# General Notice

When using this document, keep the following in mind:

1. This document is confidential. By accepting this document you acknowledge that you are bound by the terms set forth in the non-disclosure and confidentiality agreement signed separately and /in the possession of SEGA. If you have not signed such a non-disclosure agreement, please contact SEGA immediately and return this document to SEGA.

2. This document may include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new versions of the document. SEGA may make improvements and/or changes in the product(s) and/or the program(s) described in this document at any time.

3. No one is permitted to reproduce or duplicate, in any form, the whole or part of this document without SEGA'S written permission. Request for copies of this document and for technical information about SEGA products must be made to your authorized SEGA Technical Services representative.

4. No license is granted by implication or otherwise under any patents, copyrights, trademarks, or other intellectual property rights of Sega Enterprises, Ltd., Sega of America, Inc., or any third party.

5. Software, circuitry, and other examples described herein are meant merely to indicate the characteristics and performance of SEGA's products. SEGA assumes no responsibility for any intellectual property claims or other problems that may result from applications based on the examples describe herein.

6. It is possible that this document may contain reference to, or information about, SEGA products (development hardware/ software) or services that are not provided in countries other than Japan. Such references/ information must not be construed to mean that SEGA intends to provide such SEGA products or services in countries other than Japan. Any reference of a SEGA licensed product/ program in this document is not intended to state or imply that you can use only SEGA's licensed products/programs. Any functionally equivalent hardware/software can be used instead.

7. SEGA will not be held responsible for any damage to the user that may result from accidents or any other reasons during operation of the user's equipment, or programs according to this document.

NOTE: A reader's comment/correction form is provided with this document. Please address comments to :

SEGA of America, Inc., Technical Publications Group
150 Shoreline Drive, Redwood City, CA 94065

SEGA may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

(02/09/96-003)

SEGA OF AMERICA, INC.
Consumer Products Division

# dAsms
# User's Manual

Doc. # ST-228-R1-030596

# Reader Correction/Comment Sheet

## *Keep Us Updated!*

If you should come across any incorrect or outdated information while reading through the attached document, or have any questions or comments, please let us know so that we can make the required changes in subsequent revisions of the document. Simply fill out all relevant information below and return this form to the Developer Technical Support Manager at the address below. Please make more copies of this form if more space is needed. Thank you for your cooperation!

***General Information***

**Your Name:** _____ **Phone:** _____

**Document Number:** ST-228-R1-030596 **Date:** _____

**Document Name:** dAsms User's Manual

***Corrections***

| Chapter | Page | Correction |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

***Questions/Comments***

***Send Your Corrections/Comments to:***

**Fax:** 415.802.1717
Attention: Evelyn Merritt
Developer Technical Support

**E-Mail:** **dts@segaoa.com**

**Mail:** SEGA OF AMERICA, INC.
Attention: Evelyn Merritt
Developer Technical Support
150 Shoreline Drive
Redwood City, California 94065

# Table of Contents


dAsms User's Manual ..... 5
Using dAsms ..... 6
    A Basic Work Session ..... 6
Menus ..... 7
    File Menu ..... 7
    Edit Menu ..... 7
    Process Menu ..... 8
    Option Menu ..... 8
DSP Program Data Structure (Data Format) ..... 9
    Comments ..... 9
Coefficient and Address Symbol Definitions ..... 10
    Definition Format ..... 10
    Number of User-Definable Symbols ..... 10
    Reserved Symbols ..... 10
    Notation Mode in Coefficient/Address Definition Part of Source Code ..... 11
    Legal Range and Format for Initial Coefficient/Address Symbol Values ..... 12
Program Description ..... 14
    Command List ..... 14
Parameters ..... 15
    Internal Registers ..... 15
    Internal RAM ..... 15
    Memory Access Parameters ..... 15
Programming Syntax ..... 16
    Specifying External Memory Read and Load to MEMS (LDI) ..... 16
    Load to YREG (LDY) ..... 16
    Load to ADREG (LDA) ..... 16
    Multiply, Add ..... 17
    Store ..... 18
Reference 1: dAsms Executable File Format ..... 19
Reference 2: dAsms Programming Guide ..... 21
Creating DSP Programs ..... 23
    1. Coefficient Symbol Definition (=COEF) ..... 23
    2. Address Constant Symbol Definition (=ADRS) ..... 24
    3. Program Description (=PROG) ..... 25
Reference 3: Error and Warning Messages ..... 29
    1. Source Code Configuration Errors/Warnings ..... 29
    2. Coefficient/Address Constant Definition Errors ..... 29
    3. Program Description Errors ..... 30
Index ..... 32

# dAsms User's Manual

This manual describes how to use the **dAsms** SCSP/DSP assembler and its assembly language.

