Using Stellaris® Microcontrollers Internal Flash Memory to Emulate EEPROM

Application Note

AN01267-00

Copyright

Texas Instruments
108 Wild Basin, Suite 350
Austin, TX 78746
Main: +1-512-279-8800
Fax: +1-512-279-8879
http://www.luminarymicro.com

Table of Contents

Introduction

Electronically erasable programmable read-only memory (EEPROM) is ideal for applications that require the ability to write, read, and update variables in non-volatile memory. Although Stellaris® microcontrollers do not have internal EEPROM, the internal Flash memory in Stellaris microcontrollers can provide this functionality using EEPROM emulation. A system designer could use external serial EEPROM with Stellaris microcontrollers, but this solution is not ideal for cost-sensitive or pin-constrained applications. Emulating EEPROM using the internal Flash memory is a better solution and is described in detail in this application note. In addition, the EEPROM emulation drivers and an example application that makes use of these drivers are available for download from the www.luminarymicro.com web site. The sample code described in this application note can be downloaded from http://www.ti.com/lit/zip/SPMA030.

EEPROM Emulation Overview

EEPROM is a type of non-volatile memory that is used to store variables whose values need to be preserved when power is removed. EEPROM allows the values of these variables to be individually erased and written.

Flash memory allows variables to be individually written (changing bits from a 1 to a 0), but cannot be modified until the variable has been erased (changing bits from a 0 to a 1). In addition, Flash memory has the restriction that the memory is erased by a region, typically known as a page. These pages are much larger than the size of the variables that are stored in the memory. The page size on Stellaris microcontrollers is 1 K bytes. In addition to their rather large size, these Flash pages can only endure a limited number of program/erase cycles.

Given the above constraints, EEPROM emulation schemes using Flash memory have been developed. A typical emulation scheme involves using a portion of the Flash memory and dividing it into multiple EEPROM pages. The most recent data is stored in one of the pages, known as the active page, using entries which are comprised of an identifier (similar to an address) and the data. For reads, the software starts from the end of the active page and searches for the identifier and then returns the data associated with that identifier. For writes, the identifier and data are written to the active page in the next available entry. Once the active page is full, the most recent data for each identifier is copied to the next page which then becomes the active page. The full page is then available to be erased and reused as needed. The pages are used like a circular buffer to store the data.

EEPROM Emulation Implementation

Accompanying this application note is a set of EEPROM emulation drivers that use the internal Flash of the Stellaris family of microcontrollers to emulate real EEPROM. These drivers provide the user the ability to read, write, and clear the EEPROM contents. Using the drivers, the user specifies the region of Flash to be used for EEPROM emulation. In addition, the user specifies the size of the individual EEPROM pages within the emulation region which, in effect, also specifies the number of EEPROM pages within the region.

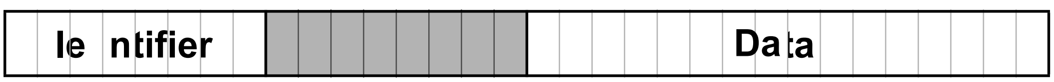
The beginning of each EEPROM page consists of two 32-bit status words used by the emulation software (see Table 1). These status words are used to distinguish between completely erased pages, the active page, and used pages. If both the first and second status words are in the erased state, then it is assumed that this page is completely erased. If the first status word is not erased and the second status word is erased, then the page is assumed to be in the active state. If both the first and second status words are not in the erased state, then the page is assumed to be in the used state.

Table 1. Status Word Relation to EEPROM Page State

State of First Status Word	State of Second Status Word	State of EEPROM Page
Erased	Erased	Completely Erased
Erased	Not Erased	Invalid state
Not Erased	Erased	Active
Not Erased	Not Erased	Used

The remainder of the EEPROM page (total page size minus the two status words) is used for storing the EEPROM entries. The Stellaris Flash has the restriction that each 32-bit word can only be programmed once between erase operations. Due to this restriction, each EEPROM entry is 32-bits. Each entry consists of an 8-bit identifier and 16-bits of data (see Figure 1). The number of entries available in an EEPROM page is ((PageSize / 4) – 2 status words). For a 1 KB page size, there are 254 entries.

Figure 1. EEPROM Page Entry



An EEPROM write operation causes the next available entry in the currently active page to be written with the given identifier and data. When the currently active EEPROM page becomes full, the next page in the emulation region is erased and the most recent data for each identifier is copied to the new page. The new page is then marked as the active page and the full page is marked as used.

The EEPROM read operation is implemented by reading the currently active EEPROM page searching for the given identifier and returning the data associated with that identifier if found. The search starts at the entry just before the next available entry and traverses the page in reverse. Therefore, the first instance of data for that identifier is the most recent.

An EEPROM clear operation erases the EEPROM contents by marking the currently active page as used, erasing the next page in the emulation region, and marking the erased page as the active page.

Accompanying Software

The software that accompanies this application note consists of the EEPROM emulation drivers as well as a simple example application that uses the drivers.

