FRONTGRADE

FACT SHEET

UT32MOR500

32-Bit-Microcontroller

4/20/2026

Version #:1.1.0

Frequently Asked Questions

This document is intended to answer common questions about the UT32M0R500. For more in-depth information, please consult the Datasheet or the Functional Manual, both of which can be found here: https://frontgrade.com/product/ut32m0r500

Q. What is the processor core of the UT32M0R500?

A. The UT32M0R500 has a 32-bit RISC based ARM™ Cortex® M0+ processor.

- Compact Thumb-2 Instruction Set
- Multi-Level Interrupt support through the Nested Vector Interrupt Controller (NVIC) enables advanced real- time processing
- Memory Protect Unit helps prevent application code and data corruption improving reliability
- Wake Up Interrupt Controller interface enables event drive power up
- Debug Access Port

Q. How fast is the UT32M0R500?

A. The UT32M0R500 has an internal 50MHz oscillator and supports an external clock source and crystal oscillator.

Q. What peripherals are included?

A. The following peripherals are included:

- 2x CAN 2.0B Controllers
- 2x UART
- SPI
- 2x I2C
- JTAG
- 4x General purpose timers
- 3x PWM
- Watchdog Timer
- Real Time Clock
- 48x GPIO (21 dedicated)
- 8x Hardware Interrupts (shared with GPIO)
- 12-bit ADC 100 ksps with PGA (16 Single Ended or 8 Differential Channels)
- 1 mA Precision Current Source
- 2x 12-bit DACs
- 2x Analog Voltage Comparators
- Temperature Sensor

Q. What Firmware Development Environment is recommended?

A. Frontgrade officially supports the Keil® µVision® IDE for firmware development. Keil® offers a free version of the software that has a code size restriction of 32Kbytes, or paid versions that can be found at https://www2.keil.com/mdk5. Additionally, to perform debugging Frontgrade recommends the use of the Keil® ULINK2® debug adapter. Both the IDE and the debug adapter are sold by Keil®, and are not included with the UT32M0R500.

Q: Will one of the other Keil® ULINK® debuggers work?

A: Yes, any debugger on the following link will work, the IDE settings (see above) will need to be configured to match the chosen debugger: https://www2.keil.com/mdk5/ulink

Q: What is provided inside Frontgrade's Software Development Kit (SDK)?

A: Inside of the SDK is:

- The Standard Peripheral Library files and UT32M0R500 Specific ARM files required for firmware creation
- Example application files for various peripherals to jump-start firmware development
- Documentation:
    - An application note about creating projects in the Keil® µVision® IDE
    - The Quick Start Guide full of useful product information
    - The UT32M0R500 EVB Users' Guide provides information on how the evaluation board functions, and how to program/debug the UT32M0R500

Q: How much memory does the UT32M0R500 have?

A: The UT32M0R500 contains 96KB of embedded Dual Port SRAM with EDAC + Scrubbing, and 64Mb of internal Flash Memory. The flash memory allocates 384KB for user application firmware in four independent 96KB sections, plus an image override sector.

Q: What is the difference between programming and debugging the UT32M0R500?

A: The UT32M0R500 can have the firmware images (dedicated Flash Memory) programmed via the UART0 , CAN0, or JTAG interfaces. Once the application is loaded into a firmware image, it can be loaded into SRAM by changing the boot configuration pins and resetting the device. The application will remain in Flash Memory regardless of resets or loss of power, and can be erased or reprogrammed through the bootloader or JTAG.

The UT32M0R500 uses the Keil® ARM™ development tools to debug software. The development tools interface with the UT32M0R500's JTAG connections to load code directly into the device SRAM. The development tools then allow the user to free-run, stop, or step through application code using the JTAG connection. Because the application code is loaded into SRAM, resetting or loss of power to the UT32M0R500 will clear the SRAM.

Application Notes for programming via UART0 (App-Note-UT32M0R500-UART-Flash-Download.pdf), via CAN0 (App- Note-UT32M0R500-CAN-Update-Protocol.pdf), and via JTAG (App-Note-UT32M0R500-and-Keil-Flash-Download.zip) can be found on the product page.

