
MSM-XCALL User's Guide

Version 3.0

March 1998

Micronetics Design Corporation

Disclaimer

Micronetics Design Corporation makes no representations or warranties with respect to this manual. Further, Micronetics Design Corporation reserves the right to make changes in the specification of the product described within this manual and without obligation of Micronetics Design Corporation to notify any person of such revision or changes.

The software described in this document is furnished under a license by Micronetics Design Corporation and may only be used or copied in accordance with the terms of such license.

This document is copyrighted. Micronetics Design Corporation grants you the right to download and copy this document only in conjunction with a validly licensed copy of the software and subject to all provisions of the software license agreement.

Copyright © 1998
Micronetics Design Corporation
1375 Piccard Drive
Rockville, Maryland 20850

Telephone: 301-258-2605
Fax: 301-840-8943
E-Mail: info@micronetics.com
WWW: www.micronetics.com

Contents

Preface	v
Acknowledgment.....	v
Documentation Conventions	v
Getting Started	1
Overview	1
MSM-XCALL Development Package	1
XCALL Concepts.....	3
XCALL Syntax.....	4
XCALL Errors	4
Defining XCALLs	5
Overview	5
Invoke XCALLMGR	6
Define an XCALL.....	6
Check the XCALL Definition	15
Generate the Source File	15
Compile the XCALL Package.....	16
General Issues	16
Compilation Under MS-DOS.....	16
Compilation Under Windows	17
Compilation Under AIX.....	18
Compilation Under Digital UNIX.....	18
Compilation Under Solaris.....	18
Managing MSM-XCALL	19
Overview	19
Load an XCALL Package	19
Using SYSGEN to Load a Package	19
Using XCALL to Load a Package.....	20
Define Default Packages	21
Using SYSGEN to Define the Default Packages List	21
Using XCDEFPK to Edit the Default Packages List.....	22
Display XCALL Information	23
XCALL Portability.....	24
Exporting an XCALL Package	24
Importing an XCALL Package	25

Parameter Options	27
Overview	27
Parameter Mode	27
Parameter Type	27
Number Size.....	28
String Type.....	28
Parameter Mechanism	29
Parameter Requirement	29
Return Interpretation	30
Sample C Language Program	31
Overview	31
Index	33

Preface

Acknowledgment

Micronetics Standard M (MSM) is an implementation, with extensions, of the ANSI Standard Specification (X11.1-1995) for the Massachusetts General Hospital Utility Multi Programming System (MUMPS). MUMPS was developed by the Laboratory of Computer Science at Massachusetts General Hospital under grant number HS00240 from the National Center for Health Services Research and Development. MUMPS was a trademark of the Massachusetts General Hospital.

Documentation Conventions

The following documentation conventions are used in this manual.

Convention	Description
RETURN	The carriage return key (normally labeled RETURN, ENTER, and so on).
CTRL+X	The CTRL key pressed at the same time as the X key, where X is any valid key used in combination with the control key.
<ERROR>	An MSM error message.
'val'	In Help messages, 'val' is used to indicate that the user can enter the indicated value. The value is entered without the quotes.
>	The MSM Programmer prompt.
...	The series of items repeats a user-specified number of times.
.	Shows a break in a list where consecutive lines have been omitted.
.	
.	
Bold	Items in a dialogue are shown in bold to indicate a user response.

Getting Started

Overview

This section provides basic information about MSM-XCALL, describes the development package for MSM-XCALL, and explains the syntax for external calls.

MSM-XCALL is a toolkit for defining the interface that allows external programs to be called from MSM. It also includes utilities for loading the external programs and making them available for use by M programs.

MSM-XCALL provides the flexibility to perform functions that are not easily executed with standard M code. It allows M programs to execute external programs (as either subroutines or functions) that are written in assembler language or the C programming language. References to such external software are called external functions, external procedures, or XCALLs.

MSM-XCALL Development Package

The MSM-XCALL development package consists of the XCALLMGR utility and a set of external files. Use the interactive utility XCALLMGR to completely define the interface between the M program and the external program. XCALLMGR is described in “Defining XCALLS” in this manual.

During installation of MSM, the external files are loaded into a directory named ‘xcall’ and are customized for use with your operating system. The package includes the following files:

- example.c
- example.pac
- xcdef.h
- README file
- Makefile (MSM-UNIX only)
- xcall.def (MSM for Windows only)
- MAKEXC.BAT (MSM-PC/PLUS only)
- XCALLPKG.MK (MSM-PC/PLUS only)
- XCALLPKG.WLK (MSM-PC/PLUS only)
- XCHEAD.ASM (MSM-PC/PLUS only)

These files are described below.

example.c

This file contains a sample C language program. The example dialog shown in “Defining XCALLs” in this manual defines an interface for accessing this program from MSM.

example.pac

This file contains a portable definition of the external calls in the sample XCALL package. To create such files, use the export capability of the utility program XCALLMGR.

xcdef.h

This file is needed while compiling the C language program generated by the utility program XCALLMGR. It contains definitions that the C compiler uses for building the XCALL interface tables. In this context, this file is known as a header file or include file.

README file

The README file, which may be included in the development package, contains installation instructions that supersede the information in this printed documentation.

Makefile

This file is provided with MSM-UNIX. It is a sample control file for the make tool which illustrates how to compile and link an XCALL package.

xcall.def

This file is provided with MSM-Server for Windows. It is a sample export definition file which exports the xfiles symbolic name.

MAKEXC.BAT

This file is provided with MSM-PC/PLUS. It is needed while compiling and linking programs in the C programming language that are compiled using the WATCOM C Compiler.

XCALLPKG.MK

This file is provided with MSM-PC/PLUS. It is needed while compiling and linking programs in the C programming language that are compiled using the WATCOM C Compiler.

XCALLPKG.WLK

This file is provided with MSM-PC/PLUS. It is needed while compiling and linking programs in the C programming language that are compiled using the WATCOM C Compiler.

XCHEAD.ASM

This file is provided with MSM-PC/PLUS. It is used to position the data segment at the beginning of the generated module. If you use a different compiler/linker, you must follow the appropriate procedures to ensure that the data segment is at the beginning of the module.

