AN13116

Entering STBY mode on a single-core RT1170

Rev. 0 — 02 February 2021

Application Note

Contents

1 Introduction

The i.MX RT1170 crossover processor is setting speed records at 1 GHz. This ground-breaking family combines superior computing power and multiple media capabilities with more usability and real-time functionality. The dual-core i.MX RT1170 runs on the Arm® Cortex®-M7 core at 1 GHz and Arm Cortex-M4 at 400 MHz, while providing best-in-class security. The i.MX RT1170 MCU offers support over a wide temperature range and is qualified for consumer, industrial, and automotive markets.

The RT1170 series is divided into two parts: single-core and dual-core. For details on the dual-core part and the most common settings, see AN13104. This application note outline steps to enter the STBY mode on a single core RT1170 and simulate a single-core state on RT1170 EVK.

NOTE

The steps to enter the STBY mode on a single core RT1170 and simulate a single-core state on RT1170 EVK are two different operations.

The hardware is MIMXRT1170 EVK RevC1 (referred as EVK) and the software is based on SDK 2.9.0 with IAR IDE. The SDK demo has different projects for single-core and dual-core. The SDK demo also supports flash target and ram target.

2 Overview

#unique_3 shows that the basic principle of entering STBY is that both CPUs enter low-power mode and issue STBY requirements. The first principle is the CPU entering a low-power mode. The low-power mode here means CPU’s status: . It can be WAIT, STOP, or SUSPEND mode. Each CPU can be in any state except run mode.

Such as:

- CM7 WAIT, CM4 STOP
- CM7 STOP, CM4 STOP
- CM7 SUSPNED, CM4 WAIT
- CM7 SUSPEND, CM4 SUSPEND

The second principle is that both CPUs have sent the STBY request. If any CPU did not send it, the system cannot enter the STBY mode.

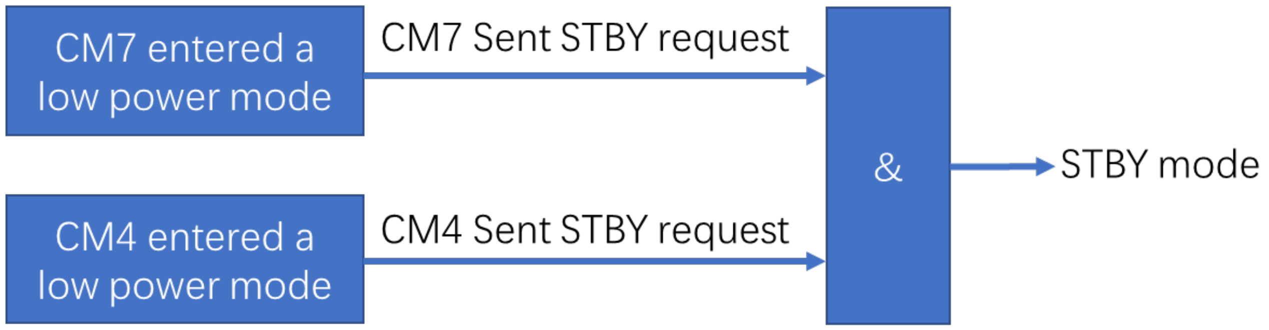


Figure 1. STBY principles for dual-core silicon

For single core silicon, CM4 cannot send this request signal. Therefore, some commands are a must to force the system enter the STBY mode.

For a single core silicon, the basic principle is CM7 enter a low-power mode and sent the stby_request signal. The low-power mode here can be WAIT, STOP, or SUSPEND mode.

The basic flow is:

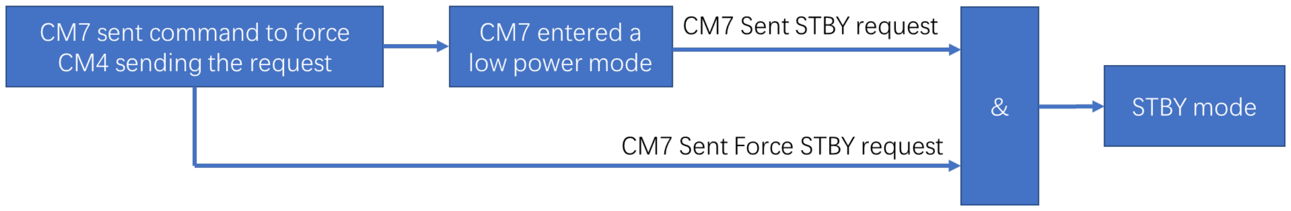

CM7 sent command to force CM4 sending the request
CM7 entered a low power mode
CM7 Sent STBY request
CM7 Sent Force STBY request
&
STBY mode

Figure 2. STBY principle for single state silicon

3 Steps to enter the STBY mode on single-core RT1170

To enter the STBY mode, both the cores send the stby_request. However, in a single-core silicon, such as RT1172 and RT1176, CM4 cannot send it. Therefore, the Force_STBY function is used. The Force_STBY function forces CM4 requesting the STBY mode. Once the M7 enters the low-power mode and sends an stby_request, the whole chip enters the STBY mode.

GPC_STBY_CTRL->STBY_MISC |= GPC_STBY_CTRL_STBY_MISC_FORCE_CPU1_STBY_MASK;

4 Steps to simulate single-core state on RT1170 EVK

EVK is based on RT1176 silicon, which is a dual-core silicon. Customer can simulate it as a single-core state after some settings.

NOTE

Simulation as a single-core state after some settings is only for early evaluation.

The basic method is CM4 entering suspend mode and sending the STBY request.

1. After system boot, CM7 configures some registers.
2. Write PGMC CM4 CPC register as CPU mode with power-off at SUSPEND.
3. Write GPC1 IRQ mask registers as 0xFFFFFFFF so that no interrupt can wake it up.
4. Write GPC1 NON IRQ mask register as 0x3 to avoid a pending interrupt from debugger stop entering low-power mode.
5. Write GPC1 control register to request SUSPEND mode and SBTY mode.
6. CM7 releases CM4 by SRC_SCR register. The silicon is now in the dual-core state.

7. CM4 runs one instruction: assert **WFI** to trigger GPC1 low-power sequence to let itself from entering SUSPEND mode and send the STBY request.
8. When CM4 is in SUSPEND mode, CM7 writes some registers to lock CM4 by software.
9. Write GPC1 AUTHEN register to lock CM1 register access.
10. Write CM4 CCGR slice in CCM to gate off CM4 clock.
11. Write CM4 CCGR AUTHEN register ‘allow_list’ and ‘lock’ fields meaning no CPU can access CM4 CCGR register. again. Thus CM7 software cannot turn on CM4 clock again.
12. Write PGMC CM4 CPC AUTHEN register to lock any access so that software cannot turn on CM4 power again.

The attachment of this application note is provided with detailed code for reference. The basic method is to let CM4 enter suspend mode and send STBY request. Based on this, mask all the wake-up sources in GPC CM1 and then lock the access rights of these registers: any CPU cannot access and modify. CM4 then enters the suspend mode and there is no wake-up source to wake it up and any CPU does not have the access right to change these settings.

The following API is able to complete above configuration.

void Powerdown_CM4(void);

For details, see the attachment with this application note.

5 Differences between a single-core silicon and simulated single-core on RT1176

There are some differences between a single-core silicon and a simulated single-core on RT1176. The first one is RDC. The single-core silicon cannot use RDC to assign a peripheral to a domain. If CM7 wants to access the RDC, a fault occurs. However, a simulated single-core on RT1176 can use RDC. For single-core silicon, all the peripherals are assigned to M7 domain. As a result, only 1 stop request from M7 domain is required during the handshake and only need to check the stop_ack from M7 domain. For details on the handshake, see AN13104, Chapter 4.5.

Table 1. Differences between a single-core silicon and a simulated single-core on RT1176

	Single Core	Simulated single core on RT1176
STBY Method	Force_STBY	CM4 enters Suspend STBY
RDC	Not support	Support
Handshake	Only CM7 sends stop_req and checks the stop_ack	See AN13104, Chapter 4.5

6 Steps to create a single-core project on SDK

The MIMXRT1170 EVK supports single-core program executions. This section lists the steps to set up a single-core project by using different IDEs (IAR, Keil, and Arm GCC).

