8A3xxxx / ClockMatrix

ClockMatrix Firmware Update through Serial Port and EEPROM v1.0

Abstract

This document describes how to update the firmware into the RAM of the ClockMatrix device.

The ClockMatrix device ships from the factory with a default firmware in the ROM. At power-up, the ROM contents are loaded into the RAM. However, the device has the option to have the firmware in the RAM updated by either writes to the serial port (I2C or SPI) or through an EEPROM. Since each die revision of the CM (ClockMatrix) devices contains a different ROM, its FW (firmware) update sequence is unique. For this reason, there is an EEPROM image and a corresponding serial update log for each die revision. Both the EEPROM images and the serial port logs contain a checksum at the end that ensures that the correct data is loaded into the RAM. If the chip determines that the contents it loaded result in an incorrect checksum, it will reject the RAM update and revert to the FW in the ROM. This prevents the chip from loading a corrupted FW image.

The following instructions provide guidance on how to update the RAM.

Contents

Figures

Tables

1. Serial Port Firmware Update into the RAM

1.1 Selecting the Log File

Renesas provides a zipped folder that contains logs for the firmware update for each die revision and each of the following protocols:

- I2C, 1-byte addressing
- I2C, 2-byte addressing
- SPI, 1-byte addressing
- SPI, 2-byte addressing

Generic devices (dash code -000) always boot in 1-byte mode and the protocol, I2C or SPI, is selected by the state of GPIO9 when nMR is de-asserted. Refer to the datasheet for more details. If the device contains an OTP (One-Time Programmable) configuration (non -000 dash codes) the protocol type and byte-size may be pre-determined by the configuration. In that case, check the configuration file values for “SER0_CONFIRM_CODE” (Main port) and “SER1_CONFIRM_CODE” (AUX port) .If either is set so 0x10, then the protocol for that port will be overwritten so use the corresponding log file.

The FW update log file will be named according to the protocol. For example, “FW4.8.7_Upgrade_revB_I2C_1-byte-Log.txt” means that it’s the FW update log for FW4.8.7 for the revB device, using the I2C protocol and 1-byte addressing.

1.2 Preparing the Log

The contents of the file are presented in a generic syntax. Customers will need to parse and reformat the log into the unique syntax that is compatible with their systems. Figure 1 below shows an excerpt of the I2C 1-byte log. Note that it contains a header showing the firmware version. The “5B” is the CM default 7-bit I2C address when I2C address pins A0 and A1 are both high at the de-assertion of nMR. (See datasheet for more details). The customer should review the A0 and A1 pin in their hardware configuration and update this address accordingly.

```
+-------------------------------------------------------------------------+
| Version: 4.8.7.46805.1                                                  |
+-------------------------------------------------------------------------+
--------------------------------<  START HERE  >--------------------------------
i2c write device 5B data FC 00 C0 10 20
i2c write device 5B data 0C AA 00 00 00
i2c write device 5B data FC 00 81 10 20
i2c write device 5B data 80 01
```

Figure 1. I2C 1-byte Log Header

The end of the log file also contains an instruction that indicates how long to wait for the firmware to update into the RAM. See Figure 2 below.

```
i2c write device 5B data FC 00 6A 10 20
i2c write device 5B data 00 F0 4F FF F7 DD BB 07 B0 BD E8 F0 8F AF F3 00 80 00 88 52 6A 74 00 00 00 70
i2c write device 5B data FC 00 C0 10 20
i2c write device 5B data 00 94 6A 10 20
NOTE: after upload procedure wait about 2 seconds before accessing device (due to eeprom scanning)
--------------------------------<   END  HERE  >--------------------------------
```

Figure 2. I2C 1-byte Log, End of File

The SPI update procedure is similar to the I2C.

1.3 Writing to the Device

At power-up, there is a pre-determined wait time (t_{wait}), after nMR de-assertion, before writing to the device. Recommendations are as follows:

- If part is OTPd and EEPROM load is disabled: t_{wait} = 15ms
- If EEPROM load is enabled but no EEPROM is present: t_{wait} = 150ms
- If EEPROM load is enabled and EEPROM is present: t_{wait} = 2s

Use the following procedure to update the FW:

1. Power up the chip with nMR held low.
2. After VDD is at 95%, de-assert nMR.
3. Wait the recommended time, t_{wait}, after nMR de-assertion and then upload the FW using the reformatted log.
4. Once the serial port updates are complete, wait for the specified amount of time recommended by the log file (see Figure 2) for the FM to update.
5. Finally, read back the FW version to determine that it updated correctly.

