AN11620

LPC82x Touch Solution Application Note

Rev. 1.0 — 22 December 2014 Application Note

Document information

Info	Content
Keywords	LPC82x Touch Solution, User Guide, Library, Capacitive Touch, Touchpad, Sensor, Electrode, Drive/Sensing lines, Dielectric, Overlay panel, Sensitivity, Touch, False Touch
Abstract	This document describes the LPC82x Touch Solution user application in detail. It contains a functional description of the Touch library APIs, library usage with example, and windows Graphical User Interface details.

Revision history

Rev	Date	Description
1.0	20141222	Initial version

Contact information

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

1. Introduction

NXP's LPC82x Touch Solution is centered around LPC824, ARM Cortex-M0+ MCU running up to 30MHz with 32 KB internal flash, and 8 KB SRAM. The solution consists of complete hardware and software platform for enabling the Touch user interface designs. It is based on a royalty free "LPC8xx Touch Library" and OM13081 kit. The Touch library drives/scans the Touch sensors, processes data for robust/noise-free Touch detection, and provides the Touch data to host/user application. This application note describes the Touch library usage in user (host) application and the Graphical User Interface tool to visualize Touch data as well as configure Touch parameters.

This document contains the following sections:

- Touch Solution Overview
- User application
- LPC824_Touch_Example
- Touch Graphical User Interface

2. Overview

Fig 1 shows the complete system architecture of NXP's Touch Solution. The physical interface and circuit schematic representation is on the right side, while the functional representation using Touch Library is on the left side.

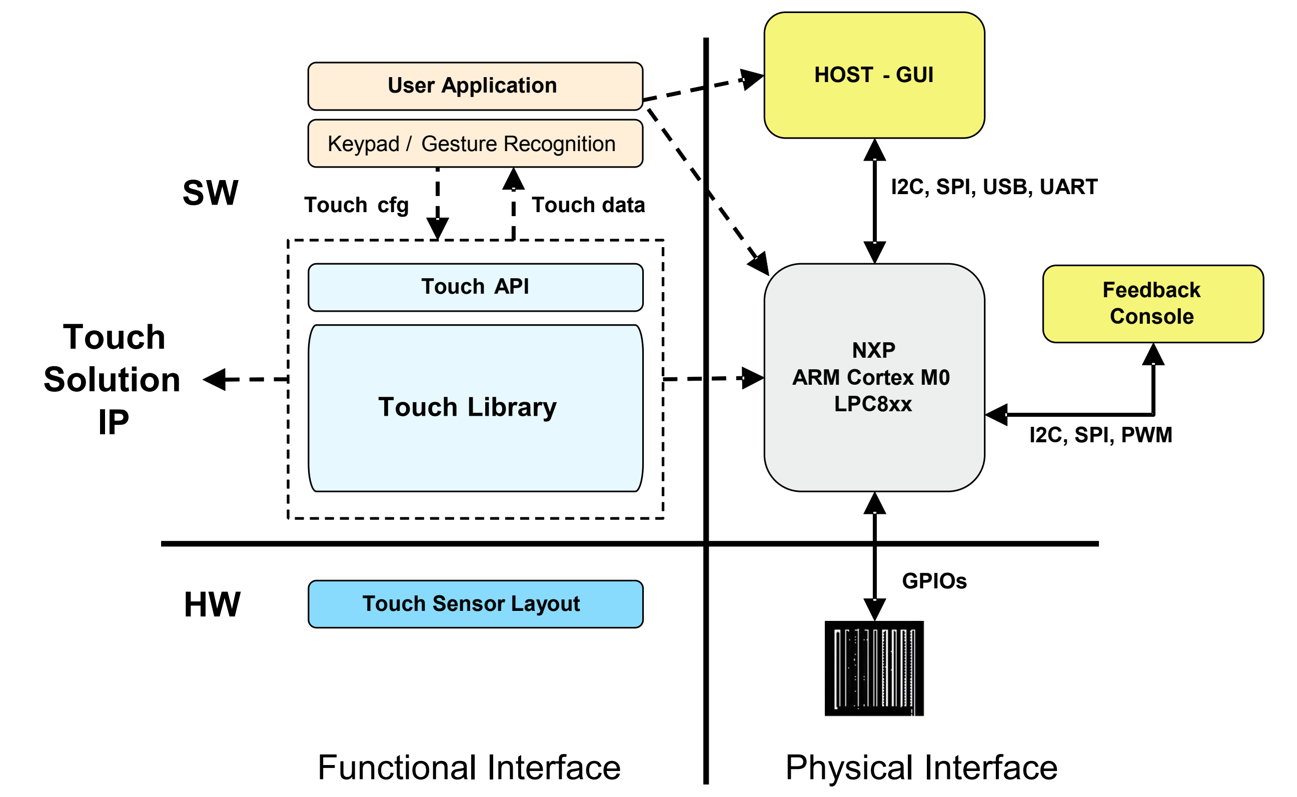


Fig 1. LPC8xx Touch Solution Overview

The Touch Library running on LPC824 drives/scans the physical capacitive sensors, processes the Touch signal data, filters out the noise and finally provides this Touch data to the user application. The user application can then utilize this Touch data to achieve the desired functionality. The Touch library provides complete flexibility in configuring the sensors (one or two dimensional sensing mode) and Touch parameters even during the run time. In one dimensional mode, the Touch library provides single sensor Touch/no Touch status, while it provides the Touch position (X-Y coordinates) in two dimensional mode. The Touch position calculation algorithm adapts automatically to a sensor layout of 3x3 matrix and the associated user settings. The position calculation is robust against ghost touches or sensor element non-uniformities.

Touch API layer handles all data and command exchange between the user application and the Touch library. See the “LPC8xx Touch Solution Library User Guide” for details on all the supported APIs.

2.1 Microcontroller Resources

The Touch Library acquisition method requires one GPIO pin (X) per sensor, two additional GPIO pins (YH and YL), and it uses the analog comparator in the process. These GPIO pins must be able to switch between High Impedance, Input and Output (High/Low), preferably with low pad/stray capacitance. Furthermore, the library uses one MRT (Multi-Rate Timer) channel (CH0 and CH1). These resources are dedicated for Touch and cannot be used by the user (host) application. The library uses just over 3 kBytes Flash and around 80 bytes SRAM (compiled with Keil V5.10).

2.2 Performance indicators & Constraints

Number of sensors

The current Touch Library supports a maximum of 9 Touch sensors in one dimensional and two dimensional modes. In two dimensional mode, the Touch position calculation algorithm adapts automatically to a sensor layout of 3x3 matrix and the associated user settings. That means, the number of rows (MAX_ROW_CNT) and columns (MAX_COL_CNT) are fixed as 3.

Timing

A complete cycle of scanning 9 sensor elements for Touch detection, eventual signal condition and position calculation takes about 3.6 milliseconds[1] (approximately 400us per sensor). This results in around 278 sample points per second with a spatial resolution of over 80dpi.

Interrupts

The Touch Solution library itself does not use any interrupts. However, during the most time critical part of the charge cycle time measurement (inside Touch core loop), interrupts are shortly disabled (generally some tenths of CCLK cycles). This can influence the real time behavior of the users’ end application.

1. Measured with LPC824 Microcontroller running at 24 MHz clock.

3. User Application

Download the latest “LPC82x Touch Library” and “LPC82x Touch Solution Library User Guide”. The user application should handle the following sequence of actions in order to use the Touch Library and add Touch functionality to the user code.

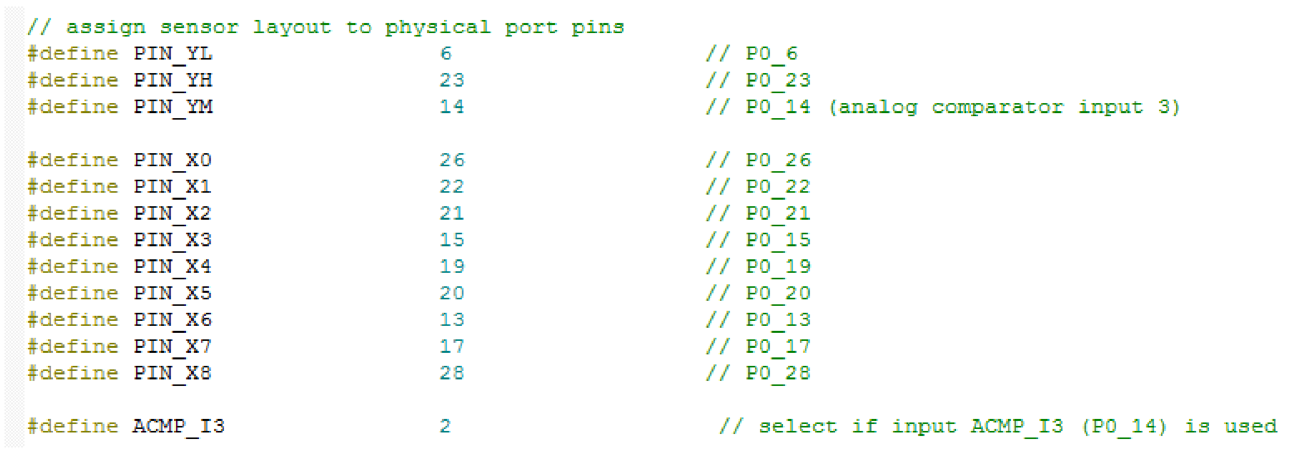
3.1.1 Include Touch Library

Ensure that the Touch library is part of your project by including the library (`LPC8xx_Touch_lib_v01.lib`) and library header file (`LPC8xx_Touch_Solution.h`) in your project. Also, include the library header file inside the source.

```
#include "LPC8xx_Touch_Solution.h”
```

3.1.2 Define and Initialize Sensors

Assign Touch sensors to microcontroller port pins. Obviously, these definitions would resemble to the Touch hardware design (number of sensors & their physical connections). As an example, let’s refer to the Keil example project “LPC824_Touch_Example1”. It is implemented for a sensor matrix of 3x3 and the sensor connections are shown in Fig 2.



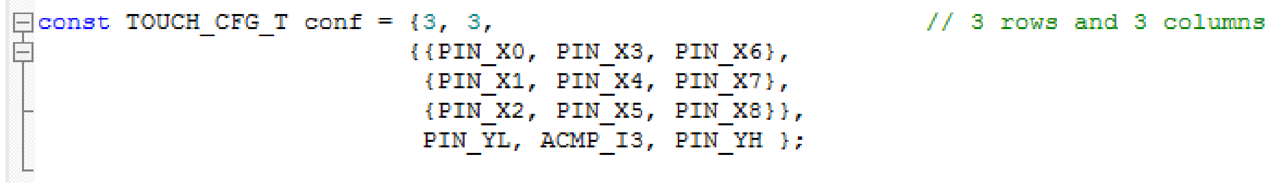
```
// assign sensor layout to physical port pins
#define PIN_YL              6                   // P0_6
#define PIN_YH              23                  // P0_23
#define PIN_YM              14                  // P0_14 (analog comparator input 3)

#define PIN_X0              26                  // P0_26
#define PIN_X1              22                  // P0_22
#define PIN_X2              21                  // P0_21
#define PIN_X3              15                  // P0_15
#define PIN_X4              19                  // P0_19
#define PIN_X5              20                  // P0_20
#define PIN_X6              13                  // P0_13
#define PIN_X7              17                  // P0_17
#define PIN_X8              28                  // P0_28

#define ACMP_I3             2                    // select if input ACMP_I3 (P0_14) is used
```

Fig 2. Define Sensors Pins

Initialize the sensor configuration structure after these sensor pins are defined.

