MPR121 Touch Sensing Design Guidelines

Introduction

Today touch sensing is a popular human machine interface you can find everywhere from consumer electronics to industry area. It is used in portable media player, eBook, notebook PC, mobile phone handset, household appliance such as LCD TV, AV system, remote controller, electrical kitchen appliance, and even in medical and personal health care devices.

MPR121 is an easy to use touch sensor. It can be used for many touch sensing functions such as touch button, keypad matrix, slide bar, touch wheel, touchpad, finger gesture recognition, hands grip/ hold detection, and near proximity detection.

Unlike straight forward mechanical button design, touch sensing design is much complex on the physical design. Many questions may be asked in a touch sensing design project using MPR121. The questions can be from product structure design, panel layout design, PCB layout, to software code programming. This application note is aimed to provide MPR121 touch sensing design guidelines so a better and reliable touch sensing design can be achieved.

MPR121 Key Features

MPR121 is Freescale second generation standalone touch sensing controller after its veteran E-field capacitance sensor series. Comparing to previous maximum 3 channels in MPR03x, MPR121 now has 12 sensing inputs while still keeping the same extreme low power consumption level. Another improvement is the advanced intelligence like auto optimization and configuration of the capacitance sensing parameter for each channel independently. This can greatly reduce the design engineer's trial and fine tune time. Beyond that, MPR121 also supports LED driver and GPIO function if the input is not used for touch sensing input.

The key features of the device are the following:

- 12 sensing inputs with 8 capable for LED driver or GPIO
- Power supply range 1.71~3.6V
- Low power consumption,
    - 29 µA typical current in run mode
    - 3µA standby current in stop mode
- Stand alone intelligent touch sensing engine
    - Auto environment capacitance calibration and baseline tracking
    - Auto configuration for capacitance sensing parameter optimization
    - Internal 2 stage signal filtering
    - Independent touch/release trip threshold provide hysteresis
    - Independent current/time setting allows buttons of various shape and size

- Near proximity sensing by simultaneous charging on all the electrodes
- Each channel provides touch/release status report as well as filtered raw data output
- I2C bus communication, and Interrupt output
- Pin selectable 4 I2C address enables channel expansion
- Compact 3x3x0.65mm, 20 pin QFN package
- -40°C to +85°C operating temperature range

Figure 1: MPR121 12 channel standalone touch sensor in a 3x3x0.65mm 20 pin QFN package(close to real size)

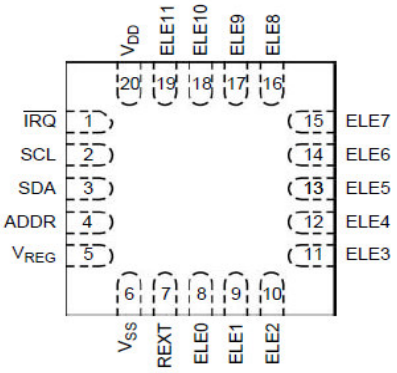


Figure 2: MPR121 pin out top view

Touch Sensing Pattern and Channels

Simple touch button

When the number of touch buttons required is below or equal to 12, we can directly assign one sensing input for each touch button, the unused inputs can be used for LED driver or GPIO function.

In this configuration, the software code is the simplest as we get all the button status report directly from MPR121 12 bits electrode status report (Registers 0x00[0:7], 0x01[0:3]). Schematic in Figure 3 shows this kind of configuration:

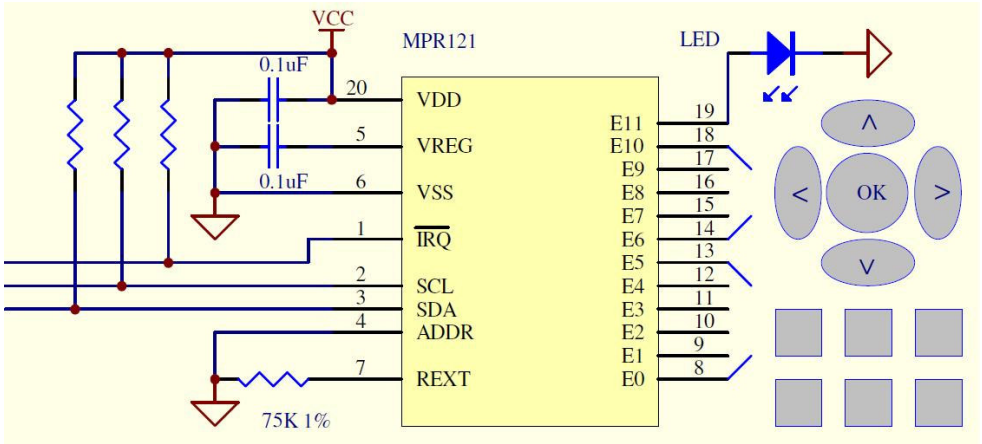


Figure 3: Schematics showing a 6 buttons array and 4 way navigation buttons plus a LED output

Expanding channels with multiple MPR121

When the required number of touch buttons is larger than 12, one possible solution is using multiple MPR121 to expand the inputs. This is very easy to do as MPR121 I2C address is pin selectable which enables easy channel expansion. When the ADDR pin is tied to VSS, VDD, SDA, SCL, the address selected is 0x5A, 0x5B, 0x5C, 0X5D respectively, refer to Table 1.