2. EEPROM Update into the RAM

2.1 Selecting the EEPROM Image

The CM can load the FW update at boot-up by reading it from an EEPROM. It supports multiple EEPROMs addresses. Here is the relevant excerpt from the datasheet section “Use of I2C EEPROM”:

> *The 8A34001 will use its I2C Master Port to attempt to access an external I2C EEPROM at base address 1010000 (binary) at an I2C frequency of 1MHz. If there is no response, this will be repeated at base address 1010001 (binary) at 1MHz. This will repeat up to address 1010111 (binary) at 1MHz. If there still are no responses, the search will be repeated at 400kHz and then again at 100kHz. If no response is received after this entire sequence, the device will assume there is no EEPROM available. Any errors in the process will be reported in status registers.*
>
> ***Device Updates in External I2C EEPROM*** *As indicated in Reset Sequence, if enabled, the 8A34001 will search for Device Update information in an external I2C EEPROM. It will first identify all valid EEPROMs attached to the I2C master port as described above. Each valid EEPROM will be checked for a valid Device Update Block header with valid checksum at address offset 0x0000 within the EEPROM. The first such valid block will be used as described in Reset Sequence.*

Renesas provides EEPROM images for each die revision and EEPROM addresses 0x50/0x51 and 0x50/0x54. See Figure 3 for an example list of the EEPROM images for FW4.8.7.

- The 0x50/0x51 address correspond to the CAT24M01 EEPROM.
- The 0x50/0x54 addresses correspond to the 24FC1025 or 24AA1025 1Mbit EEPROM.

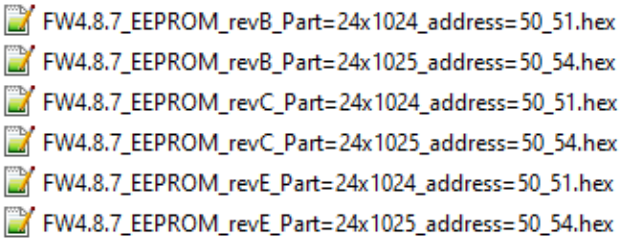

Figure 3. EEPROM Images List

EEPROM usually have an address selector pin that allows multiple EEPROMs to share the same bus. The addresses listed above are the default addresses when the address select pins are low. Contact Renesas if you need a different EEPROM address. Alternately, the Timing Commander GUI can be used to generate the image, as shown in the next section.

2.2 How to Generate the EEPROM Hex Image using Timing Commander

In the case that a specific EEPROM address is needed, or the EEPROM zip file is unavailable, the Timing Commander ClockMatrix GUI can be used to generate the EEPROM image. If a generic default configuration is needed, then the recommendation is to start with a new GUI session without making any changes to the configuration and then follow the steps below. Otherwise, the user can also load an existing configuration into the GUI and have the configuration included in the EEPROM image. The instructions are below:

1. Open the EEPROM Utility.
2. Select the EEPROM Type (which is the same as selecting an address). Use the “Other” selection if a custom address is needed (see Figure 5).

Figure 4. EEPROM Generation with GUI

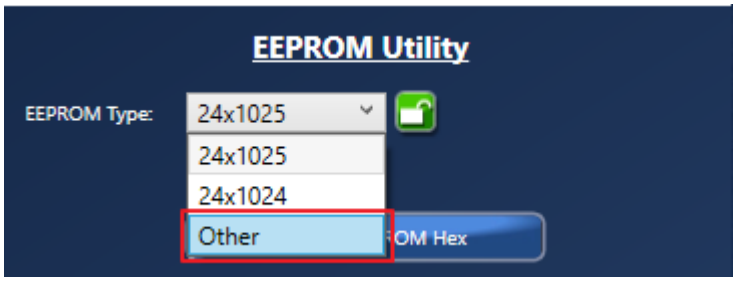

Figure 5. EEPROM Custom Address Selection

3. Press “Generate EEPROM hex” (see Figure 4).
4. Press “Yes”, so that the EEPROM image contains the Firmware (see Figure 6).

