Technical Note

Extending 90nm PCM Endurance from 1 Million WRITE Cycles Up to 1 Billion Cycles

Introduction

This technical note outlines a software solution called the parameter manager, which is used to increase the WRITE cycles for Micron's phase change memory (PCM) well beyond its standard endurance specifications. The document details the parameter manager's high-level structure, features, and function interfaces. It also outlines a solution for increasing the WRITE cycle endurance and describes the reference code, which can be written or modified for different applications. To download the reference code described in this document, visit the PCM Software page on micron.com.

The target audience for this technical note includes software engineers and designers.

This technical note presents parameter manager functionality and implementation. It covers the following:

- High-level architecture description
- Functional diagrams
- External API specification

This document uses the following terms and acronyms:

Table 1: Terms and Acronyms

Term	Definition
Parameter	The type of data (real-time data record, calibration parameters, and so forth) used to calculate electricity consumption. The parameter manager solution does not manage the RAM used for software, but the solution is suitable for a small RAM footprint.
Smart meter	An electrical device that regularly measures electricity power consumption. These devices record and store logs of energy consumption data for long periods of time. The electricity consumption can vary during a period of time based on specific parameters called calibration parameters.
WRITE cycling	The maximum number of WRITE cycles for this device is qualified at 1 million WRITE cycles using a cycling routine that writes by multiples of 32-byte page. The parameter manager solution extends the WRITE cycle endurance up to 1 billion with 1024 frames maximum redundancy at 64 bytes per frame.

Micron Technology, Inc., reserves the right to change products or specifications without notice.
©2011 Micron Technology, Inc. All rights reserved.

Products and specifications discussed herein are for evaluation and reference purposes only and are subject to change by Micron without notice. Products are only warranted by Micron to meet Micron's production data sheet specifications. All information discussed herein is provided on an "as is" basis, without warranties of any kind.

Goals and Objectives

The parameter manager software outlined in this technical note enables users to achieve up to 1 billion WRITE cycles on Micron's 90nm PCM products (Micron® P5Q and P8P family PCM). The user must ultimately define the data size and the amount of redundancy required.

Data logging is a natural application for PCM because of its non-volatility, high endurance, fast program, and bit-alterable features. Smart meters, for example, are growing quickly around the world and reliability of smart meter log data is a priority. Data consistency for writing is critical when an unwanted power loss occurs. When power is off, 2200uf capacitance has been used as a battery. Power off waveforms have been captured and the battery can last approximately 800ms at 3.3v and spend approximately 200ms to drop to 2.7v. This time frame is sufficient to write 188 bytes of data into memory. WRITE operations always succeed, and the operation WRITE order is sufficient to ensure data consistency when an unwanted power loss occurs.

The detailed solution in this document outlines how to increase WRITE cycling up to 1billion for a given set of parameters. The user must ultimately define the number of parameters that will be written and the redundancy size required. To ensure the cycling requirements, the solution requires a redundancy for storing data. However, space overhead should not be a limitation because Micron offers 90nm PCM devices up to 128Mb.

Environment

The P5Q serial PCM device works with a serial peripheral interface (SPI) where the maximum allowed frequency is 66MHz. The V_{CC} is stable and there is no power drop when erasing and programming. All WRITE operations are performed directly because there is no file system. The solution was tested on a Mainstone II board equipped with a P5Q serial PCM device in a stand-alone environment. The test was performed with direct READ and WRITE operations without the overhead of the file system.

Architecture and Functional Description

Features

The proposed solution achieves up to 1 billion WRITE operations for each parameter under the following assumptions:

- Data size is 64Kb
- READ and WRITE operations are executed by means of APIs
- Param_id identifies a parameter to be stored to/read from nonvolatile memory
- 1024 frames is the maximum redundancy considered
- Parameters are smaller than 60 bytes and always occupy 64 bytes in memory

Using these assumptions, consider an example where 200 parameters are logged. In this example, the actual amount of memory in PCM will be approximately 100Mb.

