Application Note - Using the EEPROM on S08DZ60 microcontroller

Author:
Joel - MyFreescaleWebPage
http://myfreescalewebpage.free.fr

Last revision of this document: 1.1 of 2013-12-08.

Table of contents

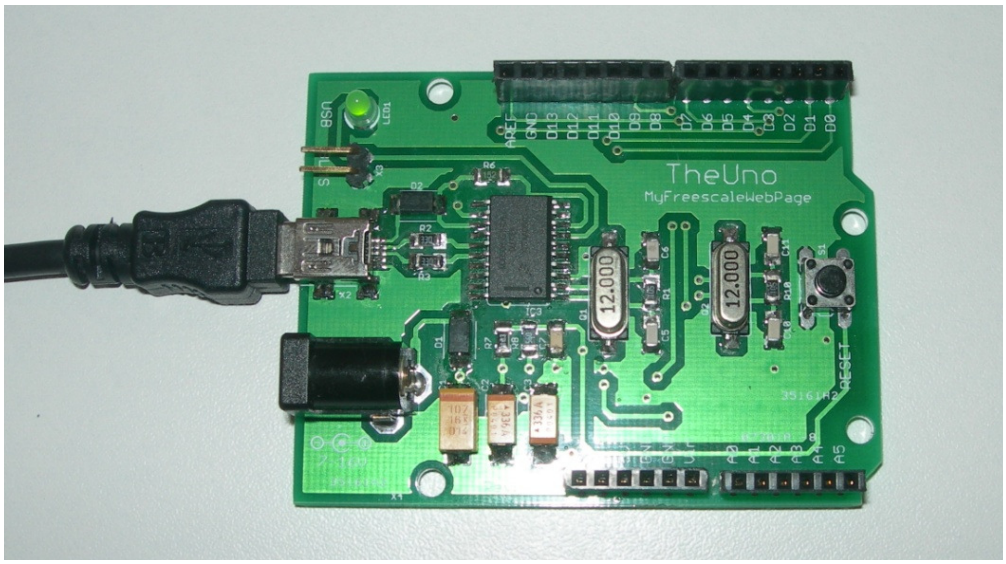
Revisions

Version	Date	Changes
1.0	2011-06-10	First release
1.1	2013-12-08	Port to CodeWarrior 10.5, rewrite the application note

Introduction

Some S08 microcontrollers from Freescale have an internal EEPROM which can be used to store some data by the application. This application note deals with the EEPROM and will show you how to read/write/erase the EEPROM on those microcontrollers.

This application note has been developed on TheUno, a development board created by MyFreescaleWebPage, based on S08DZ60 device and with a build-in open-source debugging cable. The S08DZ60 is one of the devices with an internal EEPROM.

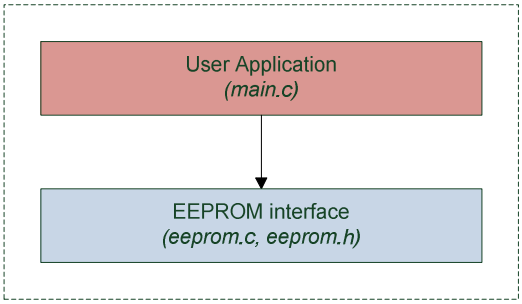


TheUno is compatible with almost every shields designed for the Arduino. It is possible to plug multiple shields simultaneously on the development board, with Ethernet, Wi-Fi, GPS, audio devices, H-Bridge, ... to simply create powerful applications using Freescale CodeWarrior, a nice development environment based on Eclipse. More details about TheUno available on http://myfreescalewebpage.free.fr.

1 Using the EEPROM

1.1 Software architecture

The software architecture is very simple. Two files "eeprom.c and "eeprom.h" are integrated to the project to access the EEPROM.



1.2 Accessing the EEPROM

The EEPROM header file must be included in the user application:

```
#include "eeprom.h"
```

The EEPROM interface contains several functions:
- EEPROM_Init() to initialize the EEPROM;
- EEPROM_Read() to read a single sector of the EEPROM;
- EEPROM_Write() to write a single sector of the EEPROM;
- EEPROM_Erase() to erase all the EEPROM.

The initialization function must be called on startup. Then, the application can read/write/erase the EEPROM when required.

The software attached to this document proposes several calls to the EEPROM functions to show how it is possible to use them. It also proposes to check the memory step by step in the debugger to see the result of the operations on the microcontroller.

2 Understanding how the EEPROM is used

This chapter focuses on the EEPROM functions to understand how the memory is accessed by the application.

2.1 Architecture of the EEPROM

On the S08DZ60 microcontroller, the EEPROM is made of two pages of 1024 bytes. Both are located at address 0x1400. A register is used to select which page is accessed at any time. The pages are made of 8-bytes sectors. Note that a second mode is also available, but I will not deals with it in this document.

This EEPROM interface proposed by this application note made a simple abstraction of the memory architecture by setting itself the right page. EEPROM functions are used like if the memory was only a single page of 2048 bytes starting at address 0x0000.

The EEPROM characteristics are set in "eeprom.h" and can be modified to match the specifications of your microcontroller.

2.2 Initialization

The initialization function selects the first page of the memory and the clock used by the EEPROM peripheral of the microcontroller.

The clock must be set between 150 and 200kHz. A divisor is used and must be set depending of the microcontroller bus clock frequency. This is register FCDIV which is described below.