Q: What is the recommended pullup/pulldown resistor configuration for the JTAG pins?

A: When being used for debugging, 10KΩ pullup resistors should be placed upon the TRST, TMS, TCK, and TDI signals, with TDO floating.

When JTAG is NOT being used (Ex. During Flight) TRST MUST have a 10KΩ pulldown resistor, TMS, TCK, and TDI should stay pulled up with 10KΩ resistors, and TDO should stay floating.

Q: What operating systems run on the UT32M0R500?

A: Both RTX and FreeRTOS runs on the UT32M0R500.

Q: What is the state of the UT32M0R500's I/O's after a reset?

A: Specific I/O's become unknown during a Power on Reset (POR), and then change to a known value after the POR. For a complete list, please consult the UT32M0R500 datasheet pinlist.

Q: What is the weight of the UT32M0R500?

A: The Column Grid Array (CGA) package option weighs 4.116 grams. The Land Grid Array (LGA) package option weighs grams.

Q: Can the sectors of the NOR Flash Memory that are not allocated for user application firmware be used for other things?

A: Yes, users can use sectors 0-7 and 17-141 to store data. Note that the NOR Flash must remain powered off 90% of the time in order to meet radiation requirements.

Q: Why are some Reserved pins marked "NC" and others "NUIL"?

A: Pins listed as RESERVED in the datasheet have one of two descriptions.

- NUIL, or "Not Used Input Low" are reserved input pins, and MUST be tied to VSS through a >10kΩ±10% resistor
- NC, or "No Connect" are reserved output pins, and MUST be left floating

Q: What's the minimum pulse width to trigger an external interrupt?

A: ~12ns for GPIO level interrupt.

Q: ADC input settling time vs LSB accuracy?

A: The datasheet specifies, depending on the device, 12-28 us settling time, but must take into account the ADC_SEQDLY requirements as specified in the reference manual, also see question below: ADC conversion time per channel.

Q: How many PWM outputs?

A: There are only 3 PWM outputs, with the 4th output only used in the paired configuration.

Q: Writing to the PWM function directly when changing the duty cycle disables the PWM interrupt: PWM_ChanInitStruct.SwitchComparator and PWM_SetChanConfig(PWM, CHAN_PWM_0, &PWM_ChanInitStruct).

A: To get around it, write to the register directly: PWM->PWM_Chans.COMP = duty_cycle_value.

Q: If two transactions to the SPI TX FIFO, would the Chip Select (CS or SSN) de-assert in between transactions, or does the Chip Select remain asserted across consecutive transactions?

A: Based on the SPI_FIFO_DEPTH, the CS will remain asserted from the start of the transfer until the end of it. So if the SPI_FIFO_DEPTH is set to 2, then the CS will remain asserted for the two transactions.

Q: Why are pad sizes different between CCGA and PBGA?

A: The different pad sizes between the CCGA and PBGA are that the CCGA columns are tall, more rigid joints for ceramic-package CTE mismatch, so they need more anchoring area and solder volume; PBGA balls collapse more and self-center on smaller lands.

Q: QML flow for the UT32M0R500?

A: UT32M0R500 will only be QML Q/Q+ due to fab location of the NOR Flash die. We are planning on having a future generation that can qualify for QML V

Q: What are ADC errors and their meanings?

A: Bit 14, DATA_ERROR:

This error is set (1) when any of the other ADC Error bits are set:

- COI_OVER
- SINC4_OVER
- DSM_OVL_FLAG
- TRIG_UNDER

This Flag is cleared (0) when read. If the other bits stay set, this bit will reset at the end of a conversion. As this flag is a logical OR of all other flags, use it to quickly check for errors for the respective ADC channel. If it is set, investigate the other flags.

