ClockMatrix

Delta Configuration Updates

Abstract

This document explains the procedure to update a configuration for a ClockMatrix device that has been previously configured.

Contents

Figures

Tables

1. Overview

The ClockMatrix device has a 32k-byte user-accessible register space. Rather than write all the registers to configure the device, it is only required to write to the non-default registers. The configuration can be applied to the device either through One-Time-Programming (OTP) at the factory, an EEPROM, or serial ports (I^2C/SPI) updates. Regardless of the method, the set of registers is composed of non-default values. For instructions on how to export an EEPROM image or Serial Port writes for programming a blank device, see the ClockMatrix GUI Step-by-Step User Guide (pages 28 and 32), respectively. The scope of this document is limited to instructions on how to update an existing configuration.

1.1 Terminology

Updating a configuration must take into account the register delta between the two configurations, the page registers, and the trigger registers. The following list provides a brief explanation of these terms:

- Register Delta – This term is used to describe the differences in registers between two different configurations. This can be obtained using a text editor tool to compare the differences between two Timing Commander Configuration (.tcs) files.
- Trigger Registers – Within the ClockMatrix register map, registers are grouped together into modules. Many register modules include a trigger register that must be written to cause register changes within that module to take effect; the trigger register must be written even if the contents of the trigger register are not changed. Where present, the trigger register is always the last register in the module so that the last write of a burst will trigger any changes in the module.
- Page Registers – As of this document's release, the ClockMatrix register map is composed of 16 pages, from 0xC0 to 0xCF, with 256 registers in each page. Writing to a specific register requires a Page Register and the Register address. Though this document includes instructions on which registers need to be written, it does not discuss how to write to those registers.

For more information on Trigger Registers and Page registers, see the 8A3xxxx Family Programming Guide (v4.8.7).

2. Register Configuration Analysis

A simplified register map can be used to understand how the Renesas tools generate configurations for the ClockMatrix device. The following example uses a register map with three registers and the following default values:

Default Registers

Register A = 0

Register B = 1

Register C = 0

Now, let's define two example configurations, with the following register values:

Configuration 0

Register A = 1

Register B = 0

Register C = 0

Configuration 1

Register A = 0

Register B = 0

Register C = 1

Table 1 and Table 2 provide the delta registers between the default and each configuration. These are the changes required to configure the device. For the reason of simplicity, trigger registers and page registers are not included in the analysis.

Configuration 0 has two registers that do not match the default values, so it would require writes to Register A and Register B. The green-shaded boxes show the required changes.

Table 1. Configuration 0 Example

	Register A	Register B	Register C
Defaults	0	1	0
Config 0	1	0	0
Delta Registers	RegA = 1	RegB = 0	

Configuration 1 has two registers that do not match the default values, so it would require writes to Register B and Register C. The green-shaded boxes show the required changes.

Table 2. Configuration 1 Example

	Register A	Register B	Register C
Defaults	0	1	0
Config 1	0	0	1
Delta Registers		RegB = 0	RegC = 1

Next, let's consider what happens if the device contains Configuration 0 and the user wants to update it to Configuration 1. The following example uses an OTP configured device that contains Configuration 0. In the example, the objective is to update the device to Configuration 1. The example uses the values from Table 1 and Table 2 for the analysis. Table 3 shows that if Configuration 1 is applied to the device without considering Configuration 0, then the resulting configuration will be a mix of both configurations, and incorrect. The green-shaded boxes indicate changes in the bit value. The red boxes show the mismatch between the expected values for Configuration 1 and the final values.

Table 3. Example of an Incorrect Configuration Update

Step	Action	Register A	Register B	Register C	Comment
1	**Power-up Boot**	0	1	0	Device Defaults Load
2	**OTP Applies Config 0 to the device all at once.**	1	0	0	OTP Configuration loads at power-up
	User Writes Config 1 to the device				
3	**RegB = 0**	1	0	0	No change
4	**RegC = 1**	1	0	1	Register C was updated

		Register A	Register B	Register C	
Comparison	**Final Config**	1	0	1	
	Configuration 1	0	0	1	Register A is incorrect.