# Using dAsms

## A Basic Work Session

The following is a sample work session for programming the SCSP/DSP using **dAsms**. It is assumed that all the necessary settings to run **dAsms** have already been made.

1. Start **dAsms**
2. Write DSP source code using the built-in text editor or load code prepared using another editor.
3. Run the assembler to compile the code.
4. Download the compiled code into Sega Saturn's memory.

# Menus

## File Menu

### New
Opens a new source file window.

### Open
Opens a DSP microprogram source code file selected from a standard Macintosh file dialog box.

### Save
Saves the edited file under the same filename.

### Save As...
Saves the edited file under a different filename. Specify a new filename when prompted.

### Revert To Saved
Returns the source file being edited to the state it was in prior to editing.

### Close
Closes the currently open file. A warning will be displayed if there are any changes that have not been saved.

### Quit
Quits **dAsms**

**Note:** The file extension `.USC` must be attached to source code files. The **Assemble...** command (described later) will not work on source code files without this extension.

## Edit Menu

### Undo, Cut, Copy, Paste, Select All
These commands perform the standard Macintosh editing functions on the source code text.

## Process Menu

### Assemble...

Assembles the DSP source code contained in the active window. You have the option of entering a different filename for the assembled output file. Otherwise, the extension `.EXC` is added to the filename of the source code file as default .

### Download...

Downloads an assembled DSP object file to the SCSP via SCSI. Select the download file from the file dialog box and execute. If the current edit file is already in a executable format (i.e. the file is assembled), its filename is used as the default download filename.

## Option Menu

### SCSI ID

**dAsms**automatically detects the SCSI ID of the development target when it firsts performs a download operation. That SCSI ID is subsequently used for data communications between the host and target. This **SCSI ID** option does not normally need to be used unless your hardware setup changes.

**CAUTION**: The SCSI ID of the development target must be different from the ID of any other SCSI device connected in the chain. Incorrectly set SCSI IDs may cause loss of data or hardware problems on the SCSI device.

## DSP Program Data Structure (Data Format)

The structure and format of **dAsms** source code are shown below.

| Data Structure | Description |
|---|---|
| **#COEF**<br>Coefficient Symbol Definition | This label denotes the start of the coefficient symbol definition. |
| **#ADRS**<br>Address Symbol Definition | This label denotes the start of the address symbol definition |
| **#PROG**<br>Program Description | This label denotes the start of program description |
| #**END** | This label denotes the end of program description |

### Comments

- Comments can be inserted in the source code by using the apostrophe character (') to start them. This causes **dAsms** to ignore everything after the apostrophe character to the end of the line (line feed).
- Comments can be placed anywhere in the source code.

## Coefficient andAddress Symbol Definitions

### Definition Format

```
(symbol name) = (initial value)
```

`symbol name`: Variable name used for coefficient/address data in the DSP code. 1 byte alpha-numeric character string of 15 characters or less.

`initial value`: Initial value given for a variable denoted by the symbol name above. The value is expressed by one of various notation methods described in this document.

**Note:** One symbol can be defined in one line of the definition expression. An initial value must be given to each symbol.

### Number of User-Definable Symbols

- Coefficient symbols 63
- Address symbols 32

### Reserved Symbols

The coefficient symbol `ZERO` is pre-defined; the user may not define a symbol with the same name. The initial value given to this symbol is `0` and placed at the beginning of the coefficient data section of the executable DSP program. User-defined coefficient symbols are placed in the order that they are defined, starting from the second coefficient data in the coefficient data section.

Although the reserved symbol `ZERO` can be used in the user program section without its pre-definition in the coefficient definition section, the value must not be modified by any means (e.g., within an DSP program or in the DSP hardware).