XCALL Concepts

In MSM-XCALL, a *package* is a collection of external routines which are linked together into an executable file. A file may contain one or more packages. A *routine* is the name of a function within a package. Before a routine can be called, the file containing its package must be loaded into memory.

Parameters may be passed from M to the XCALL routine. The parameters may be passed either by value or by reference. A parameter that is passed by value can only be an input parameter to the XCALL routine. A parameter that is passed by reference can be either an input or output parameter, or both.

The MSM system automatically performs conversions between the standard M data type and the parameter type that the XCALL routine requires. The parameter types are fully described in “Parameter Options” in this manual. If an operation other than type conversion is required for the parameters, an assistant routine may be used.

Assistant routines are entry points in a package which have a single parameter as input and which return a value. They may operate on input parameters to an XCALL routine or on output values that are being returned to MSM.

When an XCALL input parameter uses an assistant routine, the input value is passed to the assistant routine (after necessary type conversion), and the return value from the assistant routine becomes the actual input parameter for the XCALL function.

When an XCALL output parameter uses an assistant routine, the value returned by the XCALL function is passed as input to the assistant routine, and the return value from the assistant routine becomes the actual value assigned to the M output variable (after conversion to the M data type).

Packages that are loaded into memory can be added to the default packages list, a list of packages that will be searched, in order, for an XCALL routine that is called without a specific package name. MSM automatically contains one package of XCALLs, named MSM, which is always resident in memory and is the only entry in the initial default packages list. Refer to “Managing MSM-XCALL” in this document for assistance with adding to or modifying the default packages list.

XCALL Syntax

XCALL routines may be called as functions which return a value, or as subroutines.

If the XCALL routine is a function which returns a value, the routine is called with the following syntax:

```
SET VAR=$&[Package.]Routine(Parm1, Parm2, ...)
```

If the XCALL routine is called as a subroutine, the following syntax is used:

```
DO &[Package.]Routine(Parm1, Parm2, ...)
```

If the package name is omitted, MSM searches for the routine in the default packages list.

In either case, an M variable passed by reference may be modified by the XCALL routine. For example:

```
DO &INCREMENT(.VAR)
```

For compatibility with previous releases, MSM continues to support the ZCALL syntax. The following examples illustrate how routines are invoked using this syntax:

```
SET X=$ZCALL([Package.]Routine, Parm1, Parm2, ...)
```

```
ZCALL [Package.]Routine(Parm1, Parm2, ...)
```

XCALL Errors

If the MSM system detects an error during XCALL processing, the program receives an <XCALL> error. The \$ZERROR system variable will contain a major error number of 4 and a minor error number of 16. The additional information field of \$ZERROR will contain a number which further identifies the cause of the error. The following table provides a description of these error codes.

MSM-XCALL Error Codes

Error Code	Description
1001	The requested package is not loaded in memory
1002	The requested routine was not found in the specified package or in the default packages list.
1003	Unused.
1004	The requested routine entry point was found to be zero in the loaded package
1005	A required parameter is missing or too many parameters were specified.
1006-1027	Inconsistent values were found in the XCALL internal structures. These values could be caused by manual changes made to the C source code file created by XCALLMGR.
0	The return value of the XCALL routine is defined as "Non-Zero Status," but the return value was zero.
nn	The return value of the XCALL routine is defined as "Zero Status," but the return value was nn.

Defining XCALLs

Overview

This section describes how to define XCALLs.

The following list enumerates the steps required to create an XCALL. Each step is described in the following sections, with terminal sessions provided as appropriate. The first terminal session explains how to invoke the utility program XCALLMGR, which is used to define an XCALL. The following steps then are used to create the loadable XCALL package.

1. Define an XCALL
2. Check the XCALL definition
3. Generate the source file
4. Compile the XCALL package

Invoke XCALLMGR

To access the utility program XCALLMGR, you must be logged into the MGR UCI. The following terminal session demonstrates how to invoke the utility program. As with all terminal sessions in this document, your responses are displayed in bold.

```
>DO ^XCALLMGR

MSM - XCALL Maintenance

1 - Packages Definition
2 - Source Files Generation
3 - Check Package Definition
4 - Export Packages Definition
5 - Import Packages Definition

Select Option:
```

Define an XCALL

Before a package of external calls can be used with MSM, the complete interface between MSM and the external routines must be defined using the utility program XCALLMGR.

Begin by defining the package's general characteristics. The package may have an initialization routine that will be called when the package is loaded into memory. If special processing is needed to convert parameters for an XCALL routine, assistant routines may be defined.

Define an MSM XCALL

1. Define the name that the M programmer will use to refer to the XCALL routine.
2. Designate the entry point in the XCALL package that will be called to perform the XCALL processing.
3. If the XCALL routine is to be called as a function which returns a value, the type of the returned value must be described.
4. Specify the number of parameters that will be used in the M statement that calls the XCALL routine.
5. Describe the characteristics of each parameter, including its data type; whether an assistant routine should be used; whether it is an input parameter to the XCALL routine, or receives output from the XCALL routine or both; and how it is passed to the XCALL routine.
6. Define any constant parameters that are passed as input to the XCALL routine.

The following terminal session demonstrates use of the utility program XCALLMGR to define an XCALL. The function to be called, written in the C programming language, has the following prototype statement:

```
char *func1(short parm1, int parm2, int parm3, char *parm4);
```

The first three parameters are input. The function may modify the character string pointed to by *parm4*. *Parm3* contains the size of the buffer pointed to by *parm4*. The function returns a pointer to a character string.

The M statement that invokes this function has the following format:

```
SET STR=$&TEST.CHKTEMP(X1,X2,.X3)
```

The variables *X1* and *X2* contain the values for *parm1* and *parm2*, respectively. The variable *X3* receives the string that the function sets in *parm4*. The function requires that *parm3* contain the size of the buffer pointed to by *parm4*. For this example, we have arbitrarily decided that the maximum length of *X3* is 100 characters, so *parm3* will always have a constant value of 100. The variable *STR* receives the string whose pointer is returned by the function.

