

## Introduction

To save power, the MAX<sup>®</sup> II CPLD can be completely powered down into hibernation mode during an inactive period. The CPLD can power down automatically, and allow the system to power it up again when the system requires the CPLD to execute a task, while retaining the register data.

This application note discusses an example system in which MAX II registers' data is automatically stored before the power supply is cut off after a certain predefined idle period, and the data is automatically reloaded back into the registers when the system is powered up again.

## Design Example Description

In this application note, an example system has been created to show the capability of the MAX II device in a self-power-down system. The registers' data is retained to ensure that the system's operation is not affected by the power-down. The example system consists of some external hardware circuitry, as well as the design in the MAX II device itself. The hardware circuitry and the design in the CPLD work together to ensure the success of the self-power-down system.

In this example application, a 4-bit binary up-counter is the user application module in which the counter's data must be retained when the MAX II CPLD is powered down after a predefined period of inactivity. Upon power-up, the data is reloaded into the counter's registers, so the count operation resumes from the previous value.

Supplying a trigger signal (in this example, a low pulse) with a pushbutton to the counter increases the count value by one. If the counter does not receive a trigger signal after a fixed duration, the design automatically stores the count value into the user flash memory (UFM) before triggering the external circuitry to power down the MAX II device.

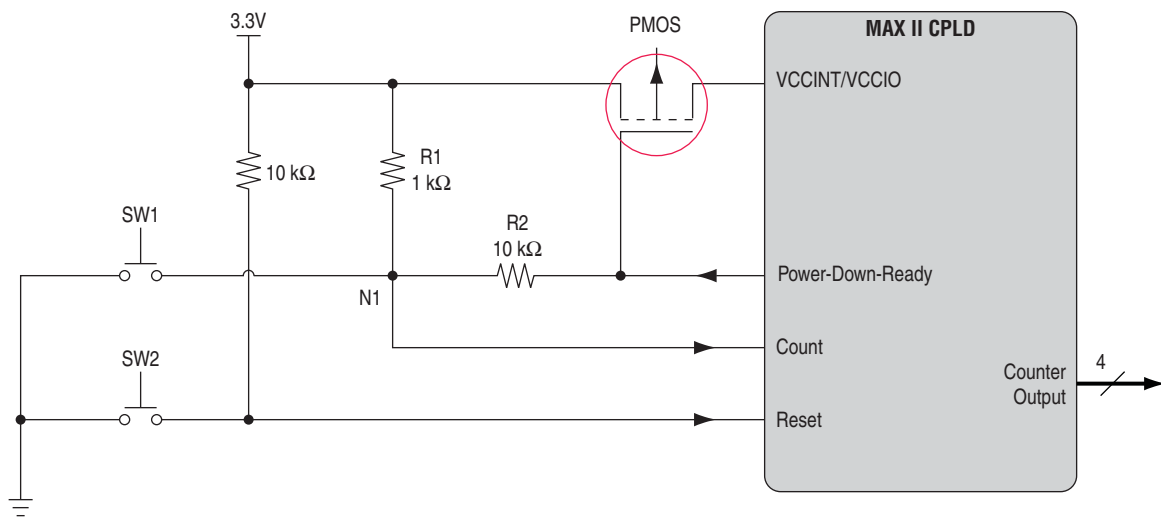
When the next trigger signal is received to increase the count, the device automatically powers up. The design then reads back the data from the correct location in the UFM and reloads the counter's registers. The count value is incremented by one from the read-back data.

You can also reset the counter by pressing the reset button that clears the count value.

## External Hardware Circuitry

A circuit outside the MAX II CPLD is used to control the power supply to the MAX II device, as well as the input to the counter. [Figure 1](#) shows the external circuitry.

Figure 1. External Hardware Circuitry



A P-Channel MOSFET is used to control the power supply to the MAX II device. If the gate of the P-Channel MOSFET is at low voltage level, the transistor is turned on to allow current to flow through from the power supply to the MAX II device. If the transistor's gate is at a high voltage level, the transistor is turned off and the MAX II device is in powered-down state. Because of the 1-k $\Omega$  pull-up resistor R1 and the fact that the MAX II device's I/O pins do not drive out if the device is powered down, the P-channel MOSFET is off until the pushbutton SW is pressed and causes the voltage level at the transistor's gate to low and allows the transistor to power up the MAX II device again.

When the MAX II device goes into user mode after being powered up (at most, 450  $\mu$ s), the MAX II device drives the MOSFET's gate low through its Power-Down-Ready output pin to allow continuous operation even though pushbutton SW1 is released. If the MAX II device detects that the counter is not active for a predefined period of time, the device then drives the Power-Down-Ready output pin high and turns off the MOSFET, thus self powering down the CPLD.


