

# **Application Note AN106**

# Using EcoXiP on the NXP i.MX RT1050 EVKB Board

www.adestotech.com

© Adesto 2019 • 3600 Peterson Way • Santa Clara CA. 95054 • Phone +1 408 400 0578

This publication contains proprietary information which is subject to change without notice and is supplied "as is", without any warranty of any kind.

# **Revision History**

Revison	Date	Tasks
Number		
A0	September 06, 2018	i.MX RT1050 EcoXiP Flash Memory Device Application Note initial release
A1	June 18, 2019	i.MX RT1050 EcoXiP Flash Memory Device Application Note reformatting



# Table of Contents

1. Introduction	4
2. Hardware Modifications	5
3. Boot / XiP Settings	5
4. Loading Code into the EcoXiP Flash	6
4.1 IAR Embedded Workbench	7
4.2 Keil MDK	9
4.3 MCUXpresso IDE	11
5. Boot / Run	13
Appendix A. Rework Instructions	14
A.1 Replace Flash Device	14
A.1.1 Enable Extension of Reset Hold Time Using Switch SW5	14
A.1.2 Jumper Pins at Location SW5	15
A.1.3 Add a DIP Switch at Location SW5	15
A.2 JEDEC Reset and eFUSE Programming	16
A.2.1 JEDEC Reset	16
A.2.2 Setting the eFuse	16



#### 1. Introduction

This document discusses the hardware and software requirements for configuring the NXP i.MX RT1050 EVKB board with an Adesto EcoXiP Flash device. An overview of hardware modifications is provided in Section 2, along with a more detailed description of the necessary hardware changes in Appendix A. From a software perspective, Sections 3 - 5 contain information on configuring the EcoXiP Flash, loading the code into the Flash using a Flash loader from various tool chains (three are provided), and executing the code. The Adesto EcoXiP Flash device conforms to the new JEDEC xSPI1 and JESD251 specifications and is the only Octal-SPI device designed specifically for XiP, providing better cost and power savings than its competitors. Figure 1 shows a photo of a modified i.MX RT1050 EVKB board with an Adesto EcoXiP Flash device.



Figure 1. NXP i.MX RT1050 EVKB Board with EcoXiP Flash

The NXP i.MX RT1050 crossover processor family has been designed without embedded Flash. Rather, it relies on external Flash devices such as Adesto's EcoXiP to store code and data. Its highly optimized Flash host controller (called FlexSPI) enables direct execution of code from external Flash. This architecture, known as execute-in-place or XiP, provides multiple benefits in terms of cost and power consumption and works well with advanced serial Flash devices such as EcoXiP which supports Octal-SPI interface, double data rate (DDR) and high clock speeds to support high-speed execution.

The i.MX RT1050 can be configured to boot directly from external Flash. Generating a Flash-bootable image requires users to include a couple of data blocks at the top of the image, one of which is product-dependent (includes parameters which vary from Flash product to Flash product). The i.MX RT1050 Software Development Kit (SDK) includes many example projects and now most example projects include one configuration or target in which the system boots and executes from external Flash.

Once a bootable image is built, it must be downloaded to the target and in the case of Flash memory be programmed into the Flash device. During development, programming can be done directly from the development tools environment, specifically from the debugger. Development tools often support Flash loader extensions (sometimes called Flash drivers or Flash algorithms) to enable that. Typically each Flash manufacturer requires its own extension or plug-in for programming. This is true for the Adesto EcoXiP Flash device.



This application note examines the process of building a Flash-bootable image for i.MX RT1050, programming the Flash, and running it. It specifically shows how to do all of this using the EcoXiP Flash. This application note refers to the NXP MCUExpresso SDK version 2.4.0.

#### 2. Hardware Modifications

This section provides an overview of the hardware modifications performed on the NXP evaluation board to facilitate its use with the Adesto EcoXiP Flash memory device. If you have already received a modified IMXRT1050-EVKB board from Adesto, this section is meant to provide a brief overview of the hardware modifications performed. If you received an original unmodified IMXRT1050-EVKB board and need to perform the required hardware modifications, or if you are just interested in understanding how the original board was modified, refer to Appendix A.