## Notation Mode in Coefficient/Address Definition Part of Source Code

The following notation modes are permitted in the coefficient/address definition section.

| Definition | Notation Mode | Identifier | Example | Conversion Method | Conversion Result |
|---|---|---|---|---|---|
| #COEF | Hexadecimal | &H | &H0FFF | No conversion | &H0FFF |
| | Decimal | None | 123 | Dec to Hex | &H007B |
| | % | % | %50 | *1 | &H0800 |
| | Fractional | . | 0.25 | *2 | &H0400 |
| #ADRS | Hexadecimal | &H | &H8000 | No conversion | &H8000 |
| | Decimal | None | 123 | Dec to Hex | &H007B |
| | Milliseconds | ms | ms300.0 | *3 | &H33AE |

**Conversion Method** shown above denotes the method used to perform conversion from a value in each notation mode to actual data. The formulas for conversion methods *1, *2 and *3 are as follows.

```
*1   (Coefficient) = 4095 * (NotationValue)/100
*2   (Coefficient) = 4096 * (NotationValue)
*3   (AddressDefinition) = 44100 * (NotationValue)/1000
```

**Note:** Floating point calculation is performed for *1, *2 and *3. The result is rounded off to the nearest zero.

The format (in an EXC file) of the converted actual data is shown below.

| Definition Section | Actual Data Format |
|---|---|
| #COEF | 13-bit two's complement format hexadecimal |
| #ADRS | 16-bit linear format hexadecimal |

**Note:** Note the difference between the constants of formulas *1 and *2 (i.e., 409**5** and 409**6**).

## Legal Range and Format for Initial Coefficient/Address Symbol Values

| Definition | Notation Mode | Range | Format |
|---|---|---|---|
| #COEF | Hexadecimal | 1000 to 0FFF[hex] | Signed hexadecimal integer (13 bit wide) |
| | Decimal | -4096 to +4095[dec] | Signed decimal integer |
| | % | -100 to +100[%] | Signed decimal integer |
| | Fractional | -1.0 to +0.99975[dec] | Signed decimal fraction (1 digit integer component, 5 digits fractional component maximum) |
| #ADRS | Hexadecimal | 0 to FFFF[hex] | Unsigned hexadecimal integer (16 bit wide) |
| | Decimal | 0 to 65535[dec] | Unsigned decimal integer |
| | Milliseconds | 0.0 to 1486.0[ms] | Unsigned decimal fractional number (4 digits integer component, 1 digit fractional component maximum) |

**Example:** The following expressions are possible in the coefficient/address definition section of the source code.

```
#COEF
CoefA = &H0FFF
CoefB = 123
CoefC = %50
CoefD = 0.25

#ADRS
AdrsA = &H8000
AdrsB = 123
AdrsC = ms300.0
```

# Program Description

## Command List

| Command | Function |
|---|---|
| @ | Start a multiplication or addition description |
| * | Multiply |
| + | Add |
| - | Subtract |
| > | Store (end of multiplication or addition description) |
| ( | Left parenthesis |
| ) | Right parenthesis |
| LDI | Memory access and load to MEMS |
| LDY | Load from INPUTS to Y register |
| LDA | Load from INPUTS to A register |

# Parameters

## Internal Registers

| Symbol | Description |
|---|---|
| REG | Denotes previous multiplication or addition results. |
| YREGH | Denotes section of (see explanation of multiply and add syntax) of YREG. |
| YREGL | Denotes section of (see explanation of multiply and add syntax) of YREG. |
| FREG | (Refer to explanation of store syntax). |
| ADREG | Address modify component(refer to explanation of store syntax). |

## Internal RAM

| Symbol | Description |
|---|---|
| MEMS00~1F | External memory load data area. |
| MIXS00~0F | Sound generator data area. |
| TEMP00~7F | Temporary storage area for multiply and add results. |
| EFREG00~0F | Processed data area. |
| EXTS00~01 | External expansion input data area. |

## Memory Access Parameters

| Symbol | Description |
|---|---|
| MR[…] | Read from memory. |
| MW[…] | Write to memory. |

**Note:** The following memory access elements are noted within the brackets according to the rules described elsewhere in this document.

- User-defined address symbol
- DEC
- ADREG
- 1
- NF

## Programming Syntax

All mnemonics and parameters must be separated by one or more units of a delimiter. A blank space, comma (`,`), tab and line feed are acceptable delimiters.

### Specifying External Memory Read and Load to MEMS (LDI)

```
LDI MEMSxx, MR[AddressElement(s)/NF]
```

- No delimiters can be included in the description of the `MR[…]` parameter for reading external memory.
- *`AddressElement(s)`* denotes the address element (described elsewhere) linked by the symbol "+".
- There must always be one address constant symbol (user-defined address symbol name) included in the address description of *`AddressElement(s)`*.

