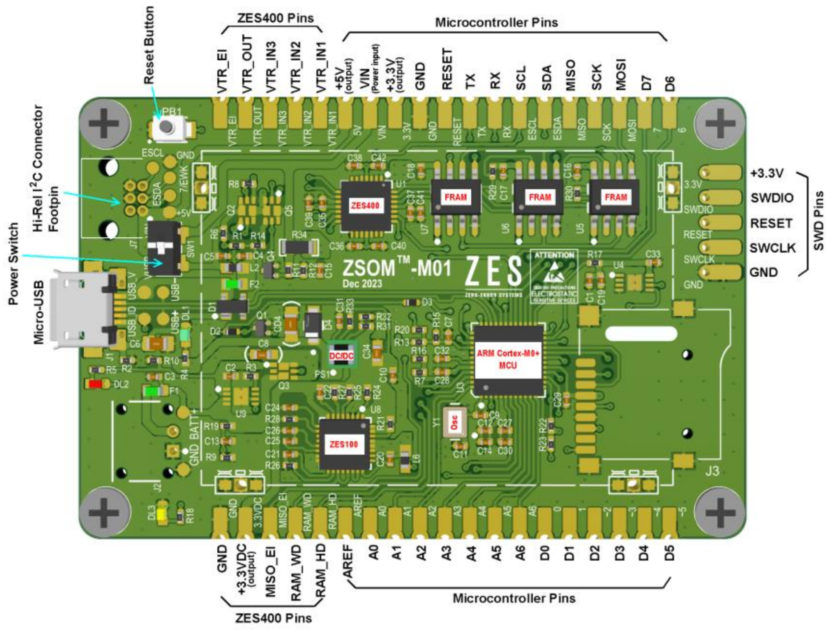
User Guide for ZSOM^{TM}-M01 Start Kit

Enabling Advanced Commercial-Off-The-Shelf (COTS) to 'Space-Grade'



Summary

ZSOM^{TM}-M01 is a radiation-tolerant system-on-module (SOM) embodying an ARM Cortex-M0+ microcontroller and ZES proprietary radiation-hardened technology for mitigating Single-Event-Latchup (SEL) and Single-Event-Upset (SEU) in irradiation environments. It can be configured within the Arduino-Compatible platform, leveraging available open-source code. ZSOM^{TM}-M01 enables a quick and reliable payload development platform for space-based applications.

Key Features

- Low power embedded microcontroller-based SOM with SEL protection and SEU protection
- SEL protection enabled by ZES100 Radiation-Hardened Latchup Detection and Protection (LDAP) chip
- External FRAM with SEU protection in Triple-Modular-Redundancy (TMR): enabled by ZES400 Radiation-Hardened Voters
- Proprietary error-detection-and-correction (EDAC) C-code for detecting/correcting multiple errors in SRAMs/FRAMs within the ARM Cortex-M0+ microcontroller
- High Reliability Manufacturing PCB (Class III)
- Arduino-Compatible Integrated Design Environment (IDE)
- **Proton test up to 200MeV @ fluence > 10^{11} particles/cm^2**
- Total Ionizing Dose (TID) (to be available soon)

Applications

- Low power embedded applications for payloads suitable for Low-Earth-Orbit (LEO)
- Watch-dog application
- Data communication applications

Table of Contents

Contents

1 Overview

ZSOM^{TM}-M01 is a radiation-tolerant system-on-module (SOM) embodying an ARM Cortex-M0+ microcontroller, a Triple-Modular-Redundancy (TMR) 256kB FRAM with a ZES400 radiation hardened voter, a ZES proprietary ZES100 Latchup Detection and Protection (LDAP) chip, and a DC/DC converter. ZSOM^{TM}-M01 can be configured within an Arduino-Compatible platform, leveraging the growing list of open-source code available for space-based applications. ZSOM^{TM}-M01 provides a quick and reliable payload development platform for developers' space-based applications.

