8-bit Microcontroller

KM101LR03/04/05 Series Application Note for ReRAM Memory Rewriting

For additional information or question, please contact Nuvoton Technology Corporation Japan.
www.nuvoton.co.jp

Oct 30, 2020 Rev 1.10 (3170502-011KE)

About this document

■ Composition of this document

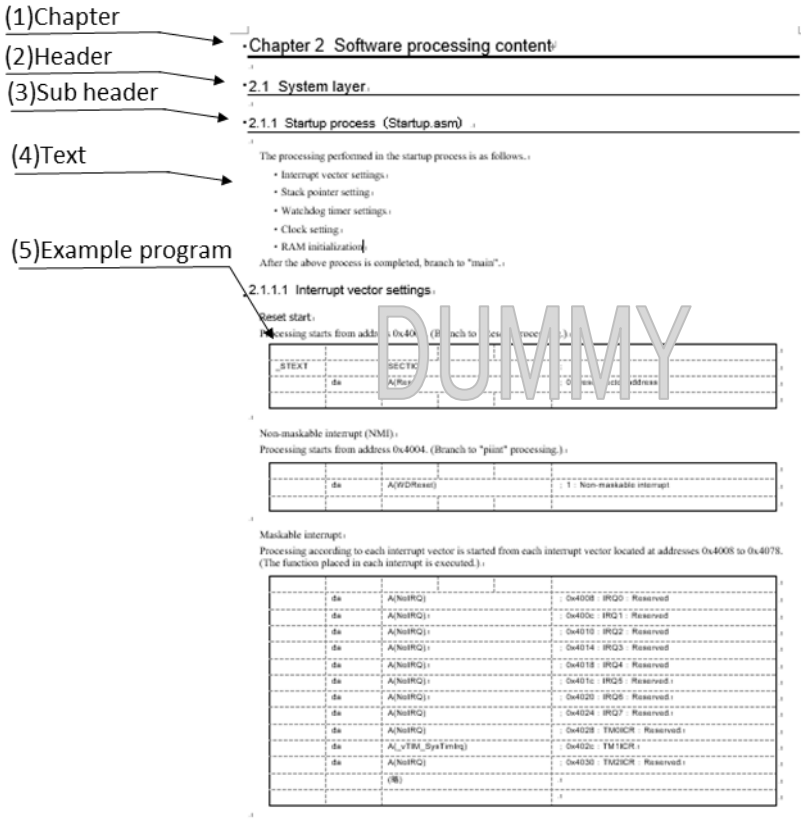
This document describes the software of the ReRAM microcomputer KM101LR05D.

Chapter 1 provides an overview of the software.

Chapter 2 describes the software processing contents.

■ Configuration as a manual

The description in this manual consists of a Chapter, Text, and Example program. The layout and definition of each part is as below.



■ List of manuals related to KM101LR05D

Manuals related to KM101LR05D are shown as below.

- “KM101LR05D/04D/03D/02D LSI manual”
- “KM101LR05D Application Note for Peripheral Circuit Control”
- “KM101LR05D ReRAM Memory Rewriting Application Note” (This document)

Contents

Chapter1 Overview

1.1 Purpose of document

This document describes the sample program that rewrites the ReRAM memory data area built into the KM101LR05D.

1.2 Evaluation board block diagram

The composition of the evaluation board is as follows.

