LogiCoA™ power solutions

Operating System for Switching Power Control MCU "RMOS"

Overview

To control a switching power supply unit using a microcontroller, it is necessary to familiarize yourself with the functions (digital PWM, A/D converters, communication interfaces, data flash memories, etc.) installed in the microcontroller and perform detailed control. In addition, large-scale programming is required for multi-function and highly controlled switching power supplies. Furthermore, when two types of power supply circuits (for example, PFC and DCDC converters) are controlled concurrently by a single microcontroller, it is necessary to consider the independence and coordination of the respective power supply circuits, which dramatically increases the difficulty of programming.

To solve these problems, we will provide an operating system RMOS (**R**eal time **M**icro **O**perating **S**ystem) for a switching power supply control microcontroller. RMOS operates on a ML62Q203x/ML62Q204x (ML62Q20xx grouping) (as of April 1, 2024). The features of RMOS are described below.

①Operating system for multitasking and real-time control
②Improved program simplicity and readability (debuggablity) by modularizing control programs
③Implemented basic functions for switching power supply control (e.g., initial setting of MCU peripherals, acquisition of AD value, status monitoring, communication function, logging function (※1))
(※1) Currently, logging function is not implemented.
④Monitoring the state of the power supply circuit and outputting "state variable" and "state flag" (executed by background processing)
⑤Two types of switching power supply circuits are simultaneously controlled by a single MCU
⑥Switching power supply control programming using "state transition control" is supported (state transition managing function is implemented in RMOS)
⑦Implements a low power operation mode that reduces the power consumption of the MCU while the switching power supply is in standby mode

To bring out 100% of the performance of the MCU, RMOS uses assembly language (a language that replaces machines that can be directly executed by CPU with English words that can be easily understood by a human) to write power control programs. You may think that programming in assembly language is difficult, but using RMOS makes it easy to write programs in assembly language.

In many cases, a power supply control program using RMOS can be programmed by describing the process for a state variable (e.g., a two-times moving average of the output voltage AD value) or a state flag (e.g., a flag that indicates whether RC pin is enabled or disabled). So that compared to other languages, difficulty of programming is low enough. In addition, since a reference program that can control various power supply circuits at a practical level can be provided, a power supply control program can be developed by making small changes to the reference program.

This manual explains RMOS functions, operations, and how to use them.

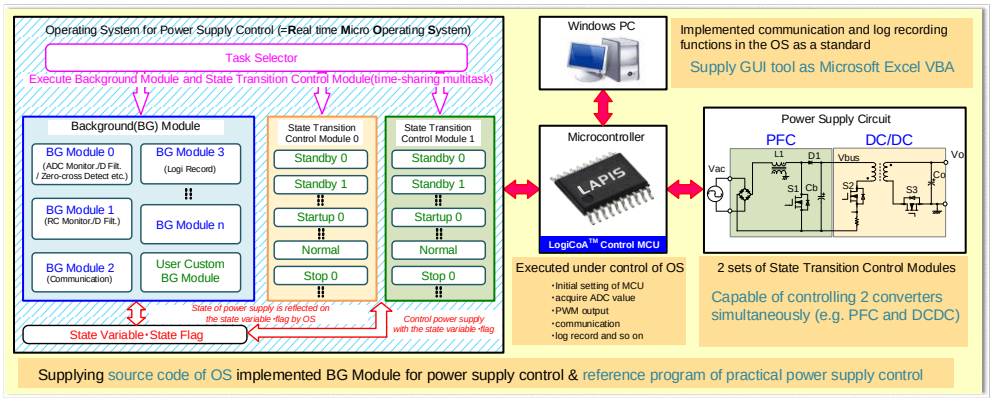


Figure. Structure and Function of Operating System for Switching Power Supply "RMOS"

No. 66AN147E Rev.001
June 2024

Table of Contains

1. Control of Switching Power Supply with RMOS (State Transition Control) 3
2. Multitasking Real-time Control of RMOS 4
3. Usage of RMOS 5
3-1. Software and Equipment 5
3-2. How to Read RMOS Projects and Use LEXIDE-Ω 5
4. Structure and Description of RMOS Projects 10
4-1. "RMOS100.ASM" File (default executable file) 10
4-2. "0_System" Folder (MCU and RMOS system setting) 10
4-3. "10_Setting" Folders (MCU hardware setting) 12
4-4. "30_Info_module" Folder (communication control module) 14
4-5. "50_BG_module" Folder (background modules) 14
4-6. "51_BG_Control_include" Folder (include files for background modules) 15
4-7. "55_BGLp_module" Folder (background modules for low power mode) 15
4-8. "70_PS0_State_Control_module" Folder (state transition control module group 0) 15
4-9. "71_PS0_Control_include" Folder (include file for state transition control module group 0) 16
4-10. "80_PS1_State_Control_module", "81_PS0_Control_include" Folder (state transition control module group 1) 16
4-11. "90_Header" Folder (header files) 16
5. Programming with RMOS 17
5-1. Resources of MCU Occupied by RMOS 17
5-2. Function to be Background Processed by RMOS 17
5-3 State Variables and State Flags 18
5-4. Low Power Operation Mode 19
5-5. Debugging Activities (breaking CPU, stepping, checking variables) 20
5-6. Checking the Allowable Time for Task Assignment (allowable count value) 21
5-7. Detection of Task Incomplete by RMOS 22
6. List of State Variables and State Flags 23
7. Reference Documentation 24

1. Control of Switching Power Supply with RMOS (State Transition Control)

RMOS implements a function that controls the switching power supply using "state transition control." Following describes the switching power supply state transition control.

The operation states of switching power supplies can be classified into following four operating states:

(1)Standby operation ・・・The switching power supply unit is not outputting a voltage (the input voltage is below the startup voltage and halted by remote ON/OFF control).
(2)Start operation ・・・The output voltage of the switching power supply rises from zero to a steady voltage.
(3)Normal operation ・・・When the output voltage of the switching power supply unit outputs a steady voltage.
(4)Stop operation ・・・Stops the output voltage of the switching power supply.

Since the above operation states are independent in the operation of the switching power supply unit, the control programs can also be described independently for each operation state. Programs can be written without considering other operating states, so programs can be simplified and written. In addition, when describing a control program, the above operation state is further subdivided, and the program is modularized for the operation state of the power supply (state transition control module). Then, "Change (transition) the program module to be executed according to the state of the power supply" control is performed (state transition control).

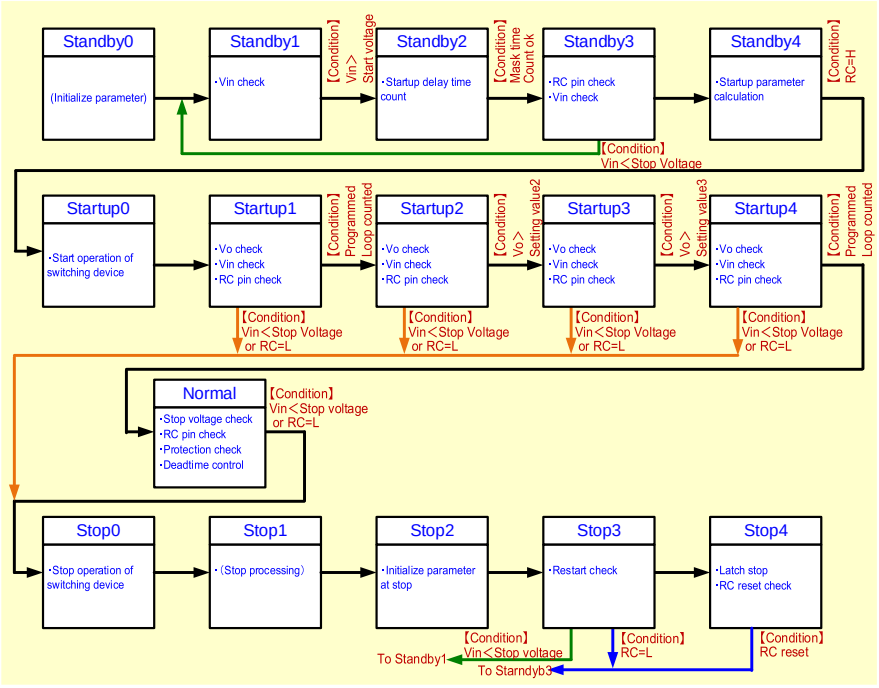


Figure 1-1. Example of the state transition diagram

In RMOS, the state transition control module description area is prepared (standby × 5, startup × 5, normal × 2, stop × 5). Programmers can design power control programs by writing programs in RMOS state transition control module description area. RMOS executes any of the specified state transition control modules every 50μs. The state transition control module to be executed can also be changed (transited) by issuing an instruction to RMOS according to the power supply state. RMOS also provides two sets of state transition control modules to control two types of switching power supply circuits simultaneously.

In RMOS, power supply control is performed by transitioning the execution of the state transition control module, so the configuration of the program is represented by the state transition diagram. The diagram describes the operation, transition conditions, and transition destination of each state transition module. When developing a program, the programmer first creates a state transition diagram and then examines the operation of the power supply in each state to prevent control missing.

2. Multitasking Real-time Control of RMOS

RMOS is an operating system that has the ability to process several tasks (programs) in parallel (multitasking). RMOS multitasking control uses preemptive multitasking (time-sharing method), which ensures realtimeness (program execution at fixed intervals). With real-time control, the operating time of the control program can be precisely designed (e.g. the time from when the input voltage reaches the start voltage until the power is started). Normally this should not be done, but even if any task runs out of control, other tasks can operate normally. The following is a summary of the multitasking real-time control of RMOS.

(1) In RMOS, the "Task Selector", which is the program that controls multitasking, and the "Background Modules" and "State Transition Control Modules" which are the program modules that are the targets of multitasking control are prepared.
(2) The programmer writes the program in the "program description area" provided in the "background module" and "state transition control module".
(3) The "Task Selector" executes one of the program modules at a predetermined cycle (e.g. 25µs, 50µs, etc). When the specified time (e.g. 7.5µs, 9.5µs, etc) has elapsed, another program module is executed.
(4) The execution cycle and allocation cycle of the program module are predetermined by RMOS.
(5) The "background module" executes all modules in parallel.
(6) In the "State Transition Control Module", one module is executed according to the state of the power supply, but only in normal operation, two modules are executed in parallel.
(7) The state transition control module to be executed is indicated to RMOS in the programming.

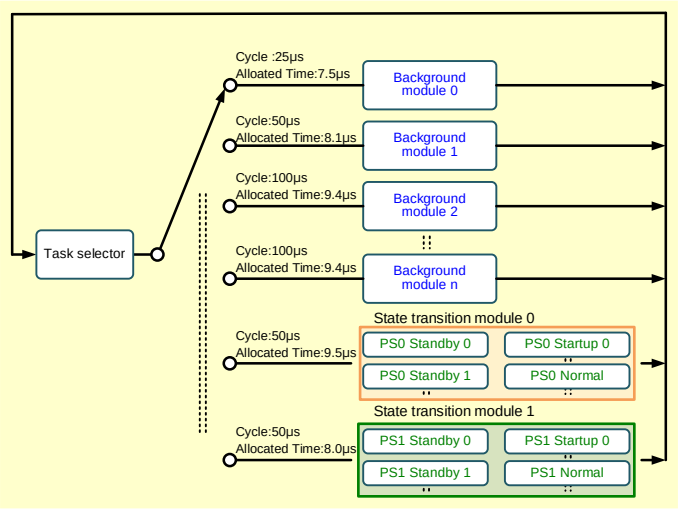


Figure 2-1. Multitasking real-time control of RMOS

Table 2-1 shows the background modules that are prepared in RMOS. Table 2-2 2-3 shows the state transition modules.