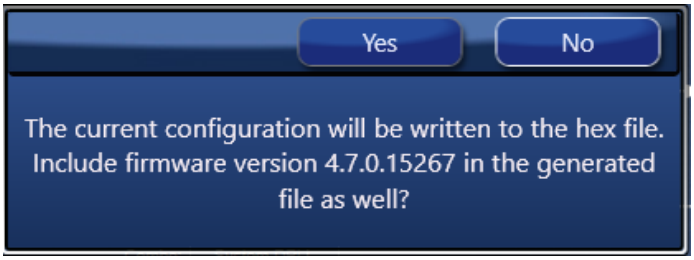

Figure 6. Confirm Firmware Inclusion in the EEPROM Image

2.3 EEPROM Hardware Setup

The example schematic is shown in Figure 7.

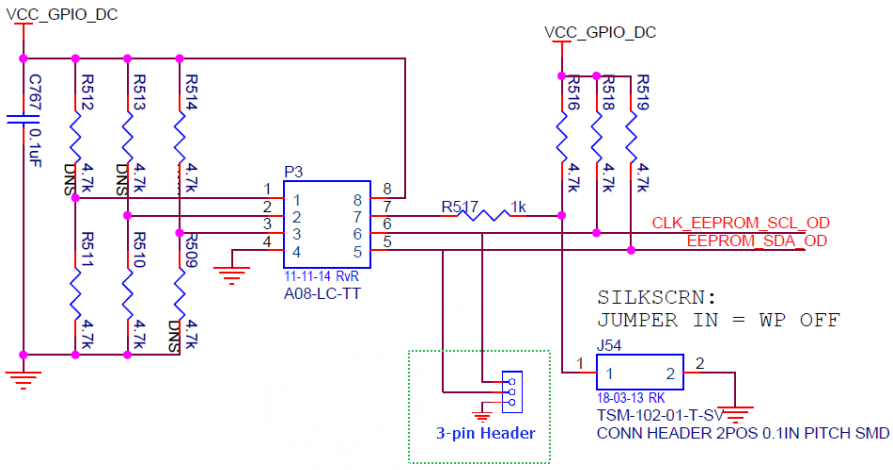

Figure 7. EEPROM Circuit

Connect the SCLK and SDA traces to the ClockMatrix device Master I2C port (SCLK and SDA). Another recommendation is to also place a 3-pin header that connects to the EEPROM SCLK, SDA, and GND pin (see green outline in Figure 7). This way an external programmer can be connected to the system for programming. Be aware that external programmers may only support 3.3V levels, so a level shifter may be necessary. Figure 8 shows an example of the translator used on the ClockMatrix board. The translator shown in the example requires that VCCA < VCCB, so VCCB would always be the 3.3V supply for the external programmer card and VCCA will be set to the same voltage level as the EEPROM supply.

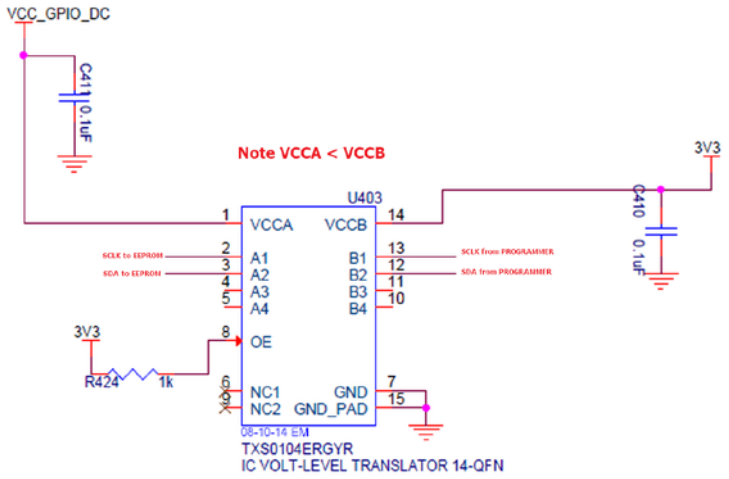

Figure 8. I2C Level Translator

In the case of prototype boards, a PDIP socket, such as the DILV8P-223TLF can be used (see Figure 9). This will allow for easy removal of the EEPROM so that it can be programmed using an external board, such as the Total Phase EEPROM Socket Board.

Figure 9. PDIP Socket

2.4 Programming the EEPROM

For instructions on how to update the firmware on an EEPROM on a ClockMatrix evaluation board, refer to the document “ClockMatrix Evaluation Board: Update Firmware on Device and EEPROM”.

There are several methods to program the EEPROM. It can be programmed using an EEPROM flash card, using a header, or through the ClockMatrix chip. The follow sections provide details for each option.

