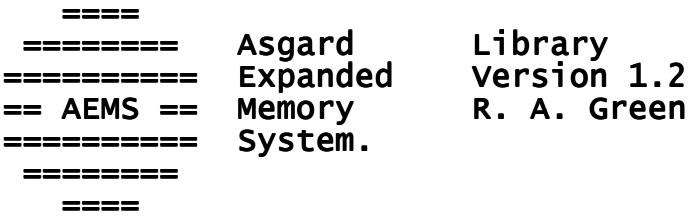
ASGARD EXPANDED MEMORY SYSTEM

AEMS SUBROUTINE LIBRARY

AEMS LIBRARY MANAGER



The AEMS Subroutine Library contains the Assembler Language routines necessary for programs to interface with the AEMS and the standard TI operating system. It also contains routines to perform often used functions, making programming easier and quicker.

The AEMS Library Manager allows you to build and maintain subroutine libraries.

CONTENTS

INTRODUCTION 1
DISK CONTENTS 1
AVAILABILITY 1
THE AEMS LIBRARY MANAGER . . . 2
RUNNING THE LIBRARY MANAGER . . 2
LIBRARY ROUTINE DESCRIPTIONS . . 3
  AEXIT 4
  AFREE 4
  ALLOC 4
  ALOAD 5
  AMOVE 5
  AVDPS 6
  AVMBR 6
  AVMBW 6
  C2F 7
  DSRLNK 7
  F2C 7
  GPLLNK 8
  H2I 8
  I2H 8
  KGET 9
  KSCAN 9
  KWAIT 9
  MBLNK 10
  MMOVE 10
  SC2I 10
  SI2C 11
  UC2I 11
  UI2C 11
  VMBR 12
  VMBW 12
  VSBR 12
  VSBW 13
  VWTR 13
  XMLLNK 13
APPENDIX A. VDP Tables 14

INTRODUCTION

The AEMS Subroutine Library contains assembler language subroutines necessary to interface to the AEMS paged memory and to the TI operating system. It also contains routines to perform repetitive functions to make programming easier and quicker. All routines are contained in a Linker Library for inclusion into your programs by the AEMS Linker. All routines are relocatable and may be loaded anywhere in memory. There are two versions or entry points to most routines. One entry for calling with parameters in registers; the other for calling with inline parameter lists.

The RAG Software AEMS Library Manager is a tool that makes building and maintaining your own libraries easy.

DISK CONTENTS

The distribution disk is a double sided single density disk containing the following files.

AEMSLIB	The AEMS Subroutine Library.
ALIBDOC	The documentation for the AEMS Subroutines and the Library Manager, to be printed by the TI WRITER formatter.
ALIBDOC1	Continuation of the documentation.
ALMAN1	The AEMS Library Manager program.
ALMAN2	The AEMS Library Manager continuation.
TIWFORM	A version of the TI Writer formatter.
ZAEMSPAT	Public Domain program file patcher.
ZAEMSPAT/D	Documentation for AEMSPAT.

AVAILABILITY

The AEMS Subroutine Library object code is PUBLIC DOMAIN. The library routines may be used and distributed freely. The routines were written to support the AEMS by R. A. Green and Joe Delekto.

The AEMS Library Manager is a FAIRWARE program by:

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

If you have any suggestions for improvements or have found any bugs in the AEMS Subroutine Library or the AEMS Library Manager please forward them to the above address.

The Assembler source for the routines in the AEMS Subroutine Library and the AEMS resident system routines, is available on a restricted basis for a $10 copying fee. Availability is restricted to those who are developing programs for the AEMS and require an intimate knowledge of how he routines operate. Those who receive the source must agree not to distribute it further without consent of RAG Software.

THE AEMS LIBRARY MANAGER

The AEMS Library Manager is used to build and maintain Linker Libraries. It allows the following functions.

Add new object file to a library.
Delete object file from a library.
Extract an object file from a library.
Print a listing of the contents of a library.

The object in a library may be either compressed or uncompressed.

RUNNING THE AEMS LIBRARY MANAGER

The RAG Software AEMS Library Manager must be run using the AEMS System. The file name of the Library Manager is ALMAN1.

The title screen for the Library Manager lists all the functions that can be performed. The library manager operates like a special purpose editor. The library manager functions are shown below.

[L]oad library
[S]ave library
[A]dd object to library
[D]elete entry from library
[E]xtract entry from library
[P]rint library contents
[Q]uit library manager
FCTN E Scroll Up
FCTN X Scroll Down
CTRL E Page Up
CTRL X Page Down

The function list can be redisplayed by pressing FCTN 7 (AID). Once a library has been loaded or object has been added to build a library, the screen shows the contents of the library. Each object in the library is numbered (initially by tens). The display shows the object number and the entry points contained in the object. The Add, Delete and Extract functions refer to the object numbers. The display can be scrolled or paged up and down to show the entire contents of the library.