Table 2-1. Background modules

No.	Module Name	Period	Allocated Time
1	BG25u_Task	25µs	7.5µs
2	BG50u_Task	50µs	8.13µs
3	BG100u_Task	100µs	9.38µs
4	BG500u_Task	500µs	9.38µs
5	BG1m_Task0	1ms	9.38µs
6	BG1m_Task1	↑	↑
7	BG10m_Task0	10ms	↑
8	BG10m_Task1	↑	↑
9	BG25m_Task0	25ms	↑
10	BG25m_Task1	↑	↑
11	BG100m_Task0	100ms	↑
12	BG100m_Task1	↑	↑
13	BG100m_Task2	↑	↑
14	BG100m_Task3	↑	↑
15	BG100m_Task4	↑	↑
16	BG500m_Task0	500ms	↑
17	BG500m_Task1	↑	↑
18	BG500m_Task2	↑	↑
19	BG500m_Task3	↑	↑
20	BG500m_Task4	↑	↑
21	BG1000m_Task0	1000ms	↑
22	BG1000m_Task1	↑	↑
23	BG1000m_Task2	↑	↑
24	BG1000m_Task3	↑	↑
25	BG1000m_Task4	↑	↑

Table 2-2. State transition module 0

No.	Module Name	Operating State	Allocated Time
1	PS0_Standby_0	Standby	9.50µs
2	PS0_Standby_1		↑
3	PS0_Standby_2		↑
4	PS0_Standby_3		↑
5	PS0_Standby_4		↑
6	PS0_Startup_0	Startup	9.50µs
7	PS0_Startup_1		↑
8	PS0_Startup_2		↑
9	PS0_Startup_3		↑
10	PS0_Startup_4		↑
11	PS0_Normal_50u	Normal	9.50µs
12	PS0_Normal_500u		9.38µs
13	PS0_Stop_0	Stop	9.50µs
14	PS0_Stop_1		↑
15	PS0_Stop_2		↑
16	PS0_Stop_3		↑
17	PS0_Stop_4		↑

*1) Control Cycle: 50µs
*2) Any one module corresponding to the state of power supply is executed. Only PS0_Normal_50u and PS0_Normal_500u are executed concurrently.

Table 2-3. State transition module 1

No.	Module Name	Operating State	Allocated Time
1	PS1_Standby_0	Standby	8.00µs
2	PS1_Standby_1		↑
3	PS1_Standby_2		↑
4	PS0_Standby_3		↑
5	PS1_Standby_4		↑
6	PS1_Startup_0	Startup	8.00µs
7	PS1_Startup_1		↑
8	PS1_Startup_2		↑
9	PS1_Startup_3		↑
10	PS1_Startup_4		↑
11	PS1_Normal_50u	Normal	8.00µs
12	PS1_Normal_500u		9.38µs
13	PS1_Stop_0	Stop	8.00µs
14	PS1_Stop_1		↑
15	PS1_Stop_2		↑
16	PS1_Stop_3		↑
17	PS1_Stop_4		↑

*1) Control Cycle: 50µs
*2) Any one module corresponding to the state of power supply is executed. Only PS1_Normal_50u and PS1_Normal_500u are executed concurrently

3. Usage of RMOS

3-1. Software and Equipment

To develop a program for switching power supply control that uses RMOS, use

(1) Integrated Development Environment LEXIDE-Ω
(2) RMOS project file (file to be read into LEXIDE-Ω and used)
(3) Windows PC (Windows10 64bitversion or Windows11 64bit version)
(4) On-chip emulator EASE1000 V2
(5) Serial communication module: FT234x FTDI Corporation
(6) Microsoft Excel 64bit version (used to check the communication function and requires permission to use the macro function)

"Integrated Development Environment LEXIDE-Ω" is a software developed based on "Eclipse," an integrated development environment for open sources, and installed on a PC for use. The installers can be downloaded from our web website.

"RMOS project file" is provided in a zip compressed format, and extracted to the any folder in the HDD (SSD) drive of Windows PC

The "Serial Communication Module" is mounted on a buck DCDC converter (hereafter buck converter) EVK LogiCoA001-EVK-001 (hereafter buck converter EVK). It is used for serial communication between PC and ML62Q203x/ML62Q204x (ML62Q20xx group).

The on-chip emulator is designed so that only 3.3V can be supplied to the MCU as the power supply for operation during MCU debugging. Therefore, a separate 5V power supply must be prepared for debugging ML62Q20xx groups (power supply voltage range: 4.5 to 5.5V). So, 5V power supply is supplied from the PC via the serial communication module. However, the jumper setting positions of the on-chip emulator and buck converter EVK differ depending on the use.

3-2. How to Read RMOS Projects and Use LEXIDE-Ω

The following describes how to use the buck converter EVK with ML62Q2035 to read LEXIDE-Ω RMOS project files, write programs to the MCU ML62Q20xx group on the buck converter EVK, execute/stop programs, and program RMOS.

Refer to LEXIDE-Ω Release Notes [1] and LEXIDE-Ω User's Manual [2] for information on how to install and use LEXIDE-Ω on PC. Also refer to the Buck Converter EVK User's Guide [3] for instructions on how to use the Buck Converter EVK (e.g. how to connect power, loads, etc.) and how to explain the control program.

(1) Connection method

a. Connection method for MCU writing and debugging

Connect the buck converter EVK, 14pin connector of the on-chip emulator, and the flat cable as shown in Figure 3-1. Set JP_REG on the left side of the buck converter EVK (power is supplied from FT234x when writing the program to the MCU) and leave JP_LDO open (to prevent reverse current from FT234x to LDO output pins).

Connect PC to the on-chip emulator in the order ① and ② using ①USBMiniB cable between the on-chip emulator and ② USBMicroB and between the serial communication module and PC.

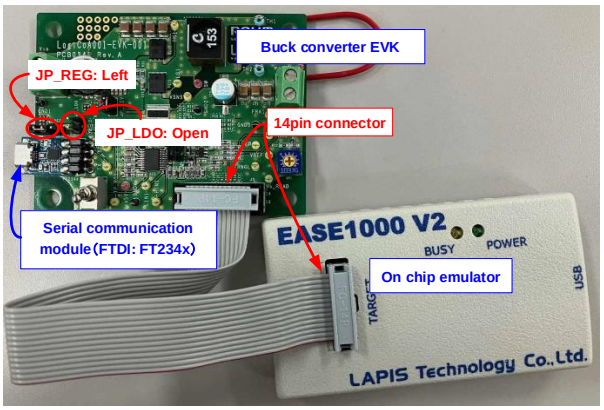


Figure 3-1. Connection for MCU Write and Debug

b. How to connect buck converter EVK in normal operation

During normal operation of the buck converter EVK, connect as shown in Figure 3-2. During normal operation, the serial communication module is used to communicate between PC and the MCU. At this time, the power, 5V, of the MCU is supplied from the voltage applied to the input voltage pin through the regulator IC. Set JP_REG to the right for the jumper on the buck converter EVK, and short JP_LDO. Do not connect the on-chip emulator when setting this jumper. Otherwise, external voltage may be applied to 3.3V output pin of the on-chip emulator, causing damage.

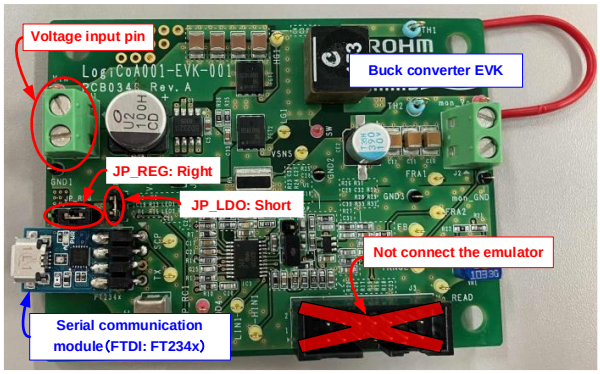


Figure 3-2. Connection during Buck Converter EVK Normal Operation

c. Debugging when input voltage is applied to buck converter EVK

When debugging the buck converter EVK with LEXIDE-Ω, connect as shown in Figure 3-3. The jumper on the buck converter EVK should open JP_REG and short JP_LDO. If the jumper of JP_REG is set to the left or right position, external voltage may be applied to the 3.3V output pin of the on-chip emulator, causing damage.

Serial communication between PC and MCU during debugging can be performed without problems as long as the programs are running using LEXIDE-Ω, even if the connection in Figure 3-3.

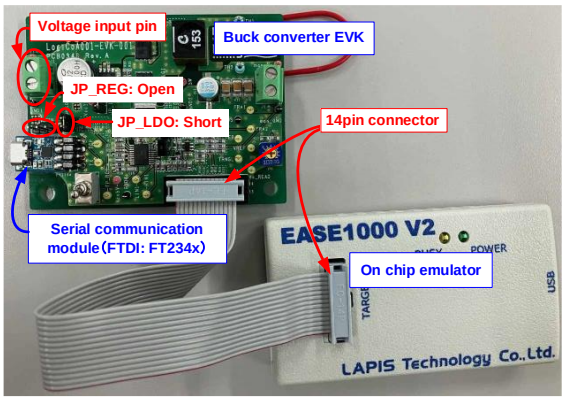


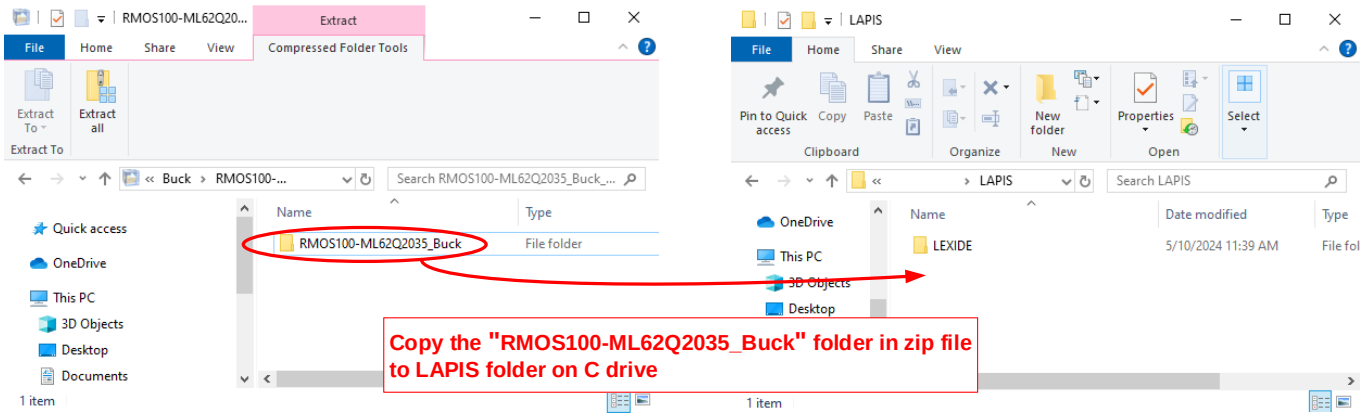
Figure 3-3. Connection for debug on applying input voltage to Buck Converter EVK

(2) Copying RMOS Project Files (Folders) for Buck Converter EVK

The RMOS100-ML62Q2035_Buck.zip is provided as a RMOS project file for the buck converter EVK.

When you open the "RMOS100-ML62Q2035_Buck.zip" in Windows Explorer (double-click the RMOS100-ML62Q2035_Buck.zip file), the content of zip file (RMOS100-ML62Q2035_Buck folder) is displayed as shown in Figure 3-4 (a).

In the PC of installing LEXIDE-Ω, "LAPIS" folder has been generated just under C drive ("LEXIDE" folder has been generated in "LAPIS" folder). Copy the "RMOS100-ML62Q2035_Buck folder" in zip file to the "LAPIS" folder.



(a)Opened RMOS100-ML62Q2035_Buck.zip (b)Opend LAPIS folder in C drive

Figure 3-4. How to Copy RMOS100-ML62Q2035_Buck Folders

(※) RMOS project files (folders) can also be copied to any folder (other than LEXIDE folder).

(3) Activation of LEXIDE-Ω

Performing steps ① to ⑨ below activate LEXIDE-Ω.

