AN1449

High-Reliability and High-Frequency EEPROM Counter

Author: *Cobus Van Eeden*
Microchip Technology Inc.

INTRODUCTION

A common problem with storing counter values to EEPROM is that the maximum count achievable will be restricted by the endurance of the Least Significant Byte (LSB) in the counter. This value on a typical EEPROM would be in the order of 100,000 counts.

When the application requires counts in excess of this number, it becomes necessary to spread the load of the counter updates over several EEPROM bytes in order to achieve the highest count possible, regardless of errors caused by the reliability of the EEPROM.

On an EEPROM, where the cell endurance is 100K writes, it is theoretically possible to increase the achievable maximum count to 100K, multiplied by the number of bytes used by the counter.

OVERVIEW

This library implements a number of strategies to ensure that a reliable counter is implemented with the maximum effective counter value achievable, given the available EEPROM space for the counter.

These strategies can be summarized as follows:

1. Use of an error-correcting code (Hamming code) to identify cell failure and correct single-bit errors.
2. Use of Gray coding to ensure that any increment in the counter will only change one byte in the EEPROM.
3. Optionally store a redundant copy of the counter in order to allow recovery from multi-bit cell failures.
4. Immediately read back any data written to verify successful writes.
5. Storing of a byte-map to keep track of cell failures and allow relocation of failed counter bytes to fresh locations, regardless of the order of cell failures.

IMPLEMENTATION DETAILS

Error Correcting Code

All data is stored using a (8,4) Hamming code. Using this code, the user will be able to correct any single-bit error and detect any 2-bit error. Additionally, it allows the detection of the majority of errors comprising more than two bits.

To avoid common EEPROM cell failure modes, the Hamming code is selected to avoid using cell values of `0xFF` and `0x00`.

A (8,4) Hamming code uses eight bits, as shown in Table 1.

TABLE 1: HAMMING CODE

Data	Code	Data	Code
0	0x80	8	0x34
1	0x07	9	0x4C
2	0x19	10	0xAD
3	0x01	11	0xD5
4	0x2A	12	0x9E
5	0x52	13	0xE6
6	0xB3	14	0x0F
7	0xCB	15	0x7F

Gray Coding

In order to ensure that single increments of the counter do not require multiple EEPROM cells to be updated, a Gray code is used to store the counter value.

A typical implementation will use 32 bits to store a counter. By converting this code to a simple 32-bit Gray coded value, only a single bit will change when the counter is incremented. This means, even with the byte-level Hamming code implemented, only a single byte/EEPROM cell will be changed for every counter increment.

The Gray code is simply generated by performing an XOR of the counter value with the same value right-shifted by one bit. This is a common and very efficient algorithm to construct a standard cyclic Gray code. This type of Gray code is known as a reflected Gray code and has the property that the LSB transitions every second count, the next bit transitions every fourth count, the third Least Significant bit (LSb) transitions every eighth count, and so forth.

Redundancy

In applications where it is critical that, when multi-bit failures of the EEPROM occur and the counter value should be retained, it is important to store the counter value more than once (add redundancy) in order to allow recovery from catastrophic cell failures.

Detecting multi-bit failures is good and well, but when this happens, the user only knows that the data being read back is corrupted, leaving the user with no way to reconstruct the correct counter value.

Furthermore, due to the nature of the EEPROM cell technology, it is possible for a cell to have valid data immediately after writing to it, but fail when reading it after a longer period of time (e.g., hours or days later). This means that simply reading back the data after a write will not ensure that the byte will still read correctly after a period of time.

Accordingly, redundant storage of the counter (duplicating the data in EEPROM) is the only reliable method to ensure the counter value is not lost when an EEPROM byte fails. Redundant data will also allow for correction of data corrupted or lost when the programming process is interrupted by loss of power.

