# Putting Altera MAX Series in Hibernation Mode Using User Flash Memory

2016.01.14

AN-547 Subscribe Send Feedback

The MAX® II, MAX V, and MAX 10 devices can be used in this example application.

You can completely power down the device into hibernation mode during an inactive period. The device powers down automatically, and allows the system to power it up again when the system requires the device to execute a task, while retaining the register data.

This example application shows the capability of the supported Altera device to store the register data into the non-volatile user flash memory (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.

**Related Information**

- **User Flash Memory (ALTUFM) Megafunction User Guide**
  Provides more information about the ALTUFM IP core.
- **Using User Flash Memory in MAX II Devices**
- **User Flash Memory in MAX V Devices**
- **MAX 10 User Flash Memory User Guide**
- **Design Example for MAX II**
  Provides the MAX II design files for this application note (AN 547).
- **Design Example for MAX 10**
  Provides the MAX 10 design files for this application note (AN 547).

## Design Example Description

The following design examples show the capability of the supported Altera device in a self-power-down system. MAX II and MAX V devices are different from MAX 10 devices due to different flash memory usage modes. The registers' data is retained to ensure that the system's operation is not affected by the power-down. The example system consists of the design implemented in a MAX II, MAX V, or MAX 10 device as well as some external hardware circuitry. The hardware circuitry and the design in the device work together to ensure the success of the self-power-down system.

In the example applications, a 4-bit binary up-counter is the user application module in which the counter's data must be retained when the supported Altera device 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 to the counter with a pushbutton (active low signal) 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 UFM before triggering the external circuitry to power down the supported Altera device.

For MAX II and MAX V devices, 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. For MAX 10 devices, pressing the trigger signal powers up the design and fetches the last value stored and puts the value out on the LED display, but doesn't increment the value.

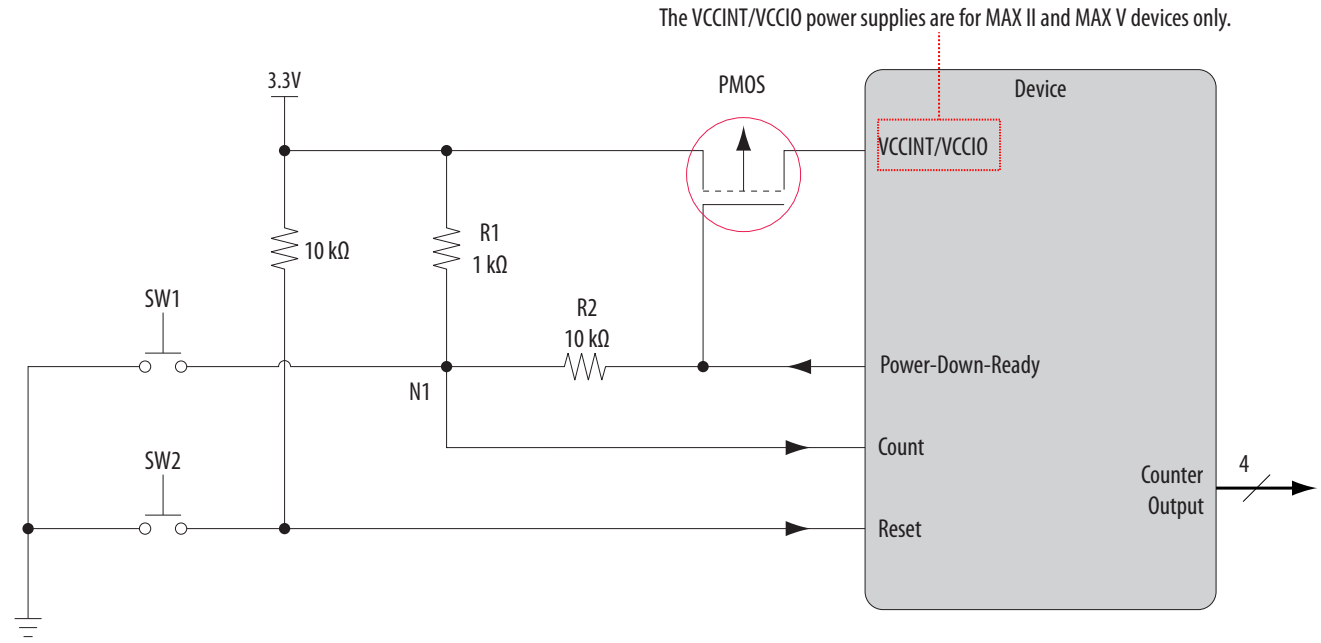
You can also reset the counter by pressing the reset button that clears the count value.