Table 1: MPR121 selectable I2C address

ADDR Pin Connection	I2C Address
VSS	0x5A
VDD	0x5B
SDA	0x5C
SCL	0x5D

Using 4 MPR121 tied to one I2C bus, we can expand the number of touch buttons to 48. This kind of channel expansion is the simplest without much design headache, the only expense is cost. Figure 4 shows how to use multiple MPR121 in one system.

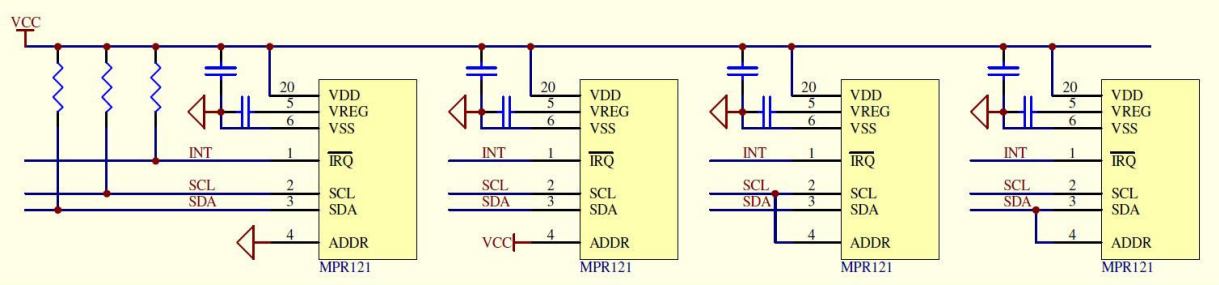


Figure 4: Parallel multiple MPR121 to one I2C bus

Multiplexed buttons

Most often when cost matters channel multiplexing can be considered. By channel multiplexing a logical touch button is composed by multiple sensing pads from different inputs, which can be 2 or more sensing inputs. The button will only be activated /deactivated when all the used sensing channels are touched /released simultaneously. Figure 5 shows a touch button composed by 2 sensing input channels. Figure 6 shows a pattern of 4 way navigation with a centered “OK” button by using 4 sensing inputs.

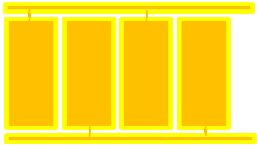


Figure 5: One touch button composed by 2 sensing inputs

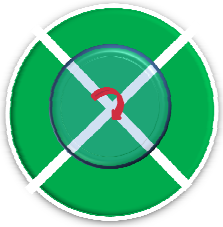


Figure 6: 4 Navigation buttons with a centered virtual acknowledge button composed by 4 sensing inputs

In Figure 5, the status of the button is decided by monitoring two channels simultaneously, only when both inputs report touch status the button is considered touched. In Figure 6, the "OK" button will only be activated when all four channels are effectively touched. The logic used here is "AND" logic for a valid touch.

By duplicating the button in Figure 5 into a XY array and connecting each button's two inputs with respective row input and column input we can get a matrix keypad, for example, the 5x3 keypad matrix in Figure 7.

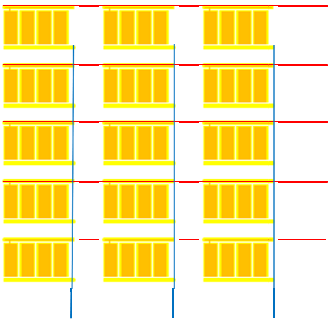


Figure 7: Keypad array by XY matrix sensing inputs

Multiplexing can effectively reduce the number of input channels while expanding the number of buttons can be supported. As an example, a 6x6 matrix configuration can provide up to 36 buttons by using one single MPR121. Using multiplexed matrix, the track routing is also much simple.

However, multiplexing is worth more careful thoughts before you embarking on the pattern layout.

1. Multiplexing can only be used when there is no ambiguity on button operation and function definition. It needs to be noticed when one channel is used in both multiplexed button and non_multiplexed button in one design, such as the case in Figure 8.

Figure 8: Media player control using 2 sensing inputs

In Figure 8 three buttons are defined for media playback but using only 2 inputs. When touching button "< "and button ">" together, or touching all three buttons at the same

time, the 2 channel status report will be exactly the same as only touching “play/stop” button. Although there is some signal level difference, these three scenarios cannot be differentiated from channel status bit. It is lucky that in practice this will not cause much confusion if we just take all three situations as a “play/stop” button touch.

When all buttons are configured in matrix mode, since each button requires distinct 2 inputs combination, there shall be no ambiguity problem.

2. Multi touch can be detected, but there are limitations. Take matrix configuration as an example, multi touch on one row or column can be correctly detected, but when multiple rows and columns are touched at the same time, then there is a “ghost“ point which cannot be distinguished from the true touch point.

   Although there are limitations, multi touch still can be detected, which is very useful in touch sensing design. For example, typically it is required that only one button touch is valid to reject unintentionally multi button touch which could be caused by user holding the device in hand. By multi touch detection, we can know how much area is touched, and we can differentiate unintentionally touch from a valid touch which can also have activated the buttons in nearby the total area is much smaller.

3. Multiplexing requires a relatively higher sensitivity since the sensing pad composing the multiplexed button could be relatively small compared to a non_multiplexed button at the same size.

   Using the example of the “play/stop” button in Figure 8, both two channels composing the button can have only about half of the total button area. As required, this half button area should be able to activate the respective channel when the button is touched.

   So most often multiplexing is used with slim cover overlay or intentionally increased button size, which effectively overcomes low sensitivity of small pad size. Refer to section below for more on the sensitivity issue.