1.1 Features

- SAMD21 ARM Cortex-M0+ Microcontroller (Automotive Grade), 256kB Flash, 32kB SRAM, 48MHz clock rate
- Single-Event-Latchup (SEL) protection: enabled by ZES100 Radiation-Hardened Latchup Detection and Protection (LDAP).
- External non-volatile memory with SEU protection: 32kB FRAMs configured in TMR enabled by ZES400 Radiation-Hardened Voters.
- Proprietary error-detection-and-correction (EDAC) C-code for detecting/correcting multiple errors (SEUs) within Flash/SRAM of the microcontroller.
- Common communication ports: Micro-USB, UART, SPI, I^{2}C
- Single power source: 5V (I/O voltage Interface @ 3.3V)
- Programing platform: C/C++ in Arduino-Compatible platform
- High Reliability Manufacturing (PCB Class III)
- Proton-tested up to 200MeV @ fluence > 10^{11} particles/cm^2
- TID test (to be available soon)

1.2 Block Diagram

Fig. 1 depicts the simplified diagram showing various key components and their interface signals/connections. The primary inputs/outputs are shown in blue; their pin definitions are delineated in Section 2.1 Table 1.

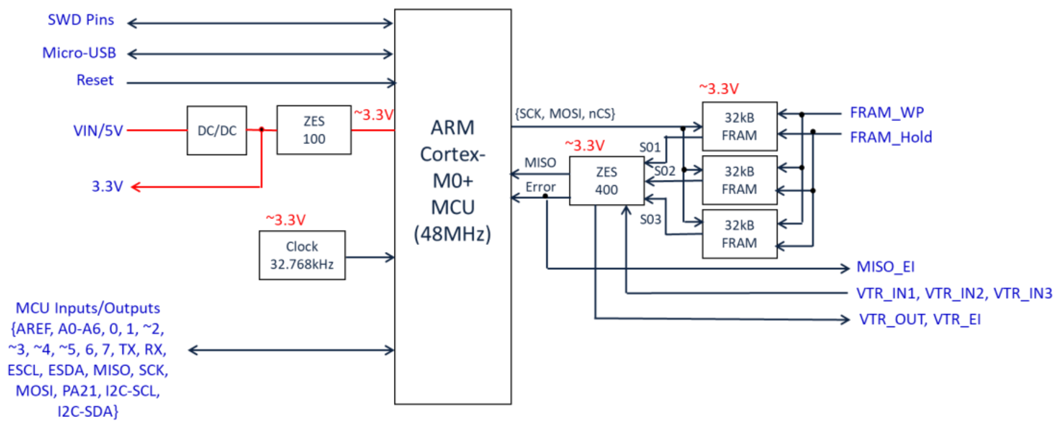


Fig. 1: The simplified block diagram of ZSOM^{TM}-M01

Please refer to the latest datasheet for the pin configuration and hardware setup.

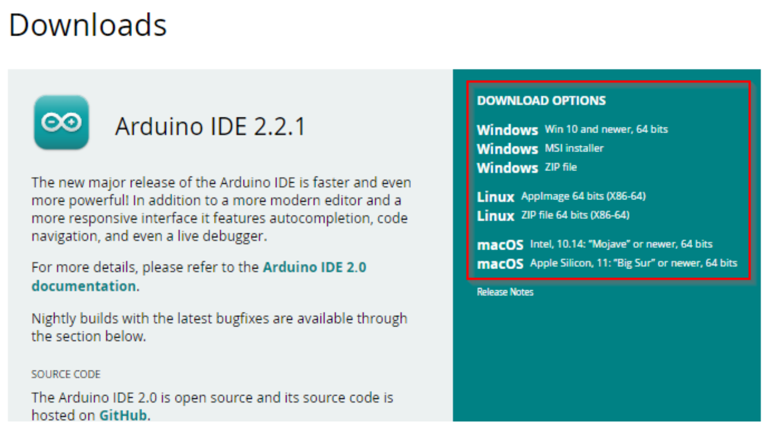
2 Basic User Guide to ZSOM-M01

The programming languages used are C++. For learning how to program/configure the SAMD21 ARM Cortex-M0+ microcontroller, please refer to the document associated with SAMD21G18A. To use this starter kit, please contact info@zero-errorsystems.com to obtain the following example sketches.