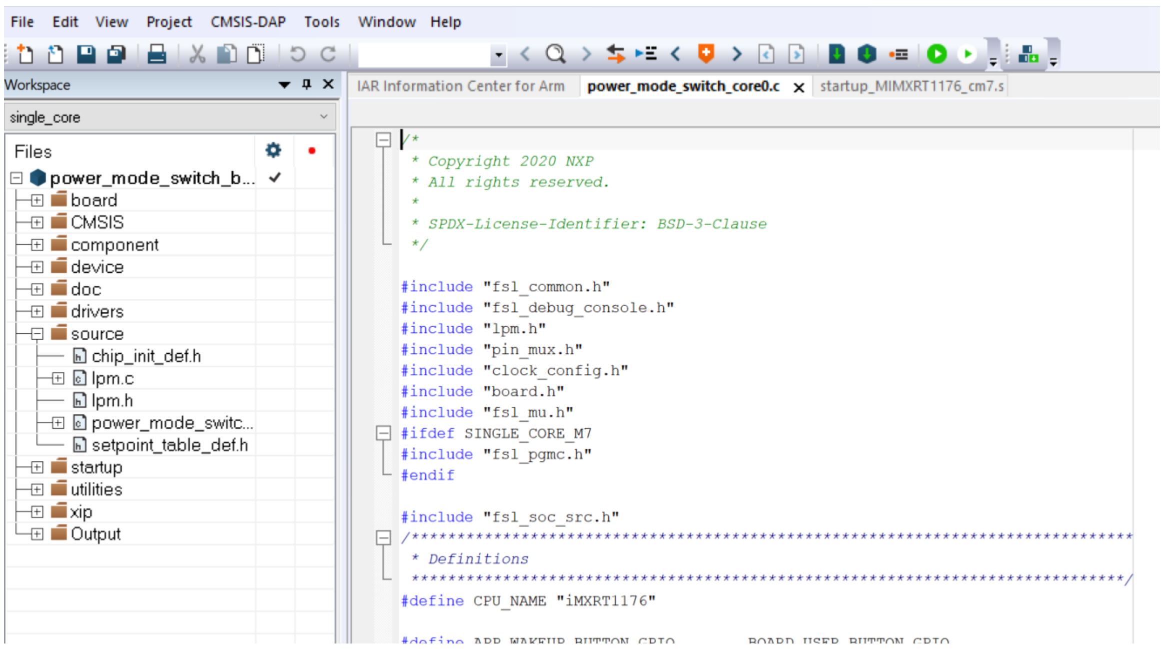
6.1 Run a demo using IAR

The following steps illustrate the power_mode_switch demo from SDK 2.9.0.

1. The location of the desired demo is:

```
<install_dir>\boards\evkmimxrt1170\demo_apps\power_mode_switch\bm\core0\iar\
power_mode_switch_bm_core0.eww
```

2. Import the demo into the IAR IDE.



```
/*
 * Copyright 2020 NXP
 * All rights reserved.
 *
 * SPDX-License-Identifier: BSD-3-Clause
 */

#include "fsl_common.h"
#include "fsl_debug_console.h"
#include "lpm.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "board.h"
#include "fsl_mu.h"
#ifdef SINGLE_CORE_M7
#include "fsl_pgmc.h"
#endif

#include "fsl_soc_src.h"
/*******************************************************************************
 * Definitions
 ******************************************************************************/
#define CPU_NAME "iMXRT1176"

#define APP_WAKEUP_BUTTON_GPIO        BOARD_USER_BUTTON_GPIO
```

Figure 3. Open the demo in IAR

3. Debug the default build target as shown in Figure 4

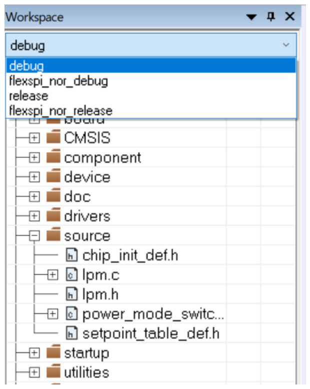


Figure 4. Build target

4. Select **Project > Edit Configuration**.

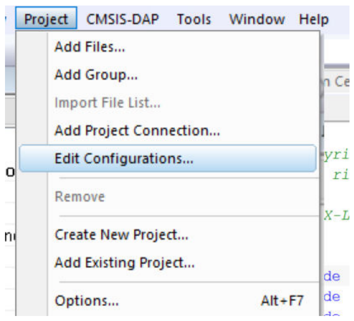


Figure 5. Edit configurations

5. Add a build target.

a. Click the **New** button.

b. Name it as single_core.

c. Click **OK**.

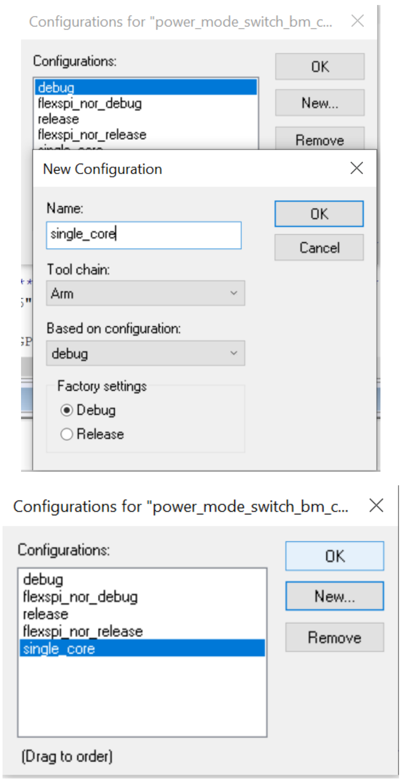


Figure 6. Add a build target

6. Enter the project option and configure the following settings.

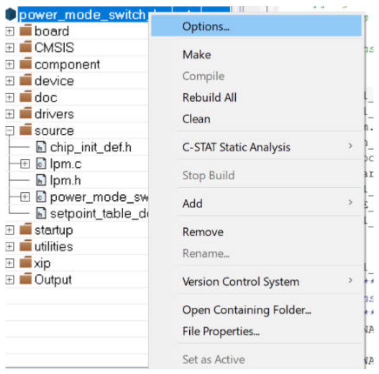
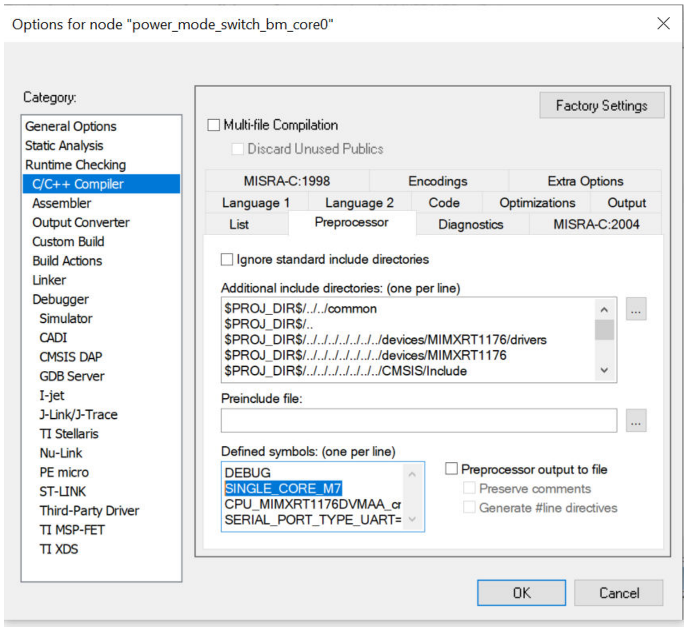


Figure 7. Project options

a. Add the macro 'SINGLE_CORE_M7' in the **Preprocessor of C/C++ Compiler** category.



Options for node "power_mode_switch_bm_core0"

Category: C/C++ Compiler — Preprocessor

Additional include directories: (one per line)

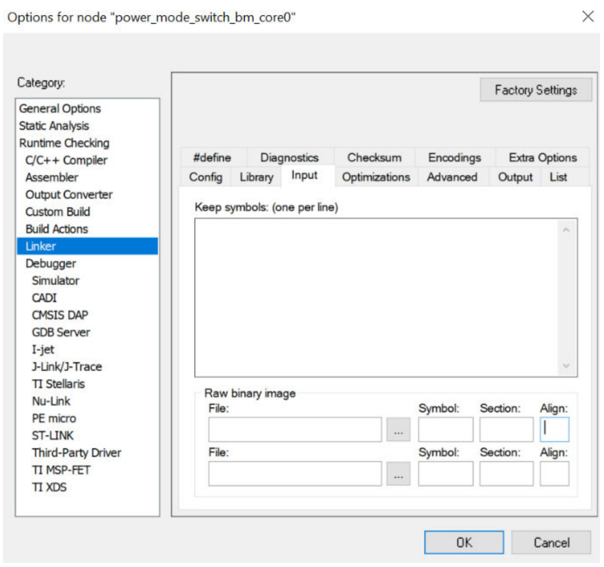
```
$PROJ_DIR$/../../common
$PROJ_DIR$/..
$PROJ_DIR$/../../../../../../../devices/MIMXRT1176/drivers
$PROJ_DIR$/../../../../../../../devices/MIMXRT1176
$PROJ_DIR$/../../../../../../../CMSIS/Include
```

Defined symbols: (one per line)

```
DEBUG
SINGLE_CORE_M7
CPU_MIMXRT1176DVMAA_cr
SERIAL_PORT_TYPE_UART=
```