4. The software code shall be still relatively simple. Situations of all kinds of different channel combination need to be considered.

In summary, the use of multiplexing needs to be evaluated carefully with the actual system setup before you can benefit from it. Figure 9 shows an example of common key pad we see in mobile handset which has 15 buttons and a 4 way navigation click wheel using just 12 inputs. With careful arrangement, the supported number of buttons can be further expanded.

Figure 9: Typical mobile keypad with navigation click wheel

Touch slider and touch wheel

A linear touch slide bar or touch wheel can be done by sequencing and combining multiple sensing inputs. These inputs are aligned in a line or a circular way.

There are three methods when design with touch slider and touch wheel. We can get the finger position information simply by monitoring which input is activated and having the highest signal level. Base on the first method, we can further figure out the finger position in between of two neighboring pads if signal levels from the neighboring pads are at the same level. The most complicated method is by using interpolation calculation, which can provide very higher resolution but using less input channels.

The number of sensing inputs used to form a bar or wheel is depended on the total span, the resolution needed, and which method will be used. For example in Figure 10, using the first approach, a slider by 6 inputs is able to cover a length of 30mm with a minimum resolution of 6 step positions. This design is common in volume adjustment we see. The finger position can be obtained by MPR121 touch status report and determined by the channel having the highest signal level.

Higher resolution can be achieved if finger position in between two neighboring pads is also considered. This is obtained by taking into consideration when the signal levels from two neighboring pads are at the same level. Using this simple method we can easily increase the resolution from originally 6 to 6+5=11.

Figure 10: Slider with 6 inputs. 6 step resolution vs. 11 step resolution

Figure 11 shows an example using the third method. Pattern in Figure 11 is often used in practice for slider bar and touch wheel. This method use spatially interleaved gradually changing electrode pads, and then the finger position can be calculated out by interpolation. In Figure 10 we actually see an example of 1 point interpolation between two neighboring regular pads. Figure 12 shows an extreme example that using only 2 sensing inputs for a touch slider.

Figure 11: Increase the resolution by interleaved pattern and interpolation calculation

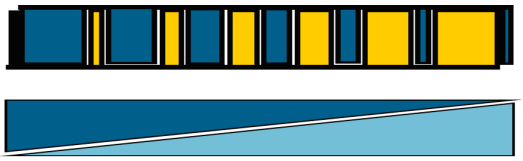


Figure 12: Linear slide bar using only 2 sensing inputs

For a spatially interleaved gradually changing electrode pads pattern, all the segments nearby contribute to the finger position depending on the level of coupling between the finger and the segments on the electrode patterns. With a design of good channel sensitivity we can get a calculated resolution as high as 256 steps on a slider bar or 128 points on a touch wheel.

The Interpolation algorithm software code can be obtained from MPR121 demo reference software code.

Touchpad

Touchpad enables 2D free finger movement on a wide surface area instead of 1D movement on a line or circle. Touchpad can be used for many functions that cannot be realized by simple touch button, slider or touch wheel, such as complex finger gestures recognition, 2D mouse move emulation and even finger track recording. On the other side, touchpad is still able to support 1D function with virtual button, slider and touch wheel.

In the simplest realization, touchpad can be composed by closely spaced button array. For example in Figure 13 a 3x3 button array can provide the basic gestures like up/down/left/right /diagonal scroll, as well as 9 touch button inputs and “anywhere” tap detection.

The array size of this kind of pattern is normally small such as 3x3 or 3x4, limited by the 12 inputs of MPR121. It is possible to have the whole pattern in a single layer since track routing is relatively easy when the array size is small. The one layer array solution gives the benefit of lower cost comparing to two layer approach, which becomes more obvious when using ITO glass design.

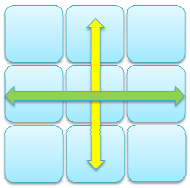


Figure 13: A 3x3 button array for simple gesture detection

Unlike the one layer button array, a 2 layer matrix pattern can help to reduce the sensing input channels while quickly expanding the array size. For example by using one MPR121, the 12 inputs can be configured into a 6x6 matrix array, providing approximately a 30mmx30mm touchpad size, which fits into most handheld portable device panel such as mobile phone, MP4 player, eBook or MID. Shown in Figure 14 is an example of 5x7 matrix touchpad. By using 2 or more MPR121, we can even support big panel size such as 12x12, 24x24.

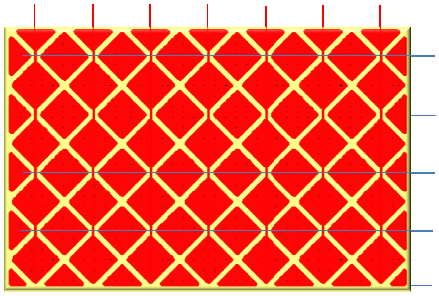


Figure 14: A 5x7 touchpad using MPR121

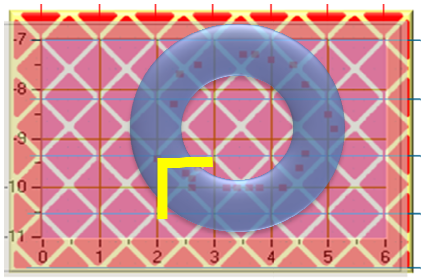


Figure 15: Touchpad with trace tracking and a circle gesture

Same as previous examples of slide bar and touch wheel, software interpolation can be used to get higher resolution than actual pixel numbers. In Figure 15, the interpolated track clearly shows a circle gesture. Using touchpad, application like picture scroll/pan, menu scroll and select, mouse move emulation can be realized. Refer to MPR121 demo for these application and reference code.

