Release Notes

|  |  |
| --- | --- |
| ***Document #:*** |  |
| ***Title:*** | Release Notes |
| ***Subtitle:*** | Projected Capacitive Firmware, Version 8 |
| ***Date:*** | 4/18/2012 |
| ***Author:*** | Lance Lamont |
| ***Description:*** | Release notes for Version 8 projected capacitive firmware |
|  |  |

Projected Capacitive Firmware, Version 8

**MICROCHIP SOFTWARE NOTICE AND DISCLAIMER:** You may use this software, and any derivatives created by any person or entity by or on your behalf, exclusively with Microchip’s products. Microchip and its licensors retain all ownership and intellectual property rights in the accompanying software and in all derivatives hereto.

This software and any accompanying information is for suggestion only. It does not modify Microchip’s standard warranty for its products. You agree that you are solely responsible for testing the software and determining its suitability. Microchip has no obligation to modify, test, certify, or support the software.

THIS SOFTWARE IS SUPPLIED BY MICROCHIP "AS IS". NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE, ITS INTERACTION WITH MICROCHIP’S PRODUCTS, COMBINATION WITH ANY OTHER PRODUCTS, OR USE IN ANY APPLICATION.

IN NO EVENT, WILL MICROCHIP BE LIABLE, WHETHER IN CONTRACT, WARRANTY, TORT (INCLUDING NEGLIGENCE OR BREACH OF STATUTORY DUTY), STRICT LIABILITY, INDEMNITY, CONTRIBUTION, OR OTHERWISE, FOR ANY INDIRECT, SPECIAL, PUNITIVE, EXEMPLARY, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, FOR COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE SOFTWARE, HOWSOEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWABLE BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THIS SOFTWARE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THIS SOFTWARE.

MICROCHIP PROVIDES THIS SOFTWARE CONDITIONALLY UPON YOUR ACCEPTANCE OF THESE TERMS.

# Table of Contents

[1.0 Table of Contents 3](#_Toc322610807)

[2.0 Summary of Changes 4](#_Toc322610808)

[3.0 Code Re-organization 4](#_Toc322610809)

[4.0 User RAM 5](#_Toc322610810)

[5.0 Communications Protocol 8](#_Toc322610811)

[5.1 Touch Packet 8](#_Toc322610812)

[5.2 Command Communications 8](#_Toc322610813)

[5.3 Diagnostic Messages 8](#_Toc322610814)

[6.0 Diagnostic Message IDs 9](#_Toc322610815)

# Summary of Changes

* New mutual CVD algorithm that uses 4 ADC measurements – eliminates ridging problem and provides nearly twice the signal level
* Implemented new stuttering algorithm to control how many TX lines are pulsed during a self measurement
* Implemented 3 scan types in scanChannels() and scanMutual() for filtering and noise:
  + BASE\_SCAN – scan is being utilized for baseline
  + NOISE\_SCAN – scan is being used for noise detection (no filtering)
  + NORMAL\_SCAN – standard scan used to detect touches
* Minor code improvements to noise detection
* New conditional compilation options for the development kit
* Code re-organization to simplify customer application development

# Code Re-organization

The firmware has now been modified to simplify implementation of customer applications:

* As many processor-specific items as possible have been moved into the processor-specific code and header files
* Processor type is now defied in the project file. The only change necessary to compile the code between processors is to select the proper communications type.
* New conditional compilation options have been added:
  + **DEVELOPMENT** – this changes version numbers to flag that this is development code
  + **DEVKIT\_HARDWARE** – this sizes all arrays to the maximum available for the given processor. If not set, arrays will be the minimum size for the given sensor.
  + **ENABLE\_DEBUG** – this enables all debug functionality
* Minimal-size code (RAM and ROM) is now achieved by leaving DEVKIT\_HARDWARE and ENABLE\_DEBUG undefined during compilation.