For this illustration, assume that the M routine and the XCALL routine use different scaling factors for the second parameter. Therefore, we must provide an assistant routine as part of the XCALL package. This routine takes one input parameter, which is the value from the variable *X2*. Its return value is properly scaled and is passed as the second parameter (*parm2*) to the function *func1*. In the sample code, the XCALL routine expects a temperature in degrees Fahrenheit, but the M routine passes degrees Celsius. The assistant routine converts the parameter from Celsius to Fahrenheit.

The package contains an entry point named *testinit*, which has no parameters and performs some initialization when it is first loaded into memory.

The C program which implements this function is provided as *example.c* in the MSM-XCALL Development Package. An exported version of the following definition is provided as *example.pac*.

Terminal session

```
Select Option: 1 - Packages Definition
```

```
Select Package: ?
```

```
Enter the name of the XCALL package to define.  
A package is a group of related XCALL routines.
```

```
Enter ^L to list existing package names.  
Enter -name to delete the definition of an existing package.  
Enter ^Q to exit from package definition.
```

```
Select Package: TEST
```

```
Defining new package: TEST
```

```
Package Initialization Routine: ?
```

```
Enter the name of a routine that should be called to initialize the  
package when it is loaded into memory. This must be an entry point  
within the package. It will be called only once, before any XCALLs  
in the package are called.
```

```
The name must begin with an alphabetic character or an  
underscore.  
Enter a minus sign (-) to remove an existing initialization routine  
name.
```

```
Package Initialization Routine: testinit
```

```
Assistant routine: ?
```

```
Enter the name of an entry point within the package that will be  
used as an assistant for converting parameters (either input or  
output) for the XCALLs in the package.  
The name must begin with an alphabetic character or an  
underscore.
```

```
Enter ^L to list existing assistant routines.  
Enter -name to delete the definition of an assistant routine.  
Enter ^ to return to the previous question.  
Enter ^Q to exit from package definition.
```

Assistant routine: **asst1**

Input Parameter Type:

- 1 - Integer
- 2 - Unsigned Integer
- 3 - Float
- 4 - String

Select Option: **1** - Integer

Integer Size:

- 1 - 1-Byte
- 2 - 2-Bytes
- 3 - 4-Bytes
- 4 - Size-Int

Select Option: **4** - Size-Int

Input Mechanism:

- 1 - Value
- 2 - Reference

Select Option: **1** - Value

Output Parameter Type:

- 1 - Integer
- 2 - Unsigned Integer
- 3 - Float
- 4 - String

Select Option: **1** - Integer

Integer Size:

- 1 - 1-Byte
- 2 - 2-Bytes
- 3 - 4-Bytes
- 4 - Size-Int

Select Option: **4** - Size-Int

Assistant routine: **RETURN**

XCALL name: ?

Enter the name of the XCALL function to be defined. This is the name that the M program will use to call the function. The name can contain alphanumeric characters and the percent character (%).

Enter ^L to list all XCALL names defined for this package.
Enter ^ to return to the previous question.
Enter ^Q to quit package definition.

XCALL name: **CHKTEMP**

Defining new XCALL: CHKTEMP

XCALL Link Name: ?

Enter the entry point name of the routine within this package that will be called to perform this XCALL function. This must be an external name that will be visible when the package is processed by the linker. The name must begin with an alphabetic character or an underscore.

XCALL Link Name: **func1**

Return Interpretation:

- 1 - Valid Value
- 2 - Ignored
- 3 - Zero Status
- 4 - Non-Zero Status
- 5 - OS Status

Select Option: ?1

The XCALL link routine returns a valid value, which should be returned as the value of the M XCALL function statement.

Select Option: ?2

The XCALL link routine does not return a valid value. If it is called as a function by the M XCALL statement, the returned value of the function is unpredictable.

Select Option: ?3

The XCALL link routine returns a value of zero when it is successful. If it returns a non-zero value, an MSM error will be generated.

Select Option: ?4

The XCALL link routine returns a non-zero value when it is successful. If it returns a value of zero, an MSM error will be generated.

Select Option: ?5

The return value of the XCALL link routine is interpreted according to the normal rule of the underlying operating system. If the return value indicates an error, an MSM error will be generated.

Select Option: 1 - Valid Value

Return Value Type:

- 1 - Integer
- 2 - Unsigned Integer
- 3 - Float
- 4 - String

Select Option: 4 - String

String Type:

- 1 - Opaque
- 2 - Zero-Terminated String
- 3 - 1-Byte Counted String
- 4 - 2-Bytes Counted String
- 5 - 4-Bytes Counted String

Select Option: 2 - Zero-Terminated String

Return Mechanism is by Reference

Number of M parameters: ?

Enter the maximum number of parameters that may be specified on the XCALL statement in the M program. Some of the parameters may be optional or have default values.

Enter ^ to return to the previous question.
Enter ^Q to quit this XCALL definition.

Number of M parameters: 3

Select M Parameter <1>: ?

Enter the number of the XCALL parameter to define. This is the parameter in the XCALL statement in the M program.

Enter -n to delete an existing parameter definition.
Enter ^L to list the currently defined parameters.
Enter ^ to return to the previous question.
Enter ^Q to quit this XCALL definition.

Select M Parameter <1>: **RETURN**

Parameter Mode:

- 1 - Input
- 2 - Output
- 3 - Input/Output

Select Option: ?1

The parameter is used only as input to the XCALL link routine.

Select Option: ?2

The parameter receives a value from the XCALL link routine. Its original value is not used.

Select Option: ?3

The parameter is used as both input and output. Its original value is passed as input to the XCALL link routine, and the link routine may modify it to return an output value.

Select Option: 1 - Input

Parameter Type:

- 1 - Integer
- 2 - Unsigned Integer
- 3 - Float
- 4 - String
- 5 - Assistant Routine
- 6 - Ignored

Select Option: ?1

The parameter is a signed binary integer.

Select Option: ?2

The parameter is an unsigned binary integer.

Select Option: ?3

The parameter is a floating point number.