The matrix touchpad can be realized using PCB, Flex PCB or ITO glass. The pixel size is determined by the cover overlay thickness, but usually is below 5mm diagonal to best accommodate typical finger tip size. For designs using plastic cover overlay such as PET material, touch sensitivity is good when the cover thickness is below 0.5mm. Using slim (<0.5mm) plastic sheet as cover material can help to get even better sensitivity.

Structure Design and PCB Layout

Industry design and PCB layout are two most important and fundamental parts in the whole design process. Capacitive touch sensing enables modern artistic industry design such as compact and curved shape, but this can also lead to the needs of careful structure design and PCB layout.

Eliminate the air gap

Quite often one meets the problem that the main PCB cannot contact with the cover overlay material directly and closely because of curved cover shape or limited by the mounting of main PCB.

For example the structure design may requires that the main PCB to be mounted in a 50mm distance from the plastic cover. In this cases, the best choice maybe a dedicated touch sensing daughter card accommodating all the sensing patterns and the MPR121 controller in one place, such as a piece of FR4 PCB, or flex PCB. Sometimes lower cost 0.3mm slim bendable PCB maybe a choice to better fit the curved shape.

It also should be pointed out air gap between the PCB and the overlay can greatly degrade the capacitance sensing field and sensitivity becomes very low, so air gap must be eliminated in some way.

Depending on the sensitivity, even small air gap below millimeter level can result a touch cannot be detected. In practice, various adhesive tape or film can be used between the PCB and the cover, this helps to close the air gap and make a seamless and stable PCB / cover structure, achieving stable and reliable touch sensing detection. Refer to Figure 16 for typical structure used.

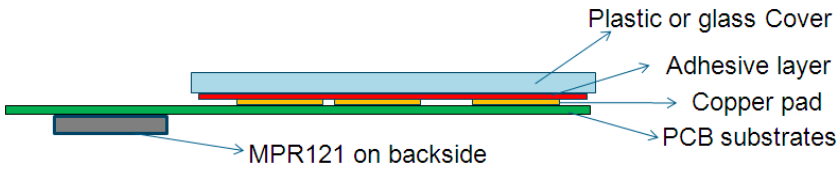


Figure 16: Cross section: no air gap between the pad and top cover. MPR121 on outside of bottom layer

In case where PCB is mounted at a distance about 10mm from the cover and the touch sensing pattern is simple round shape, we can use metal coil spring or conductive sponge foam such as used for EMC to fill the gap. Figure 17 shows an example used in electric /inductive cooker touch sensing panel.

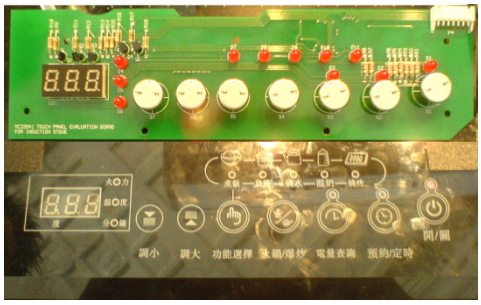
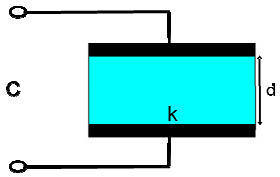


Figure 17: Using metal coil spring to close the air gap

Overlay thickness and touch sensing pattern size

The thickness as well as the dielectric characteristic of the PCB overlay and cover, and the pattern size all plays an important role on the touch sensitivity. These factors shall be carefully evaluated and decided based on the real system setup.

To understand how the touch response is affected by these 3 factors, let's look at the paralleled capacitor model below.



$$C = \frac{K\varepsilon_0 A}{d}$$

Figure 18: Parallel plate capacitor model

In above equation, C is the capacitance value, K is the material dielectric constant relative to air for the medium between two 2 plates, ε_0 is the dielectric constant of air, and A, d is the 2 plates overlaid area and relative distance between the two plates respectively.

In touch sensing application, the sensing pad provides one of the plates in the paralleled capacitor model, and body or finger and its ground return path form the other plate. Since touch/ release is detected based on the delta C amplitude, from above equation smaller d or larger A will help to give larger delta C change, which suggests to use slim overlay material and larger sensing pad size.

Table 2 and Figure 19 provide an example of delta signal change under various pad size and PET overlay material thickness by using MPR121 demo kits. It can be seen that the usable minimum pad size is about 3mm for 0.5mm PET cover, while 7mm pad can be used with 2.0mm PET cover. It is suggested that we should use slim cover and big pad size to get higher SNR.

Depending on the actual noise level, keeping the touch delta counts above 10 will result a relatively good touch sensitivity and SNR in most design cases.

