Application Report
SNAA321A–July 2018–Revised August 2019

16 MB Radiation-Hardened SRAM with Error Detection And Correction (EDAC) to Mitigate Soft Errors

ABSTRACT

The SMV512K32 is a 32-bit × 512k asynchronous static RAM (SRAM) designed for high radiation environments. The SMV512K32 was designed using patented HARDSIL® technology and utilizes fault tolerant design principles.

Contents

List of Figures

List of Tables

Trademarks

HARDSIL is a registered trademark of Vorago Technologies Inc.
All other trademarks are the property of their respective owners.

1 Introduction

The SMV512K32-SP is a unique memory device with built-in error correction and detection that can provide years of reliable memory storage even in the harsh environment of space.

Small, high energy particles in space can pass through a piece of silicon causing very isolated Single Event Upset (SEU) malfunctions that cannot be prevented. By physically spacing the individual bits in a 32-bit data word, the chance of a single particle strike corrupting two bits in a single word is near zero. Implementing a form of error correction on single bit errors is common for space applications. Error detection and correction (EDAC) working with sequential periodic reads, often referred to as a “scrub” operation, helps address single-bit errors and prevents any single memory word from accumulating two or more errors over time.

The purpose of this application note is to help the user take full advantage of the unique features of this device to provide exceptional memory reliability in a harsh environment. The SMV512K32 EDAC functionality can detect a multiple bit error and detect and correct the output from any read containing a single bit error in the memory array. In conjunction with EDAC, the SMV512K32 memory scrub capability can be used to read through the memory array and correct all single-bit errors in either slave mode (scrub controlled externally) or master mode (SRAM controls scrub operation). The scrub function has a write-back feature to correct the stored data if a single-bit error is detected. Normal reads with EDAC enabled will correct single bit errors on the output data, but will not write the corrected information back to the memory. With the proper scrub rate, the single bit errors should be corrected often enough that the occurrence of multi-bit errors (MBE) will be statistically insignificant. It is important, however, that the correct response be taken if an MBE occurs to ensure proper system operation. This is detailed in Section 3.3.

2 Overview of EDAC and Scrub

When memory contents are subject to harsh environments, error correction and detection (EDAC) techniques are used to improve the reliability of the data. The most popular technique used for EDAC in semiconductor memories and the communications industry is the Hamming Code. Hamming error codes are well documented. [1] Only a small introduction is offered here. By exclusive OR’ing certain bits in an 8, 16 or 32-bit word, a unique value, commonly called a syndrome, can be created to correct a single-bit error. By exclusive OR’ing all the data bits and the syndrome bits, a double bit error can be detected, but not corrected. When a multi-bit error is detected, recovery procedures must be put in place. This may require a block of memory to be updated or the entire system to be reset.

The SMV512K32 has a 7-bit syndrome for every 32-bits of data. When data is written to the memory, the EDAC controller automatically calculates the syndrome and stores it along with the data. When the memory is read, the syndrome is read at the same time as the data and a calculation is performed. If a single-bit error (SBE) is detected, the data (with the error corrected) is output to the data bus (DQ[31:0]) pins and the MBE pin (if enabled for SBE indication) will go high. The system software can optionally write the correct data back to the SRAM location (or the software can wait and allow the optional scrub function to find and correct the upset to the memory). If a multi-bit error (MBE) is detected, the incorrect data is output to the DQ[31:0] pins, the MBE pin (if enabled for MBE indication) will go high, and external action may be required. Syndrome information is always kept internal to the SRAM and cannot be accessed.

In addition to correcting SRAM data output through normal operation, a memory scrub mode is available on this device. The purpose of scrub is to periodically read each address of the SRAM sequentially, effectively walking through the memory looking for and automatically correcting single-bit errors. The scrub engine has a 19-bit counter and only processes one word (32-bits) per scrub cycle. To scrub the entire SMV512K32 device once would take 524,288 scrub cycles. The periodic rate of the scrub read can be controlled two different ways: 1) master mode internally generates a programmable periodic clock or 2) slave mode relies on an external set of signals to commence a scrub cycle. At the fastest scrub rate available in master mode, this would typically require 524,288 × 1100 ns = 0.58 s.

Scrub is more important for SRAM memory locations that are not frequently overwritten because this potentially allows uncorrected errors to accumulate over time. An uncorrectable error would occur if there were more than one erroneous bit in a single 32-bit word. In applications where the entire memory contents are overwritten frequently (i.e. when the entire memory is used as a frame buffer), the frequent overwriting of the memory functions as a scrub of the SRAM. In this case, just enabling EDAC (without scrub enabled) will ensure protection against single bit upsets of the data.