Micron Technology, Inc., reserves the right to change products or specifications without notice.
©2011 Micron Technology, Inc. All rights reserved.

Structural Overview

The parameter manager solution is made up of modules that are an abstraction of the major functionality of parameter management. The modules that make up the architecture are shown in Figure 1.

Parameter access is made by the API, which is implemented by the *parameter manager* layer. The real access to the physical address is transparent to the user, as each access to the parameter values are performed by the parameter manager APIs.

The *parameter cache manager* stores some information in RAM to speed up access to the PCM device. The data stored is minimal and takes into account the small footprint of the RAM resources.

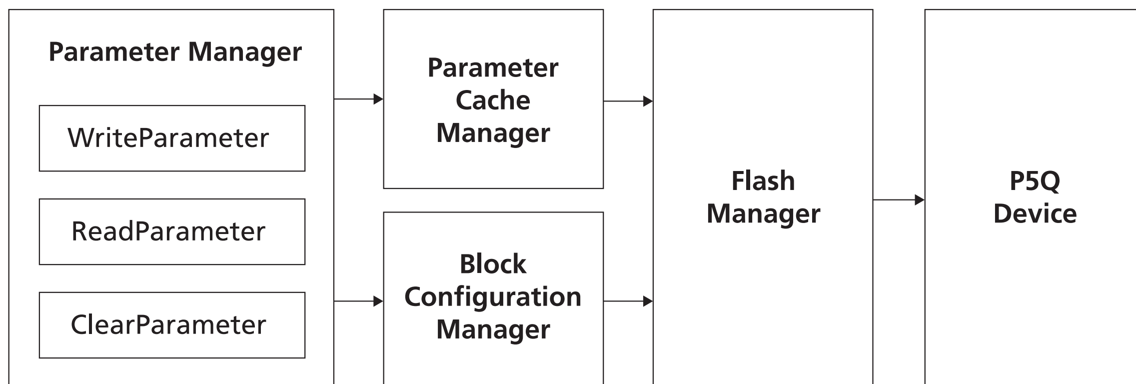
The *block configuration manager* defines the allocation on the physical address of the PCM device. To take into account the redundancy required by the user, each parameter is allocated a redundant chunk of physical addresses. This area is called the "parameter block" in the following sections. Each chunk is divided in frames whose number is fixed during configuration. Each frame occupies 64 bytes in physical memory (60 bytes for parameter content and 4 bytes for metadata). The parameter block's size is not necessarily related to the physical block of the PCM device. The size of each parameter block must be defined by the user during configuration.

There are multiple instances of a parameter's value stored in the parameter block, and only one instance at a time is the actual parameter value. Both the parameter cache manager and block configuration manager are involved in locating the physical address of the actual parameter value in the PCM device (see "PCM Usage and Cycling" on page 4). The physical address is identified with metadata tied to the actual data.

The *flash manager* module stores and retrieves row data from/to the P5Q PCM device. It accepts physical address and data directly from upper layer.

As previously stated, the physical memory stores both data and metadata. The metadata is not accessible to the user. The metadata is needed by the cache manager to recover the actual user data after a power loss occurs.

Figure 1: Data and Metadata



Micron Technology, Inc., reserves the right to change products or specifications without notice.
©2011 Micron Technology, Inc. All rights reserved.

Wear Leveling

It is not necessary to erase PCM devices before performing a new WRITE operation. In this case, wear leveling assumes the task of uniformly distributing WRITE operations on each physical address to prevent cells from prematurely wearing out. The updated data is only a portion of the block, so the remaining cells of the contiguous physical addresses are not worn during the WRITE operation.

PCM Usage and Cycling

The goal of wear leveling is to provide WRITE operations evenly across all parameter blocks. This goal is achieved with an algorithm based on circular queues and tags for each element of the queue that indicates the data that was most recently written. As a result, the entire procedure is transparent to the user. Data is stored in a physical area conceptually used as a circular buffer. The memory is never physically created during the execution–it is determined once during initialization of the algorithm. The physical space allocation is redundant and such redundancy equals the necessary number of cycling. As a result, the maximum cycling request from the user must be known at initialization to ensure the required cycling.

