x9402 - Test Force button

Version 2

Test Force button

[0] **getSensorCount**() → sensorCount
[1] **getSensorCapabilities**(sensorIdx) → technology
[2] **readMeasurement**(sensorIdx) → sensorIdx, measurement, rest_position, preload_adjustment
[3] **writeCalibrationData**(sensorIdx, thresholdNominal, thresholdMin, thresholdMax)
[4] **readCalibrationData**(sensorIdx) → sensorIdx, thresholdNominal, thresholdMin, thresholdMax
[5] **monitorTest**(sensorIdx, count)
[Event0] **monitorReport**() → sensorIdx, measurement, rest_position

Overview

The purpose of this feature is to measure the sensitivity of the force sensing button, define the activation thresholds (Nominal, high and low) then store them in the device.

The button force can be measured using different technologies : - Faraday, Logitech internal low cost solution based on inductive sensing. See here more information about the Faraday solution: Canova Inductive Button - Qorvo df8100, a high end solution.

In this document, the sensor refers to the force button. Sensors are indexed from right to left (and then furthest to closest) as seen by the user.

Functions and Events

[0] getSensorCount() → sensorCount

Returns the number of force sensors on the device.

Parameters

none

Returns

sensorCount

[1 byte] Number of force sensing buttons.

Table 1. getSensorCount() response packet format

byte \ bit	7	6	5	4	3	2	1	0
0	sensorCount							
1..15	*reserved*							

[1] getSensorCapabilities(sensorIdx) → technology

Returs the capabilities of the selected sensor.

Parameters

sensorIdx

[1 byte] The index of the sensor to get the capabilities.

Table 2. getSensorCapabilities() request packet format

byte \ bit	7	6	5	4	3	2	1	0
0	sensorIdx							
1..15	*reserved*							

Returns

technology

[1 byte] The technology used for the sensor.

```
0x00: Faraday
0x01: Qorvo df8100
Other values: RFU
```

Table 3. getSensorCapabilities() response packet format

byte \ bit	7	6	5	4	3	2	1	0
0	technology							
1..15	*reserved*							

Errors

- INVALID_ARGUMENT (2): if invalid sensor index

[2] readMeasurement(sensorIdx) → sensorIdx, measurement, rest_position, preload_adjustment

Returns the measurement from the device.

- If Qorvo df8100 technology is used, read the value through I2C at address 0x03-0x04.
- If the technology is Faraday, read the value from the device directly.

Parameters

sensorIdx

[1 byte] The index of the sensor to read the measurement from the device.

Table 4. readMeasurement() request packet format

byte \ bit	7	6	5	4	3	2	1	0
0	sensorIdx							
1..15	*reserved*							

Returns

sensorIdx

[1 byte] The index of the sensor.

measurement

[2 bytes] The ADC count measurement from the device.

rest_position

[2 bytes] The rest position of the sensor.

preload_adjustment

[1 byte] The preload adjustment of the sensor.

Table 5. readMeasurement() response packet format

byte \ bit	7	6	5	4	3	2	1	0
0	sensorIdx							
1	measurement (LSB)							
2	measurement (MSB)							

byte \ bit	7	6	5	4	3	2	1	0
3	rest_position (LSB)							
4	rest_position (MSB)							
5	preload_adjustment							
6..15	*reserved*							

Errors

- INVALID_ARGUMENT (2): if invalid sensor index
- HW_ERROR (4): when MCU is unable to read the sensor value, for example, if the sensor is not connected or not responding

[3] writeCalibrationData(sensorIdx, thresholdNominal, thresholdMin, thresholdMax)

Once the calibration procedure is finished and the thresholds are defined (see Calibration procedures below), set the final calibration values in the NVM.

Parameters

sensorIdx

[1 byte] The index of the sensor.

thresholdNominal

[2 bytes] The nominal threshold.

thresholdMin

[2 bytes] The Low threshold.

thresholdMax

[2 bytes] The High threshold.

NOTE

Threshold unit in ADC count.

Table 6. writeCalibrationData() request packet format

byte \ bit	7	6	5	4	3	2	1	0
0	sensorIdx							
1	thresholdNominal (LSB)							

byte \ bit	7	6	5	4	3	2	1	0
2	thresholdNominal (MSB)							
3	thresholdMin (LSB)							
4	thresholdMin (MSB)							
5	thresholdMax (LSB)							
6	thresholdMax (MSB)							
7..15	*reserved*							

Returns

none

Errors

- INVALID_ARGUMENT (2): if invalid sensor index
- HW_ERROR (4): when MCU is unable to write the sensor calibration data, for example, if the sensor is not connected or not responding

[4] readCalibrationData(sensorIdx) → sensorIdx, thresholdNominal, thresholdMin, thresholdMax

Read the calibration data from the NVM.