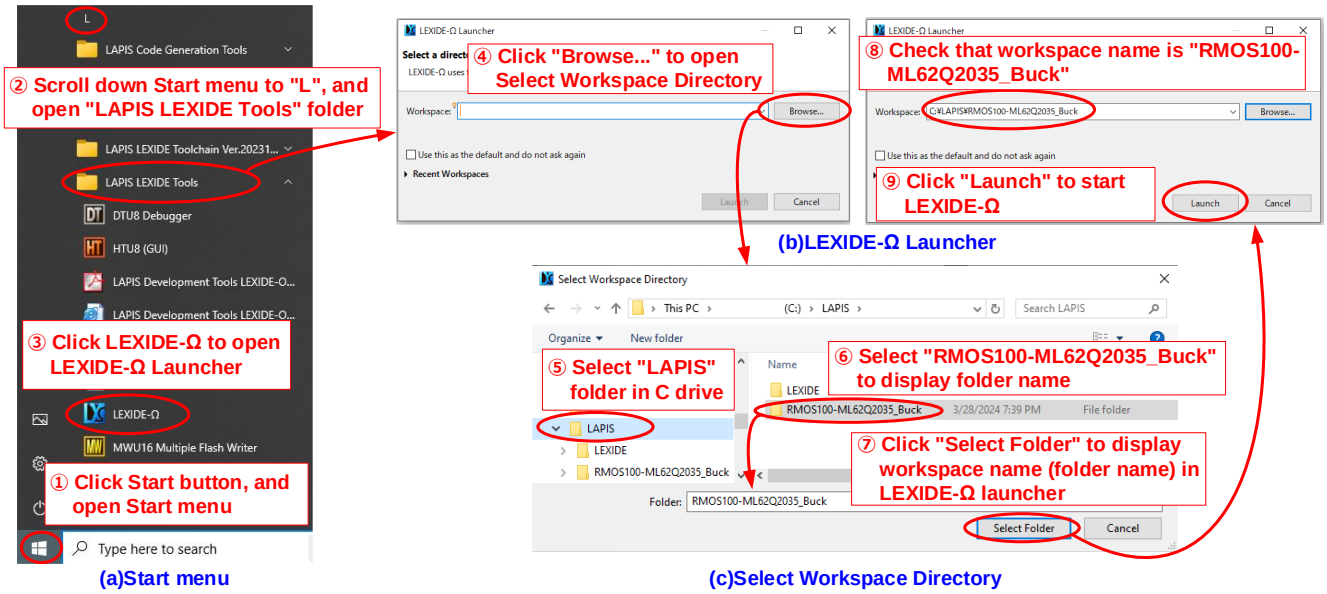


Figure 3-5. Activation of LEXIDE-Ω

① Click "Start Icon" on Windows to bring up the Start menu (Figure 3-5a).
② Open LAPIS LEXIDE Tools folders by scrolling to the L group in the Start menu.
③ LEXIDE-Ω launcher is activated by clicking the "LEXIDE-Ω" icon (Figure 3-5b).
④ Open the workspace selection window (Figure 3-5c) by clicking the "Browse pushbutton" on LEXIDE-Ω launcher.
⑤ In the workspace selection window, select "LAPIS" folders on drive C.
⑥ In the workspace selection window, select the "RMOS100-ML62Q2035_Buck" folder.
⑦ Clicking the "Select Folder" button in the workspace selection window displays the workspace name (folder name) in LEXIDE-Ω launcher.
⑧ Make sure that the workspace designation of LEXIDE-Ω launcher is "RMOS100-ML62Q2035_Buck".
⑨ LEXIDE-Ω is activated by clicking the "Launch" button LEXIDE-Ω launcher.

In addition, "LAPIS LEXIDE Software Tools Document" folder in the "L group" in ② also has LEXIDE-Ω User's Manual [2] and the Instruction Manual in Assembly Language [4]. See here for more information on using LEXIDE-Ω and describing the assembly language.

(4) Basic operation of LEXIDE-Ω

When LEXIDE-Ω is activated, Figure 3-6 window is displayed. RMOS is programmed and debugged with the "Debug" icon selected in the upper-right corner.

RMOS program modules (background modules, state transition modules, etc.) are displayed in the Project Explorer at the bottom left of the window. Here you can select the program module to be edited. You can edit the program source code at the bottom center of the screen. When the execution of the MCU program is paused, the internal information of the MCU is displayed at the lower right of the screen.

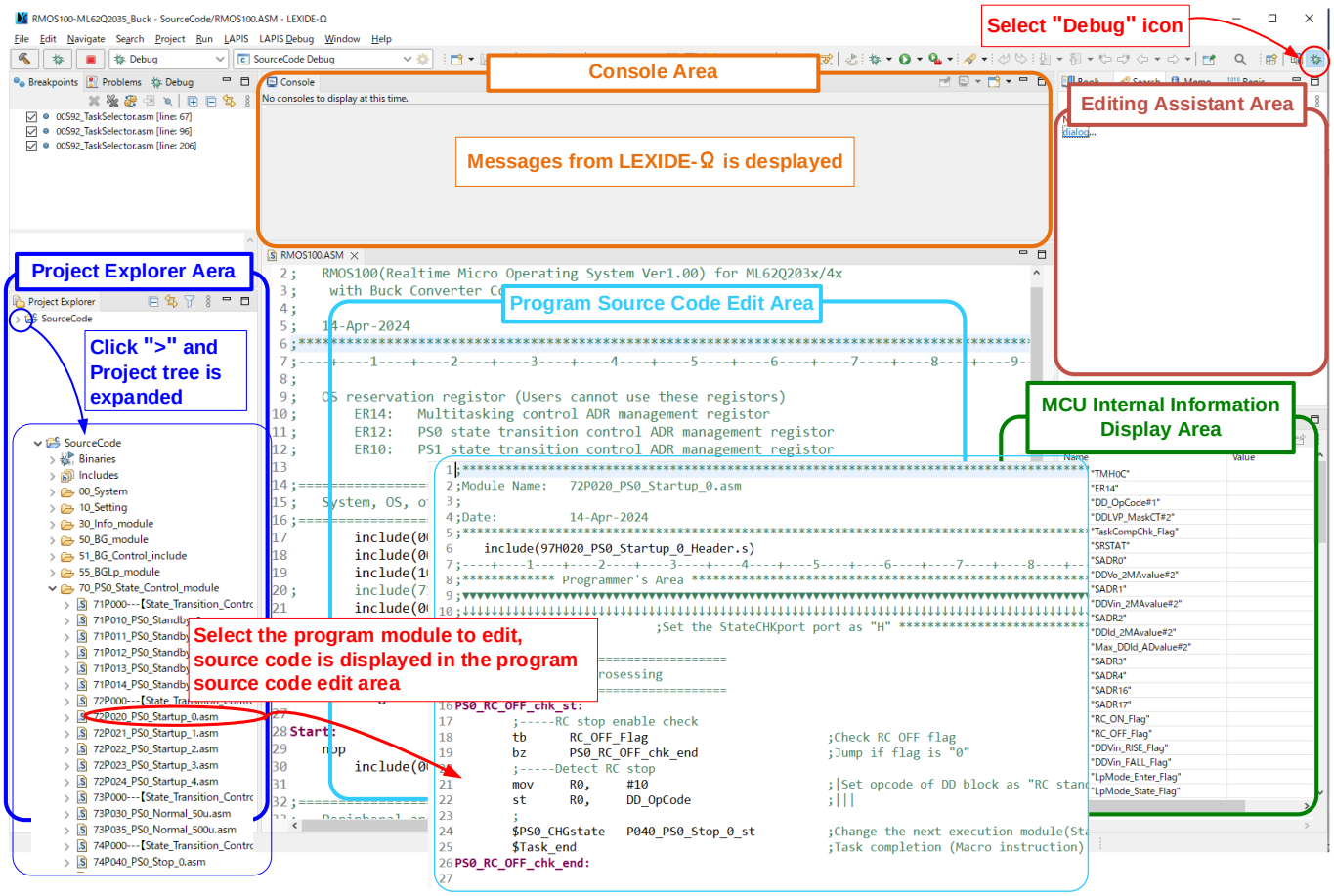


Figure 3-6. Screen when RMOS project file for buck converter EVK is started with LEXIDE-Ω

(5) Building a Program

"Build" is used to convert a program created on LEXIDE-Ω into a program that can be written to a MCU on the buck converter EVK. To build, perform steps ① to ③ below as shown in Figure 3-7.

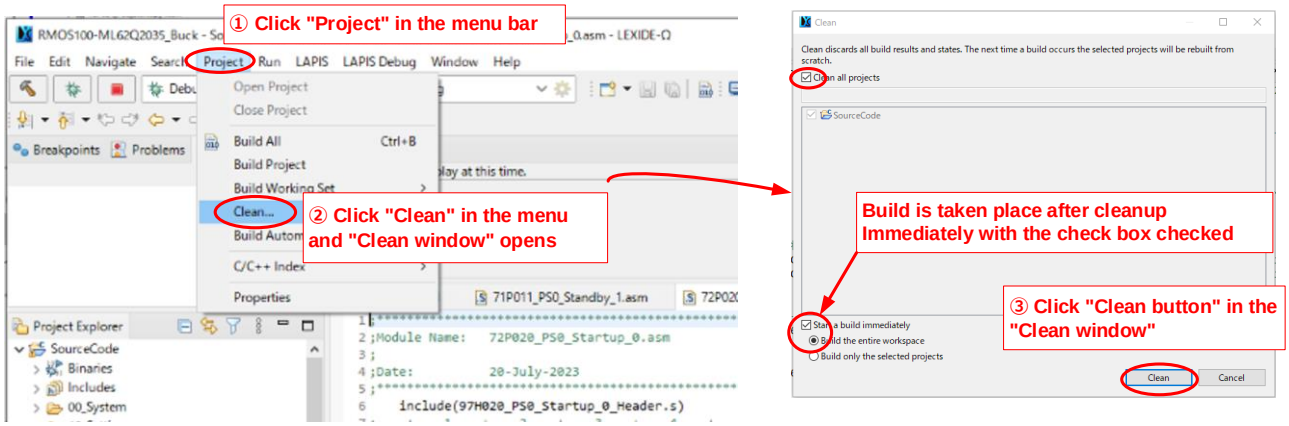


Figure 3-7. Build operation

① The menu window is displayed by clicking "Project" on the menu bar of LEXIDE-Ω.
② Clicking "Clean" in the menu window opens the "Clean window".
③ Clicking "Clean button" in "Clean window" starts the build process. Build is taken place after cleanup immediately with the checkbox of "Start a build immediately" checked.

When the build starts, the console area displays a message indicating the progress of the build process.

When scrolling of the messages displayed in the console area is stopped and the message "Build Finished. 0 errors, 0 warnings" is displayed, the build has been completed successfully.

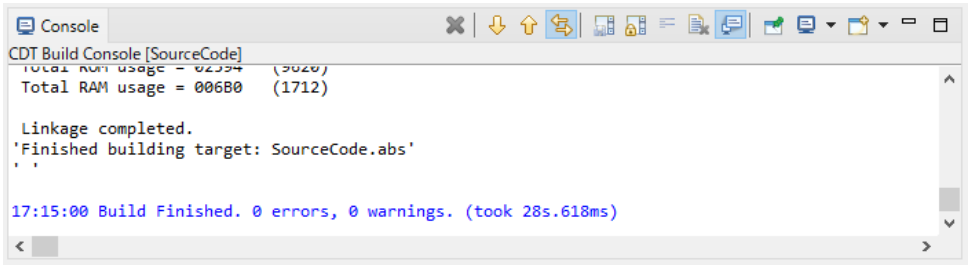


Figure 3-8. Viewing the Console Area When a Buil Completes Successfully

(6) Writing to the MCU, executing and pausing programs

To write to the MCU, execute or pause the program, perform steps ① to ⑤ below.