To prevent excessive cycling, a sort of remapping solution is implemented. The idea consists of having the data relocated at a circular selected location across one cycling block. The remapping is simple because the new available position is the next position.

The data addressing scheme consists of the following:

- One identification number that uniquely identifies the start address of the cycling block based on the parameter type in the input.
- The offset in the cycling block is used to locate and read the target data.

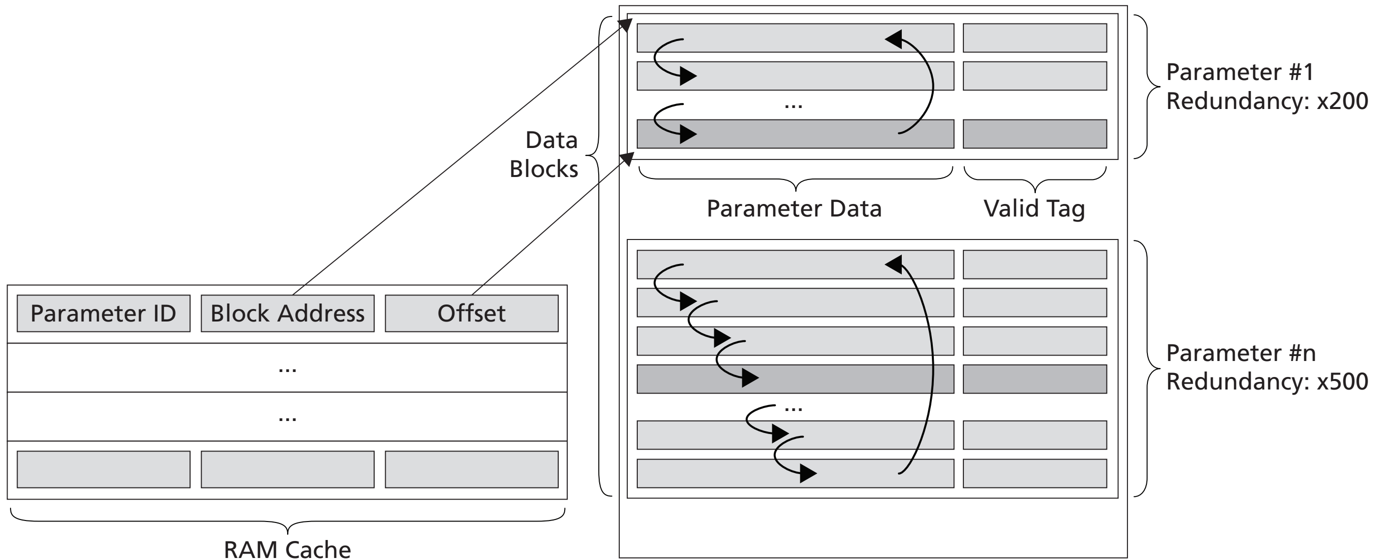
To reduce the number of writes to indicate actual data, the tag is written after the data. The following convention is assumed: If a read value of the tag is not initialized (all 0xff), the tag indicates that the location was not yet used.

Cache Management

To speed up READ and WRITE operations, a RAM cache is used. It transparently stores data needed by the algorithm so that future requests for that data can be more quickly dispatched. The data stored within a RAM cache elaborates the original values stored in the tag. The requested data is always contained in the cache, and this request can be served by simply reading the cache. Because the cache is stored in RAM, the data must be recomputed from its original storage location in the event of an unexpected power loss. The data is then read at startup from the physical tag stored in nonvolatile memory.

Micron Technology, Inc., reserves the right to change products or specifications without notice.
©2011 Micron Technology, Inc. All rights reserved.