Table 2: Delta counts under various pad size and PET thickness

Pad Size diameter	0.5mm PET ΔCounts	1.0mm PET ΔCounts	1.5mm PET ΔCounts	2.0mm PET ΔCounts
2mm	2	0	0	0
3mm	8	3	2	1
4mm	15	7	3	2
5mm	21	10	6	5
6mm	30	18	6	6
7mm	40	22	15	11
8mm	47	29	20	13
9mm	56	33	22	13
10mm	60	38	25	17
11mm	73	44	28	17
12mm	76	54	29	21
15mm	98	57	39	28

Test condition: 1.8V supply, non isolated, round pad with ground fill on MPR121 demo FR4 PCB (1.0mm) bottom layer, each PET sheet has a mean thickness of 0.5mm

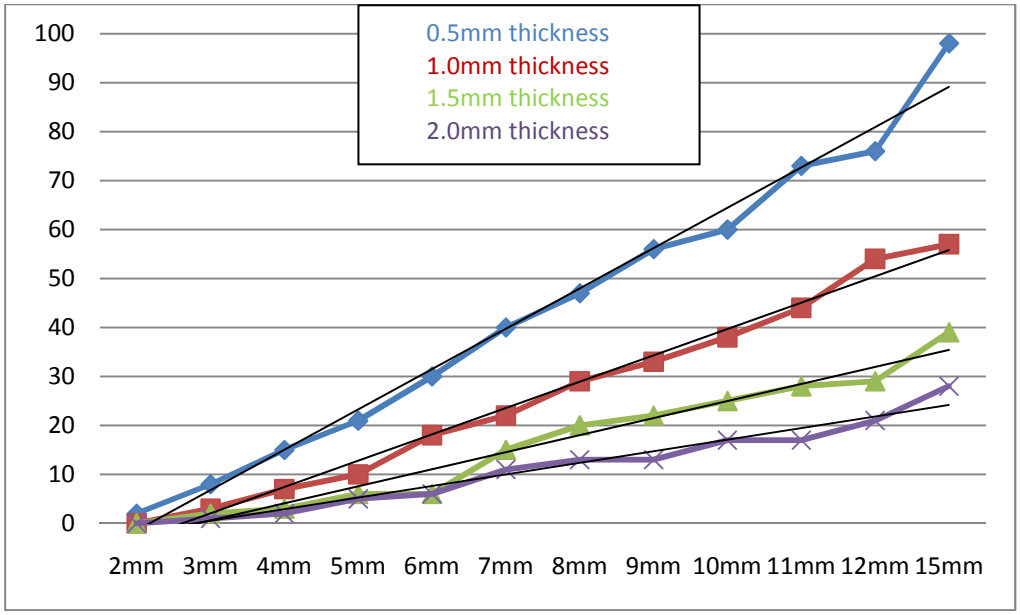


Figure 19: Delta signal change under different pad size and cover thickness

Notice K is also on the numerator as A, higher K material also helps increase the touch response. It's easy to understand that if two cover overlays are at the same thickness, then larger K material will have larger touch delta signal change then lower one. For example when the thickness are both 1.0mm, a glass (normally K>=4) cover will result a better sensitivity than a plastic cover (normally K=2).

The concept can be further extended to serial or parallel multiple capacitors. Parallel multiple sensing pads so that A is effectively increased that it can provide larger covering are hence higher sensitivity for touch. This is the concept MPR121 used for near proximity detection. Using the serial capacitor concept, when multiple overlays are stacked together, the layer with the smaller k (for example air) will be the dominator one even its thickness may be small. This also explains why we need to eliminate the air gap to get better sensitivity.

PCB Layout, Noise interference and ground fill

Capacitive touch sensing is basically an electric field detection method. The electric magnetic field generated by nearby components and devices create an electric field environment which is possible to produce a lot of noise interference to the touch sensing. This EM field can affect all elements in the touch sensing signal chain, including the controller, track, sensing pads, while the most easily to be impacted is the sensing pad as it is able to act as an open antenna.

Sometimes EMI source is inevitable and apparent, for example, the RF components in mobile handset including the power amplifier, the antenna. Sometimes it is not so obvious, but also can generate EMI such as inductors used in switching power supply circuit.

The first rule is to put the controller, track and sensing pad away from EMI components as far as possible to avoid the EMI. Put all the sensing elements on the far opposite side of the EMI source on the PCB. The distance needed to keep the sensing elements away from the EMI source is depended on the EMI strength level and the sensing elements size. It is always the best practice to take experiment to evaluate the proper distance needed.

In space limited design, EMI protective solutions like RF shielding and isolation shall be used. This can be RF shield can, PCB internal ground shielding layer and ground isolation.

On the other side, although ground isolation and ground layer can be helpful to EMI reduction, they also reduce the capacitance sensing field as it also tends to find the shortest return path to the ground. The sensing fields return directly to ground will not contribute to touch sensing but only forms the background capacitance baseline.

We need to reduce the background capacitance as much as possible, and make sure the ground connection does not deteriorate the touch sensing sensitivity too much while keeping the necessary EMI isolation. In case there is no EMI issue, put the ground fill away from signal tracks and sensing pads as far as possible. If ground fill must be used under or nearby the sensing pads, then use a 40%~50% hatched ground fill maybe enough for proper RF isolation.

For more details about pad layout, refer to Freescale application note AN3863 and AN3747.

Proximity detection

MPR121 multiplexed inputs support the combination of some or all inputs into one channel for dedicated proximity detection. With this method the same touch button keypad can be changed into a large pad for proximity detection. MPR121 has internal independent settings for

touch and proximity detection, so touch button detection and proximity detection can be performed at the same time.

Using the 12 numeric keypad demo in MPR121 demo kit can provide a distance of about 2cm for hand palm detection. In another demo, a pattern combined by two concentric circles of 10cm and 5cm diameter can provide a 5cm distance proximity detection when hand approaching.

It shall be noted that larger pad size is also more prone to EMI interference. One can use non-solid sensing element such as wire loop or 40~20% copper filled pad to act as the proximity sensing elements.