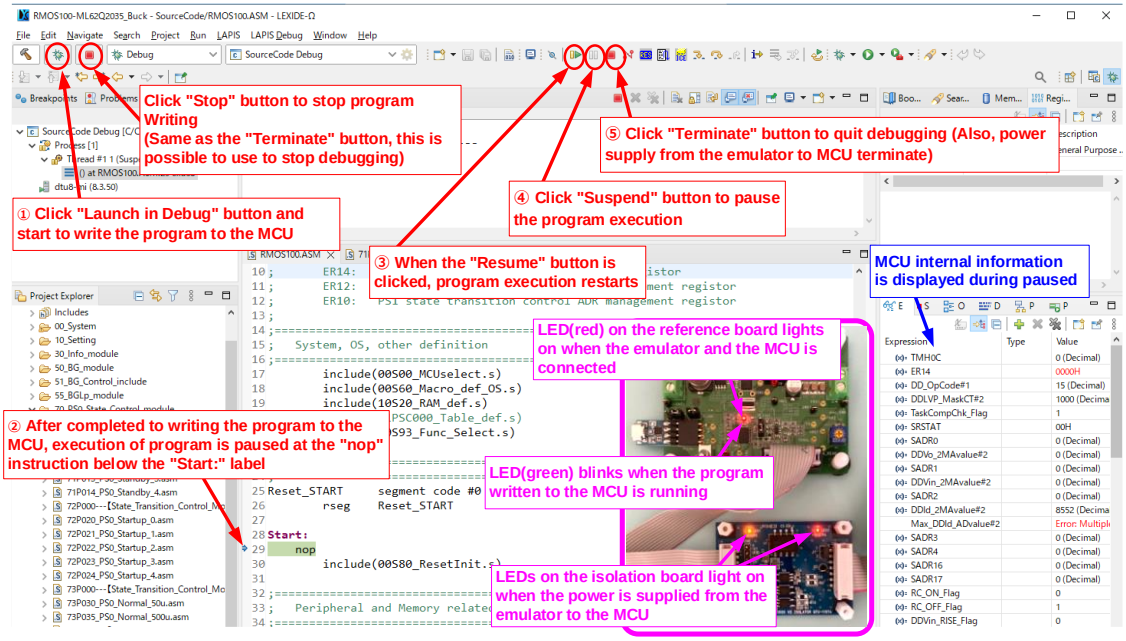


Figure 3-9. Programming, Executing and Pausing

①Clicking "Launch in Debug" starts writing the program to the MCU. At this time, power is supplied from the serial communication module to the MCU, so LED (yellow) of the on-chip emulator blinks momentarily and LED (red) on the buck converter EVK lights up. When USB cable is connected to PC, LED (blue) on the serial communication module is always on.
②When the program is written to the MCU, the program is paused at the state of the "nop" instruction below the "Start:" label.
③Clicking "Resume" executes (resumes) the program. When the program is executed, LED (yellow) of the on-chip emulator lights up and LED (green) on the buck converter EVK blinks. Blinking LED (green) is programmed to vary with the state of the power supply.
④Clicking "Suspend" pauses the program. LED (yellow) of the on-chip emulator turns off, and LED (green) of the buck converter EVK also pauses blinking.
⑤Clicking the "Terminate" button terminates the debugging operation (LED (red) on the buck converter EVK turns off because the communication between the on-chip emulator and the microcomputer is interrupted, but LED (blue) turns on if USB cable of the serial communication module is not disconnected).

4. Structure and Description of RMOS Projects

RMOS project contains a number of files (program modules). A ".asm" file is a file whose LEXIDE-Ω is to be built, and each file is handled independently. ".s" file is used as an include file (a collection of source code to improve the readability and mobility of programs) and is placed in.asm file.

In RMOS projects, files are organized and stored in several folders. Table 4-1 lists the folders (RMOS100.ASM is a file).

Table 4-1. List of Program Module Folders

No.	Folder name (File name)	Description
1	RMOS100.ASM	Program executed at MCU reset (before starting multitasking control)
2	00_System	MCU selection, RAM area definition, RAM default definition, OS function setting, OS control (multitask control, etc.) programming
3	10_Setting	Initial setting of power parameter, initial setting of power supply operation mode, and setting of built-in function of MCU (timer, A/D converters, etc.)
4	30_Info_module	Communication command definition (development), communication command execution program
5	50_BG_module	Background modules group
6	51_BG_Control_include	Include files for use with background modules
7	55_BGLp_module	Background module group executed in low power operation mode
8	70_PS0_State_Control_module	State transition control module group 0 (e.g., control of DCDC converter)
9	71_PS0_Control_include	Include file used by the state transition control module group 0
10	80_PS1_State_Control_module	State transition control module group 1 (e.g., assuming control of PFC)
11	81_PS1_Control_include	Include file used by the state transition control module group 1
12	90_Header	OS control program located at the top of every program module

The following explains the description of program modules and programs using the RMOS100-ML62Q2035_Buck project file as an example.

4-1. "RMOS100.ASM" File (default executable file)

This file is executed before multitasking control is started at MCU reset. Immediately after a program is written to the MCU, the program is paused at the "nop" instruction labeled "Start:" in this file.

In this program module, the initial settings of the MCU and the start settings of the multitasking operation are performed. Normally, you do not need to edit this file.

4-2. "0_System" Folder (MCU and RMOS system setting)

Table 4-2 files are stored in this folder. The following sections describe the files that you must be aware of when editing.

Table 4-2. Files in the "00_System" folder

No.	File name	Description
1	00S00_MCUselect.s	Select the model number of the MCU to be used
2	00S60_Macro_def_OS.s	Stores macro instructions for extending assembler instructions
3	※00S80_ResetInit.s	Initialization at MCU reset
4	※00S92_TaskSelector.asm	Controls multitasking and real-time operation of background modules and state transition control modules
5	00S93_Func_Select.s	Function setting of OS during program debugging
6	※00S95_Module_Common.s	Definition for all ".asm" files
7	※00S96_HBG_module.s	Background modules used by OS
8	※00S97_LpEnterExit.asm	Control of switching to/from low power operation mode
9	※00S98_Idling.asm	Standby upon completion of execution in each program module
10	※00S99_VectorTBL_OpByte_def.s	MCU initialization definition

※Normally, editing is not required.

(1) "00S00_MCUselect.s" File (MCU model number selection)

When changing the model number of the MCU to be used, perform editing according to Figure 4-1.

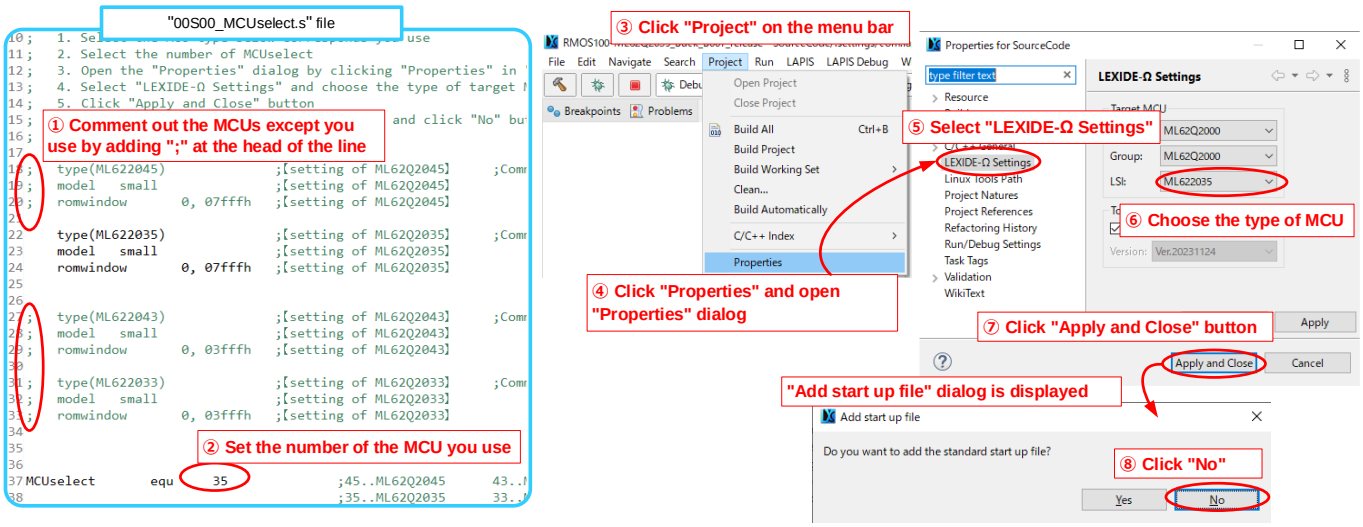


Figure 4-1. Model number change of MCU

Note: If you mistakenly clicked "Yes" in step ⑧, a "ML6220xx.ASM" file is generated in the "SourceCode" folder. If you run a build in this state, an error message is displayed. Therefore, be sure to remove the "ML6220xx.ASM" file before running the build. If the files have been deleted, you do not need to perform Figure 4-1 step again.

(2) "00S60_Macro_def_OS.s" File (definition macro instructions)

The "00S60_Macro_def_OS.s" file defines the macro instructions used for the entire project in RMOS. Macro instructions are used to instruct RMOS and extend the mnemonics of ML62Q20xx groups. In RMOS, the macro instruction is preceded by "$". Table 4-3 lists RMOS macro instructions.

Table 4-3. RMOS macro instructions

No.	Macro instructions	Classification	Description
1	$Task_end	Instructions to RMOS	Tell RMOS that the task is complete
2	$SlowTask_end		Instruct RMOS that the task is complete (module with at least the execution period 10ms)
3	$LpTask_end		Instruct RMOS to complete a task in low power operation mode
4	$PS0_CHGstate label-name		In the state transition control module group 0, instruct RMOS to execute the module to be executed in the next cycle
5	$PS1_CHGstate label-name		In the state transition control module group 1, instruct RMOS to execute the module to be executed in the next cycle
6	$add16 ERn value (symbol)	Mnemonic extended	Add 16bits length value to ERn (n=0,2,4,6,8) register
7	$cmp16 ERn ERm		Compare ERn (n=0,2,4,6,8) and ERm(m=0,2,4,6,8) register
8	$mov16 ERn value (symbol)		Assign 16bits length value to ERn(n=0,2,4,6,8) register
9	$sll16 ERn value (symbol)		Shift ERn(n=0,2,4,6,8) to the left by the number of bits specified by numerical value
10	$srl16 ERn value (symbol)		Shift ERn(n=0,2,4,6,8) to the right by the number of bits specified by numerical value
11	$sub16 ERn ERm		Subtract ERn(n=0,2,4,6,8) register and ERm(m=0,2,4,6,8) register

4-3. "10_Setting" Folders (MCU hardware setting)

The files listed in Table 4-4 are stored in this folder. The following sections describe the files that you must be aware of when editing.

Table 4-4. Files in the "10_Setting" folder

No.	File name	Description
1	10S01_Parameter_Init.s	Set the initial value of power supply operation parameters
2	10S05_PS_Mode_Set.s	Set initial value of items that are dynamically changeable programmed in power supply operation
3	10S20_RAM_def.s	Define a label to reserve a variable are on RAM
4	10S21_RAM_Init.s	Assign the default value to the variables allocated on RAM
5	10S25_Extrn_def.s	Setting for enabling the labels defined in "10S20_RAM_def.s" in other program modules
6	10S50_GPIO_Set.s	Digital I/O port assignment and operation setting
7	10S51_OperationalTimer_Set.s	Setting of Operational Timer (Power Supply Control Timer)
8	※10S52_16bitTimer_Set.s	Setting of 16bits timer (used for multitasking control), setting cannot be changed
9	10S53_CMP_Set.s	Assignment of Comparator and operation setting
10	10S54_EXTRG_Set.s	Assignment of external trigger input and operation setting (not used in buck converter)
11	10S60_ADconv_TEMPsens_PGA_Set.s	Assignment of A/D converters, PGA and operation setting, and thermal sensor setting
12	10S61_DAconverter_Set.s	Operation setting of D/A converters
13	※10S70_UART_Set.s	Assignment of UART and operation setting

※Normally, editing is not required.

(1) Version management

RMOS version is described in the "10S01_Parameter_Init.s" file as symbol names.

"RMOSVer" (OS version) and "OSBuildNo" (OS build number) represent the version number. This version number will be updated when we release it. Therefore, it does not need to be edited (Figure 4-2 is the 2024/4/1 version).

"PSFMNo" is a numerical value representing the power supply topology controlled by RMOS (Buck converter is "001"). In addition, "PSFMVer" and "PSFMBuildNo" indicate the firmware version of the power topology to be controlled. It updates the version of the designed power supply when you change the state transition module, background module, and peripheral sets. These symbol values can be freely rewritten.

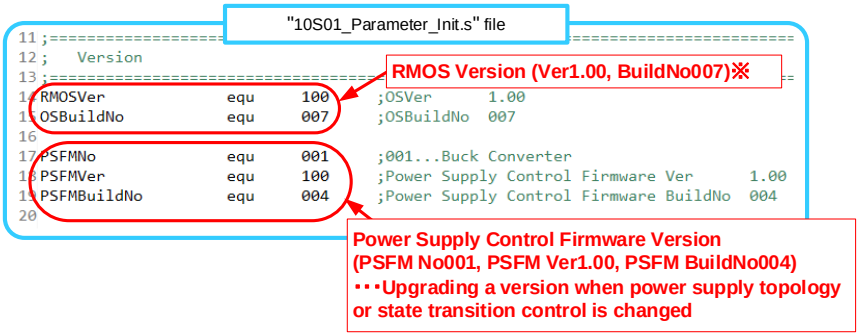


Figure 4-2. Version description

※Normally, editing is not required.

