AN11666

Gesture Recognition Application Note

Rev. 1.0 — 09 June 2015 Application Note

Document information

Info	Content
Keywords	LPC82x Touch Solution, Gesture Recognition, Touch Analysis, User Guide, Library, Capacitive Touch, Touchpad, Sensor, Drive / Sensing lines, Touch.
Abstract	This document describes how to use NXP's Touch Solution Gesture Recognition firmware library. It contains examples and a detailed description of an application that makes use of the Touch and Gesture Recognition firmware libraries.

Revision history

Rev	Date	Description
1.0	20150609	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 Gesture Recognition application can be used as an additional application layer on top of NXP's LPC82x Touch Solution. This solution is centered around the LPC824, ARM Cortex-M0+ MCU running up to 30MHz with 32 KB internal flash, and 8 KB SRAM. The solution consists of a complete hardware and software platform for building a touch user interface design. It is based on a royalty free "LPC8xx Touch Library" and the OM13081 demo kit. The touch library drives and scans the touch sensors, processes data for robust and noise-free touch detection, and provides the touch data to user's application.

The touch data is provided to the Gesture Recognition library as a series of touched positions on the touch pad. It then compares the input data with the pre-loaded reference training data for a 'gesture' match / no match. This way it is able to recognize a number of predefined (trained) drawing patterns (gestures or hand writing) created with the touch solution library code.

This application note describes the Gesture Recognition library usage in a user (host) application and the Graphical User Interface tool to create and upload the reference training data.

2. Overview

Fig 1 illustrates the overall gesture recognition application concept. To use gesture recognition, the Touch Solution firmware (light blue block in Fig 1) must be included as part of the user application and configured for two dimensional mode.

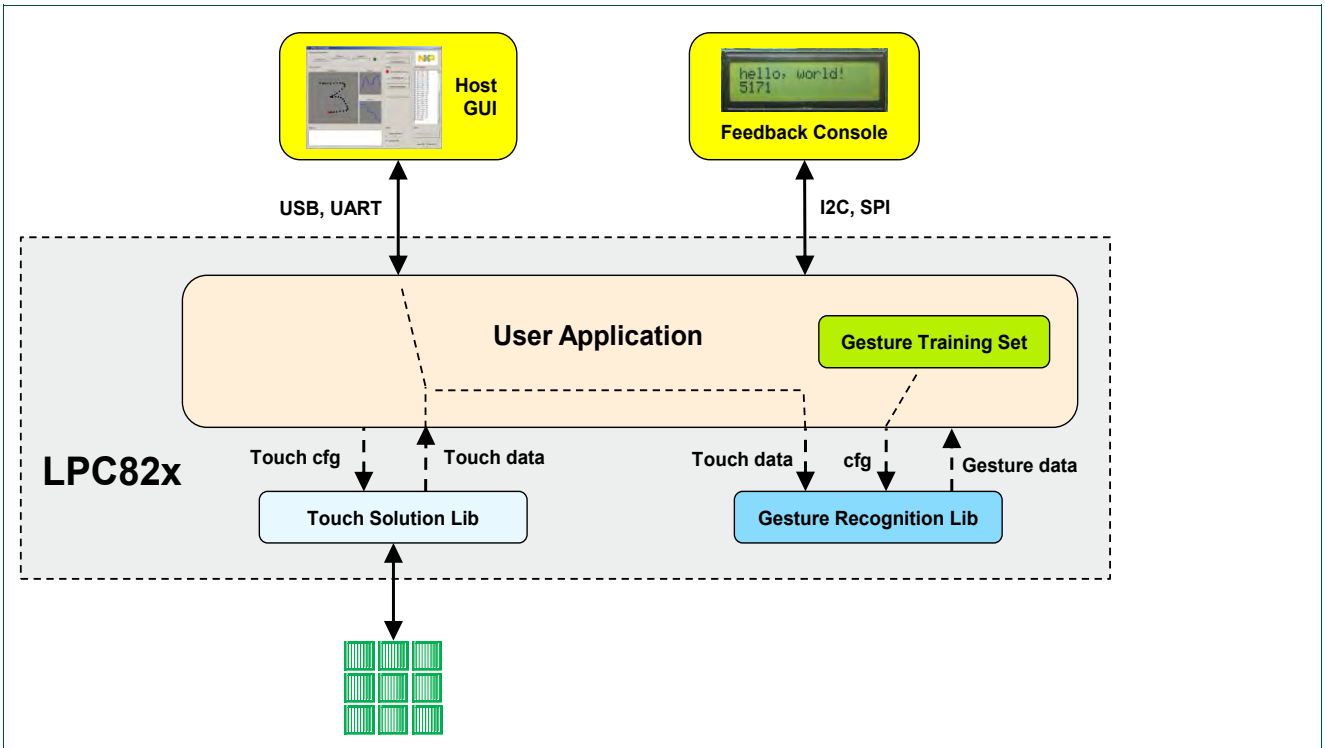

Fig 1. Gesture Recognition application

The two dimensional mode transforms the capacitive sensor layout into a touch area and touch / no touch data is provided as a two dimensional touch position. For more information regarding the LPC82x Touch Solution library see:

LPC82x Touch Solution | www.LPCware.com

The touch positions (x / y coordinates) are reported in 8-bit resolution (0-255 positions) and are subsequently passed to the Gesture Recognition library (see Fig 1). The library compares the input data with the pre-loaded reference training data to detect a drawn 'gesture' match / no match. Gesture recognition is composed of two main steps:

- Gesture training (creating the gesture reference data).
- Gesture capturing and recognition.

3. Gesture training

The first step in implementing the gesture recognition functionality is **training**. Drawn patterns for every gesture are recorded and stored into a "training set". A training set can be created with the help of a GUI (PC windows application) based NXP gesture creation and recognition application.

