SPACEPOINT™ CR
CONTROLLER MOTION–TRACKING SYSTEM

# SpacePoint CR

## VR HMD and Controller Calibration Procedure and Warm Start Guide

PNI
SENSOR CORPORATION

# VR HMD and Controller Calibration Procedure and Warm Start Guide

## Contents

## Introduction

In order to determine when a good magnetic calibration has been achieved, calibration status (CalStatus and CalScore) and gyro bias update (gBiasUpdate) must be retrieved. This document explains how to perform a magnetic calibration along with obtaining the information to indicate when the calibration has been completed.

As an option, accelerometer calibration is supported, too. This document also describes the procedure to perform a one-point accelerometer calibration as a reference.

## Magnetic Calibration Procedure

*Note: This procedure requires a **magnetically clean environment.** Perform calibration at least two meters (~6½ feet) away from any ferrous materials that could cause magnetic distortion, such as steel tables or chairs, file cabinets, computer systems, etc.*

1) Before powering up the device, make sure that it is motionless and in a magnetically clean environment.
2) Read register 0x44 bit 1, gBiasUpdate bit, to verify that the gyro bias has been learned. Note that this flag should trigger approximately every two (2) seconds while the device is still.
3) Begin the magnetic calibration procedure. Rotate the device slowly in a figure-8 pattern so that each axis gets as much coverage as possible. Keep the center of the device as a pivot point. Rotate around this pivot point to cover the surface of an imaginary sphere as evenly as possible. An example rotation is shown below in Figure 1.

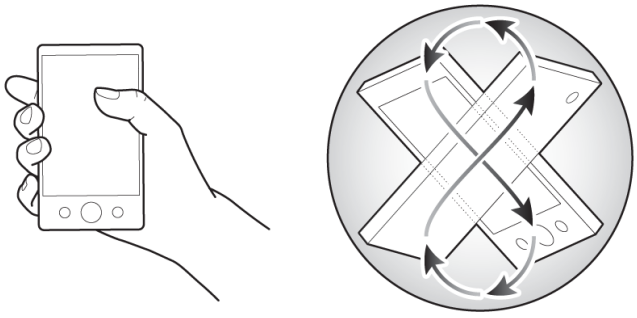

*Figure 1: Magnetic Calibration Procedure*

4) Check register 0x43, CalStatus, to verify that the calibration procedure has been completed. This register should read a value of 0x03. For better calibration result, users should also monitor calibration score registers 0x4D and 0x4E. The result is acceptable when the calibration score is about 5. The result is best when the score is about 3 or less. Refer to section “Register Map” to get the register details and instructions to parse the data on host.

5) The calibration procedure may take a few seconds to complete.

6) Once the calibration has been completed, register 0x43 reads a value of 0x03 with acceptable calibration score from registers 0x4D and 0x4E. At this point the calibration results and warm start parameters should be saved. Please refer to section “WarmStart Read Sample Code” for sample code on reading the warm start parameters.

7) At next power-on the host should write the warm start parameters back to SENtral before enabling any of the sensors. Please refer to section “WarmStart Write Sample Code” for sample code on loading the warm start parameters back into SENtral.

## Accelerometer Calibration Procedure (Optional)

1) Place the device on a flat surface so that the Z-Axis is facing down, as shown in Figure 2, assuming an NED (North East Down) coordinate system. The device must remain completely still for this entire procedure.