A modified version of the NXP evaluation board for the i.MX RT1050 uses an EcoXiP Flash. The following modifications were applied to an original IMXRT1050-EVKB board:

- Replace original Flash device in U19 with the Adesto EcoXiP Flash device.
- Populate a 4-position DIP switch in SW5 (originally unpopulated). Set switches 2 and 3 to the ON position to extend RT1050 boot ROM hold time (indicates the amount of time delay from a Flash reset to the first Flash access). As an alternative to populating the DIP switch, wires can be soldered to jumper pins 2 and 7 and pins 3 and 6 on SW5. This would accomplish the same amount of delay as described in Appendix A.1.1, Enable Extension of Reset Hold Time Using Switch SW5.
- Set the i.MX RT1050 eFuse which enables the boot ROM to reset the Flash device each time the i.MX RT1050 is reset.

#### 3. Boot / XiP Settings

In order to boot from the EcoXiP device, it is necessary to include a suitable Serial NOR Configuration Block in the image.

The Serial NOR Configuration Block occupies the first 512 bytes of the Flash memory. It provides information to the i.MX RT1050 boot ROM on how to configure the FlexSPI host controller and the Flash device with desired parameters for optimal operation. The boot ROM initially reads from the Flash using a universal read command (opcode 03h) at a low speed. Once it reads the Serial NOR Configuration Block and reconfigures the system accordingly, it jumps to the application program entry point. At that point the system starts executing code from Flash. For more information, refer to the *i.MX RT1050 Processor Reference Manua*l, Chapter 8 (System Boot).

The i.MX RT1050 SDK supports this execute-in-place (XiP) mode. Many example projects have one configuration which supports this mode. For example, users of the IAR tools will find in many cases a project configuration called 'Debug' which builds the code to run from TCM (internal SRAM) and next to it another configuration called 'flexspi\_nor\_debug' which builds the code to run from Flash in XiP mode. That project configuration uses a C file named evkbimxrt1050\_flexspi\_nor\_config.c to define the Serial NOR Configuration Block. This file can be found in the 'xip' source group of the project. Once compiled, the file is linked to offset 0 in the flash memory as this is the location where the boot ROM expects the Serial NOR Configuration Block to be located.

The version of *evkbimxrt1050\_flexspi\_nor\_config.c* which comes with i.MX RT1050 SDK is not suitable for EcoXiP. As such, Adesto provides a version of this file which works with EcoXiP. This file is attached to this application note. Note that the .c file name should not be changed. When the project is rebuilt, a bootable image is created that allows the i.MX RT1050 processor to boot from the EcoXiP Flash device. Please note that this source file works for all supported tool chains.



After replacing the *evkbimxrt1050\_flexspi\_nor\_config.c* file in the XiP folder, open your project with your development tool, select the project configuration designed for XiP and build the project to produce a bootable image. Refer to Section 5 for more information on how to load the bootable image and run it.

Try generating an EcoXiP-bootable image for the 'hello\_world' example of the SDK (the "hello world" project can be found in the following path under the SDK root folder: *boards\evkbimxrt1050\demo\_apps\hello\_world*).

The following screen shot shows the hello\_world example using the IAR Workbench.

hello_world - IAR Embedded Workbench IDE - Arr	n 8.22	.1	-	C	$\times$
<u>File Edit View Project CMSIS-DAP Tools W</u>	(indov	v <u>H</u> elp			
15 C   🗋 🗳 🕹 🖬 🖬 🖬			u < Q, > ⇆ HE < 📮 > 🕢 🖻 🗎 🌒 🛲 💽 🕒 📲 🗒		
Workspace	•	, ų ×	evkbimxrt1050_flexspi_nor_config.c ×		-
flexspi_nor_debug		~			fo
Files         Image: board         I	÷		<pre>     * #AREANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE     * DISCLATMED. IN NO SUMET SHALL THE COPYRICHT HOLDER OR CONTRIBUTORS BE LIABLE     * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAG     * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;     * LOSS OF USS, DATA, OR FROFITS; OR BUSINESS INTERREPTION, HOWEVER CAUSED AND     * ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT     * (INCLUDING NEGLIGENCE OR OTHERWISE) ANTISING IN ANY MAY OUT OF THE USE OF THI     * SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.     */     SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.     */     * List (ATXP032). Among other parameters it configues the system to operate     * with:     * - Octal-SPI     * Outble Data Rate (DDR)     * - SCLK = 133MHz     */     #include "evkbimmrt1050_flexspi_nor_config.h" </pre>	FOR IS IN	^
hello_world			<		> ~
Build				•	д×
Messages Linking				>	^ ^ ·
Build Debug Log			Frons 0, Warnings 0, In 1, Col 1, System, CAP, NUM, C	VR S	