## External Hardware Circuitry

An external circuitry is used to control the power supply to the supported Altera device, as well as the input to the counter.

**Figure 1: External Hardware Circuitry**

The VCCINT/VCCIO power supplies are applicable for MAX II and MAX V devices only. The power supplies for MAX 10 single supply devices are VCC_ONE, VCCA, and VCCIO. The power supplies for MAX 10 dual supply devices are VCC, VCCD_PLL, VCCINT, VCCA, VCCA_ADC, and VCCIO.




A P-channel MOSFET is used to control the power supply to the supported Altera 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 supported Altera device. If the transistor's gate is at a high-voltage level, the transistor is turned off and the supported Altera device is in powered-down state.

The P-channel MOSFET can be steadily turned off with the 1-kΩ pull-up resistor R1. Besides, I/O pins of the device do not drive out when the device is powered down.

To turn on the P-channel MOSFET, press the pushbutton SW1. Pressing pushbutton SW1 causes the voltage level at the transistor's gate to low and allows the transistor to power up the supported Altera device again.

After power-up, the supported Altera device goes into user mode, at the maximum of 450 µs. The supported Altera device drives the P-channel MOSFET's gate low through its `Power-Down-Ready` output pin to allow continuous operation even though pushbutton SW1 is released. If the supported Altera device detects that the counter is not active for a pre-defined period of time, the device then drives the `Power-Down-Ready` ouput pin high. Driving the `Power-Down-Ready` output pin high turns off the P-channel MOSFET, thus self powering down the supported Altera device.

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 device forces the P-channel MOSFET's gate low using the `Power-Down-Ready` output pin, the node N1 that connects to the `Count` input pin 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.

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

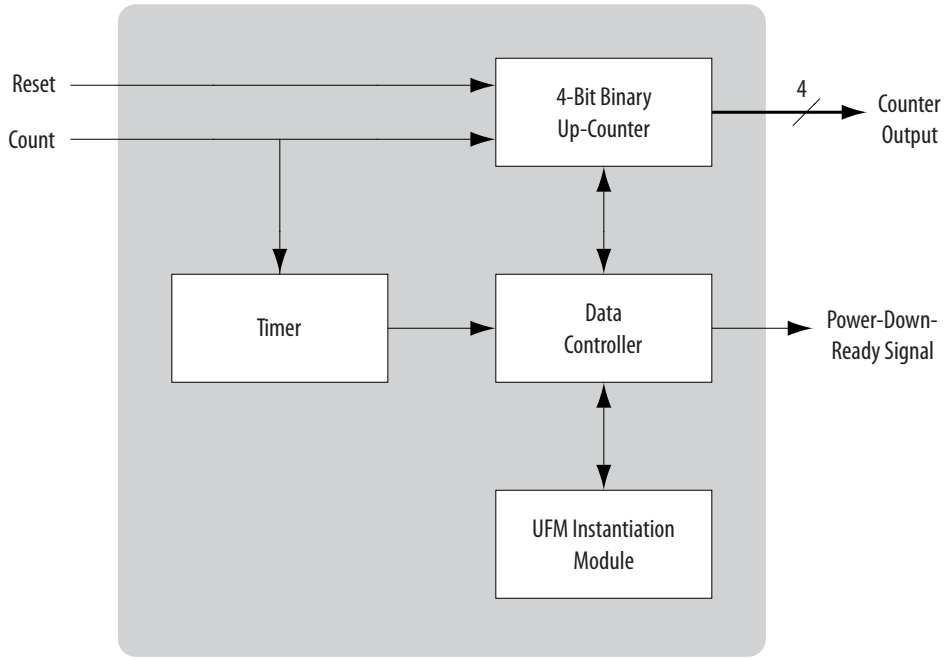
**Note:** To use the hot-socketing feature when the device is powered down, connect the `VCCINT` and `VCCIO` pins to ground.