(2) Variable allocation (definition RAM area)

To reserve variables in RAM area, the following four files are edited.

①10S20_RAM_def.s ②10S25_Extrn_def.s
③10S01_Parameter_Init.s ④10S21_RAM_Init.s

"10S20_RAM_def.s" file is used to reserve (label-define) variable-area on RAM. Defined labels are declared as public in this file.

By declaring extrn in "10S25_Extrn_def.s" file, it can be used in other program modules (".asm" module) in RMOS.

"10S01_Parameter_Init.s" file is used to define the default values of the variables allocated on RAM. For the initial value definition, the symbol name is defined and given a numeric value.

In "10S21_RAM_Init.s" file, the symbolic value defined in "10S01_Parameter_Init.s" file is assigned to the variable allocated in the "10S20_RAM_def.s" file.

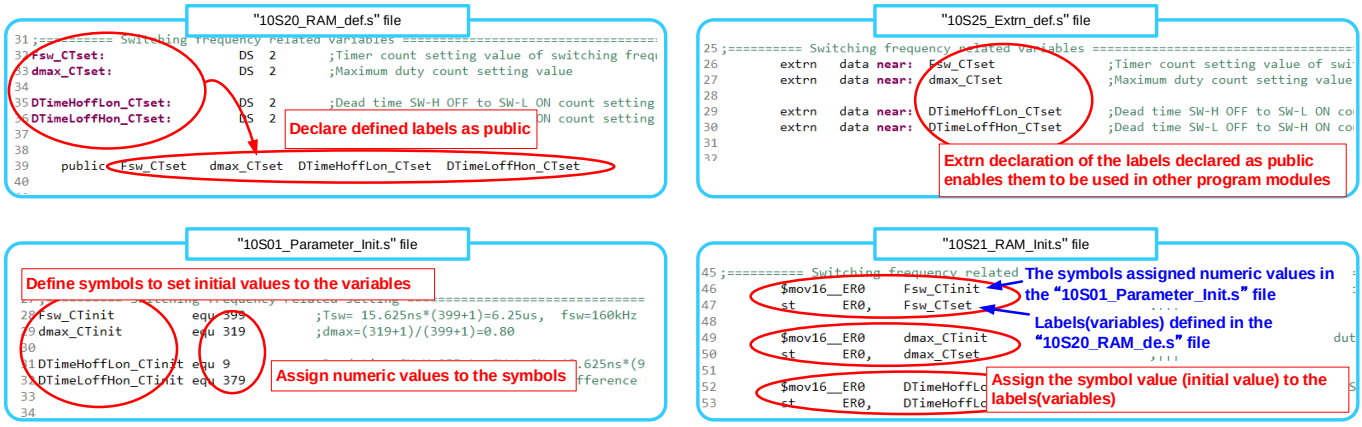


Figure 4-3. Variable definition on RAM and public/extrn declaration and initial value assignment

(3) "10S50_GPIO_Set.s" File (setting of I/O as digital input/output)

Settings for using GPIO port as an input pin or an output pin. It also assigns symbols to GPIO ports. Symbols assigned to GPIO ports can be used by other program modules in RMOS (".asm" module) by declaring public in this file and declaring extrn in the "10S25_Extrn_def.s" file.

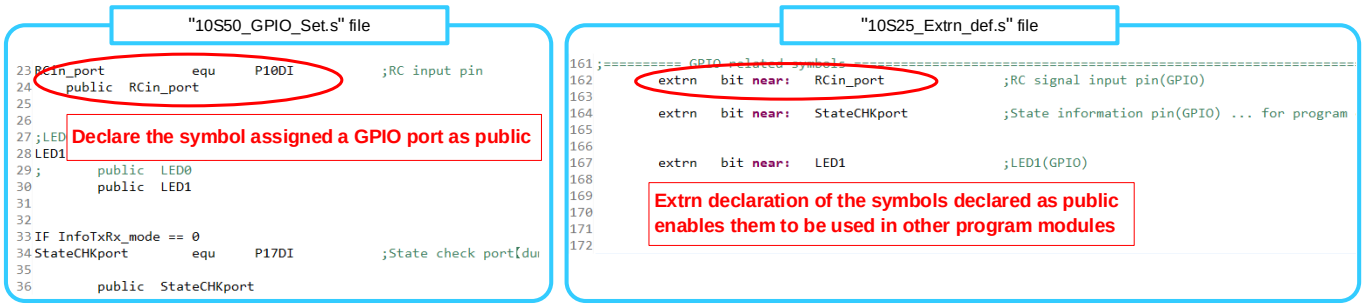


Figure 4-4. GPIO port setting and public/extrn declaration and symbol assignment

(4) "10S60_ADconv_TEMPsens_PGA_Set.s" File (setting of A/D converter, temperature sensor and PGA)

Configure the settings for A/D converters, PGA, and temperature sensors (Normally change of the temperature sensor setting is not needed), also for symbol assignment to each port. In RMOS, A/D converters are converted and automatically acquired in the background. Acquirement is performed automatically maximum of 4ch in normal operation and 2ch in low power operation. Use this file to set AD channels to be automatically acquired.

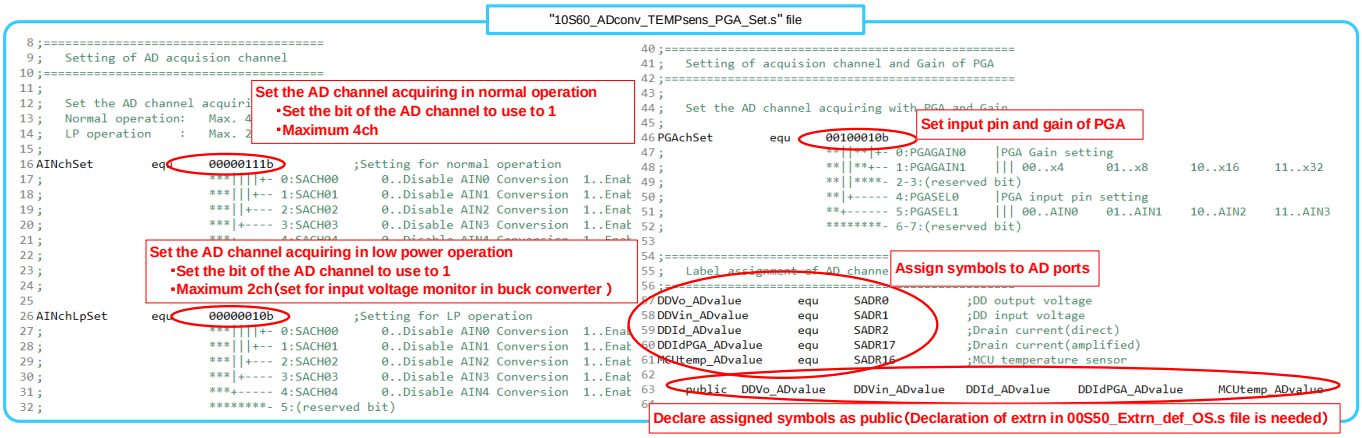


Figure 4-5. A/D Converters, PGA and Temperature Sensor Settings

4-4. "30_Info_module" Folder (communication control module)

The program module that controls the communication is stored in this folder. For more information on how to use communication, refer to the application note [5] of the Description of Communication Functions and GUI development.

Table 4-5. Files in the "30_Info_module" folder

No.	File name	Description
1	30I01_InfoCMD_Exec.asm	Write the execution program of the communication commands
2	30I11_InfoCMD_Table_def.s	Assignment of communication commands

4-5. "50_BG_module" Folder (background modules)

This folder contains background modules. Programs that must operate independently of the state transition control is written in the background module. The background module has a program description area and by writing a program here, that is executed at the default execution cycle. The programs to output the state variables and the state flags are written in this module as well.

Table 4-6. Files in the "50_BG_module" folder

No.	File name	Execution cycle	No.	File name	Execution cycle
1	50B002_BG25u_Task.asm	25µs	14	50Bb03_BG100m_Task3.asm	100ms
2	50B005_BG50u_Task.asm	50µs	15	50Bb04_BG100m_Task4.asm	↑
3	50B010_BG100u_Task.asm	100µs	16	50Bc00_BG500m_Task0.asm	100ms
4	50B050_BG500u_Task.asm	500µs	17	50Bc01_BG500m_Task1.asm	↑
5	50B100_BG1m_Task0.asm	1ms	18	50Bc02_BG500m_Task2.asm	↑
6	50B101_BG1m_Task1.asm	↑	19	50Bc03_BG500m_Task3.asm	↑
7	50Ba00_BG10m_Task0.asm	10ms	20	50Bc04_BG500m_Task4.asm	↑
8	50Ba01_BG10m_Task1.asm	↑	21	50Bd00_BG1000m_Task0.asm	1000ms
9	50Ba20_BG25m_Task0.asm	25ms	22	50Bd01_BG1000m_Task1.asm	↑
10	50Ba21_BG25m_Task1.asm	↑	23	50Bd02_BG1000m_Task2.asm	↑
11	50Bb00_BG100m_Task0.asm	100ms	24	50Bd03_BG1000m_Task3.asm	↑
12	50Bb01_BG100m_Task1.asm	↑	25	50Bd04_BG1000m_Task4.asm	↑
13	50Bb02_BG100m_Task2.asm	↑			

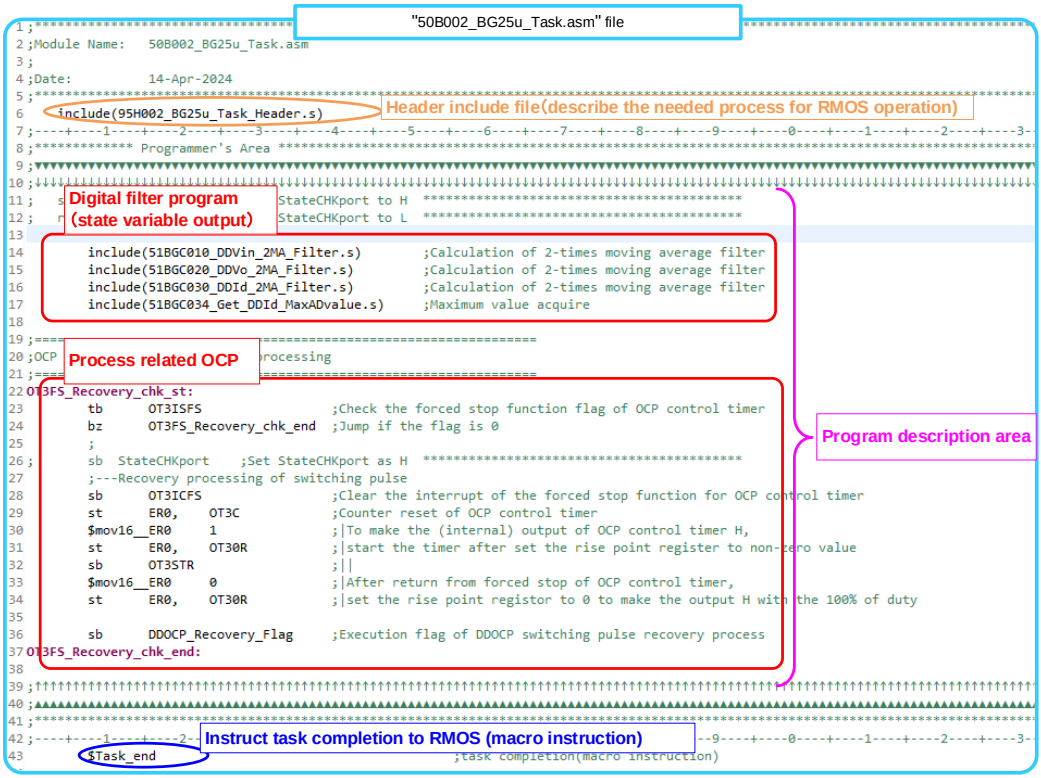


Figure 4-6. Background module description

4-6. "51_BG_Control_include" Folder (include files for background modules)

This folder contains include files (files with the extension ".s") used by background modules. Currently, the include files to output the state variables and state flags.