(a) ExampleAccessFRAM.zip

2.1 Integrated Design Environment (IDE) Software Installation and Setup

- Download Arduino IDE from Arduino website.
  [Software | Arduino]()



Downloads

Arduino IDE 2.2.1

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the **Arduino IDE 2.0 documentation.**

Nightly builds with the latest bugfixes are available through the section below.

SOURCE CODE

The Arduino IDE 2.0 is open source and its source code is hosted on **GitHub.**

DOWNLOAD OPTIONS

Windows Win 10 and newer, 64 bits
Windows MSI installer
Windows ZIP file

Linux AppImage 64 bits (X86-64)
Linux ZIP file 64 bits (X86-64)

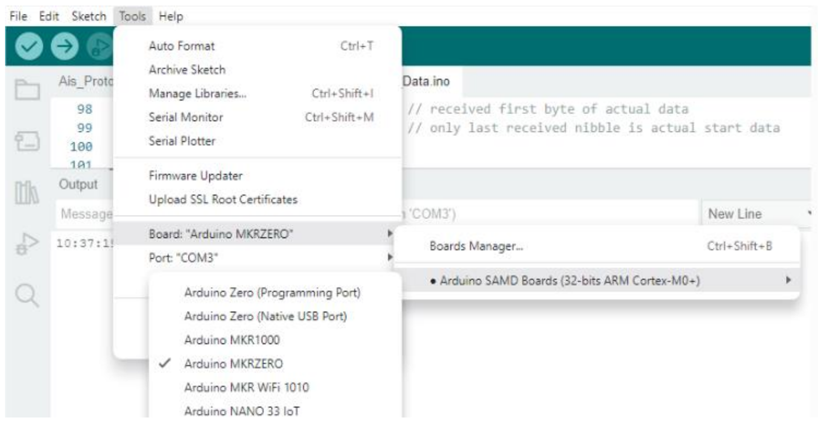
macOS Intel, 10.14: "Mojave" or newer, 64 bits
macOS Apple Silicon, 11: "Big Sur" or newer, 64 bits

Release Notes

- Install Arduino IDE and follow the installation instructions. (The installation might take several minutes depending on your system's performance.)

- **Setup IDE** and link it to a ZSOM^{TM}-M01 board.

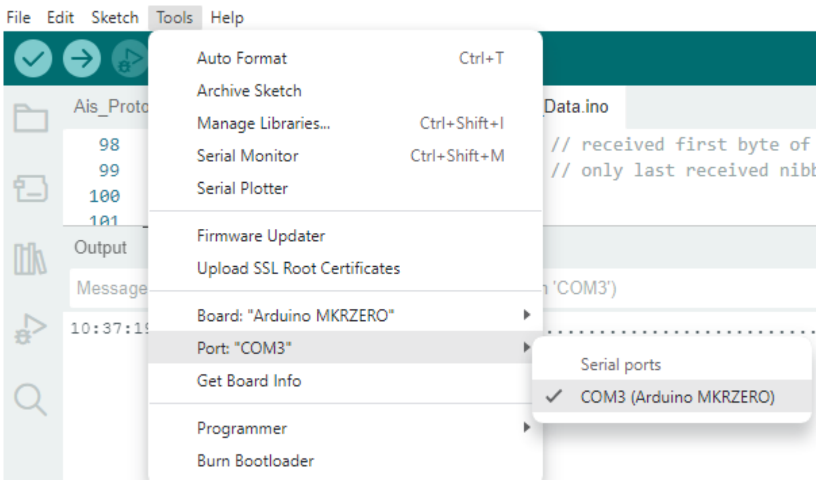
  1. Connect the ZSOM^{TM}-M01 to a PC.
  2. Open Arduino IDE.
  3. Go to "Tools" -> "Board"-> "Board Manager", look for "Arduino SAMD board" and install it.
  4. Go to "Tools" -> "Board"-> "Arduino SAMD Boards (32-bits ARM Cortex-M0+)", select "Arduino MKRZERO".