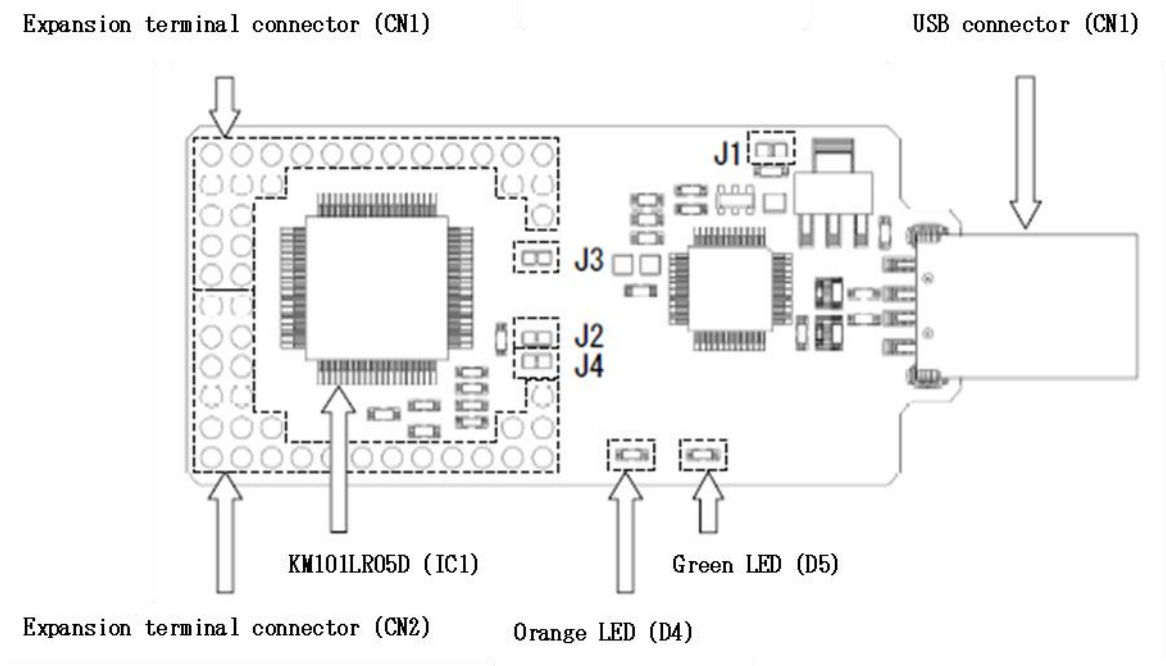


Fig 1.1 Overall view of the evaluation board

●Expansion terminal connector layout

CN3 15	CN3 14	CN3 13	CN3 12	CN3 11	CN3 10	CN3 9	CN3 8	CN3 7	CN3 6	CN3 5	CN3 4
CN3 18	CN3 17	CN3 16								CN3 3	CN3 2
CN3 20	CN3 19										CN3 1
CN3 22	CN3 21										
CN3 24	CN3 23										
CN2 26	CN2 25										
CN2 24	CN2 23										
CN2 22	CN2 21										
CN2 20	CN2 19										CN2 1
CN2 18	CN2 17	CN2 16								CN2 3	CN2 2
CN2 15	CN2 14	CN2 13	CN2 12	CN2 11	CN2 10	CN2 9	CN2 8	CN2 7	CN2 6	CN2 5	CN2 4

CN2	1	VBUS
CN2	2	VBUS
CN2	3	VDD
CN2	4	GND
CN2	5	VDD
CN2	6	NRST
CN2	7	VLC1
CN2	8	VLC2
CN2	9	VLC3
CN2	10	GND
CN2	11	P77
CN2	12	P76
CN2	13	P75
CN2	14	P74
CN2	15	GND
CN2	16	P67
CN2	17	SBT0A
CN2	18	SBO0A
CN2	19	SBI0A
CN2	20	P63
CN2	21	P62
CN2	22	P61
CN2	23	P60
CN2	24	P57
CN2	25	P56
CN2	26	P55

CN3	1	A0
CN3	2	P11
CN3	3	P12
CN3	4	GND
CN3	5	P13
CN3	6	VREFP
CN3	7	P22
CN3	8	SDA2B
CN3	9	GND
CN3	10	SCL2B
CN3	11	P25
CN3	12	RXD1A
CN3	13	TXD1A
CN3	14	P31
CN3	15	GND
CN3	16	P32
CN3	17	P45
CN3	18	P46
CN3	19	P50
CN3	20	P47
CN3	21	P52
CN3	22	P51
CN3	23	P54
CN3	24	P53

Fig 1.2 Expansion terminal connector “CN2”layout

1.3 Software configuration

The software configuration of the sample software is described.

1.3.1 Functions of the microcomputer used by the software

Function	By application	Interrupt	Interrupt Level	Remarks
Operating clock	Built-in high-speed oscillation	-	-	10MHz±2%
General purpose port	For LED lighting	-	-	-

1.3.2 Software configuration diagram

The software configuration consists of three layers: system layer, application layer, and driver layer.