It is recommended that the user software initialize the entire SRAM memory by writing all memory locations, even if the entire 16 Mb of memory is not being used in the application. To ensure that the syndrome bits contain non-random information, each memory location must be written with EDAC enabled to create the correct syndrome bits. Otherwise the user will encounter single bit and multi-bit errors when the scrub function is used by reading unwritten locations which contain random values for the data and syndrome bits. Over time, the scrub engine works its way through the entire memory space and does not differentiate between addresses with valid data and addresses which have never been written. During power-up, the SRAM data will be in a random state so any address which has not been written (either during initialization or a normal data write to the address) will have randomly set data and syndrome bits and will almost always fail the EDAC check.

2.1 Master Scrub Mode

When master scrub mode is selected, the SRAM is set up to perform the scrub function periodically. During scrub, an SRAM address is read outside of normal program operation. The bus master enables a memory scrub periodically by placing the SRAM into master mode (pin MSS = 1). The scrub rate can be programmed between approximately 0.433 kHz and 888 kHz. During master scrub operation, the SCRUBZ and BUSYZ signal are used to tell the interface device (i.e. FPGA or MPU) when the memory is busy (in a scrub cycle) and is not available for normal operation.

2.2 Slave Scrub Mode

When slave scrub mode is selected, the bus master must perform the operation of each scrub cycle. By placing the SRAM in slave mode (pin MSS = 0), an external interface device periodically forces scrub cycles by pulling the SCRUBZ pin low to perform a read operation (and repair of single bit errors) over the entire memory space. This method can be used by an operating system fairly easily, however. If a periodic task is completed before the task window expires, the processor can issue a scrub command while waiting for the next periodic task.

2.3 Hybrid Scrub Mode

A third “hybrid scrub” method, detailed in Section 3.2, is a combination of master scrub mode and slave scrub mode. In this method, smaller portions of memory, say 1k words, can be scrubbed in master mode periodically while other portions can be scrubbed in slave mode. This keeps the overall scrub rate high while not inordinately taking a large amount of time away from normal operation. Hybrid mode also eliminates the need for the bus master to issue each scrub cycle. Instead it only needs to determine when to start a hybrid scrub and how long to allocate for each hybrid scrub operation.

2.4 Manual Scrub Operation

Lastly, if writing to the entire memory space requires too much time, single bit error correction can be done by a method that does not utilize the internal scrub feature of the SRAM. The bus master can periodically read through the necessary memory space and make manual updates if single bit errors are detected. Alternatively, each location can be read and then written to with the value just read. This simplifies the bus interface and gives the user manual control of the scrub operation.

3 SMV512K32 Functionality

The full description can be found in the SMV512K32-SP data sheet. This section provides supplemental information on how use the SRAM in a final application.

3.1 Register Summary

The SMV512K32 has one control register which is used to set up the SRAM for EDAC control. Read and write access to this register is via the device address pins (A[18:0]). This register can also be used to read the internal address counter to determine the address location of the single or multi-bit error following a master mode scrub cycle which detected an error. Refer to the device data sheet for more information. Table 1 lists all the register bits associated with EDAC, scrub and memory error handling. The EDAC Control Register is not initialized and should be written to known values at system start-up regardless if scrub operation is to be used or not. EDAC Control Register bit settings for scrub rate and BUSYZ to SCRUBZ delay are only used for scrub master mode.

Table 1. EDAC Control Register

EDAC CONTROL REGISTER		
BIT POSITION(S)	**DESCRIPTION**	**OVERVIEW AND USE**
A[11]	Scrub enable bit (active low)	Enables Scrub functionality (A[8], EDAC bypass bit, must be set to zero to use scrub). Scrub is enabled when A[11] = 0 and disabled when A[11] = 1.
A[10:9]	Control word access bits. See device data sheet for more information concerning writing the EDAC Control Register.	Used to enable access to the EDAC Control Register. A[10:9] = 00 selects a write of the EDAC control register. A[10:9] = 01 selects a read of the EDAC Control Register. A[10] is also used for address counter reads in conjunction with A[7].
A[8]	EDAC bypass bit	Disables EDAC and Scrub when A[8] = 1. Enables EDAC when A[8] = 0, and enables EDAC and Scrub when A[8] = 0 and A[11] = 0.
A[10:7]	Used only for address counter reads. Note A[7] has two different purposes.	Setting bits A[10:7] = 1xx1 during an EDAC Control Register read enables reading of the internal address counter to determine the address of the last SRAM location scrubbed. The address is output on DQ[18:0].
A[7:4]	BUSYZ to SCRUBZ delay select	Sixteen different values of delay can be set to values from 80 ns to 1600 ns for master mode scrub operation. See SMV512K32-SP data sheet for more information.
A[3:0]	Scrub rate select	Twelve different scrub rates can be selected between 0.433 kHz and 888 kHz for master mode scrub operation. See SMV512K32-SP data sheet for more information.

Timing of the control signals required to write and read the EDAC control register is not critical as long as the order of assertion and de-assertion of the signals is preserved. This can be accomplished by using an FPGA state machine or by processor control of general purpose input/output pins as shown in Figure 1 and Figure 2. Refer to the SMV512K32-SP data sheet for more information.