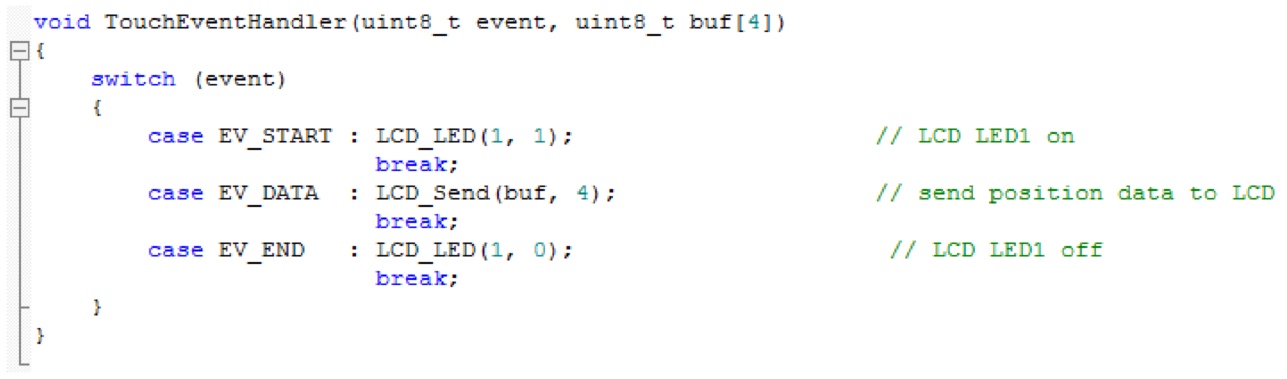


```
const TOUCH_CFG_T conf = {3, 3,                          // 3 rows and 3 columns
                         {{PIN_X0, PIN_X3, PIN_X6},
                          {PIN_X1, PIN_X4, PIN_X7},
                          {PIN_X2, PIN_X5, PIN_X8}},
                          PIN_YL, ACMP_I3, PIN_YH };
```

Fig 3. Initialize Sensors Configuration Structure

3.1.3 Define Touch Handler

In the user code, define a function that will be used as a call back by the Touch library and that handles the Touch events. See Fig 4 as an example:



```
void TouchEventHandler(uint8_t event, uint8_t buf[4])
{
    switch (event)
    {
        case EV_START : LCD_LED(1, 1);                      // LCD LED1 on
                        break;
        case EV_DATA  : LCD_Send(buf, 4);                   // send position data to LCD
                        break;
        case EV_END   : LCD_LED(1, 0);                       // LCD LED1 off
                        break;
    }
}
```

Fig 4. Touch Event Handler

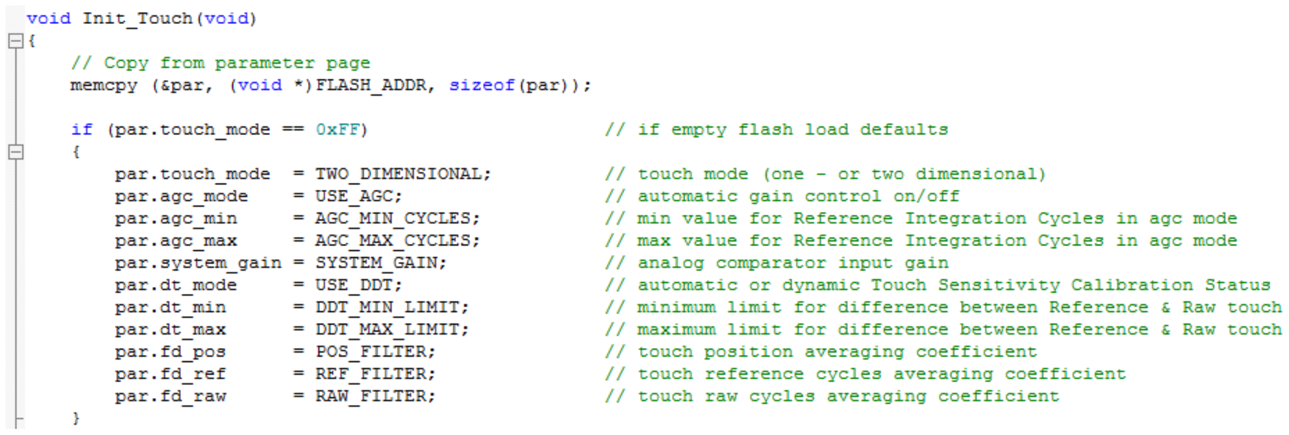
3.1.4 Add Pointer to Touch Handler

Add a pointer to the call back function to the system structure (TOUCH_SYS_T):

sys.cb_func = TouchEventHandler;

3.1.5 Initialize Touch Configuration Parameters

Initialize all the Touch configuration parameters in Touch parameters data structure (TOUCH_PAR_T). Their default values can be found in the LPC8xx_Touch_Solution header file. These parameters are stored in flash (page 511, sector 31 (last 1K)) of LPC824.



```
void Init_Touch(void)
{
    // Copy from parameter page
    memcpy (&par, (void *)FLASH_ADDR, sizeof(par));

    if (par.touch_mode == 0xFF)                     // if empty flash load defaults
    {
        par.touch_mode  = TWO_DIMENSIONAL;          // touch mode (one - or two dimensional)
        par.agc_mode    = USE_AGC;                  // automatic gain control on/off
        par.agc_min     = AGC_MIN_CYCLES;           // min value for Reference Integration Cycles in agc mode
        par.agc_max     = AGC_MAX_CYCLES;           // max value for Reference Integration Cycles in agc mode
        par.system_gain = SYSTEM_GAIN;              // analog comparator input gain
        par.dt_mode     = USE_DDT;                  // automatic or dynamic Touch Sensitivity Calibration Status
        par.dt_min      = DDT_MIN_LIMIT;            // minimum limit for difference between Reference & Raw touch
        par.dt_max      = DDT_MAX_LIMIT;            // maximum limit for difference between Reference & Raw touch
        par.fd_pos      = POS_FILTER;               // touch position averaging coefficient
        par.fd_ref      = REF_FILTER;               // touch reference cycles averaging coefficient
        par.fd_raw      = RAW_FILTER;               // touch raw cycles averaging coefficient
    }
```

Fig 5. Initialize Touch Parameters

3.1.6 Call Touch Library Initialization

Call the Touch library initialization function (passing the sensor configuration).

```
Init_Touch();
```

This brings all of the initialization to an end.

3.1.7 Call Touch Task

Finally, the user application only needs to configure a Systick timer to generate 1ms tick and periodically call the Touch_Task function to read the status of capacitive Touch sensors.

```
SysTick_Config(SystemCoreClock/1000);        // Generate interrupt each 1 ms
Touch_Task();
```

On every return from Touch_Task, the function returns a next task service time value (sys.service_timer). This time value indicates after how many milliseconds the Touch_Task function needs to be serviced (called by the user/host application) again. It is important for the user application to check for this sys.service_timer, and call Touch_Task only when this service timer has elapsed.

3.1.8 Re-Configuration or Re-Calibration

Every time the user makes changes to the system settings or Touch configuration parameters, the Touch library needs to be notified for re-configuration and re-calibration of all sensors by calling the Touch_Update function.

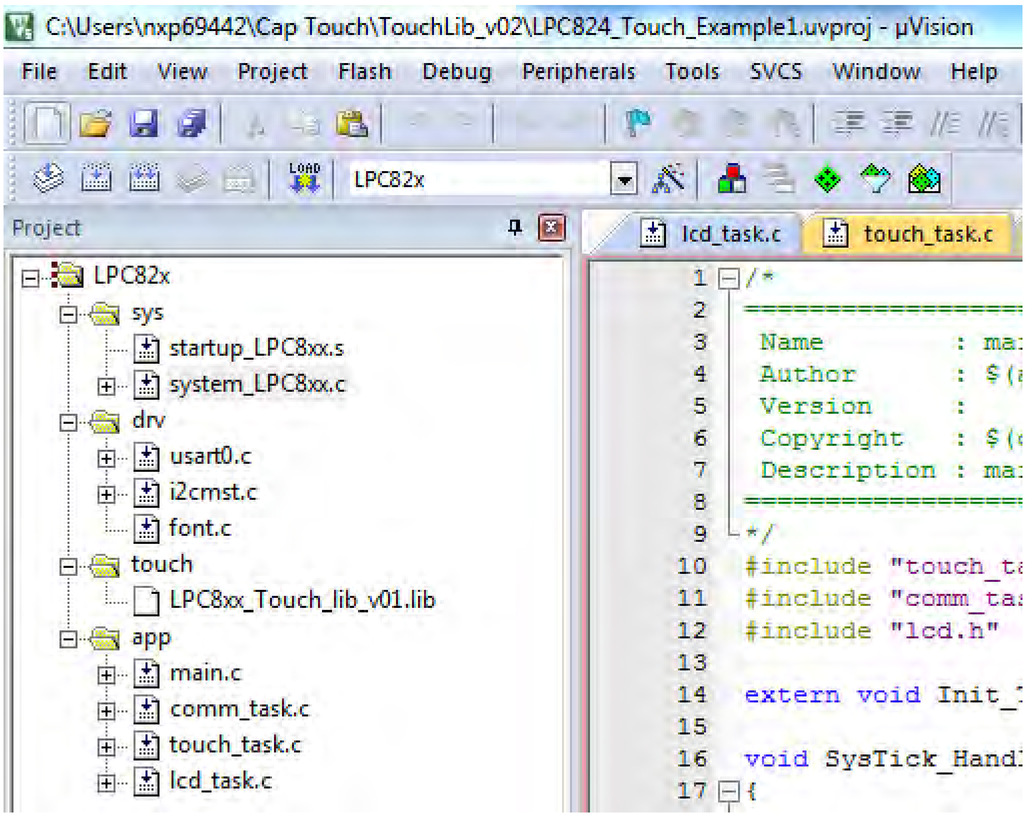
```
Touch_Update();
```

Remark: See the LPC824_Touch_Example1 for more details.

4. LPC824_Touch_Example1

The LPC82x Touch solution application note comes with an example (uVision4) project. This example code utilizes the Touch Library, which drives the on-board 3x3 Touch sensor matrix. The Touch scanning modes can be switched between one dimensional and two dimensional with a press of a switch (SW2-SP) or from Touch_GUI. The user application code displays the Touch status and/or Touch position on the on-board LCD interfaced over the I2C port, sends it over the UART to LPC11U35, which in turn sends it to the Touch_GUI running on a host PC over the USB. So, one can visualize the Touch status and/or Touch position data, and change all the Touch configuration parameters at run time from windows GUI.

To run this example on LPC82x Touch Solution (OM13081) kit, connect the kit with USB cable to a host PC and copy/paste or drag_and_drop the executable binary (LPC824_Touch_xxx.bin) from flash folder to MBED mass storage device.