File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter
Firmware Updater
Upload SSL Root Certificates
Board: "Arduino MKRZERO"
Port: "COM3"
Boards Manager... Ctrl+Shift+B
• Arduino SAMD Boards (32-bits ARM Cortex-M0+)
Arduino Zero (Programming Port)
Arduino Zero (Native USB Port)
Arduino MKR1000
✓ Arduino MKRZERO
Arduino MKR WiFi 1010
Arduino NANO 33 IoT
Ais_Proto
Data.ino
Output
Message
'COM3')
New Line

```
// received first byte of actual data
// only last received nibble is actual start data
```

```
10:37:1
```

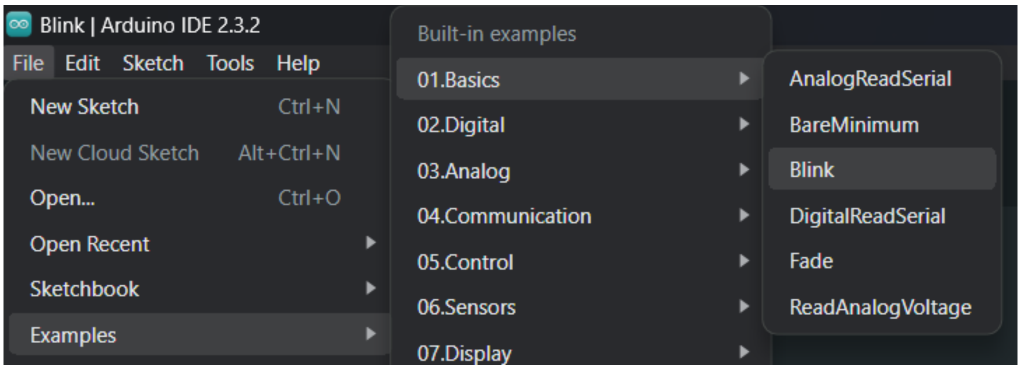
- Choose the appropriate COM port.
  Go to "Tools" -> "Port", select the appropriate COM port.



2.2 Example Sketch from Arduino IDE

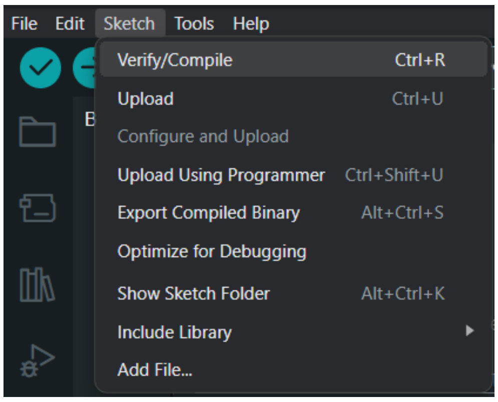
Before implementing sample sketch from ZES, users can explore one of the examples sketches available on Arduino IDE to familiarize themselves with the board programming. In the section, the example sketch, Blink, will be used.

- Go to “File” -> “Examples” -> “01.Basics” and click on “Blink” as shown below.