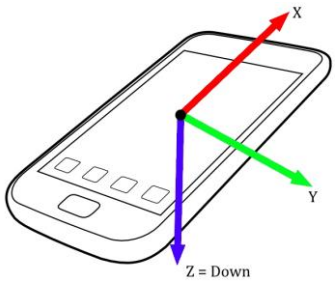


*Figure 2: Device Z-Axis facing Down*

2) Power on the device.
3) Verify that the data is set to output as scaled sensor data. AlgorithmControl register 0x54 should be 0x00.
4) Gather data on all axes of the accelerometer sensor. Accelerometer data registers are listed below; note that data is store in Little Endian convention.
    - AX = Read registers 0x1A-0x1B
    - AY = Read registers 0x1C-0x1D
    - AZ = Read registers 0x1E-0x1F
5) Calculate the offset values needed. Note: The offset values should be in units of "g." To convert the offset values to units of "g" multiply the offset values by the scale factor "0.000488." With the device oriented with the Z-Axis facing up, the values for each axis **after** the offsets have been applied should be:
    - X = 0
    - Y = 0
    - Z = 2048

    So to calculate the offset values the equation would be:

    Offset Value = Initial Value – Expected Value

**Example:**

*Table 1: Example of Accelerometer Offset Calculations*

| Axis | Initial Value (Hex) | Initial Value (Decimal) | Calculated Offset Value (Decimal) | Calculated Offset Value (Float) | Calculated Offset Value (Hex) |
|---|---|---|---|---|---|
| AX | D1 FF | -47 | -47 | -0.022936 | BC BB E4 47 |
| AY | F1 FF | -15 | -15 | -0.00732 | BB EF DC 9C |
| AZ | 2E 08 | 2094 | 46 | 0.022448 | 3C B7 E4 DE |

6) Accel offsets are saved in warmstart list as parameter ID from 27 to 29. Write and read them the same way as other warmstart parameters. Perform the ParamIO operations to load the calculated offset values to the correct parameters; see Table 2 for the parameter information. Please refer to "Appendix III – Parameter Transfer" of the SENtral Technical Datasheet for detailed information on the parameter transfer process. Some sample code on the parameter transfer process has also been provided at the end of this guide.

*Table 2: Accelerometer Offset Parameters*

| Parameter Number | Parameter Name |
|---|---|
| 27 | AX Offset |
| 28 | AY Offset |
| 29 | AZ Offset |

7) Verify that the offsets have been correctly applied by reading the accelerometer sensor data from Step 4 above. The values should now be close to or identical to those mentioned in Step 5 above.

## Register Map

*Register Map for CalStatus, CalScore and gBiasUpdate*

| Register Address (Hex) | R/W Access | Comment |
|---|---|---|
| 43 | RO | Bits [0] – [1] CalStatus. 3 = Calibration complete<br>[2] – MagTransient. 1 = Magnetic transient present |
| cal_status = read(0x43) & 0x03; MagTransient = (read(0x43)&0x0C) >> 2; | | |
| 44 | RO | [0] magCalUpdate. Flag will trigger when magnetic calibration has updated.<br>[1] gBiasUpdate. Flag will trigger when gyro bias has been updated. |
| 4D | RO | Calibration Score LSB |
| 4E | RO | Calibration Score MSB |
| calScore = ({[calScore_LSB \| (calScore_MSB << 8)] – 32768 } /32768.0 ) *100.0 | | |

## WarmStart Read Sample Code

```
/******************* (C) PNI 2013  *******************
* File Name          : ParamIOKnobs.c
* Author             : BZ
*****************************************************/
//The following is for testing only. Please read warmstart first to fill up
//this array with Sentral's values.
float warmstart[35] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
                       16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28,
                       29, 30, 31, 32, 33, 34, 35};

SInt32 ReadWarmStart(UInt8 command)
{

  SInt32 ret = SUCCESS;

  UInt8 temp =0x01;
  union {
    UInt8 bytes[4];
    float f;
    UInt32 number;
  } val;

  tClockTime start_time;

  for (int i = WARMSTART_BEGIN; i <= WARMSTART_END; i++)
  {
    start_time = Clock_Time();

    SentralWrite(PARAMREQ, (i&0x7F));
    Clock_Wait(10);
    SentralRead(ALGORITHM_CONTROL,&temp);
    Clock_Wait(10);
    temp |= 0x80;
    SentralWrite(ALGORITHM_CONTROL, temp);
    Clock_Wait(10);

    SentralRead(PARAMREQ,&temp);
    while ( temp != i)
    {
      Clock_Wait(10);
      SentralRead(PARAMREQ,&temp);
      if ((Clock_Time() - start_time) > 1000) //1ms resolution, 1000 = 1s
        return ERROR;
    }
    //float
    SentralRead(SAVEPARAMBYTE0,&temp);
    val.bytes[0] = temp;
    SentralRead(SAVEPARAMBYTE1,&temp);
    val.bytes[1] = temp;
    SentralRead(SAVEPARAMBYTE2,&temp);
    val.bytes[2] = temp;
    SentralRead(SAVEPARAMBYTE3,&temp);
    val.bytes[3] = temp;
    warmstart[i-1] = val.f;
    printf("Read ws%2u = %f, 0x%8X\n\r", i, val.f, val.number);
  }//for loop

  return ret;
}
```