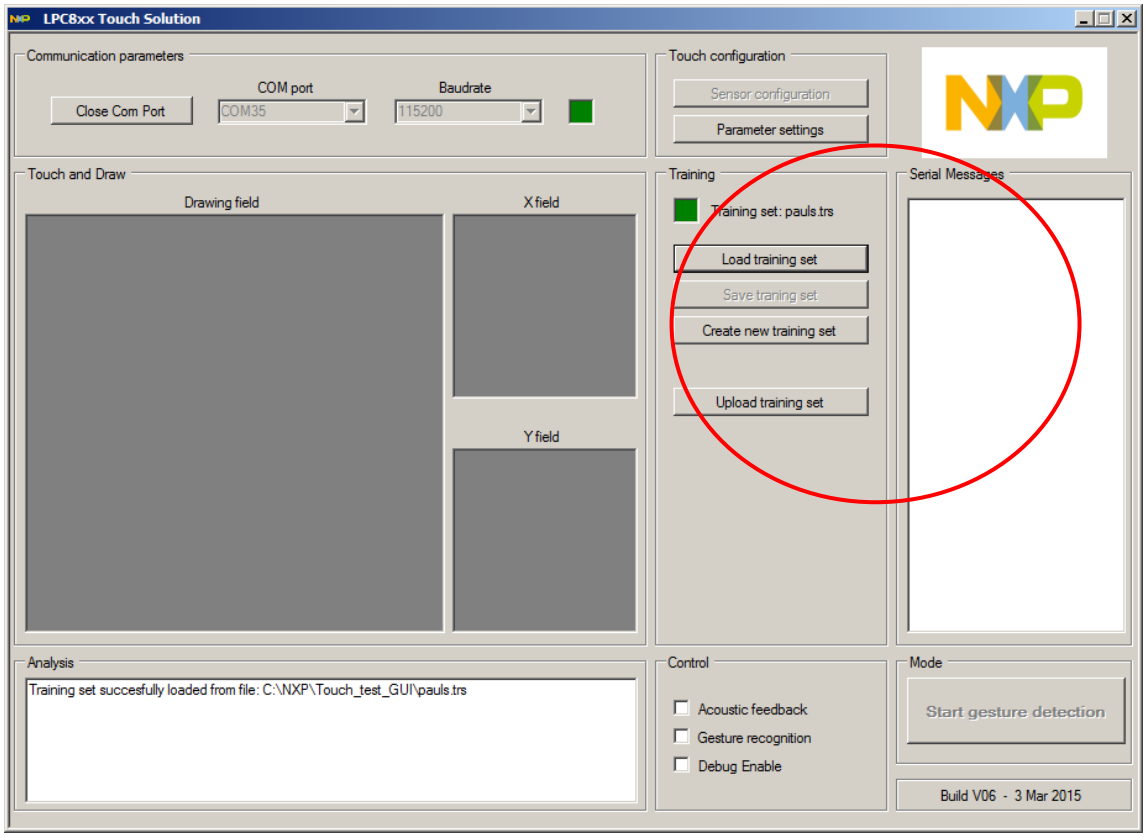

Fig 2. NXP Touch solution GUI

3.1 Create training set

A new training set can be created by clicking the “Create new training set” button. This will open a new application window, see Fig 3.

In the ‘Gesture’ panel / box on the left side, the current trained gesture can be selected (0 to 9 possible gestures). During the training every gesture is drawn and recorded repeatedly. The number of repeats is indicated by the “Training nr” textbox.

The “number of trainings per gesture” indicates how many times a specific gesture must be trained, which is user configurable up to 99 (note that a higher number of trainings per gesture achieves better recognition results).

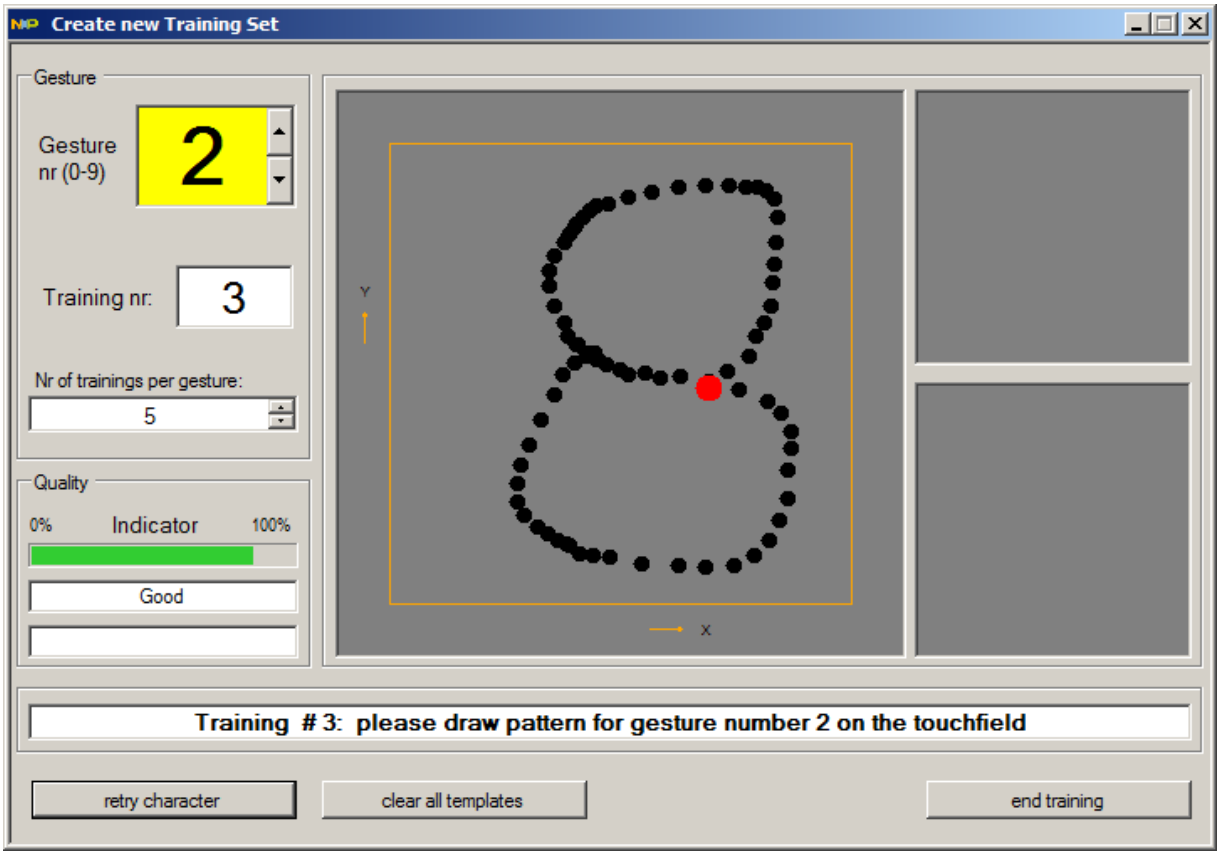

Fig 3. Gesture training application

After every drawn gesture the GUI application calculates and stores the maximum and minimum value (of X and Y coordinates) of every touch position coordinate (that is, coordinate range for every sample). Next, a short audio beep is played and a quality indication is calculated and displayed, see Fig 4.