DIV is the main divisor of the clock. PRDIV8 is an extra bit used to divide the clock by 8. Thus, the EEPROM clock frequency is:

- When PRDIV8 bit is not set:

$$F = \frac{Fbus}{DIV + 1}$$

- When PRDIV8 bit is set:

$$F = \frac{Fbus}{8 * (DIV + 1)}$$

In the application note, the bus frequency is 18MHz, so I have chosen to set the PRDIV8 bit and I have set DIV to 0x0B. This gives a clock frequency of 200kHz. You may have to change this configuration if your microcontroller is running at a different bus clock.

2.3 Reading

The EEPROM_Read() functions selects the right page of the EEPROM by setting the bit EPGSEL of the FCNFG register depending of the wanted memory address. Then it simply copy data from the EEPROM to the user application buffer.

2.4 Writing

Writing is possible only on an entire 8-bytes sector. The EEPROM_Write() function is used to write a single sector.

As already done to read the memory, the right page is selected depending where the application wants to store data. Then the sector is erased by setting FCMD register to Sector Erase Command value (0x40). This is required to write in the memory: it is not possible to write if the memory is not erased first.

Finally, the eight bytes of the sector are written by setting FCMD register to Burst Program Command value (0x25).

The functions checks for errors.

2.5 Erasing all the memory

The entire memory can be erased by calling EEPROM_Erase() function. This function simply invoke the mass erase command by setting FCMD to Mass Erase Command value (0x41).

Both pages of the EEPROM are erased.

2.6 Program and erase times

It is important to note that the accesses to the EEPROM are long. The following table indicates program and erase times at F = 200kHz.

Command	Time at F = 200kHz
Burst Command	20µs
Sector Erase	20ms
Mass Erase	100ms

Conclusion

This application note shows how to use the EEPROM on S08 devices. The sample code attached to this document can be used in your own applications.

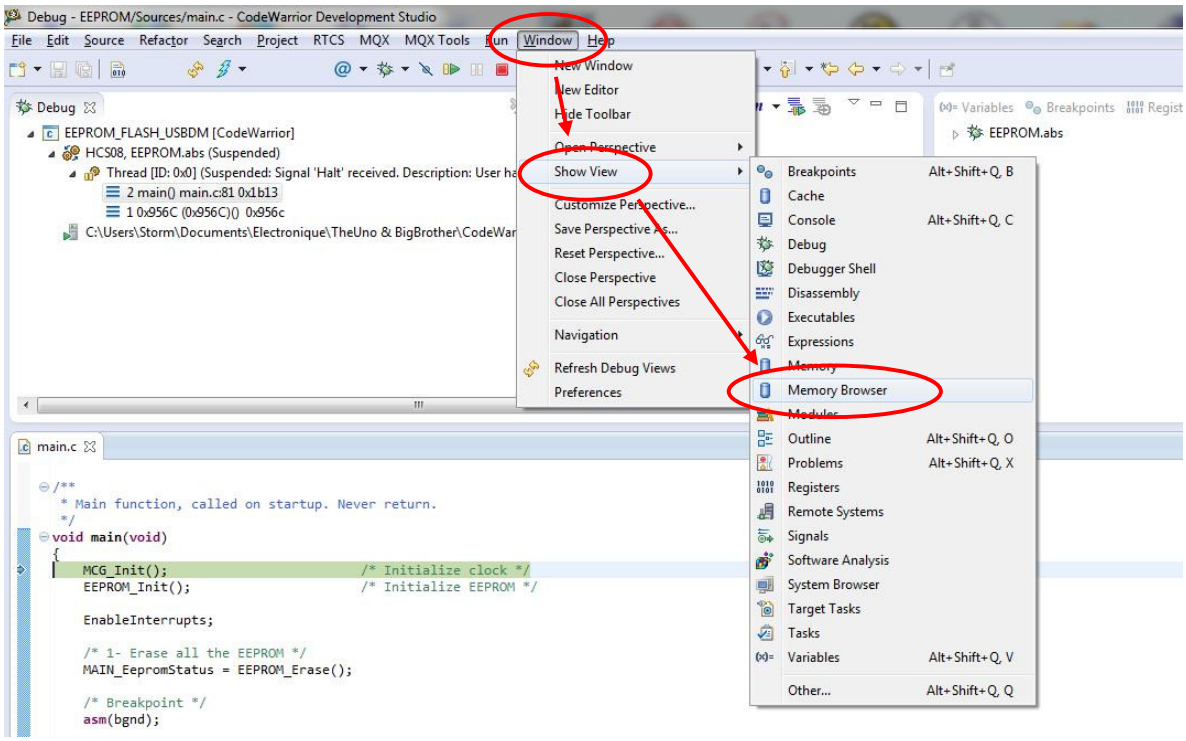
When using memories, it is important to know that the number of write/erase operations is limited. 100,000 operations are guaranteed on Freescale microcontrollers. Applications must be designed to limit the number of access to the memory in order to not burn it too quickly. You can evaluate the longevity of your application by checking how often you are programming/erasing the EEPROM.

You can refer to the datasheet of the microcontroller for more details on the EEPROM and its possibilities (security options, commands, etc.)

Appendix A - Looking to the memory of the microcontroller

The content of the memory can be spied using the tools integrated to Freescale CodeWarrior.

In the Debug perspective, first go to "Window" > "Show View" menu and choose "Memory Browser".



A new tab "Memory Browser" is available in the bottom part of the Freescale CodeWarrior window.

Enter the starting address you want to spy and press enter. Below I have written 0x1400 which is the address of the EEPROM and I'm currently running the application note application step by step. You can see the content of the memory at this time of the execution. Hexadecimal and ASCII modes are available.