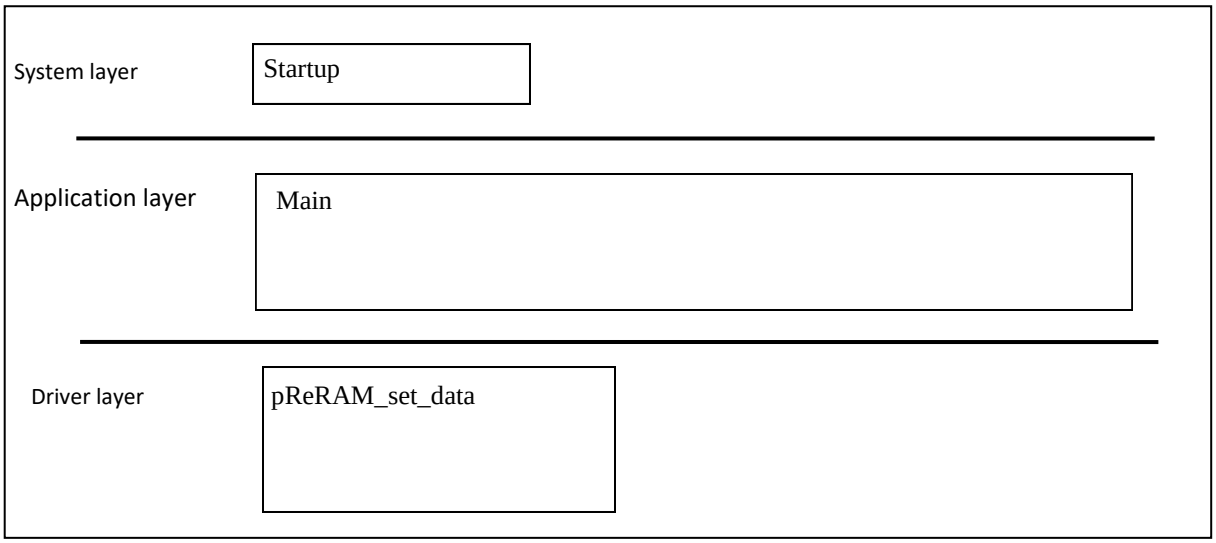


Fig 1.3 Software configuration diagram

1.3.3 Memory allocation

The memory layout of the sample program is as follows.
The startup address and interrupt vector address settings are located from 0x4000, and other programs are located from 0x4900.

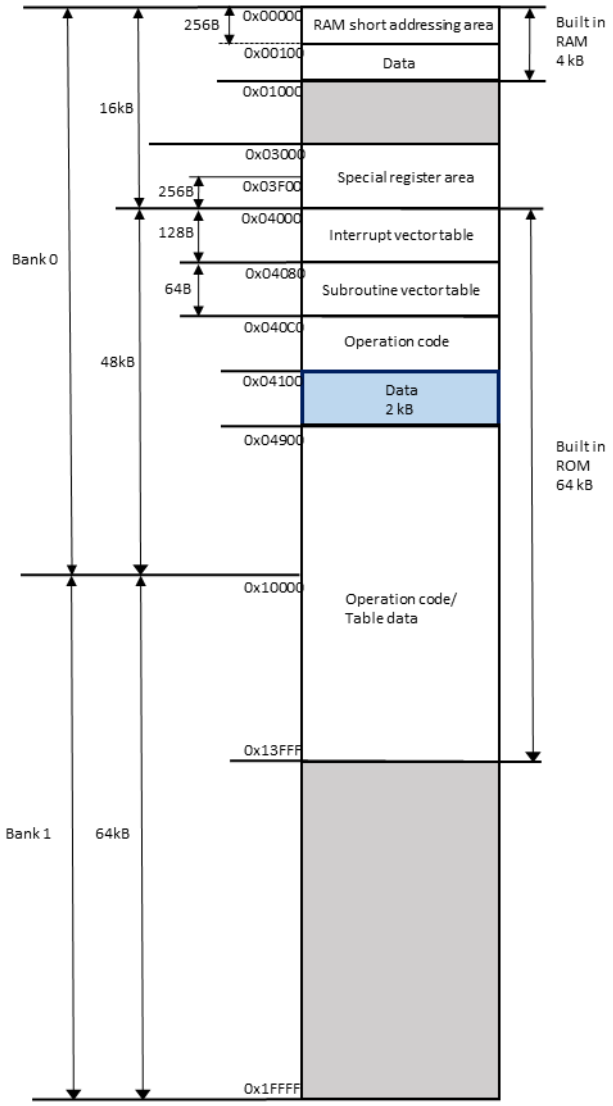


Fig 1.4 Memory layout

1.4 Development environment

This section describes how to build a development environment using "Debug Factory Builder".
Download and install "Debug Factory Builder" from the following URL.