C:\Users\nxp69442\Cap Touch\TouchLib_v02\LPC824_Touch_Example1.uvproj - µVision
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
LPC82x
Project
lcd_task.c
touch_task.c

```
LPC82x
  sys
    startup_LPC8xx.s
    system_LPC8xx.c
  drv
    usart0.c
    i2cmst.c
    font.c
  touch
    LPC8xx_Touch_lib_v01.lib
  app
    main.c
    comm_task.c
    touch_task.c
    lcd_task.c
```

```
/*
==================
 Name        : ma
 Author      : $(
 Version     :
 Copyright   : $(
 Description : ma
==================
*/
#include "touch_ta
#include "comm_ta
#include "lcd.h"

extern void Init_

void SysTick_Handl
{
```

Fig 6. LPC8xx_Touch_Example1

Fig-6 shows the complete project tree. All the Touch initialization, configurations and Touch event handling is dealt with in touch_task.c and touch_task.h files. Communication with Touch GUI over UART is handled in com_task.c, com_task.h, usart0.c, and usart0.h files.

This includes setting the baud rate, reading/writing Touch configuration parameters, and handling the gesture/pattern recognition training, etc. The default baud rate is set as 9600. If higher baud rates are required, simply change it in the com_task module.

The UART to USB bridge on the debug part (LPC11U35) communicates over USB to the GUI running on a host PC. The Touch status and/or position display on the on-board LCD is managed in lcd_task.c, lcd_task.h, i2cmst.c i2c.h and font.c files.

5. Touch Graphical User Interface (GUI)

The LPC82x Touch Solution comes with a windows based Graphical User Interface to visualize the Touch status and/or Touch position data, change Touch configuration parameters at run time, and explore the powerful gesture/pattern recognition feature. See Fig 7.

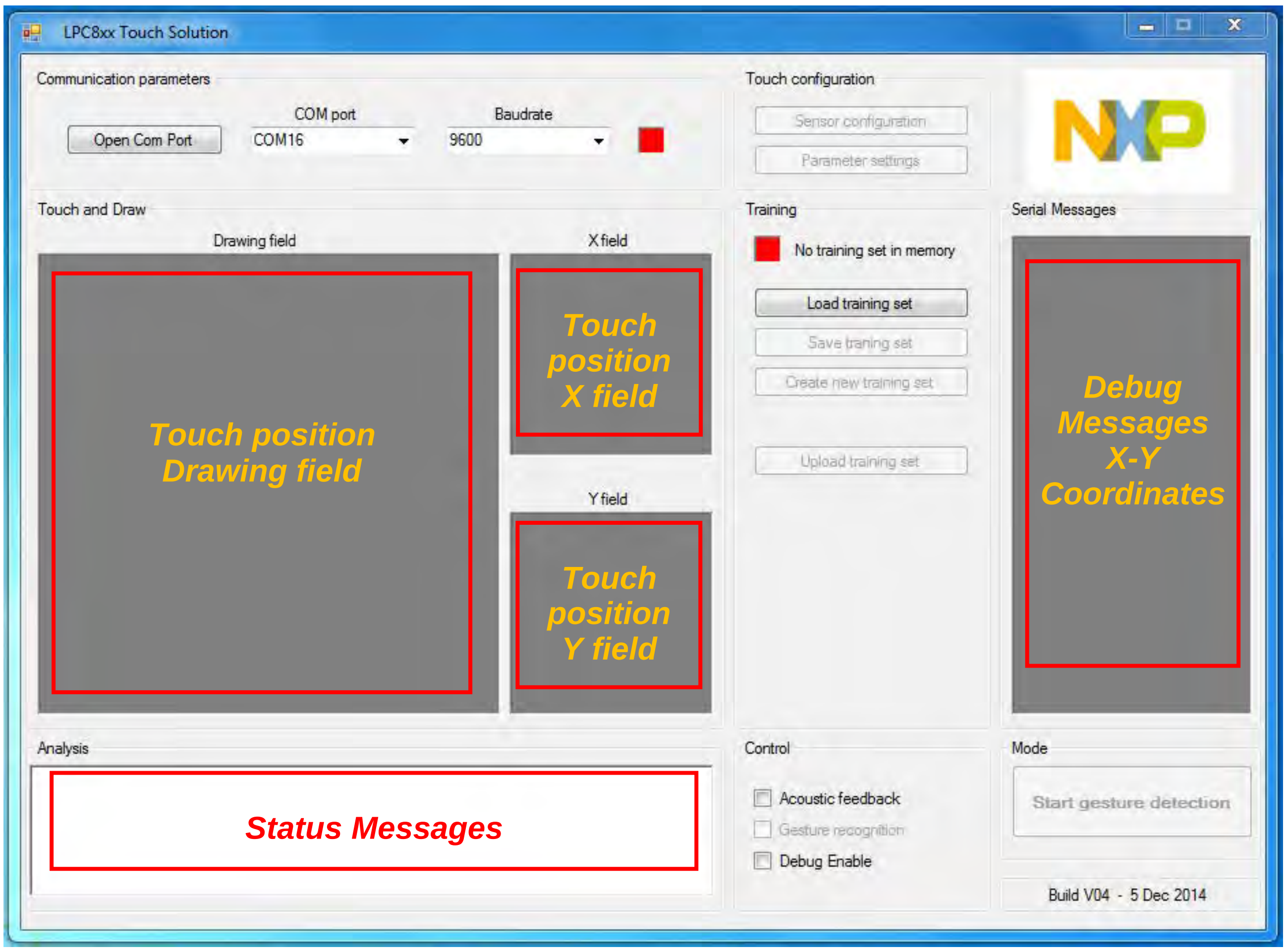


Fig 7. Touch GUI

The drawing field displays Touch status (ON/OFF) for all 9 individual sensors (S1-S9) in one dimensional mode and tracks the Touch position in two dimensional mode.

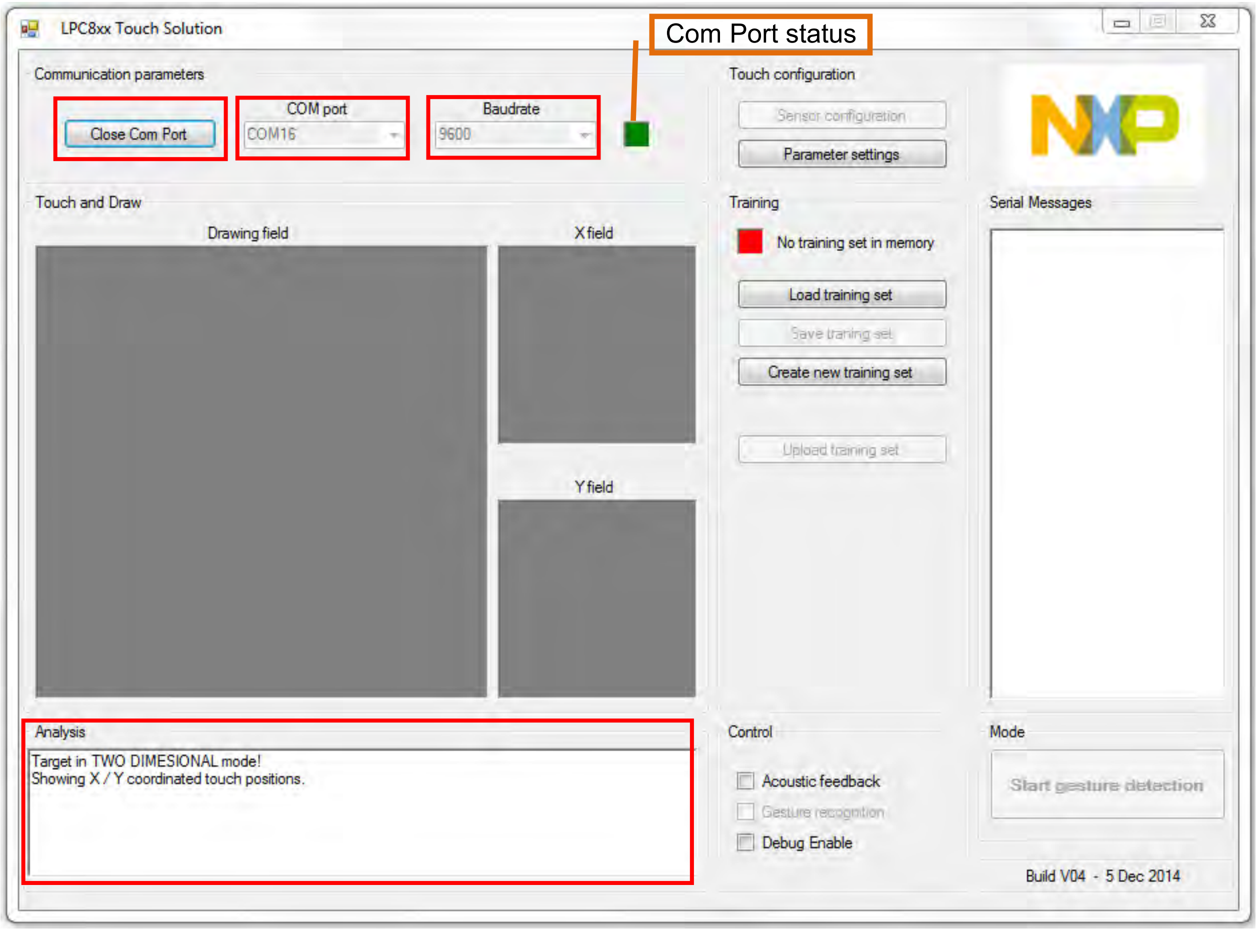
The X and Y fields to the right separately indicates the finger/Touch movements in X and Y directions. The serial messages window on the right displays the Touch data (packet ID, X and Y coordinates) received by the GUI. The analysis window at the bottom displays status and error messages. Touch configuration/Parameter settings help in handling all the Touch configuration parameters, while the control section helps to enable/disable serial debug messages, gesture recognition, and acoustic feedback during gesture recognition.

5.1.1 Launching GUI

Connect the LPC82x Touch Solution kit (already pre-programmed with an example, LPC824_Touch_EX1_v02.bin) to a host PC using micro USB cable. On powering-up, the kit should be seen as an "mbed Serial Port" in the device manager. If not, download and install the mbed windows serial (mbedWinSerial) driver available here:

www.nxp.com/redirect/mbed/handbook/Windows-serial-configuration

The Touch_GUI is now ready to be launched. Select the respective COM port, default baud rate as 9600, and open the Com port.



Target in TWO DIMESIONAL mode!
Showing X / Y coordinated touch positions.