### Load to YREG (LDY)

```
LDY INPUTS
```

### Load to ADREG (LDA)

```
LDA INPUTS
```

### Multiply Add

```
@ pM*pC±(pM*pC±(…(pM*pC±(pM*pC+pA))…))
```

pM: Multiplicand
pC: Multiplication coefficient
pA: Augend

**Note:** The same number of open parenthesis [(] and close parenthesis [)] must be present in all mathematical equations. The right most open parenthesis [(] must always be to the left side of the left most close parenthesis [)].

- The following restrictions apply when combining pM and pA.

| pM | pA |
|---|---|
| INPUTS | {REG, TEMP00-TEMP7F} |
| TEMP00-TEMP7F | {REG, T} * |

**Note:** When {TEMP00-TEMP7F} is specified with pM and also TEMP with pA, only the TEMP symbols with the identical identifiers can be used. Therefore, specify T for pA.

- When the value of YREG is used as the multiplication coefficient parameter, one of either {YREGH, YREGL} is used. The difference in notation between these two registers shown in the table below.

## Store

```
> opt Destination(s)
```

*opt*: Store option; use or omit one of S1, S2 or S3.
*Destination(s)*: Store destination specification. At least one out of Group a through d shown below must be selected. This specification cannot be omitted.

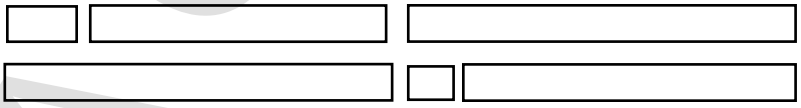
| | |
|---|---|
| Group a | One of {TEMP00 ~ 7F} |
| Group b | FREG |
| Group c | ADREG |
| Group d | One of {EFREG00 ~ 0F} |

**CAUTION:** When specifying ADREG for *Destination(s)*, S3 must not be specified for *opt*.

- The bit shift, overflow protection, and store mode settings for data is specified with *opt*. The possible combinations are shown below.

| Parameter #1 Setting | Shift | Overflow Protection | Store Mode |
|---|---|---|---|
| Omit | x 1 | Protection Enabled | A |
| S1 | x 2 | Protection Enabled | A |
| S2 | x 2 | Protection Disabled | A |
| S3 | x 1 | Protection Disabled | B |

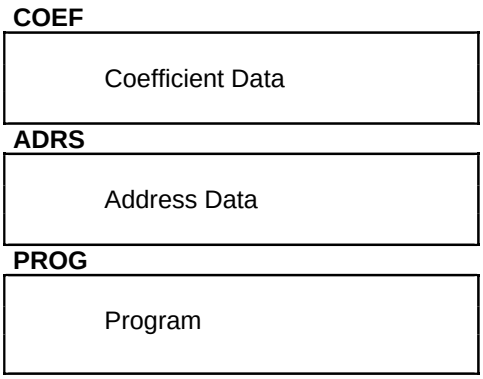
The data stored in ADREG and FREG differ depending on the specification of store mode A or B as shown below .



**Note:** The upper 4 bits (ssss in the table above) of ADREG in store mode A are filled (sign expanded) with a value (sign bit) equal to bit 23 of INPUTS.

## Reference 1: dAsms Executable File Format

Executable programs created by **dAsms**are text files based on the file format shown below. The file extension for these executables is `EXC`.




- A coefficient/address data line is formatted as follows.

```
[InternalRAM address]:[data]:[symbol name]:[notation mode/initial value]
```

**Note:** The `[notation mode/initial value]` has the same format as the right side of the coefficient/address constant definition section of the source code.

- A line of the program is formatted as shown below.

```
[ProgramRAM Address]:[data]
```

**Note:** The 64-bit data is divided up into four blocks every 16 bits in the `[data]` section. Each block represents four hexadecimal characters. Also, one blank space is inserted between each block. Refer to the executable file example shown on the next page.