Bit 23, COI_OVER:

This bit is set (1) when the channel in question is using the COI3 (3rd order Cascade of Integrators) filter and the data stream coming out of the Delta-Sigma Modulator has:

- Exceeded the maximum value. If the number of 1's sent from the DSM to the COI3 filter has repeated a specific number of times (based on the chosen OSR), indicating the input has exceeded the data range, the bit is set.
  - If users choose an OSR value not listed inside of the recommended table, the full scale data range is any input value below 7FF.
- Exceeded the minimum value. When the ADC is seeing near-zero or negative ADC values, the repeated 0's sent from the DSM to the COI3 filter will cause the COI_OVER bit to set, indicating the voltage has gone below the minimum input value.
  - This occurs for some very small but positive inputs, at and below 0.007V or 7mV with single-ended channels. There is a software workaround (see UT32M0R500 the errata section in reference manual), which is to ignore the COI_OVER error when the input voltage is very low and not negative.

The bit is cleared (0) if the filter is within the min and max thresholds.

Aside from the errata (see UT32M0R500 reference manual) concerning the incorrect low voltage threshold, this bit being set indicates that the input voltage is outside the read-able threshold values or the DSM has overloaded (bit 25 would be set), and data should be ignored.

For example, if the ADC channel is configured with a Gain of 4 V/V, this should work out to a full scale input voltage of 0.35 V (1.5/4). When ramping up the input voltage, COI_OVER error could start showing at 0.3758 V because this voltage is 0.0258 V (1.7%) above 0.35 V. The input voltage has exceeded the valid input range; hence the error flag gets set.

Bit 24, SINC4_OVER:

This bit operates the same as the COI_OVER bit, but for a channel using the SINC4 (4th order Sinc) filter.

The bit is set (1) when a voltage outside the normal range is detected. As before, the 'normal range' for this filter is based upon the OSR.

The bit is cleared (0) if the filter is within the min and max thresholds.

This bit will only set when the input exceeds the input voltage range. As the SINC4 filter relys on previous reads, this would indicate either the DSM modulator has overloaded multiple times (bit 25 would be set), or the input voltage has been out of range for multiple samples. No external reset is required.

Bit 25, DSM_OVL_FLAG:

Because the Delta-Sigma Modulator can become saturated and potentially unstable, the ADC circuitry includes a hardware reset that enables the DSM to recover. If the DSM output bitsream repeats the same value too many times (determined by STAB_CTRL.DSM_OVL_CNT), the Delta-Sigma Modulator is said to be overloaded. The overloaded DSM will automatically reset without affecting the COI3 or SINC4 filters or the ability to read from those filters. The number of ADC_CLK cycles the reset takes is determined by STAB_CTRL.DSM_OVL_RST. The UT32M0R500 ADC block has not exhibited incorrect indication of overload, or tendancy toward instability during characterization. Nevertheless, the reset feature was included as a fail-safe backup.

This bit is an copy of the STAB_CTRL register's DSM_OVL_FLAG (bit 16). The DATA register bit remains set (1) until the STAB_CTRL register is read from to clear that register's DSM_OVL_FLAG. Both bits are clear (0) if the DSM has not overloaded (experienced a bitstream repat greater than the DSM_OVL_CNT value).

As the hardware includes automatic DSM recovery once the threshold is crossed, users don't need to implement any external reset.

For Example, with a gain of 4V/V, this should work out to a full-scale input voltage of 0.35 V (1.5/4). A value of 0.472V becomes greater than 1.5V, hence the DSM_OVL_FLAG error flag is set.

The DSM_OVL_FLAG will trigger if there's a stream of repeated 1's **OR** a stream of repeated 0's equal to the value in the ADC->STAB_CTRL.DSM_OVL_CNT bits. So as long as the input voltage to the pin does not exceed the absolute maximum rating (VDDA±0.3V), a high gain setting will never damage the part.

Bit 26, TRIG_UNDER:

This bit is set (1) to indicate that a trigger was commanded while a conversion was mid-process. Note that a this trigger will re-start the conversion and set this bit.

This bit is cleared (0) to indicate the last conversion did not receive a mid-conversion trigger. At the start of a new conversion (first trigger, not a mid-conversion trigger) this bit is cleared.

As long as the user is waiting for conversions to finish, they should not see this bit set itself.

Q: What’s the relationship between OSR and resolution?