Select Option: ?4

The parameter is a character string or some other data structure that can be laid out over an M character string.

Select Option: ?5

This parameter uses an assistant routine to convert its value between the M XCALL statement and the internal XCALL link routine. The exact types will be handled in the next prompt.

Select Option: ?6

This parameter appears in the M XCALL statement, and it is evaluated, but its value is ignored.

Select Option: 1 - Integer

Integer Size:

- 1 - 1-Byte
- 2 - 2-Bytes
- 3 - 4-Bytes
- 4 - Size-Int

Select Option: ?1

The size of the integer field is one byte.

Select Option: ?2

The size of the integer field is two bytes.

Select Option: ?3

The size of the integer field is four bytes.

Select Option: ?4

The size of the integer is the 'natural' size for the underlying system.

Select Option: 2 - 2-Bytes

Position in called routine: ?

Enter the position of this parameter in the XCALL link routine parameter list.

Enter ^ to return to the previous question.
Enter ^Q to quit this XCALL definition.

Position in called routine: 1

Parameter Mechanism:

- 1 - Value
- 2 - Reference

Select Option: ?1

The value of the parameter is passed, rather than a pointer. Thus, the called routine can not modify the value of the original parameter variable.

Select Option: ?2

The parameter is passed by reference. This means that a pointer to the value is passed, rather than the value itself. Thus, the called routine can modify the value of the original parameter variable.

Select Option: 1 - Value

Parameter Requirement:

- 1 - Required
- 2 - Optional
- 3 - Default

Select Option: ?1

The parameter must be specified on the M XCALL statement.

Select Option: ?2

The parameter is optional on the M XCALL statement. It must follow all required parameters. If it is omitted, the corresponding parameter of the XCALL link routine will be omitted.

Select Option: ?3

The parameter is optional on the M XCALL statement. It must follow all required parameters. If it is omitted, a default value will be passed to the XCALL link routine.

Select Option: 1 - Required

Select M Parameter <2>: **RETURN**

Parameter Mode:

- 1 - Input
- 2 - Output
- 3 - Input/Output

Select Option: 1 - Input

Parameter Type:

- 1 - Integer
- 2 - Unsigned Integer
- 3 - Float
- 4 - String
- 5 - Assistant Routine
- 6 - Ignored

Select Option: 5 - Assistant Routine

Input Assistant Routine Name: **asst1**

Position in called routine: 2

Parameter Mechanism:

- 1 - Value
- 2 - Reference

Select Option: 1 - Value

Parameter Requirement:

- 1 - Required
- 2 - Optional
- 3 - Default

Select Option: 1 - Required

Select M Parameter <3>: **RETURN**

Parameter Mode:

- 1 - Input
- 2 - Output
- 3 - Input/Output

Select Option: 2 - Output

Parameter Type:

- 1 - Integer
- 2 - Unsigned Integer
- 3 - Float
- 4 - String
- 5 - Assistant Routine
- 6 - Ignored

Select Option: 4 - String

String Type:

- 1 - Opaque
- 2 - Zero-Terminated String
- 3 - 1-Byte Counted String
- 4 - 2-Bytes Counted String
- 5 - 4-Bytes Counted String

Select Option: ?1

The size of the string or structure is not apparent from the string itself. The length must be passed as another parameter.

Select Option: ?2

The string will be terminated by a one-byte null character. This is the standard convention for terminating a string in the C programming language.

Select Option: ?3

The string is immediately preceded by a one byte unsigned count field which contains the length of the string. The count does not include the length of the count field.

Select Option: ?4

The string is immediately preceded by a two byte unsigned count field which contains the length of the string. The count does not include the length of the count field.

Select Option: ?5

The string is immediately preceded by a four byte unsigned count field which contains the length of the string. The count does not include the length of the count field.

Select Option: 2 - Zero-Terminated String

Maximum Output Length: ?

Enter the maximum length of the returned string. The length may be a constant value, or it may be specified by one of the input parameters to the XCALL. Note that the called routine does not have access to this value unless it is passed as an input parameter.

Enter a number to specify a constant value for the length. Enter Ix if the length is specified by input parameter number 'x' on the M XCALL statement. The parameter type must be integer, and it must be 'required' or have a default value.

Maximum Output Length: 100

Position in called routine: 4

Mechanism is by Reference

Parameter Requirement:

- 1 - Required
- 2 - Optional
- 3 - Default

Select Option: 1 - Required

Select M Parameter: RETURN

Number of constant value parameters: ?

Enter the number of parameters to the XCALL link routine that will have a constant value. These parameters are not specified on the M XCALL statement.

Enter ^ to return to the previous question.
Enter ^Q to quit this XCALL definition.

Number of constant value parameters: 1

Constant Parameter <1>: **RETURN**

Parameter Type:

- 1 - Integer
- 2 - Unsigned Integer
- 3 - Float
- 4 - String
- 5 - Assistant Routine
- 6 - Ignored

Select Option: 2 - Unsigned Integer

Integer Size:

- 1 - 1-Byte
- 2 - 2-Bytes
- 3 - 4-Bytes
- 4 - Size-Int

Select Option: 4 - Size-Int

Position in called routine: 3

Parameter Mechanism:

- 1 - Value
- 2 - Reference

Select Option: 1 - Value

Constant value: ?

Enter the value for the constant parameter.

Enter ^ to return to the previous question.
Enter ^Q to quit this XCALL definition.

Constant value: 100

Constant Parameter: **RETURN**

XCALL name: **RETURN**

Check the XCALL Definition

After you define the XCALL package, use the utility program XCALLMGR to check the definition for correctness and consistency. This step verifies that all parameters are completely defined, optional parameters follow all required parameters, and parameter data types are correct (for example: a parameter containing a string length must be an integer type).

The following terminal session illustrates how this step is performed.

```
Select Option: 2 - Check Package Definition

Package selector: TEST

          1 package selected.

Package selector: RETURN

Messages <Y/N>? Y

Enter output device <1>: RETURN

..... Package TEST is O.K.

Packages checked.
```

Generate the Source File

When the XCALL definition is complete, the utility program XCALLMGR is used to generate an external source file containing the definition. This file, which is written in the C programming language, defines the internal structures that MSM will use to convert parameters and call the XCALL routines. A source file may contain one or more XCALL packages.

