SWL Release Note:
CalmSHINE16 V1.56i Release History

(RN_SWL_AIT_CalmSHINE16_Release_History_090703)

Title	Release Note: CalmSHINE16 V1.56i
Keywords	CalmSHINE16, V1.56i
Abstract	This document is the release note of CalmSHINE16 V1.56i

Contact Address ::

Samsung Electronics Co., Ltd.
San#24 Nongseo-Ri, Giheung_Eup,
Yongin_City, Gyeonggi-do, Korea 449-711

Tel: (82)-(031)-209-3197

Home Page: **http://www.samsung.com/CalmSDK**
Contact us : **CalmSupport@samsung.com**

S/W Solution.,
SOC Development,
System LSI Division,
Semiconductor Business
Samsung Electronics Co., Ltd.

Revision History

Date	Version	Author	Approver	Amendment
March 14, 2007	0.9	Minja Han		Created base on 1.56d release note
May 8, 2009	GUI 1.56i beta6	YongJoo Kwon	Minja Han	
July 3, 2009	1.56i	Yongjoo Kwon		Create document for CalmSHINE16 V1.56i Install package

Contents

1 How to get V1.56i Install Package 1
2 Release History of V1.56i 1
3 Matters that Require Attention 1
3.1 User's Guide for inserting NOP instruction in Command Line And IDE in Some Devices (from 1.56 CalmSHINE16) 1
3.1.1 Related Device Lists 1
3.1.2 Assembler (from V1.54e1S) 1
3.1.3 Linker 2
3.2 Assembler(from 1.56aS) 3
3.3 Linker(from 1.56bS) 3
3.4 S3FC9AC, S3FC9AE(S3EC9A1 eva chip) 4
3.4.1 Firmware 4
3.4.2 Linker usage (from 1.56mS) 4
4 Revision History after releasing V1.54 5
4.1 GUI-CalmSHINE16 5
4.2 Language tools 7
4.2.1 Preprocessor – Cprep16.dll 7
4.2.2 Compiler – CalmCC16.dll 7
4.2.3 Optimizer – Calmopt16.dll 17
4.2.4 Assembler – Calmasm16.dll 18
4.2.5 Linker – Calmlink16.dll 19
4.2.6 Librarian – Calmlib16.dll 20
4.2.7 Library 21

1 How to get V1.56i Install Package

http://www.samsung.com/CalmSDK
You can download CalmSHINE16 V1.56i in CalmSHINE16 Board.

2 Release History of V1.56i

Each tool version according to Packages.

Tools \ Ver.		V1.56f		V1.56h	1.56i	
		June 05,2008	Oct 14,2008	Dec 22,2008	July 3,2009	
GUI& Debugger	CalmSHINE.exe	V1.56f		V1.56h	V1.56i	
Preprocessor	Cprep16.dll	V1.56c		V1.56c	V1.56c	
Compiler	CalmCC16.dll	V1.56o	V1.56p	V1.56r	V1.56r	
Optimizer	CalmOpt16.dll	V1.56g		V1.56g	V1.56g	
Assembler	Calmasm16.dll	V1.56bS		V1.56bS	V1.56bS	
Linker	Calmlink16.dll	V1.56jS		V1.56jS	V1.56mS	
Librarian	Calmlib16.dll	V1.56S		V1.56aS	V1.56aS	

3 Matters that Require Attention

3.1 User's Guide for inserting NOP instruction in Command Line And IDE in Some Devices (from 1.56 CalmSHINE16)

3.1.1 Related Device Lists

When the user use below Devices, there is the matters that require Attention.

S3EC9E1, S3CC9E4, S3CC9E8
S3EC9G1, S3CC9G4, S3C9G8
S3EC9G0, S3CC9GC, S3CC9GW
S3EC9Q0, S3CC9EB, S3CC9EF, S3CC9NC, S3CC9NW, S3CC9NA, S3CC9TC, S3CC9TW, S3CC9TF

These Devices run incorrectly when the some code combination exist in EEPROM.

The IDE support automatically forcing below options for standard Library, so all users of above Devices should just Rebuild All EEPROM Project.

BUT, when the User library is made by IDE allocated EEPROM, After rebuilding the User Library by "-d" in Assembly Option User Input. Please use "-9 <User Library Name>" option in Linker Option User Input. And Build the Application Project.

And the Command Line User should follow below Guide line.

3.1.2 Assembler (from V1.54e1S)

Following assembler options are supported.

-b This option inserts a NOP between a specific sequence of code.

```
LDW   An, @[Ai+x]
LDB   Ry, @[An+z]    OR    LDW   Ay, @[An+z]
```

In the above pattern first instruction must be a load to An from Ai and the next instruction must be wither LDB or LDW with An as source register. Destination of first LDW must be same as source of next LDB/LDW. The NOP is inserted after first LDW if this pattern is found in EEPROM are (ECODE section).

```
eSec1   SECTION ECODE
        eSec1
        ld      a9, #_xyz
        ldw     a10, @[a9+2]
        ldb     r2, @[a10+2]
        add     r2, #2
```

In above case a10 is destination in first LDW and source in next LDB, and these instructions are placed in ECODE section, thus a NOP is inserted after first LDW.

>CalmAsm16 –b input.s

-d Same as –b but NOP is inserted regardless of the section type, i.e. NOP is inserted if sequence is found in any code section (CODE and ECODE).

This option is useful to build a library file. As mentioned earlier a NOP is inserted in all code sections (CODE and ECODE), so that during linking time any of the CODE section can be placed in EEPROM area.

>CalmAsm16 –d –L input.s

3.1.3 Linker

3.1.3.1 EEPORM library (from V1.54f5S)

Following linker option is supported:

-9 <elib> Where elib is a library file. Modules from this file will be placed in EEPROM area if requested from EEPROM code.

Linker supports two kinds of library files,

–l <lib> Option to link with normal library files.
–9 <elib> Option to link with EEPROM libraries. Where the library has been built using -b or –d assembler option.

A library module requested form CODE section (ROM area) is searched in library files mentioned in –l <lib> option. If the requested module is not found then it is searched in library files mentioned in –9 <elib> option. The requested module is then kept in ROM area.