The functions are selected by pressing a single key. Once selected, the function will open a dialog box requesting necessary information. 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 Library Manager,
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 function is executed.

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

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.

If the library has been modified and not saved when Quit is selected, a dialog box will appear giving you the option of not quitting.

LIBRARY ROUTINE DESCRIPTIONS

All of the routines in AEMSLIB are relocatable compressed object. Many of the routines use the "system" areas of RAMPAD. These system areas are used in the "defined" way and should not interfere with any other programs that use them "correctly".

The routines that require the AEMS are only small interface routines to parts of the AEMS resident code which is loaded when the AEMS is booted up. A specialized linkage to the resident code is used. This specialized interface technique provides fast execution of the routines and at the same time saves memory space in the 4A's 32K address space for your program and data. These routines use RAMPAD for a workspace and parameter passing when the system pages are mapped in. In particular, the AEMS routines use:

GPLWSP	>83E0	Workspace registers
GPLV1	>83E0	Returned value 1
GPLV2	>83E2	Returned value 2
GPLP1	>83D0	Parameter value 1
GPLP2	>83D2	Parameter value 2
GPLP3	>83D8	Parameter value 3
GPLP4	>83DA	Parameter value 4
GPLP5	>83DC	Parameter value 5

All routines may be called with interrupts enabled.

AEXIT

Function: Exit from program to AEMS main menu.

Restrictions: Requires AEMS.

Entries: AEXITI - Inline parameter list,
AEXITR - Register parameters.

Parameters: P1 R0 - Mode = 0 Normal exit,
= 1 Retain program in memory.

Results: The program is terminated le for other and either all allocated pages are freed or are retained depending upon the "mode".

AFREE

Function: Free previously allocated pages.

Restrictions: Requires AEMS.

Entries: AFREEI - Inline parameter list,
AFREER - Register parameters.

Parameters: P1 R0 - First page number.
P2 R1 - Number of pages to free.

Results: The pages are made available for other allocation requests.

ALLOC

Function: Allocate one or more pages of memory for program use.

Restrictions: Requires AEMS.

Entries: ALLOCI - Inline parameter list,
ALLOCR - Register parameters.

Parameters: P1 R0 - Number of pages to allocate.

Results: The number of the first page allocated is returned in R0. If there is insufficient memory for the allocation request, the number of the first page is set to zero.

Notes: It is good practice for a program to free all pages it allocates.

ALOAD

Function: Load a program into memory.

Restrictions: Requires AEMS.

Entries:
```
ALOADI   - Inline parameter list,
ALOADR   - Register parameters.
```

Parameters:
```
P1  R0   - Type of load, code value,
           = 0 normal load,
           else load and retain,
>8300    - 28 Character file name,
>831C    - 15 Character menu text for retain.
```

Results: A code is returned in R0 indicating how the load was processed.
```
         R0 = 0 Load OK,
         R0 = 1 I/O Error during load,
         R0 = 2 Invalid program file header,
         R0 = 3 Memory full.
```
In addition, data about the loaded program is returned in RAMPAD as follows.
```
>8300 16  Mapper register values for the program.
          Unused registers are set to >FFFF.
>8320 Execution start address.
>8322 Number of allocated pages.
>8324 Number of first allocated page.
```

Note: A retained program is added to the AEMS menu using the text supplied.

AMOVE

Function: Move data in memory.

Restrictions: Requires AEMS.

Entries:
```
AMOVEI   - Inline parameter list,
AMOVER   - Register parameters.
```

Parameters:
```
P1   R0  - From page number,
P2   R1  - From displacement,
P3   R2  - To page number,
P4   R3  - To displacement,
P5   R4  - Length of data to move.
```

Results: The data is moved mapping in pages as necessary, returning with the memory map in the original state.

AVDPS

Function: Setup VDP tables for various modes.

Restrictions: Requires AEMS.

Entries:
```
AVDPSI   - Inline parameter list,
AVDPSR   - Register parameters.
```

Parameters:
```
P1   R0  - Mode:
               = 0 GPL setup, = 1 XB setup,
               = 2 E/A setup, = 3 Text mode.
P2   R1  - 1st Byte, Character set:
               = 0 Normal,    = 1 True lower case.
           2nd Byte, Screen on return:
               = 0 Screen on, = 1 Screen off.
```

Results: Details of VDP table placement for the various modes is given in Appendix A.