Figure 2. Hello World Example - IAR Workbench

#### 4. Loading Code into the EcoXiP Flash

Once a bootable image has been created it has to be loaded into Flash memory. During the development phase this can be done as part of loading a program to target memory from the debugger. The debugger typically has a Flash loader extension or plug-in which can program an image into Flash memory in case the program's image (or part of it) is linked to the Flash memory address range. Different Flash products require different Flash loaders. The user has to let the debugger know which Flash loader to use based on the target Flash memory being used in the system.

Flash loaders for EcoXiP are available for the following major tool chains: •IAR Embedded WorkBench — Version 8.30.1 and up •Keil MDK — Version 5.25.2 and up •MCUXpresso IDE — Version 10.2.0 and up

The configuration required to support each tool chain is described in the following subsections.



#### 4.1 IAR Embedded Workbench

The IAR debugger has a minor issue related to verification after Flash loading, but this issue can easily be resolved using the procedure below. This procedure initially shows how to set up an EcoXiP Flash loader without verification, and then how to enable verification.

In the IAR project option go to Debugger and select the Download tab. Check "Use flash loader(s)", then check "Override default. board file". In the box underneath enter the following:

\$TOOLKIT DIR\$\config\flashloader\NXP\FlashIMXRT1050 EVK FlexSPI EcoXiP.board

You can also browse to find this file using the 3-dot small box to the right of the text box as shown in the following screen shot.

Options for node "exip_rt105	0_boot"		×
Options for node "exip_rt103 Category: General Options Static Analysis Runtime Checking C/C++ Compiler Assembler Output Converter Custom Build Build Actions Linker Debugger Simulator CADI CMSIS DAP GDB Server I-jet/JTAGjet J-Link/J-Trace TI Stellaris Nu-Link PE micro ST-LINK Third Party Driver TI MSP-FET TI XDS	0_boot" Setup Download Imag Ⅳverify download ☑ Suppress download ☑ Use flash loader(s) ☑ Qveride default b ⅣNXP\FlashIMXR	s Extra Options Multicore pard file 1050_EVK_FlexSPI_EcoXIP.	Factory Settings Plugins board
		ОК	Cancel

Figure 3. IAR Embedded Workbench Debugger

The above should work without the "Verify download" option (as you can see above it's not checked so far).

To enable verification, one has to use one of the following two options to work around the aforementioned IAR issue.



#### Option 1:

Enable a macro file to be called after reset. This file re-initializex the FlexSPI host controller and a couple of other things in the MCU. To apply, first copy the attached *EcoXiP\_init.mac* file to the folder where the IAR project is located (same folder where *.eww* and *.ewp* files are located). Then in the *IAR project* option, go to Debugger and select the Setup tab. Check "Use macro file(s)". In the box enter *\$PROJ\_DIR\$\EcoXiP\_init.mac* as shown in the following screen shot.

Category:	Factory Settings	1
General Options Static Analysis		
Runtime Checking	Setup Download Images Extra Ontions Multicore Plugins	
Assembler		1
Output Converter	Driver Run to	
Custom Build	CMSIS DAP v main	
Build Actions	Cotor more a	
Linker	Setup macros	
Debugger	✓ Use macro file(s)	
Simulator	\$PROJ_DIR\$\EcoXiP_init.mac	ļ
CADI CMSIS DAP		ļ
GDB Server		
I-iet/ITAGiet		ļ
J-Link/J-Trace	Device description file	ļ
TI Stellaris	Override default	ļ
Nu-Link	\$TOOLKIT_DIR\$\CONFIG\debugger\NXP\MIMXRT1052xxx6	
PE micro		
ST-LINK		
Third-Party Driver		
TI MSP-FET		

Figure 4. IAR Embedded Workbench Verification Enable - Option 1

Option 2: Change the reset style so that the debugger will not reset the MCU before verification. To apply this option go to CMSIS DAP in the IAR project option. Click on the Setup tab and in the Reset pulldown menu select "Software" as shown in the screen shot below.

