

Calculating Post-Build CRC in Arm® Keil®

Using SRecord for Post-Build CRC in Arm Keil μVision

1. Introduction

It is necessary to ensure the code/data integrity of safety-related applications. The Cyclic Redundancy Check (CRC) error-detecting code can be used to detect accidental changes to the raw data (source code or data change). The CRC code is calculated after the source-code compiling and linking and the CRC result is stored in the memory.

A problem with calculating the post-build CRC code of the flash content and merging the CRC result with the flash content into one resulting file occurred during the development of the NXP IEC60730B safety library.

The process of the post-build CRC calculation in the μVision-KEIL IDE using the external SRecord tool is comprehensively described in this application note.

Contents

1.	Introduction	1
2.	Necessary software.....	2
2.1.	Getting SRecord	2
2.2.	Copying SRecord file to project structure.....	2
3.	Using post-build steps in uVision	2
3.1.	Settings output file	3
3.2.	Setting post-build action	3
3.3.	Using edited *.hex file for downloading and debugging	4
4.	Creating debug.ini file.....	6
5.	Creating SRecord *.bat file for CRC	6
5.1.	Explanation of individual commands	6
6.	Conclusion.....	7
7.	Revision history.....	7

2. Necessary software

The following software tools are necessary to calculate the post-build CRC:

- Keil μ Vision 5
- SRecord 1.64
- Any text editor (NotePad, Pspad, or other)

2.1. Getting SRecord

SRecord is a standalone utility for memory manipulation. This utility and all information about it are available at Peter Miller's webpage <http://srecord.sourceforge.net/>.

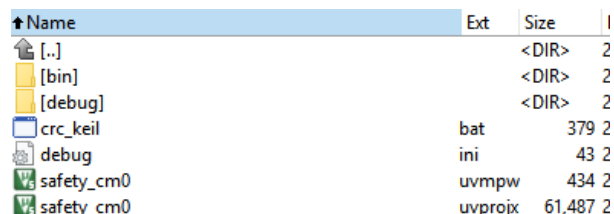
In the download page in the "SourceForge Downloads" section, select the "1.64 release .zip file" link. Click on it and download the SRecord 1.64 package. The downloaded ZIP folder contains these items:

- *srec_cat.exe*
- *srec_cmp.exe*
- *srec_info.exe*
- Reference manual *srecord-1.63.pdf*
- Readme files

The SRecord file is used to calculate the post-build CRC without any changes.

2.2. Copying SRecord file to project structure

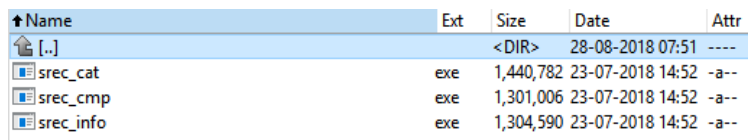
Copy the *.exe files to a project folder structure. Create a folder called *bin*, as shown in Figure 1.



Name	Ext	Size	I
[..]		<DIR>	2
[bin]		<DIR>	2
[debug]		<DIR>	2
crc_keil	bat	379	2
debug	ini	43	2
safety_cm0	uvmpw	434	2
safety_cm0	uvprojx	61,487	2

Figure 1. Project folder content

Copy all *.exe files from the *SRecord* ZIP folder into the *bin* folder:



Name	Ext	Size	Date	Attr
[..]		<DIR>	28-08-2018 07:51	----
srec_cat	exe	1,440,782	23-07-2018 14:52	-a--
srec_cmp	exe	1,301,006	23-07-2018 14:52	-a--
srec_info	exe	1,304,590	23-07-2018 14:52	-a--

Figure 2. SRecat bin folder content

3. Using post-build steps in uVision

To use the post build, configure the μ Vision-Keil to generate the *.hex output file, call the post-build action, and use the modified *.hex file for downloading/debugging.

3.1. Settings output file

To use SRecord, it is necessary to have an output file which can be modified by the SRecord. In μ Vision, a *.hex file is available.