A: The OSR determines how many samples are taken at an oversampled rate (12.5MHz nominal) by the Delta Sigma Modulator (DSM). Fewer samples from the DSM allow for a higher sample rate (as there are fewer bits out of the DSM to collect). However, there's a probability of uncertainty around each bit depending on the level of noise in the input signal. Because there aren't as many bits sampled with a low OSR, each bit that is affected by noise as a greater affect on accuracy of the sample. So when using a high OSR, the samples take longer, but the data has a much higher chance of being accurate (as bits affected by noise have less sway over the final output code). A high OSR isn't the only way to achieve a 12-bit signal, as other factors can improve signal input such as the signal noise, keeping the PGA gain at a nominal 1, and using a differential input versus a single-ended input. It's best to chose the ADC settings and setup based upon the quality of the input signal.

Here's Table 3 from the attached appnote:

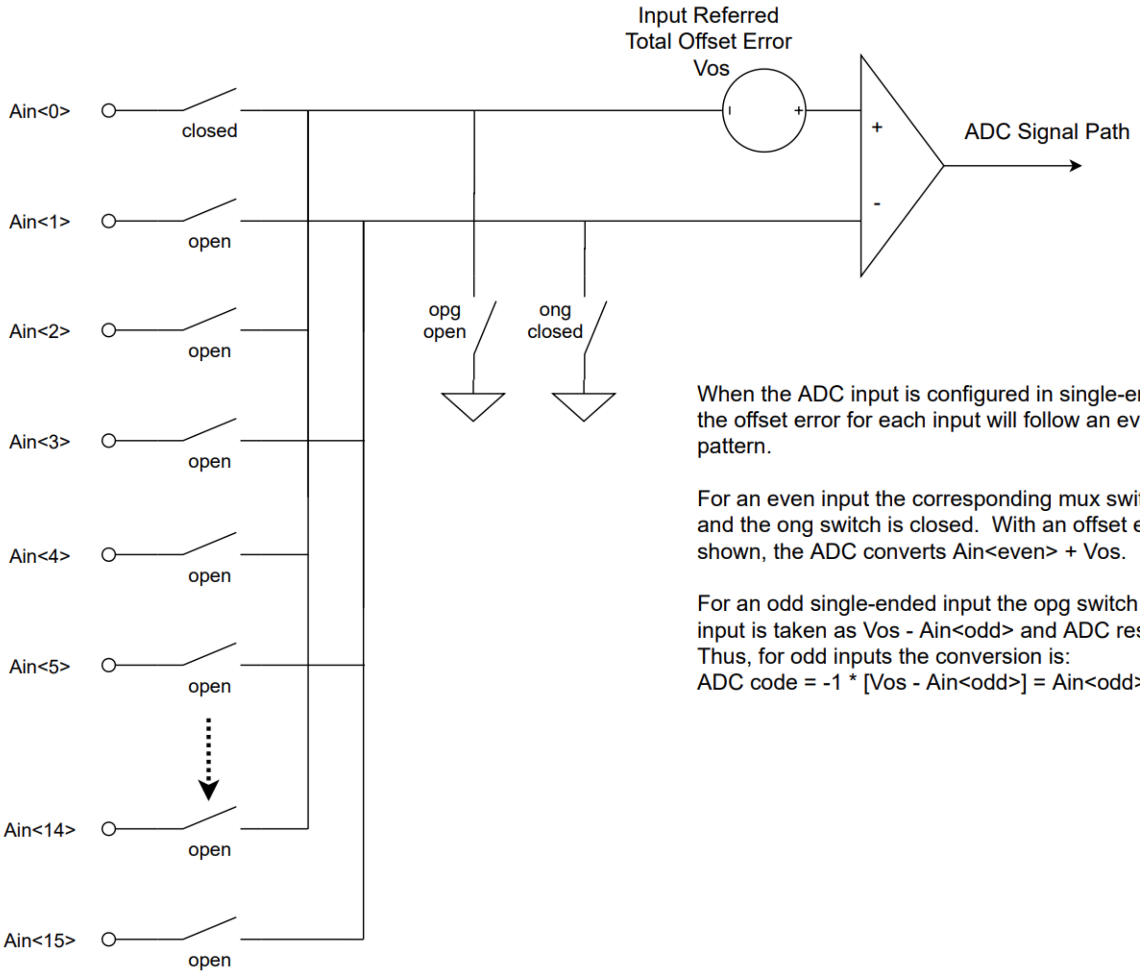
Full-scale input, single-ended [V]	PGA Gain [V/V]	Output Data Rate [SPS] (based on OSR)	Noise, RMS	Noise Input [µVrms]	Effective Resolution [bit]	Peak Resolution [bit]
1.5	1	198.4K (63 OSR)	0.8 LSB	300	12	9.7
1.5	1	156.3K (80 OSR)	<1 LSB	<370	12	10.3
1.5	1	123.8K (101 OSR)	<1 LSB	<370	12	10.3
1.5	1	98.4K (127 OSR)	<1 LSB	<370	12	10.6
1.5	1	78.1K (160 OSR)	<1 LSB	<370	12	10.7
1.5	1	61.9K (202 OSR)	<1 LSB	<370	12	10.7
1.5	1	49.0K (255 OSR)	<1 LSB	<370	12	11.2