# User RAM

Here are the revision 8 user ram values:

|  |  |  |  |
| --- | --- | --- | --- |
| Type | Name | Offset | Description |
| unsigned char | flag1 | 0 | Bitmask - Controls the operating mode of the firmware.   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** | | Unused | Disable touch functionality | Unused | Transmit Min-Max and timestamp | Transmit Raw ADC Values | Controller Processed Diagnostics | Mutual Diagnostics | Self Diagnostics | |
| unsigned char | numberOfRXChannels | 1 | Defines the number of receive channels on the sensor |
| unsigned char | numberOfTXChannels | 2 | Defines the number of transmit channels on the sensor |
| unsigned char | customFlag | 3 | Bitmask – controls acquisition parameters   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** | | Use Differential Acquisition on Mutual | Run Noise Routines on Self and Mutual Scans | Invert Mutual Scan Values | Invert Self Scan values | Pulse 2 TX during Mutual Scan | Use Charge Pump Delay | Use CVD Acquisition | Unused | |
| unsigned short | xmul | 4 | Multiplier used to convert the sensor X axis resolution data to 10-bit resolution data. |
| unsigned short | ymul | 6 | Multiplier used to convert the sensor Y axis resolution data to 10-bit resolution data. |
| General Parameters | | | |
| unsigned char | rxDiagChannel | 8 | Selects RX channel for single-channel diagnostics |
| unsigned char | txDiagChannel | 9 | Selects TX channel for single-channel diagnostics |
| unsigned char | baseUpdateTime | 10 | Set the frequency of the base update. The time is based on loop and will vary based on all parameters. |
| Self Parameters | | | |
| unsigned char | selfScanTime | 11 | Set the number of self capacitance measurements to SUM for 1 measurement |
| unsigned char | selfTouchThres | 12 | Set the threshold to compare the self measurement, if above we may have a touch |
| unsigned char | selfDelayTime | 13 | Set the delay to wait before capacitance measurement after pulsing the TX line(s) in self |
| unsigned char | selfCurrent | 14 | Set CTMU IRNG self: 1 ~ 0.55uA, 2 ~ 5.5uA, 3 ~ 55uA |
| unsigned char | selfSampleFreq | 15 | Sets a delay between self capacitance measurements, currently TMR6 is set 1 = 250nS |
| unsigned char | stutterMult | 16 | For self scans, pulse every <stutterMult>th channel. **Default: 1** |
| Mutual Parameters | | | |
| unsigned char | mutScanTime | 17 | Set the number of mutual capacitance measurements to SUM for 1 measurement |
| unsigned char | mutTouchThres | 18 | Threshold to compare mutual measurement to. If above, analyze for potential touch |
| unsigned char | mutDelayTime | 19 | Set the delay to wait before capacitance measurement after pulsing the TX line(s) in self |
| unsigned char | mutCurrent | 20 | Set CTMU IRNG mutual: 1 ~ 0.55uA, 2 ~ 5.5uA, 3 ~ 55uA |
| unsigned char | mutSampleFreq | 21 | Sets a delay between mutual capacitance measurements, currently TMR6 is set 1 = 250nS |
| Decode & Tracking Parameters | | | |
| unsigned char | flipState | 22 | This determines the orientation of the sensor with respect to the coordinate output. It is a selection of bit flags, with the following values and meanings:  Bit 1 (0x01) – flip X value ( x = 1023 – x)  Bit 2 (0x02) – flip Y value ( y = 1023 – y)  Bit 3 (0x04) – Swap X and Y ( temp = X, X = Y, Y = temp)  The flip operations are performed in the above order in the firmware. |
| unsigned char | numOfAvg | 23 | This parameter configures the number of prior coordinates to average into the current value to smooth the final output. **Default: 8**, Max: TOUCH\_HISTORY, Min: 1 |
| unsigned char | minCuspDelta | 24 | Minimum positive and negative slopes to either side of a peak required to identify a potential touch. **Default: 5**, Max: 255, Min: 1, Recommended Range: Max: 20 |
| unsigned char | weightThreshold | 25 | Weight function value that no longer allows a potential match (any value below this may be a potential match). **Default: 255**, Max: 255, Min: 1 |
| unsigned char | minTouchDistance | 26 | Minimum distance (interpolated coordinates) allowed between two touch locations. If two locations are closer than minTouchDistance, one is suppressed. **Default: 150**, Max: 255, Min: 0, Recommended Range: Max: 255, Min: 100 |
| unsigned char | penDownTimer | 27 | The number of sensor scans in a row that a touch must be identified prior to touch data being transmitted. **Default: 1**, Max: 255, Min: 0, Recommended Range: Max: 5 |
| unsigned char | penUpTimer | 28 | Then number of sensor scans in a row that a touch must NOT be identified prior to a touch up packet being transmitted. Default: 3, Max: 255, Min: 0,  Recommended Range: Max: 5, Min 1 |
| unsigned char | touchSuppressNum | 29 | The maximum number of touch points to transmit. If an ID above touchSuppressNum is allocated, it will not be transmitted. **Default: 0 (disabled)** |
| unsigned short | largeActThres | 30 | Threshold above which a “large activation” |
| Charge Pump Parameters | | | |
| unsigned char | cpTimeOut | 32 | Timeout used for chargepump delay, 1 = 256us |
| Noise Parameters | | | |
| unsigned char | selfNoiseThresh | 33 | threshold for self noise routines to start |
| unsigned char | mutNoiseThresh | 34 | threshold for mutual noise routines to start |
| unsigned char | frequencyChanges | 35 | number of frequencies to try before canceling touch |
| unsigned char | sampleSize | 36 | number of samples to check for noise |
| unsigned char | selfNoiseScanTime | 37 | number of self scans to take when checking for noise |
| unsigned char | mutNoiseScanTime | 38 | number of mutual scans to take when checking for noise |
| unsigned char | filterCoeff | 39 | MA filter coefficient value (tied to both self and mutual) If zero, filters are turned off |
| Port Maps | | | |
| unsigned char | rxPinMap[MAXRX] | 40 | Defines which pins, in sensor order, on the controller are utilized to perform measurements on the sensor. |
| unsigned char | txPinMap[MAXTX] |  | Define which pins, in sensor order, on the controller are utilized to transmit pulses on the sensor. |