https://b2bsol.panasonic.biz/semi-spt/

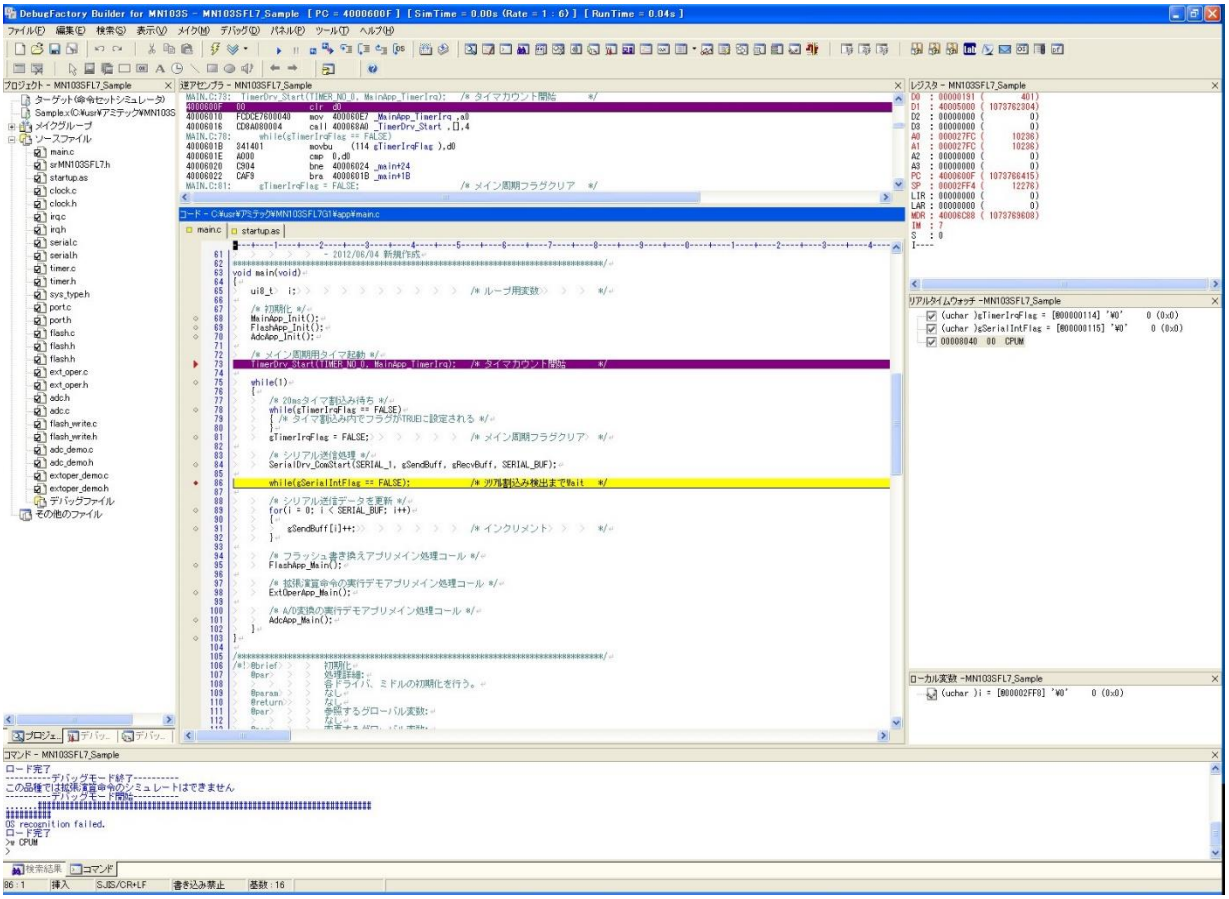


Fig 1.5 Debug Factory Builder startup screen

1.4.1 File structure and function list

The sample software has the following file / folder structure, and double-clicking ReRAM_Sample.df5 will start the project.

```
ReRAM_memwrite_sample_05D
├─prj
│   └─ReRAM_Sample.df5
├─build  (store make results)
└─src
    ├─Start1.asm   (Sample program startup assembler)
    ├─Test1.c      (Sample program body (main function))
    ├─rer_wdat.c   (ReRAM data area rewriting library body)
    ├─rer_type.h   (ReRAM rewriting library external declarations file)
    ├─rer_extn.h   (ReRAM rewriting library constant definition file)
    └─rer_defs.h   (ReRAM rewriting library type declaration file)
```

Fig 1.6 File structure

1.4.2 Target setting

“Debug (D)” ⇒ Set from the “Target setting(F)”.
“Product type (P)” : 101LR05D
“Type (T)” : Select “Starter kit”
Select "srKM101LR05D.txt" in the src folder in the "Special Register Definition file (E)".

