THE PENNSYLVANIA STATE UNIVERSITY
Computation Center

CONTRIBUTED PROGRAM
Contributor: John R. Mashey, Computer Science Dept.

360 ASSEMBLER LANGUAGE XMACRO PACKAGE
Program: July 1970(V.4.0), Jan. 1973 (V.5.0) Writeup: Jan. 1973

XMACRO PACKAGE - SUMMARY,IMPLEMENTATION, SYSTEM GENERATION

PURPOSE

This writeup is provided to summarize the characteristics and facilities of the XMACRO PACKAGE, and enable a system programmer to add the macros and subroutines to a system.

INTRODUCTION

The XMACRO package is a system of macro definitions and assembler subprograms which together offer a number of useful facilities to the assembler language programmer. In general, the parts of the system have been written following these design considerations:

ERROR-PROOF - the programs supply a great deal of error checking, and also make assumptions where required. The macros are particularly notable for the fact that they do not disturb registers or condition code, and so can normally be used almost anywhere.

SIMPLICITY - the programs are designed with many default options, and generally require as little information from the programmer as possible.

FLEXIBILITY - the programs attempt to supply many options which are not required, but can be used to good advantage by the sophisticated programmer. The design generally allows students to use the macros effectively over a period of increasing knowledge.

NOTE AT THE CURRENT TIME, ALL OF THESE MACROS ARE SUPPORTED UNDER OS/360 AND EQUIVALENT SYSTEMS. Some of the macros can be used in any systems, since they have no system-dependent code. The ones which perform I/O may require modifications to run under DOS or other systems.

The various macro definitions included in the system generally fall into several different categories, which are as follows:

1. DEBUGGING macros.

   XSNAP provides dumping of registers and storage, and permits the output to be produced or not, depending on both assembly-time and execution-time tests. XDUMP is a simpler version of XSNAP, provided mainly for compatibility with ASSIST. XSTOP supplies a simple way to control infinite loops in programs.

2. INPUT-OUTPUT macros.

   XPNCH, XPRNT, XREAD offer simple procedures for performing the input-output operations for card punch, line printer, and card reader. A single macro call suffices for each operation, thus requiring no OPEN's, DCB's, etc, and little extra JCL is needed. XGET and XPUT allow more flexibility, providing the ability to give dynamically for each call the file to be referenced. Thus, these macros can support tape, disk, drum I/O besides unit-record I/O.

3. SUBROUTINE LINKAGE macros.

XSAVE and XRETURN generate standard OS/360 subroutine entry and exit code, like the SAVE and RETURN macros, but offering many additional services. These include printing trace messages on subroutine entry/exit, setting up base registers, and generating save areas.

4. DECIMAL CONVERSION macros

XDECI allows the user to perform free-format conversion of decimal characters strings. A single call of XDECI scans for a decimal number, ignoring leading blanks, converts the value to binary, places it in a register, and also supplies a scan pointer to the end of the string, allowing for multiple numbers on an input card, for example. XDECO converts a binary register value to edited decimal form, suitable for later output.

5. HEXADECIMAL CONVERSION macros

XHEXI AND XHEXO perform the same conversions as XDECI and XDECO, except the input string used by XHEXI contains hexadecimal numbers. Also with XHEXO, a register's contents may be printed out in hexadecimal form.

6. GENERATION CONTROL macro

The macro XSET can be used to suppress generation of many of the other XMACROs. This permits debugging code to remain in a program, but suppressed for a production program. XSNAP, XSTOP, and the I/O macros can be eliminated entirely, and the trace code supplied by XSAVE and XRETURN can be removed.

7. REGISTER EQUATE macro

EQUREGS can be used to generate set(s) of register equate symbols. Although it is not an integral part of the XMACRO system, it is included because the XMACROs generally allow equate symbols to be used to indicate registers.