A library module requested form ECODE section (EEPROM area) is searched in library files mentioned in –9 <elib> option. If the requested module is not found then it is searched in library files mentioned in –l <lib> option. The requested module is then kept in EEPRROM area.

If same library module is requested from CODE and ECODE section then it is searched in library files mentioned in –l <lib> option and then in –9 <elib> option. The requested module is then kept in ROM area.

>CalmLink16 –l calm16_1.lib –l calm16_2.lib –9 eep_lib1.lib –9 eep_lib2.lib <other options> <input object files>

3.1.3.2 Memory Description file (MD file from V1.56aS)

The linker combines input sections from one or more object and library files to create executable image (HEX and HXD files). The generated output file can be viewed as group of regions with every region having different (or same) load and run address.

MD file mechanism enables user to specify memory map of an image to the linker. It gives complete control over grouping and placement of regions. MD file can be used for complex memory maps, where code and data must be placed into different areas of memory. It can be used when device has different types of memories.

Image regions are placed in the system memory map at load time. Before you can execute the image, you might have to move some of its regions to their execution addresses. For example, initialized RW data (IDATA) might have to be copied from its load address in ROM (ILOAD) to its execution address in RAM.

MD file can be provided to linker with –md <file> or –7 <file> options.

Invocation:

>**Calmlink16** –md <filename> <other options> <object files>

Where *filename* is name of the MD file, which is an ASCII text file. Option –md and the *filename* must be separated by a space or tab. But alternative –7 can be specified without space or a tab.

Following invocations are valid,

>Calmlink16 –7<filename> <other options> <object files>
>Calmlink16 –7 <filename> <other options> <object files>
>Calmlink16 –md <filename> <other options> <object files>

In case of specifying MD file in GUI, please use –7 option.

Please refer CalmSHINE16 user manual, Chapter CalmLink16, section “9. Memory description file” for more details.

3.2 Assembler(from 1.56aS)

New option -t supported to avoid printing of date and time in object file. By default assembler prints date and time information in the object file, so that CalmLib16 can use it.

3.3 Linker(from 1.56bS)

Following points are updated in linker,

- New directive #init_table_flash supported to keep initialization table in Flash memory.
- New option -SMPorder supported to change linking order in SMP file. With this option, symbols in the SMP file are searched after library files.
- Odd address warning is generated only for ABS CDATA PM sections

3.4 S3FC9AC, S3FC9AE(S3EC9A1 eva chip)

3.4.1 Firmware

- If you are S3ECA1 chip user, you should use latest version of firmware. Please check your emulator.

	Firmware Version	File Name	Release Date	Location
C1600	v1.46	C1600_1.46_090430.b	Apr, 30 2009	"₩Firmware" directory or http://www.yicsystem.com/
C3200	v1.03d	C3200_1.03e_90n_CalmFlash_20090616.bin	Jun, 16 2009	"₩Firmware" directory or http://www.yicsystem.com/

If you have any problem when using them, contact to YIC System's Customer Support Team(or regional FAE).
YIC System's homepage : http://www.yicsystem.com/

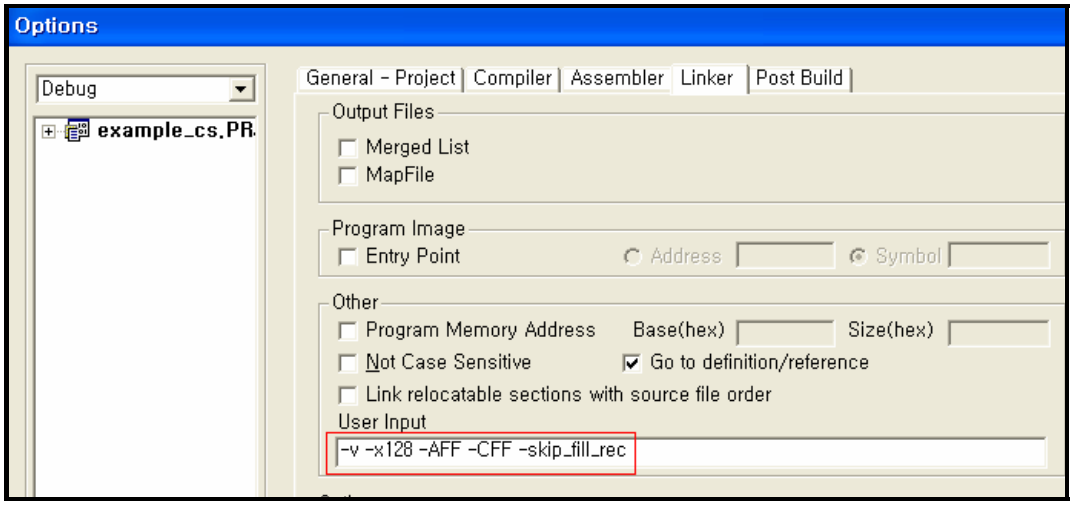
3.4.2 Linker usage (from 1.56mS)

We added some options of Linker because of flash memory read/write restriction of S3ECA1.

-v : Generate Hex dump. The dump is generated in .hex and .hxd file
-x<len> : Specify record length (Default is 32, max is 255)
-C<fill byte> : Fill value for HEX (00~FF)
-A<fill byte> : Fill value for HXD (00~FF)
-skip_fill_rec : With this option, hex/hxd records with only fill data are not printed in the output file

Example>
>CalmLink16 -v –x128 -AFF -CFF –skip_fill_rec<other options> <object files>

Or input the options at linker option as below.