Make a prototype

As previously mentioned that industry design and PCB layout are two most important and fundamental parts of the whole design process, it is helpful to make a prototype for design evaluation purpose before formal product design. The purpose is to evaluate the EMI environment, the necessary pattern shape and size for given cover material and thickness. This will help to reduce the reworks on PCB layout and make sure the SNR is within expected range.

Freescale provides evaluation tools and demo kits to help design evaluation, refer to figure 20. First you need to make a prototype of the product assembly. The prototype shall include the key components such as the RF section, other noise generating device if there is, and touch sensing sub systems with the cover material of the same thickness to be used.

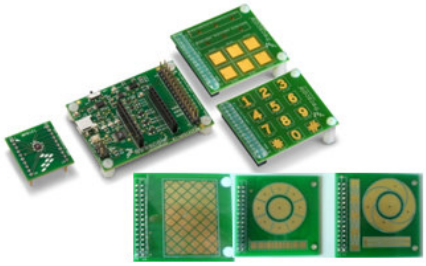


Figure 20: MPR121 evaluation and demo kit

The touch sensing sub system used can be very simple and easy to do, such as a strip of copper of the intended pattern size and shape on a ground filled PCB. You can then connect the copper pads to the input channels on MPR121 evaluation kit. Another better way is to make a prototype PCB board with MPR121 controller and sensing pattern all in on one place, such as the one in Figure 21. Then connect the interface bus, including power supply, I2C, IRQ to the socket on MPR121 evaluation kit.

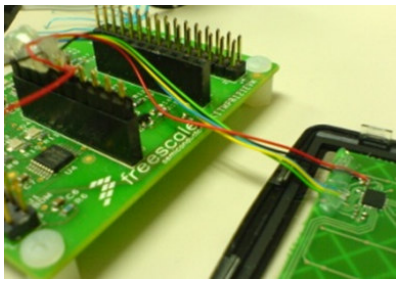


Figure 21: Linking the prototype to MPR121 demo kit

Using the GUI provided with the evaluation kit and demo kit, you shall be able to run the sensor and test it under different working situations. When using Freescale MCU, you can also use the PC Freemaster software, which provide another way of debug and test with graphical interface. Figure 21 shows a screen copy of the Freemaster GUI. The Freemaster software can be downloaded from Freescale website. For the demo reference software code running on Freescale MCU, you can refer to MPR121 evaluation kit and demo kit.

Software Implementation

I2C communication

MPR121 is a standalone touch sensor controller. It acts as a I2C slave device and communicates with the host processor by I2C bus and interrupt pin. The I2C communication can be done with either a dedicated I2C peripheral in host, or by using bit bang GPIO emulated I2C. The MPR121 has 4 selectable I2C addresses depending on ADDR connection, refer to Table 1.

Refer to Figure 23, it shall be noted that repeated start (Sr) shall be used in read byte operation, ended by not-acknowledge and stop. For a write operation to all the configuration registers except LED control registers, we can only write after setting MPR121 into stop mode (none of the area and channel enable bit is 1 in 0x5E). Below is the write and read operation format. Refer to MPR121 demo firmware code for the reference I2C communication code.

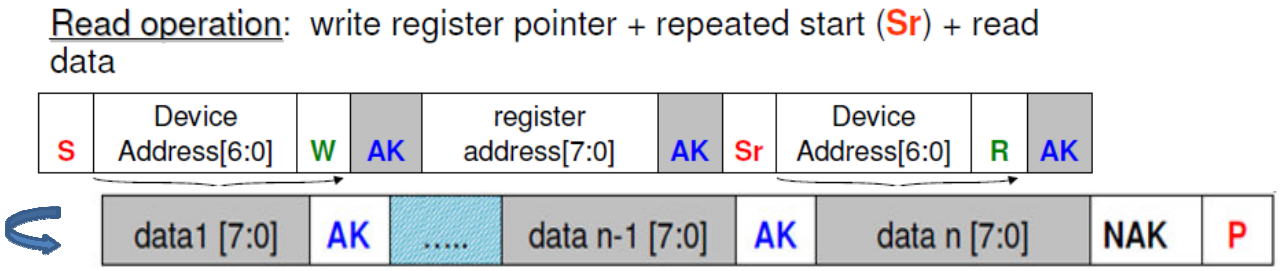


Figure 22: MPR121 read format. Gray means bits sending from MPR121 to host.

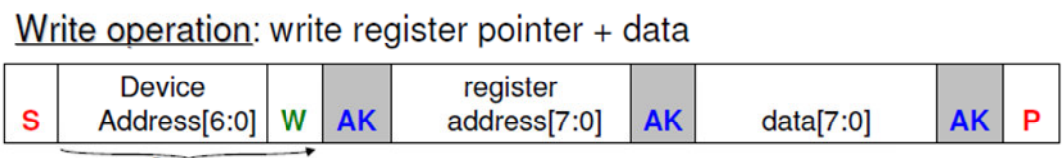


Figure 23: MPR121 write format. Gray means bits sending from MPR121 to host.

Initialization procedure

MPR121 has internal 128 user accessible registers for various functions configuration and status report. These registers can be divided into several groups according to their functions: status register, signal and baseline registers, threshold registers, charge current/time setting registers for each channel, filter setting registers, auto configuration registers and LED/GPIO function registers.