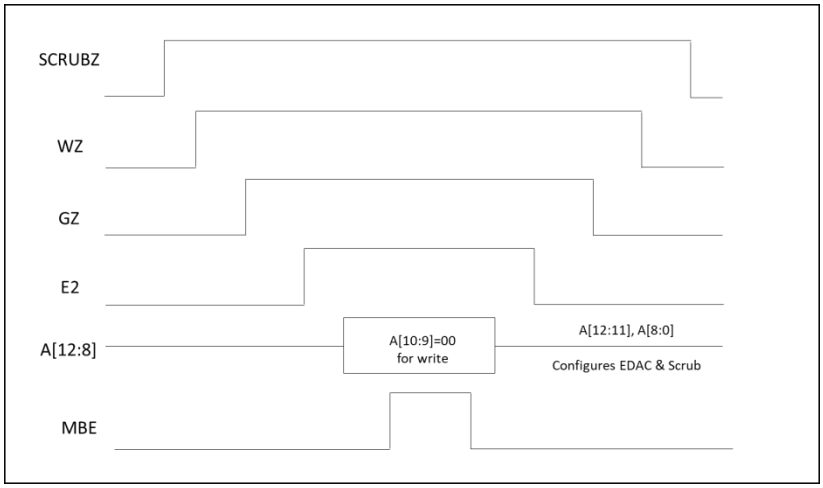

Figure 1. Writing the EDAC Control Register

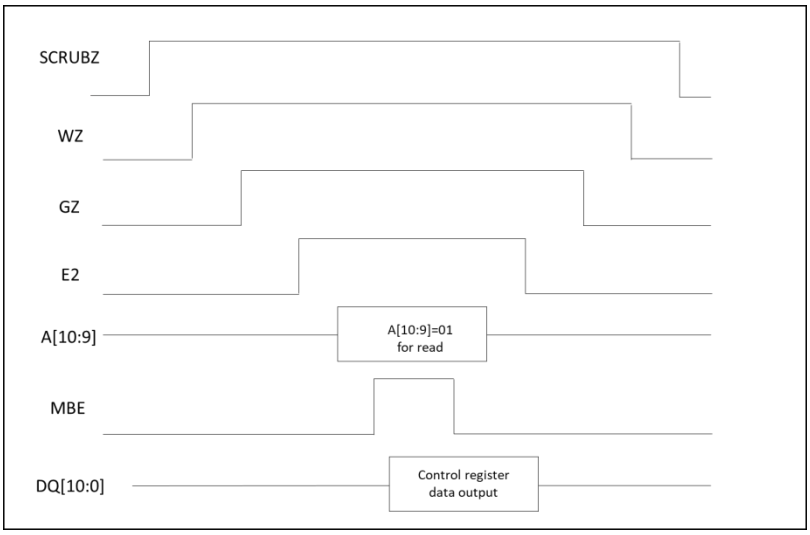

Figure 2. Reading the EDAC Control Register

3.2 Determining Whether to Use Master or Slave Scrub Mode

The MSS pin on the SMV512K32 determines the scrub mode. MSS = 1 puts the part in Master mode and the SMV512K32 generates two output signals: BUSYZ and SCRUBZ. MSS = 0 puts the part in Slave mode and the external device (FPGA or MPU) must generate the SCRUBZ signal and monitor the BUSYZ signal in accordance with the device data sheet. Master scrub mode operation is shown in Figure 3. The interface device must monitor the Master SRAM's BUSYZ signal to determine if the SRAM device is available. In a system with more than one SRAM, master scrub mode can be enabled on memory devices which are not currently being accessed.

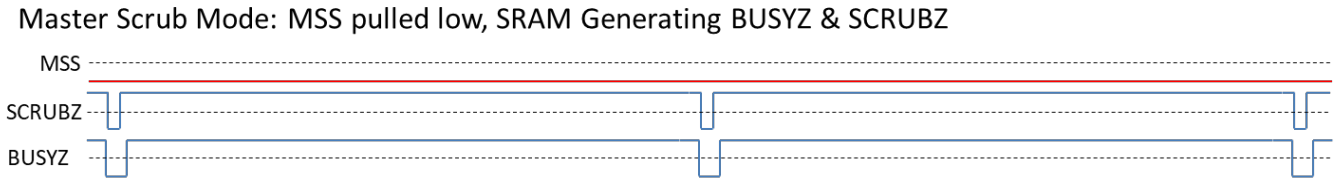

Figure 3. Scrub Master Operation

Requiring a bus master (i.e. FPGA or MCU) to prevent SRAM access when the BUSYZ signal goes active may require too much overhead. In this case, slave mode is recommended and is shown in Figure 4. Slave mode requires the bus master to generate the SCRUBZ pulses, but it avoids potential bus conflicts when both the SRAM and bus master vie for control of the data bus. Using scrub slave mode is recommended for a system with available idle time which can be used for memory scrub, such as software based on an operating system.