4 Revision History after releasing V1.54

4.1 GUI-CalmSHINE16

Version	Released date	Comments
V1.56i	**Jul. 3th , 2009**	[FIX] Dual monitor doesn't be supported [NEW] S3EC9A1 device supported. [NEW] Host firmware for C3200/C1600 updated.
V1.56h	**Dec. 22th , 2008**	[FIX] Bug fix for breakpoint icon. [FIX] Bug fix for register modifying value in register view. [FIX] C3200 firmware update. [FIX] Bug fix for Find in files error [FIX] Bug fix for Trace view
V1.56f	**June. 05th , 2008**	New1. Add disassembly view scroll up. New2. Add [Add] button in [option->Set Source Path] If user presses [add] button and adds file path list file, Tool will parse this line and then show each file path in set source path dialog. New3. Add new Function which let user know either Flash chip erased or not in tool. New4. Add register doing typing directly in 'watch register window'. Fixed1.When user single step at assemble debugging mode, STOP instruction can't be operated. Fixed2.Flash Area Download Error. Fixed3.When download data in area that is not DVM Area, display message so that user knows. Fixed4.When customer want to display the source in the MDS trace window, the scroll down is unacceptably slow.(simple trace)
V1.56e	**Mar. 6th , 2007**	New1. 'download' or 'download after merging' option is added for the flash device when you select [Debug]-[Download data]. New2: [Option]-[JTAG clock setup] menu is added for OPENice-C3200 Fixed1: Downloading error to EEPROM and flash memory Fixed2: Step In error in Conditional Statement block. Fixed3: Project was closed if you do the following sequence. Open a file -> Open a project -> click 'Cancel after selecting [File]-[Close Project] Fixed4: Disassembly window showed incorrect code after setting a breakpoint Fixed5: Data was shown incorrectly in the disassembly window. Fixed6: 'Find in Memory' window appeared if data memory 4 is clicked. Fixed7: Project open error with displaying "This project is not CalmRISC project". Fixed8: Unsupported functions by OPENice-c3200, for example timer and trace, was enabled. Fixed9:The values displayed incorrectly in the Watch Variable window
V1.56d	**Jun. 26th , 2006**	Fixed 1: Data breakpoint error at Go Setup Fixed 3:swi instruct is added for line branch instruction.
V1.56c	**Apr. 03th , 2006**	New 1:Seach function in dvm/mem file in case of Hex debugging. Fixed 1: Path search error of profiling function in the simulator mode. Fixed 2:Redownload error, caused by unclosing the opened file if an error occurred when downloading image. Fixed 3:Memory leakage problem Fixed 4:Data breakpoint error at Go Setup

V1.56b	**Mar. 02th , 2006**	Fixed 1:'Skip to cursor' bug: the actions in disassembly and c source modes are different. Fixed 2: In specific case, 'Global Project option when using source path Fixed 3: File size limitation error if the size was big. Fixed 4: HW B.P setting error in case of OPENice-c3200. Fixed 5: The delay is added if the retry is failed after the image download fail to the flash memory.
V1.56a	**Jan. 26th , 2006**	New 1: [Internal] A flag is added for making a merge hex file and the option is added to project file. New 2: Scroll speed of code memory view is enhanced. Fixed 1: C3200 MDS was worked so slowly.
V1.56	**Dec. 25th , 2005**	Fixed 1:Flash memory write error when wrting a data to the odd addresses. Fixed 2:The dll handle is closed in case of abnormal program termination. Fixed 3:New watch function added. Fixed 4:compiler/assembler option was not applied immediately.
V1.55	**Dec. 15th, 2004**	New 1: 'External program memory download' is supported. New 2:fram related items are added to the emum in the device name that supports the device containing fram instead of eepm. New 3: Three memory views are added Fixed 1:Download retry error when an error is returned during downloading Emulator OS, which cause no response. Fixed 2:Register view tab style is back to the original.
V1.54e	**Oct. 11th, 2004**	Fixed 1: Register R0 recovery bug at a specific computer that used a serial communication Fixed 2: Display a hidden option likes a "-mem" option. Fixed 3: A Hardware board reset detect by current PC mismatch case. Fixed 4: Include lib file path that project option menu can a relative path Fixed 5: GUI support a mem file parsing for a new linker option Fixed 6: No more used "-mc" option in compiler option Fixed 7: Bug in local variables view in some map file project
V1.54d	**Jul. 26th, 2004**	Fixed 1: Register R0 recovery bug Fixed 2: Assembler error message counter option bug Fixed 3: project include path using from debug mode in case of release mode bug
V1.54c	**Jul. 15th, 2004**	Fixed 1: break point set in S3FC9UB Fixed 2: Download problem in S3CC9E4 Fixed 3: Header file dependency bug in make menu
V1.54b	**Jun. 14th, 2004**	New: Modified to support a new product (S3FC9UB) debugging. Fixed 1: In project file view, a wrong display of each file option page. Fixed 2: Ctrl +F key, string display bug which current cursor exist. Fixed 3: executed "Rebuild all" even though press "Make" Fixed 4: GUI did not parse "/*"(comment indicator) in a mem file. Fixed 5: In disassemble mode, Timer & Trace flag was set a wrong line.
V1.54a	**Mar 26th, 2004**	Fixed: In option, a user input value and a predefined value are duplicated. Fixed: The default C-startup file always is added instead of the user file.

*Fixed 3 in V1.54d: We notice , when the project made by old version(before 1.54) is opened on this version, all files have to be re-compiled ONCE because from 1.54 version tool changed the information generation part. So If the user want to make new hex file (using "make") tool will rebuild all ONCE, but after successfully passing the compile stage of all files, then the tool will only execute the linking when "make" button clicked.

4.2 Language tools

4.2.1 Preprocessor – Cprep16.dll

Version	Released date	Comments
V1.56c	Jul 24th 2006	Option –t supported to hide errors if invoked by make utility
V1.56b	Mar 6th 2006	Algorithm for include files with relative path (../../) changed.
V1.56a	Jan 9th 2006	Input character '\r' is replaced with '\n'
V1.56	Oct. 28th, 2005	Just, version name is changed.
V1.55c1	June 2nd 2005	Bug in multi-line comment processing has been removed.
V1.55c		Internal version
V1.55b	Fen 25th 2005	Supported "Un-terminated string or char constants" in #error directive.
V1.55a	Jan 25th 2005	Token sequence)##xyz in string concatenation is supported.
V1.55	Dec 15th 2004	Version number changed for release
V1.54c4	Dec 3rd 2004	Build message is properly printed on DOS console
V1.54c3	Oct 27th 2004	Modified: #error directive ignored by the CPP
V1.54c2	Oct 8th 2004	New : Macros with null arguments have been supported. Fixed :Bug in "send message to UI function" has been removed
V1.54c	Jul. 12th, 2004	New 1: If EOF is inside multi-line comment then only generate error. If it is inside single-line comment then no error or warning. New 2: Change in #line format #lineNUM --> #line NUM New 3: Supported _CALM8_, __CALM16__ and __CALM32__ pre-defined macros New 4: Changed maximum number of include paths to 255 Fixed : Do not print #line for multi-line comment if multi-line comment is on single line
V1.54b	Apr. 15th, 2004	1. Maximum number of include paths to 255 2. Maximum number of arguments in -f file option to 259
V1.54a	Apr. 7th, 2004	1. Argument length in -f filename is changed to _MAX_PATH from 100 2. Change in #line format #lineNUM --> #line NUM 3. Supported _CALM8_, __CALM16__ and __CALM32__ pre-defined macros. Just build the CPP with appropriate #defined for that target