8. MISCELLANEOUS SUPPORT macros

Various macros are used directly or indirectly by the macros listed above. XCHAR permits right-end substring extraction with safety. XIDENT is used by XSAVE to create any identifier code needed at entry points. XIOGN is used to generate the supporting control sections for the I/O macros, and can be called to create new or different I/O modules tailored to the requirements of a given installation. XIONR is called by the I/O macros to generate a data block and call to the appropriate I/O module. XLOOK is used to look up one operand in a list, and is utilized by many of the other macros for decoding operands. XMUSE aids XSAVE in setting up USINGs, and XSRNR is used to generate register save/restore code for XSAVE and XRETURN. Finally, XSRTR creates trace and register dumping code when needed by XSAVE and XRETURN.
XGPGEN, XXGPSRCH are used for generation of XGET/XPUT support.

9. ASSIST COMPATIBILITY macros

A number of the macros above are used to allow ASSIST programs to be run as is under the system directly. In addition to the XLIMD macro, which is supplied for compatibility only and creates no code, the macros required are XDECI, XDECO, XDUMP, XGET, XHEXI, XHEXO, XPRNT, XPUT, XREAD, and all their inner macros.

MACRO SUMMARY TABLE

Macro Name	Flags	Code Size(dec) (bytes)	Uses Macros	Macros Used By	Calls Support Module	Module Size (bytes)
EQUREGS	P	0				
XCHAR	G	0		XRETURN,XSAVE,XSRNR		
XDECI	A P	50-56			XXXXDECI	144
XDECO	A P	42-44			XXXXDECO	80
XDUMP	A P	80-150	XSNAP		XXXXSNAP	2094
XGET	AGPS	32-42	XIONR		XXXXGET	950
XHEXI	A P	44-46			XXXXHEXI	504
XGPGEN	A	1000	ABEND,CLOSE DCB,FREEMAIN,FREEPOOL,GET, GETMAIN,OPEN,PUT,XXGPSRCH	-> creates XXXXGET/PUT modules		
XHEXO	A P	42-44			XXXXHEXO	108
XIDENT		-		XSAVE		
XIOGN	A G	500-up	ABEND,CLOSE, DCB,FREEPOOL,GET/PUT,WTO, XOPENBLK	-> creates XXXXREAD,PRNT, etc m		
XIONR	A P	-		XGET,XPNCH, XPRNT,XPUT,XREAD	(any XXXX- I/O)	
XLIMD	P	0				
XLOOK	AG	0		XSAVE,XSNAP XSRTR		
XMUSE		-		XSAVE		
XOPENBLK	AGPS	-	OPEN	XIOGN		
XPNCH	AGPS	32-42	XIONR		XXXXPNCH	664
XPRNT	AGPS	32-42	XIONR	XSRTR	XXXXPRNT	768 *1
XPUT	AGPS	32-42	XIONR		XXXXPUT	1000
XREAD	AGPS	32-42	XIONR		XXXXREAD	592
XRETURN	GPS	2-up	XCHAR,XSRNR XSRTR		XXXXPRNT XXXXSNAP	768 *2 2094 *2
XSAVE	GPS	0-up	XCHAR,XIDENT XLOOK,XSRNR, XSRTR		XXXXPRNT XXXXSNAP	768 *2 2094 *2
XSET	GP	0				
XSNAP	AGPS	80-up	XLOOK	XDUMP,XSRTR	XXXXSNAP	2094 *1
XSRNR	G	-	XCHAR	XRETURN,XSAVE		
XSRTR	G	-	XLOOK,XPRNT XSNAP	XRETURN,XSAVE		*3
XSTOP	GPS	36-46				
XXGPSRCH	AGPS	-		XGPGEN		

NOTES ON THE MACRO SUMMARY TABLE

The meaning of the code letters under Flags is as follows:

A : the macro is directly or indirectly required for complete compatibility with ASSIST assembler programs.

G : the macro references global set symbols, as noted in the next section.

P : the macro is a public macro, i.e., it can be made generally available, and a writeup exists which is easily distributable.

S : the code generation of the macro can be affected by use of the XSET macro, which can be used to cancel or allow debug code.

The values given under Code Size are approximate in some cases. If a value of 0 is given, the macro definitely generates no code. If a dash is noted, the macro is an inner macro which generates code, but its size is included in the public macro which calls it. The value #-up indicates that the macro can generate wildly varying amounts of code, depending on the operands coded. However, these macros generally go to great lengths to generate the minimum possible amount of code.

The lists of macros used and macros used by give only XMACROs used or used by directly, and do not include any IBM macros used.