Figure 2: Cache Management Diagram



Power Loss Recovery

After an unwanted power loss occurs, an algorithm recovers the physical position of the most recently written data into the PCM device. Data is labeled with a tag (metadata) that indicates the time at which it was written. The algorithm compares the values in the parameter block and determines which is the most recent. When the element being compared to equals the most recent data, the search stops and the procedure returns the position of the data. If the element is not equal to the most recent data, a comparison is made to determine whether the current data is less recent or more recent than the element. Depending on the result, the algorithm then starts over, searching only the top or bottom of the parameter block elements. If the input is not located within the array, the algorithm outputs the first position available into the array (this is when no data has been written yet). The algorithm halves the number of items to check with each successive iteration, thus locating the requested item (or determining its absence) in logarithmic time. The search is a dichotomy, and the performance is defined as:

- Worst-case performance: O(log n)
- Best-case performance: O(1)
- Average-case performance: O(log n)

The search is iterated on each parameter registered from the user, and the total average performance is defined as follows:

- Average-case performance: Number of parameters * O(log (length of parameter block))

Micron Technology, Inc., reserves the right to change products or specifications without notice.
©2011 Micron Technology, Inc. All rights reserved.

Functional Description

This section summarizes the assumptions made in this document and provides an overall description of the functionality of the parameter manager solution.

- PCM memory is split in two sets:
  - Data: Information used to store the user's parameters and can be relocated
  - Info: Information used to understand where the actual data is on the device
- The data block occupies a fixed area of the PCM device (~5Mb)
  - Data is structured in "parameter blocks" of virtual pages that are remapped over physical pages
  - 1024 frames is the maximum redundancy over physical pages
  - The actual size of a data block is dimensioned to meet cycling constraints
- Data blocks are structured so that for each parameter block the following information is available:
  - Physical pages of 64 bytes in size that indicates the user's data in the parameter block
- Info blocks are structured so that for each parameter ID, the following information is available:
  - Start address of the parameter block
  - Offset in the parameter block indicating where the valid data is on the device
- The info block occupies a fixed area of the PCM device (~1Kb)
  - The info block can be stored in RAM
  - When stored in RAM, 4 bytes of the parameter block are required to indicate where the valid data is on the device

The parameter manager enables the following operations:

- Implementation after power on (performed once at boot)
  - The valid tags are read
  - Valid tags are read from the physical pages and copied into the RAM cache table
  - Total READ operations from the physical page are ~180
  - Total RAM size required is ~1Kb
- Implementation of the ReadParameter (param_id, length, outputbuffer)
  - Info bytes are read from the RAM cache
  - The physical page is understood
  - Data is read from the physical page and copied into the outputbuffer
  - One READ operation from the physical page is performed
- Implementation of WriteParameter (param_id, length, outputbuffer)
  - The info bytes are read from the RAM cache
  - Data is copied into the next physical page from the outputbuffer
  - The info bytes are updated to the RAM cache to indicate the next page
  - One WRITE operation to the physical page is performed

Micron Technology, Inc., reserves the right to change products or specifications without notice.
©2011 Micron Technology, Inc. All rights reserved.

API Description

ClearParameter Function

This function clears one parameter. The parameter manager will not be aware of the parameter content, but no ERASE operation is performed in the nonvolatile memory.

Syntax

```
ReturnType ClearParameter (
        UINT8       param_id)
```

Parameters

- UINT8 param_id: The parameter identifier.

ConfigurationManager_Init Function

The function initializes the resources that must be called at the device's boot. This function is hardware dependent, and must be modified as needed.

Syntax

```
void ConfigurationManager_Init(void);
```

Parameters

None.

ReadParameter Function

This function reads one parameter.

Syntax

```
ReturnType ReadParameter (
        UINT8       param_id,
        UINT8       *buffer,
        UINT16      len)
```

Parameters

- UINT8 param_id: The parameter identifier.
- UINT8 *buffer: The content of the parameter to be read.
- UINT16 len: The number of bytes to be read.

Micron Technology, Inc., reserves the right to change products or specifications without notice.
©2011 Micron Technology, Inc. All rights reserved.