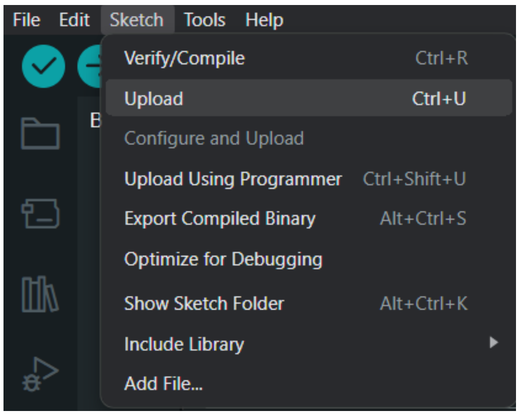


- Arduino IDE will be loaded with the program “Blink”.

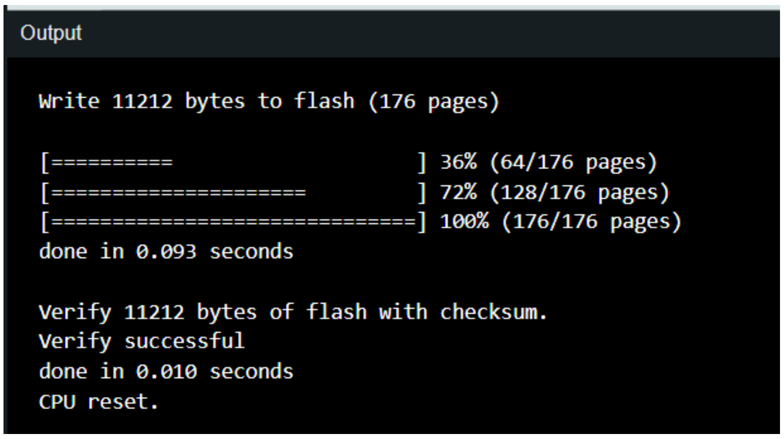
- Go to “Sketch” on Menu bar and click on “Verify/ Compile” or use hotkey “Ctrl+R” or click on the tick symbol ✓.



- Next, go to “Sketch” on Menu bar again and click on “Upload” or use the hotkey “Ctrl+U” or click on the right arrow key symbol →.

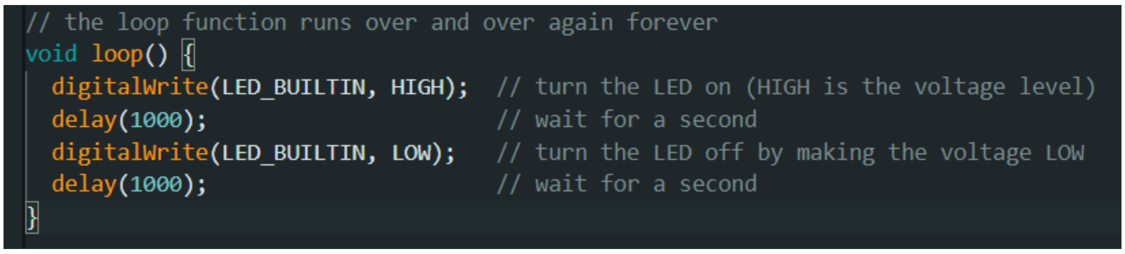


- Once the program has been uploaded successfully, the screen like below can be observed on the “Output” log of the Arduino IDE. The LED on board can also be observed blinking.



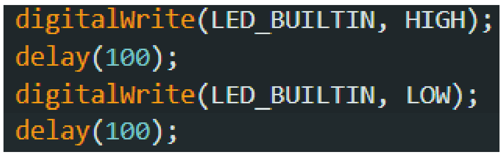
```
Write 11212 bytes to flash (176 pages)

[==========                   ] 36% (64/176 pages)
[=====================        ] 72% (128/176 pages)
[=============================] 100% (176/176 pages)
done in 0.093 seconds

Verify 11212 bytes of flash with checksum.
Verify successful
done in 0.010 seconds
CPU reset.
```

- By default, the LED blink time or on-off time is defined by the code below in the program and the time is 1 second or 1000 millisecond. The LED will be on for 1 sec and off for 1 sec. Then, the blinking will repeat infinitely since it is in a loop.



```
// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
  delay(1000);                      // wait for a second
  digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
  delay(1000);                      // wait for a second
}
```

- Next, change the LED blinking time by changing the delay time from 1000 to 100. The code should look like below to indicate the delay is 100ms. It can be observed that the LED is blinking faster than earlier.

