Date: 1-26-12

Subject: PICPCAP Version 5.2 Release Notes

Re: Version 5

From: LDL

# Summary

The following lists a summary of the changes:

* Added the new decoding algorithms. Currently fixed at up to 8 touches, can be adjusted to do less or more, RAM dependent. The new decoding algorithms are less sensitive to thresholds
* Updated communications protocols for multi-byte responses to commands
* Added significant debugging communications functionality. Created “sendDebugBytes” function to transmit all diagnostic messages.
* Converted all existing diagnostic messages over to use sendDebugBytes system.
* Added Diagnostic message filtering capability (commands SETDIAGMASKCMD and GETDIAGMASKCMD). NOTE – this can cause modes to appear to not be operating correctly, as the diagnostic messages transmitted during that mode may be suppressed.
* Significant clean-up of communications functions to improve readability and decrease code size.

# Diagnostic Masking

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.

Current Diagnostic IDs:

|  |  |  |
| --- | --- | --- |
| **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 |

# Communications Changes

These are the changes performed to the communications packets:

## Touch Packet Modifications

The location of the touch IDs has been modified in order to support additional touch IDs.

Original Touch Packet:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Packet\Bit** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **0** | 1 | T1 | T0 | 0 | 0 | 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 |

T1, T0: Touch Packet ID (Currently only uses T0 for ID0 / ID1)

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

New Touch Packet:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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

## Communications Protocol Changes

No modifications have been made to the format of the communications sent TO the controller, but modifications have been made to the responses from the controller to allow for multiple bytes of response data.

Original Communications Protocol

Command:

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

Old Command Response (from firmware)

|  |  |  |
| --- | --- | --- |
| **0x55** | **0x01** | <result/data> |

This response format would only allow a single result byte.

New Command Response:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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 the timeout

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

## Command Set

Two new commands were added to the command set:

**0xD0** – GETDIAGMASKCMD

Send

|  |  |  |
| --- | --- | --- |
| **0x55** | **0x01** | **0xD0** |

Receive

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0x55** | <size> | <result> | **0xD0** | <Max Mask Size> | <Mask 0> | … | <Mask n> |

Description:

This command retrieves all of the non-0xff values in the current diagnostic mask. (0xff is defined as “do not transmit”)

**0xD1** – SETDIAGMASKCMD

Send

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **0x55** | **0x01** | **0xD1** | <Mask 0> | … | <Mask n> |

Receive

|  |  |  |  |
| --- | --- | --- | --- |
| 0x55 | <size> | <result> | 0xD1 |

Description:

This command sets the entire contents of the diagnostic mask. Up to <Max Mask Size> mask bytes may be transmitted to the controller. If fewer than <Max Mask Size> bytes are transmitted, the remaining entries in the mask are filled with **0xff** (do not transmit).