Q: ADC Voltage Reference?

A: The ADC makes use of the 2.4V low noise VREF. This is factory-trimmed during the testing to be as close to 2.4V as possible. The VREF is used in a variety of ways inside the ADC peripheral. However, the delta-sigma architecture provides several benefits associated with artifacts (noise, offset, amplifier gain, ect) that may be introduced into the signal path, where the drift in the voltage reference would be seen as offset and/or noise, depending on the amount of drift. The use of this type of delta-sigma ADC architecture is what allows for the use of a voltage reference that may drift larger than expected as compared to other ADC architectures.

Q: Even/odd channels have roughly the same offset error but with opposite polarity.?

A: Yes, it is in the design of the ADC front end. The diagram below shows where the offset error occurs relative to the Mux and PG.



When the ADC input is configured in single-ended mode, the offset error for each input will follow an even/odd pattern.

For an even input the corresponding mux switch is closed and the ong switch is closed. With an offset error of Vos as shown, the ADC converts Ain<even> + Vos.

For an odd single-ended input the opg switch is closed. The input is taken as Vos - Ain<odd> and ADC result is inverted. Thus, for odd inputs the conversion is:
ADC code = -1 * [Vos - Ain<odd>] = Ain<odd> - Vos

Q: ADC conversion time per channel?

A: Below is the conversion time equation per channel:

*ConversionTime (per channel) = [(Input Settling Time + Filter Sampling Time) * Number of Channels] - Input Settling Time*

Note: The equation accounts for the errata found where the input settling time of the first enabled channel doesn't occur, see Reference Manual for more info on the errata.

For example, using OSR = 127, SEQ_DLY = 14, OSC_DIV = 4, Core Clock = 50MHz, the following table is calculated:

Input Settling Time = ADC_SEQDLY * (ADC_CLK period) * 25 = 14 * (1/12.5MHz) * 25 = 28µs

Filter Sampling Time = ADC_CLK period * OSR = 80ns * 127 = 10.16µs

# Channels	Conversion Time Calculated (µs)	Conversion Time Measured (µs)	Difference (Calculated - Measured)(µs)
1	10.16	12	-1.84
2	48.32	48	0.32
3	86.48	84	2.48
4	124.64	120	4.64
5	162.8	154	8.8
6	200.96	190	10.96
7	239.12	225	14.12
8	277.28	260	17.28
9	315.44	300	15.44
10	353.6	335	18.6
11	391.76	370	21.76
12	429.92	410	19.92
13	268.08	445	23.08
14	506.24	480	26.24
15	544.4	510	34.4
16	582.56	545	37.56

Q: Internal temperature sensor?

A: The internal temperature sensor is read through the ADC. The input signal is read from two additional inputs to the Analog Mux (not AIN[15:0]) and must be enabled differentially with a gain of 16V/V. The register layout is the same as any other ADC channel and can be found in the functional manual ADC chapter.