AVMBR

Function: VDP multiple byte read.

Restrictions: Requires AEMS.

Entries:
```
AVMBRI   - Inline parameter list,
AVMBRR   - Register parameters.
```

Parameters:
```
P1   R0  - VDP address,
P2   R1  - To page number,
P3   R2  - Displacement in to page,
P4   R3  - Number of bytes to read.
```

Results: The data is read from the VDP into memory, mapping in pages as necessary, returning with the memory map in the original state.

AVMBW

Function: VDP multiple byte write.

Restrictions: Requires AEMS.

Entries:
```
AVMBWI   - Inline parameter list,
AVMBWR   - Register parameters.
```

Parameters:
```
P1   R0  - VDP address,
P2   R1  - From page number,
P3   R2  - Displacement in from page,
P4   R3  - Number of bytes to write.
```

Results: The data is written to the VDP from memory, mapping in pages as necessary, returning with the memory map in the original state.

C2F

Function: Convert from character to floating point.

Restrictions: None.

Entries: C2F - Inline parameter list.

Parameters: P1 - Address of character string,
P2 - VDP address of work area.

Results: The floating point number is returned in FAC (>834A).

Note: The character string consists of a length byte followed by the characters.

DSRLNK

Function: Link to Device Service ROM.

Restrictions: None.

Entries: DSRLNK - Inline parameter list.

Parameters: P1 - 8 I/O operation.
16 Subroutine call.
>8356 - VDP address of PAB or subroutine name.

Results: EQ status is set for invalid name.

Note: Equivalent to the E/A routine.

F2C

Function: Convert floating point number to character.

Restrictions: None.

Entries: F2C - Inline parameter list.

Parameters: P1 - Address for result string,
>834A - Floating point number.
>8355 - Type of result.
0 = Same as BASIC.
x = Fixed point mode.
number of significant digits.
>8357 - Fixed point mode number.
digits to right of decimal point.

Results: The string returned at P1 consists of a length byte followed by the data.

Note: The character string returned consists of a length byte followed by the characters. F2C uses the GPL routine CNS (>14) which uses 26 bytes of VDP memory at address >03C0. In fixed point mode, if the number cannot be formatted into the field specified the field will contain one or more upper case "E".

GPLLNK

Function: Link to GPL Routine.

Restrictions: None.

Entries: GPLLNK - Inline parameter list.

Parameters: P1 - GPL routine GROM address.

Results: As set by the various GPL routines.

Note: Equivalent to the E/A routine.

H2I

Function: Convert from hexadecimal characters to integer.

Restrictions: None.

Entries: H2I - Inline parameter list.

Parameters: P1 - Character string.

Results: The integer value is returned in R0,
EQ status is set on error.

Note: The character string consists of a length byte followed by the characters.

I2H

Function: Convert from integer to hexadecimal characters.

Restrictions: None.

Entries: I2H - Inline parameter list.

Parameters: P1 - Character string,
R0 - Integer value to be converted.

Results: Exactly 4 hexadecimal digits will be returned in the character string.

KGET

Function: Get next key stroke with auto-repeat.

Restrictions: None.

Entries: KGET - Inline parameter list.

Parameters: P1 - Keyboard number.

Results: The key value is returned in R0 with the second byte of R0 set to zero.

KSCAN

Function: Scan the keyboard.

Restrictions: None.

Entries: KSCAN.

Parameters: >8374 - Keyboard number.

Results: >837C - >20 bit set on new key press,
>8375 - Key value.

Note: Equivalent to the E/A routine.

KWAIT

Function: Wait for any key press.

Restrictions: None.

Entries: KWAIT - Inline parameter list.

Parameters: P1 - Keyboard number.

Results: The key value is returned in R0 with the second byte of R0 set to zero.

MBLNK

Function: Set memory area to blanks.

Restrictions: None.

Entries: MBLNK - Inline parameter list.

Parameters: P1 - Memory start address,
P2 - Number of bytes to blank.

Results: The memory area is blanked.

MMOVE

Function: Move memory block.

Restrictions: None.

Entries: MMOVE - Inline parameter list.

Parameters: P1 - From memory address,
P2 - To memory address,
P3 - Length of data to move.

Results: The memory area is moved.

SC2I

Function: Signed character to integer conversion.

Restrictions: None.

Entries: SC2I - Inline parameter list.

Parameters: P1 - Character string.

Results: Integer value returned in R0,
EQ status set if error.

Note: The character string consists of a length byte followed by the characters.

SI2C

Function: Signed integer to character conversion.