# Communications Protocol

## Touch Packet

This packet is transmitted whenever a touch is detected on the sensor.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Packet\Bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **0** | 1 | **T3** | **T2** | **T1** | **T0** | P2 | P1 | P0 |
| **1** | 0 | X6 | X5 | X4 | X3 | X2 | X1 | X0 |
| **2** | 0 | 0 | 0 | X11 | X10 | X9 | X8 | X7 |
| **3** | 0 | Y6 | Y5 | Y4 | Y3 | Y2 | Y1 | Y0 |
| **4** | 0 | 0 | 0 | Y11 | Y10 | Y9 | Y8 | Y7 |

T3, T2, T1, T0: Touch Packet ID (Currently uses T2,T1,T0 for IDs 0-7)

P2,P1,P0: Pen State (Currently only uses P0 for Pen Down/Pen Up)

X11,X10,…,X0: X Coordinate of touch

Y11,Y10,…,Y0: Y Coordinate of touch

## Command Communications

This protocol is used for all bi-directional communications with the controller.

Command (Sent to firmware):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **0x55** | <size> | <command> | <data 0> | … | <data N> |

Command Response (from firmware):

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **0x55** | <size> | <result> | <command> | <data 0> | … | <data N> |

Size: Number of bytes remaining in the packet. Minimum size for a response is 2 – result and command, with no data.

Result: 0 for success, Non-zero for failure.

Command: The command the firmware is responding to (for synchronization)

Data 0 – Data N: The bytes of data in the response.

Potential Command Results:

**0x00** – DEFAULTSUCCESS – the command was completed successfully

**0xFE** – COMMANDTIMEOUT – An entire command was not received within timeout

**0xFF** – UNRECOGNIZEDCOMMAND – The command was not recognized

## Diagnostic Messages

Diagnostic messages, if enabled via the diagnostic mask, may be inserted into the standard data stream.