Corruption is not always caused by the EEPROM cell reaching its endurance limit. Power failures during write cycles is an example of this. The library will always attempt to re-write the correct data to a location which is found to have been corrupted, before marking the EEPROM location bad. This ensures that bytes are only discarded if the corruption was caused by a cell reaching its endurance limit.

Relocation MAP

In order to keep track of individual cell failures, it is necessary to keep a map of where in memory every part of the counter is currently located. In order to determine if the map has been initialized, the first two bytes of storage are dedicated to a magic number which, if set correctly, indicates an initialized map.

As the Gray code used in this application note is a reflected Gray code, not a balanced Gray code, the LSB will transition most often. This means significantly more failures are expected in the lower bytes of the counter than in the higher ones.

As an optimization, the location of each byte of the counter is stored using eight bits of data for the upper five nibbles of the counter (classified as type High) and two bytes of data for the three Least Significant nibbles (classified as type Low), which is expected to fail most often.

In order to maintain this balance, the map allocates MSBs from the lowest memory address (addressable with an 8-bit address) and LSBs from the highest address, addressable with 16 bits. This will allow the High type nibbles to be relocated to any memory address lower than 256 only, but the Low type nibbles can be relocated to any location up to 64K.

The relocation map is initialized with pointers to the initial locations in EEPROM of each corresponding data byte. The Low bytes will contain addresses starting at the highest address, while the High bytes will start from the top of memory and grow downward in address.

Each EEPROM location stores one nibble of the counter using a Hamming (8,4) code, and thus needs eight bytes to store a 32-bit counter. Also, since this is done redundantly, at least 22 bytes of storage are needed for the memory map.

Once a cell fails, it is relocated by selecting the next available address. As the available space is used sequentially, the next available space can be easily calculated.

The relocation map is implemented, as shown in Table 2.

TABLE 2: RELOCATION MAP

EEPROM MAP			
Magic Number	**Byte Number**	**Type**	**Storage**
	0	N/A	1 Byte
	1	N/A	1 Byte
Copy 1	Byte Number	Type	Storage
	0	Low	16 bits
	1	Low	16 bits
	2	Low	16 bits
	3	High	8 bits
	4	High	8 bits
	5	High	8 bits
	6	High	8 bits
	7	High	8 bits
Copy 2	Byte Number	Type	Storage
	0	Low	16 bits
	1	Low	16 bits
	2	Low	16 bits
	3	High	8 bits
	4	High	8 bits
	5	High	8 bits
	6	High	8 bits
	7	High	8 bits
		Total	**24 Bytes**

Algorithm for Reliable Counter Recovery

The counter is primarily stored in RAM and written to EEPROM on every increment in order to be recoverable in the event of a power failure.

Accordingly, the following algorithm is followed:

1. At power-up the counter is read from EEPROM and checked for any bit errors.
2. A 2-byte magic number stored at the beginning of the map is used to indicate that a counter is present. If not, the counter is initialized to 0, and the map is initialized according to the configuration.
3. Single-bit errors are corrected and the cells where errors were detected are relocated for safety.
4. Multi-bit errors are detected.
5. On a multi-bit error, an attempt is made to recover the corresponding byte from the alternative storage location.
6. If the second location also fails with multiple bit errors at the same time the system fails. The probability of this occurring should be very remote.
7. On every timer increment, the value is written to both the primary and redundant storage locations.
8. Directly after writing the counter, the values are read back in order to verify a successful write.
9. If the read fails at this point, the byte is abandoned and the storage is relocated to a new byte for the corresponding counter byte.

Resulting Endurance of Counter

This configuration allows for up to 4096 bytes of EEPROM to be used to represent the counter, which with redundancy included, will allow approximately 2036x times the minimum single-cell endurance for the counter as a whole. With a minimum cell endurance of 100K cycles, this would allow the counter to reach at least 203.6 million before the system fails.

As the typical endurance is usually an order of magnitude higher than the minimum specified, and using a statistically significant number of bytes to store the counter, the average life expectancy per byte should approach the typical lifetime rather than the minimum. It is, thus, likely to achieve counts in excess of two billion using this configuration on a part with 4096 bytes of EEPROM, or counts in excess of 500 million using 1024 bytes of EEPROM.