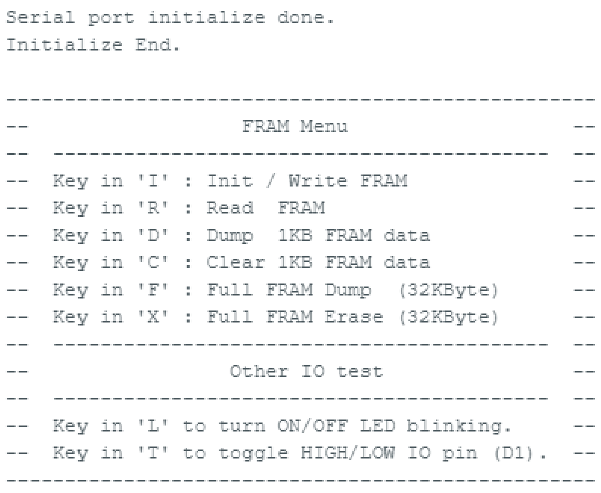


```
digitalWrite(LED_BUILTIN, HIGH);
delay(100);
digitalWrite(LED_BUILTIN, LOW);
delay(100);
```

- Users can also try out other examples available in the Arduino IDE. However, some of them will require extra components or equipment to test or observe.

2.3 Sample Sketch to Access FRAM within ZSOM™-M01

- Please refer to "**ExampleAccessFRAM.zip**".
- Run "ExampleAccessFRAM.ino" in Arduino IDE after unzipping the file.
- Make sure serial port is turn ON. [From Arduino IDE, press Ctrl + Shift + M]
- Upload sketch into ZSOM™-M01 board. [From Arduino IDE, press Ctrl + U]
- Follow instructions from "FRAM Menu" to test FRAM.



```
Serial port initialize done.
Initialize End.

--------------------------------------------------
--                  FRAM Menu                   --
--  ------------------------------------------  --
--  Key in 'I' : Init / Write FRAM              --
--  Key in 'R' : Read  FRAM                     --
--  Key in 'D' : Dump  1KB FRAM data            --
--  Key in 'C' : Clear 1KB FRAM data            --
--  Key in 'F' : Full FRAM Dump  (32KByte)      --
--  Key in 'X' : Full FRAM Erase (32KByte)      --
--  ------------------------------------------  --
--                 Other IO test                --
--  ------------------------------------------  --
--  Key in 'L' to turn ON/OFF LED blinking.     --
--  Key in 'T' to toggle HIGH/LOW IO pin (D1).  --
--------------------------------------------------
```

3 ZES Error Detection and Correction (ZEDAC) Protection

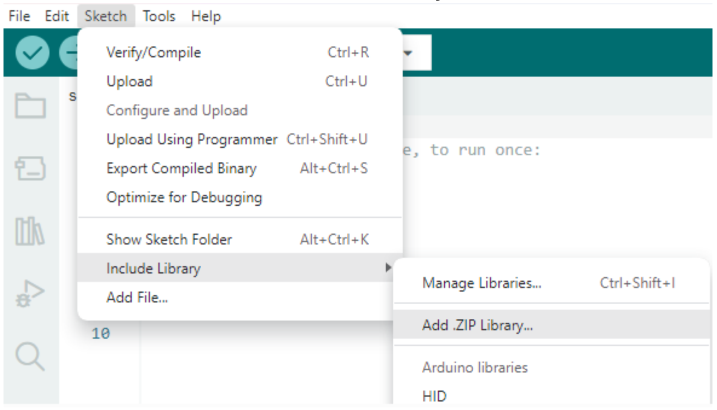
The programming languages used are C/C++. For learning how to program/configure the SAMD21 ARM Cortex-M0+ microcontroller, please refer to the document associated with SAMD21G18A. To use this starter kit, please contact info@zero-errorsystems.com to obtain the following library and example sketches.