The following terminal session illustrates how a source file is generated.

```
Select Option: 3 - Source Files Generation

Logical file name: TEST
Physical file name: xcalldef.c
Language: <C>

Package selector: TEST

          1 package selected.

Package selector: RETURN

.... writing source file ....
```

Compile the XCALL Package

After the source file is created, it must be compiled and linked with the rest of the external routines which implement the XCALL. This step is performed outside of MSM using the normal compile and link tools of the host operating system. Specific instructions and examples for each operating system are provided below or in the README file.

The result of this step is an executable load module which contains the XCALL routines, assistant routines, and definition tables. This module contains one or more XCALL packages that are now ready to be used.

General Issues

While writing programs in the C programming language to be used as XCALLs, keep the following issues in mind:

- The C source code written by the user must **not** contain the function *main()*.
- Many C run-time library functions cannot be used within your code. They depend on library initialization that normally is associated with the *main()* entry point. Since this initialization is not performed for dynamically loaded modules, incorrect results or system crashes could occur.
- While an XCALL is being executed, all other MSM processes are suspended until the XCALL completes.

For assistance with questions or concerns, contact MSM Technical Support.

Compilation Under MS-DOS

The following procedure shows how to compile and link a protected mode XCALL package using the WATCOM C Compiler. The file XCHEAD.ASM is used to position the data segment at the beginning of the generated module. If you use a different compiler/linker, you must follow the appropriate procedures to ensure that the data segment is at the beginning of the module.

The procedure assumes that the file *yourfile.c* contains the C source code you wrote to implement XCALL functions. The file *xcalldef.c* is the file created by the utility program XCALLMGR when you defined the XCALL functions in MSM.

This example creates an XCALL package named XCALLPKG.REX. You can select any other name for your XCALL package.

Step 1. Modify the provided file XCALLPKG.MK to reflect your file names. In the sample code below, the names that may require modification are bold and underlined:

```
CC      = wcc386p /fpc /ez /ox /ol /w3 /d1 /3s /s /zpl /ze
ASM     = 386asm -386p -nolist -twocase
LINK    = wlink
OBJ     = xthead.obj yourfile.obj xcalldef.obj
xcallpkg.rex :      xcallpkg.wlk $(OBJ)
                $(LINK) @xcallpkg.wlk
xthead.obj :      xthead.asm
                $(ASM) xthead.asm -object xthead.obj
```

```

yourfile.obj : yourfile.c
    $(CC) yourfile.c
xcalldef.obj : xcalldef.c xcdecl.h
    $(CC) xcalldef.c

```

Step 2. Create a file named XCHEAD.ASM file (or use the one provided):

```

_DATA SEGMENT BYTE PUBLIC 'DATA'
_DATA ENDS
END

```

Step 3. Modify the provided file XCALLPKG.WLK to reflect your file names. In the following sample code, the names that may require modification are bolded and underlined:

```

NAME XCALLPKG
OPTION MAP=XCALLPKG
OPTION U
FORMAT PHARLAP REX
FILE xthead, xcalldef, yourfile

```

Step 4. Modify the provided file MAKEXC.BAT to reflect your file names. In the sample code below, the name that may require modification is bolded and underlined:

```
@wmake /F xcallpkg.mk
```

Step 5. Execute the batch file:

```
C:> makeexc
```

The compiler and the linker now create an executable file named XCALLPKG.REX which can be loaded into memory with the utility program XCALL.

Compilation Under Windows

Your XCALL package must be linked as a DLL. The only name that must be exported from the package is xcfiles. You may export other names if the DLL will be called by applications other than MSM. The DLL file must be loaded by MSM to make the XCALL package available.

Use the following steps to create a DLL file which can be loaded into MSM using the utility program XCALL.

1. Create a directory for the new XCALLs.
2. Copy the following C source code files into the new directory:
 - The file that contains the source code for your XCALLs
 - The file created by the utility program XCALLMGR
3. Copy the include file xcdef.h into the new directory.
4. Copy the file xcall.def into the new directory.
5. Start Microsoft Developer Studio.
6. Using the new directory, create a New Workspace for a Dynamic Link Library.
7. Insert files into project: both C source code files and the file xcall.def.
8. Build the .DLL file.

Compilation Under AIX

To compile and link an XCALL package for use with MSM-UNIX on AIX, use the following procedure (change the underlined file names below to reflect your situation):

```
cc -c yourfile.c # your xcall routine
cc -c xcalldef.c # the file generated by the XCALLMGR utility
ld -e xcfiles yourfile.o xcalldef.o -o xcallpkg -lc
```

The parameter “-e xcfiles” must be specified as shown.

This sequence of commands creates a load module named xcallpkg. Use the utility program XCALL to load this module into MSM.

The requirements of your XCALL routine determine whether or not the parameter “-lc” is needed.

Compilation Under Digital UNIX

To compile and link an XCALL package for use with MSM-UNIX on Digital UNIX, use the following procedure (change the underlined file names below to reflect your situation):

```
cc -c yourfile.c # your xcall routine
cc -c xcalldef.c # the file generated by the XCALLMGR utility
ld -shared -hidden yourfile.o xcalldef.o -o xcallpkg -lc
```

The parameters “-shared -hidden” must be specified as shown.

This sequence of commands creates a load module named xcallpkg. Use the utility program XCALL to load this module into MSM.

The requirements of your XCALL routine determine whether or not the parameter “-lc” is needed.

Compilation Under Solaris

To compile and link an XCALL package for use with MSM-UNIX on Solaris, use the following procedure (change the underlined file names below to reflect your situation):

```
cc -c yourfile.c # your xcall routine
cc -c xcalldef.c # the file generated by the XCALLMGR utility
ld -G -B symbolic yourfile.o xcalldef.o -o xcallpkg -lc
```

The parameters “-G -B symbolic” must be specified as shown.

This sequence of commands creates a load module named xcallpkg. Use the utility program XCALL to load this module into MSM.