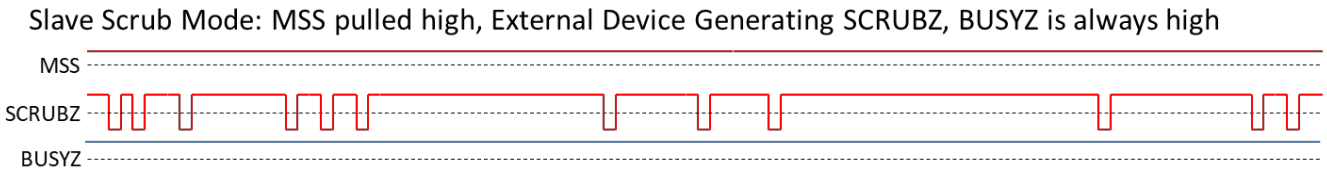

Figure 4. Scrub Slave Operation

A third “hybrid scrub” method, shown in Figure 5, can be used to periodically process smaller portions of memory. This keeps the overall scrub rate high by allowing scrub when memory access is not required such as in a stand-by mode or when multiple SRAM devices are present.

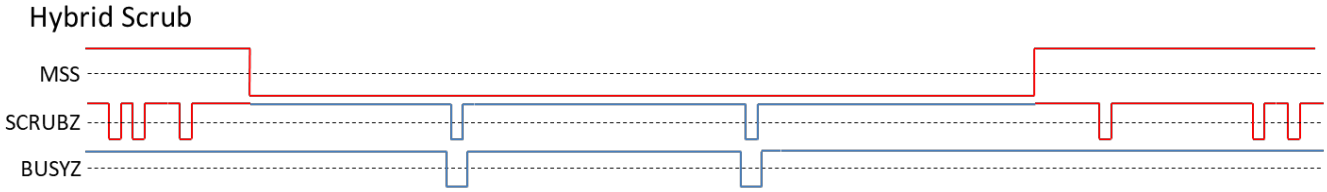

Figure 5. Scrub Hybrid Operation

Figure 6 provides additional information to explain the use of the hybrid scrub mode. Maintenance is defined as the time when the memory is not being accessed. Flare is defined as any time high levels of radiation have been detected and memory errors are more likely. Hybrid mode combines both master and slave modes and can be used in more complicated systems.