Pushbutton SW1 is also used for incrementing the count value. The system uses a single pushbutton for powering up the device, as well as incrementing the count value, so you do not need to know whether the device is powered down when you want to increment the count.

As the device recognizes a low pulse as the trigger to increment the count value, resistors R1 and R2 are used as a voltage divider. Even though the MAX II device forces the MOSFET's gate low (using the Power-Down-Ready output pin), the node N1 that connects to the input pin Count still has a voltage higher than  $V_{IH(min)}$  for the input pin to recognize node N1 as high. It also allows count operation when pushbutton SW1 is pressed, because the voltage level must be lower than  $V_{IL(max)}$  to be recognized as low.

Use pushbutton SW2 to reset the counter so that it starts from zero.



The external hardware circuitry described uses the same 3.3-V power supply for  $V_{CCINT}$  and  $V_{CCIO}$ . Modification to the external hardware circuit is needed for designs that require  $V_{CCIO}$  to be at a different level than  $V_{CCINT}$ .

 For the MAX II device to fully support the hot-socketing feature when the device is powered down, the VCCINT and VCCIO pins must be connected to ground.

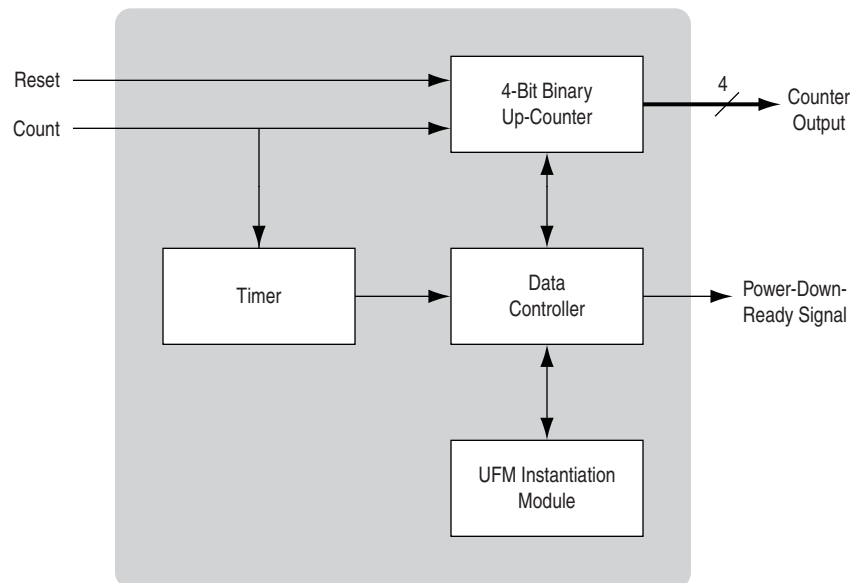
## Quartus II Design in the MAX II CPLD

The design consists of four modules:

- 4-bit binary up-counter
- Data controller
- Timer
- UFM instantiation module

Figure 2 shows the modules in the design. The design is created with the Quartus II software.

**Figure 2.** Modules in the Design



### 4-Bit Binary Up-Counter

The 4-bit binary up-counter is the user application module that counts from four bits of zeros to four bits of ones. The Reset and Count input pins of the device go to the input ports of the counter, while the output of the counter goes to four output pins. Every low pulse to the Count input pin increments the count value by one. When the count value reaches its maximum value (four bits of ones), the next low pulse to the Count input port resets the count value to four bits of zeros. To reset the count value to all zeros at any time, assert the Reset signal.

### Data Controller

When the data controller receives the signal from the timer indicating that the device does not have any activity for a predefined duration, the controller automatically reads the data from the counter and writes the data into the UFM. Upon completion of the task, the data controller then asserts the Power-Down-Ready signal high for the external circuitry to power down the device.

When the device powers up, the data controller automatically goes to the location in the UFM where the data from the counter is stored, and reads back the data before the device powers down again. The controller then reloads the registers of the counter with the data.

The data controller interfaces with the module that instantiates the UFM through the Altera proprietary interface protocol. The controller writes the data to a blank location in the UFM every time before the device is powered down, and reads back the data from the correct location when the device is powered back up.

The controller uses almost all the addresses in sector 0 of the UFM for data storing purposes. The erase operation takes additional time to complete, so when the UFM sector is full, the controller automatically erases that sector. Detailed information about how the controller determines the location in the UFM for data storage and retrieval is discussed in the [“User Flash Memory Data Save and Retrieval Method”](#) section.

