ACRONAME

Document PDOC00220
Version 1.0
Revised 2026-03-05

USBExt3c Letter of Volatility

Revisions

Revision	Date	Description
1.0	March 5, 2026	Initial release

Product: S150-USBEXT-3C (USBExt3c)
Manufacturer:
Acroname Inc.
2741 Mapleton Avenue
Boulder, CO, 80304

The tables below describe the type of memory within the product, the size and whether or not it is retained.

Volatile Memory

Type	Size	User Accessible / System Accessible	Battery Backup	Purpose	Method of Clearing
OC SRAM	256 kBytes	No / Yes	No	General purpose CPU instructions, USB and DMA	Power Cycle
DTC SRAM	192 kBytes	Yes / Yes	No	Application heap memory, temporary user storage	Power Cycle
ITC SRAM	64 kBytes	No / Yes	No	General firmware usage	Power Cycle

Non-Volatile Memory

Type	Size	User Accessible / System Accessible	Battery Backup	Purpose	Method of Clearing
Flash	32 MBits	No / Yes	No	Read-Only BrainStem Operating System	None Available to User[1]
EEPROM	128 KBits	Yes / Yes	No	BrainStem Reflex, User Configuration, and other user data	See clearing instructions

Media Storage

Type	Size	User Accessible / System Accessible	Battery Backup	Purpose	Method of Clearing
None[1]					

[1] The designation None Available to User indicates that the ability to clear this memory is not available to the user under normal operation. The utilities required to clear the memory are not distributed by Acroname to customers for normal use.

Clearing Instructions

The BrainStem internal flash store has 12 user accessible slots which are 4094 bytes long. The BrainStem API provides access to these data stores, and can be used to clear them. The following is example C++ code which will clear all internal flash storage slots. Performing this action will clear all non-volatile memory including calibration parameters.

```
#include <iostream>
#include "BrainStem2/BrainStem-all.h"
int main(int argc, const char * argv[]) {
    aErr err = aErrNone;
    aUSBExt3c stem;
    //Find and connect to device.
    err = stem.discoverAndConnect(USB);
    if (err != aErrNone) {
        printf("Error connecting to BrainStem (%d).\n", err);
        return 0;
    }
    // go through all internal flash slots
    for(int slotNum=0; slotNum<aUSBEXT3C_NUM_INTERNAL_SLOTS; slotNum++){
        size_t slotCap=0;
        size_t slotSz=0;
        // get the slot's capacity
        err = stem.store[aUSBEXT3C_STORE_EEPROM_INDEX].getSlotCapacity(slotNum, &slotCap);
        if (err != aErrNone){
            printf("Slot %d capacity returned error %d\n", slotNum, err);
        }
        else {
            // write 0's to the whole slot size; report the size
            printf("Slot %d capacity is %ld bytes; zeroing data\n", slotNum, slotCap);
            uint8_t* slotData= (uint8_t*)calloc(slotCap, sizeof(uint8_t));
            err = stem.store[aUSBEXT3C_STORE_EEPROM_INDEX].loadSlot(slotNum, slotData, slotCap);
            printf("\tDone with err %d\n", err);
            // verify slot size
            err = stem.store[aUSBEXT3C_STORE_EEPROM_INDEX].getSlotSize(slotNum, &slotSz);
            printf("\tSlot size %ld with err %d\n", slotSz, err);
            free(slotData);
            // write a zero length array to the slot
            printf("Writing 0 length to slot %d\n", slotNum);
            err = stem.store[aUSBEXT3C_STORE_EEPROM_INDEX].loadSlot(slotNum, slotData, 0);
            printf("\tDone with err %d\n", err);
            // verify slot size
            err = stem.store[aUSBEXT3C_STORE_EEPROM_INDEX].getSlotSize(slotNum, &slotSz);
            printf("\tSlot size %ld with err %d\n", slotSz, err);
        }
    }
    // Disconnect from the module
    stem.disconnect();
```

```
    return 0;
}
```

Terms and Definitions

User Accessible: Allows the user to directly write or modify the contents of the memory during normal instrument operation.

System Accessible: Does not allow the user to access or modify the memory during normal instrument operation. However, system accessible memory may be accessed or modified by background processes. This can be something that is not deliberate by the user and can be a background driver implementation, such as storing application information in RAM to increase speed of use.

Cycle Power: The process of completely removing power from the device and its components.

Volatile Memory: Requires power to maintain the stored information. When power is removed from this memory, its contents are lost.

Non-Volatile: Retains its contents when power is removed. This type of memory typically contains system and user application code.