ASSIST Assembler Replacement User's Guide

Program&Documentation: John R. Mashey

Project Supervision : Graham Campbell

PSU Computer Science Department

Preface

This manual is the key reference source for the programmer who uses the replacement facility of ASSIST. This facility allows the programmer to write and test his own versions of certain program modules which are part of the ASSIST Assembler. The modules which are replaceable perform a wide variety of functions, thus allowing for a number of different course assignments covering important segments of a running 360 Assembler. Among those replaceable are modules for management of the symbol table, base register table, scanning and conversion of various constant types, and evaluation of both self- defining terms and general expressions. The entire replacement process can be performed with low overhead, in-core, and batched, while allowing the user program no possible way to damage the rest of the ASSIST system.

The first part of this manual briefly describes the internal structure of the ASSIST assembler, and lists the steps in the entire replacement process. Also included are the overall register and linkage conventions required of all replaceable modules.

The second section describes the additional debugging facilities available to the writer of a replacement module. The third section shows the deck setup, Job Control Language, and PARM options needed to make a replacement run. The fourth section lists all messages which may be printed by the ASSIST Replace Monitor during a replacement run.

The reader should be familiar with the following manual:

ASSIST Introductory Assembler User's Manual

The above manual gives various information which may be required to write a program which can be run under ASSIST, and explains the various messages which may generated (other than Replace Monitor messages). Note also that this manual is structured similar to the above one.

For replacement of certain of the modules, it may be necessary to examine the following manual for additional required information:

ASSIST System Program Logic Manual

The ASSIST Replacement Process

Overview of the ASSIST Assembler

The ASSIST Assembler is a section of the entire ASSIST System which translates a deck of S/360 Assembler Language statements into object code, in memory. It is made up of approximately 30 control sections, of which 3 are main control programs. The overall control program is named MPCON0, which calls the main programs for each of the two passes in the assembler, and also calls all initialization and termination entry points for the various other modules in the assembler.

During the first pass, under control of MOCON1, each card in the input source deck is read, scanned for label and opcode, and processed partially according to the type of opcode. Each statement is given a location counter value during this pass, and some types of statements are completely processed, such as EQU, START, ORG, etc. Each card image and its associated information is then saved into a large dynamic work area, until an END card is encountered.

During the second pass, each statement saved in the dynamic area is retrieved and processed. Several different routines control the scanning of each statement and production of object code from it. Each statement's object code, if any, is loaded into memory, and the statement printed.

Approximately half of the modules of the assembler can be replaced using the ASSIST Replace Monitor. In general, these modules are those which are fairly low-level routines, which are not required to have communication with many other modules, and which generally do not have to be able to examine variables and flags global to the entire assembler. They definitely are never required to modify storage outside the limits of their own storage. These characteristics make it possible for them to be replaced without requiring a great deal of knowledge of the internal workings of the ASSIST Assembler.

Steps in the Replacement Process

1. The programmer writes one control section which is to be assembled and used as a replacement for the existing one in ASSIST of the same name. This control section must have the following characteristics:

   a. The CSECT and ENTRY names (if any) must be defined and spelled exactly as the existing ones.
   b. Certain replaceable modules (such as EVALUT), are permitted to call existing ASSIST modules. Any module so called can be done so by listing the module name in an EXTRN statement, then referencing the module name by use of a V-type address constant.

2. After the user program is assembled and loaded into memory, the Replace Monitor searches its list of replaceable control sections for one defined as a csect in the user program. The required entry point names are found, if possible, in the user program. During this process, the Replace Monitor modifies certain address constants in the main control table of the assembler, which will permit it to regain control every time one of the replaced entry points is called. The messages labeled AR000, AR001, and AR002 may appear on the listing at this point. If it cannot find a legally replaceable csect name, the message AR100 is printed, and the replacement process terminated. The latter can also occur if the user program contains more serious errors than given by the value of the NERR parameter.

3. Various functions are performed to initialize the user program for later execution. These include initializing the user RFLAG to the value given by the RFLAG= option in the PARM field (see PART III). Then, instead of executing the user program directly, the ASSIST Assembler is called to process a test deck, which follows the user program.

4. During the assembly of the test deck, any of the replace program's entry points may be called. Any such call is intercepted by the Replace Monitor. Using previously saved information, it supplies the parameter values to the original ASSIST entry point called, which returns the correct set of values to be computed by that entry.

   At this point, depending on certain bits in the current value of the user RFLAG, various debugging information may be printed. This may include the current card image being processed, the values of 5 parameter registers on entry to the Replace Monitor, and their correct values as returned by the original ASSIST module. These messages have labels AR051, AR052, AR054, respectively.