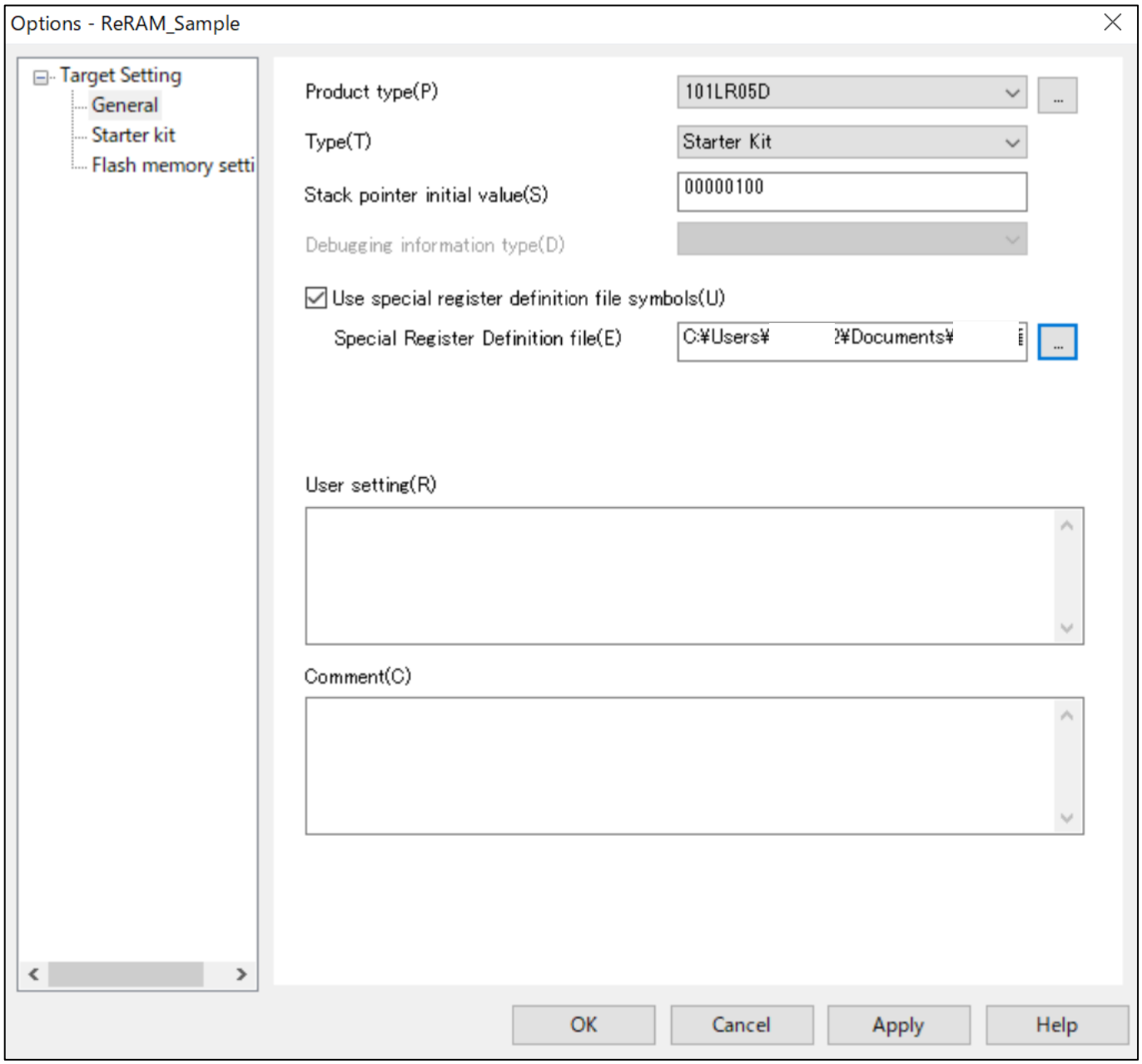


Fig 0.1 Target setting

1.4.3 Compiler option

Make (M) ⇒ Set in the "Compiler option (O)".

When developing software, it is necessary to change "Compiler option (O)", add folders, or add optimization options, when using the standard library.