Options for node "exip_rt1	050_boot" >
Category:	Factory Settings
Static Analysis	
Runtime Checking	
C/C++ Compiler	Setup Interface Breakpoints
Assembler	Report
Output Converter	Teser
Custom Build	Software
Build Actions	Duration: 300 ms Delau after: 200 ms
Linker	
Debugger	- Fmulator
Simulator	- Always prompt for probe
CADI	selection
CMSIS DAP	Serial no:
GDD Server	
1-Link/1-Trace	
TI Stellaris	
Nu-Link	
PE micro	\$PROJ_DIR\$\cspycomm.log
ST-LINK	
Third-Party Driver	
TI MSP-FET	
TI XDS	
	OK Cancel

Figure 5. IAR Embedded Workbench Verification Enable - Option 2



After applying the settings according to one of the options described above, check "*Verify download*" as shown in Figure3 above. Now you can use the EcoXiP Flash loader with verification.

To load the program image, click the *Download* and *Debug* buttons.

#### 4.2 Keil MDK

Note: If you have MDK version 5.26 or higher, continue to the next paragraph, otherwise carefully read the remainder of this paragraph. At the time this document was released, the latest MDK version was 5.25. This version includes an EcoXiP Flash loader (algorithm in the MDK terminology) which supports i.MX RT1050 rev A0. Due to an incompatible change done by NXP in i.MX RT1050 rev A1, this MCU revision requires a small change in the Flash loader. The EcoXiP Flash loader which supports i.MX RT1050 rev A1 is available and will be released in MDK version 5.26. In the meantime, if you have MDK version 5.25 you can replace the Flash loader with the one provided with this application note. Simply copy the file *MIMXRT105x\_ECOXIP\_4MB\_SEC.FLM* to the ARM\Flash folder under the MDK installation root folder, overwriting the original file.

In the Keil MDK project options, select the *Debug* tab, then click on the *Settings* button on the far right as shown in the following screen shot.

🕅 Options for Target 'hello_world flexspi_nor_debug'	×
Device     Target     Output     Listing     User     C/C++     Asm       C     Use     Simulator     with restrictions     Settings       Image: Limit Speed to Real-Time     Settings	Linker Debug Utilities
Load Application at Startup     Initialization File:     Initialization File:	Load Application at Startup  Initialization File:  evkbimxrt1050_flexspi_nor.ini  Edit
Restore Debug Session Settings         Image: Breakpoints         Image: Breakpoints	Restore Debug Session Settings         Image: Breakpoints       Image: Toolbox         Image: Watch Windows         <
Dialog DLL: Parameter: DCM.DLL pCM7	Dialog DLL: Parameter: TCM.DLL -pCM7
Wam if outdated Executable is loaded Manage Component Vie	Wam if outdated Executable is loaded
OK Car	ncel Defaults Help

Figure 6. Keil MDK - Project Options

When the Driver Setup window pops up, select the Flash Download tab as shown in the following screen shot.



CMSIS-DAP Cortex-M Target Drive	er Setup				×
Debug Trace Flash Download					
Download Function C Erase Full Chip C Erase Sectors C Do not Erase	<ul> <li>✓ Program</li> <li>✓ Verify</li> <li>✓ Reset and F</li> </ul>	RAM for A	Algorithm 0x20000000	Size: 0x00008000	D
Programming Algorithm					
Description	Device Size	Device Type	Addres	s Range	
MIMXRT105x 64mB Hyper Fl	64M	Ext. Flash SPI	6000000H	- 63FFFFFFH	
		Start:	0x6000000	Size: 0x0400000	0
	Add	Remove			
	ОК	Cance	el		Help

Figure 7. Keil MDK - Driver Setup

In the *Programming Algorithm* list, if there is already an existing algorithm for HyperFlash, click on *Remove* to remove it. Now click *Add*. A window with a list of Flash algorithms pops up. Select the algorithm called "*MIMXRT105x EcoXiP Flash*" and click *Add* as shown in the following screen shot.