For a typical application incrementing the counter twice per second (500 million counts), this will yield an operating time of 70,000 hours (eight years at 24/7 usage), using 1 kB of EEPROM (assuming that, on average, every byte will fail when it reaches one million cycles.)

NOTES

1. Most Microchip EEPROM parts specify a minimum of 100K endurance up to 85ºC. Temperature has a large effect on endurance and, thus, a device specified for 100K endurance up to 85ºC will most likely achieve the typical one million writes or more at room temperature.
2. Care should be taken when using address '0' in the EEPROM for storing sensitive data. At start-up, registers EEADRL and EEADRH are both set to '0', which leaves them referencing the EEPROM cell at address '0'. If a condition occurs where a write is accidentally initiated from software, this will consequently happen at address '0'. Although this is not very likely, it does make address '0' the most likely location to overwrite accidentally and, for this reason, it is recommended that sensitive data, such as the map and magic numbers of the counter library, should not be located at this address.
3. The write sequence of an EEPROM byte consists of two steps. First, the entire byte is erased, which will change it to either 0xFF or 0x00, depending on the specific implementation and technology used. Subsequently, the new value will be written to the location. The erase step in this process accounts for the bulk of the time to completion. The consequence of this is that, if a write is interrupted by the loss of power on the device, it is much more likely it will happen during the erase cycle, which would most likely leave the byte in a partially erased, or fully erased state when power is returned. Of course, erasing a byte will most likely result in a multi-bit error, which will not be recoverable without redundancy (storing the byte a second time at an alternate location). If your application can potentially be disconnected from power unexpectedly, it is highly recommended that the redundant storage be implemented in order to recover from interrupted writes caused by power loss.
4. In order to prevent EEPROM bytes from being discarded due to an interrupted write, the recovery sequence will always attempt to re-write a EEPROM byte and read back the data to verify it before discarding the byte and relocating the storage to a new location.

CONCLUSION

When storing a counter to EEPROM, the LSB changes on every increment of the counter. This results in the EEPROM cell storing the LSB failing at the cell endurance specification. Typically, this will yield 100,000 write cycles, which will impose a maximum number to which the counter can reliably be set.

This application note shows that spreading the storage of a typical 32-bit counter over multiple bytes and by adding redundancy and error correction, it is possible to increase the maximum value to which such a counter can reliably be incremented to over four billion.

This library represents a good basis from which a reliable high value counter can be implemented. The library can be modified according to individual requirements. Possible improvements or adjustments, which were considered, but not implemented, are listed in Table 3 below.

TABLE 3: IMPROVEMENTS FOR CONSIDERATION

	Modification	Motivation
1	Store EEPROM map using a Hamming code.	This would consume more EEPROM storage but would increase reliability.
2	Discard EEPROM cell only after multiple bit failures.	The current implementation discards a cell (byte) on any bit failure. Since the Hamming code can correct single-bit errors and the redundant storage allows recovery, usage can be extended to the first multi-bit failure.
3	Abandon the second copy to achieve higher counts.	The second copy adding redundancy can be omitted, especially if the count required does not have to be precise. As failures are most likely to occur in the LSB, recovery from catastrophic failure (multiple bit error) by setting the LSB to eight may be acceptable (resulting in an average error of only eight counts per failure)
4	Store EEPROM map in Flash memory.	The EEPROM map is modified at most as many times as you have EEPROM bytes available. This would typically be 4096 times or less. On devices that can self-program, it may be attractive to store the EEPROM map in the Flash memory instead of EEPROM.

APPENDIX A: LIBRARY API

Configuration

Some configuration is required in order to tell the library where to locate the data in EEPROM and how many bytes to use for the counter storage. This is achieved by using the following Macro definitions (e.g., to allocate 127 bytes to the counter).