## WarmStart Write Sample Code

```
/****************** (C) PNI 2013  ******************
* File Name          : ParamIOKnobs.c
* Author             : BZ
*****************************************************/
SInt32 WriteWarmStart(UInt8 command)
{
  SInt32 ret = SUCCESS;
  UInt8 array_id;

  UInt8 temp =0x01;
  union {
    UInt8 bytes[4];
    float f;
    UInt32 number;
  } val;


  for (int knobId = WARMSTART_BEGIN; knobId <= WARMSTART_END; knobId++)
  {
    tClockTime start_time;
    start_time = Clock_Time();

    array_id = knobId - WARMSTART_BEGIN;
    val.f = warmstart[array_id];
    //Can be optimized to I2C multiple write
    SentralWrite(LOADPARAMBYTE0, val.bytes[0]);
    SentralWrite(LOADPARAMBYTE1, val.bytes[1]);
    SentralWrite(LOADPARAMBYTE2, val.bytes[2]);
    SentralWrite(LOADPARAMBYTE3, val.bytes[3]);

    SentralWrite(PARAMREQ, (knobId|0x80)); //KnobID | 0x80

    Clock_Wait(10);
    SentralRead(ALGORITHM_CONTROL,&temp);
    Clock_Wait(10);
    temp |= 0x80;
    SentralWrite(ALGORITHM_CONTROL, temp);
    Clock_Wait(10);

    SentralRead(PARAMACK,&temp);
    while ( temp != (knobId|0x80))
    {
      Clock_Wait(10);
      SentralRead(PARAMACK,&temp);
      if ((Clock_Time() - start_time) > 1000) //1ms resolution, 1000 = 1s
        return ERROR;
    }
    printf("Write ws%2u = %f, 0x%8X\n\r", knobId, val.f, val.number);
  }//for loop

  return ret;
}
```

## WarmStart Header File Sample Code

```
/******************* (C) PNI 2013  *******************
* File Name          : ParamIOKnobs.h
* Author             : BZ
*****************************************************/
#include "common.h"

//Host Write to Sentral Regs
#define LOADPARAMBYTE0 0x60
#define LOADPARAMBYTE1 0x61
#define LOADPARAMBYTE2 0x62
#define LOADPARAMBYTE3 0x63
#define PARAMREQ       0x64

//Host Read from Sentral Regs
#define PARAMACK       0x3A
#define SAVEPARAMBYTE0 0x3B
#define SAVEPARAMBYTE1 0x3C
#define SAVEPARAMBYTE2 0x3D
#define SAVEPARAMBYTE3 0x3E

#define WARMSTART_BEGIN 0x01
#define WARMSTART_END   0x23 //35

SInt32 ReadWarmStart(UInt8 command);
SInt32 WriteWarmStart(UInt8 command);
```