Description	Flash Size	Device Type	Origin	
/IMXRT105x 64mB Hyper Fl	64M	Ext. Flash SPI	Device Family Package	
IIMXRT105x 8mB QuadSPI	8M	Ext. Flash SPI	Device Family Package	
M29x128 Flash	16M	Ext. Flash 16-bit	MDK Core	
(8P5615UQA Dual Flash	64M	Ext. Flash 32-bit	MDK Core	
PC18xx/43xx MX25V8035F	8M	Ext. Flash SPI	MDK Core	
PC18xx/43xx S25FL032 SP	4M	Ext. Flash SPI	MDK Core	
PC18xx/43xx S25FL064 SP	8M	Ext. Flash SPI	MDK Core	
PC407x/8x S25FL032 SPIFI	4M	Ext. Flash SPI	MDK Core	
PC5460x MT25QL128 SPIFI	16M	Ext. Flash SPI	MDK Core	
129W640FB Flash	8M	Ext. Flash 16-bit	MDK Core	
IIMXRT105x EcoXiP Flash	4M	Ext. Flash SPI	MDK Core	
C28F640J3x Dual Flash	16M	Ext. Flash 32-bit	MDK Core	
29GL064N Dual Flash	16M	Ext. Flash 32-bit	MDK Core	
29JL032H_BOT Flash	4M	Ext. Flash 16-bit	MDK Core	
29JL032H_TOP Flash	4M	Ext. Flash 16-bit	MDK Core	
Keil_v5\ARM\flash\MIMXRT1	05x_ECOXIP_	4MB_SEC.FLM		





Select the EcoXiP Flash algorithm (only one visible) and click OK as shown in the following screen shot.

CMSIS-DAP Cortex-M Target Drive	er Setup			×
Debug Trace Flash Download				
Download Function C Erase Full Chip C Erase Sectors C Do not Erase	Program     Verify     Reset and F	RAM for A Start:	Algorithm	00
Programming Algorithm				
Description	Device Size	Device Type	Address Range	
MIMXRT105x EcoXiP Rash	4M	Ext. Flash SPI	60000000H - 603FFFFFH	
		Start:	0x60000000 Size: 0x004000	00
	Add	Remove		
	Oł	K Cance	el	Help

Figure 9. Keil MDK - Select EcoXiP Flask

The program can now be loaded into the EcoXiP Flash. To load the image, click on the Load button.

### 4.3 MCUXpresso IDE

When importing the SDK example into the MCUXpresso IDE, an Advanced Settings window is displayed as shown in Figure 10 that includes a memory configuration box at the bottom. This is where you can set up the EcoXiP Flash driver.

Note: If the example project has already been imported you can still modify the memory configuration by opening the project properties dialog box and under  $C/C^{++}$  Build by selecting 'MCU Settings'.



