Release Notes
Projected Capacitive Firmware, Version 9 (MTCH6301)

PRELIMINARY

Document #:	
Title:	Release Notes
Subtitle:	Projected Capacitive Firmware, Version 9 (MTCH6301)
Date:	9/25/2012
Author:	Lance Lamont
Description:	Release notes for Version 9 projected capacitive firmware. MTCH6301

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.

PRELIMINARY

1.0 Table of Contents

1.0 TABLE OF CONTENTS 3

2.0 SUMMARY OF CHANGES 4

3.0 CODE RE-ORGANIZATION 4

4.0 USER RAM 5

4.1 High Offsets 5

4.2 Full Ram listing 6

5.0 COMMUNICATIONS PROTOCOL 12

5.1 Touch Packet 12

5.2 Gesture Packet 12

5.3 Command Communications 12

5.4 Diagnostic Messages 13

6.0 DIAGNOSTIC MESSAGE IDS 14

PRELIMINARY

2.0 Summary of Changes

- 12bit Touch Packets instead of 10bit
- Focused release for the PIC32 (PIC18 & PIC24 code not verified)
- Implemented multiple high offsets for RAM values to separate RAM into logical groups
    - Re-wrote RAM read/write functions for readability
    - Several New RAM structures:
        - SelfFineTune – fine tuning of individual self channels
        - MutFineTune – fine tuning of individual mutual channels
        - HWCfgRam – hardware configuration parameters
        - HWStatusRam – hardware status parameteres
- Implemented Display driver code for Touch Pad Demonstrator
- Basic Gesture Support – Swipe, touch, swipe-hold, touch-hold, double touch
- Added New packets
    - Gesture Packets
    - Status Packets
- Added functionality to enable/disable individual packet types
- Separated decode state machine from comm functionality
- Implemented a global counter for most timing functionality
- Sleep functionality
    - Select IDLE, SLEEP on PIC32
- New Sensor Files created for some sensors
- Removed CTMU functionality, now using CVD exclusively

3.0 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.

PRELIMINARY

4.0 User RAM

User RAM has been modified to utilize both High and Low offsets for the values. High offsets define a category of RAM values, and low offsets index into that category:

4.1 High Offsets

Value	Name	Description
0x00	UserRam “general”	Normal access to UserRam for legacy support, also flag1, numRX, numTX, customFlag, xmul, ymul
0x01	rxPinMap	Direct access to rxPinMap
0x02	txPinMap	Direct access to txPinMap
0x10	Self Parameters	Starts at UserRam.selfScanTime
0x11	Self Tuning Array	Access to the individual channel self tuning array
0x20	Mutual Parameters	Starts at UserRam.mutScanTime
0x21	Mutual Tuning Array	Access to the individual channel mutual tuning array
0x30	Decode & Tracking	Starts at UserRam.flipState
0x40	Noise & Charge Pump	Starts at UserRam.cpTimeOut
0x50	Gestures	Starts at UserRam.swipeLengthX
0xF0	Hareware Config	Access to the Hardware Config Ram structure
0xF1	Hardware Status	Access to the Hardware Status Ram structure

PRELIMINARY

4.2 Full Ram listing

4.2.1 High Offset 0x00: UserRam "general"

Type	Name	HIGH	LOW	Description
unsigned char	flag1	0x00	0x00	Bitmask - Controls the operating mode of the firmware. 7: Unused 6: Disable touch functionality 5: Unused 4: Transmit Min-Max and timestamp 3: Values Transmit Raw ADC 2: Diagnostics Controller Processed 1: Mutual Diagnostics 0: Self Diagnostics
unsigned char	numberOfRXChannels	0x00	0x01	Defines the number of receive channels on the sensor
unsigned char	numberOfTXChannels	0x00	0x02	Defines the number of transmit channels on the sensor
unsigned char	customFlag	0x00	0x03	Bitmask – controls acquisition parameters 7: Use Differential Acquisition on Mutual 6: Self and Mutual Scans Run Noise Routines on 5: Values Invert Mutual Scan 4: Invert Self Scan values 3: Mutual Scan Pulse 2 TX during Delay 2: Use Charge Pump 1: Use CVD Acquisition 0: Unused
unsigned short	xmul	0x00	0x04	Multiplier used to convert the sensor X axis resolution data to 10-bit resolution data.
unsigned short	ymul	0x00	0x06	Multiplier used to convert the sensor Y axis resolution data to 10-bit resolution data.
unsigned char	rxDiagChannel	0x00	0x08	Selects RX channel for single-channel diagnostics
unsigned char	txDiagChannel	0x00	0x09	Selects TX channel for single-channel diagnostics
unsigned char	baseUpdateTime	0x00	0x0A	Set the frequency of the base update. The time is based on loop and will vary based on all parameters.
unsigned char	stuckThreshold	0x00	0x0B	Maximum distance a touch can move before it is no longer a possible stuck touch
unsigned short	stuckTimeout	0x00	0x0C	Time a touch must be stationary for it to be considered "stuck"