In steps 3 and 4, the Configuration 1 delta registers were based on a default configuration and not Configuration 0 versus Configuration 1. The tool did not explicitly know that it must change a non-default (Register A) value back to the default. Instead, it was generated with the expectation that Register A was already defaulted to 0 and did not need an update. This led to an incorrect configuration. The next section shows how to correctly update the configuration.

3. Correct Configuration Update

There are four options available for a correct configuration update.

3.1 Option 1: Start with a Blank Device

Avoid the issue described in Table 3 by starting with a blank device and then writing the whole configuration. If using an OTP configured device, there will be an accompanying Datasheet Addendum that shows how to set the GPIOs at power-up in order to load a blank configuration. If using an EEPROM, the configuration can exported as .hex file and then programmed into the EEPROM. Alternatively, the EEPROM can be erased or removed and the configuration can be directly written to the device using the serial port.

3.2 Option 2: Use the Timing Commander "Modified Since Last Saved" Feature

1. Download and install TC v1.16.4 or later.
2. Load an existing tcs file.
3. Update the configuration.
4. Open the "Export" dialog as shown in the following figure.

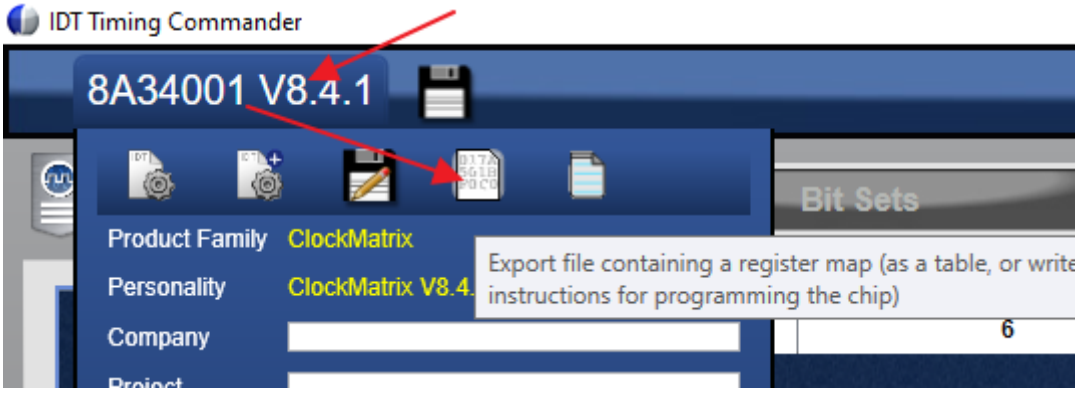

Figure 1. Export Writes Feature Access

5. Use "Browse" to select the target folder. Select the Format, the Protocol, I²C Address and the Address Type. Finally, select "Modified Since Save" as the Filter type.