(a) zesEDAC_1k_A_ver1.3.0.zip (ZEDAC library)
(b) ExampleZEDACUsage.zip

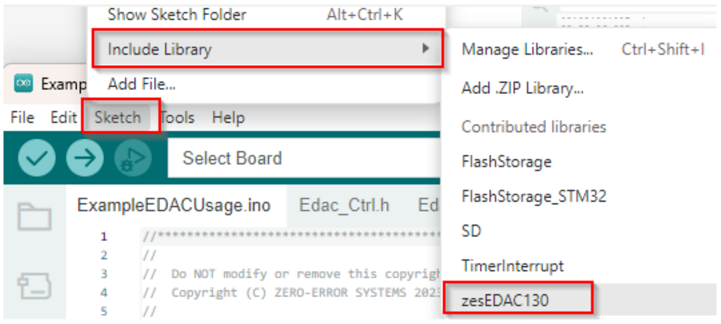
3.1 ZEDAC Library Download and Installation

a. "zesEDAC_1k_A_ver1.3.0.zip" is the ZEDAC library.

b. Add "zesEDACV130" library.
Go to "Sketch" -> "Include Library" -> "Add .ZIP Library…",
Go to folder where the ZEDAC library file is stored, select the library file.



c. Check if the library file is installed successfully.
Go to "Sketch" -> "Include Library", now "zesEDAC130" should be found under "Contributed libraries" section.

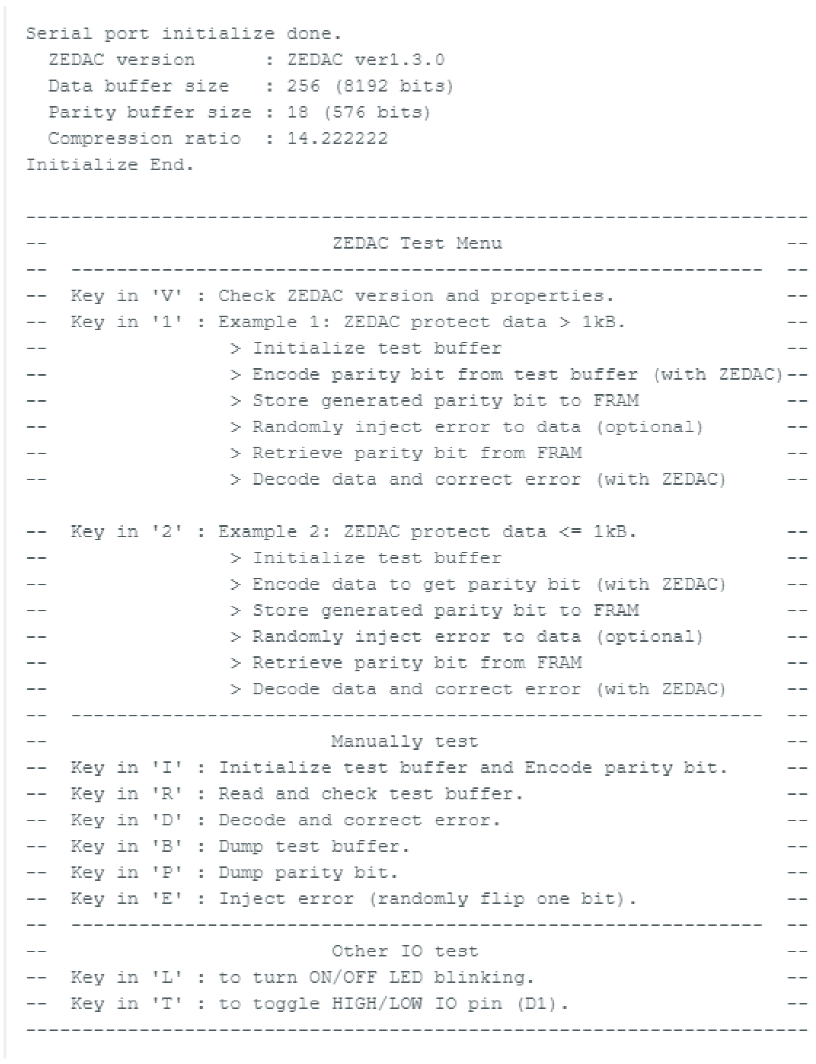


```
//*****************************************
//
//  Do NOT modify or remove this copyrig
//  Copyright (C) ZERO-ERROR SYSTEMS 202
//
```