5. At this point, a check is made to assure that the entry point called actually was defined properly by the user. If not, the AR101 message is given, user storage is dumped, and the interception of calls is terminated. Otherwise, the user registers and counters are prepared, and the user program executed beginning at the address in his program given by the called entry point. The user program is not executed directly, but is interpreted to prevent it from

damaging any part of ASSIST. The user program may thus access storage outside its area, but may not modify such storage.

6. The user program is interpreted until it either terminates normally by returning to the return address supplied to it in R14, or terminates with some error.

7. If the user program terminated normally, the register values it returned are checked against the ones returned by the original module. In some cases, the exact register values do not matter, but any value definitely wrong is noted. If anything is actually wrong, any debug information not already printed during step 4 is printed now. Then the values of the user-returned parameter registers are printed (AR058), followed by a message flagging the incorrect registers (AR059). The AR058 message may be printed in any case if the appropriate bit in the current value of the user RFLAG is turned on. Another bit in the RFLAG is set if an error has occurred. This bit may be tested by the user program the next time it is called.

   The correct values are placed in the parameter registers, and control is returned to the program which originally called the replaced entry point.

8. If the user program did not terminate normally, and the error was a branch out of the user program, it may be the case that the user program was attempting to call some other original ASSIST module. The call is checked to see if it is a legitimate one. If so, the parameter registers may be printed (AR050), and then checked to make sure they contain legal values. If they are illegal for any reason, they are flagged with message AR059, the user program is dumped, and no further calls are made to user entry points. If the call is legal, the desired routine is called, and its parameter values placed in the user's registers, and step 6 is begun once more.

9. Finally, the assembly of the test program is completed, with all calls having been made to the appropriate entry points of the user replacement program. Messages AR003 and AR004 are then printed, giving various statistics about the performance of the user program. These include the number of times each entry point was called, the total number of instructions executed by each entry, the number of times the values returned by the user program were incorrect, the average number of instructions executed per call, and the percent of the calls which were handled incorrectly.

10. If the option BATCH was specified, control returns to step 1, thus allowing different modules to be tested during one run. Otherwise, ASSIST execution terminates.

Register and Subroutine Linkage Conventions

Register Usage

The general purpose registers are referred to by two separate sets of symbols. The first is a set of absolute register equates, the symbols R0-R15 being used for registers 0-15. In addition, a second set exists which has more mnemonic meaning. The user is urged to utilize only symbolic registers in his program, and should thus include any of the required EQU instructions in his program. In particular, registers 7-11 should be coded using the symbols RA-RE. The additional symbolic register equates are as follows:

```
RW        EQU   R3                 GENERAL WORK REGISTER 1
RX        EQU   R4                 GENERAL WORK REGISTER 2
RY        EQU   R6                 GENERAL WORK REGISTER 3
RZ        EQU   R6                 GENERAL WORK REGISTER 4

RA        EQU   R7                 PARAMETER REGISTER 1
          This register is commonly used as a scan pointer register
          inside the assembler.
RB        EQU   R8                 PARAMETER REGISTER 2
          This register is commonly used to pass a control value to
          a subroutine, and on return, almost always contains either
          an error code, or a zero to show no errors.
RC        EQU   R9                 PARAMETER REGISTER 3
          This register is most often used in the assembler for passing
          a 24-bit value (such as the result of an expression or a
          self-defining term).
RD        EQU   R10                PARAMETER REGISTER 4
RE        EQU   R11                PARAMETER REGISTER 5
          Registers RD and RE may be used for subroutines needing more
          than two or three arguments, but are more commonly used as
          work temporary work registers.

RAT       EQU   R12                ASSEMBLER TABLE POINTER-READ ONLY
          This register points the main assembler table (VWXTABL csect,
          AVWXTABL dsect) during an assembly.  No subroutine in the
          assembler may modify this register.
RSA       EQU   R13                SAVE AREA POINTER/BASE REG FOR SOME
          This register is used to point to an OS/360 save area, for
          any subroutine which may call another.  Almost all subroutines
          use this as a base register if they are not lowest-level
          routines.
RET       EQU   R14                RETURN ADDRESS USED IN CALLS
          This is used in subroutine linkage for the return address to
          a calling program.  This symbol is generally used whenever
          subroutine linkage is being set up, while R14 is used when the
          register is being used as a temporary work register.
REP       EQU   R15                ENTRY POINT ADDRESS/OFTEN USED BASE
          This register is used to hold the entry point address for all
          subroutines in the assembler.  Lowest-level routines usually
          use this as a base register.  In other routines, this may be
          used as a local work register, in which case the symbol R15
          is normally coded.
```

Linkage Conventions - The Assembler

The linkage conventions inside the ASSIST assembler consist of a few modifications to the standard OS/360 linkage conventions, which have been changed mainly to save time and space. The differences are as follows:

1. Registers R0-R6 (or R0-R2, RW-RZ) are protected across any calling sequence and must be restored if changed. R14 (RET) must also be restored if changed before returning.