PRELIMINARY

4.2.2 High Offset 0x01: rxPinMap

Type	Name	HIGH	LOW	Description
unsigned char	rxPinMap[MAXRX]	0x01	0x00	Defines which pins, in sensor order, on the controller are utilized to perform measurements on the sensor.

4.2.3 High Offset 0x02: txPinMap

Type	Name	HIGH	LOW	Description
unsigned char	txPinMap[MAXTX]	0x02	0x00	Define which pins, in sensor order, on the controller are utilized to transmit pulses on the sensor.

4.2.4 High Offset 0x10: Self Parameters

Type	Name	HIGH	LOW	Description
unsigned char	selfScanTime	0x10	0x00	Set the number of self capacitance measurements to SUM for 1 measurement
unsigned char	selfTouchThres	0x10	0x01	Set the threshold to compare the self measurement, if above we may have a touch
unsigned char	selfSampleFreq	0x10	0x02	Sets a delay between self capacitance measurements, currently TMR6 is set 1 = 250nS
unsigned char	stutterMult	0x10	0x03	For self scans, pulse every <stutterMult>th channel. **Default: 1**

4.2.5 High Offset 0x11: Self Tuning Array

Type	Name	HIGH	LOW	Description
unsigned char	selfScanFineTune[MAXTX]	0x11	0x00	Value added to selfScanTime for individual channels

4.2.6 High Offset 0x20: Mutual Parameters

Type	Name	HIGH	LOW	Description
unsigned char	mutScanTime	0x20	0x00	Set the number of mutual capacitance measurements to SUM for 1 measurement
unsigned char	mutTouchThres	0x20	0x01	Threshold to compare mutual measurement to. If above, analyze for potential touch
unsigned char	mutSampleFreq	0x20	0x02	Sets a delay between mutual capacitance measurements, currently TMR6 is set 1 = 250nS

PRELIMINARY

4.2.7 High Offset 0x21: Mutual Tuning Array

Type	Name	HIGH	LOW	Description
unsigned char	mutScanFineTune[MAXTX]	0x21	0x00	Value added to mutScanTime for individual channels

4.2.8 High Offset 0x30: Decode & Tracking

Type	Name	HIGH	LOW	Description
unsigned char	flipState	0x30	0x00	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	0x30	0x01	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	0x30	0x02	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	0x30	0x03	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	0x30	0x04	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	0x30	0x05	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	0x30	0x06	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	0x30	0x07	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	0x30	0x08	Threshold above which a "large activation"

4.2.9 High Offset 0x40: Noise & Charge Pump

Type	Name	HIGH	LOW	Description
unsigned char	cpTimeOut	0x40	0x00	Timeout used for chargepump delay, 1 = 256us
unsigned char	selfNoiseThresh	0x40	0x01	threshold for self noise routines to start
unsigned char	mutNoiseThresh	0x40	0x02	threshold for mutual noise routines to start
unsigned char	frequencyChanges	0x40	0x03	number of frequencies to try before canceling touch
unsigned char	sampleSize	0x40	0x04	number of samples to check for noise
unsigned char	selfNoiseScanTime	0x40	0x05	number of self scans to take when checking for noise
unsigned char	mutNoiseScanTime	0x40	0x06	number of mutual scans to take when checking for noise
unsigned char	filterCoeff	0x40	0x07	MA filter coefficient value (tied to both self and mutual) If zero, filters are turned off

4.2.10 High Offset 0x50: Gestures

Type	Name	HIGH	LOW	Description
unsigned char	swipeLengthX	0x50	0x00	Minimum distance (in interpolated positions) the user must swipe in the x-direction to register the gesture
unsigned char	swipeLengthY	0x50	0x01	Minimum distance (in interpolated positions) the user must swipe in the y-direction to register the gesture
unsigned char	holdSwipeBoundary	0x50	0x02	The amount of distance (in interpolated positions) a swipe can move in the direction opposite to the direction being swiped before the gesture is cancelled.
unsigned char	swipeHoldThresh	0x50	0x03	The maximum distance (in interpolated positions) a swipe-and-hold gesture can move before the gesture is cancelled

