RAG SOFTWARE

AEMS MACRO ASSEMBLER

USER'S GUIDE

Macro Assembler
Version 1.1
R. A. Green

The AEMS Macro Assembler is a full featured 9900 macro assembler. It makes use of the memory available to assemble large complex programs. Features include: macro definition and expansion, compact listing format, symbol cross reference listing, string and floating data types, COMMON data area definition, complex relocation and page addresses.

CONTENTS

April 1993

INTRODUCTION

The assembler reads a source program, written in assembler language, and translates it into an object program, in machine language. The source program statements are read from the "source file" which may be augmented by one or more "copy files". The object program is written to the "object file". During assembly, an "object listing file" can be created which shows the object code generated along with the source statements.

Before the object program can be executed it must be loaded into the computer's memory. The Assembler does not execute programs, it simply translates them from assembler language to machine language.

This package is being made available via the Fairware concept. If you are using the package, send a donation to:

R. A. Green
1032 Chantenay Drive
Gloucester, Ont. Canada
K1C 2K9

And, at the same time, distribute complete copies of the package to your friends.

RUNNING THE ASSEMBLER

The RAG SOFTWARE AEMS Macro Assembler requires an Asgard Expanded Memory System card and the AEMS system software. The Assembler is run from filename AMAC1.

Assembler Input Screen

The Assembler screen is divided into three parts. The second part contains six input fields labeled: "Printer", "Macros", "Options", "Date", "Source", and "Object". The assembler returns to the input fields after every assembly allowing you to do a batch of assemblies at one time.

The "Printer" field specifies the name of the listing file to be used if any listing options are selected.

The "Macros" field specifies the name of the macro library file to be read prior to assembling the source. The macro file name may be specified as an asterisk to indicate that the same macros are to be used as for the previous assembly in the batch. A null entry (i.e. all blank) indicates that no macro library file is required for this assembly. The "Options" field specifies the options to be used for this assembly. The options are specified as a sequence of one letter option codes. Except for the "&" option, the codes may be entered in any order. The option codes are:

C - output the object file in compressed format.
E - output error messages to the printer file. If any other listing.
option is selected then this option is assumed.
F - show full assembled data in the listing. If this option is not specified then a maximum of 6 bytes of data is shown on the listing (a single line) for BYTE, DATA, FLOAT, TEXT and STRI statements. The full data is, of course, always assembled into the object file.
G - include macro generated statements in the listing. The macro generated statements are identified in the listing by a plus sign following the statement number.
L - produce a listing of source statements.
M - include macro directives in the listing. Macro directives are identified in the listing by a minus sign preceding the statement.
R - define the register symbols R0 - R15.
S - produce a symbol table listing.
X - produce a cross-reference listing.
Y - produce a cross-reference listing of all symbols except the register symbols.
& - set the system macro symbol &S0 to the remainder of the options field.

The "Date" field can be used to enter up to 9 characters of identification for this assembly. The information entered appears in three places during the assembly: on the listing heading line, on the last record of the object file, and as the value of the &S4 system macro symbol. As the field name suggests, today's date is probably the most useful identification, then both the listing and the object file will be dated.

The "Source" field specifies the name of the file containing your assembler source statements.

The "Object" field specifies the name of the file into which your object program is to be written.

The "Source" and "Object" fields are the only fields that must be specified.

During data entry the function keys perform as ordinarily defined by TI. In particular:

```
FCTN 1 (DEL)    Delete character,
FCTN 2 (INS)    Insert character,
FCTN 3 (ERASE)  Erase to end of field,
FCTN 4 (CLEAR)  Erase entire field,
FCTN 5 (BEGIN)  Begin execution of function,
FCTN 6 (PROCD)  Proceed with function,
FCTN 7 (AID)    File name selection from directory,
FCTN 8 (REDO)   Redo data in field,
FCTN 9 (BACK)   Terminate function,
```

```
FCTN = (QUIT)   Quit the Assembler,
ENTER           Move cursor to next field,
FCTN E (Up)     Move cursor to previous field,
FCTN X (Down)   Move cursor to next field,
FCTN S (Left)   Move cursor left,
FCTN D (Right)  Move cursor right.
```

When ENTER or FCTN X is pressed for the last field, the assembly begins.