d. If the library is installed successfully, "zesEDACV130.h" can be found at "...\libraries\zesEDAC130\src"
Remember to include "zesEDACV130.h" to sketch to use ZEDAC.

3.2 Sample Sketch to Use ZEDAC within ZSOM-M01

- Make sure ZSOM-M01 is connected to system.
- Choose the appropriate COM port.
  Go to "Tools" -> "Port", select the appropriate COM port.
- Please refer to “**ExampleZEDACUsage.zip**”.
- Run “ExampleZEDACUsage.ino” in Arduino IDE after unzipping the file.
- Make sure a serial port is turn ON. [From Arduino IDE, press Ctrl + Shift + M]
- Upload the sketch into ZSOM^{TM}-M01 board. [From Arduino IDE, press Ctrl + U]
- Follow instructions from “EDAC Test Menu” to test ZEDAC.



```
Serial port initialize done.
  ZEDAC version      : ZEDAC ver1.3.0
  Data buffer size   : 256 (8192 bits)
  Parity buffer size : 18 (576 bits)
  Compression ratio  : 14.222222
Initialize End.

---------------------------------------------------------------------
--                         ZEDAC Test Menu                         --
--  -------------------------------------------------------------  --
--  Key in 'V' : Check ZEDAC version and properties.               --
--  Key in '1' : Example 1: ZEDAC protect data > 1kB.              --
--                > Initialize test buffer                         --
--                > Encode parity bit from test buffer (with ZEDAC)--
--                > Store generated parity bit to FRAM             --
--                > Randomly inject error to data (optional)       --
--                > Retrieve parity bit from FRAM                  --
--                > Decode data and correct error (with ZEDAC)     --

--  Key in '2' : Example 2: ZEDAC protect data <= 1kB.             --
--                > Initialize test buffer                         --
--                > Encode data to get parity bit (with ZEDAC)     --
--                > Store generated parity bit to FRAM             --
--                > Randomly inject error to data (optional)       --
--                > Retrieve parity bit from FRAM                  --
--                > Decode data and correct error (with ZEDAC)     --
--  -------------------------------------------------------------  --
--                         Manually test                           --
--  Key in 'I' : Initialize test buffer and Encode parity bit.     --
--  Key in 'R' : Read and check test buffer.                       --
--  Key in 'D' : Decode and correct error.                         --
--  Key in 'B' : Dump test buffer.                                 --
--  Key in 'P' : Dump parity bit.                                  --
--  Key in 'E' : Inject error (randomly flip one bit).             --
--  -------------------------------------------------------------  --
--                         Other IO test                           --
--  Key in 'L' : to turn ON/OFF LED blinking.                      --
--  Key in 'T' : to toggle HIGH/LOW IO pin (D1).                   --
---------------------------------------------------------------------
```

4 Revision History

Version	Description	Date
V1.1	Update ZEDAC version (zesEDACV1.3.0).	21-Aug-2024
V1.0	First version	07-Mar-2024

For the latest version of this document, please contact us info@zero-errorsystems.com.

5 Legal

All products, product specificatios and data are subject to change without notice.