Table 4-7. Files in the "51_BG_Control_include" folder

No.	File name	Description
1	51BGC010_DDVin_2MA_Filter.s	Outputs two-times moving averages (state variable) of input voltage AD value
2	51BGC020_DDVo_2MA_Filter.s	Outputs two-times moving averages (state variable) of output voltage AD value
3	51BGC030_DDId_2MA_Filter.s	Outputs two-times moving averages (state variable) of drain current AD value
4	51BGC034_Get_DDId_MaxADvalue.s	Outputs the maximum value (state variable) of the drain current AD value
5	51BGC035_DDIdPGA_2MA_Filter.s	Outputs four-times moving averages (state variable) of drain current PGA value
6	51BGC036_DDIdPGA_8MA_Filter.s	Outputs eight-times moving averages (state variable) of drain current PGA value
7	51BGC210_DDVin_RiseFall_Chk.s	Monitors input voltage and outputs start/stop enable (state flag)
8	51BGC900_RC_Chk.s	Monitors RC pin and outputs start/stop enable (state flag)

4-7. "55_BGLp_module" Folder (background modules for low power mode)

This folder contains background modules operating in low power mode. In low power mode, the operation of the state transition control module group is stopped, and the background module after No.7 of Table 4-6 (more than the run cycle 10ms) and the background modules for Table 4-8 low power operation mode operate.

Table 4-8. Files in the "55_BGLp_module" folder

No.	File name	Execution Cycle	No.	File name	Execution Cycle
1	55B005_BGLp50u_Task.asm	50µs	2	55B010_BGLp100u_Task.asm	100µs

4-8. "70_PS0_State_Control_module" Folder (state transition control module group 0)

This is the folder in which the state transition control module group 0 is stored. A program description area is also provided in the state transition control modules. In state transition control, any one of Table 4-9 program modules (only normal state operation, two simultaneous operation) is executed. When the execution module transitions, the macro instruction that instructs RMOS to the state transition is used.

Each operation state is divided into multiple units to enhance the descriptivity and readability of the power control program. Below is the control content of each state transition module in the buck converter.

Table 4-9. Files in the "70_PS0_State_Control_module" folder

No.	File name	Operating state	Example of control in a buck converter
1	71P010_PS0_Standby_0.asm	Standby	No processing (This module is empty because standby operation can be described in 4 states)
2	71P011_PS0_Standby_1.asm		Input voltage start enable check
3	71P012_PS0_Standby_2.asm		Start delay time count
4	71P013_PS0_Standby_3.asm		RC activation enable check/soft start calculation 1
5	71P014_PS0_Standby_4.asm		Soft start calculation 2
6	72P020_PS0_Startup_0.asm	Startup	Soft start operation 0 (Clamp circuit voltage rises to the bottom of triangle wave)
7	72P021_PS0_Startup_1.asm		Soft start operation 1 (Output voltage rises to 50% of the setting value)
8	72P022_PS0_Startup_2.asm		Soft start operation 2 (Output voltage rises to 75% of the setting value)
9	72P023_PS0_Startup_3.asm		Soft start operation 3 (Output voltage rises to 93% of the setting value)
10	72P024_PS0_Startup_4.asm		Soft start operation 4 (Handover of clamp circuit and feedback control circuit)
11	73P030_PS0_Normal_50u.asm	Normal	Various stopping check, OVP check, switching pulse limitation at OCP
12	73P035_PS0_Normal_500u.asm		Switching pulse limitation release at OCP return, LVP check
13	74P040_PS0_Stop_0.asm	Stop	Switching devices stop and peripheral stop
14	74P041_PS0_Stop_1.asm		No processing (This module is empty because the stop operation can be described in 4 states)
15	74P042_PS0_Stop_2.asm		Initialization of various setting values (for next startup)
16	74P043_PS0_Stop_3.asm		Normal stop check
17	74P044_PS0_Stop_4.asm		Latch stop processing

Figure 4-7. State Transition Control Module Description

4-9. "71_PS0_Control_include" Folder (include file for state transition control module group 0)

This is the folder to store the include files (files with extension ".s") used in the state transition control module group 0. The includ files are not used in the buck converter state transition control.

4-10. "80_PS1_State_Control_module", "81_PS0_Control_include" Folder (state transition control module group 1)

This is the folder where the state transition control module group 1 and the include files (files with extension ".s") used in the state transition control module group 1 are stored.

By using state transition control module group 0 and 1, two converters can be controlled independently. In the buck converter EVK, only one converter is controlled, so state transition control module group 1 is not used.

4-11. "90_Header" Folder (header files)

This is a folder that collects header files (include files) to be placed at the beginning of background modules, state transition control modules, etc. The header file performs the processing required for RMOS to work. Normally, you do not need to edit the files in this folder.

5. Programming with RMOS

5-1. Resources of MCU Occupied by RMOS

RMOS uses the following resources: Do not use the following resources when creating a program.

Table 5-1. Resources of MCU used by RMOS

No.	Resource name	Remarks
1	ER10 register	Used to manage the program module executed by the state transition control module 0
2	ER12 register	Used to manage the program module executed by the state transition control module 1
3	ER14 register	Used to manage program modules that run in multitasking operations
4	16bit timer	Used for control cycle management of multitasking operation
5	Interruption	Prohibit use by programmers because it affects multitasking operation
6	Watchdog timer	Resetting for each 50ms

5-2. Function to be Background Processed by RMOS

RMOS processes the following functions in the background.

(1) Acquire A/D converter value

During normal operation, A/D converters up to 4ch are automatically acquired (acquisition cycle: 25μs). The channels to be acquired are set in the "10S60_ADconv_ TEMPsens_PGA_Set.s" file.

In low power operation, A/D converters of the maximum of 2ch are automatically acquired (acquisition cycle: 50μs). The channels to be acquired are set in the "10S60_ ADconv_TEMPsens_PGA_Set.s" file.

By assigning a state variable to SFR where A/D converters are recorded, such as digital filters for calculating a two-times moving average can be used.

(2) Acquiring PGA

During normal operation, PGA is automatically acquired (acquisition cycle: 50μs). Pin assignment is done in "10S60_ADconv_TEMPsens_PGA_Set.s" file.

In low power operation, acquisition of PGA is stopped.

(3) Temperature sensor value acquisition

In normal operation and in low power operation, temperature sensor readings are automatically acquired (acquisition cycle: 50ms).

(4) Communication

A communication protocol that enables communication with up to 32 power supplies is implemented for one PC by using UART as hardware and performing software processes. To use the communication function, use the state variables prepared by RMOS. For more information on communication, refer to the application note [5] of the explanation of communication functions and GUI development.

(5) Operation log and setting parameter recording ***Currently, not implemented**

A function that records operation logs of the power supply and power supply setting parameters (start voltage, stop voltage, start delay time, overcurrent setting value, etc) in the data flash memory in the microcontroller is scheduled to be implemented.

5-3 State Variables and State Flags

In creating a power control program with RMOS, you use the "state variables" and "state flags" that RMOS outputs to program. In RMOS, the background module contains a program that handles "state variables" and "state flags". The "state variable" and "state flag" processing program can also be created by the programmer.

Below is an example of the power supply start/stop judgment program using the state flag in the control program of the buck converter.

(1) Start/Stop state flag

The start/stop state flag is processed by the "51BGC210_DDVin_RiseFall_Chk.s" module. Since this module is arranged in a background module with an execution cycle of 50μs, the input voltage is monitored every 50μs. When this module is placed in a background module with an execution cycle of 100μs, for example, the monitoring cycle of the input voltage can be changed to 100μs.

The operation of this module is shown below.

①If the two-times moving average value (DDVin_2MAvalue; state variable that is output based on the input voltage AD value) reaches the starting threshold voltage (the value defined in DDVin_RISEsetinit), the specified number of times (the value set in DDVin_RISEchk_CTinit) is detected, and DDVin_RISE_Flag=1 is output.

②When the two-times moving average value (DDVin_2MAvalue) of input voltage reaches the stop threshold voltage (the value defined in DDVin_FALLsetinit) or less for the specified number of times (the value set in DDVin_FALLchk_CTinit), DDVin_FALL_Flag=1 is outputted.

"50B005_BG50u_Task.Asm" file

```
14
15      include(51BGC900_RC_Chk.s)              ;Check RC pin
16
17      include(51BGC210_DDVin_RiseFall_Chk.s)  ;DDVin startup/stop check
18
```

Figure 5-1. Program module execution period to output start/stop state flag

"10S01_Parameter_Init.s" file

```
41 ;========== Input voltage startup/stop processing ==============================================
42 DDVin_RISEsetinit   equ 12880       ;Initial value of startup voltage   12880 = 9Vdc
43 DDVin_FALLsetinit   equ 11456       ;Initial value of stop voltage      11456 = 8Vdc
44
45
46 DDVin_RISEchk_CTinit    equ 3       ;Initial value of noise rejection counter for startup volt
47 DDVin_FALLchk_CTinit    equ 3       ;Initial value of noise rejection counter for stop voltage
```

Figure 5-2. Setting of start/stop voltage, etc.

(2) Power Supply Start Judgement Program

Determine the state of the "DDVin_RISE_Flag" and make a power-up decision. When DDVin_RISE_Flag = 1, RMOS is programmed to indicate state transitions.

"71P011_PS0_Standby_1.asm" file

```
19 ;================================================
20 ;Checking and processing for Vin startup enable
21 ;================================================
22 IF DDVin_RFchk_Enable == 1
23 PS0_DDVin_RISE_chk_st:
24         ;-----DDVin startup enable check
25         tb      DDVin_RISE_Flag              ;Check the startup enable input voltage detection flag for DD
26         bz      PS0_DDVin_RISE_chk_end       ;Jump if the flag is 0
27         ;-----Detect the startup input voltage
28         l       ER0,    DDVin_RISEwait_CTset ;|Reset the startup delay counter
29         st      ER0,    DDVin_RISEwait_CT    ;|||
30
31       $PS0_CHGstate   P012_PS0_Standby_2_st  ;Change next execution module(state transition)(macro instruct
32         $Task_end
33 PS0_DDVin_RISE_chk_end:
```

Check DDVin_RISE_Flag

In case DDVin_RISE_Flag=1, instruct state transition to RMOS

Figure 5-3. Power Supply Start Judgement Program

(3) Power Supply Stop Judgement

Determine the state of DDVin_FALL_Flag and make a shutdown decision. When DDVin_FALL_Flag = 1, RMOS is programmed to indicate state transitions.

"73P030_PS0_Normal_50u.asm" file

```
60 ;================================================
61 ;Checking and processing for Vin stop enable
62 ;================================================
63 IF DDVin_RFchk_Enable == 1
64 PS0_DDVin_FALL_chk_st:
65         ;-----DDVin stop enable check
66         tb      DDVin_FALL_Flag                 ;Input voltage detection flag for DD block
67         bz      PS0_DDVin_FALL_chk_end          ;Jump if the flag is 0
68         ;-----停止電圧を検出
69         mov     R0,     #15                     ;|Set the DD block operation code to "Vin standby"
70         st      R0,     DD_OpCode               ;|||
71         ;
72         $PS0_CHGstate   P040_PS0_Stop_0_st      ;Change next execution module(state transition)(macro instruction)
73         $Task_end                               ;task completion(macro instruction)
74 PS0_DDVin_FALL_chk_end:
```

Check DDVin_FALL_Flag

In case DDVin_FALL_Flag=1, instruct state transition to RMOS

Figure 5-4. Power Supply Stop Judgement Program

5-4. Low Power Operation Mode

To switch from the normal operation mode to the low power operation mode, use the state variable "LpMode_Enter_Flag". In the control program of the buck converter, when the power supply is stopped using RC pin, it is shifted to the low power operation mode.

In low power operation mode, state transition control module group 0 and state transition control module group 1 stop. In addition, the background module of No.1 to No6 of Table 4-6 is stopped, and the background module operates for Table 4-8 low power operation mode.