In the Directory Aid dialog box, the disk device name is entered in the usual way, including hard disk sub-directories, with or without the trailing period. The file name is selected by scrolling the cursor up and down and then pressing ENTER. The selected device and file name are placed in the input field. Pressing BACK cancels the directory aid without a selection.

The dialog box may display an error message. Pressing any key clears the message and returns to the input field.

The Object File

The object file output by the Assembler contains identification so that the source, object and listing can be tied together. First, the source file name is placed on the first object record, which also contains the IDT information. Second, the last record of the object file (the colon record) contains the Assembler name/version, and the "Date" information that was entered.

The machine language object file created by the Assembler is a FIXED 80 file containing "tagged object" in either compressed or uncompressed format. The tagged object contains fields beginning with a tag followed by data. The tag identifies the type of data in the field. The first tag in an object program is either "0", which indicates that the object code is in uncompressed format, or ">01", which indicates that the object code is in compressed format.

In uncompressed format, each two byte data field is represented by four hexadecimal digits. In compressed format, each two byte data field contains the actual two bytes of data.

The following table lists the tags used and gives the content of the data field that follows the tag.

```
TAG      CONTENTS OF DATA FIELD.
-------------------------------------------------------.
>01      2 Bytes, size of relocatable portion of program.
         8 Characters, text from the IDT Directive.
 0       2 Bytes, size of relocatable portion of program.
         8 Characters, text from the IDT Directive.
 1       2 Bytes, Entry point in absolute code.
 2       2 Bytes, Entry point in relocatable code.
```

```
 3       2 Bytes, REF chain in relocatable code.
         6 Characters, REF name.
 4       2 Bytes, REF chain in absolute code.
         6 Characters, REF name.
 5       2 Bytes, DEF value in relocatable code.
         6 Characters, DEF name.
 6       2 Bytes, DEF value in absolute code.
         6 Characters, DEF name.
 9       2 Bytes, absolute location counter.
 A       2 Bytes, relocatable location counter.
 B       2 Bytes, absolute code.
 C       2 Bytes, relocatable code.
 F       No data, end of object record.
 G       2 Bytes, Complex relocatable value.
         6 Characters, REF name.
 H         Not implemented.
 I       2 Bytes, COMMON size.
         6 Characters, COMMON name.
```

The last record of an object file begins with a colon. It contains only information on the Assembler name, version and the "Date" information.

Note that if the object file contains a tag "G", "H" or "I" then the object file cannot be loaded directly using any TI object loaders. In this case the object file must be processed by the AEMS Linker. If a tag "H" or "page number address" is used it implies that the program can only run on a TI 99/4A equipped with an AEMS card, and that it must be loaded with the AEMS program loader.

Table Sizes

The Assembler builds several tables during an assembly. Each of the tables is described below.

The Symbol Table is used to save all symbols, macro names and operation codes defined in the assembly. Each entry in the table is 12 bytes. Two entries are made for each REF symbol. The symbol table is up to 24K in size or 2047 entries.

The Cross Reference Table is used to accumulate the references to each symbol in the assembly. The Y option excludes the register symbols from cross referencing. Each entry in the table is 4 bytes. The cross reference table is up to 24K in size giving a maximum of 6,140 entries.

The Macro Definition Table is used to save all macro definitions encountered in the macro file or the source file. All the lines of the macro definitions are stored as variable length strings with the length byte preceding the statement. The macro table is up to 24K in size or large enough to hold the equivalent to a 96 sector macro file.