Fig 8. Launching GUI

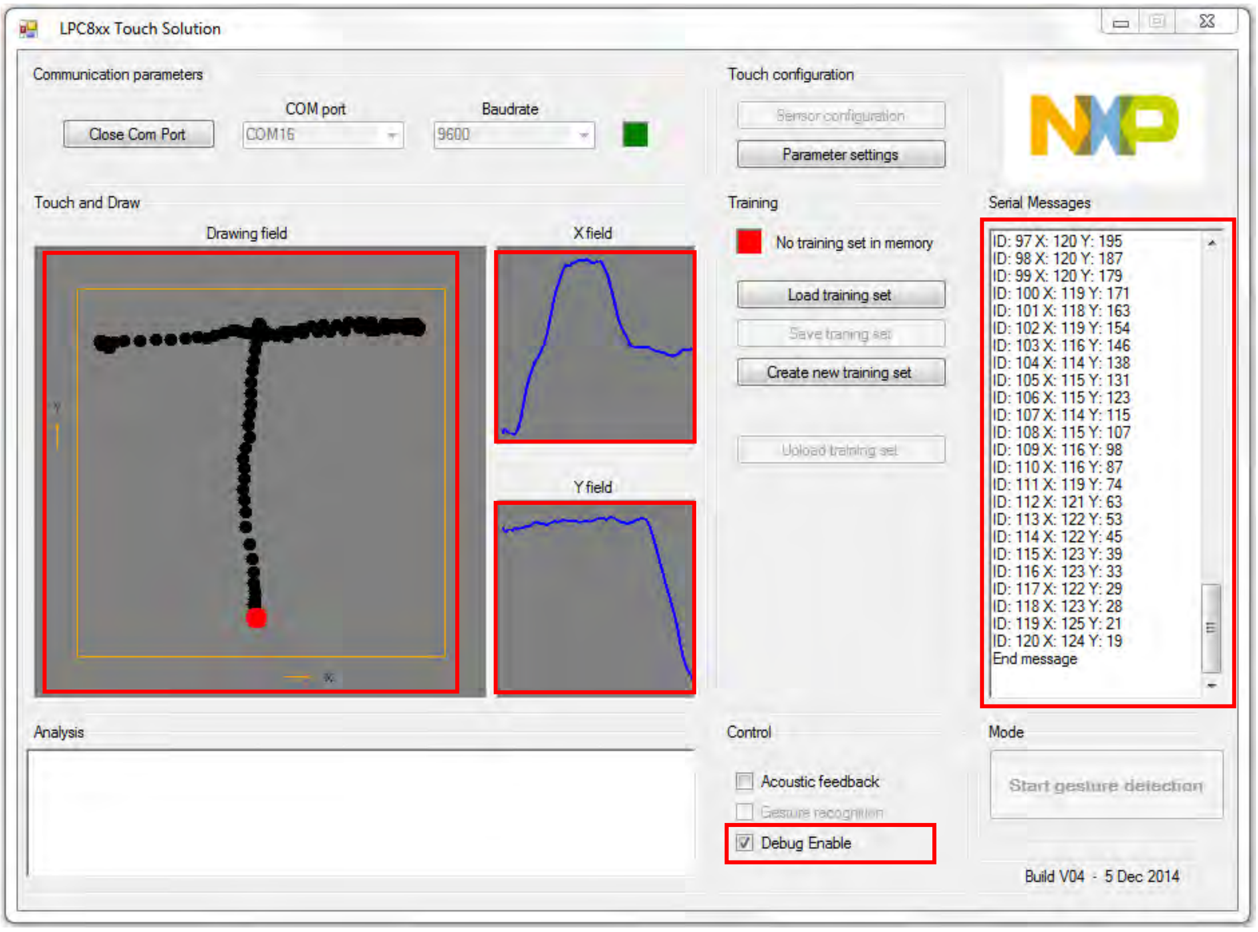
Successful com port opening is indicated by a small radio button (next to Baud rate) turning "Green" from "Red". At the same time, the analysis window displays:

Target in TWO DIMESIONAL mode!
Showing X / Y coordinated Touch positions.

The kit powers-up with two dimensional mode as the default mode.

5.1.2 Two Dimensional Mode (Touchpad)

Start touching the touchpad and Touch positions gets tracked/reflected in the "Drawing field". This mode supports only one Touch point. The X and Y fields indicates the X and Y field movements separately, and serial messages window displays the Touch data packets (ID:120 X:124 Y:19) as being received by the GUI.



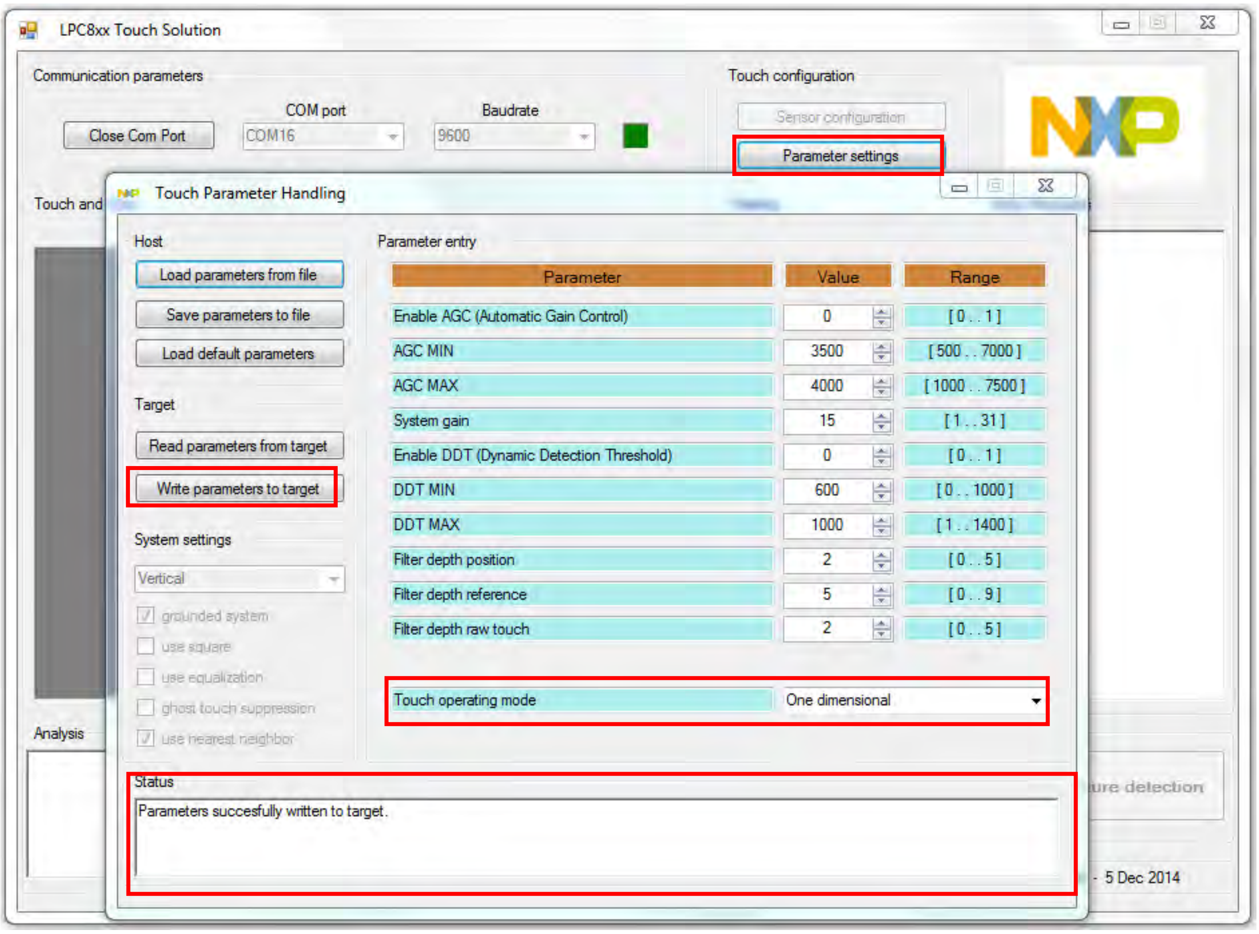
```
ID: 97 X: 120 Y: 195
ID: 98 X: 120 Y: 187
ID: 99 X: 120 Y: 179
ID: 100 X: 119 Y: 171
ID: 101 X: 118 Y: 163
ID: 102 X: 119 Y: 154
ID: 103 X: 116 Y: 146
ID: 104 X: 114 Y: 138
ID: 105 X: 115 Y: 131
ID: 106 X: 115 Y: 123
ID: 107 X: 114 Y: 115
ID: 108 X: 115 Y: 107
ID: 109 X: 116 Y: 98
ID: 110 X: 116 Y: 87
ID: 111 X: 119 Y: 74
ID: 112 X: 121 Y: 63
ID: 113 X: 122 Y: 53
ID: 114 X: 122 Y: 45
ID: 115 X: 123 Y: 39
ID: 116 X: 123 Y: 33
ID: 117 X: 122 Y: 29
ID: 118 X: 123 Y: 28
ID: 119 X: 125 Y: 21
ID: 120 X: 124 Y: 19
End message
```

Fig 9. Two Dimensional Mode

The X/Y fields will only show first MAX_POINTS (192) positions. One can tick/un-tick the "Debug Enable" to enable/disable the display of touch data packets (ID:X:Y) in serial messages window.

5.1.3 One Dimensional Mode (Keys/Buttons)

The user can switch to one dimensional mode at any time by changing the Touch operating mode in parameter settings. As soon as you click on "Parameter settings" the Touch Parameter Handling window pops up. Select the "One dimensional" mode in the drop down for Touch operating mode, and Write parameters to target. The status window displays "Parameters successfully written to target" and at the same time the on board LCD displays PARAM.


LPC8xx Touch Solution

Communication parameters

COM port: COM16
Baudrate: 9600

Close Com Port

Touch configuration

Sensor configuration
Parameter settings

Touch and

Analysis

Touch Parameter Handling

Host

Load parameters from file
Save parameters to file
Load default parameters

Target

Read parameters from target
Write parameters to target

System settings

Vertical

- [✓] grounded system
- [ ] use square
- [ ] use equalization
- [ ] ghost touch suppression
- [✓] use nearest neighbor

Parameter entry

Parameter	Value	Range
Enable AGC (Automatic Gain Control)	0	[0 .. 1]
AGC MIN	3500	[500 .. 7000]
AGC MAX	4000	[1000 .. 7500]
System gain	15	[1 .. 31]
Enable DDT (Dynamic Detection Threshold)	0	[0 .. 1]
DDT MIN	600	[0 .. 1000]
DDT MAX	1000	[1 .. 1400]
Filter depth position	2	[0 .. 5]
Filter depth reference	5	[0 .. 9]
Filter depth raw touch	2	[0 .. 5]

Touch operating mode: One dimensional

Status

Parameters succesfully written to target.