All registers in MPR121 default is 0x00 after power up reset, except register 0x5C default is 0x10 and register 0x5D default is 0x24. MPR121 needs to be properly configured before working for touch sensing.

The initialization procedure primary includes threshold setting for each channel, charge current / time setting for each channel and baseline filter configuration.

Since MPR121 provides auto configuration capability, once the auto configuration control register is properly set, there is no need to set the charge current / time setting for each channel separately, this helps to reduce the number of registers need to be configured, most important it helps to reduce the register setting fine tune time significantly. Below is the sample code for MPR121 initialization and configuration.

```
#define TouchThre 10                    // Touch threshold
#define ReleaThre 8                     // Release threshold
#define Prox_TouchThre 6                // Proximity threshold
#define Prox_ReleaThre 4                // Proximity release threshold

void MPR121_init_to_run(void)
{
 IIC_ByteWrite(0x80,0x63);              // Soft reset MPR121 if not reset correctly

 IIC_ByteWrite(0x73,0xFF);              // LED Configuration, if not used, this part can be omitted
 IIC_ByteWrite(0x74,0xFF);
 IIC_ByteWrite(0x76,0xFF);
 IIC_ByteWrite(0x77,0xFF);
 IIC_ByteWrite(0x75,0xFF);
 IIC_ByteWrite(0x81,0x00);
 IIC_ByteWrite(0x82,0x00);
 IIC_ByteWrite(0x83,0x00);
 IIC_ByteWrite(0x84,0x00);

 //touch pad baseline filter
 //rising: baseline quick rising
 IIC_ByteWrite(0x2B,0x01);              // Max half delta Rising
 IIC_ByteWrite(0x2C,0x01);              // Noise half delta Rising
 IIC_ByteWrite(0x2D,0x00);              // Noise count limit Rising
 IIC_ByteWrite(0x2E,0x00);              // Delay limit Rising

 //falling: baseline slow falling
 IIC_ByteWrite(0x2F,0x01);              // Max half delta Falling
 IIC_ByteWrite(0x30,0x01);              // Noise half delta Falling
 IIC_ByteWrite(0x31,0xFF);              // Noise count limit Falling
```

```
IIC_ByteWrite(0x32,0x0);              // Delay limit Falling

//touched: baseline keep
IIC_ByteWrite(0x33,0x00);             // Noise half delta Touched
IIC_ByteWrite(0x34,0x00);             // Noise count Touched
IIC_ByteWrite(0x35,0x00);             // Delay limit Touched

//Proximity baseline filter

//rising: very quick rising
IIC_ByteWrite(0x36,0x0f);             // Max half delta Rising
IIC_ByteWrite(0x37,0x0f);             // Noise half delta Rising
IIC_ByteWrite(0x38,0x00);             // Noise count Rising
IIC_ByteWrite(0x39,0x00);             // Delay limit Rising

//falling: very slow rising
IIC_ByteWrite(0x3A,0x01);             // Max half delta Falling
IIC_ByteWrite(0x3B,0x01);             // Noise half delta Falling
IIC_ByteWrite(0x3C,0xff);             // Noise count Falling
IIC_ByteWrite(0x3D,0xff);             // Delay limit Falling

//touched
IIC_ByteWrite(0x3E,0x00);
IIC_ByteWrite(0x3F,0x00);
IIC_ByteWrite(0x40,0x00);

//Touch pad threshold
IIC_ByteWrite(0x41,TouchThre);        // ELE0 TOUCH THRESHOLD
IIC_ByteWrite(0x42,ReleaThre);        // ELE0 RELEASE THRESHOLD
IIC_ByteWrite(0x43,TouchThre);        // ELE1 TOUCH THRESHOLD
IIC_ByteWrite(0x44,ReleaThre);        // ELE1 RELEASE THRESHOLD
IIC_ByteWrite(0x45,TouchThre);        // ELE2 TOUCH THRESHOLD
IIC_ByteWrite(0x46,ReleaThre);        // ELE2 RELEASE THRESHOLD
IIC_ByteWrite(0x47,TouchThre);        // ELE3 TOUCH THRESHOLD
IIC_ByteWrite(0x48,ReleaThre);        // ELE3 RELEASE THRESHOLD
IIC_ByteWrite(0x49,TouchThre);        // ELE4 TOUCH THRESHOLD
IIC_ByteWrite(0x4A,ReleaThre);        // ELE4 RELEASE THRESHOLD
IIC_ByteWrite(0x4B,TouchThre);        // ELE5 TOUCH THRESHOLD
IIC_ByteWrite(0x4C,ReleaThre);        // ELE5 RELEASE THRESHOLD
IIC_ByteWrite(0x4D,TouchThre);        // ELE6 TOUCH THRESHOLD
IIC_ByteWrite(0x4E,ReleaThre);        // ELE6 RELEASE THRESHOLD
IIC_ByteWrite(0x4F,TouchThre);        // ELE7 TOUCH THRESHOLD
IIC_ByteWrite(0x50,ReleaThre);        // ELE7 RELEASE THRESHOLD
IIC_ByteWrite(0x51,TouchThre);        // ELE8 TOUCH THRESHOLD
IIC_ByteWrite(0x52,ReleaThre);        // ELE8 RELEASE THRESHOLD
IIC_ByteWrite(0x53,TouchThre);        // ELE9 TOUCH THRESHOLD
IIC_ByteWrite(0x54,ReleaThre);        // ELE9 RELEASE THRESHOLD
IIC_ByteWrite(0x55,TouchThre);        // ELE10 TOUCH THRESHOLD
IIC_ByteWrite(0x56,ReleaThre);        // ELE10 RELEASE THRESHOLD
IIC_ByteWrite(0x57,TouchThre);        // ELE11 TOUCH THRESHOLD
IIC_ByteWrite(0x58,ReleaThre);        // ELE11 RELEASE THRESHOLD
```

```
  //Proximity threshold
  IIC_ByteWrite(0x59,Prox_TouchThre); // ELE12 TOUCH THRESHOLD
  IIC_ByteWrite(0x5A,Prox_ReleaThre); // ELE12 RELEASE THRESHOLD

  //touch and release interrupt debounce
  IIC_ByteWrite(0x5B,0x00);           // Not used for polling method, effective for INT mode.

  //AFE and filter configuration
  IIC_ByteWrite(0x5C,0x10);           // AFES=6 samples, same as AFES in 0x7B, Global CDC=16uA
  IIC_ByteWrite(0x5D,0x24);           // CT=0.5us, TDS=4samples, TDI=16ms
  IIC_ByteWrite(0x5E,0x80);           // Set baseline calibration enabled,   baseline loading 5MSB

  //Auto Configuration
  IIC_ByteWrite(0x7B,0x0B);           // AFES=6 samples, same as AFES in 0x5C
                                      // retry=2b00, no retry,
                                      // BVA=2b10, load 5MSB after AC,
                                      // ARE/ACE=2b11, auto configuration enabled
  //IIC_ByteWrite(0x7C,0x80);         // Skip charge time search, use setting in 0x5D,
                                      // OOR, AR, AC IE disabled
                                      // Not used. Possible Proximity CDC shall over 63uA
                                      // if only use 0.5uS CDT, the TGL for proximity cannot meet
                                      // Possible if manually set Register0x72=0x03
                                      // (Auto configure result) alone.
  IIC_ByteWrite(0x7D,0xc8);           // AC up limit      /C8/BD/C0/9C
  IIC_ByteWrite(0x7E,0x82);           // AC low limit     /82/7A/7C/65
  IIC_ByteWrite(0x7F,0xb4);           // AC target        /B4/AA/AC/8C  target  for /3.0V/2.8V/1.8V
  IIC_ByteWrite(0x5E,0xBC);           // Run 12 touch + proximity, CL=2b10, load 5MSB to baseline
}
```