The requirements of your XCALL routine determine whether or not the parameter “-lc” is needed.

Managing MSM-XCALL

Overview

This section describes the two methods that can be used to load an XCALL package; explains how to define and modify the default packages list; and discusses how to display XCALL information and import/export XCALL packages.

Load an XCALL Package

Before an XCALL package can be used, it must be loaded into memory by MSM. This can be done at any time with the utility program XCALL or automatically during system startup (by using the utility program SYSGEN to put the package into the automatic load list).

In either case, MSM needs to know the name of the XCALL load module created by compiling and linking the XCALL definition files. When the module is loaded into memory, its initialization routine is executed and the definition tables are added to the list of known XCALLs. All packages and routines within the module are then available for use by M programs.

Using SYSGEN to Load a Package

The following terminal session illustrates use of the utility program SYSGEN to put the XCALL package into the automatic load list. The option shown below is selected from the Edit Configuration Parameters menu of the SYSGEN utility. For additional information on the SYSGEN utility, refer to “Generating the System” in the *MSM System Manager’s Guide*.

After the package is added to the list by the SYSGEN utility, it will be automatically loaded into memory each time the system is started.

```
Select Option: 11 - External Calls Configuration
```

```
Select XCALL Option:
```

- 1 - Files to load at startup
- 2 - Default Packages

```
Select Option: 1 - Files to load at startup
```

Enter Index Number: ?

Enter an index number in the XCALL autoloading table which will define a file to be loaded during system startup.

Enter -index to delete an existing entry
Enter ^L to list the current autoloading table.

Enter Index Number: 1

File Name: ?

Enter a file name which contains XCALL packages that will be loaded automatically during system startup.

File Name: **/users/xcalls/mathpkg**

Enter Index Number: ^L

Index	File Name
-----	-----
1	/users/xcalls/mathpkg

Enter Index Number: **RETURN**

Select XCALL Option: **RETURN**

Using XCALL to Load a Package

The utility program XCALL can be used to immediately load an XCALL package into memory. The file is not added to the automatic load list, and will not be reloaded when the system is started. The following terminal session illustrates use of the utility program XCALL.

>DO ^XCALL

File to load: **/users/ms1/xcallrout**
Are you sure <Y/N>? **Y**

File successfully loaded
File to load: **RETURN**

Define Default Packages

The MSM system maintains a list of XCALL packages that are searched for an XCALL routine when a package name is not specified on the M statement invoking the XCALL. This list is known as the default packages list.

If no default packages list is defined, the internal package named MSM is treated as the only package in the list. If a list is defined, the MSM package is not automatically included in the list, but must be added to the list definition if its inclusion is desired.

The default packages list may be defined with the utility program SYSGEN, or the utility program XCDEFPK can be used to dynamically change the list.

Using SYSGEN to Define the Default Packages List

The following terminal session illustrates use of the utility program SYSGEN to define the default packages list. The option shown is selected from the Edit Configuration Parameters menu of the utility program SYSGEN. For additional information on the utility program SYSGEN, refer to "Generating the System" in the *MSM System Manager's Guide*.

After the default packages list is defined with the utility program SYSGEN, it will be setup automatically in memory each time the system is started.

```
Select Option: 11 - External Calls Configuration
```

```
Select XCALL Option:
```

- 1 - Files to load at startup
- 2 - Default Packages

```
Enter an index number in the default packages table which will define a package to be added to the default packages list during system startup. The default packages will be searched according to the order of the index numbers.
```

```
Enter -index to delete an existing entry  
Enter ^L to list the current default packages table.
```

```
Enter Index Number: 1
```

```
Package Name: ?
```

```
Enter a package name that will be added to the default packages list during system startup. If the internal package named MSM is to be included in the default list, it must be explicitly named in the list.
```

```
Package Name: MSM
```

```
Enter Index Number: 2
```

```
Package Name: MATH
```

```
Enter Index Number: ^L
```

```
Index   Package Name  
-----  
1       MSM  
2       MATH
```

```
Enter Index Number: RETURN
```

Using XCDEFPK to Edit the Default Packages List

The XCDEFPK utility can be used at any time to temporarily change the default packages list. The change takes effect immediately, but is only effective until the system is restarted.

The following terminal session illustrates use of the utility program XCDEFPK.

```
>DO ^XCDEFPK

      XCALL DEFAULT PACKAGES DIRECTORY
      19-SEP-97    1:46 PM

Package names:

No default packages are currently defined.

Default packages list number: 1
Package Name: MSM

Default packages list number: 2
Package Name: MYXCALL

Default packages list number: 3
Package Name: RETURN

List of entered packages:
MSM      MYXCALL

OK to proceed? <N> Y

      XCALL DEFAULT PACKAGES DIRECTORY
      19-SEP-97    1:47 PM

Package names:

      1: MSM
      2: MYXCALL
```

Display XCALL Information

The XCALL directory utility program %XCD is used to display information about XCALL packages that are currently loaded into memory. It can display the names of loaded packages, a list of routines within a package, and the default packages list.

The following terminal session illustrates use of the utility program %XCD.

```
>DO ^%XCD

XCALL Directory Options:

    1 - Loaded Packages
    2 - Routines In A Package
    3 - Default Packages List

Select Option: 1

          XCALL MODULES AND PACKAGE DIRECTORY
          19-SEP-97    1:45 PM

Module name: MSM
Package name: MSM

Module name: MYXCALL
Package name: MYXCALL

Number of modules loaded: 2
Number of packages loaded: 2

XCALL Directory Options:

    1 - Loaded Packages
    2 - Routines In A Package
    3 - Default Packages List

Select Option: 2

Enter package name: MYXCALL

          XCALL ROUTINES DIRECTORY
          19-SEP-97    1:45 PM

Routine Names in package MYXCALL:
MYFUNC                MYTEST1                MYTEST2

Total routines in package MYXCALL: 3

XCALL Directory Options:

    1 - Loaded Packages
    2 - Routines In A Package
    3 - Default Packages List

Select Option: 3

          XCALL DEFAULT PACKAGES DIRECTORY
          19-SEP-97    1:45 PM

Package names:

    1: MSM
    2: MYXCALL

XCALL Directory Options:

    1 - Loaded Packages
    2 - Routines In A Package
    3 - Default Packages List

Select Option: RETURN
```

XCALL Portability

Package definitions usually are portable to different MSM platforms, unless specific parameter size values are unique to one platform. The generated C source code file can be copied to another platform and compiled there or the XCALL definition, which is stored in the global variable ^XCALL, can be copied to another system for modification or package generation.