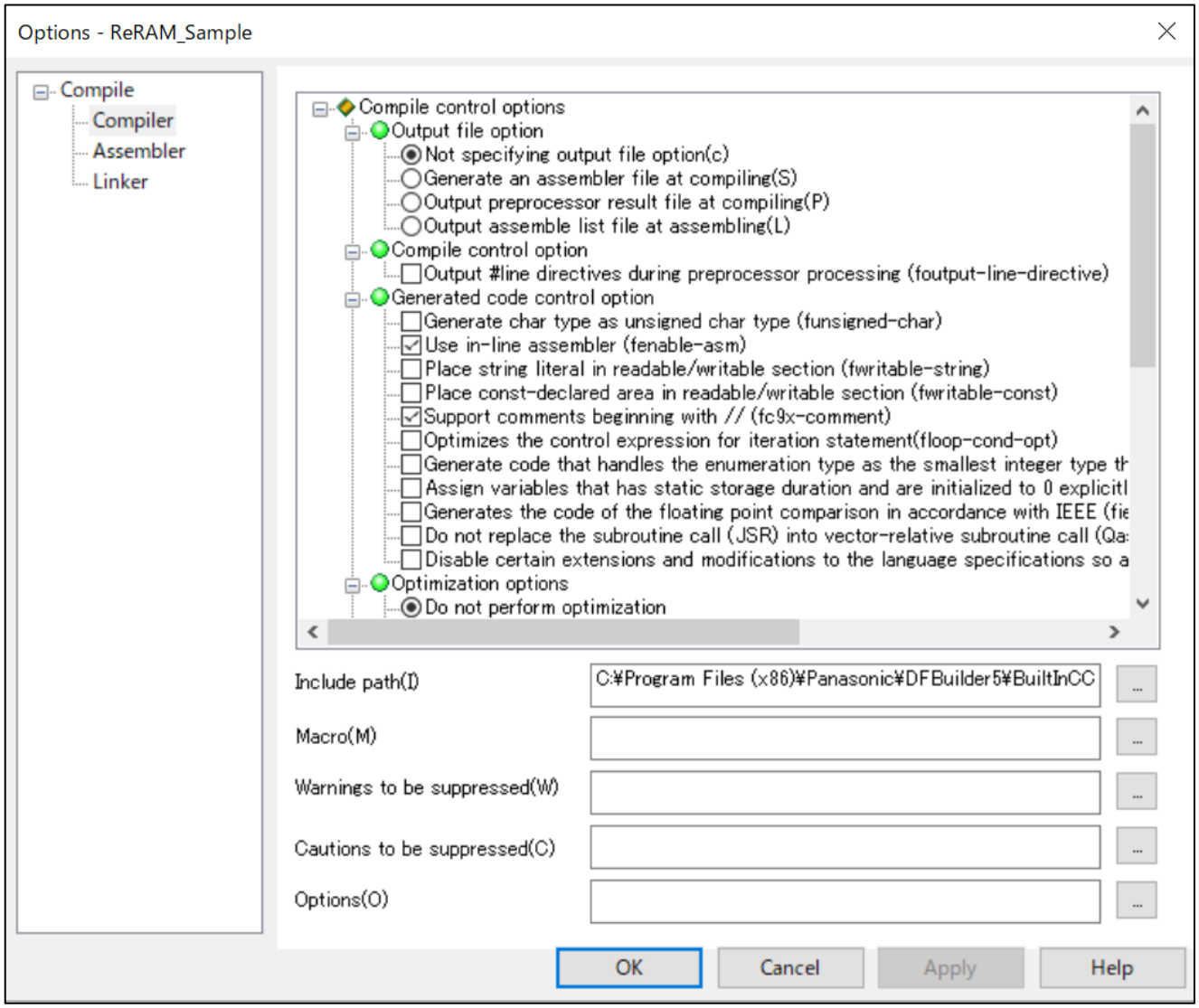


Fig 0.2 Compiler option

1.4.4 Development environment

Make (M) ⇒ Set in the "Environment settings (E)".
Set when changing the compiler, outputting the object file, converting the file to HEX format, etc.

