Application Note

Creating a Flash Programming Algorithm with the QCU Driver of STAR

Version 1.0
Document ID: ACN-02202203-001
Non-Confidential

Non-Confidential Proprietary Notice

Release Information

Document History

Issue	Date	Confidentiality	Change
A	**25/02/2022**	**Non-Confidential**	**Initial release**

Contents

Non-Confidential Proprietary Notice 1

1 About this document 4

1.1 References 4

1.2 Terms and abbreviations 4

1.3 Conventions and feedback 4

1.3.1 Feedback on this product 5

1.3.2 Feedback on documentation 5

1.3.3 Other information 5

2 Introduction 6

2.1 CMSIS 6

2.2 STAR DFP 6

2.3 QCU 6

2.4 Flash programming algorithm 7

3 Before you begin 9

4 Creating a Flash algorithm 10

5 Using the example project 14

5.1 Generating the new algorithm 14

5.2 Using the new algorithm 16

1 About this document

This Application Note is intended for developers, programmers, and users who use the Arm China STAR *Device Family Pack* (DFP). This Application Note gives you a basic understanding of the *QSPI Controller Unit* (QCU) driver in STAR and provides guidance on how to create a Flash programming algorithm with the QCU driver in Keil MDK.

1.1 References

Reference	Document number	Title
[1]	00903001_0100_00	Arm China Star Processor Technical Reference Manual

1.2 Terms and abbreviations

This document uses the following terms and abbreviations.

Term	Meaning
CMSIS	Cortex Microcontroller Software Interface Standard
DFP	Device Family Pack
QCU	QSPI Controller Unit
QSPI	Quad Serial Peripheral Interface

1.3 Conventions and feedback

The following describes the typographical conventions and how to give feedback:

Convention	Meaning
`monospace`	denotes text that can be entered at the keyboard, such as commands, file and program names, and source code.
`monospace`	denotes a permitted abbreviation for a command or option. The underlined text can be entered instead of the full command or option name.
`monospace italic`	denotes arguments to commands and functions where the argument is to be replaced by a specific value.
`monospace bold`	denotes language keywords when used outside example code.
italic	highlights important notes, introduces special terminology, denotes internal cross-references, and citations.
bold	highlights interface elements, such as menu names. Also used for emphasis in descriptive lists, where appropriate, and for Arm China processor signal names.

1.3.1 Feedback on this product

If you have any comments and suggestions about this product, contact your supplier and give:

- Your name and company.
- The serial number of the product.
- Details of the release you are using.
- Details of the platform you are using, such as the hardware platform, operating system type and version.
- A small standalone sample of code that reproduces the problem.
- A clear explanation of what you expected to happen, and what actually happened.
- The commands you used, including any command-line options.
- Sample output illustrating the problem.
- The version string of the tools, including the version number and build numbers.

1.3.2 Feedback on documentation

If you have comments on the documentation, e-mail errata@armchina.com. Give:

- The title.
- The number, [Document ID Value], [Issue].
- If viewing online, the topic names to which your comments apply.
- If viewing a PDF version of a document, the page numbers to which your comments apply.
- A concise explanation of your comments.

Arm China also welcomes general suggestions for additions and improvements.

1.3.3 Other information

- Arm Glossary, http://infocenter.arm.com/help/topic/com.arm.doc.aeg0014-/index.html.

2 Introduction

2.1 CMSIS

The *Cortex Microcontroller Software Interface Standard* (CMSIS) is a vendor-independent hardware abstraction layer for microcontrollers.

The CMSIS defines generic tool interfaces and enables consistent device support.

The CMSIS provides:

- Simple software interfaces to processor and peripherals.
- A common approach to interface to peripherals, real-time operating systems, and middleware components.

2.2 STAR DFP

For CMSIS compliant toolchains such as Keil MDK and IAR EW, additional software components and support for microcontroller devices are provided by software packs.

A DFP is one of the CMSIS software packs. It indicates that a software pack contains support for microcontroller devices.

A DFP provides essential support for the software targets on a specific device, such as startup, system, linker scripts, and debug configuration.

The STAR processor is the first processor in the Arm China STAR series processor family.

STAR is a fully featured microcontroller class processor based on the Armv8-M mainline architecture with Arm TrustZone technology (depending on the actual core).

In STAR CMSIS DFP v1.3.0 and later, there are example projects of STAR application. These example projects can help you quickly build projects and run the application software and then get a better understanding of how to use STAR.

2.3 QCU

The STAR QCU provides a mechanism to load executing programs directly from external Flash memory instead of boot-up from embedding Flash memory. It provides a low-cost and simple method to implement SoC integration.

QCU provides the necessary functionality to a host to communicate with a serial Flash device through the SPI. The unit supports most common serial Flash device instructions, such as read, program, erase, and other custom instructions. The communication with Flash devices is used by commands, which includes five phases—Instruction, Address, Alternate byte, Dummy, and Data. Any of these phases can be configured to be skipped, but at least one of them needs to be present.

QCU is highly flexible and can be configured to support a large number of SPI Flash memories. QCU also supports newer serial Flash devices with densities up to 256MB.

QCU is a specialized communication interface targeting single, dual, or quad SPI Flash memories.

QCU can work in one of the following modes:

- **Direct Read Access mode**: The external Flash memory is mapped to the device address space and is seen by the system as if it was an internal memory. Direct Read Access mode can be used to both access and directly execute code from external Flash memory, and it supports Flash memory XIP mode. After power-on, QCU changes to default Direct Read Access mode to boot up. The operation mode is accessed through AHB-Bus. When in Direct Read mode, any other modes can be inserted at any time.
- **Indirect mode**: All the operations are performed using the registers. This mode allows software to access the internal TX FIFO and RX FIFO directly. The level of FIFO can be configurable. It is used to access the volatile and non-volatile configuration registers, the legacy SPI status registers, other status and protection registers and the Flash ROM content. It is recommended that this mode is used to erase and configure the serial Flash device. When a transaction request is from Q-AHB, the transaction will be waiting until exiting the current Indirect mode into Direct mode.
- **Inactive mode**: This mode is used to disable QSPI-Bus. It offers a *clean* state to set up the QSPI's related register, such as division clock index, command mode, and command type. When a transaction request is from Q-AHB, an error is returned.

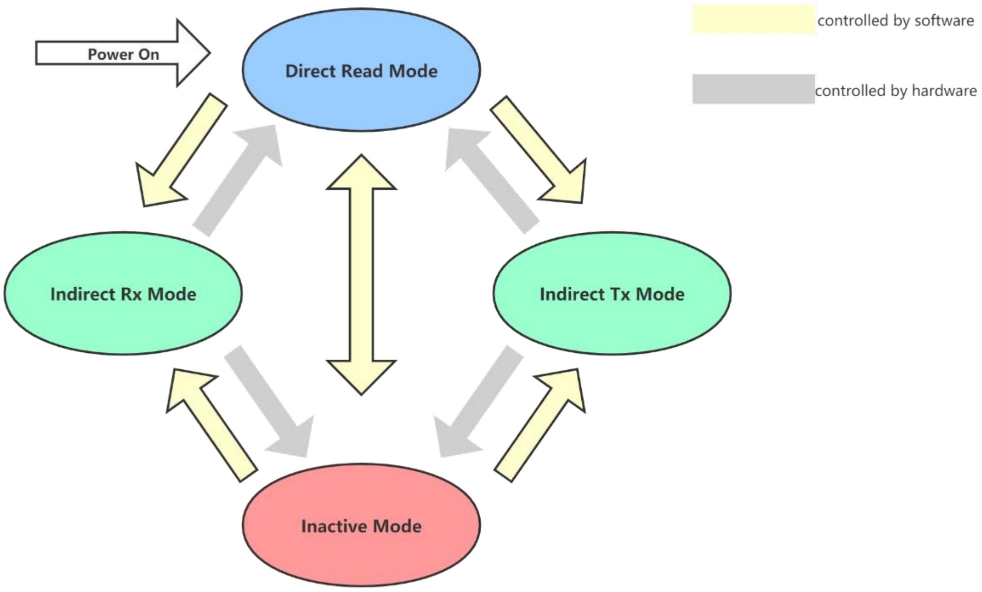


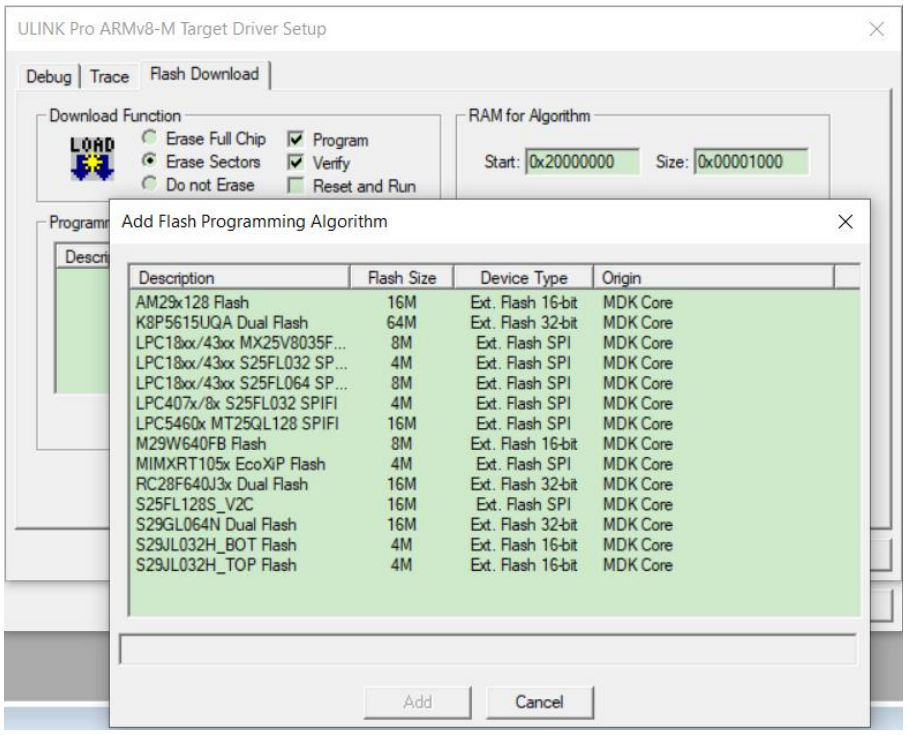
Figure 2-1 QCU modes

The initial state is Direct Read Mode after power-on or cold reset. You can configure QSPI to other states by setting the control register. Indirect Mode is a special state. When the transaction in Indirect Mode is done, hardware will be responsible for changing the state back to the previous one (Inactive or Direct Read, depending on the mode which it enters from). The status register is used to indicate whether the indirect transaction is done.