Diagnostic Message format (from firmware):

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **0xAA** | **0x55** | <size> | <diagnostic ID> | <data 0> | … | <data N> |

# Command Set

All commands are documented in comm.c – this is an overview of the available commands:

|  |  |  |
| --- | --- | --- |
| ID | Name | Description |
| 0x00 | ENABLECONTROLLER | Enable touch functionality |
| 0x01 | DISABLECONTROLLER | Disable touch functionality |
| 0x14 | SCANBASELINE | Force the firmware to scan a new baseline of the entire sensor |
| 0x15 | WRITERAM | Write a given byte at a given offset in RAM |
| 0x16 | READRAM | Read a given offset in RAM |
| 0x17 | WRITEUSEREEPROM | Write RAM to EEPROM (on devices that support EEPROM) |
| 0x18 | SOFTWARESLEEP | Enter a sleep mode |
| 0x19 | ERASEEEPROM | Erase the contents of EEPROM |
| 0x1A | CHECKIO | Perform a check of the sensor |
| 0x80 | CFGIDHIGHCMD | Return the HIGH config ID of the firmware |
| 0x81 | CFGIDLOWCMD | Return the LOW config ID of the firmware |
| 0xD0 | GETDIAGMASKCMD | Retrieve the current Diagnostic Mask (which Diagnostic IDs will be transmitted) |
| 0xD1 | SETDIAGMASKCMD | Set the current Diagnostic Mask |

# Diagnostic Message IDs

All Diagnostic Messages now have a valid, unique ID and are all transmitted by the sendDebugData function. That function implements a fairly simple diagnostic mask system. It will only transmit diagnostic messages with an ID that is present in the current diagnostic mask. Use the SETDIAGMASKCMD and GETDIAGMASKCMD commands to modify which entries are in the diagnostic mask.

|  |  |  |
| --- | --- | --- |
| **ID** | **Name** | **Description** |
| **0x01** | SELFRAWDIAGNOSTICS | Self Raw data |
| **0x02** | MUTUALRAWDIAGNOSTICS | Mutual Raw data |
| **0x05** | SELFCONTROLLERDIAGNOSTICS | Self Controller Processed data (after baselining) |
| **0x06** | MUTUALCONTROLLERDIAGNOSTICS | Mutual Controller Processed data |
| **0x09** | RAWSELFADCDIAGNOSTICS | Self Raw ADC measurements (typically for noise testing) |
| **0x0A** | RAWMUTUALADCDIAGNOSTICS | Mutual Raw ADC measurements |
| **0x11** | LONGSELFSCANDIAGNOSTICS | Bulk Self Raw ADC measurements – 512 values |
| **0x12** | LONGMUTUALSCANDIAGNOSTICS | Bulk Mutual Raw ADC measurements – 512 values |
| **0x1A** | FINDTOUCHES | Diagnostics for the findTouches function |
| **0x1B** | FINDNEXTPEAK | Diagnostics for the findNextPeak function |
| **0x1C** | FINDFINELOCATION | Diagnostics for touch interpolation (findFineLocation) |
| **0x1F** | DIAGIMAGESTART | Diagnostic message that indicate the beginning of a decode cycle |
| **0x1E** | DIAGIMAGEEND | Not used |
| **0x20** | DIAGSELFDATA | Self Data captured during touch decoding |
| **0x21** | DIAGMUTDATA | Mutual Data captured during touch decoding |
| **0x22** | TOUCHDATAROUGH | Rough touch location early in the processing cycle |
| **0x23** | TOUCHDATAFINE | Fine(interpolated) touch location early in the processing cycle |
| **0x24** | DIAGNUDGE | Diagnostic information for the Nudge function |
| **0x25** | DIAGCOLCACHE | Diagnostics for the mutual measurement column cache  This ID has several “sub-IDs” for different data. |
| **0x26** | DIAGTOUCHREPORT | Touch Report Diagnostics – contains all information on final touch location. |
| **0x27** | DIAGASSOCIATETOUCH | Diagnostics for the touch association function. |
| **0x30** | DIAGFINEX | Fine X Location from the interpolation function |
| **0x31** | DIAGFINEY | Fine Y Location from the interpolation function |
| **0x32** | SELFNOISEDEV | Self Noise Deviation information |
| **0x33** | SELFNOISEFREQ | Self Noise Frequency information |
| **0x34** | MUTNOISEDEV | Mutual Noise Deviation information |
| **0x35** | MUTNOISEFREQ | Mutual Noise Frequency information |