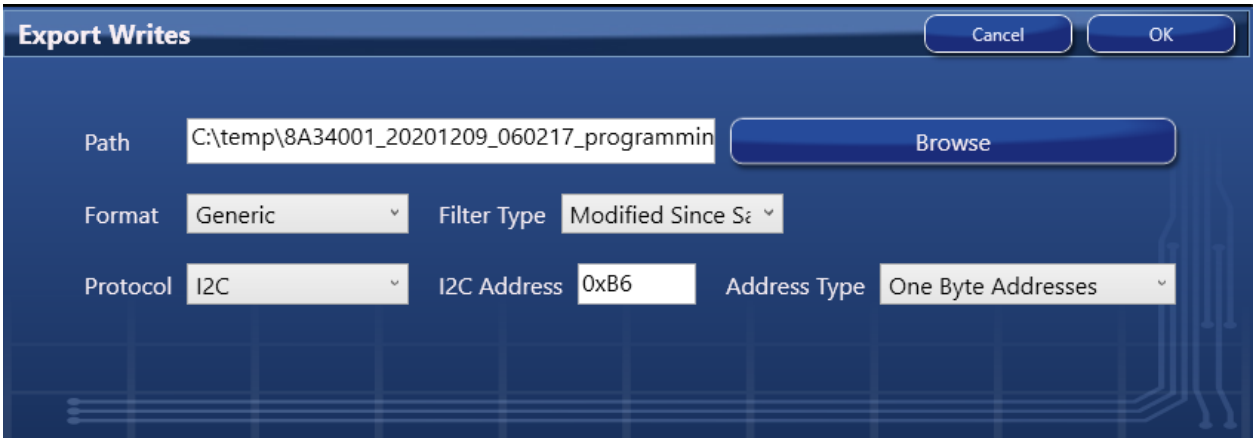

Figure 2. Export Writes Settings

6. Click “OK” to export the register changes into the file.
7. In the next dialog, click “Yes” to see a text version of the file or “No” to not view the file.

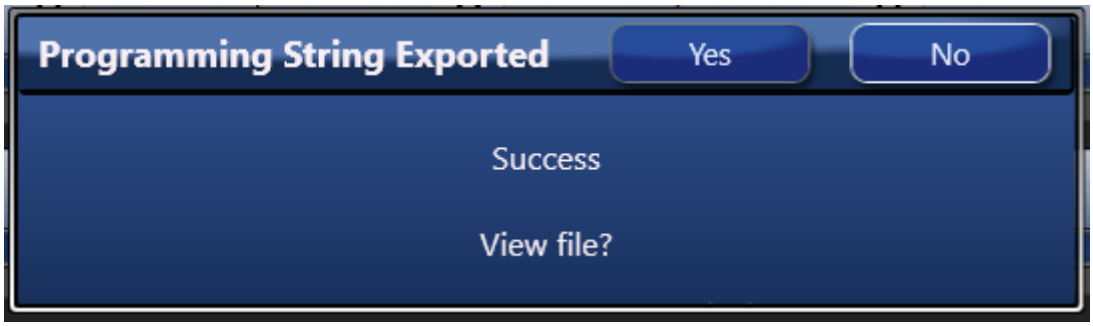

Figure 3. Export Writes File View Confirmation

8. Figure 4 shows the contents of the export file for a change in CLK0 from 25MHz to 125MHz with Format = Aardvark, Protocol = I2C, I2C Address = 0xB6, and One-Byte Addresses.

```
<aardvark>
<configure i2c="1" spi="0" gpio="0" tpower="1" pullups="1" />
<i2c_write addr="0xB6" radix="16">FC 00 C0 10 20</i2c_write>
<i2c_write addr="0xB6" radix="16">FC 00 81 10 20</i2c_write>
<i2c_write addr="0xB6" radix="16">FC 00 C1 10 20</i2c_write>
<i2c_write addr="0xB6" radix="16">B0 40 59 73 07 00 00</i2c_write>
<i2c_write addr="0xB6" radix="16">FC 00 CA 10 20</i2c_write>
<i2c_write addr="0xB6" radix="16">44 00 00 00</i2c_write>
<i2c_write addr="0xB6" radix="16">47 00</i2c_write>
<i2c_write addr="0xB6" radix="16">FC 00 C1 10 20</i2c_write>
<i2c_write addr="0xB6" radix="16">BD 01</i2c_write>
</aardvark>
```

Figure 4. Example Export Writes File

3.3 Option 3: Use Timing Commander Connected to a Device

Use a live Timing Commander connection to the device and allow the software to record the register changes. The following steps show how to capture a log of the serial port writes:

1. Load the initial configuration into the GUI (for more information, see the ClockMatrix GUI Step-by-Step User Guide). This is the configuration that is expected to be in the device due to a prior configuration through either OTP, the EEPROM, or a serial port update.
2. Connect to the device.
3. Open the Timing Commander Debug Window, as shown in Figure 5.