Example code for enabling the temperature monitor channel:

```
ADC_TypeDef *ADC = (ADC_TypeDef *) ADC_BASE;
ADC_InitTypeDef ADC_InitStruct;
ADC_ChanCfgTypeDef ADC_ChanCfgStruct;

ADC_StructInit(&ADC_InitStruct);
ADC_InitStruct.SweepType = ADC_SWEEP_SINGLE;
ADC_InitStruct.OscillatorDivisor = ADC_OSCDIV_BY_4;
ADC_InitStruct.OBD_Delay = ADC_OBD_DELAY_0;
ADC_InitStruct.ModulatorSamples = 101;
ADC_InitStruct.SequenceDelay = 16;
ADC_InitStruct.ModulatorResetClocks = 8;
ADC_InitStruct.OverloadCounterThreshold = 30;
ADC_Init(ADC, &ADC_InitStruct);

ADC_ChanCfgStruct.UseDDF2 = FALSE;
ADC_ChanCfgStruct.Enable = ENABLE;
ADC_ChanCfgStruct.Gain = ADC_GAIN_16VperV;
ADC_SetChannelConfig(ADC, ADC_TEMP_CHAN, &ADC_ChanCfgStruct);
```

Q: ADC codes to temperature?

A: For converting the ADC Codes to a temperature:

y = mx + c
y is the ADC code
m is the slope (code to temp, will vary between parts)
x is the temperature
c is the y-intercept (offset, will vary between parts)

To find the slope, two ADC readings were taken at -45°C and 105°C:
m = dY/dX = (2797 - 2217) / (-45 - 105) = -3.8715

To find the y-intercept (x = 0°C), c = 2622.6

Once a value for m and c is known, you can solve for temperature:
x = (y-c)/m
x = (2254-2622.6)/(-3.8715)
x = 95°C

EXAMPLE Raw ADC readings with a 3.3V power supply:

Temp (°C)	Two's compliment ADC readings
105	2217
95	2254
85	2294
45	2448
35	2487
25	2525
5	2604
-5	2642
-15	2681
-35	2758
-45	2797

Q: Clock Source Configuration?

A: *The XTAL1/2 pins can be configured to work with either a crystal oscillator or with a single ended clock signal. In crystal oscillator mode (0x4), XTAL1 *is the crystal input and XTAL2 is the output feedback for the crystal. In single ended clock mode (0x6), use XTAL2 as the clock input and tie XTAL1 low through a pulldown resistor (it's a CMOS input).*

possible configurations are:

Clock Source	**CLKSEL**	**XTAL1**	**XTAL2**	**OSC_SHUTDOWN[2:0]**
Internal 50MHz Oscillator	LOW	Unused, so pulldown resistor to VSS	Unused, reset state is an output so leave floating	b'000
Crystal Oscillator	HIGH	Clock Input	Clock Feedback Output	b'10X
Single-Ended Oscillator (Recommended)	HIGH	Unused, tie low	Clock Input	b'11X
Single-Ended Oscillator (Alternative)	HIGH	Clock Input	Enabled output, leave floating	b'10X

Note: 'X' is don't care, depending on if the user wants to disable the internal oscillator while using an external clock. OSC_SHUTDOWN is the Clock Control Register.

Q: The BOOT_DONE bit set to 1?

A: The BOOT_DONE bit signifies that the μC is done downloading firmware from the NOR Flash into SRAM, and it can now switch over to the external clock (if the CLKSEL pin is HIGH). The BootROM will set this bit before user firmware starts executing. However, if the user is trying to debug their uC, entering Keil's JTAG debug mode doesn't trigger the bootROM, so the BOOT_DONE bit doesn't get set. This is only a problem if the user is trying to debug the part while using an external oscillator. If the BOOT_DONE bit doesn't get set, the part being debugged will use the internal oscillator.

Q: There are two ball grid array packages, the Ceramic BGA (CBGA) and Plastic BGA (PBGA).

A: The CBGA uses 10Sn/90Pb solderballs.

The PBGA uses 63Sn/37Pb solderballs.

Q: SPI tsu(ssn)?

A: T(SU), aka the time from the CS line going low to the first clock pulse, depends on the baud rate of the part. Typically, the first clock edge will occur after 1.5 clock periods have expired. Ex: Clock divider of 4 means tSCK = 80ns, so the first clock edge would happen 120ns after SSN goes low.

Q: GPIO pins state at initial power on?