"71P013_PS0_Standby_3.asm" file

```
27 ;=====================================
28 ;RC startup check and prosessing
29 ;=====================================
30 PS0_RC_chk_st:
31         ;-----RC startup enable check
32         tb      RC_ON_Flag                      ;Check RC ON flag
33         bnz     PS0_RC_ON_detect                ;Jump if the flag is 1
34         ;-----Detect RC stop
35         sb      LpMode_Enter_Flag
36         b       PS0_RC_chk_end
37 PS0_RC_ON_detect:
```

By setting LpMode_Enter_Flag to 1, operating mode moves from normal operation to low power operation

Figure 5-5. Entering low power operation mode

To switch from low power operation mode to normal operation mode, use "LpMode_Exit_Flag". The control program of the buck converter is shifting to the normal operation mode when RC pin is used to cancel the power supply shutdown.

When the mode changes to the normal operation, the state transition control module operates the module that was being executed immediately before the mode changes to the low power operation mode.

"55B005_BGLp50u_Task.asm" file

```
37 ;=====================================
38 ;RC startup check and prosessing
39 ;=====================================
40 RC_chk_st:
41         ;-----Check RC reset process enable
42         tb      LpMode_Use_RcReset_Flag         ;Check RC reset process use flag
43         bnz     RC_chk_end                      ;Jump to disable the RC startup if the flag is 1
44         ;-----RC startup enable check
45         tb      RC_ON_Flag                      ;Check RC ON flag
46         bz      RC_chk_end
47         ;-----Detect RC startup
48         sb      LpMode_Exit_Flag
49 RC_chk_end:
```

By setting LpMode_Exit_Flag to 1, operating mode moves from low power operation to normal operation

Figure 5-6. Entering the normal operation mode

5-5. Debugging Activities (breaking CPU, stepping, checking variables)

In RMOS, the timer, comparators, and D/A converters are set to continue operating even if CPU is broken. Normally, switching power supplies with analog-digital hybrid control are designed to perform switching operation without using a CPU, so the power supply circuit continues to operate even if CPU is stopped. Therefore, the program operation can be checked using step execution while the switching operation of the power supply circuit is continued.

(1) Set Breakpoint

Figure 5-7 shows how to set a breakpoint at "73P030_PS0_Normal_50u.asm". Breakpoints can be set in the files their extension is ".asm", but not in the ".s" (specifications for LEXIDE-Ω).

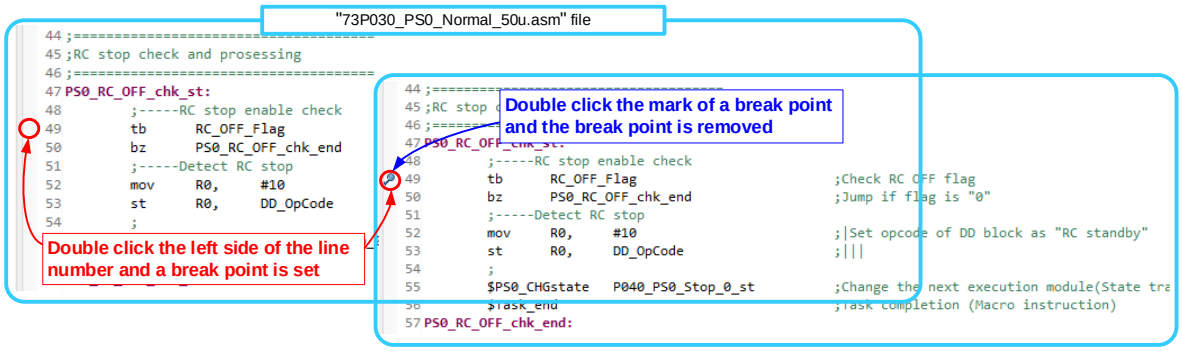


Figure 5-7. Set Breakpoint

If you run the program with a breakpoint set, the program breaks when the program reaches the breakpoint. For example, setting the breakpoint to "73P030_PS0_Normal_50u.asm" will cause the program to break when the buck converter reaches steady state operation (5V goes out). The program breaks, but the switching operation continues, so the output of the power supply maintains 5V output.

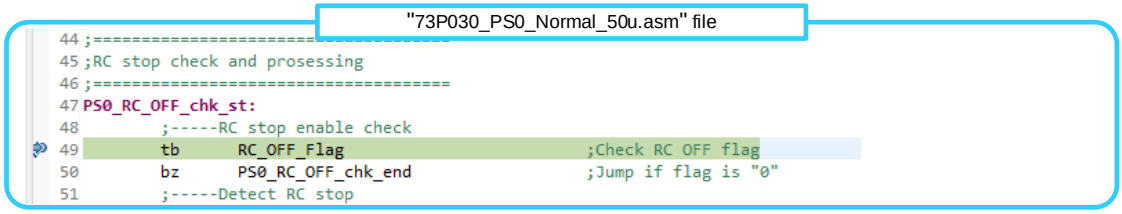


Figure 5-8. Screen at break

When a program breaks, "Expression window" in the "Microcontroller's Internal Information Display Area" at the lower right of the screen displays values of variables, registers, and SFR. "Expression window" can also show the variables and flags set in "00S20_RAM_def_OS.s". Figure 5-9 shows the state variables and state flags that RMOS outputs. To change the variables or flags that are displayed in "Expression window", change the name of "Expression" field. At this time, "#1" is appended to the variable name as "Display byte count specification" for the 1-byte variable (variable allocated by DS 1 instruction) in "00S20_RAM_def_OS.s". For a 2-byte variable (a variable allocated by DS 2 instruction), "#2" is appended to the variable name.

To display the flag, it is not necessary to specify the number of displayed bytes.

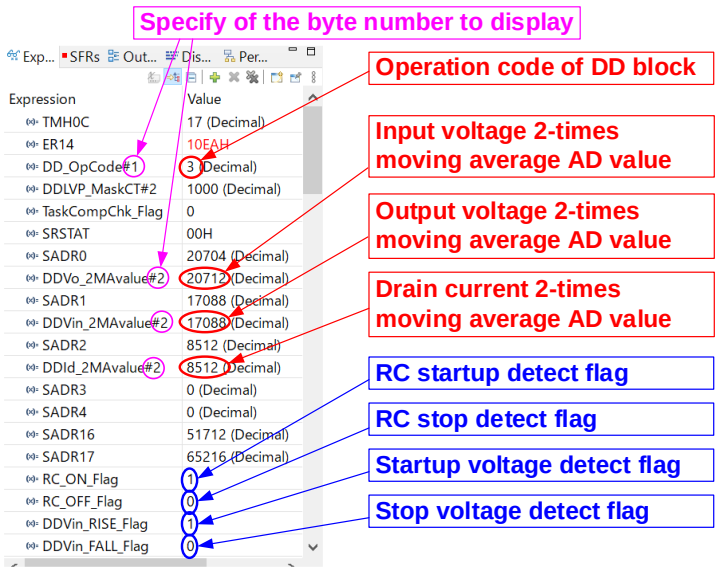


Figure 5-9. Displaying MCU internal information display area

A program that has been broken can be stepped up one line at a time by performing step execution. When the program is restarted, the program continues until the next execution cycle. If a breakpoint is set in "73P030_PS0_Normal_50u.asm" and the program is restarted, the MCU will break after 50μs (execution period).

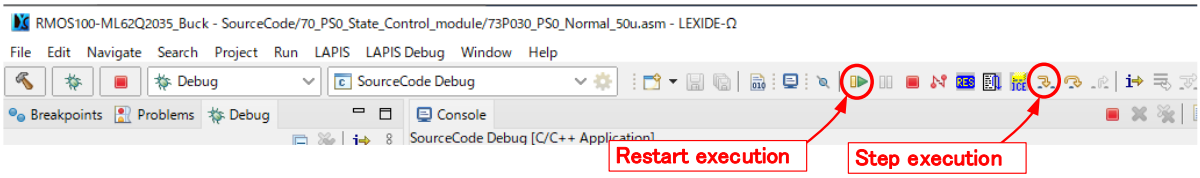


Figure 5-10. Step execution/restart button of the program

5-6. Checking the Allowable Time for Task Assignment (allowable count value)

In RMOS multitasking process, the program must be programmed so that the program operation is completed within the program module allocation time. If the program operation is not completed within the allotted time, the program execution is stopped when the allotted time is reached, and control is transferred to the next program module.)

Task allocation time is managed by 16bit timer, so it can be checked by counting 16bit timer. To check 16bit timer count value, follow Figure 5-11 steps. Check that the count value of 16bit timer at the end of the program in the program modules is within the allowable count value in the table below.

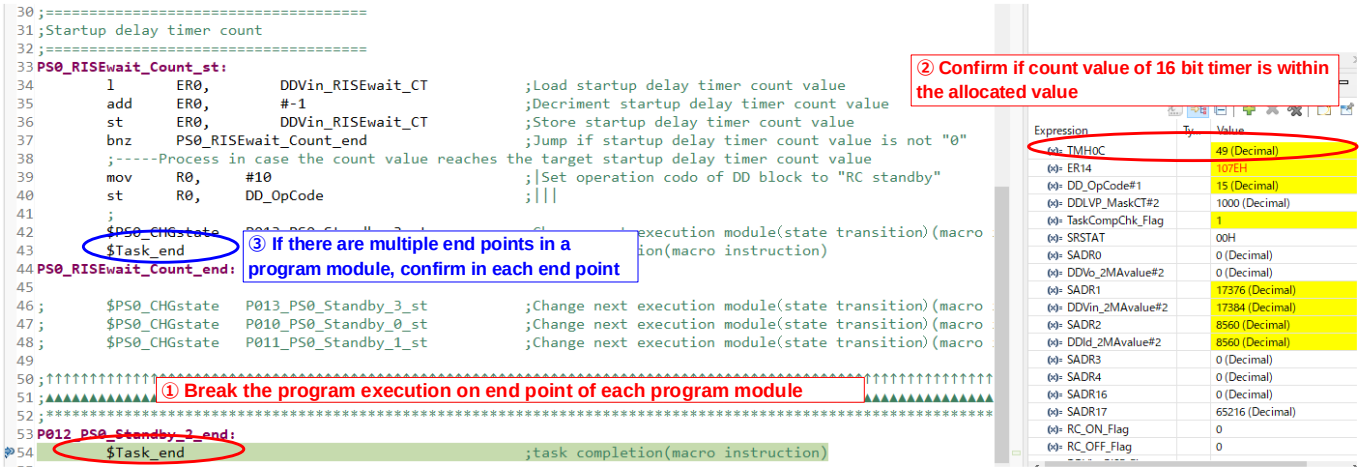


Figure 5-11. Checking the task allocation time using 16bit timer count value

Table 5-2. Allowable count value for task assignment of background modules (normal operation)

No.	File name	Allowable count value	No.	File name	Allowable count value
1	50B002_BG25u_Task.asm	114	14	50Bb03_BG100m_Task3.asm	140
2	50B005_BG50u_Task.asm	124	15	50Bb04_BG100m_Task4.asm	↑
3	50B010_BG100u_Task.asm	144	16	50Bc00_BG500m_Task0.asm	↑
4	50B050_BG500u_Task.asm	↑	17	50Bc01_BG500m_Task1.asm	↑
5	50B100_BG1m_Task0.asm	↑	18	50Bc02_BG500m_Task2.asm	↑
6	50B101_BG1m_Task1.asm	↑	19	50Bc03_BG500m_Task3.asm	↑
7	50Ba00_BG10m_Task0.asm	140	20	50Bc04_BG500m_Task4.asm	↑
8	50Ba01_BG10m_Task1.asm	↑	21	50Bd00_BG1000m_Task0.asm	↑
9	50Ba20_BG25m_Task0.asm	↑	22	50Bd01_BG1000m_Task1.asm	↑
10	50Ba21_BG25m_Task1.asm	↑	23	50Bd02_BG1000m_Task2.asm	↑
11	50Bb00_BG100m_Task0.asm	↑	24	50Bd03_BG1000m_Task3.asm	↑
12	50Bb01_BG100m_Task1.asm	↑	25	50Bd04_BG1000m_Task4.asm	↑
13	50Bb02_BG100m_Task2.asm	↑			

Table 5-3. Allowable count value for task assignment of state transition control module group 0 (Normal Operation)