4.2.2 Compiler – CalmCC16.dll

Version	Released date	Comments
V1.56r	Dec 16th 2008	Fixed : fixed stack overflow problem
V1.56q	Nov 13th 2008	Fixed : fixed crashing problem with generic keyword.
V1.56p	Oct 14th 2008	Fixed : fixed switch case bug when case value is negative. Fixed : fixed crashing problem when address optimization is performed.(the value which is declared as extern is declared after function definition .)
V1.56o	May 02nd 2008	New : Support _at_ and overlay keyword. Ex> int i[2] _at_ 0x20000 , overlay;
V1.56m	Jan 17 th 2008	Fixed : -bitsupport option do not save A8 register in the interrupt routine.
V1.56l	Dec 18 th 2007	New :-bitsupport option added for general usage of A8 register.

V1.56k	**Mar 28th 2007**	Fixed: Wrong implicit conversion for arithmetic expression for variables that have 'unsigned char' type.
		
		Ex>
		#include <stdio.h>
		unsigned short a[2] = {1, 2};
		unsigned short *p = &a[1];
		void main(void)
		{
		unsigned char n = 1, m = 2;
		
		p += n - m; // The expression 'n-m' should be dealt with 'int' type
		// even if they is 'unsigned char' type.
		if (*p != a[0])
		printf("fail\n");
		}

V1.56h	**Sep 8th 2006**	Fixed: Warning is generated about static variable that is referenced over optimization level 1.
		Ex>
		static int i;
		void foo(void) {
		i = 1; // warning generated : static int _i is not referenced.
		}

		Fixed: parameter passing is incorrect when calling function pointer with variable argument.
		Ex>
		void var_para(const char *, …);
		void (*fpt)(const char *, …);
		void foo(void)
		{
		fpt = var_para;
		fpt("dafdkf", 1); // All parameters should be passed by stack instead of
		// register.
		}

		Fixed: Crush occurs when calling absolute address.
		Ex>
		void foo(void) {
		(void (*)(int)0x10)(1);
		}

V1.56g_ beta3	**July 14th 2006**	Fixed:
		LDW Rn, @[An+imm] : imm-odd number
		=>
		ADD An,#imm
		LDW Rn,@[An]
		
		Fixed: expression "0x3E+3" has compile error
		Fixed: If compile error occur, then delete output *.z file.

V1.56g_ beta2	**July 12th 2006**	Fixed: bug fixed with "-ms" and "-igcode", "-iggeneric" option.
V1.56g_ beta1	**July 11th 2006**	Fixed: bug fixed with "-ms" and "-igcode", "-iggeneric" option.
V1.56g_ beta	**June 27th 2006**	Fixed: bug fixed with "-ms" and "-igcode", "-iggeneric" option.
		Fixed: block copy bug fixed for generic and code keyword.

V1.56d	**Mar. 28th 2006**	Ext: support SWI function call
		
		*Ex>
		#pragma function = interrupt_swi 5
		extern void inter1(int i);
		
		test()
		{
		inter1(3); // can be changed "SWI #5"
		}*

V1.56f_ beta5	**June 12th 2006**	Fixed:"-Nswi" option have some problem in optimization level0. If local variable is usded in the SWI function body in the level0, then SSR_SWI value does not restored properly
V1.56f_ beta4	**June 8th 2006**	Ext:"-Nswi" option added for SWI function . When this option is enabled , push /pop SSR_SWI.

V1.56f_ beta3	**May 16th 2006**	Fixed: Over the optimization level 1, 'ldw/ldb' instruction is used in access to 3-dimension array with 'code' keyword
		
		*Ex>
		#include <stdio.h>
		code int arr[2][2][2] = {{{1,2},{3,4}}, {{5,6},{7,8}}};
		void main(void) {
		if (arr[1][0][0] != 5) // should be accessed by 'ldc' instruction
		printf("fail\n");
		}*

V1.56f_ beta2	**Apr. 24th 2006**	Fixed: interrupt function not declared with extern has some problem.
		Fixed: when address optimization , compiler crashed.
V1.56f_ beta	**Mar. 28th 2006**	Fixed: function pointer bug fixed. *Ex> ((unsigned char (*)(void)) (CmdTbl[0] & 0xffffff)) () ;*
		Fixed: block copy bug fixed for generic and code keyword.

V1.56d	**Mar. 28th 2006**	Ext: support SWI function call
		
		*Ex>
		#pragma function = interrupt_swi 5
		extern void inter1(int i);
		
		test()
		{
		inter1(3); // can be changed "SWI #5"
		}*

V1.56c	**Feb. 28th 2006**	Fixed: When literal includes the character "`", "$", or "@", the warning was generated.  *Ex> char *p = "`$@"; // warning: "string literal contains non-portable characters"*
		Fixed: When the option "-ms" – "Global variable optimization (under 64K offset)" in IDE - isn't used, interrupt handler did not initialize "R9" even if "R9" is used in the interrupt handler.

		Fixed: The definition of variable with _at_ in local area generated infinite errors.
		