*1 calls to these modules may be generated depending on the type of operand coded in a TR= option of XSAVE and XRETURN, and will not be called if that macro is turned offby XSET at the time the code is generated.

*2 XRETURN and XSAVE may call these modules, as noted in *1.

*3 XSRTR is called to analyze the TR= option for both XRETURN and XSAVE, and generates approriate calls to XPRNT or XSNAP as required. If trace code is not to be included, this macro can be dummied out to eliminate the possibility of any trace and dumping code. In addition, it can be altered to add additional tracing features, such as adding an IF= test on the embedded XSNAP, having it dump storage areas, etc. See the macro definition itself for details.

GLOBAL SET SYMBOL TABLE

The following table is provided for the programmer who wishes to ensure that any set symbols he uses will not cause conflicts with those employed by XMACROs.

Variable Name	Type	Used by Macros	Comments
&XCSECT	GBLC	XSAVE	used to save &SYSECT
&XIOGNST	GBLB	XIOGN	dsect generation cntrl
&XPNCHST	GBLB	XPNCH,XSET	generation control
&XPRNTST	GBLB	XPRNT,XSET,XSRTR	generation control
&XREADST	GBLB	XREAD,XSET	generation control
&XRETUST	GBLB	XRETURN,XSET	generation control
&XSAVE	GBLC	XRETURN,XSAVE	current savearea name
&XSAVEST	GBLB	XSAVE,XSET	TR= generation control
&XSNAPST	GBLB	XSET,XSNAP,XSRTR	generation control
&XSTOPST	GBLB	XSET,XSTOP	generation control
&XTIMEST	GBLB	XSET, (XTIME-future)	generation control
&XXCHAR	GBLC	XCHAR,XRETURN,XSAVE,XSRNR	returns substring
&XXLOOK	GBLA	XLOOK,XSAVE,XSNAP,XSRTR	returns list index

NOTES ON THE GLOBAL SET SYMBOL TABLE

Each generation control variable (one ending with characters ST) allows code generation by the corresponding macro when it has its initial value of 0, and suppresses it when set to 1 by XSET. Note that the only code of XSAVE and XRETURN which is suppressed is the trace code generated by calls to XSRTR.

The XTIME macro mentioned is part of a timing system which is still under test, and so is not yet available.

SUPPORT MODULE TABLE

Many of the XMACROs create calls to support modules which actually perform the requested operations. In general, these calls are transparent to the registers and condition code. This table lists these modules and some of their characteristics.

Module	Size(dec)	Comments
XXXXDECI	144	used by XDECI. does scanning and conversion.
XXXXDECO	80	used by XDECO. does output conversion.
XXXXGET	950	manages XGET files
XXXXHEXI	504	used by XHEXI, does scanning and conversions
XXXXHEXO	108	used by XHEXO, does the output conversions.
XXXXOPEN	?	called by I/O modules to do special OPEN
XXXXPNCH	664	XPNCH support. generated by 1 call to XIOGN.
XXXXPRNT	768	XPRNT support. generated by 1 call to XIOGN.
XXXXPUT	1000	manages XPUT files
XXXXREAD	592	XREAD support. generated by 1 call to XIOGN.
XXXXSNAP	2094	XSNAP, XDUMP support.

ADDING XMACRO PACKAGE TO A SYSTEM

Two files on the distribution tape can be used to add the components of the XMACRO package to the appropriate system libraries (note: consult the writeup ASDISTRB, which lists all the files on the distribution tape and also gives other information about them). This process can be summarized as follows:

1. From the file XMACDEFS create a partitioned dataset of the macro definitions, using the utility IEBUPDTE.
2. From the file XMSOURCE create a partitioned dataset of the source programs of the required support modules.
3. Assemble each member of the second dataset and place it (or a load module form of it) into an appropriate library of object modules or load modules. Note that the first partitioned dataset must be concatenated to SYS1.MACLIB when performing these assemblies, since a number of the modules use certain of the macros (EQUREGS, XIOGN).

The remainder of this section gives an example of the programs required to perform this process. It is assumed that the symbol disttape is the name of the distribution tape, and that LABEL=# gives the correct label number for each file used, as described in the FILELIST writeup. It is also assumed that the reader is familiar with the IBM utility IEBUPDTE.