EEPROM Emulation Drivers

The EEPROM emulation drivers provide the ability to read, write, and clear the EEPROM contents. The application programmer's interface (API) consists of the following four functions:

- `SoftEEPROMInit`
  ```
  long SoftEEPROMInit(unsigned long ulStart, unsigned long ulEnd,
  unsigned long ulSize);
  ```

  SoftEEPROMInit initializes the EEPROM emulation region within the Flash. This function must be called prior to using any of the other functions in the API. The user must specify the start address of the EEPROM emulation region, the end address of the EEPROM region, and the size of the EEPROM pages within the EEPROM region. A value of 0 is returned if the initialization is successful. A non-zero value indicates a failure.

- `SoftEEPROMWrite`
  ```
  long SoftEEPROMWrite(unsigned short usID, unsigned short usData);
  ```

  SoftEEPROMWrite writes the user specified identifier and data to the next available entry in the currently active EEPROM page. If the page is full, a page swap occurs. A value of 0 is returned if the write is successful. A non-zero value indicates a failure.

- `SoftEEPROMRead`
  ```
  long SoftEEPROMRead(unsigned char usID, unsigned short *pusData,
  tBoolean *pbFound);
  ```

  SoftEEPROMRead reads the most recent data associated with the specified identifier. The user must provide a pointer to a variable to store the data in and a pointer to a variable to store a value which indicates whether or not the identifier was found. A value of 0 is returned if the read is successful. A non-zero value indicates a failure.

- `SoftEEPROMClear`
  ```
  long SoftEEPROMClear(void);
  ```

  SoftEEPROMClear erases the EEPROM contents. A value of 0 is returned if the clear is successful. A non-zero value indicates a failure.

For full details on the EEPROM emulation drivers, see the software reference manual provided with the software.

EEPROM Emulation Timing

The tables below show the timing parameters associated with the EEPROM emulation drivers for 1-KB and 2-KB EEPROM page sizes. A 50-MHz system clock is used for all of the measurements. Data was collected from the LM3S811, LM3S6965, and LM3S9B90 evaluation boards. The timing for the LM3S811 was taken using Rev C2 silicon and is representative of the remainder of the Sandstorm class of microcontrollers. The timing for the LM3S6965 was taken using Rev A2 silicon and is representative of the remainder of the Fury class of microcontrollers. The timing for the LM3S9B90 was taken using Rev B1 silicon and is representative of the remainder of the Tempest class of microcontrollers. Timings may vary some due to slight variations in program and erase times for each individual part.

Below is a list of the EEPROM operations measured and an explanation of each:

- EEPROM Write Operation (Normal – no page swap) – A write operation which does not trigger a page swap.

- EEPROM Write Operation (with page swap) – A write operation which triggers a page swap. The minimum value applies when there is only one EEPROM identifier being used and therefore only one entry that has to be copied to the next page. The maximum value applies when the maximum number of identifiers (without exceeding the identifier limit of 255) must be copied to the next page.
- EEPROM Read Operation – A read operation to the EEPROM. The minimum value applies when the first entry read contains the requested identifier. The maximum value applies when the last possible entry read in the page contains the first instance of the requested identifier.
- EEPROM Clear Operation – A clear operation of the EEPROM. The contents of the EEPROM are erased.

Table 2. EEPROM Timing on LM3S811 Microcontroller – 1-K EEPROM Page Size

EEPROM Operation	Min	Nom	Max	Unit
EEPROM Write Operation (Normal – no page swap)	-	41.5	-	µS
EEPROM Write Operation (with page swap)[a]	16.6	-	26.3	mS
EEPROM Read Operation[b]	1.7	-	82.66	µS
EEPROM Clear Operation	-	16.4	-	mS

a.The amount of time required for a write operation with a page swap depends on the number of unique identifiers with data to be copied to the new page.
b.The amount of time required for a read operation depends on where the identifier being searched for is located in the active EEPROM page.

Table 3. EEPROM Timing on LM3S811 Microcontroller – 2-K EEPROM Page Size

EEPROM Operation	Min	Nom	Max	Unit
EEPROM Write Operation (Normal – no page swap)	-	41.5	-	µS
EEPROM Write Operation (with page swap)[a]	33.1	-	42.9[b]	mS
EEPROM Read Operation[c]	1.7	-	164.6	µS
EEPROM Clear Operation	-	32.7	-	mS

a.The amount of time required for a write operation with a page swap depends on the number of unique identifiers with data to be copied to the new page.
b.The maximum time for this operation is not double that of the 1-KB page size case due to the limit of 255 identifiers and the 510 entries for a 2-KB EEPROM page.
c.The amount of time required for a read operation depends on where the identifier being searched for is located in the active EEPROM page.