The quality indicator indicates the best compromise between FAR (False Acceptance Rate) and FRR (False Rejection Rate). It is a relation between the gesture area and full area of ~10%. A lower value sets the limits very close together and it is hard to repeat the exact gesture, resulting in a higher FRR. If the error limits are too wide, nearly every drawing will be recognized as a gesture.

A drawing pattern for a gesture can be arbitrary and does not need to match the gesture exactly in size or shape. If a complete gesture is trained, three short audio beeps are played and the reference pattern considering deviations (within predetermined tolerance range) of each iteration is calculated. Multiple iterations while training a specific gesture helps in taking into consideration all the user variations or dynamics (for example, speed and size). The system is now ready to select the next gesture to be trained.

Fig 4. Left: drawn gesture quality indicator. Right: training set uniqueness check.

At any time, one can retry training a particular gesture or character by clicking on “retry character”, or clear the entire training set by clicking on “clear all templates”.

Finally, when the creation of a training set is ended, a uniqueness (distinctive gestures) check is performed. It checks if the gestures are not too similar (that is, ‘1’ and ‘I’). The test uses a Monte Carlo algorithm (=random points inside a gesture limit, which are compared to all other gestures for distinctiveness).

Note:

Clear all the previous gestures in memory before starting the creation of a new training set, otherwise the uniqueness check might fail. By clicking OK on the uniqueness check pop-up, the GUI returns to its main application form (indicating unsaved training set available in memory) as shown in Fig 5.

This training set is available as long as the current touch GUI is still open. One must create a new training set or load previously saved training set every time the touch GUI is opened. Currently, the maximum number of gestures a training set can contain is 10 (numbered 0 – 9).

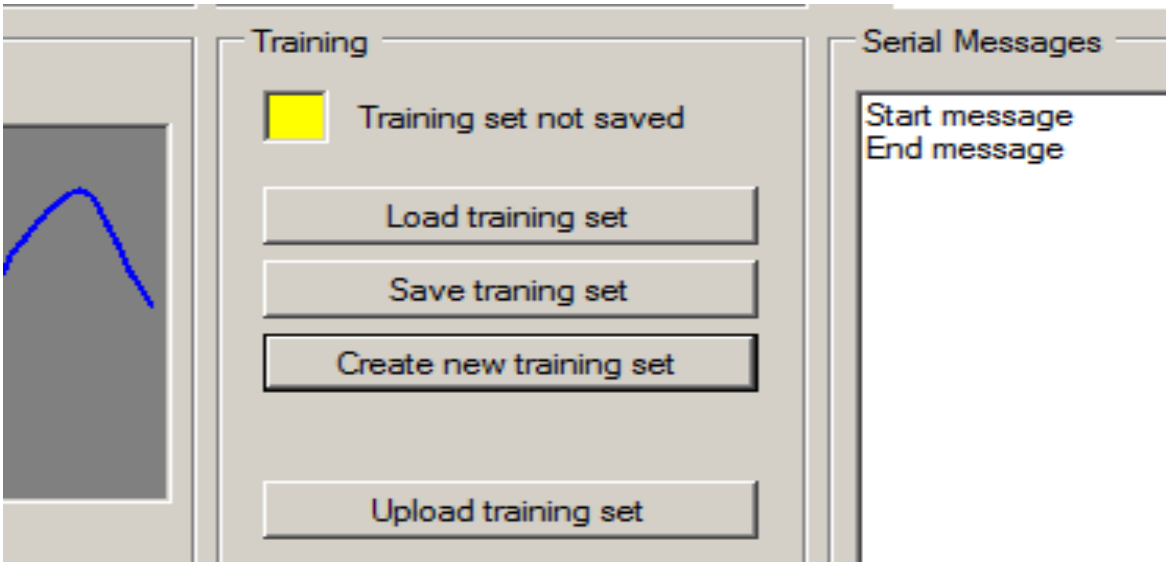

Fig 5. Unsaved Training set loaded in memory

3.2 Load training set

The Touch Solution GUI allows loading existing training sets. By clicking the 'Load training set' button, an open file dialog opens (see Fig 6). Navigate to the training set file (*.trs) and open it.

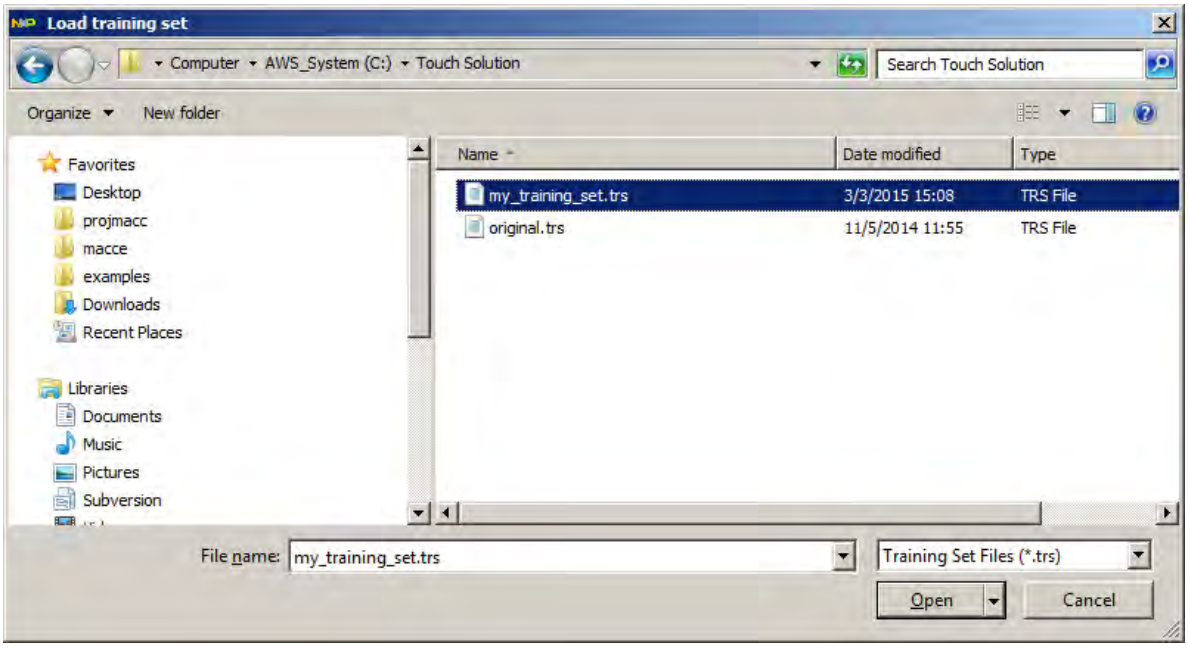

Fig 6. Load training set dialog

After opening the file, the loaded training set file name is shown. See Fig 7.