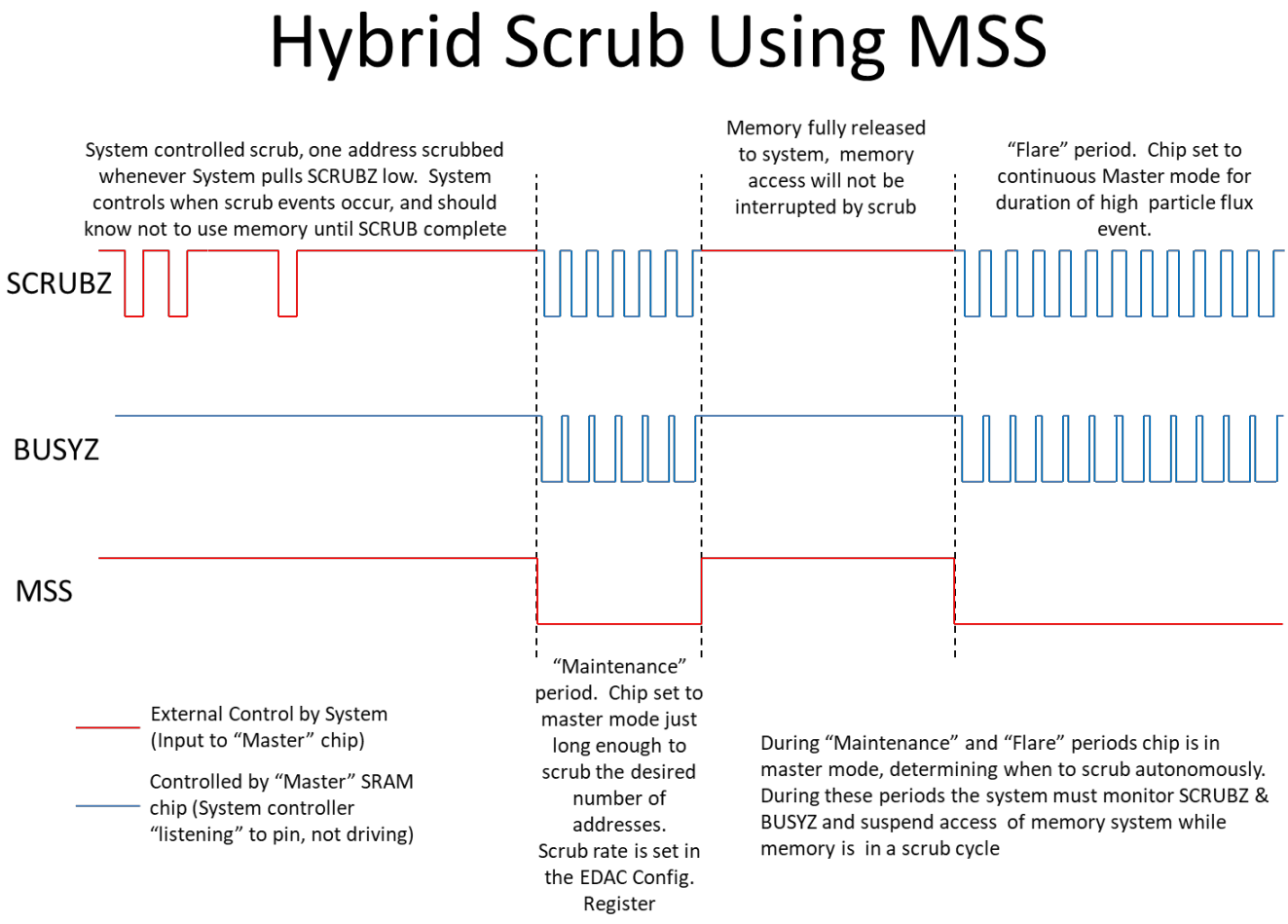

Figure 6. Hybrid Scrub Operation

3.3 Proper Response to MBE Indication During Scrub

As mentioned earlier, when an MBE occurs and is set to detect a multi-bit error, this is most likely an indication of high levels of memory upset and therefore an unrecoverable event. If this occurs in program memory, the system will need to be rebooted. In the case where the SRAM purely stores data, the suspect data may have to be restored. When an MBE occurs and is set to detect a single bit error, the memory location will be corrected, and no further action is required. If the scrub rate is properly selected, single bit errors should not accumulate into multi-bit errors. If multi-bit errors are detected during normal read accesses, the scrub rate may have to be increased. It is recommended that the MBE pin output be disabled (by setting GZ = 1) during scrub operation unless the entire memory has been initialized at start-up. This allows single bit errors to be corrected while ignoring multi-bit errors on uninitialized memory.

3.4 Proper Operation Following an MBE Indication

As noted in the SMV512K32 data sheet, proper care must be taken following an MBE indication if system reset is not required. When the MBE pin is driven high by the SRAM, the user must:

- disable the SRAM (either E2 going low, E1Z going high, or both).
- wait until the MBE pin is pulled low by an external resistor.
- change the address pins to read a different SRAM location before the next memory read.

If this procedure is not followed, there is a possibility that the high state of the MBE pin can be interpreted as an unintended write to the EDAC Control Register.

3.5 Verifying Error Handling Code

This section describes a way to inject errors to prove the memory recovery strategy is working as planned.

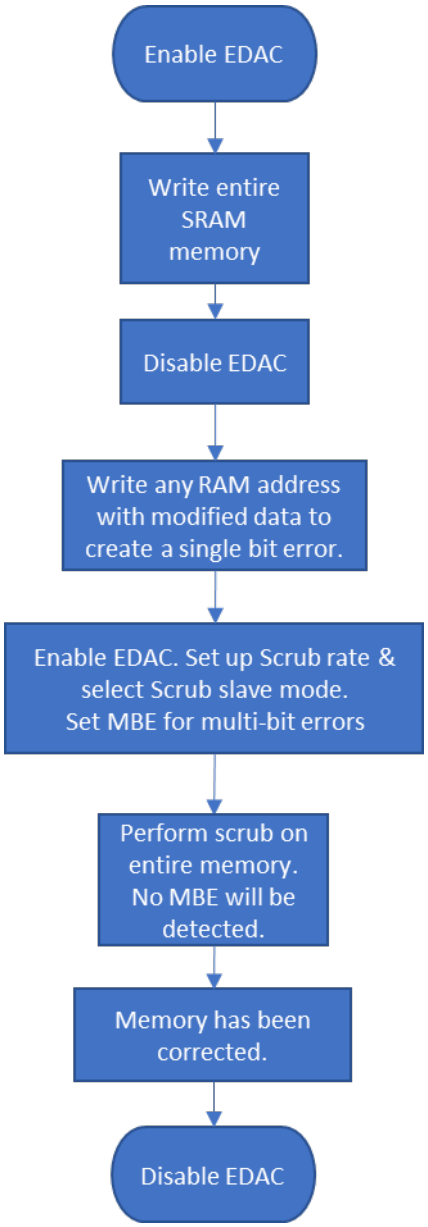

Figure 7. Example Manual Procedure to Check Scrub is Working

3.6 Monitoring Errors

When a memory error has been detected and interrupts are enabled on a controlling MCU or FPGA and triggered by the MBE pin going active, an interrupt subroutine, such as the one shown in Figure 8, can be used to isolate the cause of the error and report the error count.

