Product:
S79-USBHUB-3P (USBHub3+)

Manufacturer:
Acroname Inc.
4822 Sterling Drive
Boulder, CO, 80301-2350

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

## Volatile Memory

<table>
<thead>
<tr>
<th>Type</th>
<th>Size</th>
<th>User Accessible / System Accessible</th>
<th>Battery Backup</th>
<th>Purpose</th>
<th>Method of Clearing</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRAM</td>
<td>32 kBytes</td>
<td>No / Yes</td>
<td>No</td>
<td>General purpose CPU instructions, USB and DMA</td>
<td>Power Cycle</td>
</tr>
<tr>
<td>AHB SRAM/1</td>
<td>28 kBytes</td>
<td>No / Yes</td>
<td>No</td>
<td>IO bridge</td>
<td>Power Cycle</td>
</tr>
<tr>
<td>SRAM</td>
<td>4 kBytes</td>
<td>Yes / Yes</td>
<td>No</td>
<td>BrainStem Reflex Virtual machine slot</td>
<td>Power Cycle OR see clearing instructions</td>
</tr>
</tbody>
</table>

## Non-Volatile Memory

<table>
<thead>
<tr>
<th>Type</th>
<th>Size</th>
<th>User Accessible / System Accessible</th>
<th>Battery Backup</th>
<th>Purpose</th>
<th>Method of Clearing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flash</td>
<td>464 kBytes</td>
<td>No / Yes</td>
<td>No</td>
<td>BrainStem Operating System</td>
<td>None Available to User¹</td>
</tr>
<tr>
<td>Flash</td>
<td>4 kBytes (x12)</td>
<td>Yes / Yes</td>
<td>No</td>
<td>BrainStem Reflex Virtual machines and other user data</td>
<td>See clearing instructions</td>
</tr>
</tbody>
</table>

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 National Instruments 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 follow is example C++ code which will clear all internal flash storage slots.

```cpp
#include "BrainStem2/BrainStem-all.h"

int main(int argc, const char * argv[]) {
  aErr err = aErrNone;
  aUSBHub3p stem;
  //Find and connect to device.
  err = stem.discoverAndConnect(USB);

  if (err != aErrNone) {
    printf("Error connecting to BrainStem (%d).
    return 0;
  }

  // go through all internal flash slots
  for(int slotNum=0; slotNum<aUSBHUB3P_NUM_INTERNAL_SLOTS; slotNum++){
    size_t slotCap=0;
    size_t slotSz=0;
    // get the slot's capacity
    err = stem.store[storeInternalStore].getSlotCapacity(slotNum, &slotCap);
    if (err != aErrNone){
      printf("Slot %d capacity returned error %d
", 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[storeInternalStore].loadSlot(slotNum, slotData, slotCap);
      printf("\tDone with err %d
", err);
      // verify slot size
      err = stem.store[storeInternalStore].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[storeInternalStore].loadSlot(slotNum, slotData, 0);
      printf("\tDone with err %d\n", err);
      // verify slot size
      err = stem.store[storeInternalStore].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.