Figure 8. Add macro to preprocessor

b. Delete all the information in the linker input as shown in Figure 9



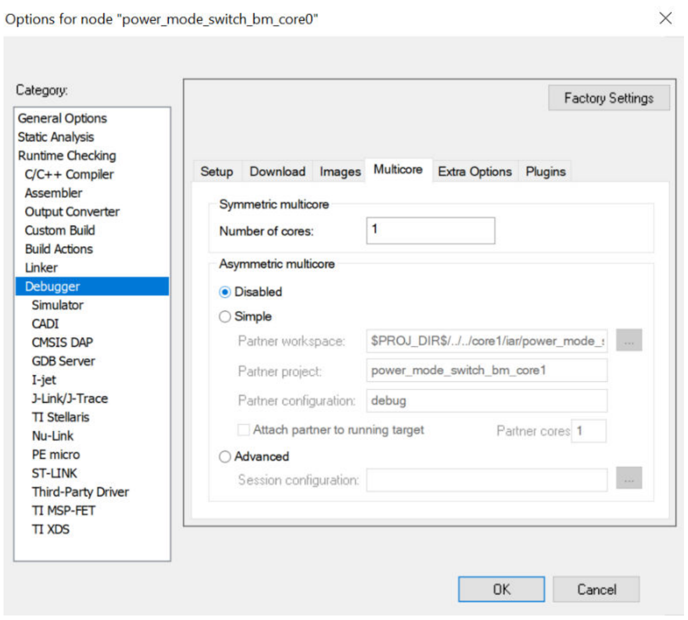
Options for node "power_mode_switch_bm_core0"

Category: Linker — Input

Keep symbols: (one per line)

Figure 9. Clear input

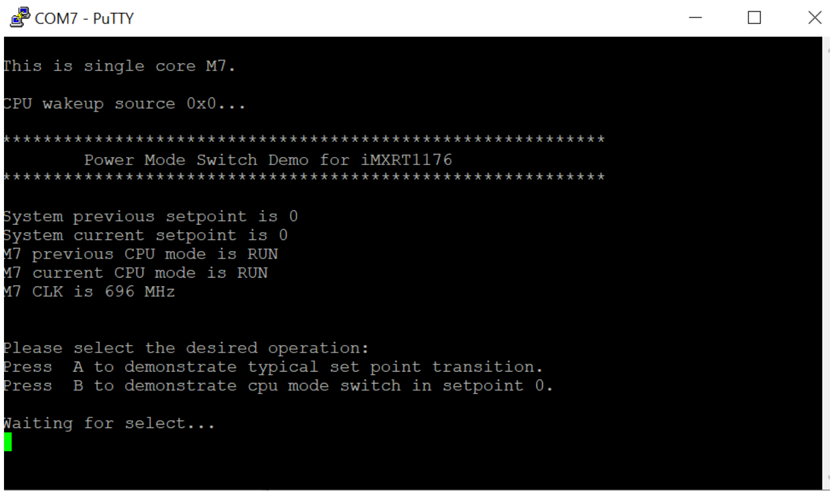
c. In the Debugger category, disable the multicore.



Symmetric multicore
Number of cores: 1
Asymmetric multicore
(●) Disabled
( ) Simple
Partner workspace: $PROJ_DIR$/../../core1/iar/power_mode_s
Partner project: power_mode_switch_bm_core1
Partner configuration: debug
[ ] Attach partner to running target   Partner cores: 1
( ) Advanced
Session configuration:

Figure 10. Disabled multicore

7. Debug and run the code.
8. On successful debug, the terminal window prompts the following message.



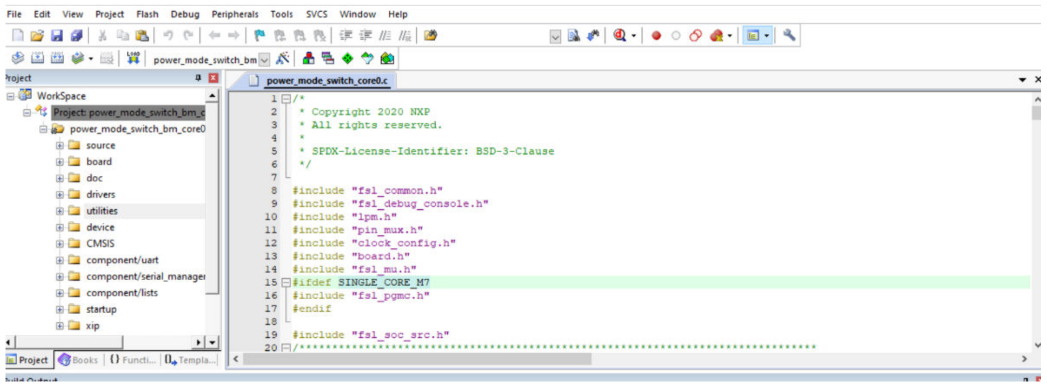
```
This is single core M7.

CPU wakeup source 0x0...

***********************************************************
        Power Mode Switch Demo for iMXRT1176
***********************************************************

System previous setpoint is 0
System current setpoint is 0
M7 previous CPU mode is RUN
M7 current CPU mode is RUN
M7 CLK is 696 MHz


Please select the desired operation:
Press  A to demonstrate typical set point transition.
Press  B to demonstrate cpu mode switch in setpoint 0.

Waiting for select...
```

Figure 11. Terminal output

6.2 Run a demo using Keil® MDK/µVision

1. Open the power_mode_switch project by using Keil. The project location is:

```
<install_dir>\boards\evkmimxrt1170\demo_apps\power_mode_switch\bm\core0\mdk\power_mode_switch_bm_
core0.uvmpw
```



```
/*
 * Copyright 2020 NXP
 * All rights reserved.
 *
 * SPDX-License-Identifier: BSD-3-Clause
 */

#include "fsl_common.h"
#include "fsl_debug_console.h"
#include "lpm.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "board.h"
#include "fsl_mu.h"
#ifdef SINGLE_CORE_M7
#include "fsl_pgmc.h"
#endif

#include "fsl_soc_src.h"
/*******************************************************************************
```

Figure 12. Open the demo

2. Change the target to flexsip_nor_debug.

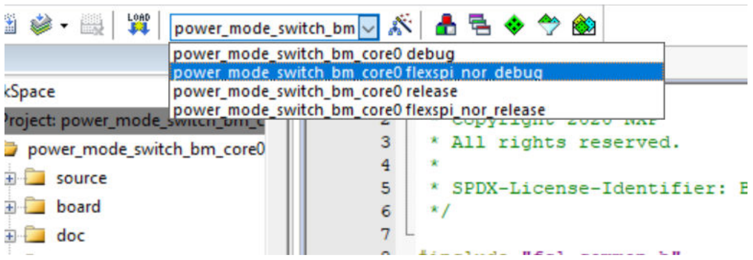
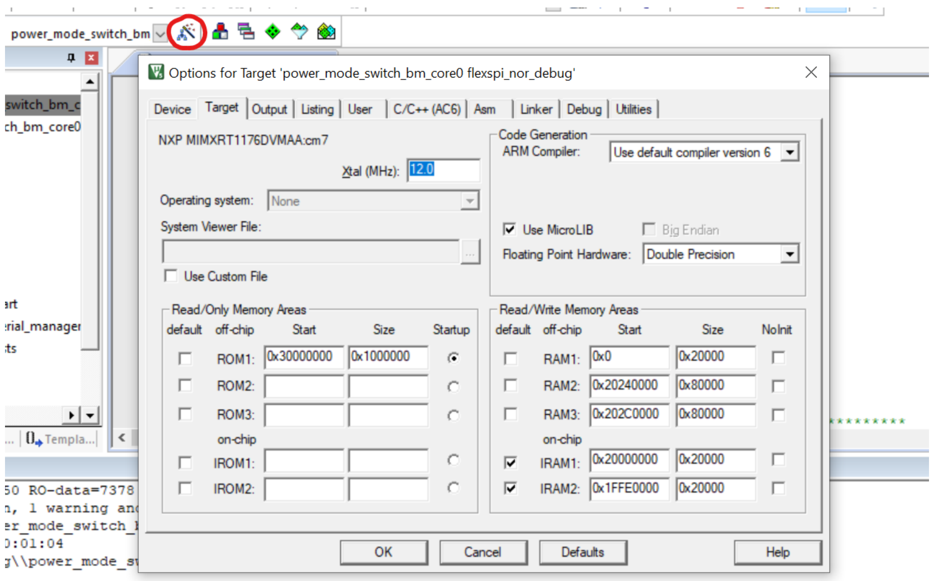

power_mode_switch_bm
power_mode_switch_bm_core0 debug
power_mode_switch_bm_core0 flexspi_nor_debug
power_mode_switch_bm_core0 release
power_mode_switch_bm_core0 flexspi_nor_release
kSpace
Project: power_mode_switch_bm_c
power_mode_switch_bm_core0
source
board
doc
* All rights reserved.
*
* SPDX-License-Identifier: B
*/

Figure 13. Change the target

3. Open the options for the target.


power_mode_switch_bm
Device Target Output Listing User C/C++ (AC6) Asm Linker Debug Utilities
OK Cancel Defaults Help

Options for Target 'power_mode_switch_bm_core0 flexspi_nor_debug'

NXP MIMXRT1176DVMAA:cm7

Xtal (MHz): 12.0

Operating system: None

System Viewer File:

☐ Use Custom File

Code Generation

ARM Compiler: Use default compiler version 6

☑ Use MicroLIB ☐ Big Endian

Floating Point Hardware: Double Precision

Read/Only Memory Areas

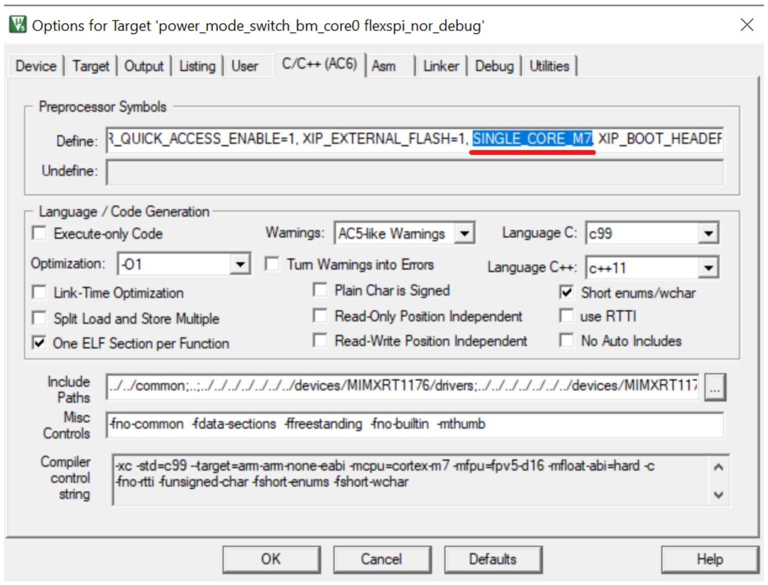
default	off-chip	Start	Size	Startup
☐	ROM1:	0x30000000	0x1000000	●
☐	ROM2:			○
☐	ROM3:			○
	on-chip			
☐	IROM1:			○
☐	IROM2:			○

Read/Write Memory Areas

default	off-chip	Start	Size	NoInit
☐	RAM1:	0x0	0x20000	☐
☐	RAM2:	0x20240000	0x80000	☐
☐	RAM3:	0x202C0000	0x80000	☐
	on-chip			
☑	IRAM1:	0x20000000	0x20000	☐
☑	IRAM2:	0x1FFE0000	0x20000	☐

Figure 14. Open options

4. In C/C++ section, add ‘SINGLE_CORE_M7’ to the **Define** field.



Options for Target 'power_mode_switch_bm_core0 flexspi_nor_debug'

Device | Target | Output | Listing | User | C/C++ (AC6) | Asm | Linker | Debug | Utilities

Preprocessor Symbols

Define: R_QUICK_ACCESS_ENABLE=1, XIP_EXTERNAL_FLASH=1, SINGLE_CORE_M7 XIP_BOOT_HEADEF

Undefine:

Language / Code Generation

- [ ] Execute-only Code
- Warnings: AC5-like Warnings
- Language C: c99
- Optimization: -O1
- [ ] Turn Warnings into Errors
- Language C++: c++11
- [ ] Link-Time Optimization
- [ ] Plain Char is Signed
- [✓] Short enums/wchar
- [ ] Split Load and Store Multiple
- [ ] Read-Only Position Independent
- [ ] use RTTI
- [✓] One ELF Section per Function
- [ ] Read-Write Position Independent
- [ ] No Auto Includes

Include Paths: ../../common;..;../../../../../../devices/MIMXRT1176/drivers;../../../../../../devices/MIMXRT117

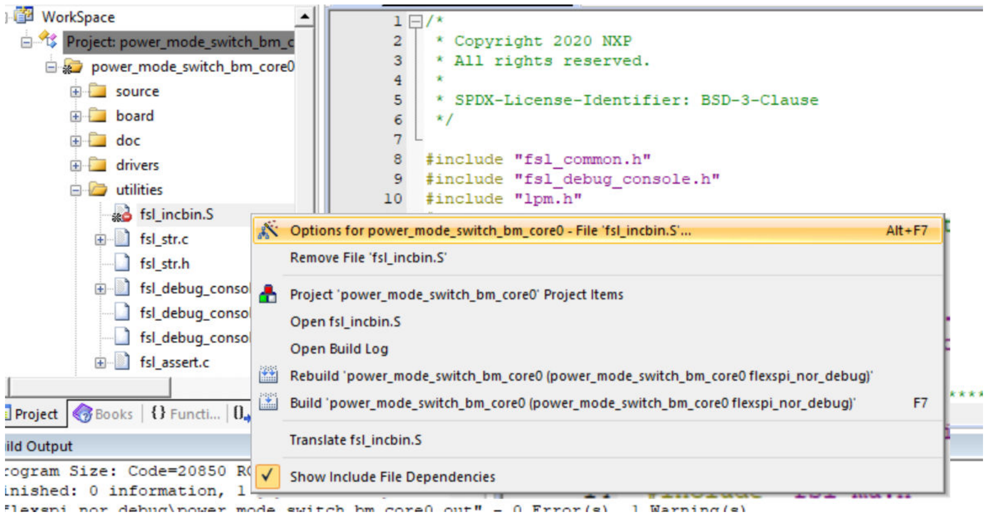
Misc Controls: -fno-common -fdata-sections -ffreestanding -fno-builtin -mthumb

Compiler control string: -xc -std=c99 --target=arm-arm-none-eabi -mcpu=cortex-m7 -mfpu=fpv5-d16 -mfloat-abi=hard -c -fno-rtti -funsigned-char -fshort-enums -fshort-wchar

OK | Cancel | Defaults | Help

Figure 15. Add macro define

5. Open the options for the file fsl_incbin.S under utilities.



WorkSpace
- Project: power_mode_switch_bm_c
  - power_mode_switch_bm_core0
    - source
    - board
    - doc
    - drivers
    - utilities
      - fsl_incbin.S
      - fsl_str.c
      - fsl_str.h
      - fsl_debug_consol
      - fsl_debug_consol
      - fsl_debug_consol
      - fsl_assert.c

Project | Books | {} Functi... | 0

```
/*
 * Copyright 2020 NXP
 * All rights reserved.
 *
 * SPDX-License-Identifier: BSD-3-Clause
 */

#include "fsl_common.h"
#include "fsl_debug_console.h"
#include "lpm.h"
```

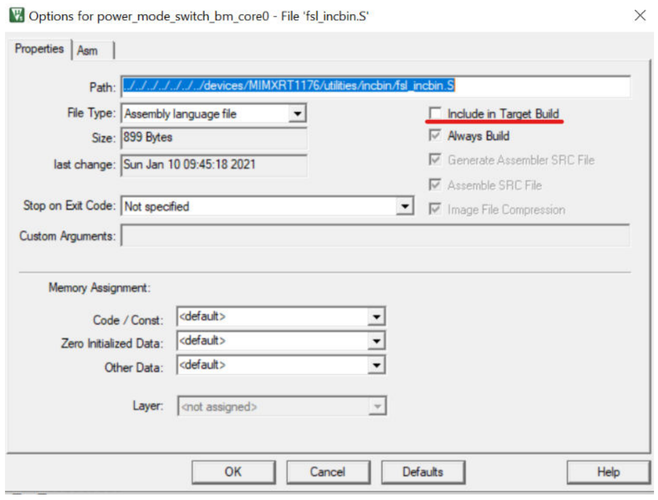
- Options for power_mode_switch_bm_core0 - File 'fsl_incbin.S'... Alt+F7
- Remove File 'fsl_incbin.S'
- Project 'power_mode_switch_bm_core0' Project Items
- Open fsl_incbin.S
- Open Build Log
- Rebuild 'power_mode_switch_bm_core0 (power_mode_switch_bm_core0 flexspi_nor_debug)'
- Build 'power_mode_switch_bm_core0 (power_mode_switch_bm_core0 flexspi_nor_debug)' F7
- Translate fsl_incbin.S
- ✓ Show Include File Dependencies

ild Output

```
rogram Size: Code=20850 RO
inished: 0 information, 1
flexspi_nor_debug\power_mode_switch_bm_core0.out" - 0 Error(s), 1 Warning(s).
```

Figure 16. Open options for fsl_incbin.S

6. Deselect the **Include in Target Build** checkbox.



Options for power_mode_switch_bm_core0 - File 'fsl_incbin.S'

Properties | Asm

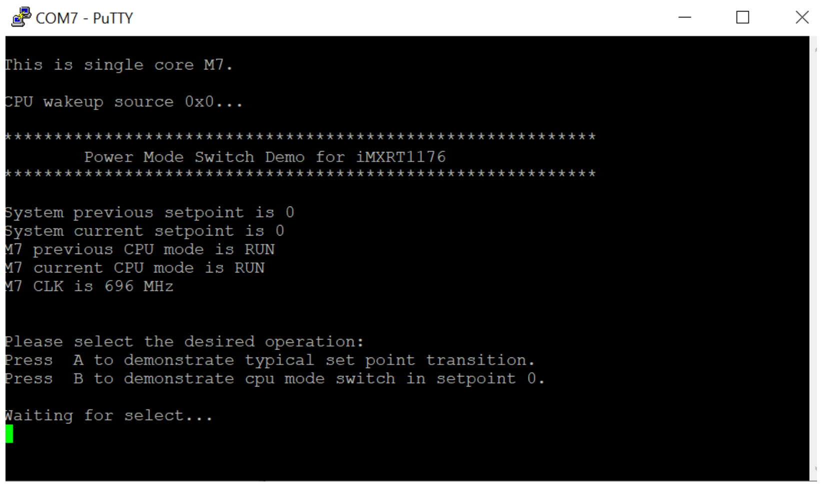
Path: ../../../../../../devices/MIMXRT1176/utilities/incbin/fsl_incbin.S
File Type: Assembly language file
Size: 899 Bytes
last change: Sun Jan 10 09:45:18 2021
Stop on Exit Code: Not specified
Custom Arguments:

[ ] Include in Target Build
[✓] Always Build
[✓] Generate Assembler SRC File
[✓] Assemble SRC File
[✓] Image File Compression

Memory Assignment:
Code / Const: <default>
Zero Initialized Data: <default>
Other Data: <default>
Layer: <not assigned>

Figure 17. Disabled include in Target Build

7. Debug and run the project.



COM7 - PuTTY

```
This is single core M7.

CPU wakeup source 0x0...

***********************************************************
        Power Mode Switch Demo for iMXRT1176
***********************************************************

System previous setpoint is 0
System current setpoint is 0
M7 previous CPU mode is RUN
M7 current CPU mode is RUN
M7 CLK is 696 MHz


Please select the desired operation:
Press  A to demonstrate typical set point transition.
Press  B to demonstrate cpu mode switch in setpoint 0.

Waiting for select...
```

Figure 18. Terminal output

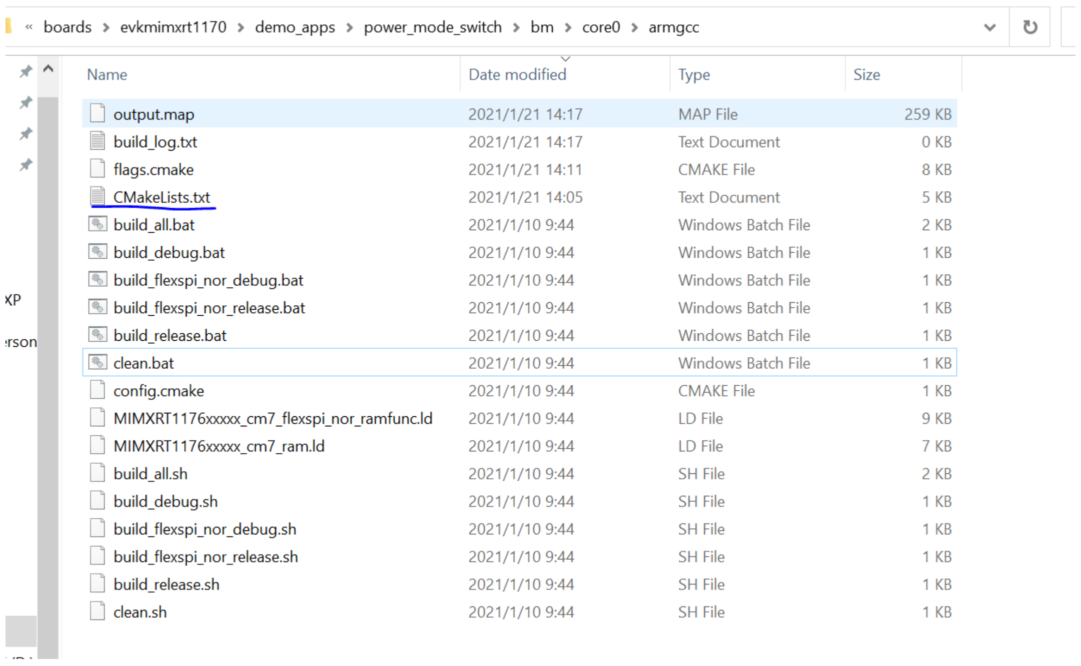
6.3 Run a demo using Arm® GCC

This section shows how to run a demo in a single-core mode by using GCC IDE.

NOTE

Ensure that CMake and Segger J-Link are installed.

1. Open the cmakelist.txt in the project directory.



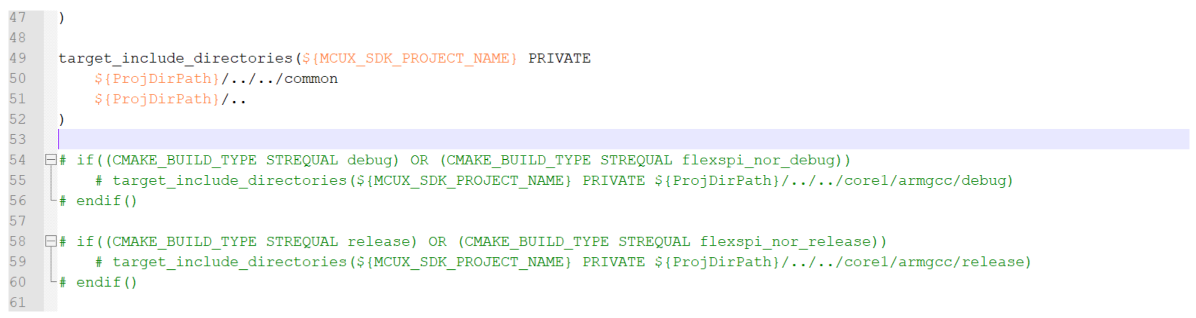
« boards › evkmimxrt1170 › demo_apps › power_mode_switch › bm › core0 › armgcc

Name	Date modified	Type	Size
output.map	2021/1/21 14:17	MAP File	259 KB
build_log.txt	2021/1/21 14:17	Text Document	0 KB
flags.cmake	2021/1/21 14:11	CMAKE File	8 KB
CMakeLists.txt	2021/1/21 14:05	Text Document	5 KB
build_all.bat	2021/1/10 9:44	Windows Batch File	2 KB
build_debug.bat	2021/1/10 9:44	Windows Batch File	1 KB
build_flexspi_nor_debug.bat	2021/1/10 9:44	Windows Batch File	1 KB
build_flexspi_nor_release.bat	2021/1/10 9:44	Windows Batch File	1 KB
build_release.bat	2021/1/10 9:44	Windows Batch File	1 KB
clean.bat	2021/1/10 9:44	Windows Batch File	1 KB
config.cmake	2021/1/10 9:44	CMAKE File	1 KB
MIMXRT1176xxxxx_cm7_flexspi_nor_ramfunc.ld	2021/1/10 9:44	LD File	9 KB
MIMXRT1176xxxxx_cm7_ram.ld	2021/1/10 9:44	LD File	7 KB
build_all.sh	2021/1/10 9:44	SH File	2 KB
build_debug.sh	2021/1/10 9:44	SH File	1 KB
build_flexspi_nor_debug.sh	2021/1/10 9:44	SH File	1 KB
build_flexspi_nor_release.sh	2021/1/10 9:44	SH File	1 KB
build_release.sh	2021/1/10 9:44	SH File	1 KB
clean.sh	2021/1/10 9:44	SH File	1 KB

Figure 19. Open CMakeLists.txt

2. Delete or comments the following three parts of the code need.

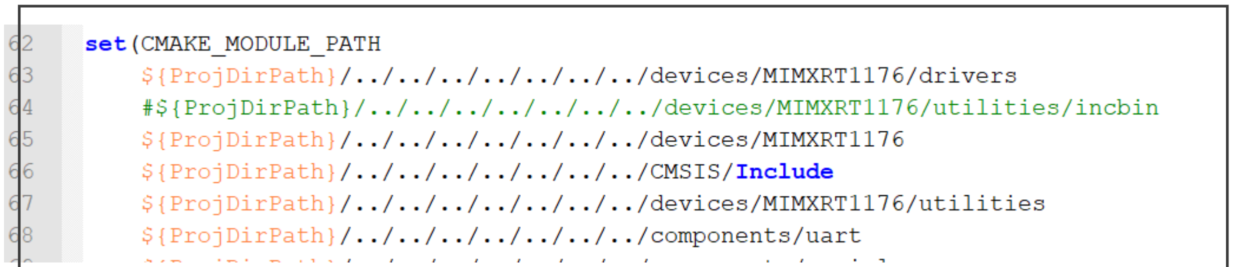
   a. The code that include core1.



```
47  )
48
49  target_include_directories(${MCUX_SDK_PROJECT_NAME} PRIVATE
50      ${ProjDirPath}/../../common
51      ${ProjDirPath}/..
52  )
53
54  # if((CMAKE_BUILD_TYPE STREQUAL debug) OR (CMAKE_BUILD_TYPE STREQUAL flexspi_nor_debug))
55      # target_include_directories(${MCUX_SDK_PROJECT_NAME} PRIVATE ${ProjDirPath}/../../core1/armgcc/debug)
56  # endif()
57
58  # if((CMAKE_BUILD_TYPE STREQUAL release) OR (CMAKE_BUILD_TYPE STREQUAL flexspi_nor_release))
59      # target_include_directories(${MCUX_SDK_PROJECT_NAME} PRIVATE ${ProjDirPath}/../../core1/armgcc/release)
60  # endif()
61
```