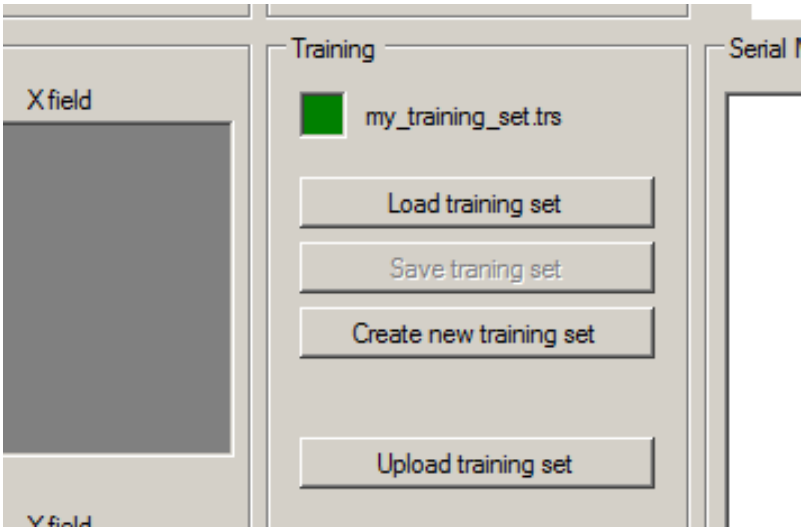

Fig 7. File: my_training_set.trs loaded in memory

3.3 Save Training set

Training sets that were created can be saved to a training set file (*.trs) and the name of the saved training set file is shown in the Training box status field

3.4 Upload training set

The Touch Solution GUI allows the direct uploading/storing of the training set to the target hardware (using USB - UART communication). As such the training set can be uploaded/stored in SRAM, flash, or EEPROM based on the target memory availability. The user must initialize/configure sufficient memory area accordingly in the application code. The current solution (application code) stores it in the Flash memory. After the training set is uploaded/stored, it can be used by the target gesture recognition library.

To upload a training set, use the ‘Upload training set’ button in the ‘Training’ panel. The status of the upload is shown in the analysis box. Currently, the 10 gestures are uploaded as 40 packets of 64 bytes (+ packet number and 16 bit CRC).

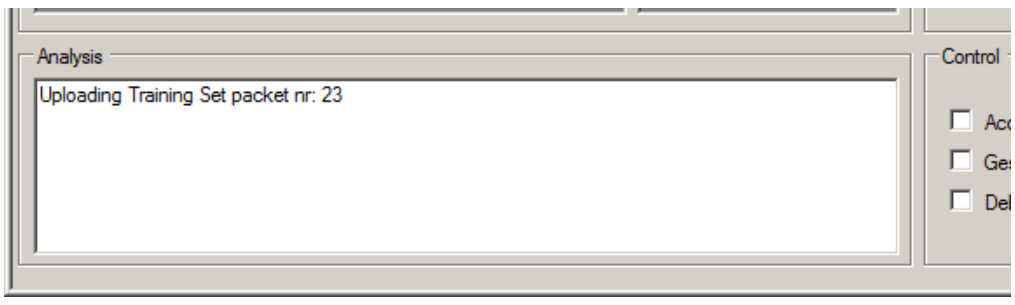

Fig 8. Training set uploading status

After the training set is uploaded to the target microcontroller, gesture recognition can be done. When the user draws the gesture on the touch pad, the recognition process can run and if there is a match found, it informs the user application about it.

3.5 Training set test

Before the currently created or loaded training set is uploaded to the target, the Touch Solution GUI provides an option to try and test it. To enable the GUI recognition, check the 'Gesture recognition' control box.

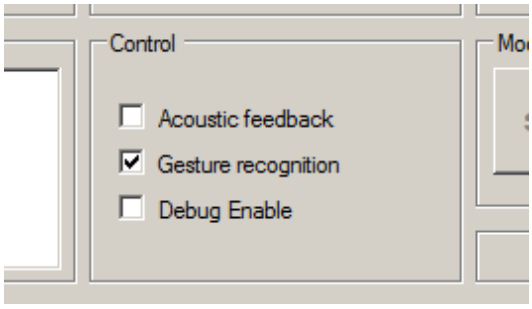

Fig 9. GUI enable gesture recognition

It is also possible to use both the GUI application and the target to do gesture recognition based on their own loaded training sets at the same time.

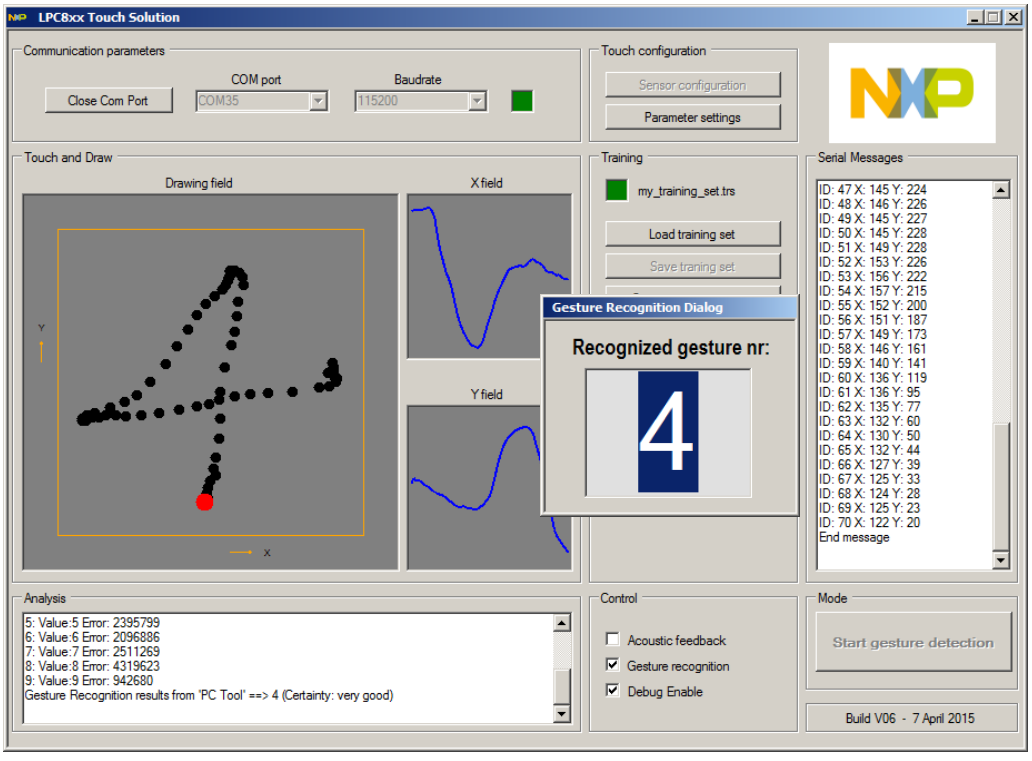

Fig 10. GUI gesture recognition

4. User Application

To build a user application that makes use of the Touch Solution and Gesture Recognition library, it is recommended to first download the latest library releases from LPCWare.com. Next, follow the sequence of actions described in this chapter to use the Gesture Recognition Library and its functionality.