The Direct Read Mode and Indirect Mode use different sets of registers to construct respective SPI communication. These settings take effect only after the control register is configured, which means that you must confirm the target mode, Direct or Indirect mode, then write the corresponding registers. When the control register is configured, the QCU will load Mode-specific parameters according to the value of the control register.

2.4 Flash programming algorithm

Flash programming algorithms are a piece of software to erase or download applications to Flash devices. A pack with device support usually contains predefined Flash algorithms for programming the devices that are supported by the DFP.



Download Function

○ Erase Full Chip
● Erase Sectors
○ Do not Erase

☑ Program
☑ Verify
□ Reset and Run

RAM for Algorithm

Start: 0x20000000 Size: 0x00001000

Add Flash Programming Algorithm

Description	Flash Size	Device Type	Origin
AM29x128 Flash	16M	Ext. Flash 16-bit	MDK Core
K8P5615UQA Dual Flash	64M	Ext. Flash 32-bit	MDK Core
LPC18xx/43xx MX25V8035F...	8M	Ext. Flash SPI	MDK Core
LPC18xx/43xx S25FL032 SP...	4M	Ext. Flash SPI	MDK Core
LPC18xx/43xx S25FL064 SP...	8M	Ext. Flash SPI	MDK Core
LPC407x/8x S25FL032 SPIFI	4M	Ext. Flash SPI	MDK Core
LPC5460x MT25QL128 SPIFI	16M	Ext. Flash SPI	MDK Core
M29W640FB Flash	8M	Ext. Flash 16-bit	MDK Core
MIMXRT105x EcoXiP Flash	4M	Ext. Flash SPI	MDK Core
RC28F640J3x Dual Flash	16M	Ext. Flash 32-bit	MDK Core
S25FL128S_V2C	16M	Ext. Flash SPI	MDK Core
S29GL064N Dual Flash	16M	Ext. Flash 32-bit	MDK Core
S29JL032H_BOT Flash	4M	Ext. Flash 16-bit	MDK Core
S29JL032H_TOP Flash	4M	Ext. Flash 16-bit	MDK Core

Figure 2-2 Predefined programming algorithms

But for a new Flash device, you need to create an algorithm according to a template in the ARM:CMSIS pack. For more information about how to create an algorithm, see *4 Creating a Flash algorithm*.

In STAR DFP v2.0.0 and later versions, a new Flash programing algorithm SST26VF064B.FLM is added and automatically presented in the options of the **Add Flash Programming Algorithm** dialog box.

3 Before you begin

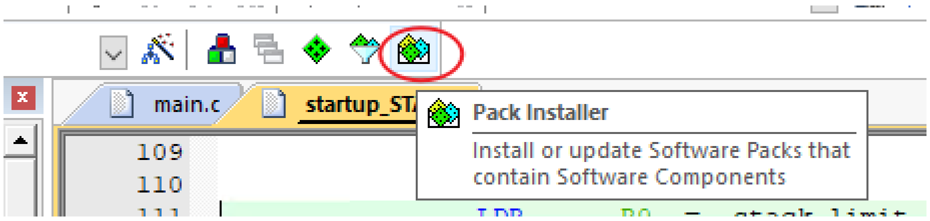
The example project of the application software runs on an MPS3 FPGA board. The generated algorithm is only applicable to STAR QCU and SST26VF064B Flash.

Before using the example project, you need to:

- Ensure that you have an MPS3 FPGA board and have a STAR-based device implemented with QCU on the board.
- Check the STAR CMSIS DFP version which should be v2.0.0 or later.

To check the STAR CMSIS DFP version:

1. Start MDK.
2. On the toolbar, click the **Pack Installer** icon.

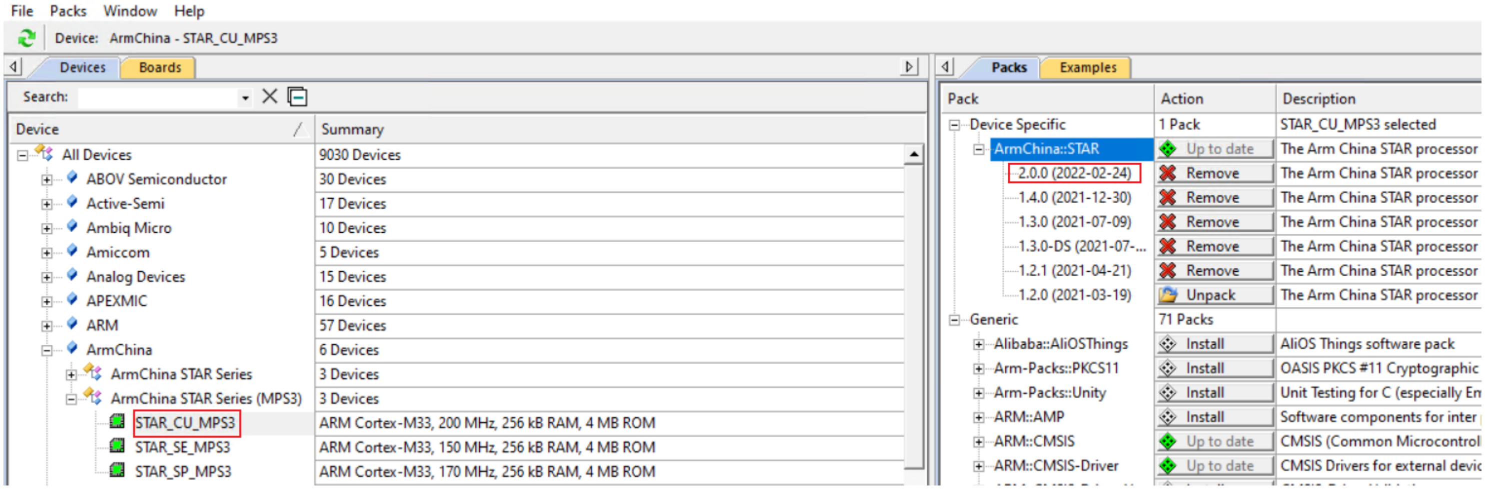


Pack Installer

Install or update Software Packs that contain Software Components

3. On the **Devices** tab, select a device (for example, STAR_CU_MPS3) and check the version of the installed pack.

   As shown in the following figure, the version of the ArmChinaSTAR pack should be 2.0.0 or later.



Device: ArmChina - STAR_CU_MPS3

Device	Summary
All Devices	9030 Devices
ABOV Semiconductor	30 Devices
Active-Semi	17 Devices
Ambiq Micro	10 Devices
Amiccom	5 Devices
Analog Devices	15 Devices
APEXMIC	16 Devices
ARM	57 Devices
ArmChina	6 Devices
ArmChina STAR Series	3 Devices
ArmChina STAR Series (MPS3)	3 Devices
STAR_CU_MPS3	ARM Cortex-M33, 200 MHz, 256 kB RAM, 4 MB ROM
STAR_SE_MPS3	ARM Cortex-M33, 150 MHz, 256 kB RAM, 4 MB ROM
STAR_SP_MPS3	ARM Cortex-M33, 170 MHz, 256 kB RAM, 4 MB ROM

Pack	Action	Description
Device Specific	1 Pack	STAR_CU_MPS3 selected
ArmChina::STAR	Up to date	The Arm China STAR processor
2.0.0 (2022-02-24)	Remove	The Arm China STAR processor
1.4.0 (2021-12-30)	Remove	The Arm China STAR processor
1.3.0 (2021-07-09)	Remove	The Arm China STAR processor
1.3.0-DS (2021-07-...	Remove	The Arm China STAR processor
1.2.1 (2021-04-21)	Remove	The Arm China STAR processor
1.2.0 (2021-03-19)	Unpack	The Arm China STAR processor
Generic	71 Packs	
Alibaba::AliOSThings	Install	AliOS Things software pack
Arm-Packs::PKCS11	Install	OASIS PKCS #11 Cryptographic
Arm-Packs::Unity	Install	Unit Testing for C (especially Em
ARM::AMP	Install	Software components for inter
ARM::CMSIS	Up to date	CMSIS (Common Microcontrol
ARM::CMSIS-Driver	Up to date	CMSIS Drivers for external devic

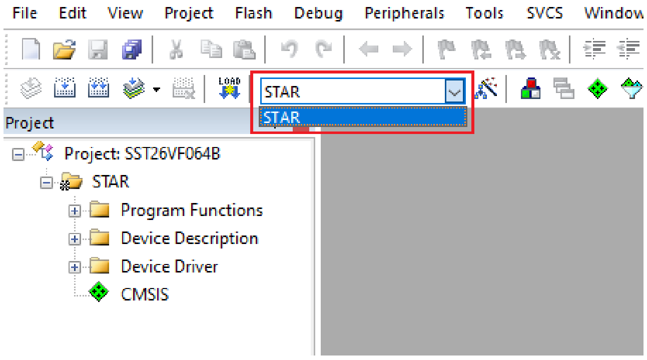
4 Creating a Flash algorithm

Follow these steps to create and configure a Flash programming algorithm:

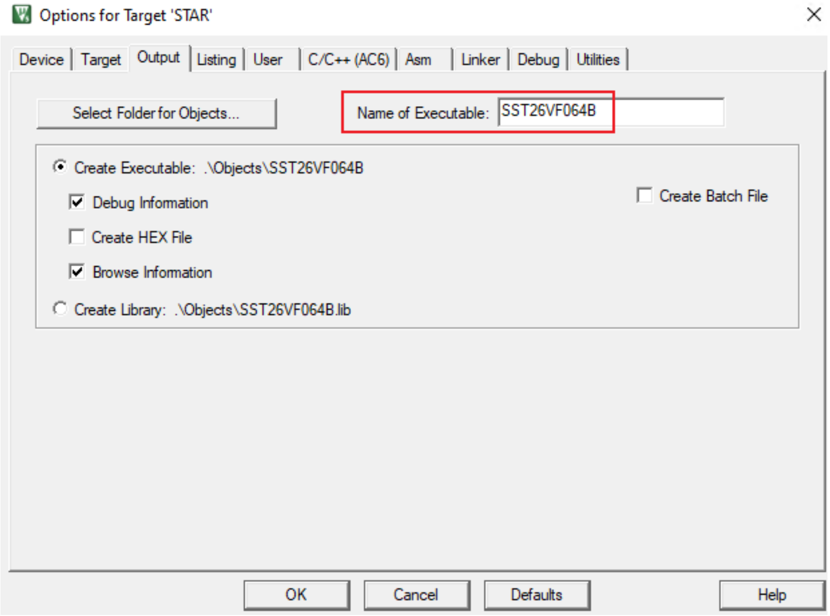
1. Copy the content from the ARM:CMSIS pack folder (usually %CMSIS_PACK_ROOT%\ARM\CMSIS\%version%\Device_Template_Flash) to a new folder.

2. Rename the project file NewDevice.uvprojx to represent the new Flash ROM device name, for example, SST26VF064B.uvprojx.

3. Open the project with µVision. On the toolbar, use the **Select Target** drop-down list to define the processor architecture. STAR fits for Arm China STAR devices. The configuration assumes a little-endian microcontroller. In case of a big-endian microcontroller, click **Project** > **Options for Target** > **Device** to select the correct processor core.



4. Click **Project** > **Options for Target**. In the dialog box that appears, click the **Output** tab and change the value of the **Name of Executable** field to represent the device, for example, SST26VF064B.



Name of Executable: SST26VF064B

● Create Executable: .\Objects\SST26VF064B

☑ Debug Information

☐ Create HEX File

☑ Browse Information

☐ Create Batch File

○ Create Library: .\Objects\SST26VF064B.lib

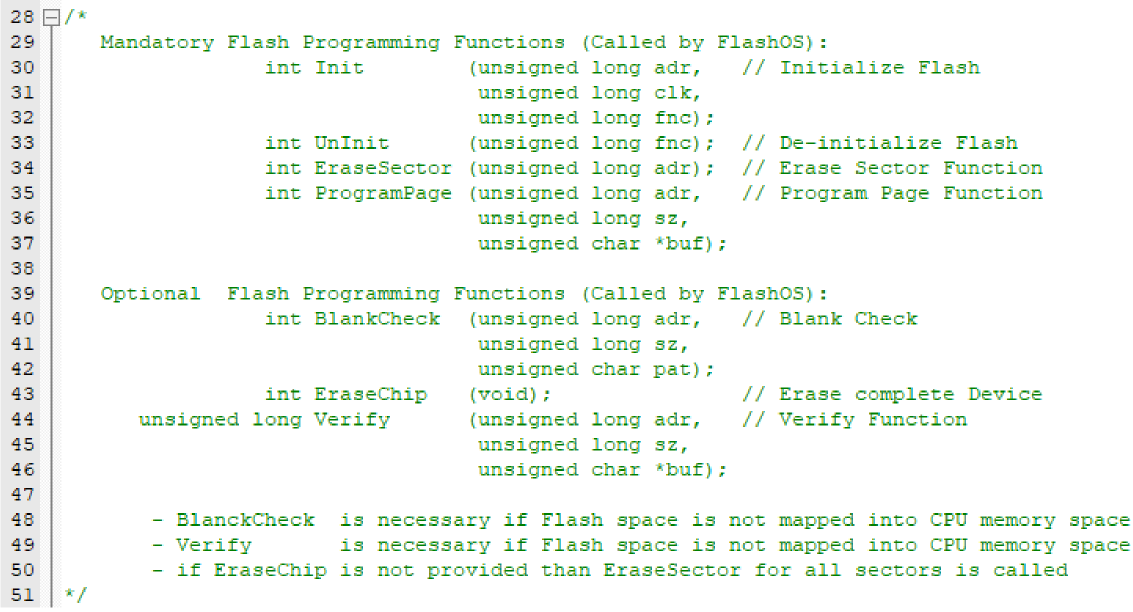
5. Adapt the programming algorithms in the file FlashPrg.c.

The file FlashPrg.c contains the following mandatory Flash programming functions:

- Init()
- UnInit()
- EraseSector()
- ProgramPage()

Optionally, depending on the device features (or to speed up execution), some additional functions can be implemented:

- EraseChip()
- BlankCheck()
- Verify()



```
/*
   Mandatory Flash Programming Functions (Called by FlashOS):
                int Init        (unsigned long adr,   // Initialize Flash
                                 unsigned long clk,
                                 unsigned long fnc);
                int UnInit      (unsigned long fnc);  // De-initialize Flash
                int EraseSector (unsigned long adr);  // Erase Sector Function
                int ProgramPage (unsigned long adr,   // Program Page Function
                                 unsigned long sz,
                                 unsigned char *buf);

   Optional  Flash Programming Functions (Called by FlashOS):
                int BlankCheck  (unsigned long adr,   // Blank Check
                                 unsigned long sz,
                                 unsigned char pat);
                int EraseChip   (void);               // Erase complete Device
      unsigned long Verify      (unsigned long adr,   // Verify Function
                                 unsigned long sz,
                                 unsigned char *buf);

       - BlanckCheck  is necessary if Flash space is not mapped into CPU memory space
       - Verify       is necessary if Flash space is not mapped into CPU memory space
       - if EraseChip is not provided than EraseSector for all sectors is called
*/
```

At this step, you need to adapt the functions listed above with your actual implementation. For example, to fulfill the page programing functionality, you should adapt the common template of the ProgramPage function.

You can walk through the calling chain as shown in the following table to modify the codes in the functions per actual need.

Calling order	Function	File
0	ProgramPage( )	FlashPrg.c
1	App_ProgramPage( )	Flash_App.c
2	Api_ProgramPage( )	Flash_api.c
3	QCU_Write( )	QCU_driver.c
4	Flash_GetParameter( )	Flash_driver.c

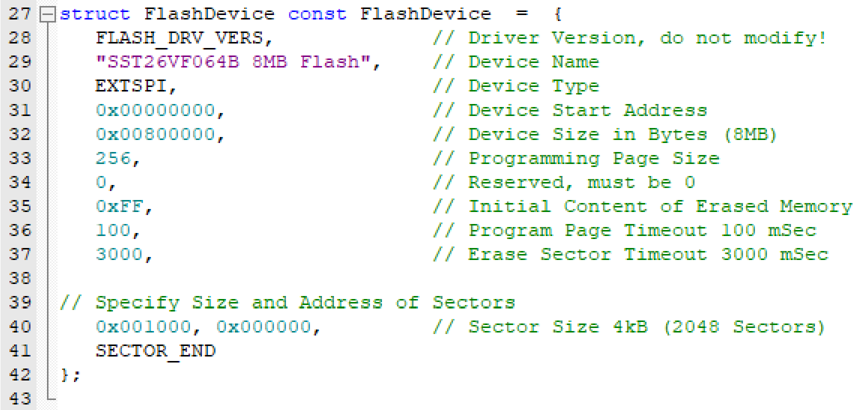
Table 4-1 Calling order and the functions

For more information, see the source code in the following directories:

- Packs\ArmChina\STAR\2.0.0\Examples\Keil\STAR_FLM_MPS3\Source\app
- Packs\ArmChina\STAR\2.0.0\Examples\Keil\STAR_FLM_MPS3\Source\driver
- Packs\ArmChina\STAR\2.0.0\Examples\Keil\STAR_FLM_MPS3\Source\prg
- Packs\ArmChina\STAR\2.0.0\Driver\DriverTemplates

6. Adapt the device parameters in the file FlashDev.c.

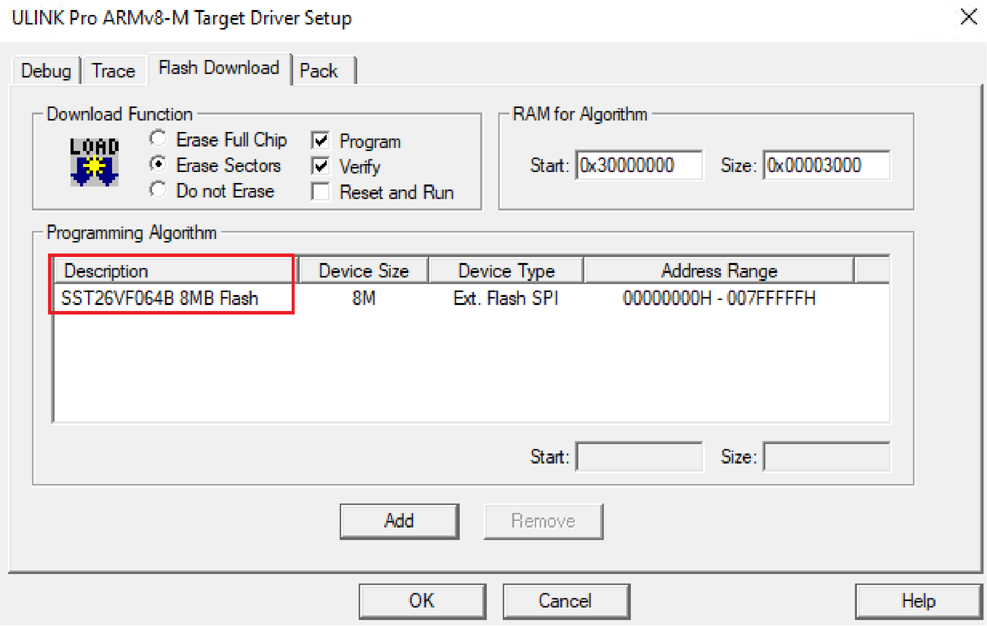
   The file FlashDev.c contains parameter definitions for the Flash programming functions. The **FlashDevice** structure is as follows:



```
struct FlashDevice const FlashDevice  =  {
   FLASH_DRV_VERS,             // Driver Version, do not modify!
   "SST26VF064B 8MB Flash",    // Device Name
   EXTSPI,                     // Device Type
   0x00000000,                 // Device Start Address
   0x00800000,                 // Device Size in Bytes (8MB)
   256,                        // Programming Page Size
   0,                          // Reserved, must be 0
   0xFF,                       // Initial Content of Erased Memory
   100,                        // Program Page Timeout 100 mSec
   3000,                       // Erase Sector Timeout 3000 mSec

// Specify Size and Address of Sectors
   0x001000, 0x000000,         // Sector Size 4kB (2048 Sectors)
   SECTOR_END
};
```