		Ex>
		void main(void) {
		* int i _at_ 0x201000; // generated infinite errors*
		}

V1.56b	**Jan. 20th 2006**	Fixed: crash when you use the reserved keyword – for example int, align, code and so on – following 'pragma'. *Ex>* *#pragma align // crashed → generates warning "unknown pragma"* *#pragma code // crashed → generates warning "unknown pragma"*
		Fixed: When variable is defined by keyword '_at_' and the corresponing source line is across internal buffer used in compiling, the compiler error is generated for correct code because of mishandling buffer.
		Fixed: Changed wrong debug information for stack depth in case of "main" function .
V1.56a	**Nov. 18th 2005**	Ext: support the 'pragma' that makes the fiq/irq handler without vector setting. *Syntax :* *#pragma function=interrupt_fiq_novector* *#pragma function=interrupt_irq_novector* *Ex> making the fiq handler without vector* *#pragma function=interrupt_fiq_novector* *void fiq_handler(void) {* *}*

V1.56	**Nov. 2nd, 2005**	Fixed: When variable's definition follows the definition of function and there is the function's declaration, sorting variables for the address optimization is incorrect. It occurs only over the optimization level 1.
		
		Ex>
		#include <stdio.h>
		int a0;
		char c4;
		void foo(void);
		void main(void); // declaration
		void main(void) // body of function defined before the definition of variables a2
		{
		* a0 = 1;*
		* c4 = 2;*
		* foo(); // a2 = 0;*
		* if (c4 != 2) printf("fail\n");*
		}
		int a2; // When a variable's definition follows the definition of function
		// and there is the function's declaration,
		// sorting variables for the address optimization is incorrect.
		void foo(void)
		{
		* a2 = 0;*
		}

Fixed: In the expression that a variable which has the long type is masked with the value '0xffff' its result takes incorrectly the value of higher word instead of lower word only when the corresponding temporary register has been spilled due to the absence of allocable register. It occurs only over the optimization level 1.

Ex>

```
#include <stdio.h>
typedef struct {
   char c;
   int * pi;
} SS;
static SS ssArr = {1, (int *)0x81100};
static SS *ss = &ssArr;
static void func(unsigned int i, unsigned int j)
{
   if (j != 0x1100) printf("fail\n");
}
static unsigned long gl1 = 1; gl2 = 2; gl3 = 3; gl4 = 4; gl5 = 5; gl6 = 6; gl7 = 7;
static unsigned long l1;
static void spill(void)
{
   unsigned long l1=gl1,l2=gl2,l3=gl3,l4=gl4,l5=gl5,l6=gl6,l7=gl7;
   unsigned int i = l1;

   //The lower word of '(unsigned long)ss->pi' should be passed as parameter.
   func(0x2200 +((unsigned long)ss->pi>>16), (unsigned int)((unsigned long)ss->pi &
   0xffff));

   l2++;l3++;l4++;l5++;l6++;l7++;
}
void main(void)
{
   spill();
}
```

		Fixed: The offset calculation for the address optimization is made incorrectly. The variable defined after function's definition should be excluded from the target of the address optimization. It occurs only over the optimization level 1. *Ex>* *#include <stdio.h>* *int i0;* *char c0;* *void foo(void);* *void main(void)* *{* *i0 = 1;* *c0 = 2;* *foo();* *//The offset value of the variable 'c0' for the address optimization is changed by the variable 'i1' defined after function's definition. So the variable defined after function's definition should be excluded from the address optimization.* *if (c0 != 2)* *printf("fail\n");* *}* *int i1;* *void foo(void)* *{* *i1 = 0x1122;* *}*

Fixed: The addition of pointer is calculated as signed integer even if the type is unsigned. So when adding pointer with a value that exceeds the range of signed integer, its result is different from the expected value because of the 16-signed addition.

```
Ex>
#include <stdio.h>
void main(void) {
  unsigned char* p = (unsigned char*)0x80000;
  unsigned char  a = 0x80;
  unsigned char  b = 0x00;
  unsigned int i = 0x00ff;
  unsigned long l = 0x10000;

  // 0x80000 + ~0x00ff = 0x8ff00
  if ((p + ~i) != (unsigned char*)0x8ff00)
    printf("fail\n");
  // 0x80000 + (unsigned int)(0x10000>>1) = 0x88000
  if ((p + (unsigned int)(l>>1)) != (unsigned char*)0x88000)
    printf("fail\n");
  // 0x80000 + (0x00ff|0xff00) = 0x8ffff
  if ((p + (i|0xff00)) != (unsigned char*)0x8ffff)
    printf("fail\n");
  // 0x80000 + (0x00ff^0xff00) = 0x8ffff
  if ((p + (i^0xff00)) != (unsigned char*)0x8ffff)
    printf("fail\n");
  i = 0xff00;
  // 0x80000 + (0xff00&0xf000) = 0x8f000
  if ((p + (i&0xf000)) != (unsigned char*)0x8f000)
    printf("fail\n");
}
```

EXT: Inside function's body the definition of variable with the keyword 'code' should be not permitted.

```
Ex>
void foo(void) {
   code int i = 0x23;   // not permitted, the error should be generated.
}
```

Fixed: In release mode when using inline assembly function '__asm' the following error is generated.
Error message : *error: syntax error; invalid inline assembly*

```
Ex>
void foo(void) {
   int i;

   i = 1;
   __asm("nop");
}
```

EXT: Support the function placed the variable with the keyword 'code' at the absolute address of ROM using the keyword '_at"

```
Ex>
code int arr[10] = {2,} _at_ 0x10000;  // 0x10000 is the address of ROM
```

		Fixed: When some initialized and un-initialized variables are defined together within the same 'data_seg' of 'pragma', the section of the variables is generated incorrectly.
		
		Ex>
		#pragma memory=data_seg(DATASEC)
		int uninitialized_data0; // should have 'zdata' section attribute
		int initialized_data = 7; // should have 'idata' section attribute
		int uninitialized_data1; // should have 'zdata' section attribute
		// All variables defined with 'pragma' should be initialized additionally in 'cstartup'.
		#pragma memory=default

V1.55r	**Jul. 11th, 2005**	Fixed: When over optimization level 1 accessing a global variable, it is accessed through the wrong calculated address. It occurs only when 1byte structure is set as the base for global variable access.
		
