AN13044

How to enable EOnCE connection for FreeMASTER with MCUXpresso SDK and Config Tool on MC56F83xxx and MC56F81xxx

Rev. 0 — 11/2020

Application Note

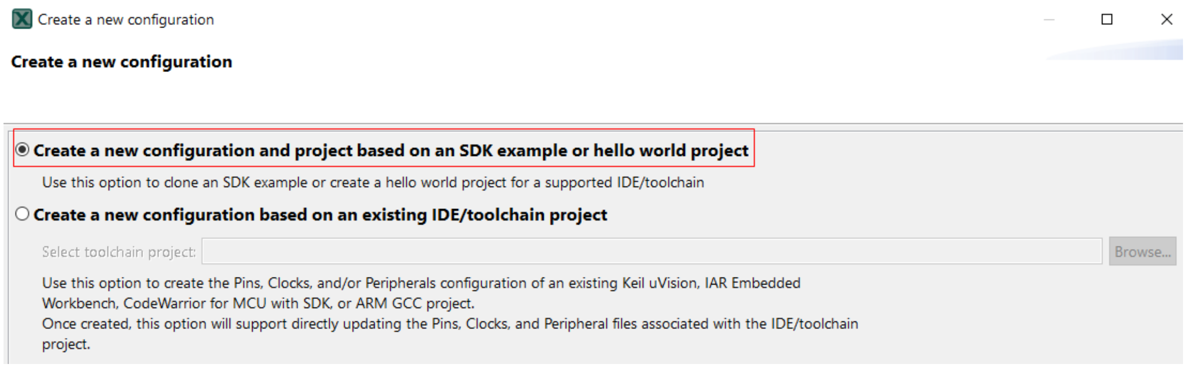
Contents

1 Introduction

The config tool v8.1 doesn't support Enhanced On-chip Emulator (EOnCE) for FreeMASTER communication interface directly, but FreeMASTER driver supports it. The config tool will support it in the coming Version 9. This article describes how to enable EOnCE for FreeMASTER with config tool v8.1 so that people can use MultiLink or OSBDM for FreeMASTER connection instead of serial port. This connection can be useful when there's no extra serial port on user's board.

2 Steps

1. Use config tool to clone the `project_template` project into a CodeWarrior workspace.
   a. Open config tool v8.1, click **File -> New...**, and choose **Create a new configuration and project based on an SDK example or hello world project**, as shown in Figure 1. Then, press **Next**.



Create a new configuration

● **Create a new configuration and project based on an SDK example or hello world project**

Use this option to clone an SDK example or create a hello world project for a supported IDE/toolchain

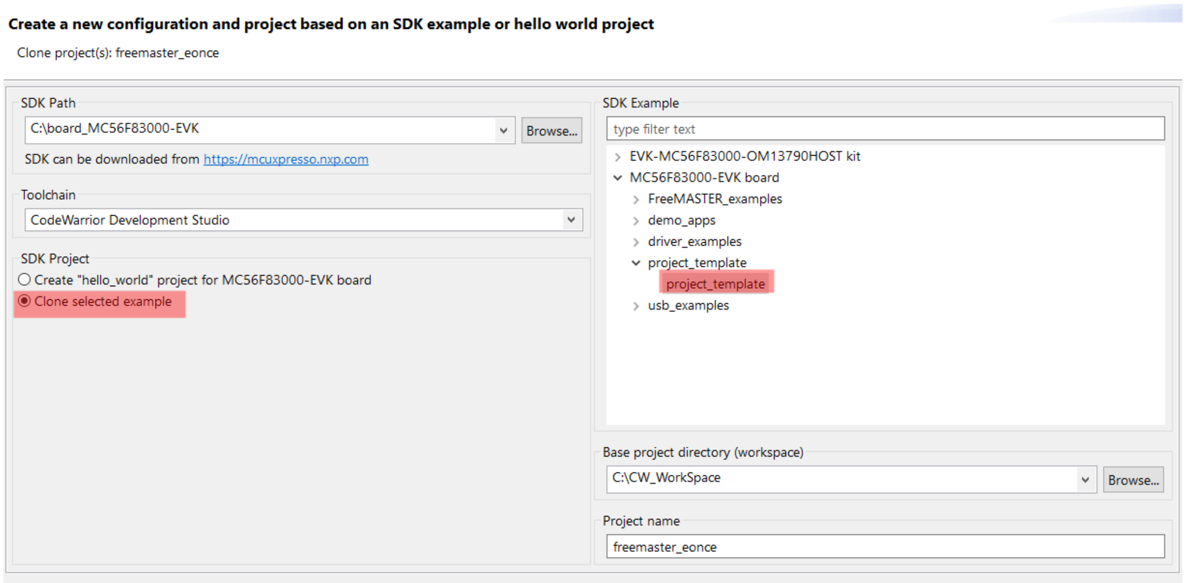
○ **Create a new configuration based on an existing IDE/toolchain project**

Select toolchain project:

Use this option to create the Pins, Clocks, and/or Peripherals configuration of an existing Keil uVision, IAR Embedded Workbench, CodeWarrior for MCU with SDK, or ARM GCC project.
Once created, this option will support directly updating the Pins, Clocks, and Peripheral files associated with the IDE/toolchain project.