2.4.1. External Card Option

For this option, use the PDIP socket. Use the Total Phase EEPROM socket board to program the EEPROM and then place the EEPROM into the socket. Refer to document “Programming the 8A3x000_EEPROM_with_TotalPhase_v1.3.pdf” for details.

Alternately, another vendor’s EEPROM flash card can be used.

2.4.2. Header Option

In this option, the EEPROM is installed in the system board and an accompanying 3-pin header is Use the same process as described in the “Programming the 8A3x000_EEPROM_with_TotalPhase_v1.3.pdf” document. However, connect the Total Phase Aardvark card directly to the 3-pin header and use the Total Phase software to program the EEPROM.

2.4.3. ClockMatrix Option

In this option, the ClockMatrix chip serves as the intermediary, transferring the data from the System Device to the EEPROM, as shown in Figure 10. Either a CPU, an FPGA, or a different system device will “talk” to the ClockMatrix device through the Main or Auxiliary serial port and send it the firmware data. In turn, the ClockMatrix device will send the data to the EEPROM chip. Note that it is the customer’s responsibility to parse the .hex file into a format that is usable by their system.

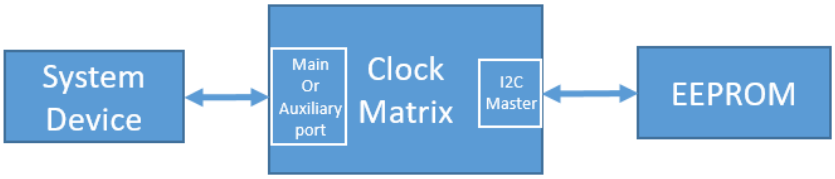

Figure 10. Programming the EEPROM through ClockMatrix

ClockMatrix contains a register buffer that functions as an EEPROM read/write buffer. Access to the buffer is obtained by using the registers shown below. For register offset values, refer to the “8A3xxxx Family Programming Guide” that corresponds to the firmware present in the ROM.

2.4.4. EEPROM Read/Write Buffer Registers

Module: EEPROM

Access EEPROM.

Table 369: EEPROM Register Index

Offset (Hex)	Register Module Base Address: CF68h	
	Individual Register Name	Register Description
000h	EEPROM.EEPROM_I2C_ADDR	EEPROM I2C address.
001h	EEPROM.EEPROM_SIZE	EEPROM data transfer size.
002h	EEPROM.EEPROM_OFFSET	EEPROM offset.
004h	EEPROM.EEPROM_CMD	EEPROM command.

Figure 11. EEPROM Register Index

EEPROM.EEPROM_SIZE

Configure the number of bytes to read or write.

Table 371: EEPROM.EEPROM_SIZE Bit Field Locations and Descriptions

Offset Address (Hex)	EEPROM.EEPROM_SIZE Bit Field Locations							
	D7	D6	D5	D4	D3	D2	D1	D0
001h	BYTES[7:0]							

EEPROM.EEPROM_SIZE Bit Field Descriptions			
Bit Field Name	Field Type	Default Value	Description
BYTES[7:0]	R/W	0	Number of bytes to read or write. These bytes are transferred through OTP_EEPROM_BUFF. Valid range is 0 to 128.

Figure 12. EEPROM.EEPROM_SIZE Register

EEPROM.EEPROM_I2C_ADDR

EEPROM I2C address.

Table 370: EEPROM.EEPROM_I2C_ADDR Bit Field Locations and Descriptions

Offset Address (Hex)	EEPROM.EEPROM_I2C_ADDR Bit Field Locations							
	D7	D6	D5	D4	D3	D2	D1	D0
000h	RESERVED	I2C_ADDR[6:0]						

EEPROM.EEPROM_I2C_ADDR Bit Field Descriptions			
Bit Field Name	Field Type	Default Value	Description
RESERVED	N/A	-	This field must not be modified from the read value
I2C_ADDR[6:0]	R/W	0	I2C address of the EEPROM.