TABLE 4:

Macro Name	Example	Description
EEPROM_MAP_LOCATION	1	Locates the 24 bytes of map starting at Address 1
COUNTER_EEPROM_START	25	Start of counter data EEPROM storage
COUNTER_EEPROM_END	128	End of counter data EEPROM storage

Storage of the counter in RAM and the EEPROM map in RAM also needs to be provided. This is achieved by declaring the following variables as global symbols in the program:

```
// Create 2 maps, one for base and redundant storage
volatile eeprom_map EepromMap[2];
// Create storage in RAM for Counter and init to 0
unsigned long int Counter = 0;
```

Consequently, the library consumes 26 bytes of RAM. Code space required depends on compiler settings and device used (approximately 2000 words of program space).

Interface

To use the library, only two public functions need to be called. These functions are described below.

`unsigned char eepromMapInit(void)`

This method checks the magic number. If it has been set correctly the user must have a stored counter value and it will recover the counter value from the EEPROM storage. If the magic number is not set, this method will initialize the counter to zero and set up the map for initial use.

This method returns success (1), if a counter value was successfully recovered, and failure (0), if the map had not been initialized before, and the counter was reset to zero.

`unsigned char writeCounters(unsigned long int c)`

This method persists the counter value that is supplied as a parameter redundantly to the EEPROM. It internally remaps the storage locations based on any single bit failure detected upon writing. It returns an error (0) if a failure is detected and no more storage is available.

Debugging

In Debug mode the library will print any relocation events or errors to `stdout` when they happen. When including *stdio.h* with the HI-TECH C® compiler, `stdout` is implemented via the putch method, which the user must implement to print one character at a time to the desired location. The example project redirects this to the serial port on the device.

NOTES:

APPENDIX B: CODE EXAMPLE

Software License Agreement

The software supplied herewith by Microchip Technology Incorporated (the “Company”) is intended and supplied to you, the Company’s customer, for use solely and exclusively with products manufactured by the Company.

The software is owned by the Company and/or its supplier, and is protected under applicable copyright laws. All rights are reserved. Any use in violation of the foregoing restrictions may subject the user to criminal sanctions under applicable laws, as well as to civil liability for the breach of the terms and conditions of this license.

THIS SOFTWARE IS PROVIDED IN AN “AS IS” CONDITION. NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. THE COMPANY SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.

```
#include <htc.h>
#include <stdio.h>
#include "balancedCounter.h"

/*  Update the following values in balancedCounter.h include file
// Define bytes 1 to 128 for storage, 24 first bytes are for the MAP, rest for the data
#define EEPROM_MAP_LOCATION 1
#define COUNTER_EEPROM_START 25
#define COUNTER_EEPROM_END 128
*/

// Create 2 map instances in RAM, one for primary and redundant storage each volatile eeprom_map
EepromMap[2];
// Create storage in RAM for Counter and init to 0 unsigned long int Counter = 0;

char Normal_Operation = 1;

void main(void)
{
    if (eepromMapInit() == 1)
        printf(“Counter value recovered successfully”);

     while (Normal_Operation == 1)
     {
         if  (writeCounters(++Counter) == 0)
         {
             // Handle irrecoverable error here
             printf(“Unable to persist counter without errors”);
             while(1); // wait
         }
     }
}
```

NOTES:

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Trademarks

Printed on recycled paper.

ISBN: 9781620764381

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Cleveland
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

Toronto
Mississauga, Ontario, Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Hangzhou
Tel: 86-571-2819-3187
Fax: 86-571-2819-3189

China - Hong Kong SAR
Tel: 852-2401-1200
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

Japan - Osaka
Tel: 81-66-152-7160
Fax: 81-66-152-9310

Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-5778-366
Fax: 886-3-5770-955

Taiwan - Kaohsiung
Tel: 886-7-536-4818
Fax: 886-7-330-9305

Taiwan - Taipei
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

11/29/11