- An example of an executable file is shown below.

```
COEF
00:0000:ZERO   (coefficient data from reserved coefficient symbol ZERO)
01:007B:CoefB:123
02:0800:CoefC:%50
03:0400:CoefD:0.25
04:1F85:CoefE:-123      (negative value)
05:1800:CoefF:%-50      (negative value)
06:1C00:CoefG:-0.25     (negative value)
ADRS
00:8000:AdrsA
01:007B:AdrsB:123
02:33AE:AdrsC:ms300.0
PROG
00:xxxx xxxx xxxx xxxx
01:xxxx xxxx xxxx xxxx
(omitted)
7F:xxxx xxxx xxxx xxxx
```

**Note:** An "x" represents one hexadecimal digit (4 bits) according to the PROG section format.

## Reference 2: dAsms Programming Guide

When coding **dAsms**programs, all instructions can be treated as part of a sequential processing framework. There is no need to pay attention to pipeline processing. The following processing steps are considered to be one processing unit.

1. Prepare data required for calculation (`LDI/LDY/LDA`)
2. Multiply/add expression description (`@...`)
3. Specify store destination (`>`)

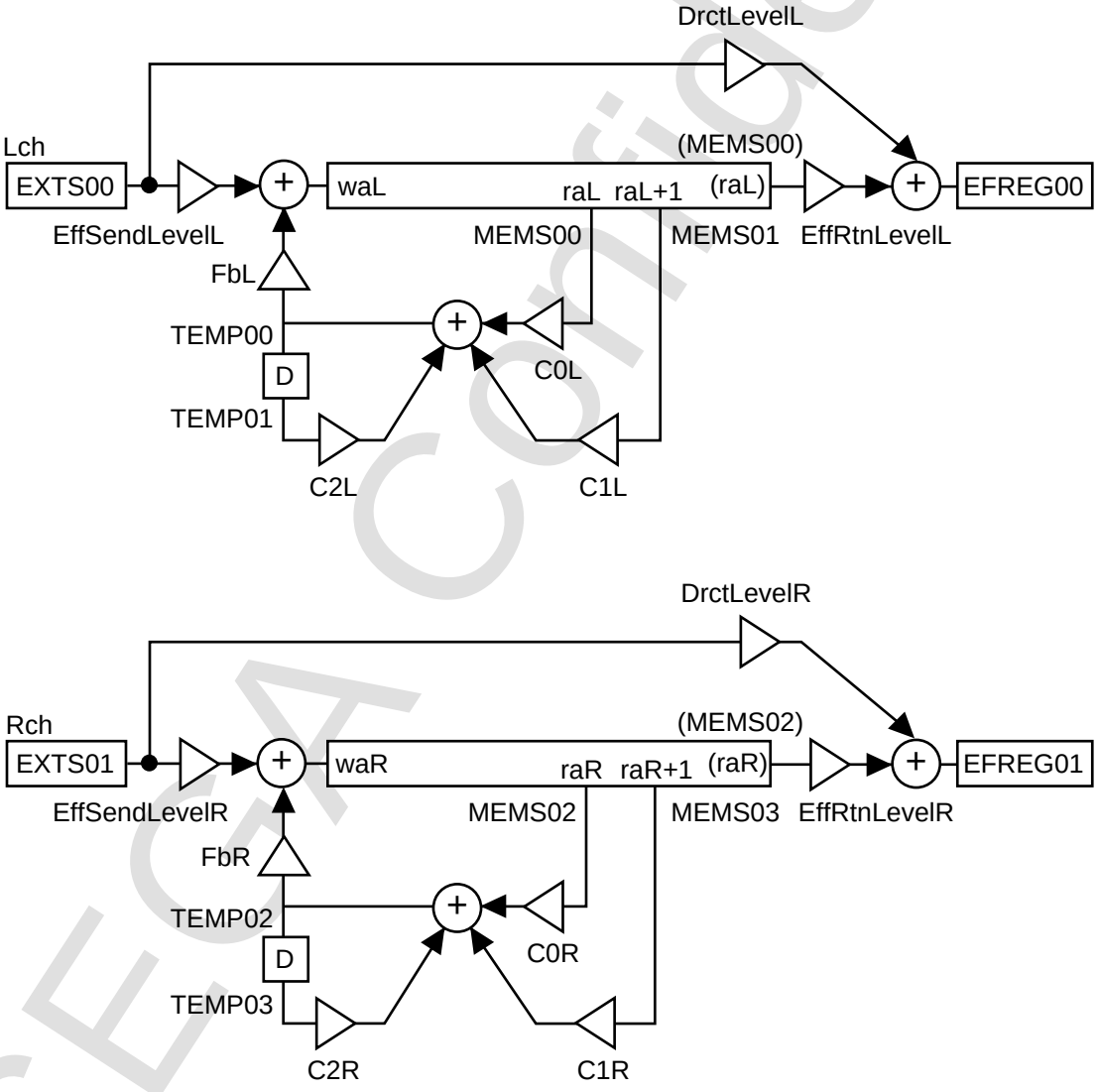
Source code can be written by taking an appropriate functional component from the signal flowchart and fitting it in the context of the processing unit. The DSP code is written by repeating this process in the signal flow processing order.