ure detection

5 Dec 2014

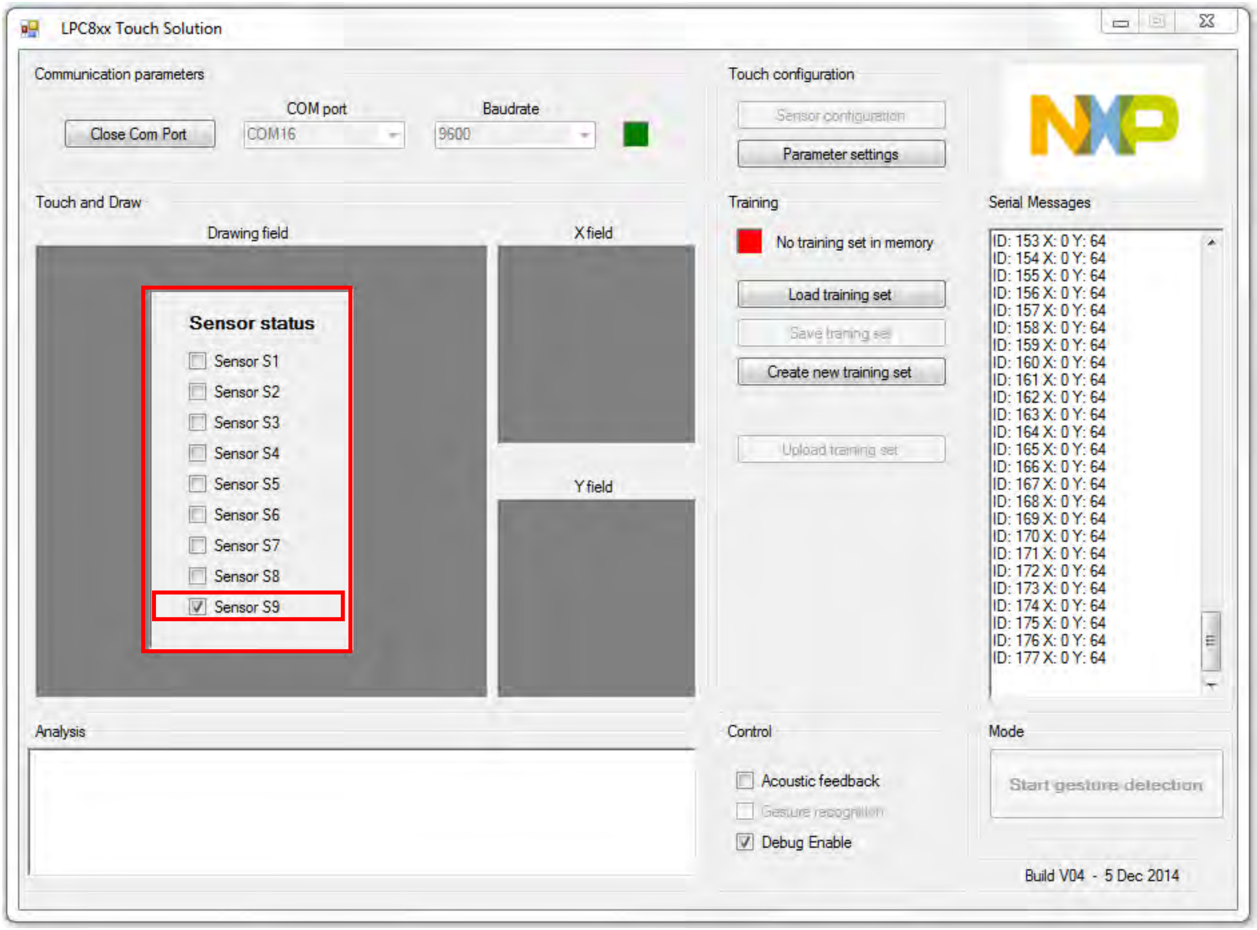
Fig 10. Touch Parameter Handling

Close the Touch Parameter Handling window after the altered parameters are successfully written to the target. Now, the individual Sensors (S1-S9) can be seen in the Drawing field and the analysis window at the bottom will display. See Fig 11.

Target in ONE DIMESIONAL mode!
Showing individual sensor (touch / no touch) status.

Touching any sensor on the board, a corresponding change (√) will be reflected in the sensor status of respective sensor. At the same time, the on board LCD will display the hexadecimal (16 bit) value representing each sensor status (bit0:sensor0, bit1:sensor1, etc.). If debug enable is set (√), serial message window will also display the appropriate hex value. This value is based on the physical connection of these sensors on the hardware.

The (√) will disappear or respective sensor status will be updated as soon as the user removes the finger from the sensor.



```
ID: 153 X: 0 Y: 64
ID: 154 X: 0 Y: 64
ID: 155 X: 0 Y: 64
ID: 156 X: 0 Y: 64
ID: 157 X: 0 Y: 64
ID: 158 X: 0 Y: 64
ID: 159 X: 0 Y: 64
ID: 160 X: 0 Y: 64
ID: 161 X: 0 Y: 64
ID: 162 X: 0 Y: 64
ID: 163 X: 0 Y: 64
ID: 164 X: 0 Y: 64
ID: 165 X: 0 Y: 64
ID: 166 X: 0 Y: 64
ID: 167 X: 0 Y: 64
ID: 168 X: 0 Y: 64
ID: 169 X: 0 Y: 64
ID: 170 X: 0 Y: 64
ID: 171 X: 0 Y: 64
ID: 172 X: 0 Y: 64
ID: 173 X: 0 Y: 64
ID: 174 X: 0 Y: 64
ID: 175 X: 0 Y: 64
ID: 176 X: 0 Y: 64
ID: 177 X: 0 Y: 64
```

Fig 11. One Dimensional Mode

The user can always switch back to the two dimensional mode at any time by changing the Touch operating mode in parameter settings again. The status window displays “Parameters successfully written to target” and at the same time the on board LCD displays PARAM. On closing the Touch Parameter Handling, the analysis window will now display:

Target in TWO DIMESIONAL mode!
Showing X / Y coordinated touch positions.

5.1.4 Gesture Recognition Training

Gesture recognition is a unique feature of NXP's LPC82x Touch Solution. With this feature, one can provide training to the Touch solution with specific gestures/patterns, and then it recognizes those gestures/patterns when re-drawn.

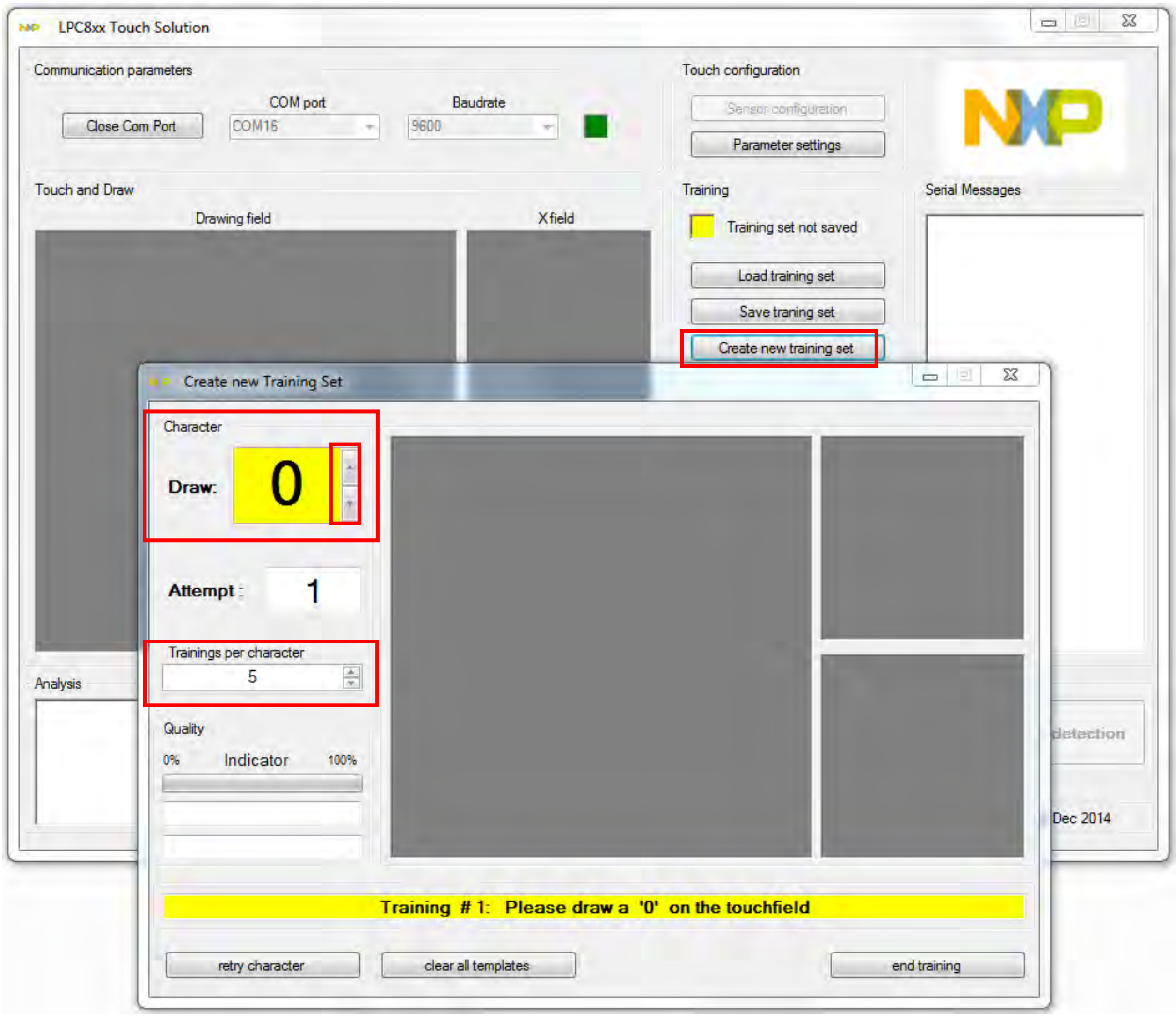


Fig 12. Create new training set

To create a training set, click on "Create new training set" and a new window pops up. See Fig-12. The user can select the number of trainings per character between 1 and 99 and start drawing the character or number (on touchpad) as prompted to draw. By default, it starts with "0" and goes on till "9". Otherwise, one can also select a particular character or number they would like to train by clicking on the drop down arrows next to the prompted character to be drawn on top left corner. The quality indicator will show the uniqueness or variation of the character or number drawn. See Fig-13.

The attempt window shows the current attempt number (out of selected total trainings per character) for that number, and goes on reducing. As soon as the attempts get over for that number, the character or number gets advanced to the next number. At any point in time, the user can also select a particular character or number they would like to train by clicking on the drop down arrows next to prompted character to be drawn on top left corner.

The user can retry the current character or number (all attempts) or clear all previous templates. One can create a training set covering the entire set of characters or numbers from 0 to 9, or can stop at any number in between and end of the training session.