Advanced Settings      C(C++ Library Settings      Set library type (and hosting variant)     Redlik Use floating point version of printf     Redlik Use floating Point type     Phys-SP-D16 (HardABI)      MCU Compiler     Longuage standard GNU C99 (-std=gnu99)      MCU Linker     LInk application to RAM      Memory Configuration     Memory details     Type     Name Alias Location	NewlibNano: Use floating point versi     NewlibNano: Use floating point versi     Redirect printf/scanf to UART      Scation     Size     Goddood     Goddood     Goddood	rsion of printf rsion of scanf
Advanced Settings  C(C++ likrary Settings  C(C++ likrary Settings  Set likrary type (and hotsing variant) Redilib (semihost-nf)  Redire: Use floating point version of printf Redire: SDK "PRINTF" to C likrary "printf" to C likrary "printf"	NewlibNanc: Use floating point versi     NewlibNanc: Use floating point versi     Redirect printf/scanf to TM     Redirect printf/scanf to UART  cotation Size 0400000 04000000 04000000 04000000 04000000	rsion of printf rsion of scanf
Advanced Settings  C(C++ library Settings  Set library type (and hosting variant) Redib (semihost-nf)  Redib: Use floating point version of printf Redib: Use floating Pointf Redib: Use floating Redib: Use floating Pointf Redib: Use floating Pointf Redib: Use floating Pointf Redib: Use floating Redib: Use fl	NewlibNano: Use floating point versi     NewlibNano: Use floating point versi     Redirect print/scanf to ITM     Redirect print/scanf to UART  c.c.tion     Size     doccooo     doccooo	rsion of printf rsion of scanf
	NewlibNanc: Use floating point versi     NewlibNanc: Use floating point versi     Redirect printf/scanf to UART     Redirect printf/scanf to UART	rsion of printf rsion of scanf
Set library type (and hosting variant)  Redlik (semihost-nf)  Redlik: Use floating point version of printf Redlik: Use floatacter rather than string based printf Redlik: Use floating Point Yer (birsyr 'printf' Concluse semihost HardFault handler  Hardware settings Set Floating Point type [FpV3-SP-D16 (HardABI)  KUC Compiler Longuage standard (RNU C99 (-std=gnu99)  KUL linker Link application to RAM Komory details  Type Name Alias Location		rsion of printf rsion of scanf
	NewlibNano: Use floating point versi     NewlibNano: Use floating point versi     Redirect printf/scanf to TIM     Redirect printf/scanf to UART  .ccation     Size     b6000000     b4000000	rsion of printf rsion of scanf
	NewlishAnce Use floating point versi     Redirect printf/scanf to TIM     Redirect printf/scanf to UART      Size     Size     Socation     Size     Socococo     Socococo	rsion of scanf
□ Redirect SDK *PRINTF* to C library *printf*       ✓ Include semihos HardFault handler       ✓ Include semihos HardFault handler       ✓ Bradware settings       Set Floating Point type       Fpv5-SP-D16 (HardABi)       ✓ MCU Compiler       ✓ MCU Linker       ✓ Includesition to RAM       ✓ Memory Configuration       ✓ Type       Name     Alias	coation     Size     0000000     000000	
Include semihost HardFault handler       • Hardware settings       Set Floating Point type       • MCU Compiler       Language standard       ONU C99 (+std=gnu99)       • MCU Linker       □ Link application to RAM       • Memory configuration       Memory details       Type     Name       Alias     Location	Redirect print/scanf to UART	
Hardware settings Set Floating Point type     Fby-5.SP-D16 (HardABI)      MCU C Compiler  Longuage standard     GNU C99 (-std=gnu99)      MCU Linker      Link application to RAM      Memory Configuration      Memory data      Type     Name Link      Alas     Location	.ocation Size	
Set Floating Point type Set Floating Point Set Set Set Set Set Set Set Set Set Se	.ocation Size	
Ar MOU Compiler     Induction of RAM     Void University     MOU Compiler     Induction of RAM     Void University     Mamory Configuration Memory details     Type     Name     Alias     Location	.ocation Size	
MCU C Compiler  Language standard     GNU C99 (-std=gnu99)      MCU Linker      Link application to RAM      Memory Configuration  Memory details      Type     Name Alias     Location	.ocation Size	
Language standard GNU C99 (-std:sgnu99)  + MCU Linker  Inte splication to RAM  + Memory Configuration  Type Name Alias Location	.ocation Size	
MCU Linker     Link application to RAM     Memory Configuration     Memory details     Type Name Alias Location	Location Size	
MC-Durker     Minese      Memory Configuration     Memory details     Type     Name     Alias     Location	Location Size	
Link splication to NAM ★ Memory Configuration Memory details Type Name Alias Location	.ocation Size	
Memory Configuration Memory details Type Name Alias Location	Location Size	
Memory details           Type         Name         Alias         Location           Tr         Country of the second	Location Size	
Type Name Alias Location	Location Size 0x6000000 0x4000000 0x2000000 0x20000	
	0x4000000 0x4000000 0x20000	Driver
Hash BOAKD_FLASH Hash 0x0000000	N-20000000	MIMXRT1050-EVK_S26KS512S.cfx
RAM SRAM_DTC RAM 0x2000000	00200000	Fe
RAM SRAM_TIC RAM2 UXU	x0 0x20000	122
RAM POAPD SDRAM PAMA 0.0000000	>20200000 0x40000	
ICAINI BOARD_JDICAINI ICAINIA 00000000	0,200000	

Figure 10. Advanced Settings Display Window

Click on the Edit button on the right side of the Memory Details box in Figure 10. The following screen shot is displayed.

emory	configuration					
efault f	lash driver					Browse
Туре	Name	Alias	Location	Size	Driver	4
Flash	BOARD_FLASH	Flash	0x6000000	0x4000000	MIMXRT105	)   🚆
RAM	SRAM_DTC	RAM	0x20000000	0x20000		
RAM	SRAM_ITC	RAM2	0x0	0x20000		
RAM	SRAM_OC	RAM3	0x20200000	0x40000		
RAM	BOARD_SDRAM	RAM4	0x8000000	0x2000000		

Figure 11. MCUExpresso IDE Memory Configuration Editor

There will be one Flash entry listed. On the right hand side in the Driver column above, double-click on the 3 dots to the right of the driver's name as noted by the orange circle in Figure 11. The following screen shot is displayed.