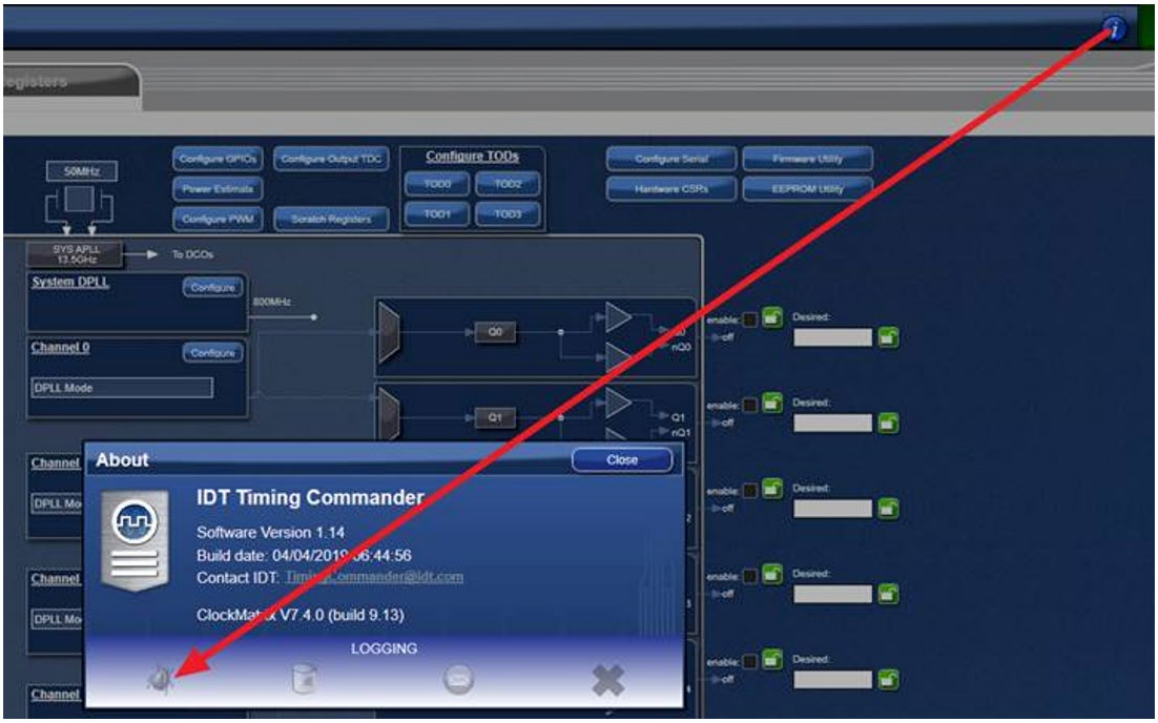

Figure 5. Open Timing Commander Debug Window

Figure 6 shows an excerpt of the log window. It contains the following tabs:

- IO: Input and Output transactions – Contains all the write and read transaction.
- IO (read): Read transactions – Contains only the read transactions.
- IO (write): Write transactions – Contains only the write transaction.

The Log Window also provides options to “Save” the log to a file, “Copy” the log to the Windows Clipboard, or “Clear” the log.

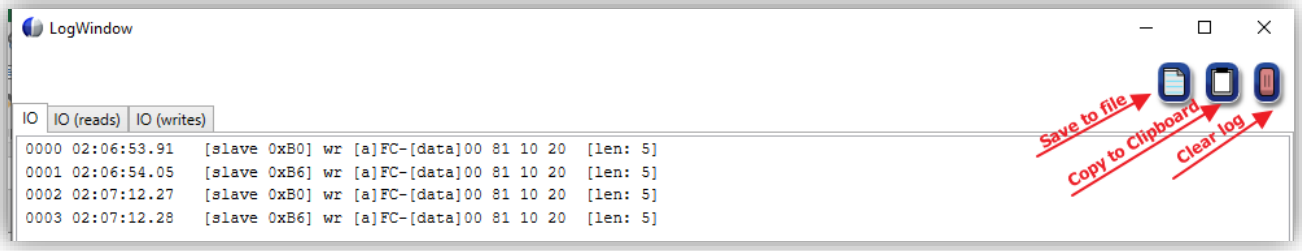

Figure 6. Log Window

4. Clear the log by pressing the red “Clear” button (see button in Figure 6).
5. Read register 0xC024, as shown in Figure 7. Select the “Registers” tab, Page “C0” and scroll down to register “0024” and press the “Read” button. The reason for this step is that it forces the GUI to a known Page Registers, and ensures updates will automatically include the Page Registers.