To set this, open these settings in your μ Vision project:

- Project -> Options for Target -> OUTPUT -> Enable „Create HEX File“

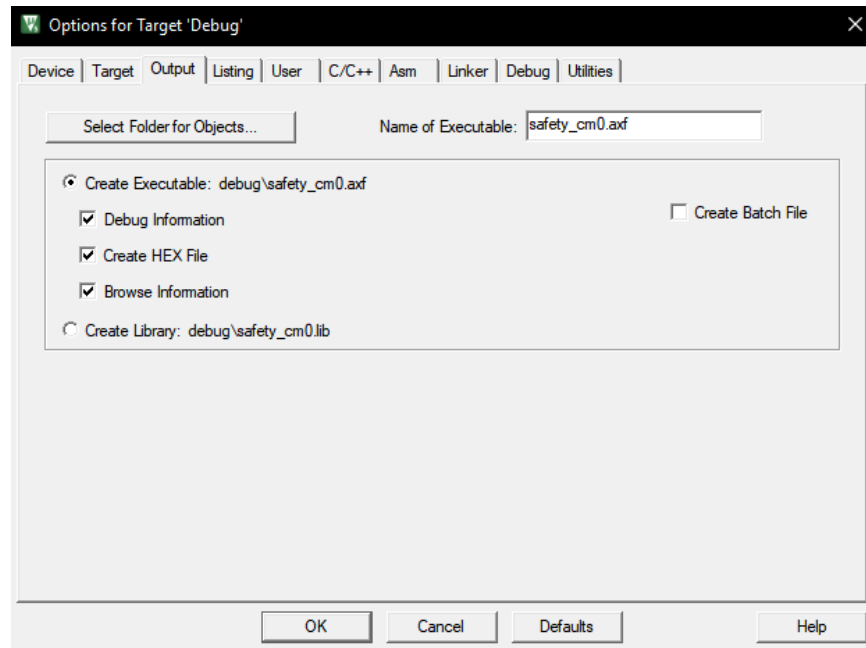


Figure 3. Enabling *.hex file generation

In the setting card shown above, fill the “Name of Executable” field. This name is used for the generated *.hex file.

3.2. Setting post-build action

The μ Vision-Keil can make three types of build actions:

- “Before Compile C/C++ File”
- “Before Build/Rebuild”
- “After Build/Rebuild”

For the CRC calculation, it is necessary to select the “After Build/Rebuild” option. It is possible to write the SRecord command directly to the field. For better clarity, the approach using a *.bat file is presented.

To use this file, write “.\crc_keil.bat” into the field. This presents the relative path from the *safety_cm0.uvprojx* project file to the *crc_keil.bat* file, as shown in Figure 1 and Figure 2. The *crc_keil.bat* file is in the same folder as the *safety_cm0.uvprojx* file). Enable the post-build step using the check box (see Figure 4).

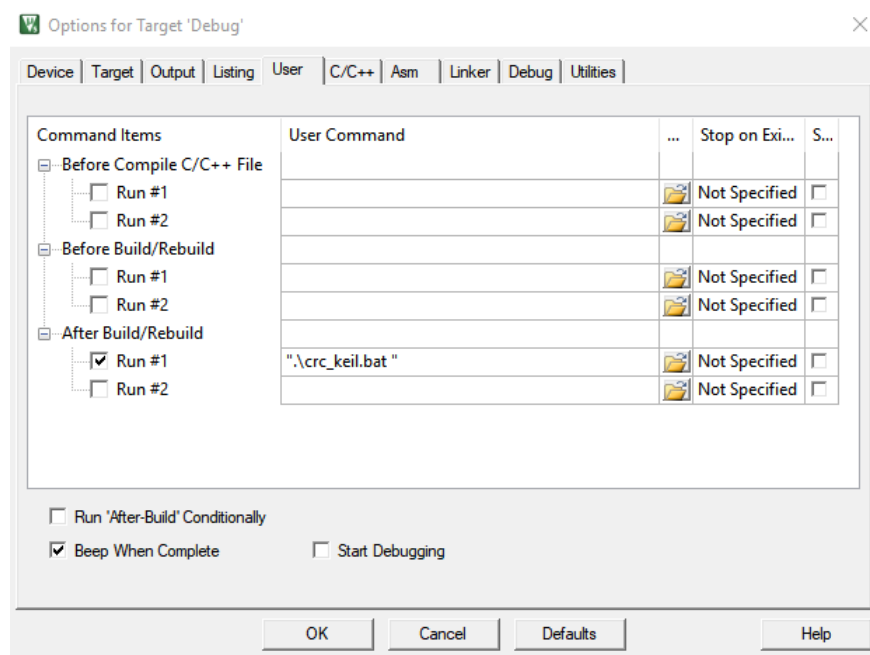


Figure 4. Post-build action command enable

3.3. Using edited *.hex file for downloading and debugging

After the post-build steps, the edited *.hex file with a CRC value is used for debugging. This is provided by the *.ini file. The creation of this *.ini file is described in [Section 4, “Creating debug.ini file”](#).