## Design in the MAX II Device

The MAX II design consists of four modules:

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

**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 pre-defined 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.

**Related Information**

**User Flash Memory Data Save and Retrieval Method** on page 6
Provides detailed information about how the controller determines the location in the UFM for data storage and the retrieval method.

## Timer

You can use the timer to determine the maximum idle time allowed before the supported Altera device is powered down. 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 supported Altera device.

**Figure 3: Wait Timer Equation**

The following equation shows how to calculate the wait time where $T$ is wait time, $n$ is the width of the counter, and $f$ is the frequency of the internal oscillator that clocks the counter, typically 5 MHz. The width of the counter determines the wait time.

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

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

You can also implement the timer outside of the supported Altera 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 example application is using the Altera Serial Interface (NONE) of the user flash memory (ALTUFM_NONE) IP core. The data controller is created based on the interface protocol of this IP core.

The UFM has two sectors. The example application uses sector 0 of the UFM for data storage purposes. You can use sector 1 for storing other user data. Erasing sector 0 does not affect the data in sector 1.

Using this IP core, you can also utilize the internal oscillator of the supported Altera device. 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.

**Related Information**

- **User Flash Memory (ALTUFM) Megafunction User Guide**
  Provides more information about the ALTUFM IP core.
- **Using User Flash Memory in MAX II Devices**
  Provides more information about user flash memory.

- **User Flash Memory in MAX V Devices**
  Provides more information about user flash memory.

## 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 written to the existing content in the UFM through an AND logic. In other words, if a `0` 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. When the device is powered up, the controller goes directly to that address to read back the data.

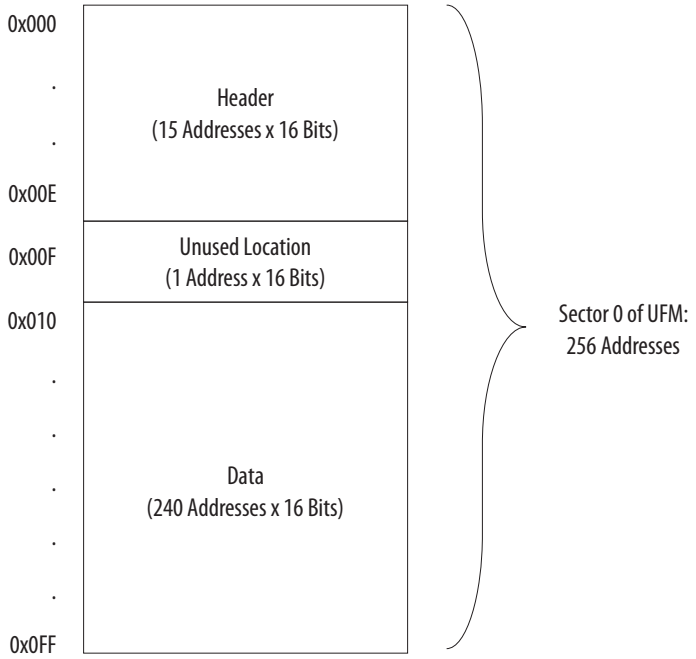
Using the same address allows the address to be hard-coded in the design, thus making the design simpler. However, one drawback is that every time before the write operation occurs, the entire sector of the UFM must be erased before new data can be written into it. This process slows down the operation as erase time is much longer as compared to the duration for read or write operations.

Because the supported Altera device 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, the device cannot recall 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.