The two options described in the following sections are provided in the utility program XCALLMGR to allow copying of the XCALL definition.

Exporting an XCALL Package

To export a package definition to another MSM system, the appropriate nodes of the global variable ^XCALL are saved in an external file, as shown in the following terminal session.

```
>DO ^XCALLMGR

      MSM - XCALL Maintenance

      1 - Packages Definition
      2 - Source Files Generation
      3 - Check Package Definition
      4 - Export Packages Definition
      5 - Import Packages Definition

Select Option: 4 - Export Packages Definition

      MSM - Export Packages of External Calls

Enter output device <HFS>: RETURN Host File Server
File Name >: myxcall.exp
Enter size of save medium (if applicable):
Enter comment for dump header : definition of MYXCALL package

Package selector: MYXCALL
                1 package selected.

Package selector: RETURN

Saving ...

^XCALL

Save complete.
```

Importing an XCALL Package

When a package definition is imported from another MSM system, the global variable ^XCALL is read from an external file. The import operation will not proceed if there is an existing package with the same name as the package to be imported. The existing package must be deleted, or the new package can be imported with a different name. Other existing package definitions are not affected by importing a new definition.

The following terminal session shows how a package is imported.

```
>DO ^XCALLMGR

      MSM - XCALL Maintenance

      1 - Packages Definition
      2 - Source Files Generation
      3 - Check Package Definition
      4 - Export Packages Definition
      5 - Import Packages Definition

Select Option: 5 - Import Packages Definition

      MSM - Import Packages of External Calls

Enter input device <HFS>: RETURN Host File Server
File Name >: myxcall.exp

Package(s) saved at 1:57 PM 19-SEP-97.
Header comment is : definition of MYXCALL package
Selective restore (allows rename) <N>: NO

Restoring...
Package: MYXCALL ... Restored
Restore Complete
```


Parameter Options

Overview

This section describes the options for defining the parameters and return value of an XCALL routine.

Parameter Mode

An *input parameter* is passed from M to the XCALL routine. The XCALL routine cannot change the original parameter value.

An *output parameter* is an M variable which receives a value from the XCALL routine. The M variable must be passed by reference in the M command which invokes the XCALL.

An *input/output parameter* passes a value to the XCALL routine and may receive a new value from the XCALL routine.

Parameter Type

The parameter type describes the basic type that the XCALL routine expects for the parameter. MSM will perform the conversion from the standard M data type to the type required by the XCALL routine. Conversely, for an output parameter, MSM will convert from the external type to the M data type.

Parameter Types

Parameter Type	Description
INTEGER	The parameter is a binary integer.
UINTEGER	The parameter is an unsigned binary integer
FLOAT	The parameter is a floating point number.
STRING	The parameter is a character string.
ASSISTANT_ROUTINE	The parameter will be passed as input to entry point included in the XCALL package. The return value of the assistant routine is then used as the actual parameter.
IGNORED	The parameter appears in the M statement, and is evaluated, but is not passed to the XCALL routine.

Number Size

If the parameter type is INTEGER, UINTEGER, or FLOAT, the size of the parameter must be specified. The following tables detail number size by parameter type.

INTEGER or UINTEGER Size

Integer Size	Description
1-BYTE	The size of the field is one byte.
2-BYTES	The size of the field is two bytes.
4-BYTES	The size of the field is four bytes
SIZE-INT	The size of the field is the natural integer size of the host system.

Float Size

Float Size	Description
REAL	The float value is stored in 32 bits (single precision).
DOUBLE	The float value is stored in 64 bits (double precision).

String Type

If the parameter type is STRING, the type of the string must be further described so that MSM knows how to pass the string length to the XCALL routine or to receive the length of an output string. For counted strings, the value in the count field is the length of the string only; it does not include the size of the count field itself.

For all output strings, a value must be specified for the maximum size of the returned string so memory can be allocated for it before the XCALL routine is called.

The descriptions provided here apply to the actual parameters of the XCALL routine, not the M variables that are used to pass or receive the values. The MSM system provides the conversion between the types described here and the M variables. The following table provides descriptions of string types.

String Types

String Type	Description
OPAQUE	The address of the string itself is passed. The size of the string is not contained in the string. For an output parameter, the returned length must be passed as a separate parameter or as the XCALL return value
ZERO-TERMINATED	For input strings, one character of X'00' is added to the end of the string. For output strings, MSM will look for an ending X'00' character to mark the end of the string. This is the normal convention for strings in the C programming language.
1-BYTE COUNTED	The length of the string is contained in a one byte field which immediately precedes the string. The address of the length field is passed as the parameter value.

String Type	Description
2-BYTES COUNTED	The length of the string is contained in a two-byte field which immediately precedes the string. The address of the length field is passed as the parameter value.
4-BYTES COUNTED	The length of the string is contained in a four-byte field which immediately precedes the string. The address of the length field is passed as the parameter value.

Parameter Mechanism

The parameter mechanism specifies how the MSM conversion routine passes the parameter to the XCALL routine. STRING parameters and output parameters are always passed by reference.

Pass by value means that the actual value of the parameter is passed to the XCALL routine. The variable containing the parameter value can not be modified by the routine.

Pass by reference means that the address of a field containing the parameter's value is passed to the XCALL routine. The routine can use the address to store a new value into the variable containing the parameter.

When a parameter is passed by reference, the MSM conversion routine makes a copy of the parameter and passes the address of the copy. The address of the true MSM data is never passed to the XCALL routine. If the routine modifies the parameter, it will not be copied back to the MSM variable unless the parameter is specified as an output parameter. The M syntax in the command invoking the XCALL routine must specify call by reference for the M variable that is to be modified.