STEP 1. CREATE MACRO LIBRARY

The file XMACDEFS contains all of the macro definitions, in alphabetical order, separated by IEBUPDTE control cards, which are of the following form:

```
./ ADD LEVEL=40,SOURCE=0,NAME=macroname
```

Thus, the following program will add all the macros to the dataset called XMACRLIB. For best use, this dataset should be one which is included in the SYSLIB cards of local catalogued procedures for assembler usage, i.e., the macros should be added to the local written system macro library, if one exists.

```
//CREATEXM EXEC PGM=IEBUPDTE,PARM=NEW
//SYSPRINT DD SYSOUT=A
//SYSIN DD UNIT=tape,VOL=SER=disttape,LABEL=#,DSN=XMACDEFS,
//             DISP=(OLD,PASS)
//SYSUT2 DD UNIT=unit,VOL=SER=volser,DSN=XMACRLIB,DISP=(NEW,CATLG),
//             SPACE=(CYL,(1,,1)),DCB=SYS1.MACLIB
```

Note that the SPACE allocation may need to be altered, but the DCB must be the same as SYS1.MACLIB, in order for it to be concatenated with it and used by the assembler later.

If changes must be made to the macro definitions, an extra preliminary pass can be made using IEBUPDTE to create a modified version of XMACDEFS on another file, then running the above sort of program using the new file. Unwanted macros can be deleted this way.

STEP 2. CREATE PARTITIONED DATASET OF SOURCE MODULES

The file XMSOURCE contains the source programs for the support modules which may be required by the XMACROs. They are separated by ./ ADD cards in the same way as are the macro definitions. Thus the following program can be used to create a source library containing them:

```
//CREATESO EXEC PGM=IEBUPDTE,PARM=NEW
//SYSPRINT DD SYSOUT=A
//SYSIN     DD     UNIT=tape,VOL=SER=disttape,LABEL=#,DSN=XMSOURCE,
//             DISP=(OLD,KEEP)
//SYSUT2 DD  UNIT=unit,VOL=SER=volser,DSN=XMODLIB,DISP=(NEW,CATLG),
//             SPACE=(CYL,(1,,1)
```

Again, if modifications are desired, IEBUPDTE can be used to make them before this step is done. In particular, the I/O support modules may require modification to make them more convenient for a particular installation. This should generally involve only coding a different set of operands in the calls to XIOGN which actually generate the I/O modules (XXXXPNCH,XXXXPRNT,XXXXREAD). The writeup XIOGN/XIONR should be consulted for details on creating I/O modules tailored to specific needs. Note that the source programs given (which each consist of a few macro calls) are those used at PSU, and are set up to follow local conventions. A typical modification might be to make XPNCH, XPRNT, and XREAD use SYSPUNCH, SYSPRINT, and SYSIN instead of the DDNAMEs supplied.

After this step has been completed, the dataset should include the members XXXXDECI, XXXXDECO, XXXXGET, XXXXHEXI, XXXXHEXO, XXXXOPEN, XXXXPRNT, XXXXPNCH, XXXXPUT, XXXXREAD, and XXXXSNAP, if none of them have been deleted.

STEP 3. ASSEMBLE SOURCE MODULES

The following program assembles the source module XXXXSNAP. The same process should be repeated for other modules desired.

```
//ASSMBL EXEC ASMFC
//ASM.SYSLIB DD DSN=XMACRLIB,DISP=SHR
// DD DSN=SYS1.MACLIB,DISP=SHR
//ASM.SYSIN DD DSN=XMODLIB(XXXXSNAP),DISP=SHR
```

The resulting object module can be placed in an object module library, or if ASMFCL (or equivalent) is used, the resulting load module (on SYSGO) can be placed in a load module library.

After all of the macros and support modules have been added to their respective libraries, they may be tested using the test programs supplied in the file XMACTEST. This file should be punched, and the programs used as test decks. Note that the Job Control Language cards which are part of this file may have to be modified to fit local conditions, since they are currently set up for PSU catalogued procedures. This will be especially necessary if DDNAMEs are changed for the I/O modules.