### Timer

The timer determines the maximum idle time allowed before the MAX II CPLD is powered down. The timer is a counter in which the most significant bit (MSB) signal is used to trigger the process of saving the counter data into the UFM and powering down the CPLD. The width of the counter determines the wait time. The wait time ( $T$ ) is shown in [Equation 1](#).

#### Equation 1.

$$T = \frac{2^n}{2 \times f}$$

#### Where:

- (1)  $n$  is the width of the counter
- (2)  $f$  is the frequency of the internal oscillator that clocks the counter, typically 5 MHz

For a longer wait time, increase the counter's width. The signal from the pushbutton that increments the 4-bit binary up-counter actually resets the timer and prevents the CPLD from being powered down. For this design example, the width of the counter is 26 bits wide, so the wait time is approximately 7 seconds.

The timer can also be implemented outside the MAX II device using other external components to reduce logic element (LE) usage.

### User Flash Memory Instantiation Module

This module instantiates the UFM of the device and all communications with the UFM goes through this module. This module is the Altera serial interface megafunction (the ALTUFM\_NONE megafunction) that is created with the Quartus II Megawizard® Plug-In Manager. The data controller is created based on the interface protocol of this megafunction.

The UFM has two sectors. The design example only uses sector 0 of the UFM for data storage purposes. Sector 1 of the UFM can be used for storing other user data. Erasing sector 0 does not affect the data in sector 1.

Using this megafunction, you can also utilize the MAX II internal oscillator. Other modules in the design use the oscillator output as the clock source. The frequency of the oscillator output is between 3.3 MHz and 5.5 MHz.

 For more information about the ALTUFM\_NONE megafunction, refer to the [ALTUFM Megafunction User Guide](#) and to the [Using User Flash Memory in MAX II Devices](#) chapter in the *MAX II Device Handbook*.

## User Flash Memory Data Save and Retrieval Method

The design writes the data into the UFM before the device is powered down, and reads back the data from the UFM when the device is powered up again. The challenge is to determine which location or address in the UFM the design should write the data into, and then be able to go back to the same address upon power-up to read back the data.

The erased UFM has all 1's as the content. When you write any data to the UFM, the data is "AND-ed" with the existing data in the UFM. In other words, if a zero is written into a location that has 1, the content becomes 0. If a 1 is written into a location that has 1, the content remains 1.

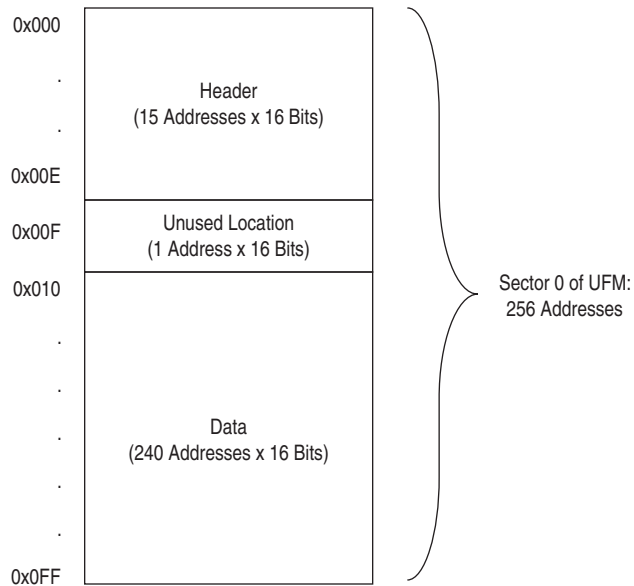
When the content becomes 0, you cannot write 1 to that location again, except when the entire sector of the UFM is erased.

Alternatively, the controller uses only one address in the UFM for data storage. Every time before the device is powered down, the controller writes the data to a fixed address in the UFM, and when the device is powered up, the controller goes directly to that address to read back the data. Using the same address allows the address to be hard-coded in the design, thus making the design simpler. However, one drawback is that every time before the write operation occurs, the entire sector of the UFM must be erased before new data can be written into it. This slows down the operation as erase time is much longer (as compared to the duration for read or write operations).

Because MAX II CPLD is SRAM-based, any data not stored in the flash memory is lost when a power cycle occurs. Although the data is safely stored in the UFM, there is no way for the device to "remember" the location in the UFM where the data is stored. Searching all of the addresses in the UFM is time-consuming and does not actually tell the controller which data is the most recent data written into the UFM.