4.1 Library include

Ensure that both the Touch library (***LPC82x_Touch_lib_v0100.lib***) and the Gesture library (***Gesture_lib_v0100.lib***) are included as part of your project. See the Keil example in Fig 11.

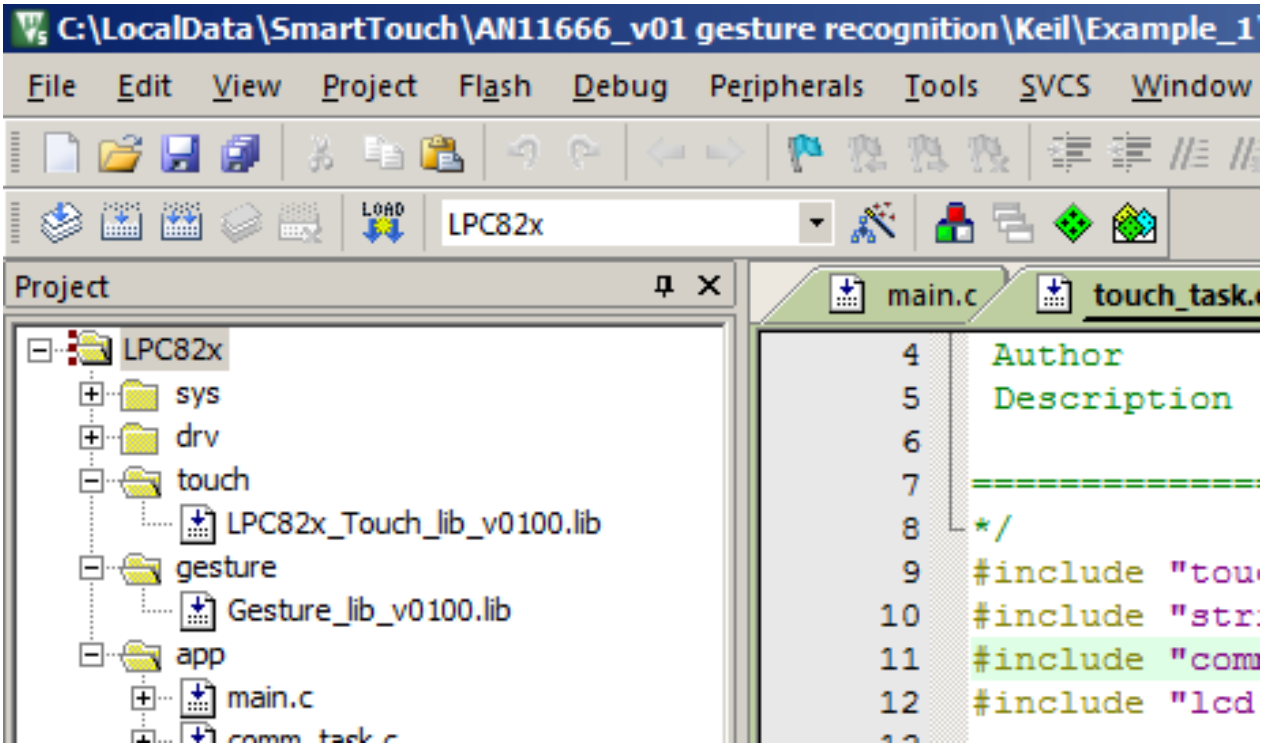

Fig 11. Library include

Note:

Source code modules that interface to these libraries should include the related header files ***LPC8xx_Touch_Solution.h*** and ***Gesture_Recognition.h***

4.2 Touch library configuration

All initialization and configuration of the capacitive Touch Solution library (for example, assignment of touch sensors to microcontroller port pins, touch algorithm parameter settings) is not handled in this document. For a detailed description see:

LPC82x Touch Solution | www.LPCware.com

Apart from handling the touch library events and passing the touch data, the next sections of this chapter focusses on Gesture Recognition library initialization and usage.

4.3 Training set check

Gesture recognition can only be used if a valid training set is loaded in the memory. The library initialization function requires a pointer to the memory location of that training set. Therefore, the user must check if the pointer is available before initializing the library. See Fig 12 for an example of how it is done. This example checks if the first 4 bytes of the training set (located at TRS_ADDR) are unequal to 0xFF.

```
void Init_Touch(void)
{
  uint8_t trs[4];

    // check if training set loaded in flash
    memcpy (&trs, (void *)TRS_ADDR, sizeof(trs));
    if (trs[0] != 0xFF && trs[1] != 0xFF && trs[2] != 0xFF && trs[3] != 0xFF)
    {
        ts_loaded = 1;
        gr.nr_of_gestures = 4;
        gr.trs            = (uint8_t *)TRS_ADDR;
        Gesture_Init();
    }
```

Fig 12. Check if training set loaded in memory

4.4 Gesture library initialization

Initialize the gesture configuration data structure parameters (GESTURE_T) and pass them to the library, calling the gesture library initialization function.

```
Gesture_Init();
```

Only two parameters need to be initialized, as an example:

```
gr.nr_of_gestures = 6;            // 6 gestures in loaded training set
gr.trs = (uint8_t *)TRS_ADDR;     // pointer to training set location
Gesture_Init();                   // call gesture init function
```

4.5 Passing touch data

After all the initialization is done, the user needs to handle the touch events and pass the touch position data to the Gesture_Task using the data type TOUCH_DAT_T, see Fig 13.

```
void TouchEventHandler(uint8_t event, uint8_t buf[4])
{
  TOUCH_DAT_T td;
  uint8_t     result = 0xFF;

    if (ts_loaded)
    {
        td.index = (buf[0] << 8) + buf[1];
        td.x     = buf[2];
        td.y     = buf[3];
        result   = Gesture_Task(&td);            // pass touch data to Gesture Task
    }
```

Fig 13. Touch event handler passing data to Gesture_Task

The Gesture_Task function should be called every time the touch solution library generates a touch event. It runs the recognition algorithm to identify the same pattern match with reference to the trained gestures.