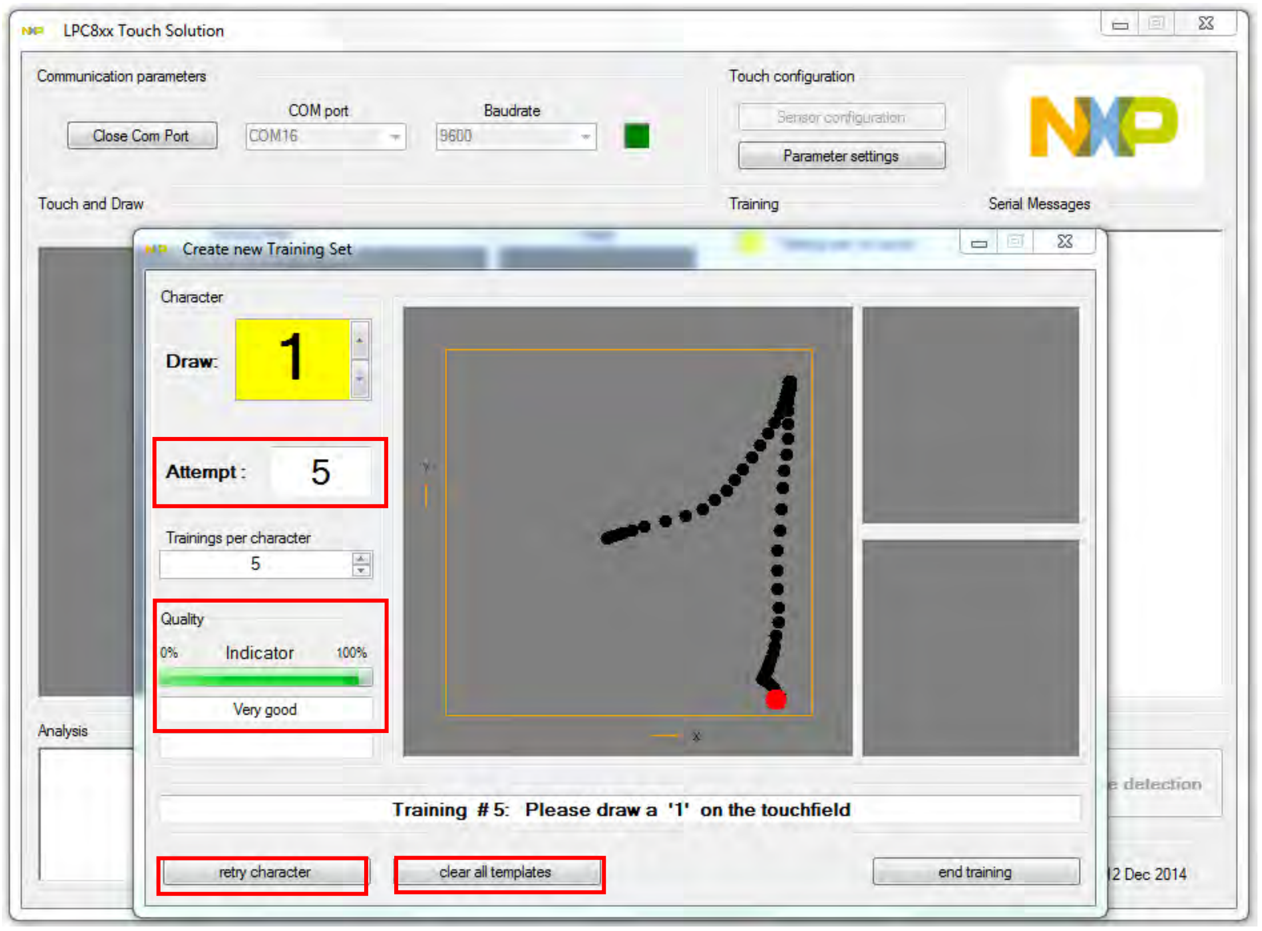


Fig 13. Training a Character/Number

On clicking the “end training”, a uniqueness message pops up stating “Uniqueness check passed!” or “Uniqueness check failed!” indicating the uniqueness of the characters or numbers that are drawn. See Fig-14.

Remark: The “Uniqueness check failed!” message does not require any action from the user.

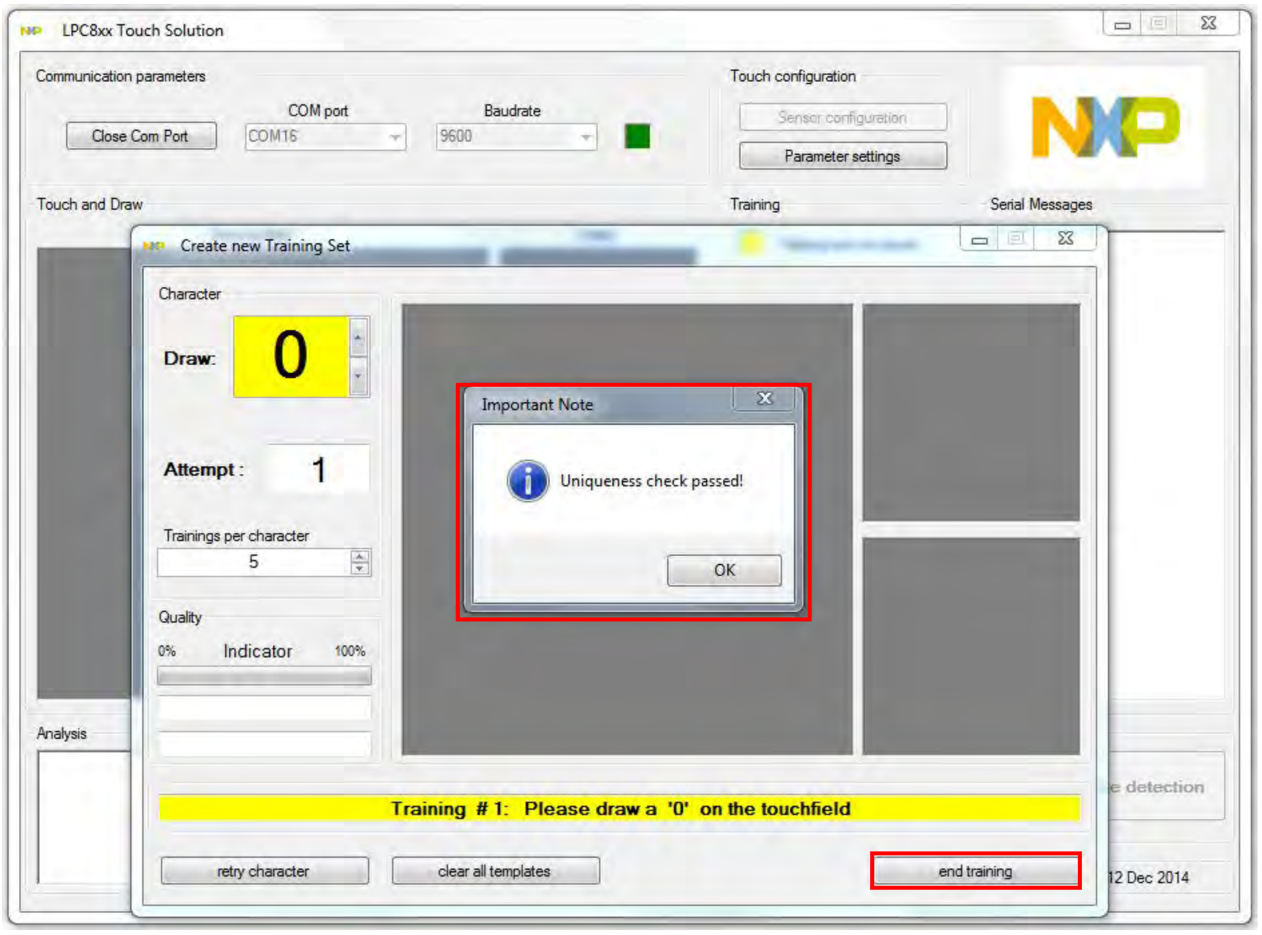
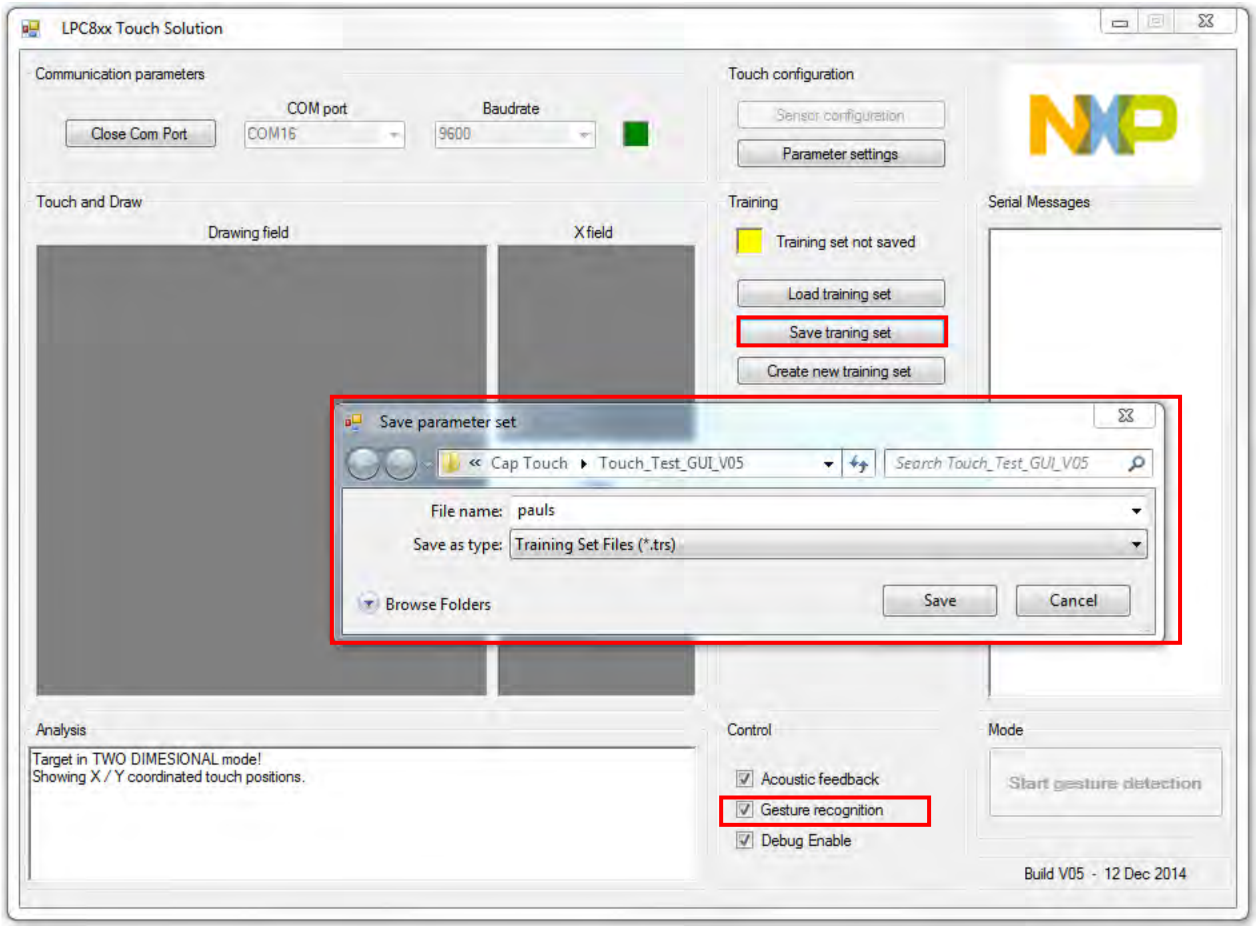


Fig 14. End Training

Once the training has ended, the training window gets closed and ‘Save training set’ and ‘Gesture recognition’ radio buttons gets activated on the main window, as seen in Fig-15. You can save the created training set as a .trs file to the host PC, if you would like to use it in the future. For the same click on ‘Save training set’, and name the file (say, pauls.trs) and save it to the directory/folder of your choice. Now, you are ready to explore the gesture recognition feature.



File name: pauls

Save as type: Training Set Files (*.trs)

Target in TWO DIMESIONAL mode!
Showing X / Y coordinated touch positions.