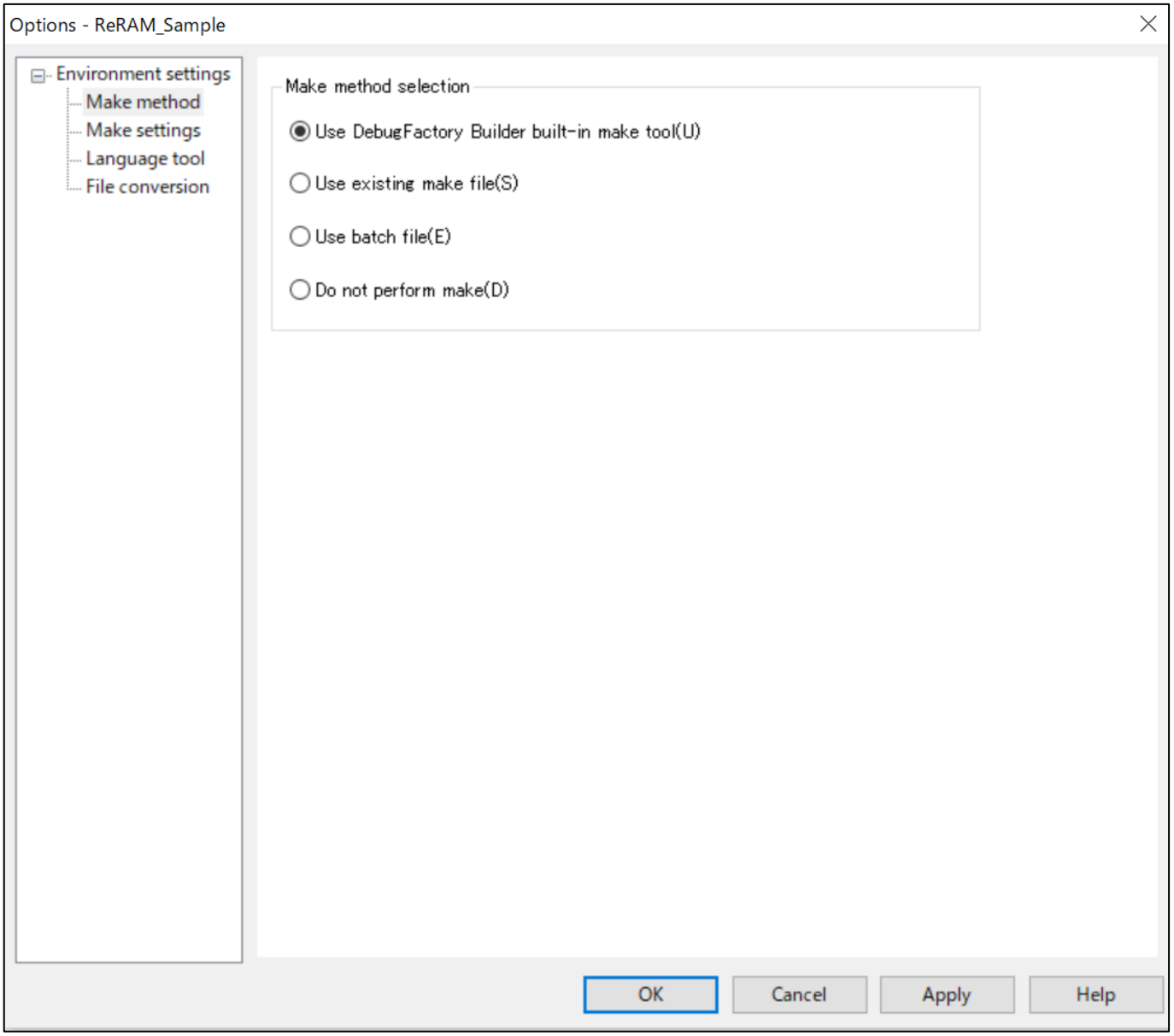


Fig 0.3 Environment settings

Chapter 2 Software processing content

2.1 System layer

2.1.1 Startup process（Startup.asm）

The processing performed in the startup process is as follows.

- Interrupt vector settings
- Stack pointer setting
- RAM initialization

After the above process is completed, branch to "main".

2.1.1.1 Interrupt vector settings

Reset start

Processing starts from address 0x4000. (Branch to "Reset" processing.)

```
_STEXT          SECTION             ;
                da      A(Reset)    ; 0 : reset vector address
```

Maskable interrupt

Processing according to each interrupt vector is started from each interrupt vector located at addresses 0x4008 to 0x4078. (The function placed in each interrupt is executed.)

```
                da      A(NoIRQ)            ; 0x4008 : IRQ0 : Reserved
                da      A(NoIRQ)            ; 0x400c : IRQ1 : Reserved
                da      A(NoIRQ)            ; 0x4010 : IRQ2 : Reserved
                da      A(NoIRQ)            ; 0x4014 : IRQ3 : Reserved
                da      A(NoIRQ)            ; 0x4018 : IRQ4 : Reserved
                da      A(NoIRQ)            ; 0x401c : IRQ5 : Reserved
                da      A(NoIRQ)            ; 0x4020 : IRQ6 : Reserved
                da      A(NoIRQ)            ; 0x4024 : IRQ7 : Reserved
                da      A(NoIRQ)            ; 0x4028 : TM0ICR : Reserved
                da      A(_vTIM_SysTimIrq)  ; 0x402c : TM1ICR
                da      A(NoIRQ)            ; 0x4030 : TM2ICR : Reserved
                        --abridgement--
```

2.1.1.2 Stack pointer setting

When reset start, the stack pointer is initialized.
Because RAM area of the microcomputer is of 0x0000 ~ 0x0FFF (4KB), and 0x1000 is set in the SP register.

```
Reset：
        movw    0x1000,A0       ；initialization of stack pointer
        movw    A0,SP           ；
```