On return, this function notifies the host application whether or not it found a match between the latest touch input data and the pre-loaded training set gestures.

```
uint8_t Gesture_Task(TOUCH_DAT_T *td);
```

Return	Value	Description
uint8_t	0 – 9	Match: number of the recognized gesture.
	0xFF	No match: no gesture pattern recognized.

4.6 Result handling

Finally, the user needs to handle the result returned from the gesture recognition task. This is completely application dependent. An example following the previous step is shown in Fig 14 .

```
        td.y     = buf[3];
        result   = Gesture_Task(&td);            // pass touch data to Gesture Task
    }

    switch (event)
    {
        case EV_START : LCD_PrintFullString("      ");
                        break;
        case EV_DATA  : break;
        case EV_END   : switch (result)
                        {
                            case 0  :   LCD_PrintFullString("LEFT  "); break;
                            case 1  :   LCD_PrintFullString("RIGHT "); break;
                            case 2  :   LCD_PrintFullString("DOWN  "); break;
                            case 3  :   LCD_PrintFullString("UP    "); break;
                            default :   LCD_PrintFullString("  --- "); break;
                        }
                        break;
    }
```

Fig 14. Gesture task result handling

This example checks for a touch “END” event and then sends a gesture match / no match result message to an LCD module.

Remark: see examples Gesture_EX1 and Gesture_EX2 described in section 5, User application examples for more details.

5. User application examples

The Gesture recognition application note shows two application example projects. They are published under the name "***AN11666 source code.zip***" and can be found at:

LPC82x Touch Solution | www.LPCware.com

5.1 Biometric behavior

The gesture recognition allows the biometric (behaviometric) identification of the user. As every user has his or her own style of writing different characters or numbers, it is quite challenging for any other person to reproduce them. Hence, this solution can find its applications in the areas of security, access control, or authentication.

As with most biometric authentication systems, the designers typically have to make a tradeoff between security and usability. A system that takes very precise readings of a hand writing, for example, may take a longer time to scan. The high level of precision makes the system more secure, but the slowness make it less usable. In general, higher security typically leads to lower usability, and vice versa.

5.2 Example 1

The first example uses the Touch and Gesture library to recognize 10 different drawn patterns from a loaded training set (that contains 10 gestures). It uses the LCD of the OM13081 kit to show the matching results as: nr 0, 1, 2 …9.

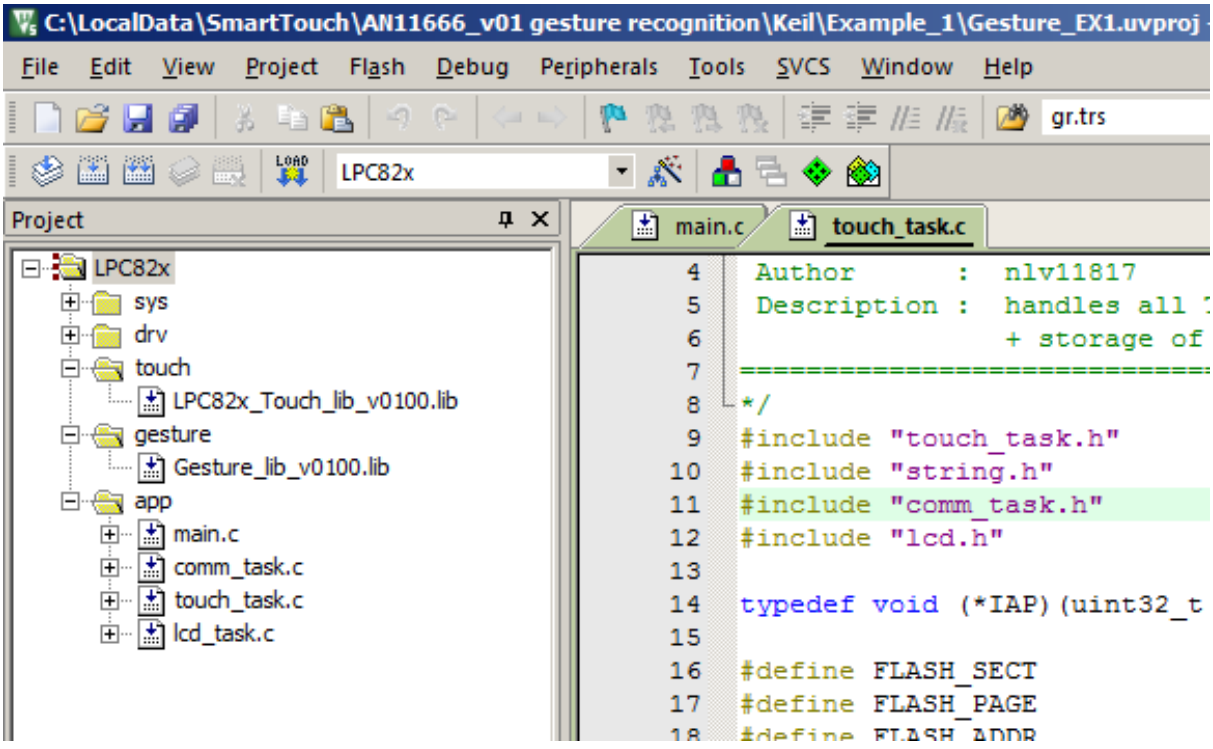

Fig 15. Gesture Recognition Example 1

To run this example on the 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 (***Gesture_EX1.bin***) from the main folder to the MBED mass storage device and manually reset when finished programming. During start-up the LCD shows messages:

NXP – V01.00 – GEST_1

Another option is to open the examples project file (for example in Keil), build it and program it to the LPC824 target. Fig 15 shows the complete project tree. All the touch and gesture initialization, configurations, storage of parameters and training set in flash, as well as touch event handling and passing of touch data to the gesture library is dealt with in ***touch_task.c*** and ***touch_task.h*** files.

All communication with touch GUI over UART is handled in ***com_task.c***, ***com_task.h*** and the driver files: ***usart0.c***, and ***usart0.h*** files. This includes setting the baud rate, reading / writing touch configuration parameters and downloading the gesture training set. The default baud rate is set as 115200. If other baud rates are required, change it in the com_task module. The UART to USB bridge on the debug part (LPC11U35 at the MAX-LPC824 board) communicates over USB to the Touch Solution GUI running on a host PC.