MCUXpresso IDE	×
LinkServer flash driver	X
Flash Driver	
Flash driver MIMXRT1050-EcoXiP_ATXP032.cfx	V Browse project Browse workspace
	OK Cancel

Figure 12. MCUExpresso IDE LinkServer Flash Driver

From the Flash Driver pull-down menu, select the EcoXiP Flash driver: *MIMXRT1050-EcoXiP-ATXP032.cfx*, as shown in the above screen shot. Click *OK* to return to the Memory Configuration Editor. Click *OK* to exit the editor.

The debugger is now ready to load the program. To load the image, click on the Debug button.

#### 5. Boot / Run

After the program image has been loaded into Flash memory, the program can be executed in one of two ways:

- Stay in the debug session and run the application program from the debugger
- Exit the debug session and run the application program stand alone

To run the application program stand-alone, make sure that the board is set up as follows:

- On DIP switch SW7, set switch 3 to the ON position, and switches 1, 2, and 4 to the OFF position. This configures the boot ROM to boot from the EcoXiP Flash device.
- On DIP switch SW5, set switches 2 and 3 to the ON position, and switches 1 and 4 to the OFF position. If no DIP switch is installed in SW5, two wires must be placed, one connecting pads 2 and 7 and another connecting pads 3 and 6 (see Appendix A). This configures the hold time (the time between Flash reset and the first Flash access).

Once the board is set up correctly, either press the reset button (SW3) or power cycle the board. In both cases the application program automatically runs from the EcoXiP Flash.

In the case of the "hello world" example project the following message appears in the display window.



Figure 13. Hello World Display



#### APPENDIX A. REWORK INSTRUCTIONS

The following steps are used to rework the NXP board to work with the Adesto EcoXiP Flash device.

#### A.1 Replace Flash Device

The first step is to replace the Flash component at U19 on the board (most likely a HyperFlash device) with the Adesto EcoXiP ATXP032 Flash.

Note that both devices come in a 24-pin BGA package.



Figure 14. Replace U19 with Adesto EcoXiP Flash Device

#### A.1.1 Enable Extension of Reset Hold Time Using Switch SW5

When the board is shipped from NXP, location SW5 is not populated. However, for synchronization purposes, it is recommended that the EcoXiP Flash device be reset at the same time at the i.MX RT1050 processor. This requires modification to the pins associated with device SW5 on the board.

Pins 2 and 7, and pins 3 and 6 of SW5 must be connected to configure the appropriate amount of delay to allow the EcoXiP Flash to stabilize before being accessed by the i.MX RT1050 processor.

There are two methods that can be used to connect pins 2 and 7, and pins 3 and 6 at location SW5.

- Use wires to jumper pins 2 and 7, and pins 3 and 6 to manually configure the appropriate i.MX RT1050 processor pins to a logic HIGH.
- Install a 4-position DIP switch at location SW5 and set switches 2 and 3 to the ON position.

To solder wires between the DIP switch pins in order to manually set them to the ON position, refer to Section A.1.2.

To add a DIP switch to location SW5, refer to Section A.1.3.



#### A.1.2 Jumper Pins at Location SW5

To manually configure the appropriate processor pins to a logic HIGH, wires must be soldered between pins 2 and 7, and pins 3 and 6 of SW5 as shown in Figure 15.



Figure 15. Using Wires to Jumper Select Pins at Location SW5

#### A.1.3 Add a DIP Switch at Location SW5

As an alterative to connecting wires between selected pins at location SW5 as described in the previous subsection, a DIP switch can be added, and then switches 2 and 3 set to the ON position.