In this example 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 4: Addresses in UFM Used for the Header and Data Storage**

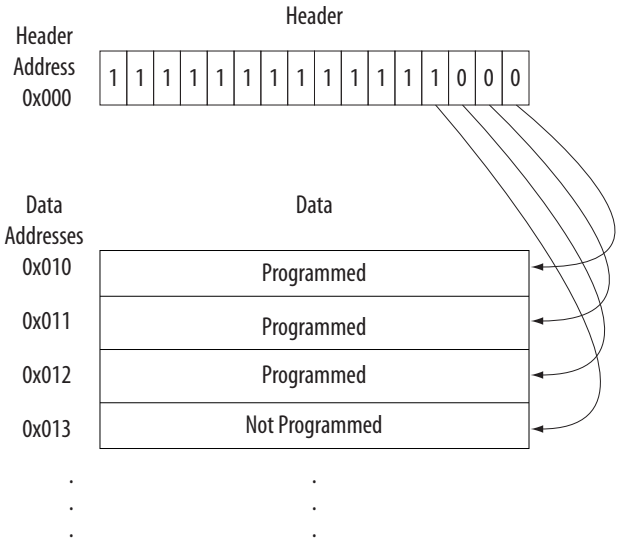
The UFM described in this figure is applicable for MAX II and MAX V devices. For the size of the UFM and data length of the MAX 10 devices, refer to the *MAX 10 User Flash Memory User Guide*.




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 5: Header Section and Data Section**



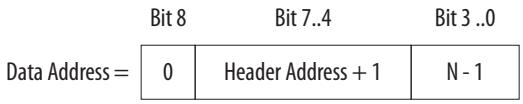

During power-up, the controller reads the header address one by one. Data is written into addresses in the data section sequentially, from address 0x000 to address 0x015, from LSB to MSB of each address. The controller can determine which addresses in the data section has been used based on the number 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 6: Data Address**

The following figure 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.




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:

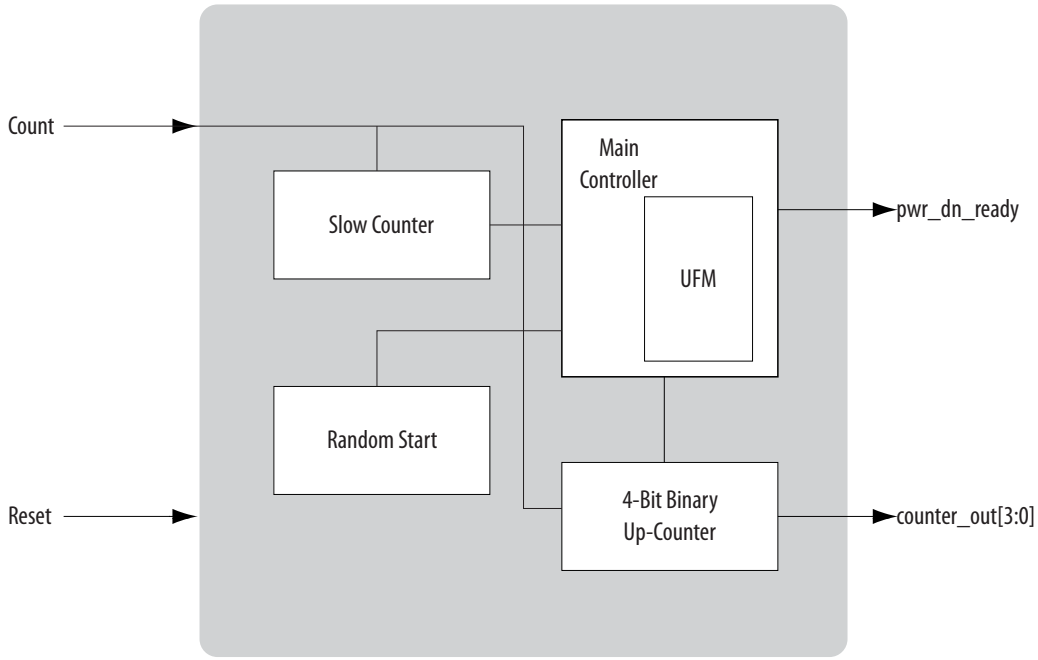
- The controller does not need to search all the addresses in sector 0 of the UFM to find the correct data. The header section only occupies a total of 15 addresses in sector 0 of the UFM. The controller does not have to search for more than 15 addresses to determine where the most recent data is located.
- 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.