Follow these steps to use the *.ini file:

1. In your μ Vision project, open these settings:
“Project -> Options for Target -> Debug”
2. Put the **.\debug.ini** command into the “Initialization File” field (“.” means the relative path from the *safety_cm0.uvprojx* project file to the *debug.ini* file, as shown in [Figure 1](#) and [Figure 2](#)). The correct settings are shown in [Figure 5](#).

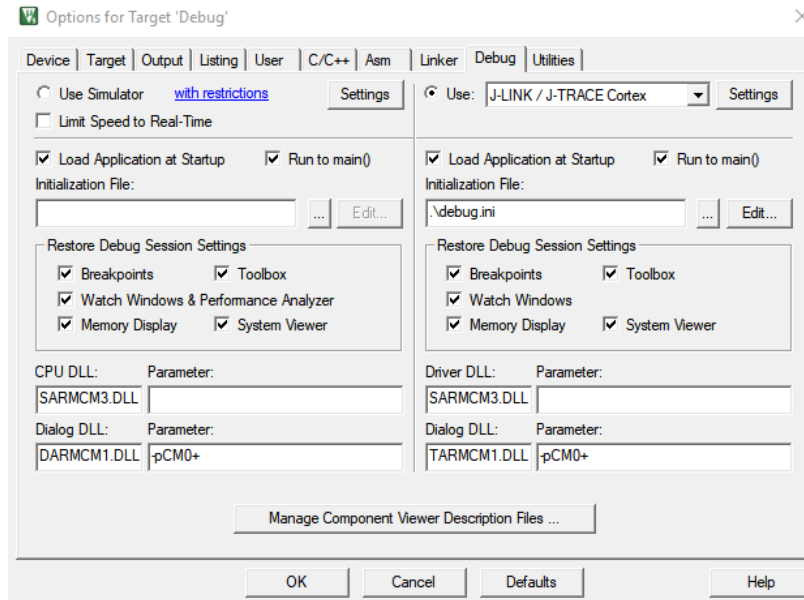


Figure 5. Setting of debug.ini file

3. Open the “Utilities” option, as follows:
“Project -> Options for Target -> Utilities”
4. Fill the “Init File” field with the **.\debug.ini** command, as shown in the previous step.

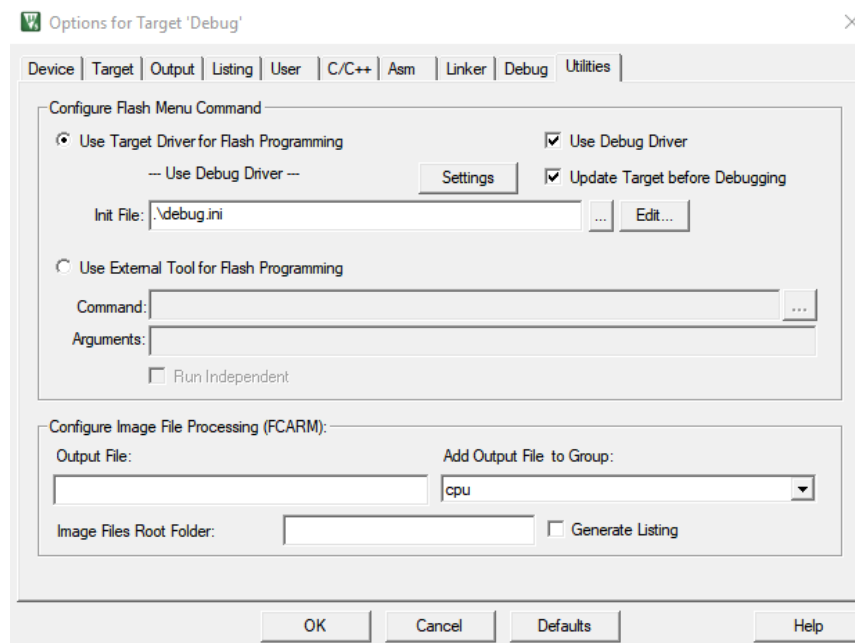


Figure 6. Setting of "Utilities" cart

4. Creating debug.ini file

Use the *debug.ini* file when using an output file different from the default one. In the presented case, a *.hex file with an added CRC is used.