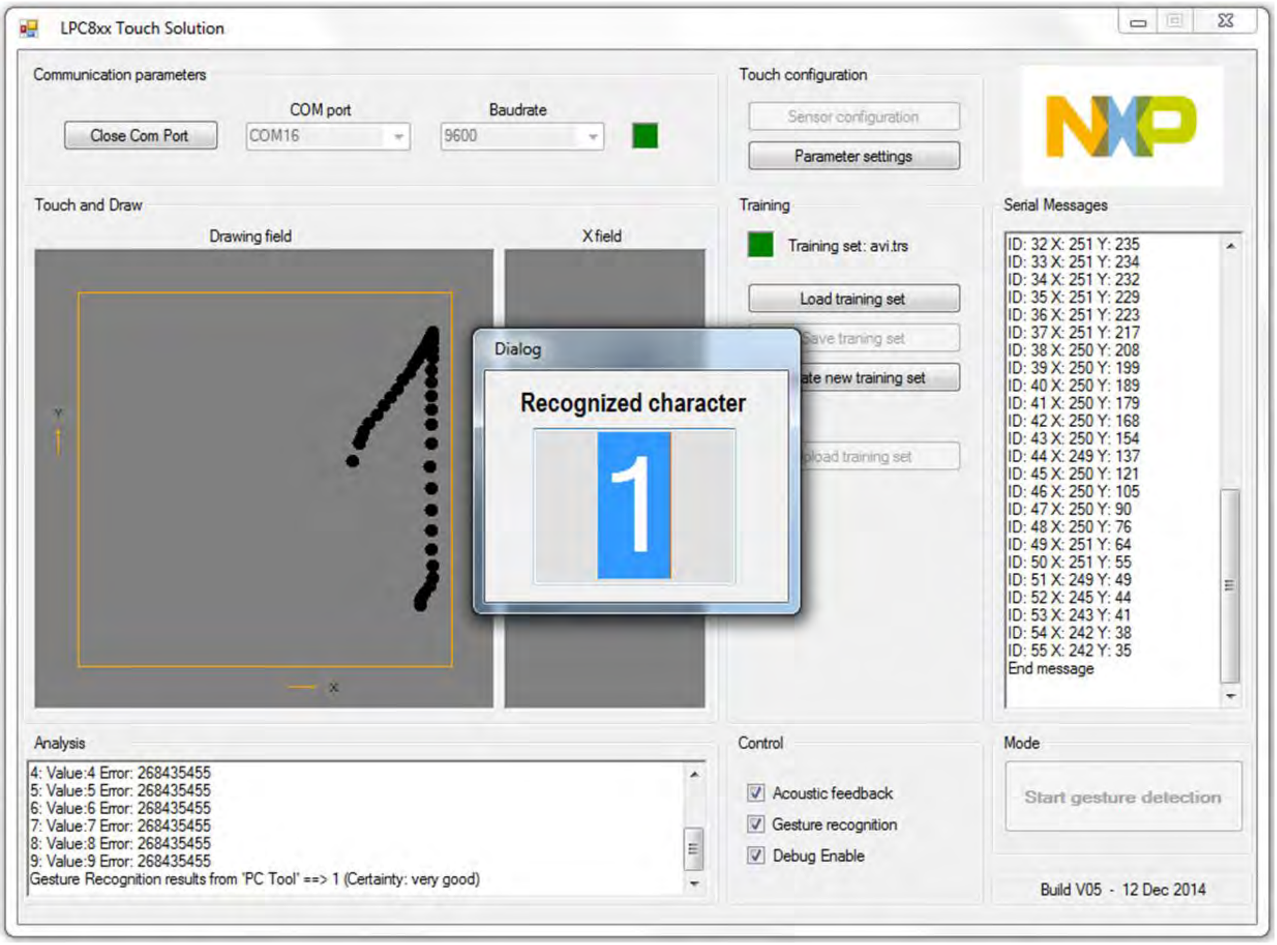
☑ Acoustic feedback
☑ Gesture recognition
☑ Debug Enable

Fig 15. Save training set

5.1.5 Gesture Recognition

Now, you can start using the gesture recognition feature with the training set just created Enable (√) the 'Gesture recognition' (in Control section) to turn ON gesture recognition feature and have fun with it! Draw any character/number your training set covers, and it will recognize the same, as seen in Fig-16. At the same time, the analysis window at the bottom will display:

Gesture Recognition results from 'PC Tool' ==> 1 (Certainty: perfect or very good)



```
ID: 32 X: 251 Y: 235
ID: 33 X: 251 Y: 234
ID: 34 X: 251 Y: 232
ID: 35 X: 251 Y: 229
ID: 36 X: 251 Y: 223
ID: 37 X: 251 Y: 217
ID: 38 X: 250 Y: 208
ID: 39 X: 250 Y: 199
ID: 40 X: 250 Y: 189
ID: 41 X: 250 Y: 179
ID: 42 X: 250 Y: 168
ID: 43 X: 250 Y: 154
ID: 44 X: 249 Y: 137
ID: 45 X: 250 Y: 121
ID: 46 X: 250 Y: 105
ID: 47 X: 250 Y: 90
ID: 48 X: 250 Y: 76
ID: 49 X: 251 Y: 64
ID: 50 X: 251 Y: 55
ID: 51 X: 249 Y: 49
ID: 52 X: 245 Y: 44
ID: 53 X: 243 Y: 41
ID: 54 X: 242 Y: 38
ID: 55 X: 242 Y: 35
End message
```

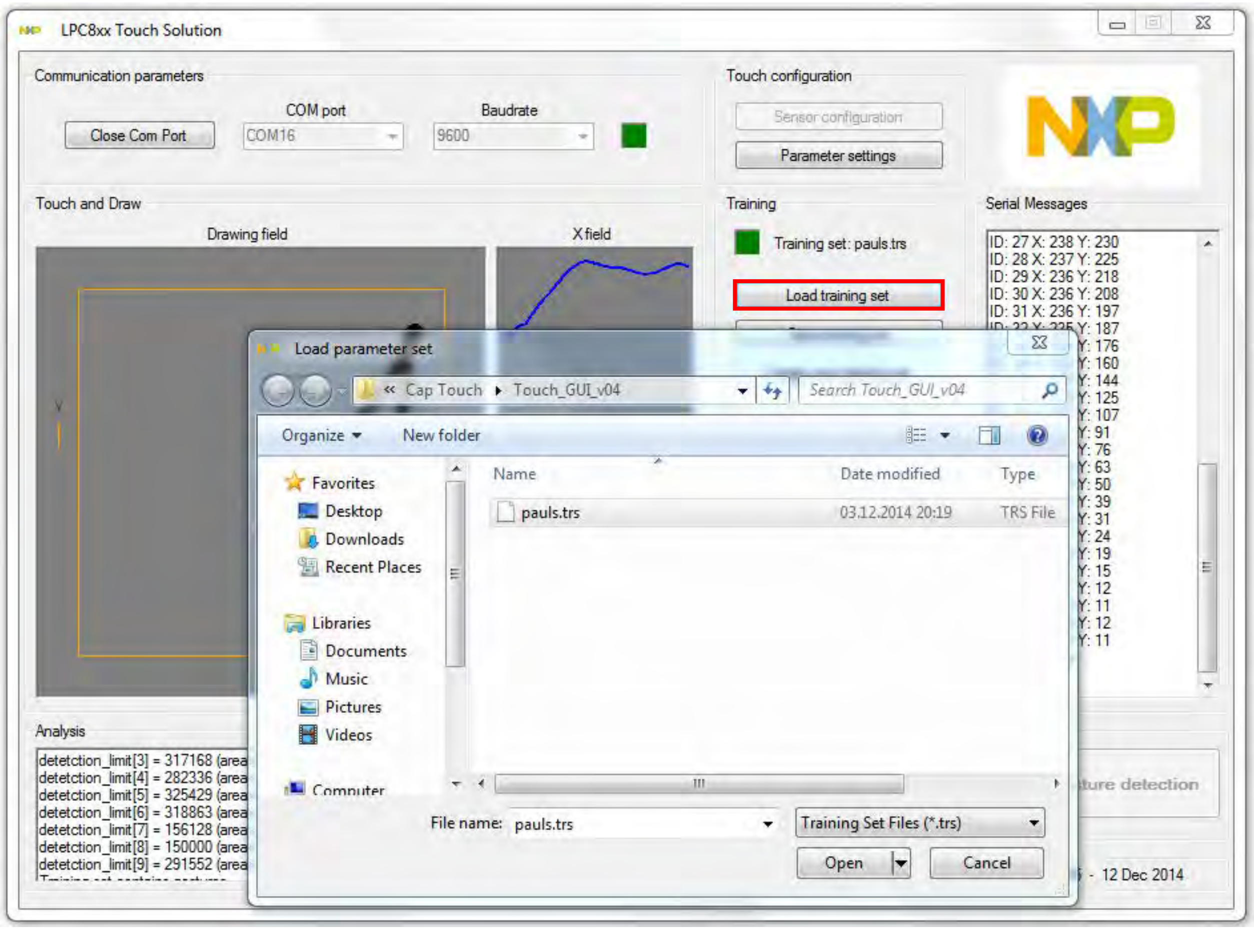
```
4: Value:4 Error: 268435455
5: Value:5 Error: 268435455
6: Value:6 Error: 268435455
7: Value:7 Error: 268435455
8: Value:8 Error: 268435455
9: Value:9 Error: 268435455
Gesture Recognition results from 'PC Tool' ==> 1 (Certainty: very good)
```

☑ Acoustic feedback
☑ Gesture recognition
☑ Debug Enable

Fig 16. Gesture Recognition

The recognition will fail if the respective character/number is not a part of the training set loaded or drawn differently than the training set. Whenever a character/number is not recognized, the recognized character dialog will display a "?". If the Acoustic feedback (in Control section) is enabled, you will also hear audio beep when a gesture/pattern is recognized or even if it's not recognized.

You can also load and use any training set (pauls.trs) previously created and stored on the host PC. For the same, use the ‘Load training set’ button in the training section. You just need to select the training set and open it.