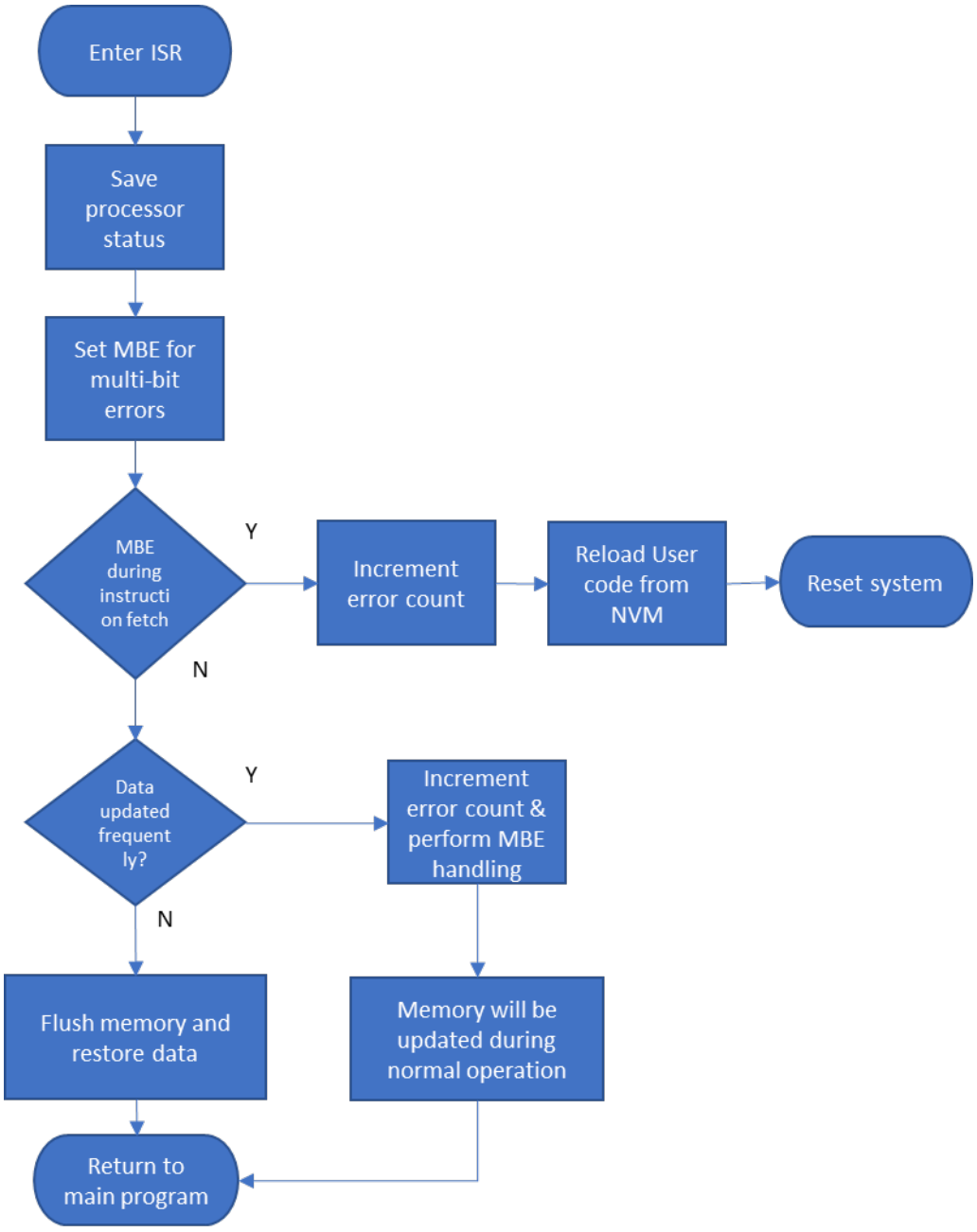

Figure 8. Example ISR to Process Errors

4 Working With Systems Containing Multiple SRAMs

Some systems require more than one SRAM device if memory requirements exceed the size of one SRAM, or for redundancy. For these systems, different scrub methodologies can be used. In the first scenario, one SRAM is set up for scrub Master mode and the others are set up for scrub Slave mode requiring minimal intervention from the system controller (FPGA or MPU). As SCRUBZ and BUSYZ are output signals from the SRAM in Master mode, multiple SRAMs can have their SCRUBZ pins connected together, but only one Master can exist on this common connection at any time. All other SRAM devices on the common SCRUBZ line must be in Slave mode. BUSYZ is always an output from each SRAM, so the BUSYZ pins must never be tied together. The BUSYZ from an SRAM in Master mode must be monitored by the system controller to determine if the SRAM is available for access.