2. Register R12 (RAT) may not be changed by any routine.

3. Registers R7-R11 (RA-RE) are used for parameters and temporary work registers, and are not protected at all across calls. No routine ever requires more than five arguments, so these five registers are sufficient.

4. Except for the above, all normal OS/360 conventions are followed regarding save area linkage requirements and usage. In general, most routine only save as many registers as required. Lowest-level routines use R15 as a base, and do not perform save area linkage, other routines usually use R13 as a base and save area pointer.

5. For replacement runs, the user must include any needed EQU symbols for registers. Note that all documentation and output produced by the Replace Monitor refers to registers 7-11 as RA-RE, so that using these symbols in a replacement program will aid reading the various diagnostic output produced.

Replace Monitor Debugging Aids

The RFLAG

Communication between the user program and the Replace Monitor is achieved through the use of the User Replace Flag, called the RFLAG. This is a two-byte area of storage which may initialized for an entire run using the RFLAG= option in the PARM field. Certain bits in it determine which diagnostic messages the Replace Monitor prints when it intercepts a call to a replaced module. These bits can also be changed by the user program during execution, thus allowing the user to obtain additional information when needed. The various bits of the RFLAG are used as described in the table below.

BYTE	BITS	DECIMAL	BINARY	MEANING IF BIT ON	(AR### MESSAGE)
0	0-7			currently unused, user can set or test for his own purposes.	
1	7	1	00000001	print current statement on entry	(AR051)
	6	2	00000010	print registers RA-RE on entry	(AR052)
	5	4	00000100	print correct regs RA-RE, on exit from original ASSIST module	(AR054)
	4	8	00001000	print registers RA-RE on exit from user replacement module	(AR058)
	3	16	00010000	print registers RA-RE if user module calls an original ASSIST module.	(AR050)
	1,2	64,32	01100000	reserved for future use	
	0	128	10000000	is set to 1 when there is an error parameter registers returned by the user program. Is set to 0 if acceptable.	(AR059)

Bit 0 of byte 1 can be used to start extra debugging output only after an error occurs. See the XREPL example for this action. The entire first byte is reserved for the user program, such as for additional debugging flag bits for controlling the program. Note that bits 5,6,7 are tested before the call to the user program. Thus, changing them affects output beginning at the next call to a user module.

The XREPL Instruction

The XREPL instruction is an SI format instruction, in which the immediate field is used to specify a type of action. It is coded as

```
XREPL addr,code
```

with CODE meaning as follows:

0	set the RFLAG from the 2-byte area specified by ADDR.
1	fetch the RFLAG into the 2-byte area specified by ADDR.
2	fetch the number of instructions left into the 4-byte area given by ADDR. This value is decremented each time an instruction is done.

The following gives an example of the use of XREPL:

```
XREPL  MYRFLAG,1     get the value of the RFLAG
TM     MYRFLAG+1,128  was there an error last time
BZ     *+12          no, don't reset it
OI     MYRFLAG+1,8+4+2+1  set all these for debug output
XREPL  MYRFLAG,0     reset the RFLAG to new setting
```

Job Control Language and PARM Options

Job Control Language

The deck setup for a single-job replacement run is as follows:

```
//  a JOB CARD
// EXEC ASSIST,PARM='REPL,other options if any'
//SYSIN DD *
.....user-written replacement program.....
        END  1   end card of replacement program
..... user test deck for his replacement program.....
/*
//
```

The deck setup for a replace program run under BATCH is:

```
$JOB   ASSIST  ACCT#,REPL,other options, if any
..... user-written replacement program
$ENTRY              (required to initiate test)
..... user test deck for replacement program
$ENTRY              (optional, if user wants assembled test
                    program to execute also - unlikely)
```