Where,

- *Device Name* is shown in tools to identify the Flash algorithm as follows:



Download Function
○ Erase Full Chip
● Erase Sectors
○ Do not Erase
☑ Program
☑ Verify
□ Reset and Run

RAM for Algorithm
Start: 0x30000000 Size: 0x00003000

Programming Algorithm

Description	Device Size	Device Type	Address Range
SST26VF064B 8MB Flash	8M	Ext. Flash SPI	00000000H - 007FFFFFH

Start: Size:

- *Device Start Address* must be consistent with the memory map.
- *Device Size in Bytes*, *Programming Page Size* and *Sector Size* are based on your Flash device. *Programming Page Size* specifies the block size for programming using the function ProgramPage().

7. Click **Project** > **Build Target** to generate the new Flash programming algorithm.

8. Copy the output file SST26VF064B.FLM to the directory (for example, Keil_v5\ARM\Flash) of Keil MDK before using this algorithm.

   This step is optional if you work with STAR DFP v2.0.0 because SST26VF064B.FLM has been embedded into Keil MDK by installing the pack.

   Note:

   > Creating a Flash programming algorithm with MDK-Lite is not supported.

   For more information, see the documentation at https://arm-software.github.io/CMSIS_5/Pack/html/FlashAlgorithm.html.

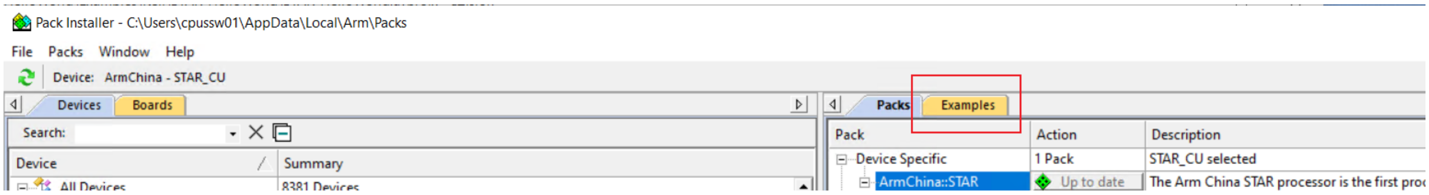
5 Using the example project

An example project, which demonstrates how to create a Flash programming algorithm using the QCU driver in Keil MDK, is available in STAR DFP v2.0.0.

5.1 Generating the new algorithm

Follow these steps to use this example and understand the QCU operations and Flash programming algorithm:

1. In the Pack Installer, click the **Examples** tab.

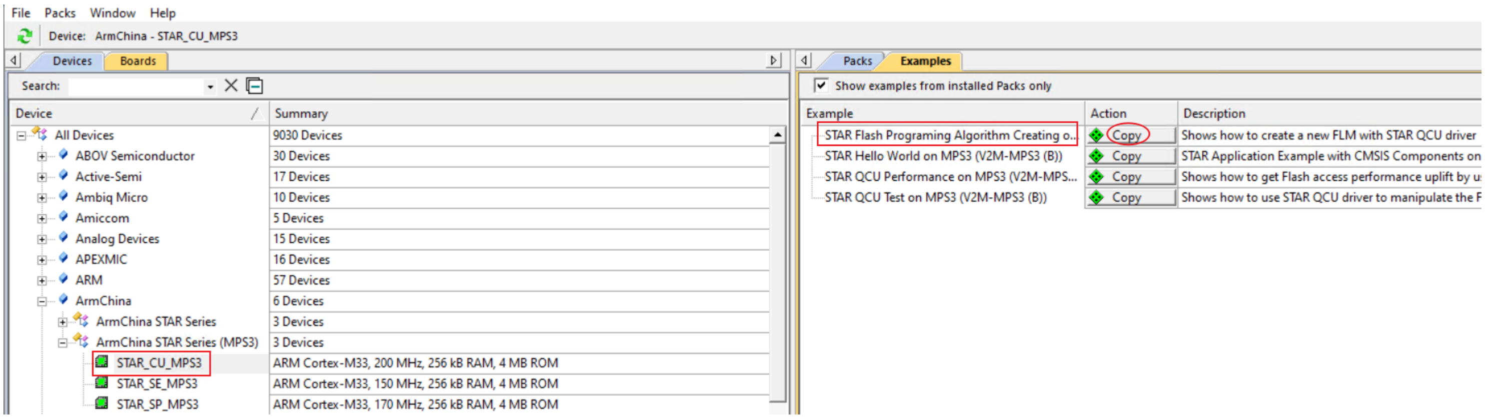

Pack Installer - C:\Users\cpussw01\AppData\Local\Arm\Packs
File
Packs
Window
Help
Device: ArmChina - STAR_CU
Devices
Boards
Packs
Examples
Search:

Device	Summary
All Devices	8381 Devices

Pack	Action	Description
Device Specific	1 Pack	STAR_CU selected
ArmChina::STAR	Up to date	The Arm China STAR processor is the first proc

2. On the **Examples** tab, select the example that you want to use and click **Copy**.

   In STAR CMSIS DFP v2.0.0, there is an example named **STAR Flash Programming Algorithm Creating on MPS3** available. This example project invokes the Flash programming algorithm using on the STAR MPS3 board and downloads the application into the Flash memory.



Device	Summary
All Devices	9030 Devices
ABOV Semiconductor	30 Devices
Active-Semi	17 Devices
Ambiq Micro	10 Devices
Amiccom	5 Devices
Analog Devices	15 Devices
APEXMIC	16 Devices
ARM	57 Devices
ArmChina	6 Devices
ArmChina STAR Series	3 Devices
ArmChina STAR Series (MPS3)	3 Devices
STAR_CU_MPS3	ARM Cortex-M33, 200 MHz, 256 kB RAM, 4 MB ROM
STAR_SE_MPS3	ARM Cortex-M33, 150 MHz, 256 kB RAM, 4 MB ROM
STAR_SP_MPS3	ARM Cortex-M33, 170 MHz, 256 kB RAM, 4 MB ROM

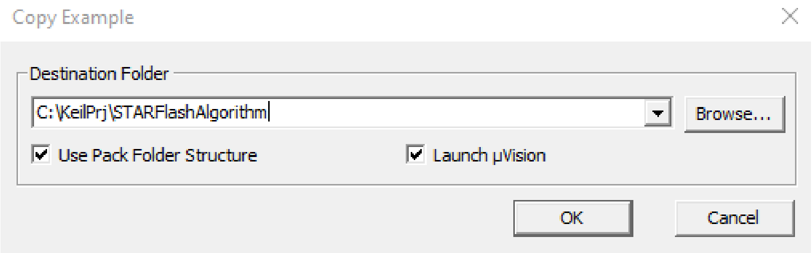
☑ Show examples from installed Packs only

Example	Action	Description
STAR Flash Programing Algorithm Creating o...	Copy	Shows how to create a new FLM with STAR QCU driver
STAR Hello World on MPS3 (V2M-MPS3 (B))	Copy	STAR Application Example with CMSIS Components on
STAR QCU Performance on MPS3 (V2M-MPS...	Copy	Shows how to get Flash access performance uplift by u
STAR QCU Test on MPS3 (V2M-MPS3 (B))	Copy	Shows how to use STAR QCU driver to manipulate the F

   Note:

   > Sometimes the Copy button is disabled in gray because there are some packs need to be updated. You can check the progress bar to confirm this situation. When the progress reaches 100%, the Copy button will be enabled.

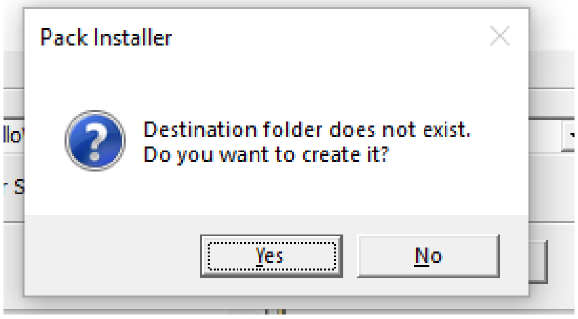
3. In the **Copy Example** dialog box that appears, specify the destination folder path to save the project, and then click **OK**.



C:\KeilPrj\STARFlashAlgorithm

☑ Use Pack Folder Structure ☑ Launch µVision

If the destination folder does not exist, click **Yes** to create it.