Figure 20. Delete or comment line 54 through line 60

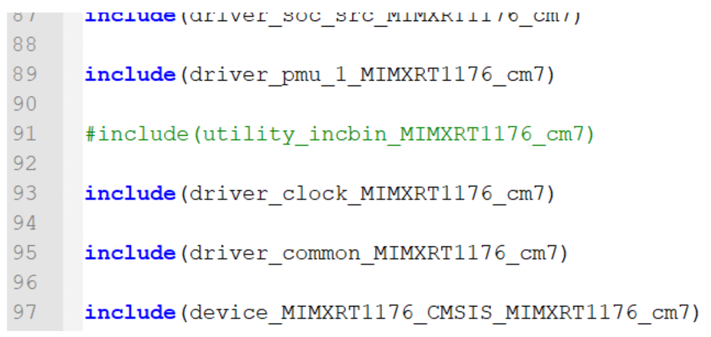
   b. The path that include *incbin*.



```
62  set(CMAKE_MODULE_PATH
63      ${ProjDirPath}/../../../../../../../devices/MIMXRT1176/drivers
64      #${ProjDirPath}/../../../../../../../devices/MIMXRT1176/utilities/incbin
65      ${ProjDirPath}/../../../../../../../devices/MIMXRT1176
66      ${ProjDirPath}/../../../../../../../CMSIS/Include
67      ${ProjDirPath}/../../../../../../../devices/MIMXRT1176/utilities
68      ${ProjDirPath}/../../../../../../../components/uart
```

Figure 21. Delete or comment line 64

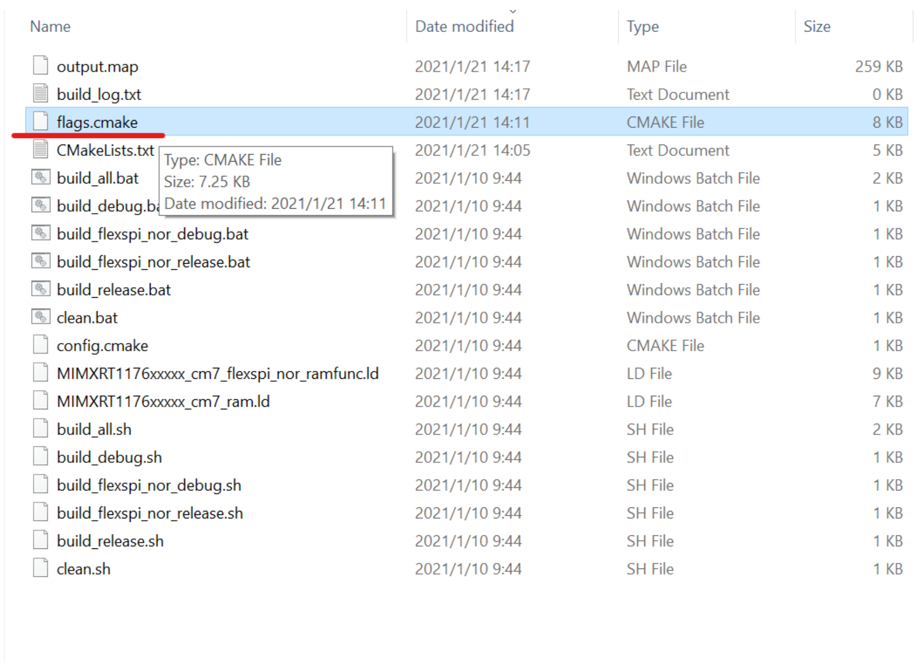
   c. The include for *incbin*.



```
87  include(driver_soc_src_MIMXRT1176_cm7)
88
89  include(driver_pmu_1_MIMXRT1176_cm7)
90
91  #include(utility_incbin_MIMXRT1176_cm7)
92
93  include(driver_clock_MIMXRT1176_cm7)
94
95  include(driver_common_MIMXRT1176_cm7)
96
97  include(device_MIMXRT1176_CMSIS_MIMXRT1176_cm7)
```

Figure 22. Delete or comment line 91

3. Open the file flags.cmake.

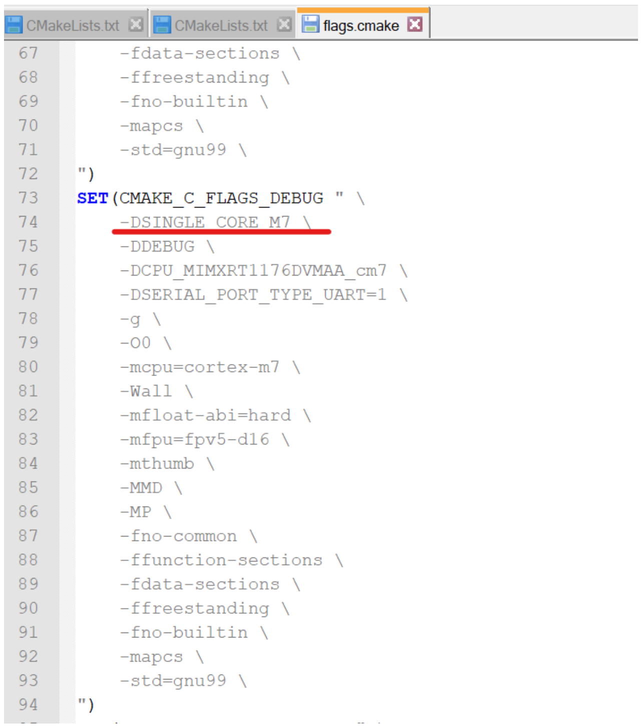


Name	Date modified	Type	Size
output.map	2021/1/21 14:17	MAP File	259 KB
build_log.txt	2021/1/21 14:17	Text Document	0 KB
flags.cmake	2021/1/21 14:11	CMAKE File	8 KB
CMakeLists.txt	2021/1/21 14:05	Text Document	5 KB
build_all.bat	2021/1/10 9:44	Windows Batch File	2 KB
build_debug.ba	2021/1/10 9:44	Windows Batch File	1 KB
build_flexspi_nor_debug.bat	2021/1/10 9:44	Windows Batch File	1 KB
build_flexspi_nor_release.bat	2021/1/10 9:44	Windows Batch File	1 KB
build_release.bat	2021/1/10 9:44	Windows Batch File	1 KB
clean.bat	2021/1/10 9:44	Windows Batch File	1 KB
config.cmake	2021/1/10 9:44	CMAKE File	1 KB
MIMXRT1176xxxxx_cm7_flexspi_nor_ramfunc.ld	2021/1/10 9:44	LD File	9 KB
MIMXRT1176xxxxx_cm7_ram.ld	2021/1/10 9:44	LD File	7 KB
build_all.sh	2021/1/10 9:44	SH File	2 KB
build_debug.sh	2021/1/10 9:44	SH File	1 KB
build_flexspi_nor_debug.sh	2021/1/10 9:44	SH File	1 KB
build_flexspi_nor_release.sh	2021/1/10 9:44	SH File	1 KB
build_release.sh	2021/1/10 9:44	SH File	1 KB
clean.sh	2021/1/10 9:44	SH File	1 KB

Type: CMAKE File
Size: 7.25 KB
Date modified: 2021/1/21 14:11

Figure 23. Open flags.cmake

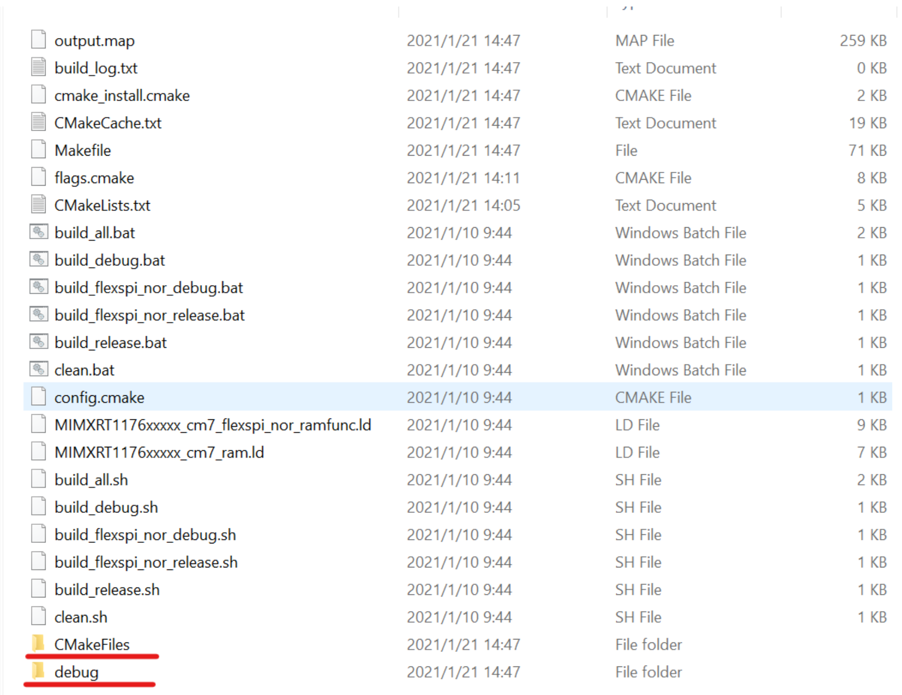
4. Add macro ‘SINGLE_CORE_M7’ to the target build.



```
    -fdata-sections \
    -ffreestanding \
    -fno-builtin \
    -mapcs \
    -std=gnu99 \
")
SET(CMAKE_C_FLAGS_DEBUG " \
    -DSINGLE_CORE_M7 \
    -DDEBUG \
    -DCPU_MIMXRT1176DVMAA_cm7 \
    -DSERIAL_PORT_TYPE_UART=1 \
    -g \
    -O0 \
    -mcpu=cortex-m7 \
    -Wall \
    -mfloat-abi=hard \
    -mfpu=fpv5-d16 \
    -mthumb \
    -MMD \
    -MP \
    -fno-common \
    -ffunction-sections \
    -fdata-sections \
    -ffreestanding \
    -fno-builtin \
    -mapcs \
    -std=gnu99 \
")
```