PARM Options

The following PARM field options are of particular interest to the user of the replacement facility. (see PART III. of USER'S GUIDE).

```
 REPL      required if the run is to be a  replacement  run  rather  than
           just a normal assembly and execution.

 RFLAG=number   coded to initialize the value of the RFLAG for the entire
           run.  The default value is 0.

 BATCH     may be coded if the user wants to test more than  one  module,
           or more than one version of the same module in the same run.

 I=number  the instruction count limit specified applies to each call  of
           a replacement module.  It is therefore recommended that this
           optional operand be coded, and that its value be fairly small.
```

Replace Monitor Messages

The following lists the messages which may be produced during a replace run by the Replace Monitor. Note that all these messages are printed inline with output produced by other sections of ASSIST. In particular, Replace Monitor output is embedded in the listing of the user test program, which can possibly make it difficult to read in some cases. A helpful procedure is to run the test program by itself under ASSIST, thus obtaining a listing, then insert a PRINT OFF command at the beginning. This will remove most of he test program listing.

All Replace Monitor messages are of the form ///AR### message. The type of message is indicated by the value of ###, as follows:

000-049 - informative or warning messages.
050-099 - debugging output messages,produced during intercepted call.
100-199 - severe error message, causing replacement interception to end.

AR000
REPLACE CSECT: name ///
This message appears immediately after the replace csect has been assembled, with name being the name of the replacing csect.

AR001
REPLACE ENTRY: name AT LOCATION: xxxxxx ///
If message AR000 appears, each properly defined entry point in the csect will be listed here with its location xxxxxx in memory. Note that a csect which can be entered through its csect name is also listed.

AR002
REPLACE ENTRY: name NOT FOUND AS CSECT OR ENTRY ///
This message may appear with the AR000 and AR001 messages for any entry or csect name which is required, but either not defined in the user program or not declared as CSECT or ENTRY. If this entry name is called during execution, its execution will be terminated with an AR101 message and storage dump.

AR003
STATISTICS: # INSTRUCTIONS # CALLS # WRONG INSTRS/CALL %WRONG
This message appears after the test program is assembled.

AR004
name : 5 decimal numbers
One of this message appears for each entry point after AR003. It describes the performance of the named entry point during the run.

AR050
ON CALL TO name REGISTERS RA-RE (values of regs 7-11)
This message may be printed if the RFLAG byte 1 bit 3 is set and the user program calls some other ASSIST module. It may also be printed if the user program tries to pass illegal parameter values to the routine name.

AR051
ON ENTRY TO name STMT ADDR: xxxxxx -> card image

This message is printed before calling the user program, and shows the current statement being processed, if any. The address of the card image is given by xxxxxx, which corresponds to the first character following the '>' in the message. The message is if RFLAG byte 1 bit 7 is set before the call, or if an error occurs in the user program.

AR052

ON ENTRY TO name REGISTERS RA-RE: (values of 5 registers)

This message displays the 5 parameter registers before the user program name is called, and is printed if RFLAG byte 1 bit 6 is on before the user program is called, or if there is an error.

AR054

ON EXIT FROM name REGISTERS RA-RE: (values of 5 registers)

This message shows the correct values of the parameter registers as returned by the original ASSIST module name. It is printed if RFLAG byte 1 bit 5 is on before call to the module, or if the user program makes an error.

AR058

ON EXIT FROM name REGISTERS RA-RE: (values of 5 registers)

If RFLAG byte 1 bit 4 is on after completion of the user program, or if there is an error, this message appears, and gives the values of the parameter registers as returned by the user entry name.

AR059

WARNING: ERROR IN USER REGS: error list

If any of the user registers has an incorrect value, this message is printed, either following AR050 or AR058, depending on whether the incorrect value(s) were in a call to another module or in a return of values to the calling program.

The error list consists of one or more of the following:

```
R0-R6       when a user program returned, the values in registers 0-6
            were not all the same as when it was called.
R12         the user program modified  the  value of the assembler table
            pointer, which is not permitted.
R13         the user did not restore the save area pointer.
$$$$$$$$    The dollar signs indicate a register shown in messages AR050
            or AR058 as incorrect.
```

If this message appears, RFLAG byte 1 bit 0 is set to 1 for the next time the user program is called.

AR100

REPLACE CSECT NOT FOUND - REPLACE ABORT ///

This message appears immediately after the assembly of the user program. None of the allowable csect names were found as a csect in the user program.

AR101

INVALID ENTRYPOINT NAME: name CALLED. REPLACE ACTION ABORTED ///

If name appeared in an AR002 message and is called, this message appears, followed by a dump of user storage and the last values of the user registers.

AR102

USER PROGRAM ABENDED DURING REPLACEMENT ///

Replace action is aborted and a dump given.