		Ex>
		#include <stdio.h>
		typedef struct {
		char bit0;
		} BITFIELD_ONESIZE;
		typedef struct {
		char c;
		} STRUCT_ONESIZE;
		BITFIELD_ONESIZE bf; // &bf = 0x200058
		STRUCT_ONESIZE st; // &st = 0x200059
		int i; // &I = 0x20005A
		
		void main(void)
		{
		st.c = 0; // 'st' is set as base for global variable access
		i = 0x1122; // but 'I' is accessed by wrong calculated address '&st+2'
		if (i != 0x1122)
		printf("fail\n");
		}
		
		Fixed: The operation of option "-code" is wrong. A initial value of local variable that has 'float' and 'double' is stored in some tables and so under option "-code" tables of initial value should be located in ROM and be initialized by accessing 'ldc' instruction. It is same for local variable of 'char array', too.
		
		Ex>
		#include <stdio.h>
		void main(void)
		{
		float f = 1.0; // initial value is placed in ROM and when initializing 'f'
		// initial value should be accessed by 'ldc' instruction
		char arr[] = "abc";
		if ((int)f != 1)
		printf("fail\n");
		}

V1.55q	**Jun. 16th, 2005**	Fixed: When using option "-ar", the address calculation for switch statement is incorrect. Fixed: If the function located in EEPROM area is declared without pragma of "eep_seg" like the following example, the redefinition error is generated. *Ex>* *void eepFunc(void);* *#pragma memory=eep_seg()* *void eepFunc(void) // Redefinition error should not be generated.* *{ }* *#pragma memory=default* Ext: When accessing the element of multi-dimensioned array, the calculation of the access address is executed by a 16-bit operation, so improving the code density and performance. Ext: The option "-bigoffset" is newly added. It makes 32-bit operation instead of 16-bit operation when calculating access address of array's element.
V1.55f	**Mar. 2nd, 2005**	Ext: Remove setting the entry point as "_main" for the starting location of debugging. Instead of compiler, linker will do it.
V1.55d	**Feb. 18th, 2005**	Fixed: R0 reg. is not pushed/poped in optimization level 0 in using it. *Ex>* *void foo(void)* *{* *// When initializing local array, R0 reg. is used. So R0 reg. should be* *// pushed and popped in function prolog and epilog.* *char c[2] = {0x11, 0x22};* *}*
V1.55c	**Jan. 26th, 2005**	Fixed: Cast not applied when a pointer variable is casted "void*" to "generic int*". *Ex>* *generic char* myptr;* *void foo(void)* *{* *// Because "myptr" casted to final type "generic int*", the library* *// call that dereferences a generic pointer should be generated.* *int x = *(generic int*)(void*)myptr;* *}*
V1.55a	**Dec. 20th, 2005**	Fixed: Executed as a 16-bit operation when shift 4 bytes variable in both(left, right) directions. *Ex>* *unsigned char* ptr;* *struct APPLE {* *unsigned int weight;* *} *ptr2;* *void foo(void)* *{* *// the shift left should be executed as a 32-bit operation* *ptr = ptr - ((unsigned long)ptr2->weight << 2);* *}*

V1.55	**Dec. 14th, 2004**	Fixed: change how to restore "A9" register in the interrupt handler
		from
		LD A9,#|IMAGE$$IDATA$$BASE|&0x3ffff
		to
		LD A9,#|IMAGE$$IDATA$$BASE|
		LD R9,#0
		Fixed: change library "__genuread4romp_x" to "__genuread4p_x" because both is exactly same.

V1.54s1	**Nov. 27th, 2004**	Fixed: assertion occurs when changing "pointer to code" to "pointer to generic"
V1.54r	**Oct. 28th, 2004**	Fixed: When casting from "unsigned long" to "code unsigned char*",wrong codes are generated.
V1.54q	**Oct. 27th, 2004**	New: Error directive produces error message that includes the specified sequence of preprocessing tokens at compiler time. When #error directives are encountered, compilation terminates. Fixed: When casting between code pointer and non code pointer, compiler generate error.
V1.54p	**Oct. 12th, 2004**	Fixed : When accessing generic pointer, wrong codes are generated
V1.54o	**Oct. 7th, 2004**	1. optimize the performance in switch statement. change the 'addpXX' library call to 'add' instruction' replace the library call that performs 'signed add' with the 'add' instruction that does 'unsigned add' about which 'signed add' is unnecessary. 2. change path that temporary files generated -_t_m_array.t, _t_m_charr.t - from current working directory to debug one. 3. fix the mistake checking type : When assign a pointer to non-const memory to a pointer to const volatile memory, error occurs. 4. fix problem that the following error message is generated for valid code : "code data and non code data cannot be located in same pragma" 5. modify debugging information for viewing variables located at the code memory in debugging. 6. fix problem that error is generated when variables defined by "_at_" are initialized by const expression.
V1.54j	**Jul. 20th, 2004**	The code generated by the switch statement is optimized. Instead of the library call which calculates the entry address of each 'case', the 'add' instruction is generated except that the 'case' includes the 'goto' statement.
V1.54i	**Jul. 15th, 2004**	1. Remove the function of 'overlay' in '_at' 2. Modify the error handling in '_at' 3. Fix the error which is generated due to wrong processing for very long line
V1.54h	**Jun. 19th, 2004**	1. Generate stack frame size in .sm directive.
V1.54f	**Jun. 14th, 2004**	1. When a variable is defined by '_at_', it is allocated at wrong address. unsigned int A[8] _at_ 0x200000; unsigned long Example[100] _at_ 0x00200080 ; When apply global optimization, variable 'Example' is allocated at wrong address. 2. Added "#pragma function=interrupt" for SmartCard device. This pragma makes irq interrupt handler without vector setting.