No.	File name	Allowable count value	No.	File name	Allowable count value
1	71P010_PS0_Standby_0.asm	146	11	73P030_PS0_Normal_50u.asm	146
2	71P011_PS0_Standby_1.asm	↑	12	73P035_PS0_Normal_500u.asm	144
3	71P012_PS0_Standby_2.asm	↑	13	74P040_PS0_Stop_0.asm	146
4	71P013_PS0_Standby_3.asm	↑	14	74P041_PS0_Stop_1.asm	↑
5	71P014_PS0_Standby_4.asm	↑	15	74P042_PS0_Stop_2.asm	↑
6	72P020_PS0_Startup_0.asm	↑	16	74P043_PS0_Stop_3.asm	↑
7	72P021_PS0_Startup_1.asm	↑	17	74P044_PS0_Stop_4.asm	↑
8	72P022_PS0_Startup_2.asm	↑			
9	72P023_PS0_Startup_3.asm	↑			
10	72P024_PS0_Startup_4.asm	↑			

Table 5-4. Allowable count value for task assignment of state transition control module group 1 (Normal Operation)

No.	File name	Allowable count value	No.	File name	Allowable count value
1	81P110_PS1_Standby_0.asm	122	11	83P130_PS1_Normal_50u.asm	122
2	81P111_PS1_Standby_1.asm	↑	12	83P135_PS1_Normal_500u.asm	144
3	81P112_PS1_Standby_2.asm	↑	13	84P140_PS1_Stop_0.asm	122
4	81P113_PS1_Standby_3.asm	↑	14	84P141_PS1_Stop_1.asm	↑
5	81P114_PS1_Standby_4.asm	↑	15	84P142_PS1_Stop_2.asm	↑
6	82P120_PS1_Startup_0.asm	↑	16	84P143_PS1_Stop_3.asm	↑
7	82P121_PS1_Startup_1.asm	↑	17	84P144_PS1_Stop_4.asm	↑
8	82P122_PS1_Startup_2.asm	↑			
9	82P123_PS1_Startup_3.asm	↑			
10	82P124_PS1_Startup_4.asm	↑			

Table 5-5. Allowable count value for task assignment of background modules (low power operation)

No.	File name	Allowable count value	No.	File name	Allowable count value
1	55B005_BGLp50u_Task.asm	315	2	55B010_BGLp100u_Task.asm	315

5-7. Detection of Task Incomplete by RMOS

RMOS has a state flag "TaskCompChk_Flag" for checking that the program operation was not completed within the program module allocation time (task not completed). If TaskCompChk_Flag=1 is set when control is transferred to the "Task Selector" in multitasking operation, this indicates that the task has not been completed. A breakpoint on nop instruction can be used to detect that a task has not been completed.

- Normal operation mode : Place a breakpoint at line 96 of "00S92_TaskSelector.asm"
- Low power operation mode: Place a breakpoint at line 67 of "00S92_TaskSelector.asm"

Figure 5-12 indicates that a breakpoint is placed on line 96 of "00S92_TaskSelector.asm" for checking that tasks have not been completed in normal operation mode. When the task not completed is detected, the program is stopped at line 96 of "00S92_TaskSelector.asm".

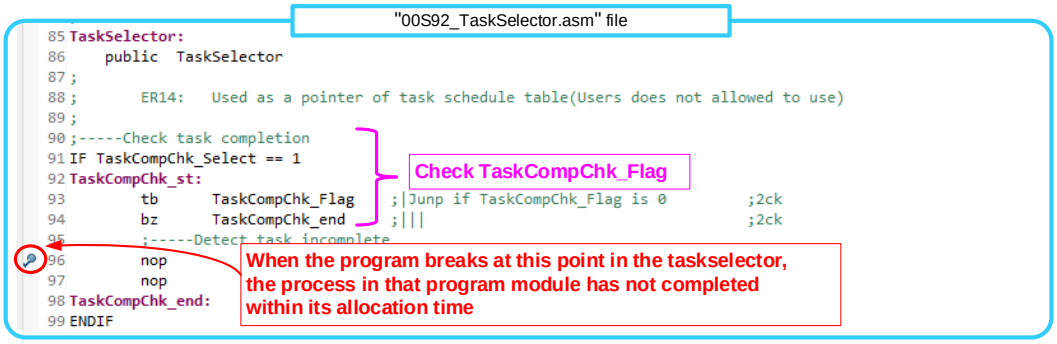


Figure 5-12. Detection of Task Not Completed

6. List of State Variables and State Flags

The table below lists the state variables and state flags that RMOS is currently operating in the background. Note that the version of RMOS in which the parameters listed below are set is "RMOSVer=1.00, OSBuildNo=007, PSFMNo=001, PSFMVer=1.00, PSFMBuildNo=004" (same as the version described in 4-3 Section (1)).

Table 6-1. List of RMOS state variables and state flags (defined in 10S20_RAM_def.s file)

(1)Start/Stop

No.	Variable name・Flag name	Byte	Function	Symbole name for initial value setting	Default value	Comment
1	DDVin_RISEset	2	Startup input voltage setting value	DDVin_RISEsetinit	12880	Monitor cycle: 50μs
2	DDVin_RISE_Flag	Flag	Startup enable input voltage detection for DD block	–	–	
3	DDVin_RISEchk_CT	1	Noise rejection counter	DDVin_RISEchk_CTinit	3	
4	DDVin_FALLset	2	Stop input voltage setting value	DDVin_FALLsetinit	11459	
5	DDVin_FALL_Flag	Flag	Stop enable input voltage detection for DD block	–	–	
6	DDVin_FALLchk_CT	1	Noise rejection counter	DDVin_FALLchk_CTinit	3	

(2)Remote ON/OFF

No.	Variable name・Flag name	Byte	Function	Symbole name for initial value setting	Default value	Comment
1	RC_ON_Flag	Flag	Startup enable detection by RC pin	–	–	Monitor cycle: 50μs
2	RC_ONchk_CT	1	Noise rejection counter	RC_ONchk_CTinit	3	
3	RC_OFF_Flag	Flag	Stop enable detection by RC pin	–	–	
4	RC_OFFchk_CT	1	Noise rejection counter	RC_OFFchk_CTinit	25	
5	RClogic_Inv_Flag	Flag	RC pin input logic invert flag	–	–	

(3)Digital Filter

No	Variable name	Byte	Function	Reference variable name	Byte	Comment
1	DDVin_2MAvalue	2	Two-times moving average of input voltage AD value	DDVin_ADvalue	2	Calculation cycle: 25μs
2	DDVo_2MAvalue	2	Two-times moving average of output voltage AD value	DDVo_ADvalue	2	Calculation cycle: 25μs
3	DDIdPGA_2MAvalue	2	Two-times moving average of drain current PGA value	DDIdPGA_ADvalue	2	Calculation cycle: 50μs
4	DDIdPGA_8MAvalue	2	Eight-times moving average of drain current PGA value	DDIdPGA_ADvalue	2	Calculation cycle: 50μs
5	DDId_2MAvalue	2	Two-times moving average of drain current AD value	DDId_ADvalue	2	Calculation cycle: 25μs
6	DDId_MaxADvalue	2	Maximum of drain current AD value	DDId_ADvalue	2	Acquire cycle: 25μs

(4)Communication

No	Variable name	Byte	Function	Symbole name for initial value setting	Default value	Comment
1	PS_ADR	1	Communication address setting	PS_ADR_init	31	
2	RXD_CmdGr	1	Cmd group received by communication	–	–	
3	RXD_CmdNo	1	Cmd no. received by communication	–	–	
4	RXD_Data16	2	16bit data received by communication	–	–	

(5)Control of low power operation mode

No	Variable name	Byte	Function	Comment
1	LpMode_Enter_Flag	Flag	Enter the low power operation mode	To latch stop in abnormal termination, enable RC reset by set LpMode_Use_RcReset_Flag=1 before move to low power operation mode
2	LpMode_Exit_Flag	Flag	Enter the normal operation mode	
3	LpMode_Use_RcReset_Flag	Flag	Enable RC reset process	

(6)System

No	Variable name	Byte	Function	Comment
1	DD_OpCode	1	Operation code of DCDC block	
2	DD_OpCode_FailRec	1	Operation code of DCDC block abnormal termination	
3	TaskCompChk_Flag	Flag	Check flag of task completion	

(7)LED blinking

No	Symbol name	Byte	Function	Comment
1	LED1FP_VinStby	–	Blink pattern of standby under startup voltage	
2	LED1FP_RcStby	–	Blink pattern of standby by RC	
3	LED1FP_NomOP	–	Blink pattern of normal operation	
4	LED1FP_FAIL	–	Blink pattern of abnormal termination	

A list of the state variables and state flags used for controlling the buck converter is shown below for reference.

Table 6-2. List of state variables and state flags for buck converter control

(1)Switching device・Synchronous rectifier device setting

No	Variable name	Byte	Function	Symbole name for initial value setting	Default value	Comment
1	Fsw_CTset	2	Switching frequency (OTM) setting	Fsw_CTinit	399	
2	dmax_CTset	2	Maximum duty (OTM) setting of switching device	dmax_CTinit	319	
3	DTimeHoffLon_CTset	2	Dead time (OTM) setting between switching device off and synchronous rectifier device on	DTimeHoffLon_CTinit	9	
4	DTimeLoffHon_CTset	2	Dead time (OTM) setting between synchronous device off and switching device on	DTimeLoffHon_CTinit	379	

(2)Startup delay

No.	Variable name	Byte	Function	Symbole name for initial value setting	Default value	Comment
1	DDVin_RISEwait_CTset	2	Startup delay setting	DDVin_RISEwait_CTinit	20000	Count cycle: 50μs
2	DDVin_RISEwait_CT	2	Startup delay counter	–	–	

(3)Softstart

No	Variable name	Byte	Function	Symbole name for initial value setting	Default value	Comment
1	SSRampRate1set	1	Softstart ramplate 1 setting	SSRampRate1init	50	
2	SSRampRate2set	1	Softstart ramplate 2 setting	SSRampRate2init	25	
3	SSRampRate3set	1	Softstart ramplate 3 setting	SSRampRate3init	12	
4	SSTimeupCTset	2	Startup time over detect time	SSTimeupCTinit	1000	
5	SS0loopCTset	1	Loop count before softstart	SS0loopCTinit	3	
6	SS4loopCTset	1	Loop count to normal operation	SS4loopCTinit	150	

(4)Output voltage setting

No	Variable name・Flag name	Byte	Function	Symbole name for initial value setting	Default value	Comment
1	DDVo_DACset	1	DAC value for DCDC block output voltage	DDVo_DACinit	101	
2	DDVo_DAC_MaxLmt	–	DAC max limit value for DCDC block output voltage	(this variable is a sybol)	181	

(5)Protection

No	Variable name	Byte	Function	Symbole name for initial value setting	Default value	Comment
1	DDOCP_Ioset	2	Over current protection setting value for DCDC block	DDOCP_Ioinit	6000	
2	DDLVP_VoADset	2	Low voltage protection setting value for DCDC block	DDLVP_VoADinit	12528	
3	DDLVP_MaskCTset	2	Mask time (Noise rejection) setting	DDLVP_MaskCTinit	1000	
4	DDOVP_VoADset	2	Output over voltage protection setting valuefor DCDC block	DDOVP_VoADinit	25056	
5	DDOVP_MaskCTset	1	Mask time (Noise rejection) setting	DDOVP_MaskCTinit	5	
6	DDIVP_VinADset	2	Input over voltage protection setting valuefor DCDC block	DDIVP_VinADinit	54432	
7	DDIVP_MaskCTset	1	Mask time (Noise rejection) setting	DDIVP_MaskCTinit	5	

7. Reference Documentation

[1] ReleaseNote_LEXIDE_V1_1_1_e, LAPIS Development Tools LEXIDE-Ω V1.1.1 Release Notes

[2] FEXTLEXIDE_OMEGA_UM-02, LEXIDE-Ω User’s Manual

[3] 66UG090E, Rev.001, Synchronous Buck DCDC Converter Evaluation Board LogiCoA001-EVK-001

[4] FEUL-U16-100-INST-03, nX-U16/100 Core Instruction Manual

[5] 66AN149E, Rev.001, Serial communication of RMOS and GUI developing manual

Revision History

Date	Revision Number	Description
12. June. 2024	001	Initial release.