Figure 24. Add macro to the target build

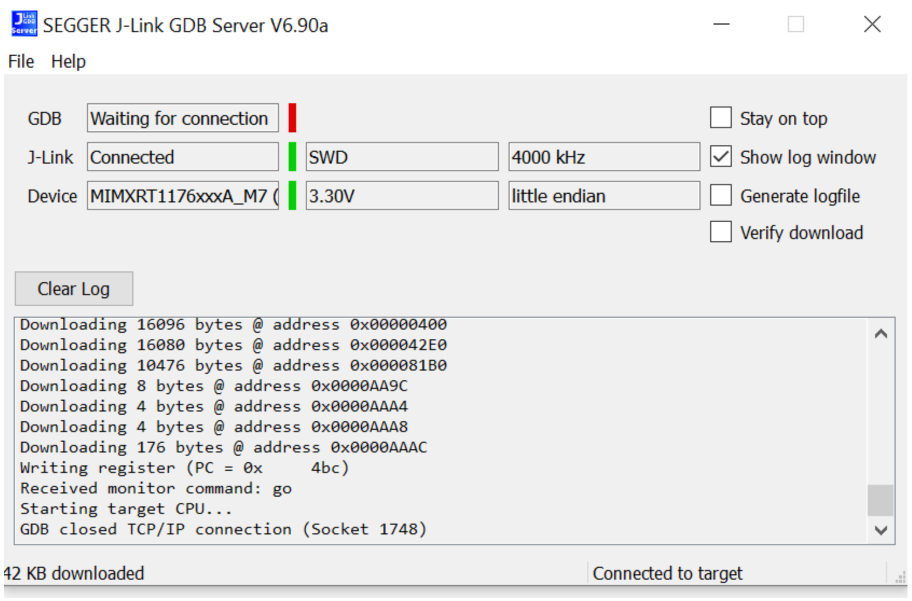
5. Save changes, double-click on **build_debug.bat** to build the target.
6. If successful, there are two new folders appear in the project. The folders are: CmakeFiles, and debug.



output.map	2021/1/21 14:47	MAP File	259 KB
build_log.txt	2021/1/21 14:47	Text Document	0 KB
cmake_install.cmake	2021/1/21 14:47	CMAKE File	2 KB
CMakeCache.txt	2021/1/21 14:47	Text Document	19 KB
Makefile	2021/1/21 14:47	File	71 KB
flags.cmake	2021/1/21 14:11	CMAKE File	8 KB
CMakeLists.txt	2021/1/21 14:05	Text Document	5 KB
build_all.bat	2021/1/10 9:44	Windows Batch File	2 KB
build_debug.bat	2021/1/10 9:44	Windows Batch File	1 KB
build_flexspi_nor_debug.bat	2021/1/10 9:44	Windows Batch File	1 KB
build_flexspi_nor_release.bat	2021/1/10 9:44	Windows Batch File	1 KB
build_release.bat	2021/1/10 9:44	Windows Batch File	1 KB
clean.bat	2021/1/10 9:44	Windows Batch File	1 KB
config.cmake	2021/1/10 9:44	CMAKE File	1 KB
MIMXRT1176xxxxx_cm7_flexspi_nor_ramfunc.ld	2021/1/10 9:44	LD File	9 KB
MIMXRT1176xxxxx_cm7_ram.ld	2021/1/10 9:44	LD File	7 KB
build_all.sh	2021/1/10 9:44	SH File	2 KB
build_debug.sh	2021/1/10 9:44	SH File	1 KB
build_flexspi_nor_debug.sh	2021/1/10 9:44	SH File	1 KB
build_flexspi_nor_release.sh	2021/1/10 9:44	SH File	1 KB
build_release.sh	2021/1/10 9:44	SH File	1 KB
clean.sh	2021/1/10 9:44	SH File	1 KB
CMakeFiles	2021/1/21 14:47	File folder	
debug	2021/1/21 14:47	File folder	

Figure 25. Demo build successful

7. Open the J-Link GDB server application and connect to the device.



GDB: Waiting for connection
J-Link: Connected | SWD | 4000 kHz
Device: MIMXRT1176xxxA_M7 (| 3.30V | little endian

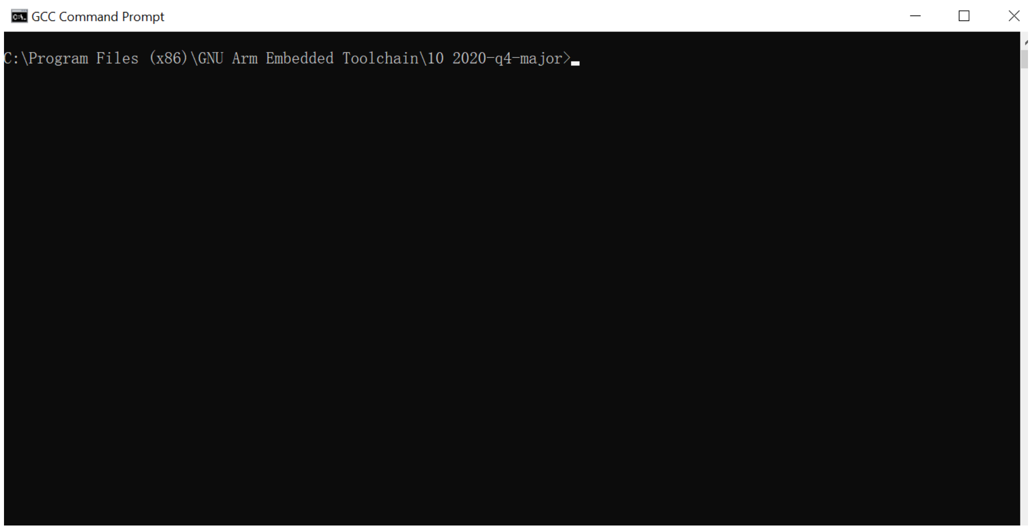
☐ Stay on top
☑ Show log window
☐ Generate logfile
☐ Verify download

```
Downloading 16096 bytes @ address 0x00000400
Downloading 16080 bytes @ address 0x000042E0
Downloading 10476 bytes @ address 0x000081B0
Downloading 8 bytes @ address 0x0000AA9C
Downloading 4 bytes @ address 0x0000AAA4
Downloading 4 bytes @ address 0x0000AAA8
Downloading 176 bytes @ address 0x0000AAAC
Writing register (PC = 0x      4bc)
Received monitor command: go
Starting target CPU...
GDB closed TCP/IP connection (Socket 1748)
```

42 KB downloaded   Connected to target

Figure 26. Segger J-link GDB server after successful connection

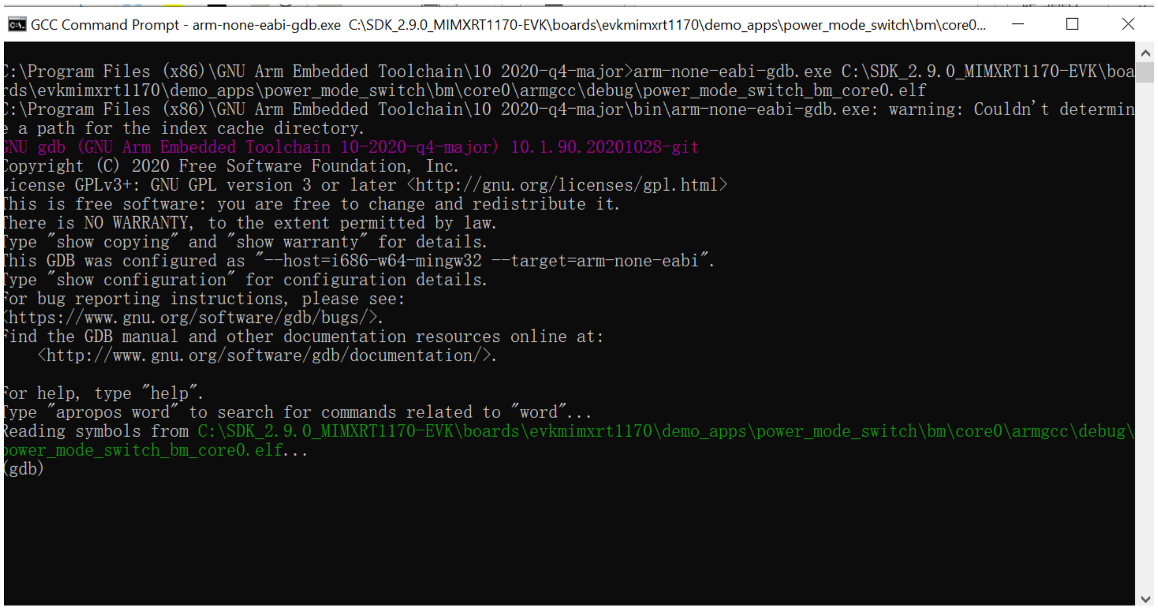
8. Open the GCC ARM Embedded tool chain command window.
9. To launch the window, from the Windows operating system select, **Start menu > GNU Tools ARM Embedded <version> > GCC Command Prompt**.



```
C:\Program Files (x86)\GNU Arm Embedded Toolchain\10 2020-q4-major>_
```