2.1.1.3 RAM initialization

Startup.asm initializes the RAM that has not been the setting of the initial value.

```
；initialization of static variables
        movw    _BSS,A0         ；set start address of _BSS domain to A0 register
        sub     D0,D0           ；D0 = 0
clear1：
        mov     D0,(A0)
        addw    0x1,A0
        cmpw    _BSSEND,A0      ；Is it the last of a RAM domain?
        blt     clear1          ；if _BSSEND > A0, jump to clear1
；Initialization of static variable with default value
raminit：
        movw    _ROMDATA,A0
        movw    _DATA,A1
        cmpw    _GROMDATA,A0
        beq     next1
init1：                         ；ROMDATA(A0) -> DATA(A1)
        mov     (A0),D0
        mov     D0,(A1)
        addw    0x1,A0
        addw    0x1,A1
        cmpw    _GROMDATA,A0
        bne     init1
next1：
        movw    _GROMDATA,A0
        movw    _GDATA,A1
        cmpw    _TEXT,A0
        beq     next2
                (略)
```

Startup.asm initializes only the RAM for the sections defined by default. If the user defines additional sections, it is necessary to add initialization processing.

2.2 Application layer

2.2.1 main processing (main.c)

The outline of the "main" function is shown. It is called after the initialization process, and it is completed to initialize and execute each module.

This sample program executes the processing in the following order.

- First, the sample software writes 0xff to the entire ReRAM data area (0x4100 to 0x48ff) in the main function.
- Then, The sample software writes 0xa5,0x5a, 0x55,0x00 from 0x4100 to 4 bytes.
- After the above write process, the sample software will check if 0xa5,0x5a, 0x55,0x00 are correctly written from 0x4100 to 0x4102.
- As a result of the above confirmation, if it is written correctly, the sample software will blink the LED-D5 (green LED) mounted on the AM13L-STK2.
- As a result of the above confirmation, if it is not written correctly, the sample software will blink the LED-D4 (orange LED) mounted on AM13L-STK2.

2.3 Driver layer

2.3.1 Rewrite library (rer_wdat.c)

This section describes the library function when rewriting ReRAM.
The sample software uses an internal operating voltage of 1.8V and an operating clock of the built-in high-speed clock when rewriting ReRAM.

Interface function

```
Function name  :  pReRAM_set_data
Argument       : U32 dst_adrs , U16 src_adrs , U16 length
Return value  : Error code
      0 (RC_OK) : If the process is successful
      1 (RC_FALSE_ADRS) : If the address and size are
        incorrect
      3 (RC_FALSE_LIB) : Failure to ReRAM writing
      process
        ※RC_: Constants are defined in rer_extn.h
Function       : Rewriting Re-RAM
        This function writes from dst_adrs to the ReRAM data area for the length bytes.
        When using this function, specify dst_adrs in the range of 0x4100 to 0x48ff with an
        even address, and specify the length with an even number.
        Also, specify the address where the data you want to write is stored in src_adrs.
Example :
       This example writes 0x55,0xaa, 0xa5,0x5a data to a ReRAM data area of 0x4100
       to 4 bytes.


        U8 ret, data[4];
        data[0] = 0x55; data[1] = 0xaa; data[2] = 0xa5; data[3] = 0x5a;
    ret = pReRAM_set_data(0x4100, (U16)&data[0], 4);
    if (ret != RC_OK) {
                /* In case of error */
                        :
    }
```

- This environment is provided as a sample of KM101LR05D built-in ReRAM rewriting.
  When actually using it, it is necessary to change it according to the environment.
- The ReRAM area that can be written by this ReRAM rewrite library API is the data area (0x4100 to 0x48ff).
  Rewriting of the code area is not supported.
- When using the ReRAM rewrite library, it uses a 16-byte stack. Therefore, allow 16 bytes for the stack.

Revision history

Details of revision from Ver.1.0 to Ver.1.1 in this manual is shown below.
According to the details of revision, "Definition" of the table below is classified into seven groups.
Revision concerning descriptions in this Manual:
Writing error correction / Description change / Description addition / Description deletion
Revision concerning specifications:
Specification change / Specification addition / Specification deletion

Page	Definition	Details of revision	
		Ver1.0	Ver1.1
Cover	Description addition	---	Addition of information
P2	Description deletion	Deletion of information	---
Last page	Description addition	---	Addition of Important Notice

Inquiries

If you have questions regarding technical information on this manual, please visit the following URL.

Nuvoton Technology Corporation Japan

URL: https://www.nuvoton.co.jp/en/contact/

・Microcomputer Home Page

https://www.nuvoton.co.jp/en/products/

KM101LR05D
ReRAM Memory Rewriting
Application Note

October 30, 2020 1st Edition 1st Printing

Issued by
Nuvoton Technology Corporation Japan

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.