The gesture match / no match result is displayed on the on-board LCD module and is managed in ***lcd_task.c***, ***lcd_task.h***, and the driver files: ***i2cmst.c***, ***i2c.h*** and ***font.c***

The next step is to create your own training set that contains a maximum of 10 different gestures. For a quick start, the user can also skip this by loading an already created training set file called: ***number_0_9.trs***

Fig 16 shows how the gestures/patterns inside that training set are drawn. It contains 10 gestures/drawings for numerical numbers 0, 1, 2 . . . 9.

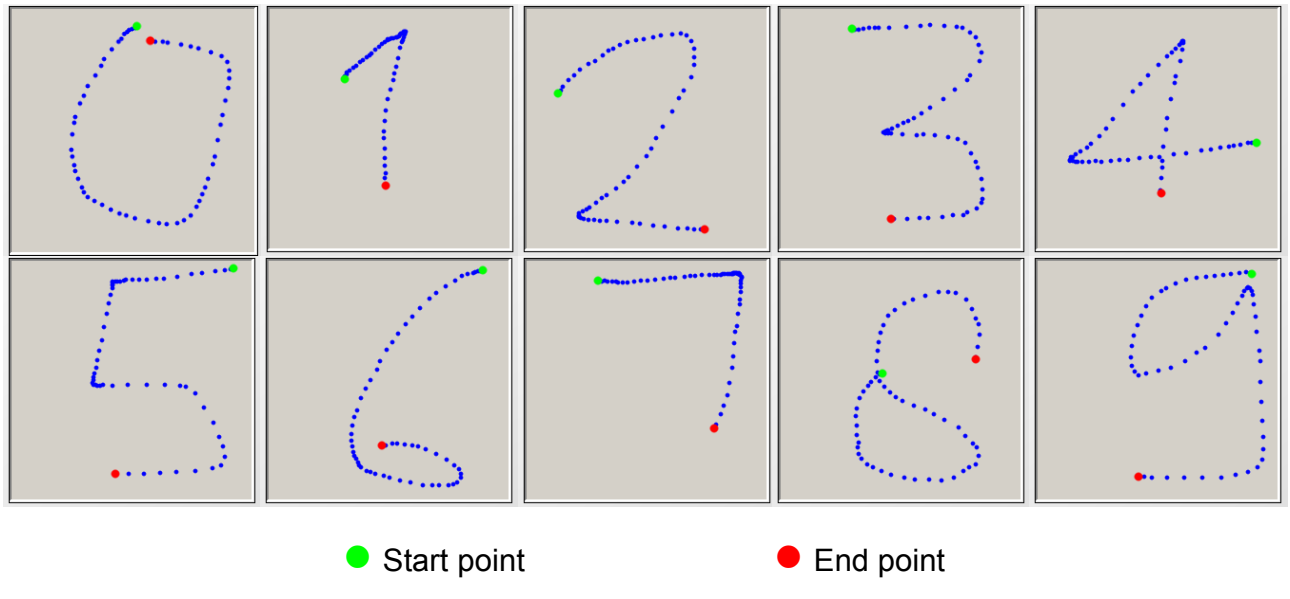

Fig 16. Trained gestures of training set: number_0_9.trs

After the training set is uploaded to the target, the example 1 application code recognizes and displays the status on the on-board LCD module.

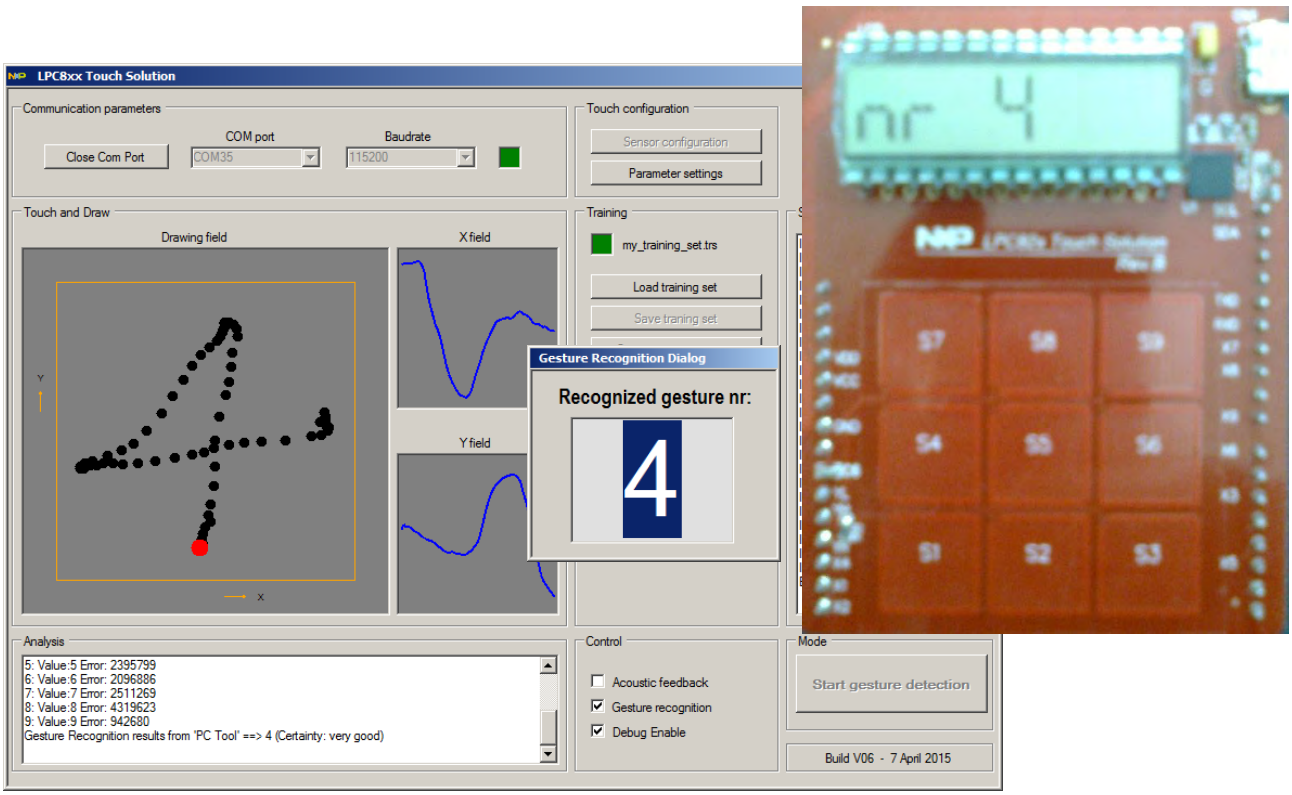

Fig 17. GUI and LPC824 target example 1 gesture recognition

5.3 Example 2

The second example uses the Touch and Gesture library to recognize four different swipe moves: up – down – left – right. The main difference with Example 1 is that the user application now has (pre) knowledge about the loaded gestures inside the training set. For example, it knows that the first gesture inside the training set is a swipe LEFT. If recognized, it uses the LCD of the OM13081 kit to show the results as: LEFT.