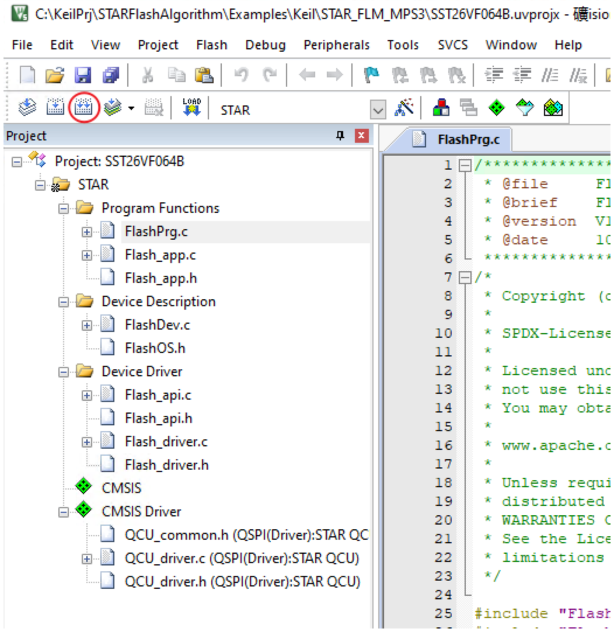


Destination folder does not exist.
Do you want to create it?

A project is created in the destination folder.

The µVision will start automatically and open the created project. In the Project pane, you can see all the required files.

4. Click the **Rebuild** icon to recompile and build the project.
   This operation will generate the Flash programming algorithm file SST26VF064B.FLM in the project folder.


C:\KeilPrj\STARFlashAlgorithm\Examples\Keil\STAR_FLM_MPS3\SST26VF064B.uvprojx - 礦isio
File
Edit
View
Project
Flash
Debug
Peripherals
Tools
SVCS
Window
Help
STAR
Project
FlashPrg.c

Project: SST26VF064B
- STAR
  - Program Functions
    - FlashPrg.c
    - Flash_app.c
    - Flash_app.h
  - Device Description
    - FlashDev.c
    - FlashOS.h
  - Device Driver
    - Flash_api.c
    - Flash_api.h
    - Flash_driver.c
    - Flash_driver.h
  - CMSIS
  - CMSIS Driver
    - QCU_common.h (QSPI(Driver):STAR QC
    - QCU_driver.c (QSPI(Driver):STAR QCU)
    - QCU_driver.h (QSPI(Driver):STAR QCU)

```
/**************
 * @file     Fl
 * @brief    Fl
 * @version  V1
 * @date     10
 **************
/*
 * Copyright (c
 *
 * SPDX-License
 *
 * Licensed unc
 * not use this
 * You may obta
 *
 * www.apache.c
 *
 * Unless requi
 * distributed
 * WARRANTIES C
 * See the Lice
 * limitations
 */

#include "Flash
```

5. Copy the .FLM file to $Keil_install_folder\ARM\Flash (optional for SST26VF064B.FLM which is embedded in STAR DFP v2.0.0).

   Now the Flash programming algorithm for the new Flash device has been generated.

You can use this algorithm to download and debug the application software from a Flash device.

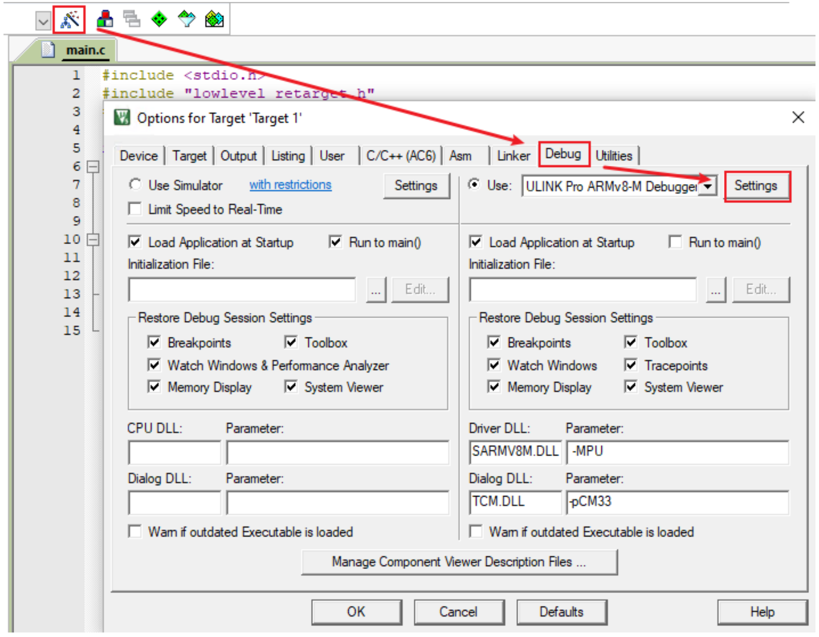
5.2 Using the new algorithm

To use the new algorithm, perform the following steps:

1. On the **Examples** tab, select the **STAR Hello World on MPS3** project as the example and rebuild the project.

2. Configure debug options in Keil MDK for your application.

   a. In the **Options for Target** dialog box, click the **Debug** tab, and then click **Settings**.



```
#include <stdio.h>
#include "lowlevel_retarget.h"
```

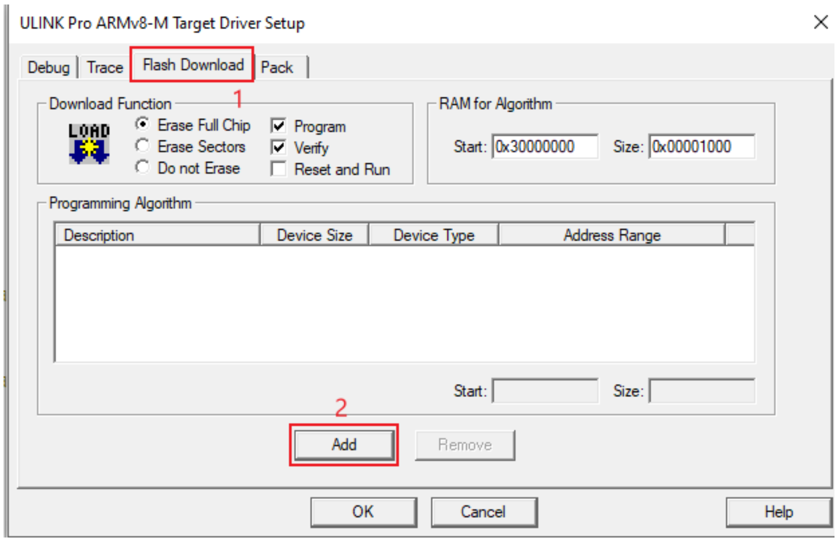
Options for Target 'Target 1'

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

○ Use Simulator with restrictions Settings
□ Limit Speed to Real-Time
☑ Load Application at Startup ☑ Run to main()
Initialization File:
Restore Debug Session Settings
☑ Breakpoints ☑ Toolbox
☑ Watch Windows & Performance Analyzer
☑ Memory Display ☑ System Viewer
CPU DLL: Parameter:
Dialog DLL: Parameter:
□ Warn if outdated Executable is loaded

● Use: ULINK Pro ARMv8-M Debugger Settings
☑ Load Application at Startup □ Run to main()
Initialization File:
Restore Debug Session Settings
☑ Breakpoints ☑ Toolbox
☑ Watch Windows ☑ Tracepoints
☑ Memory Display ☑ System Viewer
Driver DLL: SARMV8M.DLL Parameter: -MPU
Dialog DLL: TCM.DLL Parameter: -pCM33
□ Warn if outdated Executable is loaded

   b. On the **Flash Download** tab, select the download functions, and then click **Add** to select and add the Flash programming algorithm.



ULINK Pro ARMv8-M Target Driver Setup

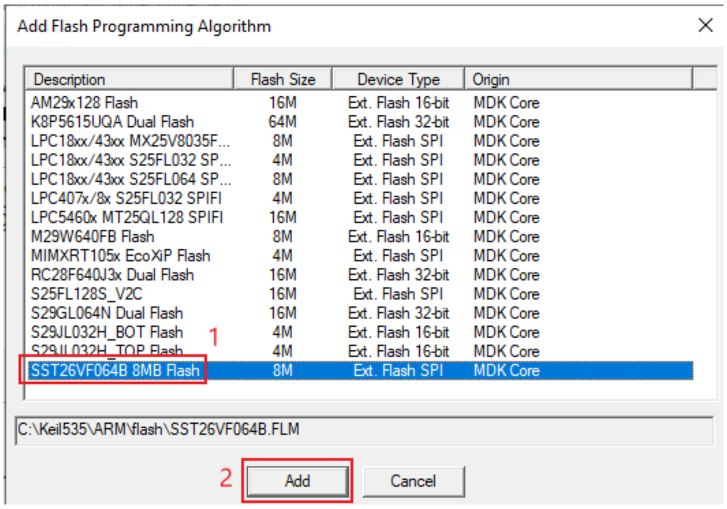
Download Function
● Erase Full Chip ☑ Program
○ Erase Sectors ☑ Verify
○ Do not Erase □ Reset and Run

RAM for Algorithm
Start: 0x30000000 Size: 0x00001000

Programming Algorithm

Description	Device Size	Device Type	Address Range

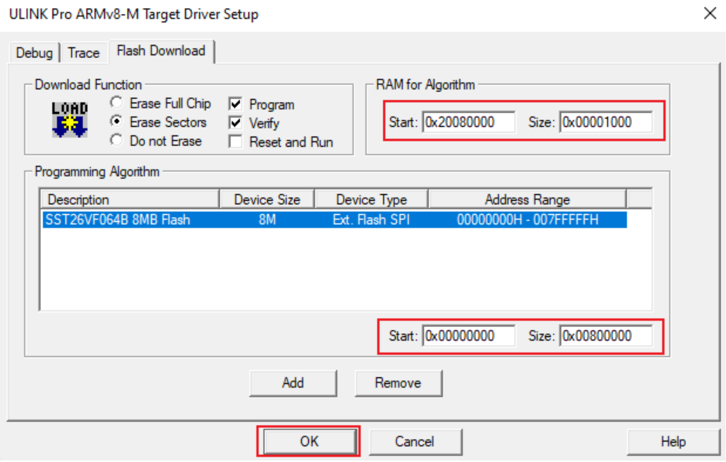
Start: Size:


Add Flash Programming Algorithm

Description	Flash Size	Device Type	Origin
AM29x128 Flash	16M	Ext. Flash 16-bit	MDK Core
K8P5615UQA Dual Flash	64M	Ext. Flash 32-bit	MDK Core
LPC18xx/43xx MX25V8035F...	8M	Ext. Flash SPI	MDK Core
LPC18xx/43xx S25FL032 SP...	4M	Ext. Flash SPI	MDK Core
LPC18xx/43xx S25FL064 SP...	8M	Ext. Flash SPI	MDK Core
LPC407x/8x S25FL032 SPIFI	4M	Ext. Flash SPI	MDK Core
LPC5460x MT25QL128 SPIFI	16M	Ext. Flash SPI	MDK Core
M29W640FB Flash	8M	Ext. Flash 16-bit	MDK Core
MIMXRT105x EcoXiP Flash	4M	Ext. Flash SPI	MDK Core
RC28F640J3x Dual Flash	16M	Ext. Flash 32-bit	MDK Core
S25FL128S_V2C	16M	Ext. Flash SPI	MDK Core
S29GL064N Dual Flash	16M	Ext. Flash 32-bit	MDK Core
S29JL032H_BOT Flash	4M	Ext. Flash 16-bit	MDK Core
S29JL032H_TOP Flash	4M	Ext. Flash 16-bit	MDK Core
SST26VF064B 8MB Flash	8M	Ext. Flash SPI	MDK Core

C:\Keil535\ARM\flash\SST26VF064B.FLM

c. Configure the RAM for algorithm and programming address, and then click **OK**.


ULINK Pro ARMv8-M Target Driver Setup

Debug | Trace | Flash Download

Download Function

○ Erase Full Chip ☑ Program
● Erase Sectors ☑ Verify
○ Do not Erase ☐ Reset and Run

RAM for Algorithm

Start: 0x20080000 Size: 0x00001000

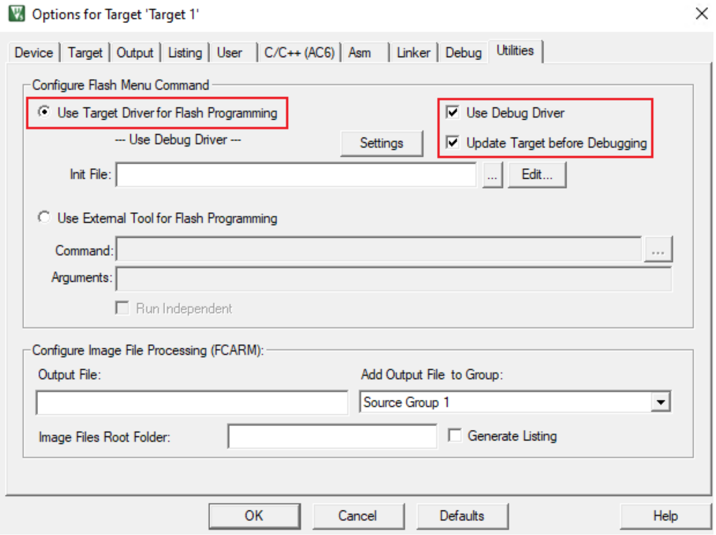
Programming Algorithm

Description	Device Size	Device Type	Address Range
SST26VF064B 8MB Flash	8M	Ext. Flash SPI	00000000H - 007FFFFFH

Start: 0x00000000 Size: 0x00800000

3. In the **Options for Target** dialog box, click the **Utilities** tab to configure the utilities options.

4. In the **Configure Flash Menu Command** area, select **Use Target Driver for Flash Programming**, **Use Debug Driver** and **Update Target before Debugging**, and then click **OK**.


Options for Target 'Target 1'

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

Configure Flash Menu Command

● Use Target Driver for Flash Programming ☑ Use Debug Driver

-- Use Debug Driver -- ☑ Update Target before Debugging

Init File:

○ Use External Tool for Flash Programming

Command:

Arguments:

☐ Run Independent

Configure Image File Processing (FCARM):

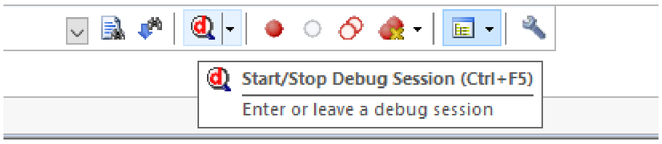
Output File:

Add Output File to Group: Source Group 1

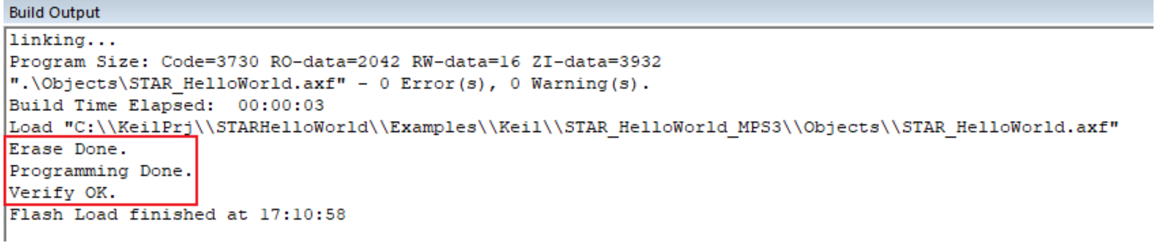
Image Files Root Folder:

☐ Generate Listing

5. Click **Start/Stop Debug Session** to start the debug session.


Start/Stop Debug Session (Ctrl+F5)
Enter or leave a debug session

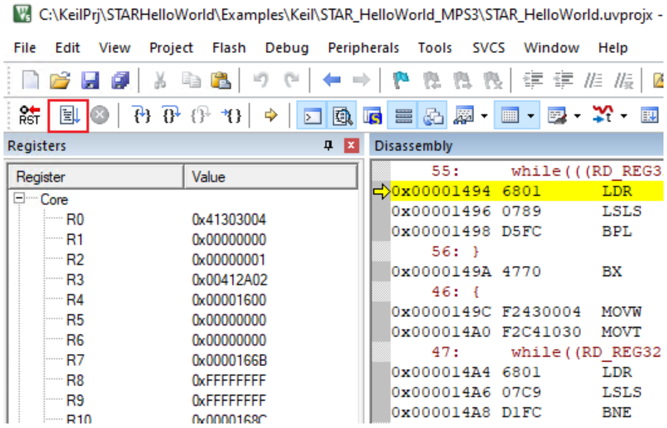
You can see the programming process in the Build Output pane.



Build Output

```
linking...
Program Size: Code=3730 RO-data=2042 RW-data=16 ZI-data=3932
".\Objects\STAR_HelloWorld.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:03
Load "C:\\KeilPrj\\STARHelloWorld\\Examples\\Keil\\STAR_HelloWorld_MPS3\\Objects\\STAR_HelloWorld.axf"
Erase Done.
Programming Done.
Verify OK.
Flash Load finished at 17:10:58
```

6. Click the Run icon to run the built software.

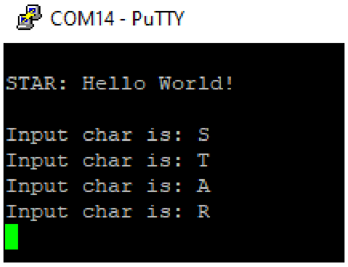

C:\KeilPrj\STARHelloWorld\Examples\Keil\STAR_HelloWorld_MPS3\STAR_HelloWorld.uvprojx -
File
Edit
View
Project
Flash
Debug
Peripherals
Tools
SVCS
Window
Help
RST
Registers
Disassembly

Register	Value
Core	
R0	0x41303004
R1	0x00000000
R2	0x00000001
R3	0x00412A02
R4	0x00001600
R5	0x00000000
R6	0x00000000
R7	0x0000166B
R8	0xFFFFFFFF
R9	0xFFFFFFFF
R10	0x0000168C

```
    55:      while(((RD_REG3
0x00001494 6801      LDR
0x00001496 0789      LSLS
0x00001498 D5FC      BPL
    56: }
0x0000149A 4770      BX
    46: {
0x0000149C F2430004  MOVW
0x000014A0 F2C41030  MOVT
    47:      while((RD_REG32
0x000014A4 6801      LDR
0x000014A6 07C9      LSLS
0x000014A8 D1FC      BNE
```

You can see the log message of the example project in the UART terminal window.

The window can also receive the char you input.



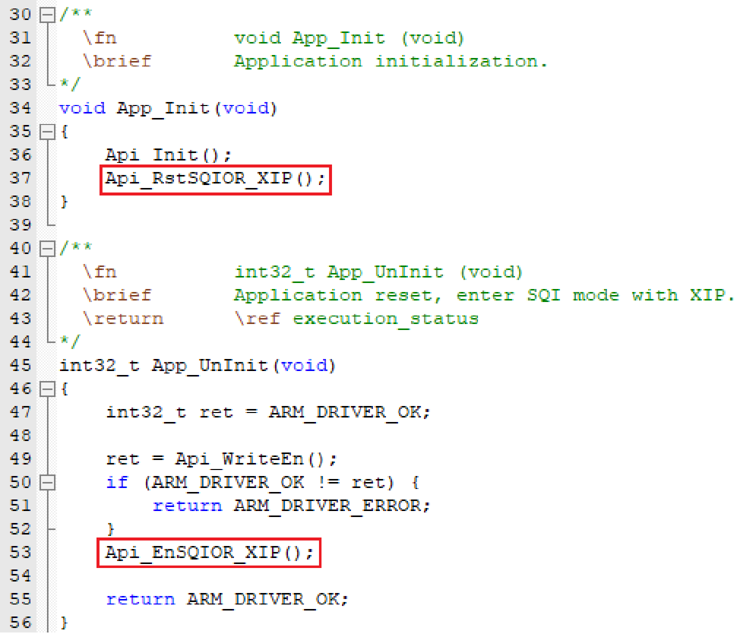
COM14 - PuTTY

```
STAR: Hello World!

Input char is: S
Input char is: T
Input char is: A
Input char is: R
```

Based on this project, you can start to use the QCU driver and try more Flash related operations in your STAR-based application software.