Figure 1. Selecting Create a new configuration and project based on an SDK example or hello world project

   b. In the popped out window, select the correct SDK path. Select **Clone selected example** and choose **project_template**, as shown in Figure 2. Make up a proper CodeWarrior project name in the **Project name** textbox. Click **Finish**.



Create a new configuration and project based on an SDK example or hello world project

Clone project(s): freemaster_eonce

SDK Path: C:\board_MC56F83000-EVK

SDK can be downloaded from https://mcuxpresso.nxp.com

Toolchain: CodeWarrior Development Studio

SDK Project

○ Create "hello_world" project for MC56F83000-EVK board

● Clone selected example

SDK Example

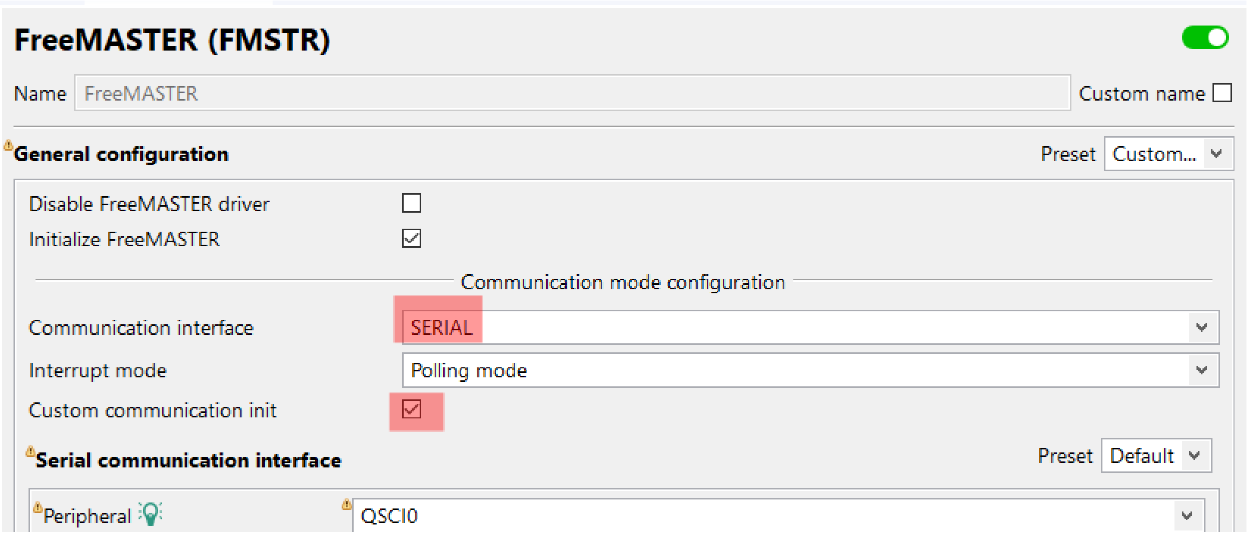
- EVK-MC56F83000-OM13790HOST kit
- MC56F83000-EVK board
  - FreeMASTER_examples
  - demo_apps
  - driver_examples
  - project_template
    - project_template
  - usb_examples

Base project directory (workspace): C:\CW_WorkSpace

Project name: freemaster_eonce

Figure 2. Selecting project_template

2. Add FreeMASTER component into Middleware panel, and make necessary settings as shown in Figure 3. Choose **SERIAL** as communication interface and check **Custom communication init**. The remaining FreeMASTER settings, such as, scope, recorder, TSA, can be configured as wanted. Click the **Update Code** button in the tool.



FreeMASTER (FMSTR)

Name: FreeMASTER   Custom name □

General configuration   Preset: Custom...