To run this example on the 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 (***Gesture_EX2.bin***) from the main folder to the MBED mass storage device and when finished programming give a manual reset. During start-up the LCD shows messages:

NXP – V01.00 – GEST_2

Another option is to open the examples project file (for example in Keil), build it, and program it to the LPC824 target. The next step is to create your own training set using the touch GUI that contains the following four swipe gestures.

Gesture 0 – LEFT Gesture 1 – RIGHT Gesture 2 – DOWN Gesture 3 – UP

For a quick start, the user can also skip this by loading an already created training set file called: ***4x_swipe.trs***. After the training set is created or loaded, the user has to upload it to the target microcontroller by clicking on “upload training set”.

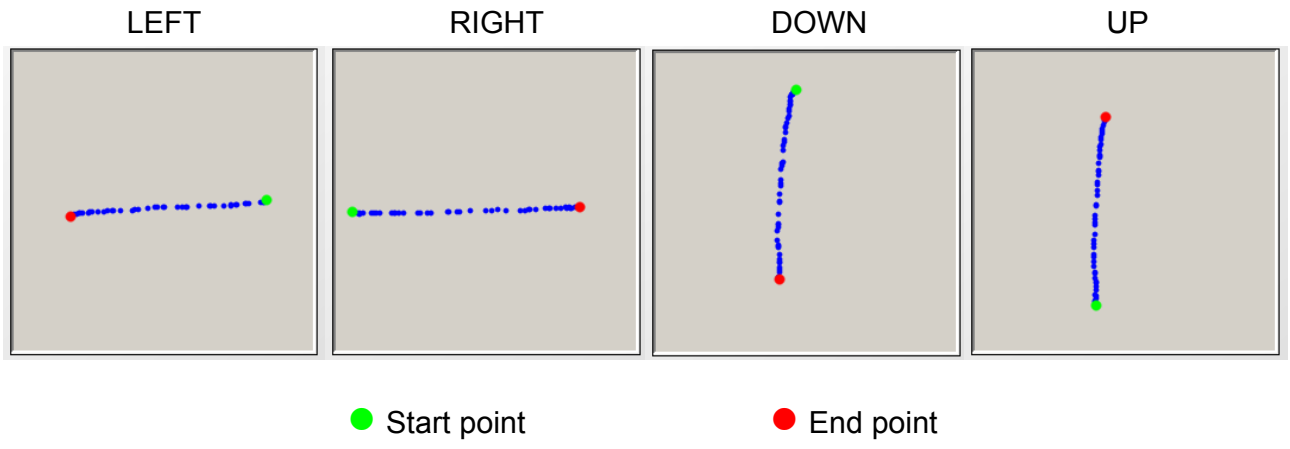

Fig 18. Trained gestures of training set: 4x_swipe.trs

This training set contains four gesture drawings for swipes LEFT – RIGHT – UP – DOWN. After the training set is uploaded to the target, the example application code recognizes these swipes and displays the result on the on-board LCD module. See Fig 19.

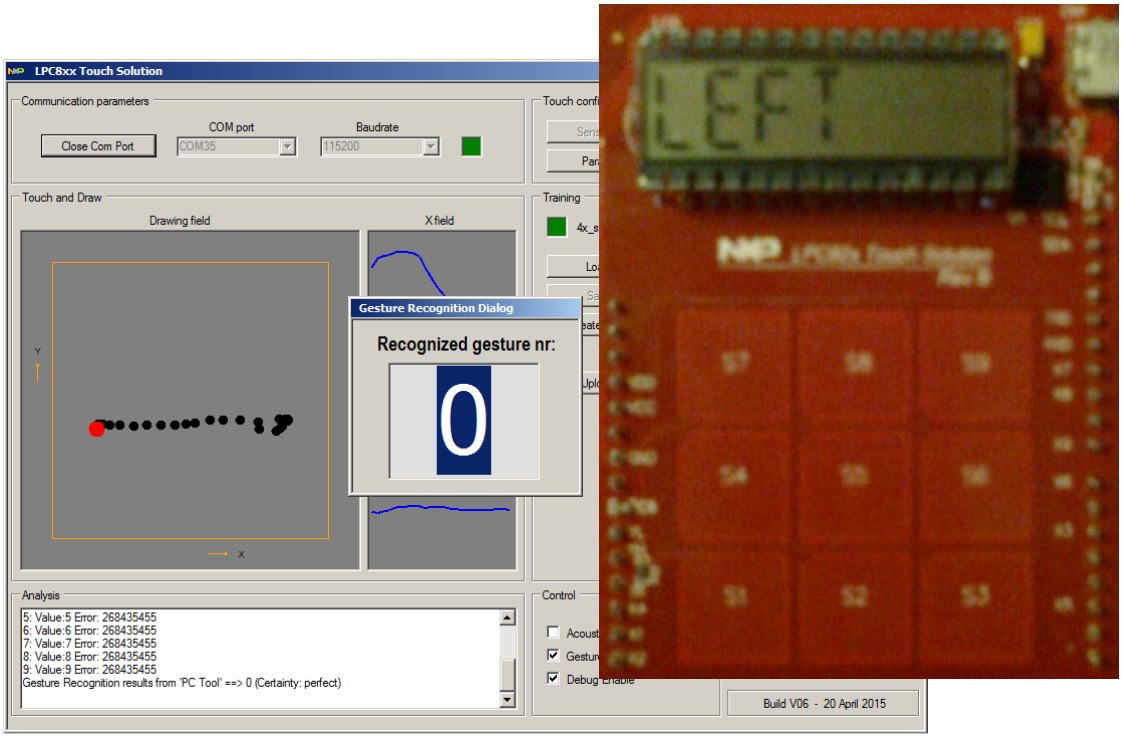

Fig 19. GUI and LPC824 target example 2 gesture recognition

6. Legal information

6.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

6.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

6.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

7. Contents

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 09 June 2015
Document identifier: AN11666