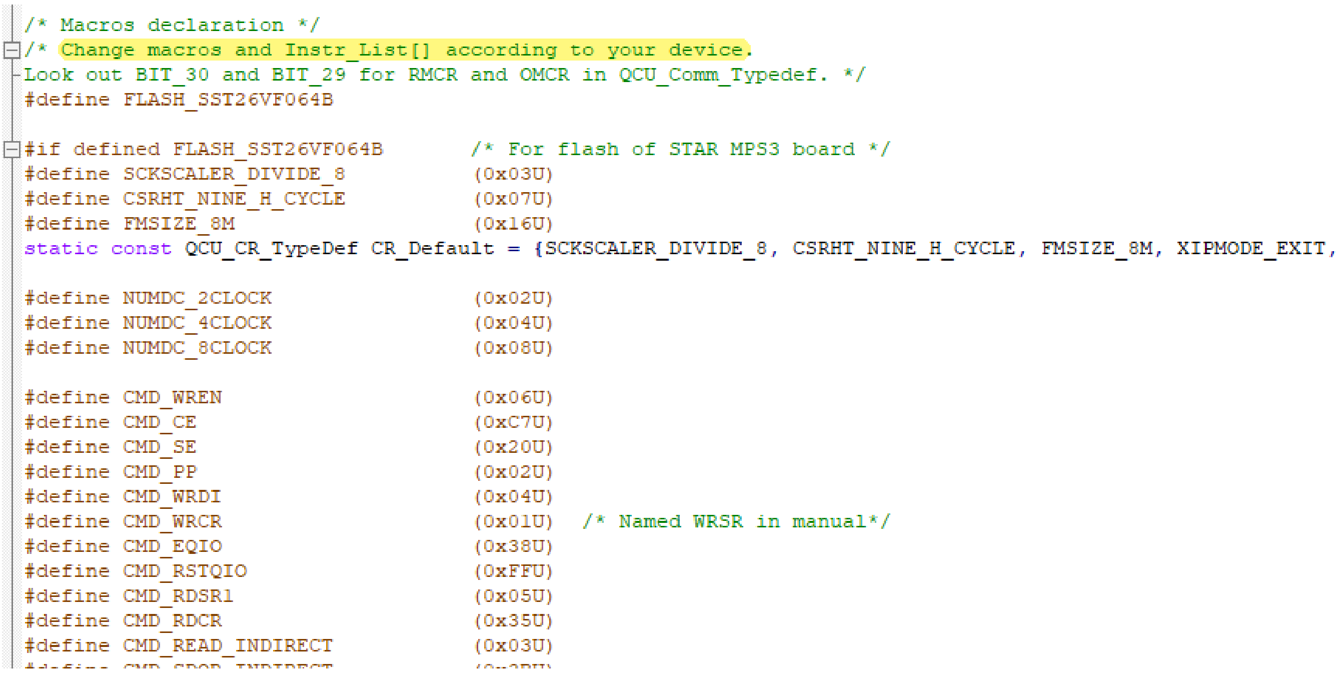
Notes:

- You need to replace the implementation of mandatory functions in FlashPrg.c and device parameters in FlashDev.c when changing to another Flash device. Be careful of the method to activate and terminate Flash XIP mode in App_Init() and App_UnInit() functions.

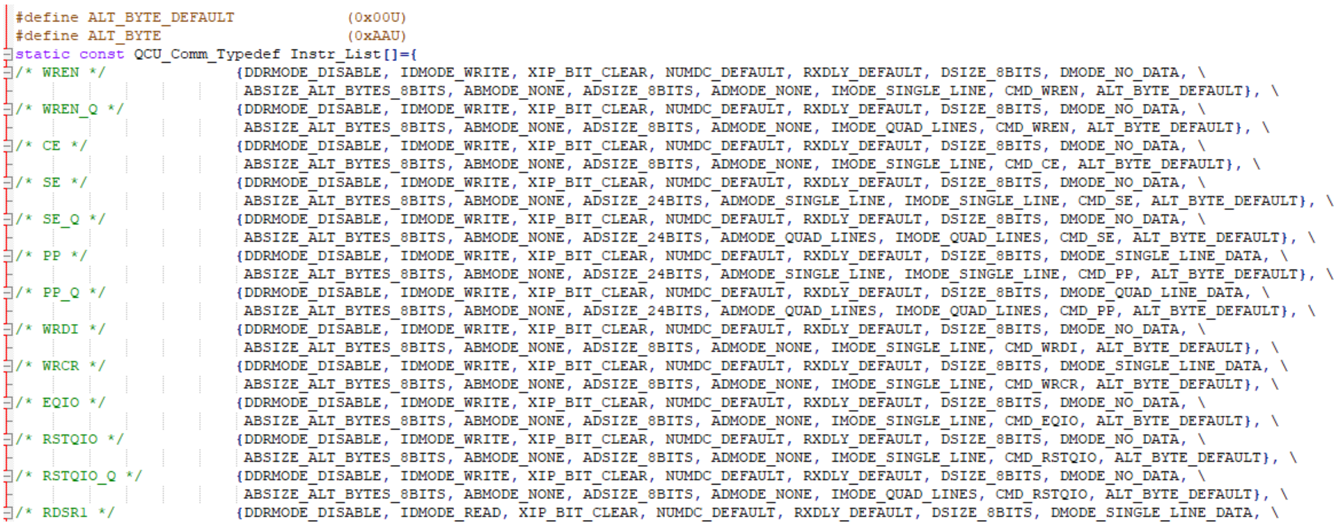


```
/**
  \fn         void App_Init (void)
  \brief      Application initialization.
*/
void App_Init(void)
{
    Api_Init();
    Api_RstSQIOR_XIP();
}

/**
  \fn         int32_t App_UnInit (void)
  \brief      Application reset, enter SQI mode with XIP.
  \return     \ref execution_status
*/
int32_t App_UnInit(void)
{
    int32_t ret = ARM_DRIVER_OK;

    ret = Api_WriteEn();
    if (ARM_DRIVER_OK != ret) {
        return ARM_DRIVER_ERROR;
    }
    Api_EnSQIOR_XIP();

    return ARM_DRIVER_OK;
}
```

- You also need to modify the instruction list in Flash driver files (Flash_driver.c and Flash_driver.h) according to the Flash device specification.