Restrictions: None.

Entries: SI2C - Inline parameter list,
SI2CZ - Inline parameter list, leading zeros.

Parameters: R0 - Integer value to convert,
P1 - Character string.

Results: The character string returned consists of a length byte followed by the characters. The area for the string must be at least 7 bytes long.

UC2I

Function: Unsigned character to integer conversion.

Restrictions: None.

Entries: UC2I - Inline parameter list.

Parameters: P1 - Character string.

Results: Integer value returned in R0,
EQ status set if error.

Note: The character string consists of a length byte followed by the characters.

UI2C

Function: Unsigned integer to character conversion.

Restrictions: None.

Entries: UI2C - Inline parameter list,
UI2CZ - Inline parameter list, leading zeros.

Parameters: R0 - Integer value to convert,
P1 - Character string.

Results: The character string returned consists of a length byte followed by the characters. The area for the string must be at least 6 bytes long.

VMBR

Function: VDP multiple byte read.

Restrictions: None.

Entries: VMBRI - Inline parameter list,
VMBR - Register parameters.

Parameters: P1 R0 - VDP address,
P2 R1 - To memory address,
P3 R2 - Length of data to read.

Results: No return values.

Note: VMBR is equivalent to the E/A routine.

VMBW

Function: VDP multiple byte write.

Restrictions: None.

Entries: VMBWI - Inline parameter list,
VMBW - Register parameters.

Parameters: P1 R0 - VDP address,
P2 R1 - From memory address,
P3 R2 - Length of data to write.

Results: No return values.

Note: VMBW is equivalent to the E/A routine.

VSBR

Function: VDP single byte read.

Restrictions: None.

Entries: VSBRI - Inline parameter list,
VSBR - Register parameters.

Parameters: P1 R0 - VDP address,

Results: The byte read is returned in the first byte of R1.

Note: VSBR is equivalent to the E/A routine.

VSBW

Function: VDP single byte write.

Restrictions: None.

Entries: VSBWI - Inline parameter list,
VSBW - Register parameters.

Parameters: P1 R0 - VDP address,
P2 R1 - Byte to write.

Results: No return values.

Note: SBW is equivalent to the E/A routine.

VWTR

Function: Write to VDP register.

Restrictions: None.

Entries: VWTRI - Inline parameter list,
VWTR - Register parameters.

Parameters: P1 R0 - From memory address,

Results: No return values.

Note: VWTR is equivalent to the E/A routine.

XMLLNK

Function: Link to assembler language routine in ROM.

Restrictions: None.

Entries: XMLLNK - Inline parameter list,

Parameters: P1 - table number, routine number.

Results: The called routine may return results.

Note: Equivalent to the E/A routine.

APPENDIX A. VDP Tables

Mode 0. GPL Setup

```
VDP R0 = >00 Bit Map off
VDP R1 = >E0 16K, Screen, Interrupt
VDP R2 = >00 Screen Image at >0000
VDP R3 = >0E Colour Table at >0380
VDP R4 = >01 Pattern Table at >0800
VDP R5 = >06 Sprite Attributes at >0300
VDP R6 = >00 Sprite Patterns at >0000
VDP R7 = >17 Black/Cyan
```

Mode 1. Extended Basic Setup

```
VDP R0 = >00 Bit Map off
VDP R1 = >E0 16K, Screen, Interrupt
VDP R2 = >00 Screen Image at >0000
VDP R3 = >20 Colour Table at >0800
VDP R4 = >00 Pattern Table at >0000
VDP R5 = >06 Sprite Attributes at >0300
VDP R6 = >00 Sprite Patterns at >0000
VDP R7 = >17 Black/Cyan
```

Mode 2. Editor/Assembler Setup

```
VDP R0 = >00 Bit Map off.
VDP R1 = >E0 16K, Screen, Interrupt
VDP R2 = >00 Screen Image at >0000
VDP R3 = >0E Colour Table at >0380
VDP R4 = >01 Pattern Table at >0800
VDP R5 = >06 Sprite Attributes at >0300
VDP R6 = >00 Sprite Patterns at >0000
VDP R7 = >F5 White/Light Blue
```

Mode 3. Text Mode Setup

```
VDP R0 = >00 Bit Map off.
VDP R1 = >F0 16K, Screen, Interrupt, Text
VDP R2 = >00 Screen Image at >0000
VDP R3 = >0E Unused.  Colour Table
VDP R4 = >01 Pattern Table at >0800
VDP R5 = >06 Unused.  Sprite Attributes
VDP R6 = >00 Unused.  Sprite Patterns
VDP R7 = >17 Black/Cyan
```