Figure 13. EEPROM.EEPROM_I2C_ADDR Register

EEPROM.EEPROM_OFFSET

-

Table 365: EEPROM.EEPROM_OFFSET Bit Field Locations and Descriptions

Offset Address (Hex)	EEPROM.EEPROM_OFFSET Bit Field Locations							
	D7	D6	D5	D4	D3	D2	D1	D0
002h	EEPROM_OFFSET[7:0]							
003h	EEPROM_OFFSET[15:8]							

EEPROM.EEPROM_OFFSET Bit Field Descriptions			
Bit Field Name	Field Type	Default Value	Description
EEPROM_OFFSET[15:0]	R/W	0	-

Figure 14. EEPROM.EEPROM_OFFSET Register

EEPROM.EEPROM_CMD

-

Table 366: EEPROM.EEPROM_CMD Bit Field Locations and Descriptions

Offset Address (Hex)	EEPROM.EEPROM_CMD Bit Field Locations							
	D7	D6	D5	D4	D3	D2	D1	D0
004h	EEPROM_CMD[7:0]							
005h	EEPROM_CMD[15:8]							

EEPROM.EEPROM_CMD Bit Field Descriptions			
Bit Field Name	Field Type	Default Value	Description
EEPROM_CMD[15:0]	R/W	0	- EE01 = Read from EEPROM EE02 = Write to EEPROM EE03 = Write (no verify)

Figure 15. EEPROM.EEPROM_CMD Register

GENERAL_STATUS.EEPROM_STATUS

-

Table 11: GENERAL_STATUS.EEPROM_STATUS Bit Field Locations and Descriptions

Offset Address (Hex)	GENERAL_STATUS.EEPROM_STATUS Bit Field Locations							
	D7	D6	D5	D4	D3	D2	D1	D0
008h	EEPROM_STATUS[7:0]							
009h	EEPROM_STATUS[15:8]							

GENERAL_STATUS.EEPROM_STATUS Bit Field Descriptions			
Bit Field Name	Field Type	Default Value	Description
EEPROM_STATUS[15:0]	R/O	0	Current status of EEPROM. 0x0000 = no status 0x8000 = ok 0x8001 = unknown command 0x8002 = wrong size 0x8003 = out of range 0x8004 = read failed 0x8005 = write failed 0x8006 = verification failed

Figure 16. GENERAL_STATUS.EEPROM_STATUS Register

2.4.4.1. EEPROM Buffer Register

This is 128 bytes in length and is the location of the data that is either written to the EEPROM or read from the EEPROM. For PR4.0, the location is 0xCF80.

Below is an example of an SPI 2-byte addressing sequence that shows how to program the first 128 bytes of an EEPROM. This example is based on PR4.0 Firmware. For a different FW, refer to the Programming Guide.

2.4.4.2. Write EEPROM Example

Table 1. Write EEPROM Example

7F FD 80 10 20	#Set the ClockMatrix page register (SPI 2-byte addressing)
4F 68 50	#Write to CF68, set EEPROM_I2C_ADDR = 0x50
4F 69 80	#Write to CF69, set EEPROM_SIZE = 128 bytes
4F 6A 00 00	#Write to CF6A, Set EEPROM_OFFSET = 0 (write starting at location 0)
4F 80 85 AA 85 AA 84 AA 84 AA 83 AA 83 AA 82 AA 82 AA 81 AA 81 AA 80 AA 80 AA 79 AA 79 AA 78 AA 78 AA 77 AA 77 AA 76 AA 76 AA 75 AA 75 AA 74 AA 74 AA 73 AA 73 AA 72 AA 72 AA 71 AA 71 AA 70 AA 70 AA 69 AA 69 AA 68 AA 68 AA 67 AA 67 AA 66 AA 66 AA 65 AA 65 AA 64 AA 64 AA 63 AA 63 AA 62 AA 62 AA 61 AA 61 AA 60 AA 60 AA 59 AA 59 AA 58 AA 58 AA 57 AA 57 AA 56 AA 56 AA 55 AA 55 AA 00 00 00 00 #Write to CF80, these are the 128 bytes that will be written to the EEPROM.	
4F 6C 02 EE	#Write to CF6C, EEPROM_CMD = ”0xEE02”, trigger EEPROM write

#add a wait time to allow the chip to write to the EEPROM. For now, use 100ms. Alternately, read back the EEPROM_STATUS and verify that it is 0x8000 and then proceed.

Repeat this for each 128 bytes of EEPROM data.

2.4.4.3. Read EEPROM Example

Table 2. Read EEPROM Example

4F 6A 00 00	#Write to CF6A, set offset
4F 6C 01 EE	#Write to CF6C, trigger EEPROM read
#add a wait time to allow the chip to write to the EEPROM. For now, use 100ms.	
7F FD 80 10 20	#Set page register
	#Read from CF80, read back buffer (note that MSB is set for read commands)
CF 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	

3. Revision History

Revision	Date	Description
1.00	Jan.28.20	Initial release.

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers skilled in the art designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only for development of an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising out of your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Rev.1.0 Mar 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.