The OBJREC Directive Table is used to save the BEFORE text during pass 1 and to save the AFTER text during pass 2. Each text entry is stored as a variable length string with the length byte preceding the text. The table size is 6,144 bytes, providing enough space to hold about 150 40-byte object records.

DIAGNOSING ASSEMBLER FAILURES

Every attempt has been made to insure that the Assembler has no bugs, however, in every complex program the possibility of bugs always exists. In addition, bugs may also exist in the operating system facilities used by the Assembler. There are three possible type of failures.

1. The Assembler completes normally, but some instruction or data was assembled incorrectly.
2. The Assembler completed, but the listing or disk files were incomplete or in error.
3. The Assembler did not complete and/or the system required rebooting.

In all three cases, the first thing to do is to correct all source errors that were found by the Assembler. The Assembler can really only correctly assemble correct programs, although it tries to diagnose incorrect statements. As the old saying goes: "Garbage in equals Garbage out".

In the first case above, you should be sure that you understand what should be assembled. The language supported is fully described in the "Assembler Language Reference" document supplied with the Assembler. The compatibility statement made in that document is for information only and not as a definition of the language supported. If the EQUV assembler directive and/or macro definitions which test which pass the Assembler is in are used, these should be checked carefully.

In the second and third cases above, either the Assembler or the operating system may be suspect. In these cases, you should reduce the dependency of the assembler on the operating system. This can be done by not using any of the listing options.

Finally, all bugs discovered as well as any usability problems should be reported to:

RAG SOFTWARE
R. A. Green.
1032 Chantenay Dr.
Gloucester, Ont.
CANADA K1C 2K9

If possible, a disk with the source program that can not be assembled should be sent. This will make finding the bug easier.

If a disk is sent, the material on the disk will &only be used for finding the bug and will be returned with a corrected version of the assembler.

TECHNICAL MATERIAL

The documentation distributed with the Assembler consists of three manuals:

AMACLR/D - Assembler Language Reference
AMACMR/D - Macro Reference and Tutorial
AMAC99/D - 9900 CPU Reference

which can be printed with the TI Writer Formatter.

More information on the hardware and programming the computer can be obtained in other publications such as those listed below.

1. TMS 9900 16-Bit Microcomputer
   Preliminary Data Manual
   Texas Instruments, Inc., 1981
2. Texas Instruments Home Computer
   Editor/Assembler
   Texas Instruments, Inc., 1981
3. Introduction to Assembly Language for the TI Home Computer
   Ralph Molesworth, 1983
   Steve Davis Publishing
   P.O. Box 190831
   Dallas, Texas 75219
   ISBN 0-911061-01-0
4. Learning TI 99/4A Home Computer
   Assembly Language Programming
   Ira Mc Comic, 1984
   Wordware Publishing, Inc.
   Plano, Texas 75074
   ISBN 0-13-527862-7
5. Fundamentals of TI 99/4A Assembly Language
   M. S. Morley, 1984
   Tab Books, Inc.
   Blue Ridge Summit, PA 17214
   ISBN 0-8306-1722-1

TAILORING THE ASSEMBLER

The Assembler can be tailored in two ways to meet your requirements. First, the initial contents of the input screen areas can be specified and second, printer setup data can be provided. This tailoring is done by applying patches to the Assembler using the Z-AEMSPAT program supplied on the Assembler distribution disks.

The patches are easy to make. There are two model patch files on the distribution disk one for the initial input screen contents and one for the printer setup data. The two model files are those that could be used to restore the Assembler to its "distribution" state.

You simply modify either or both files with any editor and then apply your patches by running the Z-AEMSPAT program, specifying the names of your patch files.

The names of the patch files on the distribution disk are:

SRCDIST	Input screen contents as distributed.
PRTDIST	Generic printer setup data as distributed.
PRT10X	Star Gemini 10X printer setup data.
PRTNX1000	Star NX1000 printer setup data.