A: GPIO pins powered on is captured in the UT32M0R500 datasheet pinlist, where two columns, Value During POR and Reset Value After POR, detail how those pins will behave as the UT32M0R500 is powered on. During a pre-Power-On-Reset condition, the GPIO pins are in an unknown state. Possible variation could be due to power-on time between boards, part variation, ect.

Q: Is there a requirement (min/max) on the 3.3 V ramp-up at POR?

A: No requirement, but slower ramp rates increase the time in the Pre-POR state.

Q: Are there any sequencing requirements between VDD and VDDA? For example, do they need to be turned on together within N ms? Does VDD need to be turned on first? Etc.

A: No, VDD and VDDA can be powered on in any order with any delay between power on. The device will remain in the Power-On-Reset mode until both supplies are on.

Q: What is the minimum required time to assert RSTN?

A: RSTN should be held low for at least 3 clock cycles (60ns)

Q: What is the operation of the UT32M0R500 SRAM EDAC and Scrubbing Controller?

A: PORT B Side - Scrubber

Enable and configure scrubbing (SCRUB_EN) over the entire SRAM address range.

The SBE_COUNT_B and MBE_COUNT_B will increment if any SBEs or MBEs, respectively, are detected during scrubber SRAM read accesses (via PORTB).

Detected SRAM SBEs will be corrected (in-situ) by the EDAC as part of the scrubber read-write cycle.

Detected MBEs will not be corrected and will not generate an interrupt.

PORT A Side – Processor

The SBE_COUNT_A and MBE_COUNT_A will increment if any SBEs or MBEs, respectively, are detected during processor SRAM read accesses (via PORTA).

Detected SBEs will be corrected by the EDAC before reaching the processor but will not be corrected in SRAM.

Detected MBEs will not be corrected but will immediately generate a processor interrupt.

Enable (MBEA_INT_EN) processor interrupts for MBEs detected during processor SRAM read accesses.

The Processor MBE Interrupt Handler could initiate a warm reset, this is not an automatic process, but can be generated by a system call below.

```
// Reset the system since there are multiple errors

NVIC_SystemReset();
```

Q: Exactly what happens during a RSTN (hard) reset?

A: A hard reset, which can be triggered by the RSTN pin or by cycling the VDD and VDDA power supplies Power-On-Reset (POR), will reset the processor, the peripherals, the System Control Space (SCS), and the debug logic. The UT32M0R500 will then sample the BOOTCFG[0:1] pins to determine the boot operation (see the Functional Manual section 12 for what each boot mode involves).

Q: Exactly what happens during a soft reset (WD initiated)?

A: A soft reset will only reset the processor, the peripherals, and the System Control Space (SCS), but not the debug logic, meaning a debug session can continue through a soft reset. Additionally, users have the option to prevent the GPIO outputs and DAC outputs from being reset, meaning inputs/outputs stay in the same state/voltage level. The soft reset will still sample the BOOTCFG[0:1] pins to determine the boot operation, and follow that boot sequence.

Q: Is there a difference between a power cycle reset and a RSTN reset?

A: No.

Q: Is a warm reset with all the keep alive functionality disabled equivalent to a hard reset?

A: With none of the keep alive functionality enabled, a "warm" (soft) reset is the same as a hard reset except the user can continue a debug session. Additionally, the type of reset will still be logged in the System Controller Registers (section 8.3 Functional Manual) like RESET_INFO and RESET_COUNT.

Q: Is the internal POR function equivalent to the voltage monitoring function of an external voltage supervisor?

A: From the Datasheet:

"The UT32M0R500 contains power-on-reset (POR) circuitry. The POR monitors the VDD and VDDA power supplies. The POR also monitors the internally regulated core voltage (VDDC). The POR supports an external reset mode using the RSTN pin."

Q: Does the POR reset the MCU if the VDD and VDDA drop below thresholds?

A: Yes.

Q: UART TX and RX FIFO size?

A: TX and RX FIFO maximum is 8 bytes.

6.0 Revision History

Date	Revision #	Author	Change Description	Page #
9/14/2021	1.0.0	OW	Initial release.	
4/20/2026	1.1.0	JA	Multiple updates from pg 4-15	