI-1
&PF_LIND(&PF_LI-1) SETC  '&PF_LIND(&PF_LI+1)'
.TESTLP  ANOP
&PF_CTR  SETA  &PF_CTR+2
         AIF   ('&SYSLIST(&PF_CTR-1)' NE '').LOOP
```

This article concludes in next month's issue of *MQ Update*.

Pieter Wiid
Senior Systems Engineer
Outsource (South Africa)

MQ news

Sterling Commerce has launched CONNECT:MQ, a file transfer product that uses MQSeries as its messaging backbone. CONNECT:MQ is based on Message Quest's File Transfer Facility for MQSeries (FTF/MQ), and its key features include: using multiple channels for increased bandwidth and load balancing, modular design to reduce memory overhead, and parallel file transfers.

Available now on MVS/ESA, NT 4, OS/400, OS Warp 3, Solaris, HP-UX, and AIX. No details were received on pricing.

For further information contact:
Sterling Commerce, 300 Crescent Court, Suite 1200, Dallas, TX 75201, USA
Tel: +1 214 981 1000
Fax: +1 214 981 1255
Web: http://www.sterlingcommerce.com

Sterling Software, Commerce Services Group, 1 Longwalk Road, Stockley Park, Uxbridge, Middlesex UB11 1DB, UK
Tel: +44 181 867 8020
Fax: +44 181 867 8008

* * *

Candle has announced additions and upgrades to its Roma family of application integration products that allow middleware products, including MQSeries and MQSeries Integrator, to act as the communication medium between applications, taking on responsibility for routing, delivery, and message transformation.

Roma gets a number of new components, including an IDE, system management tools, and pre-packaged support for applications such as SAP R/3. The most interesting one from a middleware point of view, though, is the Roma Broker, which is able to use MQSeries and MQSeries Integrator for handling communication between applications.

Out now, prices weren't announced.

For further details contact:
Candle Corp, 2425 Olympic Blvd, Santa Monica, CA 90404, USA
Tel: +1 310 829 5800
Fax: +1 310 582 4287
Web: http://www.candle.com

Candle Ltd, 1 Archipelago, Lyon Way, Frimley, Camberley, Surrey GU16 5ER, UK
Tel: +44 1276 4147000
Fax: +44 1276 414777

* * *

IBM has released Net.Commerce Version 3.0 for AS/400. Among the new bits is an adapter that allows the product to use MQSeries for back-end integration. The product has also been redesigned to improve security, scalability, and extensibility. Out now, prices range from US$7,500 to US$60,000. Upgrades start from US$3,750.

* * *

CORRECTION

We incorrectly stated in last month's *MQ news* that the price of MQSoftware's QPasa! starts at US$20,000 for 15 queue managers. This is not the case, and the price of QPasa! is available by request from MQSoftware.