Figure 16. Adding a DIP Switch at Location SW5



#### A.2 JEDEC Reset and eFUSE Programming

To ensure that the i.MX RT1050 MCU and the EcoXiP Flash never lose synchronization with each other, it is best if the EcoXiP Flash is reset each time the MCU is reset. To achieve this, the i.MX RT1050 boot ROM option must be enabled to automatically apply a JEDEC reset to the EcoXiP Flash right after the MCU reset.

# A.2.1 JEDEC Reset

A JEDEC reset is a sequence of signals sent by a host MCU to the Flash device which results in a reset event occurring on the Flash device. The sequence is transmitted by the host over two of the SPI interface pins. The EcoXiP Flash supports the JEDEC reset option as an alternative to applying a reset on its RESET input pin.

To enable an automatic JEDEC reset by the i.MX RT1050 boot ROM, it is necessary to set an eFuse of the i.MX RT1050, specifically bit 6 in the OTP register located at offset 0x6E0 in the eFuse array of the i.MX RT1050 processor.

# A.2.2 Setting the eFuse

Setting an eFuse is a quick and easy procedure done with the help of NXP's MfgTool (manufacturing tool program). This tool comes as part of NXP's "Flashloader i.MX-RT1050" package which can be downloaded from here.

https://www.nxp.com/products/processors-and-microcontrollers/arm-based-processors-and-mcus/i.mx-applications-processors/i.mx-rt-series/i.mx-rt1050-crossover-processor-with-arm-cortex-m7-core:i.MX-RT1050?&tab=Design\_Tools\_Tab

On the above NXP web page, scroll down to select Flashloader i.MX\_RT1050 and click Download.

Once the package is downloaded, the *MfgTool* program can be found in the following path under the package root folder: Tools\mfgtools-rel. The program's name is MfgTool2.exe. Before running the program, copy the attached boot\_image.sb file to the following folder under the package root folder: Tools\mfgtoolsrel\Profiles\MXRT105X\OS Firmware. This is an input script for *MfgTool* and its only purpose is to set the above eFuse to 1.

Once the file is copied, connect power to the NXP I.MX RT1050 EVKB board to power as normal by connecting a USB cable from a PC host to location J28 on the NXP board. In addition, connect a USB cable from that PC host to location J9 on the NXP board (this is the communication channel of the MfgTool program). Set all DIP switches on SW7 to the OFF position, then push the SW3 button to reset the system.

Invoke the MfgTool program by double-clicking on the MfgTool2.exe file as shown below.

[	Name ^	Date modified	Туре	Size
	Profiles	5/17/2018 1:18 PM	File folder	
	📓 cfg.ini	5/17/2018 1:18 PM	Configuration sett	1 KB
	MfgTool.log	6/25/2018 2:50 PM	Text Document	7 KB
	MfgTool2.exe	5/17/2018 1:18 PM	Application	1,955 KB
	🚳 MfgToolLib.dll	5/17/2018 1:18 PM	Application extens	2,205 KB
	📓 UlCfg.ini	5/17/2018 1:18 PM	Configuration sett	1 KB

Figure 17. Invoke the MfgTool Program



The *MfgTool* window is displayed as shown below.

MfgTool_MultiPanel (Library: 2.7.0)	—	X
Hub 4Port 3	Status Information	
Drive(s):	Successful Operations:	0
HID-compliant vendor-defined device	Failed Operations:	0
	Failure Rate:	0 %
	Start	Exit

Figure 18. MfgTool Main Window

In the above screen shot, the message: HID-compliant vendor-defined device is displayed.

*Click Start.* The operation should complete instantly and green bars should be displayed at the bottom. Once these bars are visible, click *Stop* and then *Exit.* At this point the JEDEC Reset eFuse is set.



#### **Additional Information**

Adesto Technologies Corporation (NASDAQ:IOTS) is a leading provider of innovative application-specific semiconductors and embedded systems for the IoT era. The company's technology is used by more than 2,000 customers worldwide who are creating differentiated solutions across industrial, consumer, medical and communications markets. With its growing portfolio of high-value technologies, Adesto is helping its customers usher in the era of the Internet of Things.

Visit: www.adestotech.com

Adesto and the Adesto logo are trademarks or registered trademarks of Adesto Technologies Corporation or its subsidiaries in the United States and other countries. Other company, product and service names may be trademarks or service marks of others. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Adesto.

Copyright © 2019 Adesto Technologies Corporation. All rights reserved.