WriteParameter Function

This function writes one parameter.

Syntax

```
ReturnType WriteParameter (
        UINT8       param_id,
        UINT8       *buffer,
        UINT16      len)
```

Parameters

- UINT8 param_id: The parameter identifier.
- UINT8 *buffer: The content of the parameter to be written.
- UINT16 len: The number of bytes to be written.

Configuration Example

This section provides information about the parameter definition and an example of the parameter definition.

Each parameter is identified by its parameter ID, which must be unique. The RC parameter is used from the algorithm to calculate the start address of each parameter. The default value of this parameter ensures a redundancy of 1024 frames and enables the ability to reach WRITE cycling up to 1 billion for each parameter. The parameter table must be configured in the *configuration_manager.c* file. The algorithm populates the structures in RAM based on this table.

```
#define NPARAM 2
// example with two parameters
#define MAX_NPARAM 20
                #define PARAM_ID_0 0
#define PARAM_ID_1 1

#define DEFAULT_REDUDANCY 1

typedef struct {
UINT8 id;
UINT8 rc;
cache_table_t *ct;
} parameter_table_t;
extern parameter_table_t parameter_table[MAX_NPARAM];
An example of a parameter definition could be:
const parameter_table_t parameter_table_const[MAX_NPARAM] = {
{PARAM_ID_0, DEFAULT_REDUDANCY, NULL},
{PARAM_ID_1, DEFAULT_REDUDANCY, NULL},
};
```


Micron Technology, Inc., reserves the right to change products or specifications without notice.
©2011 Micron Technology, Inc. All rights reserved.

To start, write a function *main()* and include the *common.h* header file that contains the definition of the main structures of the algorithm.

The following example writes two parameters. The first is made up of 3 bytes and the second is made up of 56 bytes. The *ConfigurationManager_Init()* function initializes the board after powering on, populates the RAM with the parameter table, and checks the data structures to recover it from power loss. The functions must be adapted for initializing the board as needed. Comments within the source code explain how it should be modified for individual target hardware. In addition, the Flash manager component should be modified for target hardware.

```
#include "common.h"

int main( int argc, char ** argv )
{
    unsigned char wbuffer[60], rbuffer[64];
    int i;

    for (i=0;i<60;i++)
        wbuffer[i]=i;

    for (i=0;i<64;i++)
        rbuffer[i]=0x0;

            ConfigurationManager_Init();

            WriteParameter (0, &wbuffer[0], 3);
            WriteParameter (1, &wbuffer[0], 56);

                ReadParameter (0, &rbuffer[0], 3);

                ReadParameter (1, &rbuffer[0], 56);

                return 0;
}
```


Micron Technology, Inc., reserves the right to change products or specifications without notice.
©2011 Micron Technology, Inc. All rights reserved.

Conclusion

The parameter manager can be applied to a wide range of applications to support extensive data WRITE cycling requirements. The solution enables the customer to maximize memory performance, reduce the bill of materials, and speed up design time. Customers can consolidate multiple components in their design into a single PCM device and still meet the extensive data logging requirements of their application using the parameter manager model.

The simple interface further minimizes integration time and has a minimal impact on design resources. This general API also ensures that existing designs can get up and running as soon as possible. This enables a faster development and maintenance model for existing projects. The key results are minimized system complexity and reduced time-to-market for new solutions.

8000 S. Federal Way, P.O. Box 6, Boise, ID 83707-0006, Tel: 208-368-3900
www.micron.com/productsupport Customer Comment Line: 800-932-4992
Micron and the Micron logo are trademarks of Micron Technology, Inc. All other trademarks are the property of their respective owners.

Micron Technology, Inc., reserves the right to change products or specifications without notice.
©2011 Micron Technology, Inc. All rights reserved.

Revision History

Rev. A **11/11**

- Initial release of document.

Micron Technology, Inc., reserves the right to change products or specifications without notice.
©2011 Micron Technology, Inc. All rights reserved.