The DSP programming method used in **dAsms**is explained next based on a simple example.

• Programming example: stereo (independent left and right channel) delay

This example uses the external expansion inputs `EXTS00` and `EXTS01`, through which audio data streams from the CD-ROM can be input as the left and right channels. The results of the independent right and left channel delay processing (echo) are output to `EFREG00` and `EFREG01`.

• Signal Flow Diagram



# Creating DSP Programs

## 1. Coefficient Symbol Definition (#COEF)

### 1.1 Defining LevelAttenuation-Related Coefficients

The following are defined as level attenuation-related coefficients.

- Send level to delay: `EffSendLevelL/R`
- Direct signal level: `DrctLevelL/R`
- Return level from delay: `EffRtnLevelL/R`
- Feedback level of delay signal: `Fbl/R`

Assuming that a level resolution of 1% increments is sufficient for level attenuation, the coefficient is defined using percent notation. Refer elsewhere in this manual regarding the definition format. The definition description in the **dAsms**code is as follows.

```
EffSendLevelL=%100
EffSendLevelR=%100
DrctLevelL=%50
DrctLevelR=%50
EffRtnLevelL=%75
EffRtnLevelR=%75
FbL=%50
FbR=%50
```

### 1.2 Defining Filter-Related Coefficients for Feedback

The following coefficients are used with the 1st stage IIR filter, which becomes the feedback signal path for the delay signal shown in the signal flow diagram.

- `C0L/R`
- `C1L/R`
- `C2L/R`

Assuming that the filter coefficient is a normalized value of 1, the coefficient is defined in fractional notation. When useful cut-off frequencies are taken into consideration, the definition in the **dAsms**code is as follows.

```
C0L=0.40893
C0R=0.40893
C1L=0.40893
C1R=0.40893
C2L=0.18164
C2R=0.18164
```

## 2. Address Constant Symbol Definition (#ADRS)

### 2.1 Defining Write and Read Addresses for the Delay Ring Buffer

The rectangles in the middle of the left and right channel in the signal flow diagram represent delay. Delay is produced by the delay buffer, and the delay time is proportional to the difference between the write address and the read address. One address is equivalent to the delay of one sample. The following are defined as addresses:

- `waL/R`
- `raL/R`

Although it is possible to assign ring buffer addresses in hexadecimal or decimal integers as initial values for these address constant symbols, **dAsms** is capable of using addresses converted to time (milliseconds) notation. The following definitions are made based on some useful delay times:

```
waL=ms0.0
raL=ms149.9
waR=ms150.0
raR=ms249.9
```

# 3. Program Description (#PROG)

## 3.1 Preparing Data for the Left Channel

Memory access (reads from the ring buffer) is required in order to obtain the data necessary for calculation. The addresses in the ring buffer that need to be read are

- raL
- raL+1

However, since these addresses are only relative positions in the ring buffer, the address element DEC is inserted in the [...] part of the external memory read parameter MR[...]. DEC represents a counter that is decremented by 1 per each sample. Ring buffer operations are made possible by adding DEC to the address element notation. Assuming that the data read from raL and raL+1 positions in the ring buffer are loaded to the following,

- MEMS00
- MEMS01

the data is represented in the **dAsms** source code as shown here.

```
LDI MEMS00,MR[raL+DEC]
LDI MEMS01,MR[raL+DEC+1]
```

As an example, the addresses in the ring buffer accessed by the two MR[...] above are shifted down one address per sample by DEC while maintaining a one-address relative interval between them.

## 3.2 Filter Calculations for the Left Channel Feedback

The next step is to perform calculations on the prepared data. Calculation of the 1st stage IIR filter, which is the feedback path for the delay signal, is done using the following procedure.

1. The value loaded in MEMS00 is multiplied by the coefficient C0L.
2. The value loaded in MEMS01 is multiplied by the coefficient C1L. The result is added to the result of step 1.
3. The value of TEMP01 is multiplied by the coefficient C2L and is added to the result of step 2.
4. The result of step 3 is stored in TEMP00.