Parameters

sensorIdx

[1 byte] The index of the sensor to read the calibration data from.

Table 7. readCalibrationData() request packet format

byte \ bit	7	6	5	4	3	2	1	0
0	sensorIdx							
1..15	*reserved*							

Returns

sensorIdx

[1 byte] The index of the sensor.

thresholdNominal

[2 bytes] The nominal threshold.

thresholdMin

[2 bytes] The Low threshold.

thresholdMax

[2 bytes] The High threshold.

Table 8. readCalibrationData() response packet format

byte \ bit	7	6	5	4	3	2	1	0
0	sensorIdx							
1	thresholdNominal (LSB)							
2	thresholdNominal (MSB)							
3	thresholdMin (LSB)							
4	thresholdMin (MSB)							
5	thresholdMax (LSB)							
6	thresholdMax (MSB)							
7..15	*reserved*							

Errors

- INVALID_ARGUMENT (2): if invalid sensor index
- HW_ERROR (4): when MCU is unable to read the sensor calibration data, for example, if the sensor is not connected or not responding

[5] monitorTest(sensorIdx, count)

Start monitor Test, returning real time measurements of selected sensor.

The sensor must be in the normal state before starting the monitor test. Once the monitor test is launched, a total of count events monitorReport will be sent.

To abort the test before completion, call monitorTest again with count = 0.

Parameters

sensorIdx

[1 byte] The index of the sensor to monitor.

count

[2 bytes] The number of monitorReport to be sent.

```
0x0000: Abort the monitor test.
0xFFFF: Infinite monitor test.
```

Table 9. monitorTest() request packet format

byte \ bit	7	6	5	4	3	2	1	0
0	sensorIdx							
1	count (LSB)							
2	count (MSB)							
3..15	*reserved*							

Returns

none

Errors

- INVALID_ARGUMENT (2): if invalid sensor index or count
- HW_ERROR (4): when MCU is unable to start the monitor test, for example, if the sensor is not connected or not responding

[Event0] monitorReport() → sensorIdx, measurement, rest_position

Event sent by the device to report the real time measurements of the selected sensor.

NOTE

To compensate for the slow monitorReport notification frequency over BLEPP compared to the ADC measurement data rate, each notification packet includes up to 6 samples.

Parameters

none

Returns

sensorIdx

[1 byte] The index of the sensor.

packetIdx

[2 byte] The index of notification packet.

measurement[5]

[2 bytes] The ADC count measurement from the device.

rest_position

[2 bytes] The rest position of the sensor. The value is related to the measurement[0].

Table 10. monitorTest() response packet format

byte \ bit	7	6	5	4	3	2	1	0
0	sensorIdx							
1	packetIdx (LSB)							
2	packetIdx (MSB)							
3	measurement[0] (LSB)							
4	measurement[0] (MSB)							
5	measurement[1] (LSB)							
6	measurement[1] (MSB)							
7	measurement[2] (LSB)							
8	measurement[2] (MSB)							
9	measurement[3] (LSB)							
10	measurement[3] (MSB)							
11	measurement[4] (LSB)							
12	measurement[4] (MSB)							
13	rest_position (LSB)							
14	rest_position (MSB)							
15	*reserved*							

Calibration procedure

The following are the **steps** of the calibration :

1. An unloaded value is read from the mouse, measurement M_0, by calling readMeasurement function.
2. A force F_1 is applied to the island at a predetermined point (care should be taken to be as repeatable as possible on this point)
3. A loaded value is read from the mouse, M_1, by calling readMeasurement function.
4. Steps 2-3 are repeated for each Force required for the test (F_n).
5. SensitivitCheck to see if linearity is within expected range (only possible when measuring more than 1 point)
6. y is calculated (for only 1 force, F1 the calculation is : $S = (M_1 - M_0)/F_1$)
7. Check to see if Sensitivity is within expected bounds.
8. Check to see if Hysteresis is within expected range (only possible when measuring the same force, once from no load, and the other from max load)
9. Nominal TH is calculated, $TH_n = F_n \times S$, where F_n is nominal press force.
10. High TH is calculated, $TH_H = TH_n \times XX$ (YY is product specif and need TBD with UX team)
11. Low TH is calculated, $TH_L = TH_n \times YY$ (YY is product specif and need TBD with UX team)
12. All TH values are stored in the NVM by calling writeCalibrationData function.

NOTE

The amount of force steps that we measure are defined with the TDE/MTX team. The minimum is 1, which would be the nominal force. The Maximum is 3, nominal, max measurable force, and back to nominal to measure hysteresis.

ChangeLog

- Version 2: Add Error handling for multiple functions
- Version 1: Add preload adjustment to readMeasurement
- Version 0: Initial version