# User Ram Parameters

Here is the entirety of userRam, with descriptions for the new parameters

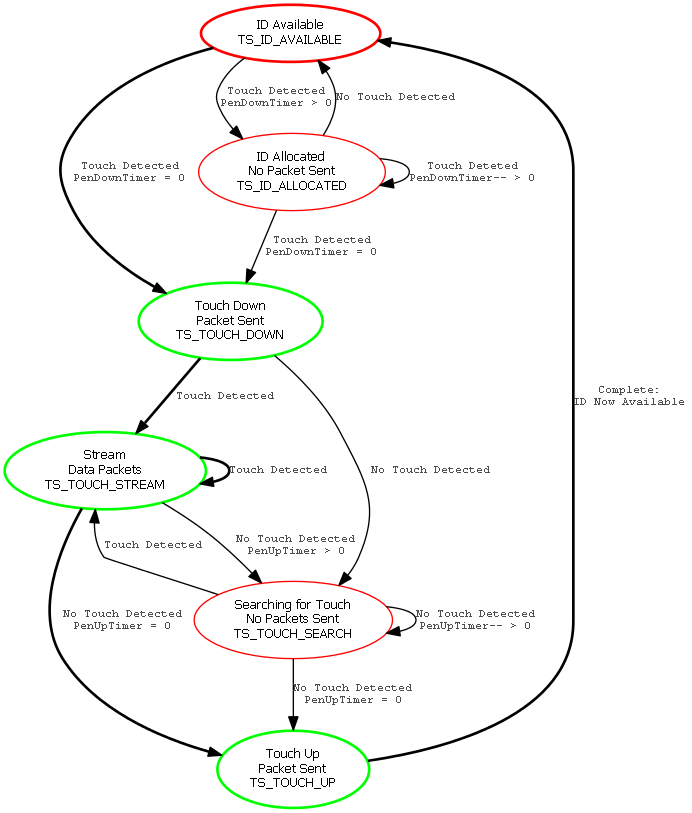
|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Offset | Description |
| flag1 | unsigned char | 0 |  |
| numberOfRXChannels | unsigned char | 1 |  |
| numberOfTXChannels | unsigned char | 2 |  |
| customFlag | unsigned char | 3 |  |
| xmul | unsigned short | 4 |  |
| ymul | unsigned short | 6 |  |
| General Parameters | | | |
| rxDiagChannel | unsigned char | 8 |  |
| txDiagChannel | unsigned char | 9 |  |
| adcADCS | unsigned char | 10 |  |
| shChargeTime | unsigned char | 11 |  |
| baseUpdateTime | unsigned char | 12 |  |
| Self Parameters | | | |
| selfScanTime | unsigned char | 13 |  |
| selfTouchThres | unsigned char | 14 |  |
| selfDivider | unsigned char | 15 |  |
| selfDelayTime | unsigned char | 16 |  |
| selfCurrent | unsigned char | 17 |  |
| maxminSelfThres | unsigned char | 18 |  |
| selfSampleFreq | unsigned char | 19 |  |
| selfOversample | unsigned char | 20 |  |
| selfReplaceThres | unsigned char | 21 |  |
| selfLPFilterValue | unsigned char | 22 |  |
| selfMaxDelta | unsigned char | 23 |  |
| PVCFGSelf | unsigned char | 24 |  |
| FVRSelf | unsigned char | 25 |  |
| Mutual Parameters | | | |
| mutScanTime | unsigned char | 26 |  |
| mutThres | unsigned char | 27 |  |
| mutDivider | unsigned char | 28 |  |
| mutDelayTime | unsigned char | 29 |  |
| mutCurrent | unsigned char | 30 |  |
| maxminMutThres | unsigned char | 31 |  |
| mutSampleFreq | unsigned char | 32 |  |
| mutOversample | unsigned char | 33 |  |
| mutReplaceThres | unsigned char | 34 |  |
| mutLPFilterValue | unsigned char | 35 |  |
| mutMaxDelta | unsigned char | 36 |  |
| FVRMut | unsigned char | 37 |  |
| PVCFGMut | unsigned char | 38 |  |
| Decode & Tracking Parameters | | | |
| flipState | unsigned char | 39 | 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. |
| numOfAvg | unsigned char | 40 | 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 |
| minCuspDelta | unsigned char | 41 | 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 |
| weightThreshold | unsigned char | 42 | 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 |
| minTouchDistance | unsigned char | 43 | 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 |
| penDownTimer | unsigned char | 44 | 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 |
| penUpTimer | unsigned char | 45 | 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 |
| Port Maps | | | |
| rxPinMap[MAXRX] | unsigned char | Starts at 46 |  |
| txPinMap[MAXTX] | unsigned char | Starts at 46 + MAXRX |  |

# Decode & Tracking Algorithm

## Key Concepts

1. Touch State Machine

Every touch utilizes the following state machine:



All green states include data packets being transmitted. All red states do not transmit data. The bold lines (around the “outside”) are the minimum state sequence, although during normal operation all states are utilized.

## Functions & Short Descriptions

For full documentation, please review the Doxygen output located in the Doxygen directory of the firmware release.

|  |  |
| --- | --- |
| Functions | |
| void | **decodeInit** (void) |
|  | Initializes all decoding related data structures. |
| unsigned char | **findTouches** (void) |
|  | drives the touch decoding algorithm. Loads potential touch locations into the touchSet data structure. |
| unsigned char | **findNextPeak** (unsigned short \*values, unsigned char numValues, unsigned char startLocation, unsigned char threshold) |
|  | Analyzes the data in 'values' and attempts to identify peaks or 'cusps' in the data. Will flag significant changes in slope or a change from a positive to a negative slope. |
| unsigned char | **nudgeLoc** (**TOUCHDATA** \*tData) |
|  | Analyzes selected node and adjacent nodes to identify where the "true" peak is. If current node is not the highest point, it "nudges" that point to the higher location, then re-analyzes. |
| unsigned char | **findFineLocation** (**TOUCHDATA** \*tData) |
|  | Analyzes the mutual measurements of the points adjacent to the peak point to interpolate a higher resolution location. |
| unsigned char | **associateTouches** (void) |
|  | Associates touches in touchSet with touch IDs in touchIDset. This is done by looking at a "matrix" of touch weights, which are determined by using the 'touchWeight' function. The "best" match for all touches will attempt to be made, including using a slightly worse weight for one touch to allow for a better weight for another touch. |
| unsigned char | **touchWeight** (unsigned char touchID, unsigned char potentialTouch) |
|  | Generates a "weight" for a given touch point with regard to an existing touch ID. The lower the return value, the better the match. |
| unsigned char | **simpleDistance** (**UINTCOORD** pointA, **UINTCOORD** pointB) |
|  | Provides a simple distance between the two given points. avoids all "high" math - divide, square, square root, etc. So generates a "manhattan" distance - that is, the distance on the X axis, plus the distance on the Y axis. |
| void | **extrapolateTouch** (**TOUCHID** \*thisTouch, **UINTCOORD** \*next) |
|  | Linear extrapolation of where the next touch location will be, given the history of the associated touches with the ID. |
| void | **handleTouches** (void) |
|  | Looks through every touch ID, process and transmits those that are in a state that should be transmitted - TOUCH\_DOWN, TOUCH\_STREAM, and TOUCH\_UP are the only states that should be transmitted. |