## WarmStart Parameter Description

| Coefficient Number | Coefficient Name | Value Saved From | Value Restored Into | Note |
|---|---|---|---|---|
| 1 | xstates[1] | k1.x[7] | kSet.x_init[7] | |
| 2 | xstates[2] | k1.x[9] | kSet.x_init[9] | |
| 3 | gparams[1] | k1.gbias_mode | k1.gbias_mode | (1) |
| 4 | gparams[2] | k1.GyroThresh[1] | k1.GyroThresh[1] | |
| 5 | gparams[3] | k1.GyroThresh[2] | k1.GyroThresh[2] | (2) |
| 6 | gparams[4] | k1.GyroThresh[3] | k1.GyroThresh[3] | (2) |
| 7 | gparams[5] | k1.GyroThresh[4] | k1.GyroThresh[4] | (2) |
| 8 | gparams[6] | k1.GyroThresh[5] | k1.GyroThresh[5] | |
| 9 | nparams[1] | k1.noiseLvls[1] | k1.noiseLvls[1] | (2) |
| 10 | nparams[2] | k1.noiseLvls[2] | k1.noiseLvls[2] | (2) |
| 11 | nparams[3] | k1.noiseLvls[3] | k1.noiseLvls[3] | (2) |
| 12 | nparams[4] | k1.noiseLvls[4] | k1.noiseLvls[4] | (2) |
| 13 | nparams[5] | k1.noiseLvls[5] | k1.noiseLvls[5] | (2) |
| 14 | nparams[6] | k1.noiseLvls[6] | k1.noiseLvls[6] | (2) |
| 15 | nparams[7] | k1.noiseLvls[7] | k1.noiseLvls[7] | (2) |
| 16 | nparams[8] | k1.noiseLvls[8] | k1.noiseLvls[8] | (2) |
| 17 | nparams[9] | k1.noiseLvls[9] | k1.noiseLvls[9] | (2) |
| 18 | mcoeffs[1] | mcalOut.mcalSIHI(1,1) | mcalOut.mcalSIHI(1,1) | (3) |
| 19 | mcoeffs[2] | mcalOut.mcalSIHI(2,2) | mcalOut.mcalSIHI(2,2) | (3) |
| 20 | mcoeffs[3] | mcalOut.mcalSIHI(3,3) | mcalOut.mcalSIHI(3,3) | (3) |
| 21 | mcoeffs[4] | mcalOut.mcalSIHI(1,4) | mcalOut.mcalSIHI(1,4) | (3) |
| 21 | mcoeffs[4] | mcalOut.mcalSIHI(1,4) | mcalOut.mcalSIHI(1,4) | (3) |
| 22 | mcoeffs[5] | mcalOut.mcalSIHI(1,5) | mcalOut.mcalSIHI(1,5) | (3) |
| 23 | mcoeffs[6] | mcalOut.mcalSIHI(1,6) | mcalOut.mcalSIHI(1,6) | (3) |
| 24 | acoeffs[1] | 1 | Not used | |
| 25 | acoeffs[2] | 1 | Not used | |
| 26 | acoeffs[3] | 1 | Not used | |
| 27 | acoeffs[4] | 0 | Not used | |
| 28 | acoeffs[5] | 0 | Not used | |
| 29 | acoeffs[6] | 0 | Not used | |
| 30 | gcoeffs[1] | 1 | Not used | |
| 31 | gcoeffs[2] | 1 | Not used | |
| 32 | gcoeffs[3] | 1 | Not used | |
| 33 | gcoeffs[4] | k1.x[4] | k1.x[4], kSet.x_init[4], k1.gbias[1], ktemp.x[4] | |
| 34 | gcoeffs[5] | k1.x[5] | k1.x[5], kSet.x_init[5], k1.gbias[2], ktemp.x[5] | |
| 35 | gcoeffs[6] | k1.x[6] | k1.x[6], kSet.x_init[6], k1.gbias[3], ktemp.x[6] | |

**Notes:**

(1) Uint8 saved as Float

(2) Saved value is multiplied by `noiseMultFactor` before Restore

(3) Remaining `mcalOut.mcalSIHI` coefficients are set to 0.

Revision Control Block

| Revision | Description of Change | Effective Date | Approval |
|---|---|---|---|
| 01 | Initial Release | October 10, 2016 | B. Thomlinson |