# Design in the MAX 10 Device

The MAX 10 design consists of four major modules:

- 4-bit binary counter
- 16-bit binary slow counter
- Main controller, which instantiates the UFM module within it
- Random start block which initiates a power-on reset (POR)

**Figure 7: Modules in the MAX 10 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 to drive LEDs.

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.

Upon power-up, the Main Controller fetches the UFM stored counter value and sends it to the 4-bit binary up-counter to write out the value to the output pins. When a shut down is initiated by the 16-bit binary slow counter module, the main controller fetches the current stored count value from the 4-bit binary up-counter to write into the UFM before asserting pwr_dn_ready signal.

## 16-Bit Binary Slow Counter

The 16-bit binary up-counter is the user application module that counts from 16 bits of zeros to 16 bits of ones. It is triggered upon the first Count signal received after the main controller has fetched the UFM stored counter value and written it to the 4-bit binary up-counter display. After the first Count signal is received, the counter counts to all ones. When a new Count signal is received, the counter resets all zeros and resumes counting again. The Reset signal clears the slow counter and waits till a new Count signal is received before counting again.

When the 16-bit slow counter reaches the maximum value, it sends a signal to the main controller to begin the shut-down sequence of fetching the current counter value from the 4-bit binary up-counter, storing it in the UFM, and asserting pwr_dn_ready signal which shuts down the power supply to the device as shown in the Modules in the MAX 10 Design figure.

The 16-bit slow counter, as shown in the Modules in the MAX 10 Design figure, is clocked by the internal oscillator, which can operate up to a maximum frequency of 116 MHz in the 10M08 device used in this design example. The 16-bit resolution is used to make simulation time short, but in real-time operation, this counter will reach the maximum count value in a fraction of a second (2^16*8.62 ns), which is not ideal for testing on an evaluation kit. Everytime you press the Count button, the device would power off before you could press it again.

To change the design for board-level testing, replace the design file **./core/slow_counter.v** with the alternate design file provided **./core/slow_counter_30bit.v** to be used in the **.qsf** assignments for compilation. This will increase the number of bits used to an acceptable value for real-time operation. A counter of 30-bits will extend the slow counter to take approximately 9 seconds before issuing a shut-down request.

## Main Controller

The main controller is the heart of the design. The main controller instantiates the UFM which has an Avalon® Memory-Mapped (MM) Slave interface and controls all of the reading and writing to the UFM.

At power-up, the random start block issues a POR cycle and then sends a signal to the main data controller to read the UFM stored counter value at a static address and sends it to the 4-bit binary up-counter.

After the power-up fetch process completes, the main controller sits idle until the Reset signal is asserted or when it receives a signal from the 16-bit binary slow counter to initiate the power-down sequence.

When the main controller receives the power-down signal from the 16-bit binary slow counter, the main controller reads the current 4-bit binary up-counter value, removes the write protection of the UFM, writes the 4-bit binary up-counter value to the static address hard-coded in the design, turns back on the write protection for the UFM, and then sends the pwr_dwn_ready signal out of the device, which shuts off the power supply to the MAX 10 device.

## Random Start Block

At power-up, the random start block issues a POR cycle and then sends a fetch request signal to the main controller to read the UFM stored counter value at a static address and sends it to the 4-bit binary up-counter. When the `Reset` signal is asserted, it clears the system and reissues the fetch request to the main controller.

# Document Revision History

| Date | Version | Changes |
|---|---|---|
| January 2016 | 2016.01.14 | Removed the *User Flash Memory Data Save and Retrieval Method in MAX 10 Devices* section. The UFM in MAX 10 devices must be erased before you write new data. |
| November 2014 | 2014.11.07 | Added MAX V and MAX 10 devices. |
| January 2009 | 1.0 | Initial release. |