PRELIMINARY

unsigned short	swipeTime	0x50	0x04	The maximum amount of time (in ms) the user has to perform a swipe after initial pen down
unsigned short	tapTime	0x50	0x06	The maximum amount of time (in ms) the user has to perform a click after initial pen down
unsigned char	tapThresh	0x50	0x08	The maximum distance (in interpolated positions) a tap gesture can move before it is no long recognized as a tap
unsigned char	minSwipeVelocity	0x50	0x09	The minimum velocity a swipe must maintain to be a swipe gesture. Values below this will either cancel the gesture (if touch removed) or move to the swipe-and-hold state (if touch is still present)
unsigned short	maxClickTime	0x50	0x0A	This is the maximum amount of time allowed between the two taps of a double tap (in ms)
unsigned char	edgeKeepoutDistance	0x50	0x0C	This value determines the width of a keepout barrier around the edge of the active touch area. This helps remove edge effect issues.

4.2.11 High Offset 0xF0: Hardware Configuration

Type	Name	HIGH	LOW	Description
unsigned int	sleepTimeout	0xF0	0x00	Number of milliseconds of no interaction before the controller goes to sleep. Value of 0 disables sleep.
unsigned char	sleepConfig	0xF0	0x04	Configures type of sleep: None=0, Idle=1, Sleep=2
unsigned char	wdtTimeout	0xF0	0x05	Timeout for the watchdog timer in the controller (hardware disabled on the MTCH6301)
unsigned char	diagPacketCfg	0xF0	0x06	Diagnostic Packet Configuration – Enabled: 0x81, Disabled: 0x01
unsigned char	touchPacketCfg	0xF0	0x07	Touch Packet Configuration – Enabled: 0x81, Disabled: 0x01
unsigned char	commandPacketCfg	0xF0	0x08	Command Packet Configuration – Enabled: 0x81, Disabled: 0x01
unsigned char	gesturePacketCfg	0xF0	0x09	Gesture Packet Configuration – Enabled: 0x81, Disabled: 0x01
unsigned char	statusPacketCfg	0xF0	0x0A	Status Packet Configuration – Enabled: 0x81, Disabled: 0x01

4.2.12 High Offset 0xF1: Hardware Status

Type	Name	HIGH	LOW	Description
unsigned char	generalStatus	0xF1	0x00	General operational status of the

				firmware (reserved for future)
unsigned int	txShortStatus	0xF1	0x02	Identifies which TX pins are shorted after calling checkIO
unsigned int	rxShortStatus	0xF1	0x06	Identifies which RX pins are shorted after calling checkIO

PRELIMINARY

5.0 Communications Protocol

5.1 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	0	0	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)
P0: Pen State - P0 for Pen Down/Pen Up
X11,X10,…,X0: X Coordinate of touch
Y11,Y10,…,Y0: Y Coordinate of touch

5.2 Gesture Packet

This packet is transmitted whenever a gesture is detected on the sensor. NOTE: Gestures are disabled by default via “gesturePacketCfg”.

Packet\Bit	7	6	5	4	3	2	1	0
0	1	T3	T2	T1	T0	1	0	0
1	0	G6	G5	G4	G3	G2	G1	G0

T3, T2, T1, T0: Touch Packet ID (Currently uses T2,T1,T0 for IDs 0-7)
G6,..,G0: Gesture ID

Gesture IDs:

Gesture Name	Gesture Value
GESTURE_SINGLETAP	0x10
GESTURE_HOLDTAP	0x11
GESTURE_DOUBLETAP	0x20
GESTURE_RIGHT_SWIPE	0x41
GESTURE_RIGHT_SWIPE_HOLD	0x42
GESTURE_LEFT_SWIPE	0x61
GESTURE_LEFT_SWIPE_HOLD	0x62
GESTURE_UP_SWIPE	0x31
GESTURE_UP_SWIPE_HOLD	0x32
GESTURE_DOWN_SWIPE	0x51
GESTURE_DOWN_SWIPE_HOLD	0x52

5.3 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>

PRELIMINARY

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:

Value	Name	Description
0xFC	PARAMETERCOUNTERROR	Missing or Extra parameter for the given command
0xFD	INVALIDPARAMETER	Invalid parameter for the given command
0xFF	UNRECOGNIZEDCOMMAND	Response to an unrecognized command
0xFE	COMMANDTIMEOUT	Response to a partial command
0x80	PARAMETEROUTOFRANGE	Response if a command is sent with data that is out of range
0x00	DEFAULTSUCCESS	Response when the controller successfully completes a command

5.4 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

PRELIMINARY

6.0 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

PRELIMINARY