Parameter Requirement

This option indicates whether or not the user must specify a value for the parameter on the M statement invoking the XCALL. Required parameters must be the first ones in the parameter list for both the M statement and the XCALL routine. The following table provides descriptions of the parameter requirement options.

Parameter Requirement

Parameter Requirement	Description
REQUIRED	A value must always be specified for this parameter
OPTIONAL	If this parameter is not specified on the M statement, the parameter list for the XCALL routine is truncated and no values are passed for this or any following parameters.
DEFAULT	If this parameter is not specified on the M statement, a default value is passed to the XCALL routine.

Return Interpretation

The return interpretation option describes how MSM should interpret the return value of the XCALL routine. If the XCALL is always called as a subroutine, the return value may be ignored or checked for error conditions. If the XCALL routine is called as a function, MSM must know how to convert the return value into an MSM variable data type. The following table contains descriptions of return interpretation options.

Return Interpretation

Return Interpretation	Description
VALID VALUE	The XCALL routine returns a value which should be returned as the value of the M function which invoked the XCALL.
IGNORED	The return value of the XCALL routine should be ignored. If the XCALL is invoked as an M function, the value of the function is unpredictable.
ZERO STATUS	The XCALL routine returns a value of zero when it is successful. If it returns a non-zero value, an MSM error will be generated.
NON-ZERO STATUS	The XCALL routine returns a non-zero value when it is successful. If it returns a zero value, an MSM error will be generated.
OS STATUS	The return value of the XCALL routine is interpreted according to the normal rule of the host operating system. If the return value indicates an error, an MSM error will be generated.

Sample C Language Program

Overview

This section contains a sample program in the C programming language that could be called using the XCALL definition that is provided as an example in the “Defining XCALLs section. This program is provided as example.c in the MSM-XCALL Development Package.

Example C program

```
#include <string.h>

/*****
/*
/* Define global variables
/*
/*
*****/

int max[2];
int high[2];
int low[2];

char valid_resp[] = "VALID";
char invalid_resp[] = "INVALID";

/*****
/* testinit :
/*
/* Initialize global variables
*****/

void testinit()
{
    max[0] = 104; max[1] = 84;
    high[0] = 100; high[1] = 80;
    low[0] = 97; low[1] = 77;
}

/*****
/* asst1 :
/*
/* Accepts temperatures in Celsius from M,
/* outputs to XCALL routine in Fahrenheit
/*
*****/
```

```

int asst1(int celsius)
{
    int fahrenheit;
    fahrenheit = ((celsius * 9) / 5) + 32;
    return(fahrenheit);
}

/*****
/* func1 :
/*
/* Input:
/* p1 - selector for temperature ranges
/* p2 - temperature in degrees Fahrenheit
/* p3 - length of output buffer
/* p4 - pointer to output buffer
/* Output:
/* return value - pointer to string 'VALID' or 'INVALID'
/* p4 buffer - characterization of input temperature
/* compared to the selected ranges
*****/

char *func1(short p1, int p2, int p3, char *p4)
{
    char *status;
    int len;

    if ((p1 < 1) || (p1 > 2)) { /* validate
range selector */
        *p4 = '\0'; /* returned string must be
valid */
        return invalid_resp; /* return error indication */
    }

    p1 -= 1; /* adjust for subscript use */

    if (p2 > max[p1]) /* check input against range */
        status = "CRITICAL";
    else if (p2 > high[p1])
        status = "HIGH";
    else if (p2 < low[p1])
        status = "LOW";
    else
        status = "NORMAL";

    len = strlen(status);
    if ((len + 1) < p3)
        strcpy(p4, status); /* status fits in buffer */
    else {
        strncpy(p4, status, p3-1); /* truncate response */
        *(p4 + p3 - 1) = '\0'; /* provide ending delimiter */
    }
    return valid_resp; /* return success */
}

```

Index

\$

\$ZCALL, 4
\$ZERROR, 4

%

%XCD utility program, 23

^

^XCALL global variable, 24, 25

<

<XCALL>, 4

A

AIX, 18
Assistant routine, 3, 6, 7, 27
Automatic load list, 19, 20

C

C
Programming language, 2, 6, 15, 16, 28, 31
Run-time library functions, 16
Check definition, 15
Compile
Package, 16

D

Data type, 3, 27
Default packages list, 3, 4, 21, 22, 23
Defining XCALLs, 5
Definition
Check, 15
Development package, 1

Digital UNIX, 18
Display XCALL Information, 23
DLL, 17
Dynamic Link Library, 17

E

Edit Configuration Parameters, 19, 21
Error codes, 4
example.c, 1, 2, 7, 31
example.pac, 1, 2, 7
Export
Package, 24

G

Generate source file, 15

I

Import
Package, 25

L

Load
Module, 16, 18, 19
Package into memory, 6, 19, 20

M

main(), 16
make, 2
Makefile, 1, 2
MAKEXC.BAT, 1, 2, 17
Microsoft Developer Studio, 17
MS-DOS, 16

N

Number size, 28

P

Package, 3

 Compile, 16

 Import, 25

Parameter, 3

 Mechanism, 29

 Mode, 27

 Options, 27

 Requirement, 29

 Type, 27

Pass by

 Reference, 3, 4, 29

 Value, 3, 29

Portability, 24

R

README, 1, 2

Return Interpretation, 30

Routine, 3

S

Solaris, 18

Source file

 Generate, 15

String

 Type, 28

Syntax, 4

SYSGEN utility program, 19, 21

W

WATCOM C Compiler, 2, 16

Windows, 17

X

XCALL utility program, 17, 18, 19, 20

xcall.def, 1, 2, 17

XCALLMGR utility program, 2, 5, 6, 15, 16, 24

XCALLPKG.MK, 1, 2, 16

XCALLPKG.REX, 16, 17

XCALLPKG.WLK, 1, 2, 17

xcdef.h, 1, 2, 17

XCDEFPK utility program, 21, 22

xcfiles, 17

XCHEAD.ASM, 1, 2, 16, 17

Z

ZCALL, 4