V1.54e	**May 20th, 2004**	Const expression admission in '_at_'
		syntax in previous version
		[type-specifier] variable-name _at_ address[,eeprom];
		new syntax
		[type-specifier] variable-name _at_ address[,eeprom];
		in address field const expressions is supported.
		int i0 _at_ 0x200; // ok
		int i1 _at_ (0x200); // ok
		int i2 _at_ (0x200+0x10); // ok

V1.54d	**May 20th, 2004**	crash when compiling below code unsigned int i, j; unsigned char k; i = (i++)*k;
V1.54c	**-**	Internal version
V1.54b	**May 04th, 2004**	Fallacy in checking type extern const int i; const int i; Even if the above code is correct, the following error is generated. error: redeclaration of '_i' previously declared at corresponding filename

V1.54a	**Apr. 28th, 2004**	Incorrect optimization of long type
		LD A12,#_NVM_var1
		LD R4,R12
		LD R5,E12
		SUB R4,#>0x80000
		SBC R5,#<0x80000
		LDW @[SP+2],A12 ;arg long
		Above instead of a12 the value r4 and r5 is loaded in stack.
		So LDW @[SP+2],A12
		should be replaced with
		LDW @[SP+2],R5 ;arg high
		LDW @[SP+4],R4 ;arg low

4.2.3 Optimizer – Calmopt16.dll

Version	Released date	Comments
V1.56g	**Jan. 24th, 2008**	Fixed:Push/Pop A8 instruction added at prolog and epillog in the interrupt routine.
V1.56f	**Apr. 12th, 2007**	Fixed: There was some assertion error – "num_of_spill_var > 0" during register allocation. New: Support call with absolute address
V1.56e	**Jul. 11th, 2006**	Fixed: At a copy propagation optimization, there was incorrect processing for CMPU instruction.
V1.56d	**June 26 th, 2006**	Just, version name is changed
V1.56d_ beta2	**June. 8th 2006**	Fixed: For function with "#pragma function=interrupt_swi_withbody num" pattern, optimizer modified to treat as normal function prolog. This means that kinds of interrupt handler function doesn't have push/pop for temporary register as general function.
V1.56d_ beta	**Apr. 24th 2006**	Fixed: Modified to make SWI call instruction into fully normal function call.
V1.56c	**Mar. 28th, 2006**	New: From this version, optimizer assumes that C level SWI interrupt handler routine can have return value.
V1.56b	**Feb. 20th, 2006**	Fixed: Constant propagation and copy propagation bug fix

V1.56a	**Nov. 15th, 2005**	Fixed: At BSRD optimization, there was problem in instruction counting for inline assembly.
V1.56	**Oct. 27th, 2005**	Just, version name is changed.
V1.56_beta2	**Aug. 19th, 2005**	Fixed: Because of internal incorrect written register information of code Library "$__gen2_..." a push operation was missed for some register.
V1.56_beta	**Aug. 22nd, 2005**	Fixed 1: When the pointer type variable is used also as integer type through explicit type casting and the variable is spilled, incorrect code was generated for accessing the spilled An register.(in release mode) Fixed 2: Optimizer parsing error was occurred in release mode.
V1.55b	**Feb. 23rd, 2005**	Fixed: At Optimize Level 2, Incorrect optimization was applied. For the spilled local variable, The loading from stack before each operation for the variable was eliminated incorrectly.
V1.55a	**Jan. 25th, 2005**	Fixed: It could be a crash at optimizing time when *.z contains the long input line more than 280 characters. New: There was some redundant copy without being optimized at the following pattern. From this version, following 2nd register move can be removed. `LD Ra,Eb` `LD Eb,Ra` → this copy can be removed.
V1.55	**Dec. 15th, 2004**	Just, version name is changed.
V1.54i	**Dec. 8th, 2004**	Fixed: The program is compiled at optimize level 1 works incorrectly. (works well at level 0) -. Access instructions for long type ROM data were incorrect.
V1.54h	**Dec. 2nd, 2004**	Fixed: There is a crash at optimizing. -. An assertion error occurred at spilling a register.
V1.54g	**Nov. 20th, 2004**	Fixed: The program is compiled at optimize level 1 works incorrectly. (works well at level 0) -. Access instructions for long type ROM data were incorrect.
V1.54f	**Nov. 26th, 2004**	Fixed 1: Optimizer has removed A8 initialization as base control register at even interrupt pragma routine. Fixed 2: When Some long variable is used, optimizer crashed. New 1: Error handling for option mismatch between calmcc16 and calmopt16 was added New 2: Refinement of push registers list at interrupt function New 3: A10 register is not excepted for dead code elimination. New 4: Added the interrupt function detecting method.
V1.54e	**Jul. 26th, 2004**	Fixed: In optimization level 2, some instructions that access array with constant index were removed incorrectly
V1.54d	**Jul. 20th, 2004**	Fixed: When user used a function name longer than 49 characters, user might get an assembly error
V1.54c	**Jul. 7th, 2004**	New 1: Frame size is added in debug information New 2: Building logo is added in only Dos Version Fixed 1: Spilled local variable case made incorrect debug information Fixed 2: When Calmopt16 optimizes an input file, there was some crash. The reason there was a problem in register coalescing
V1.54b		Internal version
V1.54a		Internal version

4.2.4 Assembler – Calmasm16.dll

Version	Released date	Comments
V1.56cS	**Mar 18th 2009**	[FIX] Size of parsing buffer increased
V1.56bS	**Apr 6th 2007**	[FIX] Debug directive .fe generated for include files with .end directive
V1.56aS	**Feb 28th 2006**	[New] Option –t is supported to avoid printing of date and time in object file

V1.56S	**Oct. 28th, 2005**	Just, version name is changed.
V1.55c5S	**Oct 21st 2005**	External symbols with expressions supported in .equ directive
V1.55c4S		Internal version
V1.55c3S	**July 1st 2005**	Limitation of number of include paths has been removed.
V1.55c2S	**June 2nd 2005**	Branch to different section but same file is supported.
V1.55c1S	**May 2nd 2005**	-T option supported for profiling
V1.55cS	**Mar 26th 2005**	Error checking added for LDB, AND, XOR, TST, OR and MUL instructions.
V1.55bS	**Jan 17th 2005**	Internal version (NOP insertion for Profiling instructions)
V1.55aS	**Jan 3rd 2005**	1. Error checking added for DM,CDATA section 2. Error handling added for BNZD instruction. 3. Relocatable symbols supported in BLOCK directives
V1.55	**Dec 15th 2004**	1. Relocatable symbols in .equ directive are supported 2. Alignment at the beginning of a section is supported.
V1.54eS	**Oct. 8th ,2004**	1. New mnemonics EFZ, EFS, EFST and EFZT supported for MAC2424 for EFZ16, EFS16, EFS16T and EFZ16T 2. Global symbols supported in token pasting
V1.54cS	**Jul. 26th, 2004**	1. 8bit offset allowed in BNZD instruction 2. Operand checking added in BSRD, BRT/D, BRF/D instructions
V1.54bS	**Jul. 12th, 2004**	1. Token pasting (##) operator supported
V1.54aS	**Mar. 12th, 2004**	1. Source file path has been added in the include dir list 2. ADD SUB LD instruction with label bug solved 3. Argument length in -c filename is changed to _MAX_PATH from 100 4. Bug solved in WARNING directive 5. Bug solved in parsing of BITR/S/C/T instructions