Table 4. EEPROM Timing on LM3S6965 Microcontroller – 1-K EEPROM Page Size

EEPROM Operation	Min	Nom	Max	Unit
EEPROM Write Operation (Normal – no page swap)	-	54.4	-	µS

Table 4. EEPROM Timing on LM3S6965 Microcontroller – 1-K EEPROM Page Size

EEPROM Operation	Min	Nom	Max	Unit
EEPROM Write Operation (with page swap)[a]	22.4	-	35.4	mS
EEPROM Read Operation[b]	1.7	-	82.7	µS
EEPROM Clear Operation	-	22.1	-	mS

a.The amount of time required for a write operation with a page swap depends on the number of unique identifiers with data to be copied to the new page.
b.The amount of time required for a read operation depends on where the identifier being searched for is located in the active EEPROM page.

Table 5. EEPROM Timing on LM3S6965 Microcontroller – 2-K EEPROM Page Size

EEPROM Operation	Min	Nom	Max	Unit
EEPROM Write Operation (Normal – no page swap)	-	54.4	-	µS
EEPROM Write Operation (with page swap)[a]	44.6	-	57.6[b]	mS
EEPROM Read Operation[c]	1.7	-	164.6	µS
EEPROM Clear Operation	-	44.1	-	mS

a.The amount of time required for a write operation with a page swap depends on the number of unique identifiers with data to be copied to the new page.
b.The maximum time for this operation is not double that of the 1-KB page size case due to the limit of 255 identifiers and the 510 entries for a 2-KB EEPROM page.
c.The amount of time required for a read operation depends on where the identifier being searched for is located in the active EEPROM page.

Table 6. EEPROM Timing on LM3S9B90 Microcontroller – 1-K EEPROM Page Size

EEPROM Operation	Min	Nom	Max	Unit
EEPROM Write Operation (Normal – no page swap)	-	310.2	-	µS
EEPROM Write Operation (with page swap)[a]	18.9	-	45.9	mS
EEPROM Read Operation[b]	1.7	-	82.7	µS
EEPROM Clear Operation	-	17.8	-	mS

a.The amount of time required for a write operation with a page swap depends on the number of unique identifiers with data to be copied to the new page.
b.The amount of time required for a read operation depends on where the identifier being searched for is located in the active EEPROM page.

Table 7. EEPROM Timing on LM3S9B90 – 2-K EEPROM Page Size

EEPROM Operation	Min	Nom	Max	Unit
EEPROM Write Operation (Normal – no page swap)	-	310.2	-	µS
EEPROM Write Operation (with page swap)[a]	35.1	-	56.8[b]	mS
EEPROM Read Operation[c]	1.7	-	164.6	µS
EEPROM Clear Operation	-	33.3	-	mS

a.The amount of time required for a write operation with a page swap depends on the number of unique identifiers with data to be copied to the new page.
b.The maximum time for this operation is not double that of the 1-KB page size case due to the limit of 255 identifiers and the 510 entries for a 2-KB EEPROM page.
c.The amount of time required for a read operation depends on where the identifier being searched for is located in the active EEPROM page.

Example Application

The application provided with the EEPROM emulation drivers is an example that uses UART0 to interface to a PC COM port and allows the user to read and write the emulated EEPROM variables. In addition, the user can dump and clear the emulated EEPROM contents. The example is configured for a board with an 8-MHz crystal. The appropriate parameter in the call to `SYSCTLCLOCKSET()` must be changed if a different crystal is used on the board.

To run the example:

1. Start Hyperterminal.
2. Select the COM port associated with the board.
3. Select 115200 bits per second.
4. Select 8 data bits.
5. Select no parity.
6. Select 1 stop bit.
7. Select no flow control.

At the command prompt, type “help” to see the list of commands supported.

Conclusion

The use of external EEPROM consumes pins that could be used for other purposes, increases bill-of-material costs, and takes up board space. To overcome these issues, EEPROM can be emulated using internal Flash memory. Texas Instruments provides a set of EEPROM emulation drivers for the Stellaris family as well as an example application that makes use of these drivers.

References

The following documents and source code are available for download at www.luminarymicro.com:

- *Stellaris LM3S811 microcontroller data sheet*, Publication Number DS-LM3S811
- *Stellaris LM3S6965 microcontroller data sheet*, Publication Number DS-LM3S6965
- *Stellaris LM3S9B95 microcontroller data sheet*, Publication Number DS-LM3S9B95
- Source code for application note AN01267 - *Using Stellaris® Microcontrollers Internal Flash Memory to Emulate EEPROM*

Important Notice

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DLP® Products	www.dlp.com	Broadband	www.ti.com/broadband
DSP	dsp.ti.com	Digital Control	www.ti.com/digitalcontrol
Clocks and Timers	www.ti.com/clocks	Medical	www.ti.com/medical
Interface	interface.ti.com	Military	www.ti.com/military
Logic	logic.ti.com	Optical Networking	www.ti.com/opticalnetwork
Power Mgmt	power.ti.com	Security	www.ti.com/security
Microcontrollers	microcontroller.ti.com	Telephony	www.ti.com/telephony
RFID	www.ti-rfid.com	Video & Imaging	www.ti.com/video
RF/IF and ZigBee® Solutions	www.ti.com/lprf	Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

IMPORTANT NOTICE AND DISCLAIMER