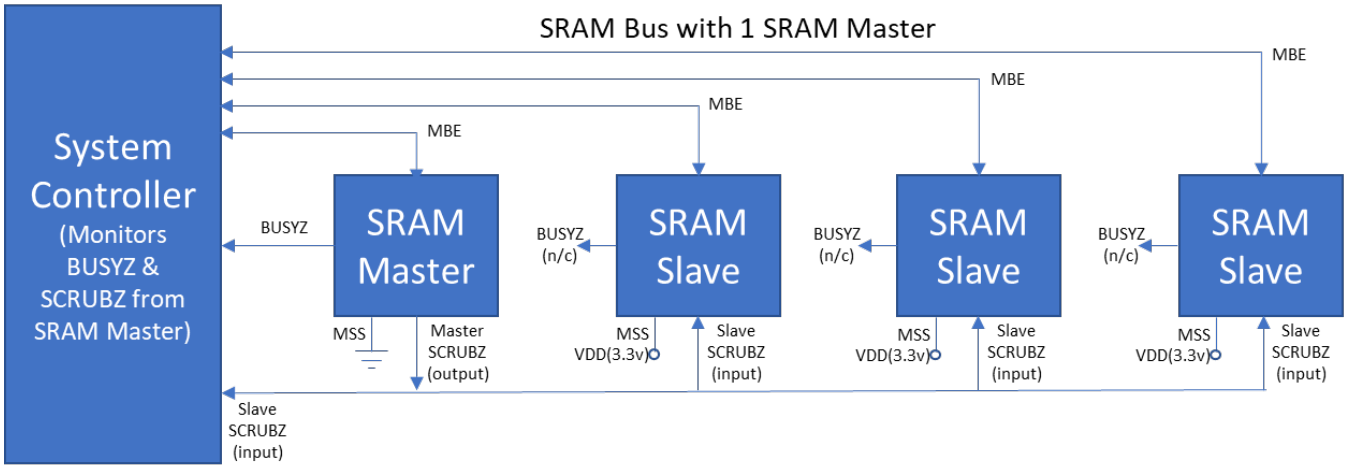

Figure 9. Multiple SRAMs With a Single SRAM in Scrub Master Mode

In the second scenario, the system controller (FPGA or MPU) is used as the bus master. Scrub slave operation is selected for all SRAM devices. As SCRUBZ is an input to the SRAM in Slave mode, multiple Slave mode SRAMs can have their SCRUBZ pins connected together, but one and only one bus master (SRAM in Master mode or system controller) must exist on this common connection to signal when the Slave SRAMs are to perform a scrub operation. All other SRAM devices on the common SCRUBZ line must be in Slave mode. BUSYZ is always an output from each SRAM, so BUSYZ pins must never be tied together. In Slave mode, BUSYZ is always pulled high regardless of the SCRUBZ state. As such, the Slave mode SRAM BUSYZ output is a meaningless signal and should be ignored (pin left floating).