4.2.5 Linker - Calmlink16.dll

Version	Released date	Comments
V1.56mS	**May 8th 2009**	[New] Hex dump is generated in .hex and .hxd files. .rom and .eep are not generated.
V1.56jS	**May 2nd 2008**	[New] Option –summary supported to print used/unused areas in the memory.
V1.56iS	**Oct 11th 2006**	[New] Error generated if no space in data memory for stack
V1.56hS	**Sept 19th 2006**	[New] Output files .hex, .hxd, .map etc. deleted in case of error
V1.56gS	**May 15th 2006**	[New] Option –flashcode updated to keep IDATALOAD in flash even for default sections (Non-MD file sections).
V1.56fS	**April 25th 2006**	Internal version
V1.56eS	**April 24th 2006**	[New] Option -flashcode updated to keep IDATALOAD in flash memory.
V1.56dS	**April 4th 2006**	[New] New option -FlashCode added to convert CODE and CDATA to ECODE and EDATA respectively
V1.56cS	**Mar 3rd 2006**	[New] Linker now allows odd sized EDATA sections. It also aligns a load region if it contains a code section.
V1.56bS	**Feb 28th 2006**	- [New] New directive #init_table_flash supported - [New] New option -SMPorder supported to change linking order in SMP file - [Fixed] Odd address warning is generated only for ABS CDATA PM sections
V1.56aS	**Jan 25th 2006**	Memory description file supported.
V1.56S	**Oct. 28th, 2005**	Just, version name is changed.
V1.55c3S	**Oct 19th 2005**	Segment given in –L<seg> option can be defined in @group and normal segments in MEM file.
V1.55c2S	**June 2nd 2005**	Stack setting done properly even when any one of -T, -U or -W option is given
V1.55c1S	**May 5th 2005**	Error message changed to display memory segment name and absolute

		address of a section, if section cannot be allocated in given memory area.
V1.55cS	**April 14th2005**	Bug solved in flexible memory segment processing.
V1.55b2S	**Mar 2nd 2005**	Default entry point set to _main
V1.55b1S	**Jan 25th 2005**	Linker now tries to avoid "IDATA/ZDATA region split" warning by allocating IDATA/ZDATA section contiguously in DM.
V1.55bS	**Jan 3rd 2005**	1. Bug removed in overlay section allocation. Now relocatable section does not interfere with overlay sections. 2. MEM file specification for overlapped segment has been changed
V1.55aS	**Dec 20th 2005**	Alignment for sections in @group is supported
V1.55	**Dec 15th 2004**	1. Alignment at the beginning of a section is supported 2. Library function realloc() error has been removed.
V1.547fS	**Dec 10th 2004**	Default section initialization has been done before any access to it is made.
V1.54f6S	**Nov 3rd 2004**	Sections with odd size had been taken care in the new syntax of MEM file.
V1.54f5S	**Nov 3rd 2004**	New option -9/-elib <libfile> has been added to link with EEPROM libraries.
V1.54f4S	**Oct. 29th,2004**	Fixed: The problem occurred when linker tries to process IDATA section in EEP area.
V1.54f3S	**Oct. 8th ,2004**	1. Placement of sections with –L <seg> changed. No IDATALOAD is created form EEP section. 2. New option -fvlist <file> (or -6 <file>) has been supported, to print function and variable information in <file>. 3. Use of tmpfile() library function has been removed because of ClearCase dynamic view problem. 4. Un-limited number of memory segment names in a MEM file is supported. 5. New syntax @group with overlay keyword has been supported in MEM file. 6. Overlay keyword with flexible memory segments (‘[‘ and ‘]’) supported in MEM file. 7. Maximum stack depth has been printed at the end of MAP file 8. New option -sort / -8 option supported to sort map file symbols on address. 9. Crash removed during MEM file parsing. 10. Change in MEM file error message format.
V1.54eS	**Jul. 26th, 2004**	New : Motorola S28 record file generation supported with -S28 or -5 command line options
V1.54dS	**Jul. 12th, 2004**	New 1: -X option supported to generate ROM in .b and EEP in .bdt New 2: Supported @ cmd.txt --> for all command line arguments New 3: Call tree and stack depth view added in MAP file New 4: MAP file is generated even in case of error New 5: -A <value> option supported to fill HXD file New 6: Supported -str<addr> -end<addr> to dump EEPROM in .epp file New 7: Motorola hex file generation supported with -S38 option
V1.54cS	**Mar. 31st, 2004**	1. Option -v changed to generate .rom and .epp files 2. Supported @ cmd.txt --> for all command line arguments 3. Supports file paths with spaces and quote ("), if given in -f file
V1.54bS	**Mar. 26th, 2004**	1. Proper error generated for missing MEM attribute for a section
V1.54aS	**Mar. 12th, 2004**	1. -X option supported to generate ROM in .b and EEP in .bdt -b option remains as it is.

4.2.6 Librarian – Calmlib16.dll

Version	Released date	Comments

V1.56aS	**Dec 26th 2007**	Library file supported as one of the input files. Librarian reads the input object files from given library file.
V1.56S	**Mar 15th 2006**	New version for SmartCard created.
V1.56	**Oct. 28th, 2005**	Just, version name is changed
V1.55	**Dec 15th 2004**	Multiple option processing supported.

4.2.7 Library

Version	Released date	Comments
V1.56a	**Jan. 27th 2006**	Fixed:Setjmp,longjmp were wrong with “const regarded as code” option.
V1.56	**Oct. 29th, 2005**	New: add library compiled with optimization level2.