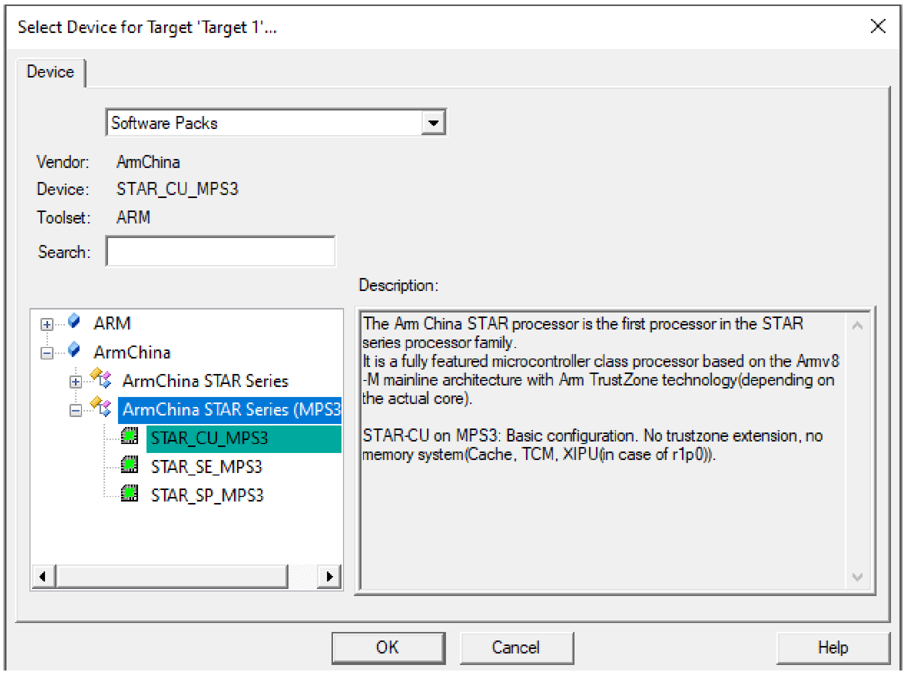


```
/* Macros declaration */
/* Change macros and Instr_List[] according to your device.
Look out BIT_30 and BIT_29 for RMCR and OMCR in QCU_Comm_Typedef. */
#define FLASH_SST26VF064B

#if defined FLASH_SST26VF064B       /* For flash of STAR MPS3 board */
#define SCKSCALER_DIVIDE_8          (0x03U)
#define CSRHT_NINE_H_CYCLE          (0x07U)
#define FMSIZE_8M                   (0x16U)
static const QCU_CR_TypeDef CR_Default = {SCKSCALER_DIVIDE_8, CSRHT_NINE_H_CYCLE, FMSIZE_8M, XIPMODE_EXIT,

#define NUMDC_2CLOCK                (0x02U)
#define NUMDC_4CLOCK                (0x04U)
#define NUMDC_8CLOCK                (0x08U)

#define CMD_WREN                    (0x06U)
#define CMD_CE                      (0xC7U)
#define CMD_SE                      (0x20U)
#define CMD_PP                      (0x02U)
#define CMD_WRDI                    (0x04U)
#define CMD_WRCR                    (0x01U)   /* Named WRSR in manual*/
#define CMD_EQIO                    (0x38U)
#define CMD_RSTQIO                  (0xFFU)
#define CMD_RDSR1                   (0x05U)
#define CMD_RDCR                    (0x35U)
#define CMD_READ_INDIRECT           (0x03U)
```



```
#define ALT_BYTE_DEFAULT              (0x00U)
#define ALT_BYTE                      (0xAAU)
static const QCU_Comm_Typedef Instr_List[]={
/* WREN */                 {DDRMODE_DISABLE, IDMODE_WRITE, XIP_BIT_CLEAR, NUMDC_DEFAULT, RXDLY_DEFAULT, DSIZE_8BITS, DMODE_NO_DATA, \
                            ABSIZE_ALT_BYTES_8BITS, ABMODE_NONE, ADSIZE_8BITS, ADMODE_NONE, IMODE_SINGLE_LINE, CMD_WREN, ALT_BYTE_DEFAULT}, \
/* WREN_Q */               {DDRMODE_DISABLE, IDMODE_WRITE, XIP_BIT_CLEAR, NUMDC_DEFAULT, RXDLY_DEFAULT, DSIZE_8BITS, DMODE_NO_DATA, \
                            ABSIZE_ALT_BYTES_8BITS, ABMODE_NONE, ADSIZE_8BITS, ADMODE_NONE, IMODE_QUAD_LINES, CMD_WREN, ALT_BYTE_DEFAULT}, \
/* CE */                   {DDRMODE_DISABLE, IDMODE_WRITE, XIP_BIT_CLEAR, NUMDC_DEFAULT, RXDLY_DEFAULT, DSIZE_8BITS, DMODE_NO_DATA, \
                            ABSIZE_ALT_BYTES_8BITS, ABMODE_NONE, ADSIZE_8BITS, ADMODE_NONE, IMODE_SINGLE_LINE, CMD_CE, ALT_BYTE_DEFAULT}, \
/* SE */                   {DDRMODE_DISABLE, IDMODE_WRITE, XIP_BIT_CLEAR, NUMDC_DEFAULT, RXDLY_DEFAULT, DSIZE_8BITS, DMODE_NO_DATA, \
                            ABSIZE_ALT_BYTES_8BITS, ABMODE_NONE, ADSIZE_24BITS, ADMODE_SINGLE_LINE, IMODE_SINGLE_LINE, CMD_SE, ALT_BYTE_DEFAULT}, \
/* SE_Q */                 {DDRMODE_DISABLE, IDMODE_WRITE, XIP_BIT_CLEAR, NUMDC_DEFAULT, RXDLY_DEFAULT, DSIZE_8BITS, DMODE_NO_DATA, \
                            ABSIZE_ALT_BYTES_8BITS, ABMODE_NONE, ADSIZE_24BITS, ADMODE_QUAD_LINES, IMODE_QUAD_LINES, CMD_SE, ALT_BYTE_DEFAULT}, \
/* PP */                   {DDRMODE_DISABLE, IDMODE_WRITE, XIP_BIT_CLEAR, NUMDC_DEFAULT, RXDLY_DEFAULT, DSIZE_8BITS, DMODE_SINGLE_LINE_DATA, \
                            ABSIZE_ALT_BYTES_8BITS, ABMODE_NONE, ADSIZE_24BITS, ADMODE_SINGLE_LINE, IMODE_SINGLE_LINE, CMD_PP, ALT_BYTE_DEFAULT}, \
/* PP_Q */                 {DDRMODE_DISABLE, IDMODE_WRITE, XIP_BIT_CLEAR, NUMDC_DEFAULT, RXDLY_DEFAULT, DSIZE_8BITS, DMODE_QUAD_LINE_DATA, \
                            ABSIZE_ALT_BYTES_8BITS, ABMODE_NONE, ADSIZE_24BITS, ADMODE_QUAD_LINES, IMODE_QUAD_LINES, CMD_PP, ALT_BYTE_DEFAULT}, \
/* WRDI */                 {DDRMODE_DISABLE, IDMODE_WRITE, XIP_BIT_CLEAR, NUMDC_DEFAULT, RXDLY_DEFAULT, DSIZE_8BITS, DMODE_NO_DATA, \
                            ABSIZE_ALT_BYTES_8BITS, ABMODE_NONE, ADSIZE_8BITS, ADMODE_NONE, IMODE_SINGLE_LINE, CMD_WRDI, ALT_BYTE_DEFAULT}, \
/* WRCR */                 {DDRMODE_DISABLE, IDMODE_WRITE, XIP_BIT_CLEAR, NUMDC_DEFAULT, RXDLY_DEFAULT, DSIZE_8BITS, DMODE_SINGLE_LINE_DATA, \
                            ABSIZE_ALT_BYTES_8BITS, ABMODE_NONE, ADSIZE_8BITS, ADMODE_NONE, IMODE_SINGLE_LINE, CMD_WRCR, ALT_BYTE_DEFAULT}, \
/* EQIO */                 {DDRMODE_DISABLE, IDMODE_WRITE, XIP_BIT_CLEAR, NUMDC_DEFAULT, RXDLY_DEFAULT, DSIZE_8BITS, DMODE_NO_DATA, \
                            ABSIZE_ALT_BYTES_8BITS, ABMODE_NONE, ADSIZE_8BITS, ADMODE_NONE, IMODE_SINGLE_LINE, CMD_EQIO, ALT_BYTE_DEFAULT}, \
/* RSTQIO */               {DDRMODE_DISABLE, IDMODE_WRITE, XIP_BIT_CLEAR, NUMDC_DEFAULT, RXDLY_DEFAULT, DSIZE_8BITS, DMODE_NO_DATA, \
                            ABSIZE_ALT_BYTES_8BITS, ABMODE_NONE, ADSIZE_8BITS, ADMODE_NONE, IMODE_SINGLE_LINE, CMD_RSTQIO, ALT_BYTE_DEFAULT}, \
/* RSTQIO_Q */             {DDRMODE_DISABLE, IDMODE_WRITE, XIP_BIT_CLEAR, NUMDC_DEFAULT, RXDLY_DEFAULT, DSIZE_8BITS, DMODE_NO_DATA, \
                            ABSIZE_ALT_BYTES_8BITS, ABMODE_NONE, ADSIZE_8BITS, ADMODE_NONE, IMODE_QUAD_LINES, CMD_RSTQIO, ALT_BYTE_DEFAULT}, \
/* RDSR1 */                {DDRMODE_DISABLE, IDMODE_READ, XIP_BIT_CLEAR, NUMDC_DEFAULT, RXDLY_DEFAULT, DSIZE_8BITS, DMODE_SINGLE_LINE_DATA, \
```

- From STAR CMSIS v2.0.0 onwards, there is a Flash algorithm by default when you create a new MDK project based on ArmChina STAR Series (MPS3).



Select Device for Target 'Target 1'...

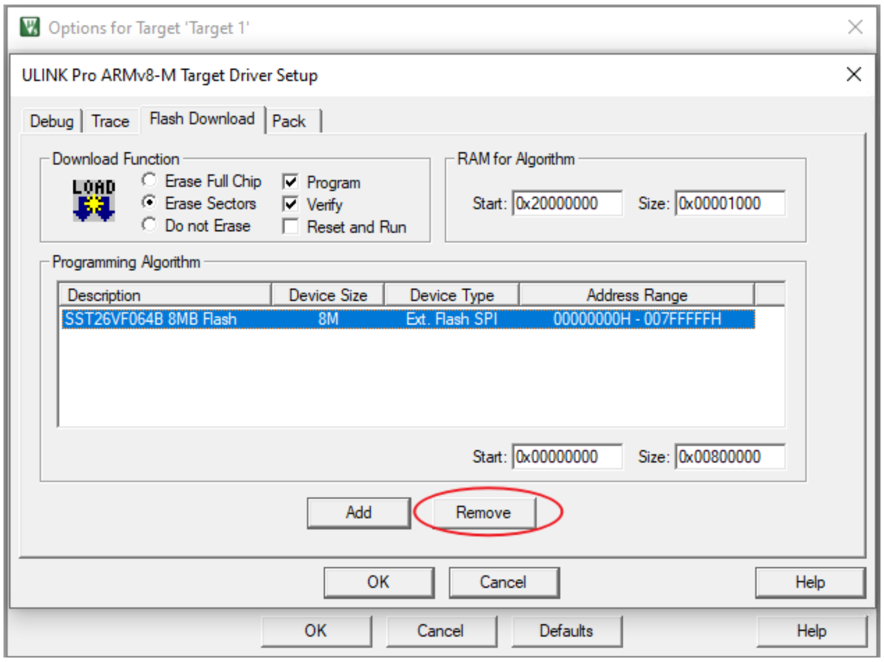
Software Packs

Vendor: ArmChina
Device: STAR_CU_MPS3
Toolset: ARM
Search:

Description:

The Arm China STAR processor is the first processor in the STAR series processor family.
It is a fully featured microcontroller class processor based on the Armv8-M mainline architecture with Arm TrustZone technology(depending on the actual core).

STAR-CU on MPS3: Basic configuration. No trustzone extension, no memory system(Cache, TCM, XIPU(in case of r1p0)).



Download Function

○ Erase Full Chip
● Erase Sectors
○ Do not Erase

☑ Program
☑ Verify
□ Reset and Run

RAM for Algorithm

Start: 0x20000000 Size: 0x00001000

Programming Algorithm

Description	Device Size	Device Type	Address Range
SST26VF064B 8MB Flash	8M	Ext. Flash SPI	00000000H - 007FFFFFH

Start: 0x00000000 Size: 0x00800000

If this programming algorithm is not the one you want to use or you are not going to use Flash memory, remove the default programming algorithm *SST26VF064B 8M Flash*.