Use a text editor to create the *debug.ini* file. Create an empty file, save it with the *.ini extension, and copy this command into the file: **LOAD .\debug\safety_cm0_crc.hex INCREMENTAL**.

This command loads the *safety_cm0_crc.hex* file from the .\debug\ relative path and this address is relative to the project file (*safety_cm0.uvprojx* in the presented case). It means that the file is in the *debug* folder, as shown in [Figure 1](#) and [Figure 2](#).

5. Creating SRecord *.bat file for CRC

Create an empty file in a text editor and save the file with the *.bat extension. The name of the file must be the same as the name in [Section 3.2, “Setting post-build action”](#) (*crc_keil.bat* in the presented case).

Copy these commands into the new *.bat file:

- **.\bin\srec_cat .\debug\safety_cm0.hex -intel -crop 0x410 0x4DFE -fill 0xFF 0x410 0x4DFE -CRC16_Little_Endian 0x4DFE -o .\debug\safety_cm0_temp.hex -intel**
- **.\bin\srec_cat .\debug\safety_cm0.hex -intel -exclude 0x410 0x4E00 -o .\debug\safety_cm0_ex.hex -intel**
- **.\bin\srec_cat .\debug\safety_cm0_ex.hex -intel .\debug\safety_cm0_temp.hex -intel -o .\debug\safety_cm0_crc.hex -intel**

5.1. Explanation of individual commands

- **.\bin\srec_cat**—this is a path to the *srec_cat.exe* file in the *bin* folder.
- **.\debug\safety_cm0.hex**—this selects the *safety_cm0.hex* output file in the *debug* folder.
- **-crop 0x410 0x4DFE**—crops the area from the *safety_cm0.hex* file in the address range from 0x410 to 0x4DFE (the rest is discarded).
- **-fill 0xFF 0x410 0x4DFE**—fills the unused space in the address range by 0xFF.
- **-CRC16_Little_Endian 0x4DFE -o .\debug\safety_cm0_temp.hex -intel**—calculates CRC16 on the cropped address area and saves the result to the 0x4DFE address in the new *safety_cm0_temp.hex* file. The *safety_cm0_temp.hex* file now contains the address range from 0x410 to 0x4E00.
- **.\bin\srec_cat .\debug\safety_cm0.hex -intel -exclude 0x410 0x4E00 -o .\debug\safety_cm0_ex.hex -intel**—this statement excludes the address range from the *safety_cm0.h* file. As a result, there is an empty gap from address 0x410 to address 0x4E00 and it is stored to the *safety_cm0_ex.hex* file.
- **.\bin\srec_cat .\debug\safety_cm0_ex.hex -intel .\debug\safety_cm0_temp.hex -intel -o .\debug\safety_cm0_crc.hex -intel**—this command merges the *safety_cm0_ex.hex* file with the *safety_cm0_temp.hex* file and saves the result to the new *safety_cm0_crc.hex* file.

6. Conclusion

This application note describes how to calculate the CRC result over a specified area of flash memory. For more details about the SRecord tool, see the reference manual. The reference manual is included in the downloaded package or available for download at <http://srecord.sourceforge.net/>.

The SRecord tool can also be used in the MCUXpresso toolchain. The IAR IDE can generate the post-build CRC directly using the IAR linker setting without using any external tools.

7. Revision history

Table summarizes the changes done to this document since the initial release.

Table 1. Revision history

Revision number	Date	Substantive changes
0	06/2019	Initial release.
1	07/2019	Changed the document title.

How to Reach Us:

Home Page:

www.nxp.com

Web Support:

www.nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address:

www.nxp.com/SalesTermsandConditions.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer's applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, Altivec, C 5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. Arm, AMBA, Arm Powered, Artisan, Cortex, Jazelle, Keil, SecurCore, Thumb, TrustZone, and μ Vision are registered trademarks of Arm Limited (or its subsidiaries) in the EU and/or elsewhere. Arm7, Arm9, Arm11, big.LITTLE, CoreLink, CoreSight, DesignStart, Mali, Mbed, NEON, POP, Sensinode, Socrates, ULINK and Versatile are trademarks of Arm Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2019 NXP B.V.

Document Number: AN12520
Rev. 1
07/2019