For this application, a new method is used. Instead of using only one address from the total of 256 addresses from sector 0 of the UFM, this method uses 240 addresses for data storage, and another 15 addresses to store header information. [Figure 3](#) shows how addresses in the UFM are used.

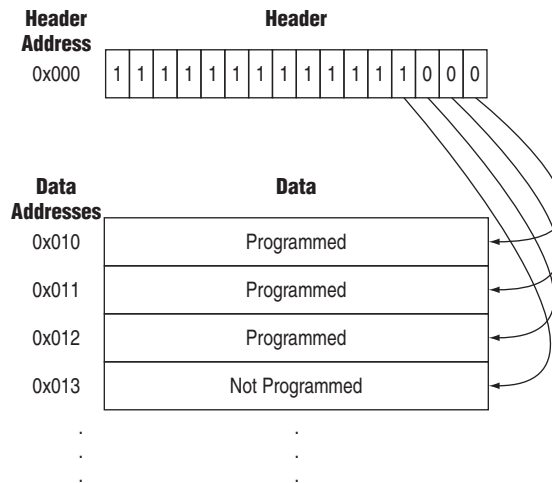
**Figure 3.** Addresses in the UFM Used for Header and Data Storage



The header section is used to keep track of the most recent address into which the controller has written. Upon power-up, the controller only checks the header section to determine the location where the data is stored just before the device is powered down. The header section spans from address 0x000 to address 0x00E—15 addresses total.

Each bit in the header section represents the validity of the data in one address in the data section. The first bit in the header section, which is the least significant bit (LSB) of address 0x000, represents address 0x010 in the data section, and so on. If the header bit is 0, this means that the address in the data section contains valid data. Header bit 1 means the address in the data section is blank. Figure 4 shows the relationship between the header section and the data section.

**Figure 4.** Header Section and Data Section



During power-up, the controller reads the header address one by one. Data is written into addresses in the data section sequentially, from address 0x000 to address 0x015, from LSB to MSB of each address. The controller “knows” how many addresses in the data section are used based on the number of 0’s in the header section.

If the controller identifies that the MSB in a particular header address is 0’s, it recognizes that the header address contains 16 bits of 0’s and proceeds to read from the next header address. If the controller identifies that the MSB in the header address is 1, the controller then proceeds to check how many 0’s are in that header address to determine the exact location in which the data is stored. However, if the data in this header address is all 1’s, the controller recognizes that the previous header address has all 0’s.

By checking the header section, the controller can determine the exact location where the data was stored before the device was powered down. Figure 5 shows how the controller obtains the data address for the controller from which to read the data, based on the information from the header bits.  $N$  is the number of 0’s in the header address last read by the controller.

**Figure 5.** Data Address

---

	Bit 8	Bit 7..4	Bit 3 ..0
Data Address =	0	Header Address + 1	$N - 1$

---

When the device is powered down, the controller writes the counter data into the next blank data address, and then writes a 0 in the subsequent header bit location. If the UFM sector is full, the controller first erases the sector before writing the data into the first address in the data section, and then writes 0 into the first bit in the header section.

The benefits of using header data include the following:

- The controller does not need to search all the addresses in sector 0 of the UFM to find the correct data. The header section only occupies a total of 15 addresses in sector 0 of the UFM; the controller does not have to search for more than 15 addresses to determine where the most recent data is located.
- The header indicates to the controller whether the data in a particular location is valid. If the data written into an address is coincidentally all 1’s, it will appear that the address is blank, because erasing the UFM sets all the data in all addresses to all 1’s. In this case, the header bit shows that the data is valid.

## Conclusion

The design example shows the capability of the MAX II CPLD to store the register data into the non-volatile UFM before self-powering-down after an idle period. The device also reads back the data from the UFM and reloads the registers for the device to resume operation shortly after powering up. The method for putting the CPLD into hibernation mode is detailed in this application note.

## Referenced Documents

This application note references the following documents:

- *ALTUFM Megafunction User Guide*
- *Using User Flash Memory in MAX II Devices* chapter in the *MAX II Device Handbook*

## Revision History

Table 1 shows the revision history for this application note.

**Table 1. Document Revision History**

Date and Version	Changes Made	Summary of Changes
January 2009 v.1.0	Initial release	—



101 Innovation Drive  
 San Jose, CA 95134  
[www.altera.com](http://www.altera.com)  
 Technical Support  
[www.altera.com/support](http://www.altera.com/support)

Copyright © January 2009. Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.



I.S. EN ISO 9001