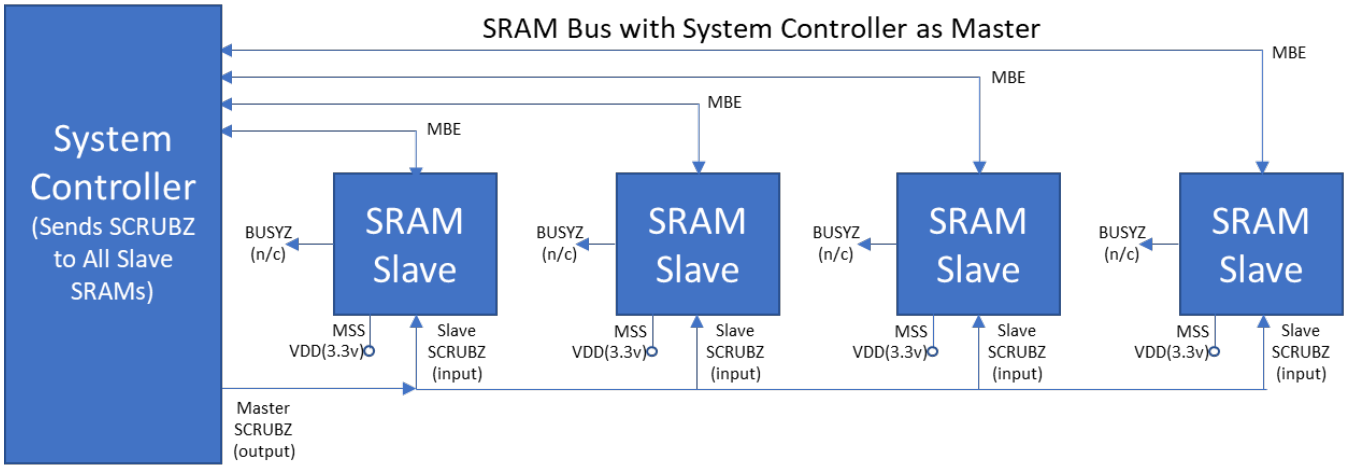

Figure 10. Multiple SRAMs With a System Controller as the Master

In the last scenario, the scrub operation can be controlled by either the bus master (FPGA or MPU) or one of the SRAM devices making use of the hybrid scrub mode mentioned earlier. As SCRUBZ and BUSYZ are output signals from the SRAM in Master mode and input signals to the SRAM in Slave mode, multiple SRAMs can have their SCRUBZ pins connected together, but only one Master can exist on this common connection at any time. All other SRAM devices on the common SCRUBZ line must be in Slave mode. BUSYZ is always an output from each SRAM, so BUSYZ pins must never be tied together. The BUSYZ from the SRAM in Master mode must be monitored by the system controller to determine if the SRAM is available for access. The BUSYZ from the SRAMs in Slave mode should be left floating.

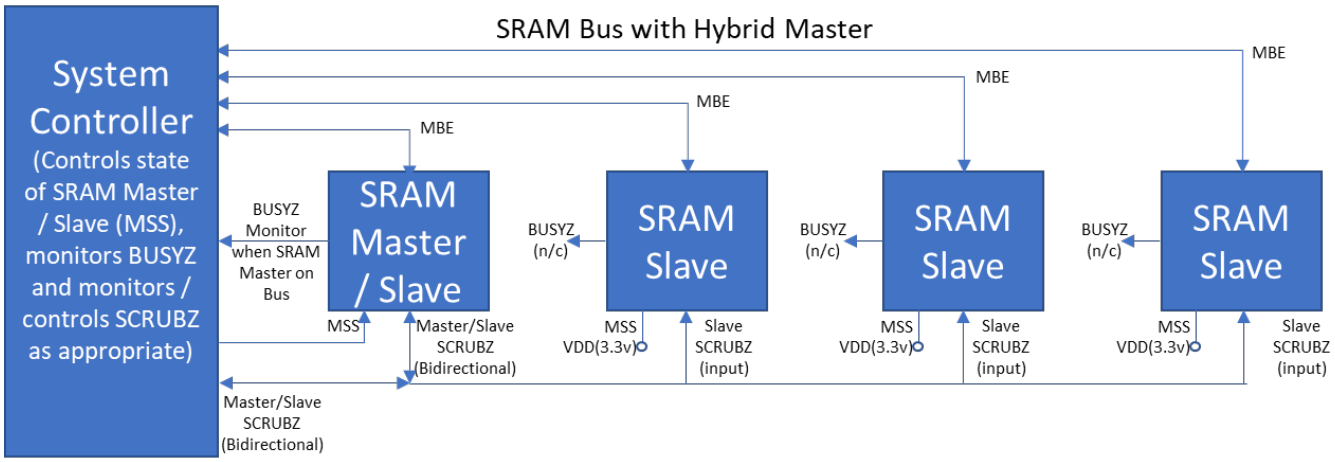

Figure 11. Multiple SRAMs in a Hybrid Scrub Configuration

5 Summary

The SMV512K32 has EDAC and scrub features which are essential to combating single event upsets that will occur in space environments. This application note has provided several example operations for setting up these features and testing them. The user should be able to enable these features in the application.

6 Common Questions and Issues

6.1 *What is the Right Scrub Rate?*

This will depend upon the type of radiation environment the part will operate and the power consumption sensitivity. The important thing is to perform scrub frequently enough to prevent the accumulation of single-bit errors in each SRAM used in a system. The following table shows some examples. Each scrub cycle takes approximately 330 ns so a high scrub rate will impact system performance. For a typical space application for geosynchronous orbit at solar minimum, a scrub rate of 7 kHz should be often enough for continuous operation for more than 10 years.

Table 2. Sample Scrub Times

SCRUB RATE (kHz)	PROCESSOR BUS FREQUENCY (MHz)	BUS CYCLES BETWEEN SCRUBS	SCRUB OVERHEAD 330 ns / (Bus Cycle Time × Number of Cycles)	TIME TO SCRUB ENTIRE ARRAY (seconds)
888	50 (20-ns bus cycle)	55	30%	0.58
111		450	3.75%	4.72
14		3550	0.48%	37.45
7		7100	0.24%	74.6
3.5		14,300	0.12%	149.8
0.433		115,000	0.015%	1210.8

6.2 *Why Not Perform Full System Reset Whenever an MBE Occurs?*

In most cases an MBE signifying a multi-bit error justifies a full reboot since the error is uncorrectable and may reside in code space or stack space. However, if an MBE occurs in data RAM, the user may decide this is not critical and may rely on software to refresh this area without a full boot.

7 References

(1) Hamming Code Calculator. www.ecs.umass.edu/ece/koren/FaultTolerantSystems/simulator/Hamming/HammingCodes.html
(2) SMV512K32-SP Data Sheet

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Original (July 2018) to A Revision	Page
• Changed the title.	1

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

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