It is important here that TEMP be set up as the ring buffer. The xx in TEMPxx represent the relative address of the ring buffer. The data stored in TEMP00 appear in TEMP01 for the next sample. This is used in steps 3 and 4 above to generate a one sample delay.

In **dAsms**, the method of expressing calculations is determined by the characteristics of the SCSP/DSP hardware (target). Steps 1 through 4 on the previous page are expressed in **dAsms** code as follows.

```
@ TEMP01 * C2L + (MEMS01 * C1L + (MEMS00 * C0L +)) > TEMP00
```

In order to express the sequential multiplies and adds as shown in steps 1 through 3, each multiply must be ordered in reverse within the code compared with the actual processing order.

### 3.3 Writing to the Left Channel Ring Buffer

As can be seen from the signal flow diagram, only the sum of the following need to be written to the ring buffer:

a. The product of `EXTS00` and the coefficient `EffSendLevelL`.
b. The product of the value stored in `TEMP00` in step 4 of section 3.2 and the coefficient `FbL`.

The write address in the ring buffer is `waL`. As with `MR[...]` in section 3.1, the address element `DEC` is used. The above is expressed in **dAsms** code as follows.

```
@ TEMP * Fbl + (EXTS00 * EffSendLevelL +) > MW[waL + DEC]
```

### 3.4 Generating Left Channel Output Data and Writing to EFREG

Once again, the signal flow diagram shows that only the sum of the following 2 data need to be stored in `EFREG00`.

a. The product of the data read from address `raL` in the ring buffer and the coefficient `EffRtnLevelL`.
b. The product of `EXTS00`, which is the input value, and the coefficient `DrctLevelL`.

The data (read from the ring buffer) required for calculation in step a is already loaded in `MEMS00` by the process described in section 3.1. This process is expressed in **dAsms** code as follows.

```
@ EXTS00 * DrctLevelL + (MEMS00 * EffRtnLevelL +) > EFREG00
```

This completes the coding for the left channel.

### 3.5 Coding for the Right Channel

This example is an independent left and right stereo delay DSP program. The processing for the right channel is identical to that of the left channel, as seen from the signal flow diagram. However, the coefficient/address constant symbol names, register names, etc. are different. Code for the right channel can be written by copying and pasting the code for the left channel described above and modifying the relevant parts that need to be changed.

### 3.6 Finished!

Coding for this programming example is now complete. The entire source code that conforms with the *DSP Program Data Structure (Data Format)* section in this manual is presented below.

```
‘dAsms sample program.
‘Function:L/R independent delay
‘CD Lch Direct + Delayed -> EFREG00
‘CD Rch Direct + Delayed -> EFREG01

#COEF

‘Levels
EffSendLevelL=%100
EffSendLevelR=%100
DrctLevelL=%50
DrctLevelR=%50
EffRtnLevelL=%75
EffRtnLevelR=%75
FbL=%50
FbR=%50

‘FilterCoefs
C0L=0.40893
C0R=0.40893
C1L=0.40893
C1R=0.40893
C2L=0.18164
C2R=0.18164

#ADRS
waL=ms0.0
raL=ms149.9
waR=ms150.0
raR=ms249.9
```

```
#PROG

‘Lch
LDI MEMS00,MR[raL+DEC]
LDI MEMS01,MR[raL+DEC+1]
@ TEMP01 * C2L + (MEMS01 * C1L + (MEMS00 * C0L +))>TEMP00
@ TEMP00 * FbL + (EXTS00 * EffSendLevelL +)>MW[waL + DEC]
@ EXTS00 * DrctLevelL + (MEMS00 * EffRtnLevelL +)>EFREG00

‘Rch
LDI MEMS02,MR[raR+DEC]
LDI MEMS03,MR[raR+DEC+1]
@ TEMP03 * C2R + (MEMS03 * C1R + (MEMS02 * C0R +))>TEMP02
@ TEMP02 * FbR + (EXTS01 * EffSendLevelR +)>MW[waR + DEC]
@ EXTS01 * DrctLevelR + (MEMS02 * EffRtnLevelR +)>EFREG01

=END
```

## Reference 3: Error and Warning Messages

If an error occurs, code assembly is not possible. If a warning occurs, then assembly is executed, but a problem may exist in the program.