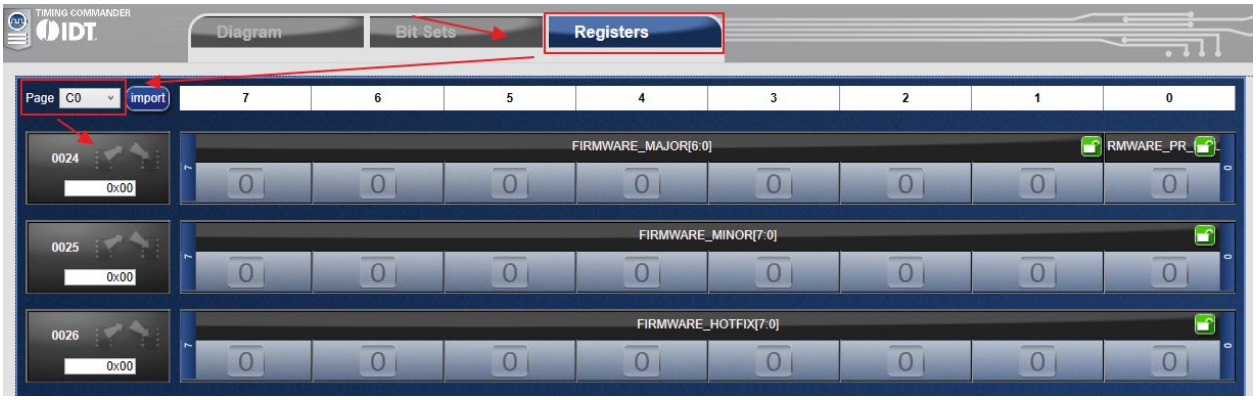

Figure 7. Read Register 0xC024

6. Make the changes to the configuration and use the Timing Commander log window to either Save or Copy the log to a file.

In step 6, the GUI now knows the register values that were used by the initial configuration and it captures the changes made by the user. An additional benefit for this procedure is that the GUI also captures the required Page Registers and Trigger registers without any extra effort by the user.

3.4 Option 4: Use a Script

Use a script to find the differences between the two .tcs files and generate a file containing the delta registers, the trigger registers, and the page registers. The follow steps provide a description of how to generate the register set:

1. Find the file differences. A software tool such as Windows Meld or TKDIFF can be used. A script can also be used. Record the registers that have different values. Save list as the “delta registers”.
2. Append the trigger registers to the list. For information on how to identify the trigger registers, see the ClockMatrix Trigger Register Appnote.
3. Parse the register list into the corresponding Serial Mode syntax, as described in 8A3xxxx Family Programming Guide (v4.8.7), section “Addressing Registers within a Device.”
4. Add Page Registers: for each new page, add the corresponding Page Register writes. For example, if writing to 0x**C0**FF and then 0x**C1**00, a Page Register update is required to change from Page 0xC0 to 0xC1. Details can be found in the same document and section as the previous step.

The Appendix contains pseudo code that provides the algorithm for script to accomplish steps 1 to 3 in Option 4.

4. Conclusion

Updating a device that already has a configuration must take into account the ***register delta*** between the two configurations, the ***page registers***, and the ***trigger registers***. There are four options to figure out how which registers to write: start with a blank device, using the Timing Commander Export Writes feature, use Timing Commander connected to a device or write a script.

For further assistance, please contact Applications Engineering team at IDT-Support-sync@lm.renesas.com.

5. Revision History

Revision	Date	Description
1.0	Dec.11.20	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 who are 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 to develop 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 from 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.

(Disclaimer Rev.1.01)

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-us/.

Trademarks

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

© 2025 Renesas Electronics Corporation. All rights reserved.