Figure 27. Open gcc command prompt

10. Run the arm-none-eabi-gdb.exe <application_name>.elf. The .elf file appears in the debug folder. For this example, it is arm-none-eabi-gdb.exe power_mode_switch_bm_core0.elf.

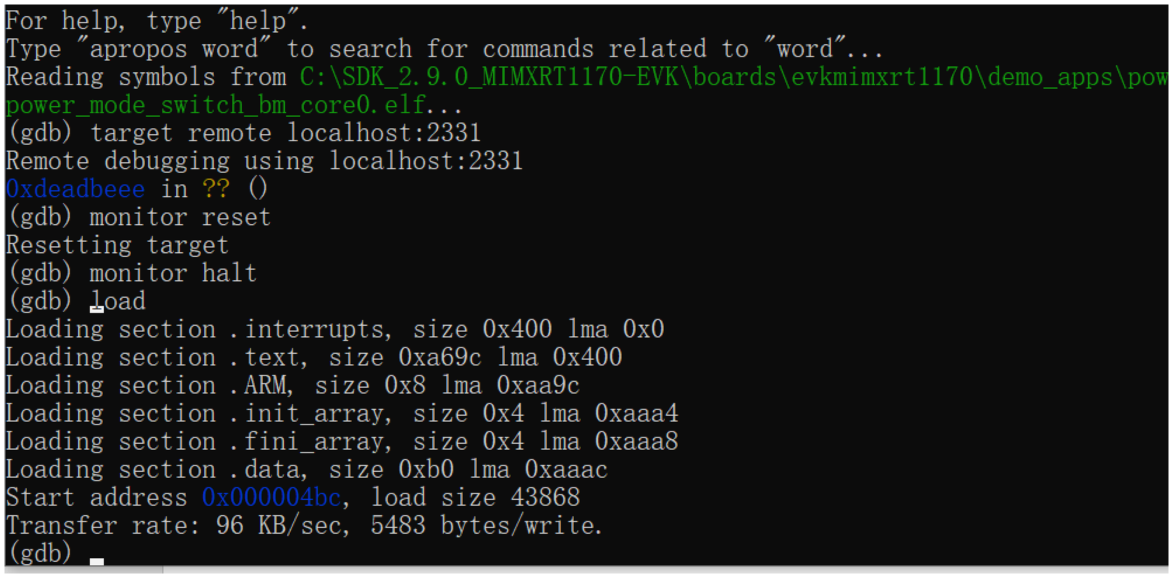


GCC Command Prompt - arm-none-eabi-gdb.exe C:\SDK_2.9.0_MIMXRT1170-EVK\boards\evkmimxrt1170\demo_apps\power_mode_switch\bm\core0...

```
:\Program Files (x86)\GNU Arm Embedded Toolchain\10 2020-q4-major>arm-none-eabi-gdb.exe C:\SDK_2.9.0_MIMXRT1170-EVK\boa
rds\evkmimxrt1170\demo_apps\power_mode_switch\bm\core0\armgcc\debug\power_mode_switch_bm_core0.elf
:\Program Files (x86)\GNU Arm Embedded Toolchain\10 2020-q4-major\bin\arm-none-eabi-gdb.exe: warning: Couldn't determin
e a path for the index cache directory.
NU gdb (GNU Arm Embedded Toolchain 10-2020-q4-major) 10.1.90.20201028-git
opyright (C) 2020 Free Software Foundation, Inc.
icense GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
his is free software: you are free to change and redistribute it.
here is NO WARRANTY, to the extent permitted by law.
ype "show copying" and "show warranty" for details.
his GDB was configured as "--host=i686-w64-mingw32 --target=arm-none-eabi".
ype "show configuration" for configuration details.
or bug reporting instructions, please see:
https://www.gnu.org/software/gdb/bugs/>.
ind the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

or help, type "help".
ype "apropos word" to search for commands related to "word"...
eading symbols from C:\SDK_2.9.0_MIMXRT1170-EVK\boards\evkmimxrt1170\demo_apps\power_mode_switch\bm\core0\armgcc\debug\
ower_mode_switch_bm_core0.elf...
(gdb)
```

Figure 28. Run arm-none-eabi-gdb

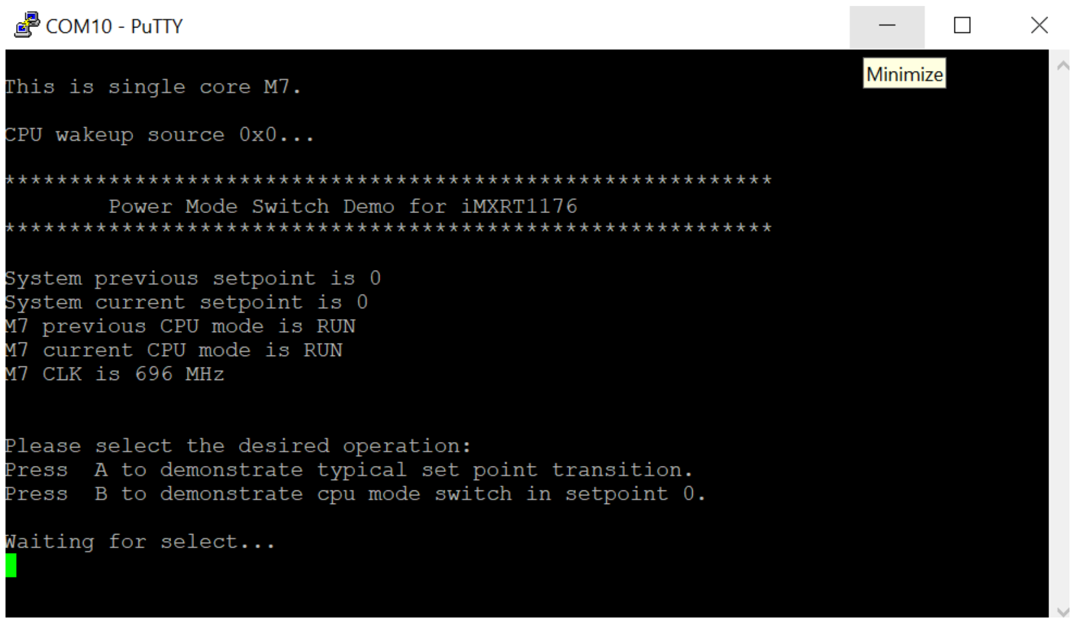
11. Run the following commands:
    a. Target remote localhost: 2331
    b. Monitor reset
    c. Monitor halt
    d. Load



```
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from C:\SDK_2.9.0_MIMXRT1170-EVK\boards\evkmimxrt1170\demo_apps\pow
power_mode_switch_bm_core0.elf...
(gdb) target remote localhost:2331
Remote debugging using localhost:2331
0xdeadbeee in ?? ()
(gdb) monitor reset
Resetting target
(gdb) monitor halt
(gdb) load
Loading section .interrupts, size 0x400 lma 0x0
Loading section .text, size 0xa69c lma 0x400
Loading section .ARM, size 0x8 lma 0xaa9c
Loading section .init_array, size 0x4 lma 0xaaa4
Loading section .fini_array, size 0x4 lma 0xaaa8
Loading section .data, size 0xb0 lma 0xaaac
Start address 0x000004bc, load size 43868
Transfer rate: 96 KB/sec, 5483 bytes/write.
(gdb)
```

Figure 29. Run the debug commands

12. Run the command **monitor go** to start the demo and the result appears on the terminal window.



```
This is single core M7.

CPU wakeup source 0x0...

***********************************************************
        Power Mode Switch Demo for iMXRT1176
***********************************************************

System previous setpoint is 0
System current setpoint is 0
M7 previous CPU mode is RUN
M7 current CPU mode is RUN
M7 CLK is 696 MHz


Please select the desired operation:
Press  A to demonstrate typical set point transition.
Press  B to demonstrate cpu mode switch in setpoint 0.

Waiting for select...
```

Figure 30. Terminal output

Date of release: 02 February 2021

Document identifier: AN13116