### 1. Source Code Configuration Errors and Warnings

**Error : Missing "#COEF".**
No #COEF label can be found.

**Error : Missing "#ADRS".**
No #ADRS label can be found.

**Error : Missing "#PROG".**
No #PROG label can be found.

**Error : Missing "#END".**
No #END label can be found.

**Error : Illegal source text.**
There was an error in the source code. The problem may be caused by an illegal ordering of the labels #COEF, #ADRS, #PROG, and #END.

### 2. Coefficient/Address Constant Definition Errors

**Error : Unknown error occurred in #COEF**
There was an unknown error in the description of the #COEF (coefficient definition) section.

**Error : Unknown error occurred in "#ADRS".**
There was an unknown error in the description of the #ADRS (address definition) section.

**Error : Initial value for *'symbol'* is out of range.**
The initial value given to *symbol* is out of range.

**Error : Invalid mode for *'symbol'*.**
The notation mode of the initial value given to *symbol* is out of range.

**Error : Too many COEF defines.**
The number of COEF definitions exceeded the maximum number allowed.

**Error : Too many ADRS defines.**
The number of ADRS definitions exceeded the maximum number allowed.

**Warning : symbol *"symbol"* never used.**

This warning is given when an unused coefficient/address constant symbol exists.

**Warning : Multiple definition.**

This warning is given when multiple definitions exist that use the same symbol name. The last definition takes priority in this case.

## 3. Program Description Errors

**Error : Undefined symbol *"symbol"* detected.**

An undefined symbol was detected.

**Error : Invalid parameter *"parameter"* detected.**

An invalid parameter was detected.

**Error : Invalid description *"string"***

There is a problem in the character string *string* in the multiplication or addition description.

**Error : Illegal parentheses.**

There is an error in the relationship between the locations of the opening "(" and closing ")"parenthesis in the multiplication or addition description.

**Error : Parentheses mismatch.**

The number of opening "(" and closing ")" parenthesis in the multiplication or addition description do not match.

**Error : Missing ">" before this line.**

There is no ">" corresponding to the previous "@".

**Error : Invalid ">" detected.**

An invalid ">" was detected.

**Error : Invalid "(" detected.**

An invalid opening parenthesis "(" was detected.

**Error : Invalid ")" detected.**

An invalid closing parenthesis ")" was detected.

**Error : Missing "]".**

A closing bracket "]" could not be found in the description of the external memory access parameter.

**Error : Invalid elements [...] detected.**

There is an error in the address element expression of the external memory access parameter.

**Error : Undefined address element "*string*" detected.**

An undefined character string was detected in the address element of the external memory access parameter.

**Error : Missing base address.**

An address constant was not included in the address element of the external memory access parameter.

**Error : Use "S3" to store to ADREG.**

The S3 option is necessary for storing the calculation result in ADREG.

**Error : PROG code overflows (*steps*).**

The number of program steps exceeded the maximum number allowed.

**Warning: Multiple STR to EFREGxx.**

This warning is given when a store is performed two or more times to the same address of EFREG. When this occurs, only the last store is valid.

# Index


Address constant symbol definition ..... 24
Assemble... ..... 8
Assembling files ..... 8
Bit shift ..... 18
Close ..... 7
Coding method ..... 21
Coefficient and address ..... 10
Coefficient and address symbol definitions ..... 10
Coefficient and address symbols ..... 10
Comments ..... 9
Data structure ..... 9
Download... ..... 8
Edit menu ..... 7
External memory read ..... 16
Feedback filter (1st stage IIR) ..... 23, 25
File menu ..... 7
    Close ..... 7
    New ..... 7
    Open ..... 7
    Quit ..... 7
    Revert To Saved ..... 7
    Save ..... 7
    Save As ..... 7
Initial value ..... 10
Internal registers ..... 15
Internal RAM ..... 15
Level attenuation ..... 23
Memory access parameters ..... 15
Multiply, add ..... 17
New ..... 7
Notation mode ..... 11
Number of user-definable symbols ..... 10
Open ..... 7
Option menu ..... 8
    SCSI ID ..... 8
Output data ..... 26
Overflow protection ..... 18
Process menu ..... 8
    Assemble ..... 8
    Download ..... 8
Quit ..... 7
Read address ..... 24
Reserved symbols ..... 10
Revert To Saved ..... 7
Ring buffer ..... 24, 25
Sample program ..... 27
Save ..... 7
Save As ..... 7

SCSI ID ..... 8
Source file ..... 7
Store ..... 18
Store mode ..... 18
Symbol name ..... 10
Write address ..... 24
ZERO ..... 10