COM port: COM16
Baudrate: 9600
Training set: pauls.trs

```
ID: 27 X: 238 Y: 230
ID: 28 X: 237 Y: 225
ID: 29 X: 236 Y: 218
ID: 30 X: 236 Y: 208
ID: 31 X: 236 Y: 197
ID: 32 X: 235 Y: 187
Y: 176
Y: 160
Y: 144
Y: 125
Y: 107
Y: 91
Y: 76
Y: 63
Y: 50
Y: 39
Y: 31
Y: 24
Y: 19
Y: 15
Y: 12
Y: 11
Y: 12
Y: 11
```

```
detetction_limit[3] = 317168 (area
detetction_limit[4] = 282336 (area
detetction_limit[5] = 325429 (area
detetction_limit[6] = 318863 (area
detetction_limit[7] = 156128 (area
detetction_limit[8] = 150000 (area
detetction_limit[9] = 291552 (area
```

Name	Date modified	Type
pauls.trs	03.12.2014 20:19	TRS File

File name: pauls.trs
Training Set Files (*.trs)

Fig 17. Load training set

Once the training set is loaded, Gesture recognition (in Control section) gets activated and analysis window displays:

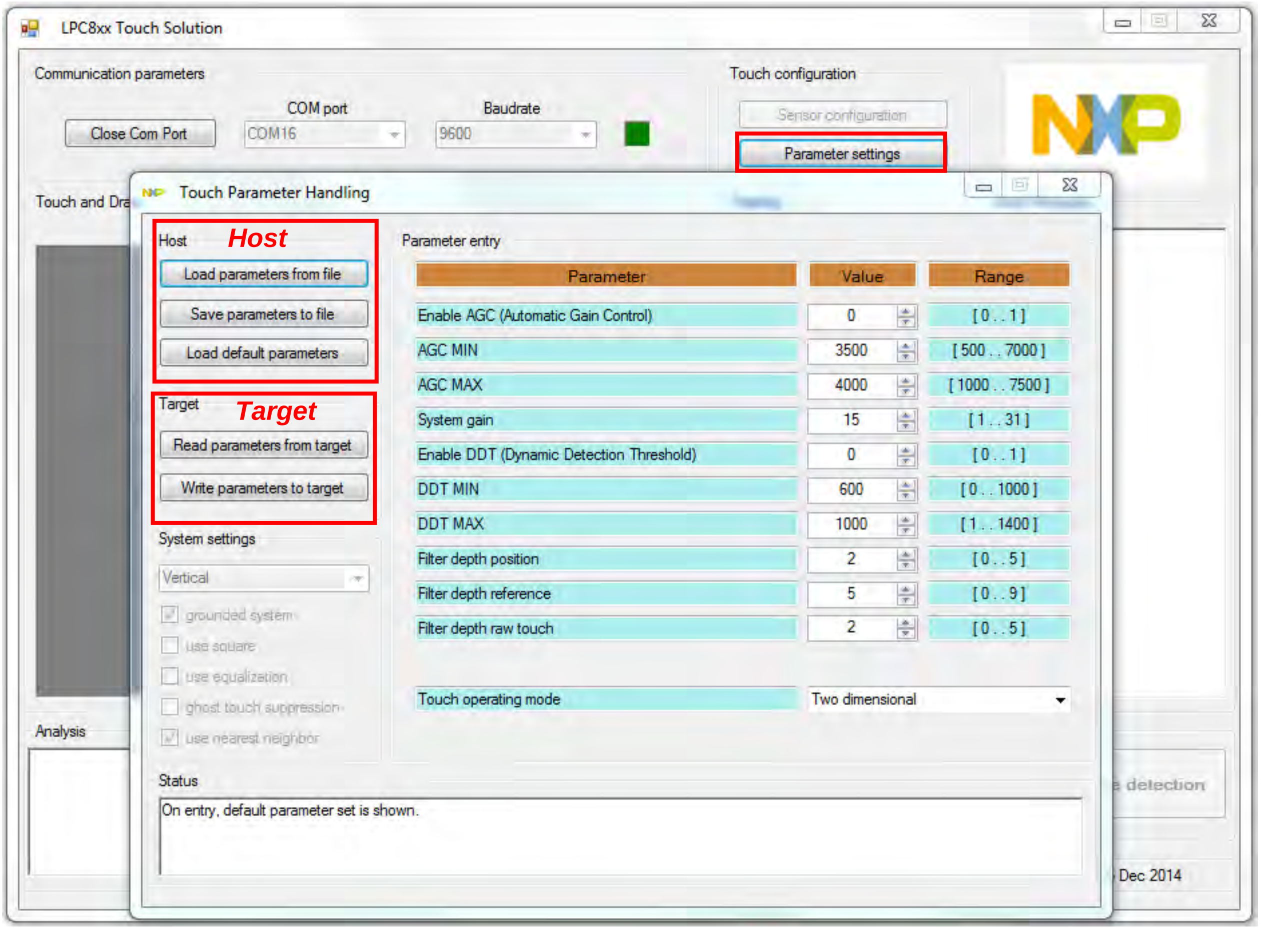
> Training set successfully loaded from file: C:\ \ \Cap Touch\Touch_GUI_v04\Johns.trs

And at the same time the on board LCD displays PARAM as indication of the change. The training set currently loaded and in use can be seen just above the ‘Load training set’ radio button in training section.

5.1.6 Touch Configuration/Parameter Handling

The Touch configuration parameters can be changed at run time using the Touch GUI. Click on "Parameter settings" (Fig-18) to see and alter the Touch configuration parameters, default set of parameters is shown on the entry. One can read the parameters from target hardware or write the altered parameters to the target hardware using the radio buttons in Target section.

It's always a good practice to read the existing parameters form the target and then alter any specific parameter you would like to change. Once the parameter is changed, you need to click on "Write parameters to target" to make the change effective.



COM port: COM16
Baudrate: 9600
System settings: Vertical

Parameter	Value	Range
Enable AGC (Automatic Gain Control)	0	[0 . . 1]
AGC MIN	3500	[500 . . 7000]
AGC MAX	4000	[1000 . . 7500]
System gain	15	[1 . . 31]
Enable DDT (Dynamic Detection Threshold)	0	[0 . . 1]
DDT MIN	600	[0 . . 1000]
DDT MAX	1000	[1 . . 1400]
Filter depth position	2	[0 . . 5]
Filter depth reference	5	[0 . . 9]
Filter depth raw touch	2	[0 . . 5]

Touch operating mode: Two dimensional

On entry, default parameter set is shown.

Fig 18. Touch Parameter settings

There's also a provision to save the current parameters to the host/PC or load the pre-set set of parameters from the host/PC in order to write to the target, see radio buttons in Host section. While changing the parameters (value) settings, the range shown on the right is your guideline. Even if you set out of range values, it will be truncated to the maximum value. Any point in time, one can always fall back and load the default set of parameters to cross verify the settings. Refer the Table-7 below for the details on these parameters.

Table 1. TOUCH_PARAMETERS

Parameter	Value	Description
Touch operating mode	One dimensional / Two dimensional	Touch operating mode (one - or two dimensional). One dimensional Mode: Keys/Buttons (Touch or no Touch) Two dimensional Mode: Touchpad/Gesture (X-Y position)
Enable AGC (Automatic Gain Control)	0 / 1	Automatic Gain Control feature for adjusting the sensitivity as desired. It enables auto-setting independent gain for each sensor such that they all have uniform sensitivity level as a group. The Touch Solution library internally varies gain of individual sensors to get a uniform behavior for the complete matrix. The gain values are set such that there is a considerable difference between Touch and no-Touch state of sensors. ON (1): AGC feature enabled. It's recommended for one dimensional (Keys/Buttons) mode, when reasonable sensitivity couldn't be achieved only by fixing the overall system_gain value. OFF (0): AGC feature disabled, the common system_gain value loaded automatically for all the sensors. It's recommended to disable it for two dimensional mode.
AGC MIN	500 ~ 7000	Only when Automatic Gain Control feature is enabled, the AGC minimum value is applicable. AGC MIN is the minimum gain value setting for any sensor to get reasonable sensitivity between touched and not-touched state. It's range is limited from 500 to 7000 based on manual calibration experiments.
AGC MAX	1000 ~ 7500	Only when Automatic Gain Control feature is enabled, the AGC maximum value is applicable. AGC Max is the maximum gain value setting for any sensor to get reasonable sensitivity between touched and not-touched state. Please note that by increasing the gain, the sensitivity can be increased, but at the cost of speed. So developers have to strike a balance between speed and sensitivity based on the application requirement. It's range is limited from 1000 to 7500 based on manual calibration experiments.
System gain (only used if AGC is off)	1 ~ 31	The System gain is the common overall gain value for all sensors. It's applicable only when the AGC mode is disabled. It's range is limited from 0 to 31 based on manual calibration experiments.
Enable DDT (Dynamic Detection Threshold)	0 / 1	Dynamic Detection Threshold feature for changing the threshold value dynamically as needed sometimes while dealing with the harsh conditions arising due to the presence of moisture/water, dust, grease/fat, temperature variations, stray electric field, etc. ON (1): Dynamic Detection Threshold feature enabled. The detection threshold value will be dynamically changed from DT_MIN to DT_MAX to achieve the reasonable sensitivity at all times in harsh or varying conditions. The Touch Solution library will automatically vary the detection threshold based on the sensitivity level at that point in time. OFF (0): Dynamic Detection Threshold feature disabled. The DT_MIN value will act as the Detection Threshold setting.
DDT MIN	0 ~ 1000	DT MIN is the minimum value of Detection Threshold range. The detection threshold cannot have a large range of values, otherwise the performance will have an adverse effect.
DDT MAX	1 ~ 1400	Maximum value of Detection Threshold range. Please note that setting the DT min or max to lower values increases the sensitivity of the system.
Filter depth raw touch	0 ~ 5	It's the IIR filter tap size for instantaneous signal (number of cycles

Parameter	Value	Description
		necessary to charge the storage cap to sys gain level). Filtering reduces the jitter in the calculated position and hence these cycles are averaged before position calculation to get stable values. Based on comparison with reference, they can be classified as Touch cycles in case of Touch event or update to reference cycles in case of no-Touch event. In order to avoid transient noises, these cycles are also averaged. An IIR filter is realized in SW and filter depth refers to the number of taps. Typical value of 2 denotes the tap size of 2 to the power 2.
Filter depth reference	0 ~ 9	It's the IIR filter tap size for reference signal, number of cycles necessary to charge the storage cap to system gain level under untouched conditions. In order to avoid transient noises, these cycles are as well averaged using IIR filter realized in SW and filter depth refers to the number of taps. Typical value of 5 denotes the tap size of 2 to the power 5.
Filter depth position	0 ~ 5	It's the Moving Average filter tap size for Touch position data. Touch position value calculated is not always stable due to non-uniform sensitivities of individual sensors or because of noise or even interference. This results in a small jitter in the calculated position value. To reduce this jitter, a Moving average filter is realized in the SW. The filter depth refers to the number of taps in the moving average filter. Maximum value of 3 denotes the tap size of 2 to the power 3.

6. Legal information

6.1 Definitions

6.2 Disclaimers

6.3 Trademarks

7. Contents

1. Introduction 3
2. Overview 3
2.1 Microcontroller Resources 4
2.2 Performance indicators & Constraints 4
3. User Application 5
3.1.1 Include Touch Library 5
3.1.2 Define and Initialize Sensors 5
3.1.3 Define Touch Handler 6
3.1.4 Add Pointer to Touch Handler 6
3.1.5 Initialize Touch Configuration Parameters 6
3.1.6 Call Touch Library Initialization 7
3.1.7 Call Touch Task 7
3.1.8 Re-Configuration or Re-Calibration 7
4. LPC824_Touch_Example1 7
5. Touch Graphical User Interface (GUI) 9
5.1.1 Launching GUI 10
5.1.2 Two Dimensional Mode (Touchpad) 11
5.1.3 One Dimensional Mode (Keys/Buttons) 11
5.1.4 Gesture Recognition Training 14
5.1.5 Gesture Recognition 18
5.1.6 Touch Configuration/Parameter Handling 20
6. Legal information 23
6.1 Definitions 23
6.2 Disclaimers 23
6.3 Trademarks 23
7. Contents 24

Date of release: 22 December 2014
Document identifier: AN11620