Disable FreeMASTER driver □

Initialize FreeMASTER ☑

Communication mode configuration

Communication interface: SERIAL

Interrupt mode: Polling mode

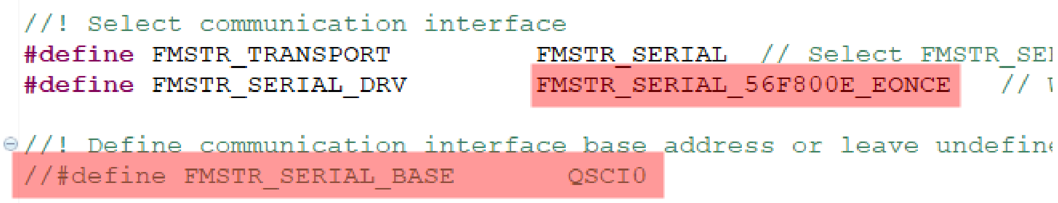
Custom communication init ☑

Serial communication interface   Preset: Default

Peripheral: QSCI0

Figure 3. FreeMaster settings

3. Import the cloned project into CodeWarrior. Open the **freemaster_cfg.h** file and make the following two modifications:
   a. Change **FMSTR_SERIAL_MCUX_QSCI** to **FMSTR_SERIAL_56F800E_EONCE**.
   b. Comment **FMSTR_SERIAL_BASE** definition.



```
//! Select communication interface
#define FMSTR_TRANSPORT           FMSTR_SERIAL   // Select FMSTR_SE
#define FMSTR_SERIAL_DRV          FMSTR_SERIAL_56F800E_EONCE    // \

//! Define communication interface base address or leave undefin
//#define FMSTR_SERIAL_BASE          QSCI0
```

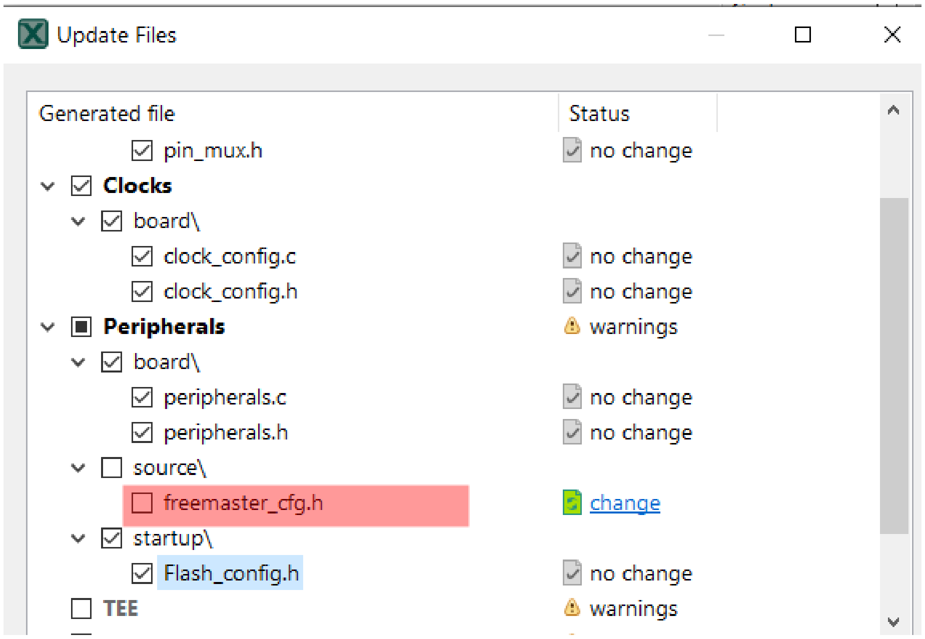
Figure 4. Modifying freemaster_cfg.h file

4. Now FreeMASTER has been configured to use EOnCE for communication. Add **FMSTR_Poll()** in `while(1)` of the `main.c` file, and FreeMASTER will work properly.

NOTE

Remember NOT to update FreeMASTER configurations through the tool ever since, because it will modify *freemaster_cfg.h* file.

Uncheck the checkbox highlighted in Figure 5, so that FreeMASTER configuration will not be updated.



Update Files

Generated file	Status
☑ pin_mux.h	no change
☑ **Clocks**	
☑ board\	
☑ clock_config.c	no change
☑ clock_config.h	no change
▣ **Peripherals**	warnings
☑ board\	
☑ peripherals.c	no change
☑ peripherals.h	no change
☐ source\	
☐ freemaster_cfg.h	change
☑ startup\	
☑ Flash_config.h	no change
☐ **TEE**	warnings

Figure 5. Unchecking freemaster_cfg.h file

By default, the EOnCE communication uses polling mode which requires `FMSTR_Poll()` to be called in the main application loop or in the main task. Refer to `fmstr_eonce` example which is available in SDK package for additional information about enabling interrupt-driven mode in EOnCE communication. Also see `fmstr_uart` and other SDK examples of using the FreeMASTER driver.

3 References

- Codewarrior IDE
- Freemaster
- MCUXpresso config tool
- MCUXpresso SDK

How To Reach Us

Home Page:

nxp.com

Web Support:

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: 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.

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 11/2020
Document identifier: AN13044