For the detailed explanation on each registers please refer to the datasheet and relevant application note. All the registers can be read at anytime, but it's suggested to use multiple consecutive byte read to save the operation time and avoid data integrity issue. For example, we can use one line code to read out registers from 0x00 to 0x2A to get all the sensing channels output data and status information. All the registers except 0x5E and LED configuration registers cannot be written in run mode but only when set into stop mode.

After initialization and set into run mode, the MPR121 will work automatically and continuously until you set it into stop mode. MPR121 provides both touch status and touch sensing data and provides interrupt output when touch/release status changed. The host processor can read back output data and status upon interrupt, or poll the MPR121 periodically on timer overflow.

Parameter adjustment to get the most from MPR121

It's easy to understand, a stronger E-field helps to give larger touch response hence higher sensitivity. Using auto configuration, the settings of charge current and time for each channel is automatically optimized. The charge is at the maximum allowed level without any repeatedly trial on different charge time, charge current, and input capacitance level combinations.

Though there is not many fine tune work at software code level, several points need to be understand here.

1. Use of auto configuration. We choose to set the charge as high as possible to increase the touch sensitivity and hence better SNR, the expense is a little bit higher power consumption but this is still negligible.

   Normally we set the target charge level at 90% of the full available dynamic input range. The 90% is selected so that the charge is close to the input upper limited, but still allows for channel variation and power supply change during actual work.

   Note the 90% target charge is scaled with supply voltage, so you should use the highest supply voltage in your system to get the highest charge. But in system where the supply voltage is dynamically changing, the lowest working supply voltage shall be used. For example when use direct 2 cell AA dry battery power supply, the typical working supply voltage range is from 3.0V to 2.0V when battery finally drain out, then the target shall be set at 2.0V instead of 3.0V to avoid possible auto configuration out of range error.

2. Touch /release threshold can be fine tuned. You can use one set of touch/release threshold for all 12 channels, or each channel can be further fine tuned and configured to get highest sensitivity.

   Decrease the touch threshold will increase the sensitivity. The touch/release threshold should be set according to the required signal to noise ratio (SNR). Suppose we have a long term observed noise counts at 5LSB when no touch, then a touch threshold at 10LSB counts is relatively reliable. When a touch delta data change is 20 counts, this setting provides a minimum SNR of 20/10=2 and a typical value of 20/5=4. The release threshold shall be also configured in the same way.

   Depending on the actual noise level, for most design cases, keeping the touch delta counts above 10 will result a relatively good touch sensitivity and SNR. If touch threshold setting at 10 counts cannot detect a touch, then probably either the sensing pad size needs to be increased, or the cover thickness needs to be reduced.

3. Adjust the data filtering, baseline filter and touch/release debounce for noise rejection. In case there is occasional random noise, such as a quick noise spike on touch sensing pads, touch/release debounce can be used to avoid these occasionally happened noises. Figure 24 shows another example. In this case the interfered noise causes both sensing data and baseline to move upward, but the signal drops faster than the baseline which result a dead lock of touch. Figure 25 shows that the baseline rising filter can be adjusted to a slower speed to avoid this kind of noise.

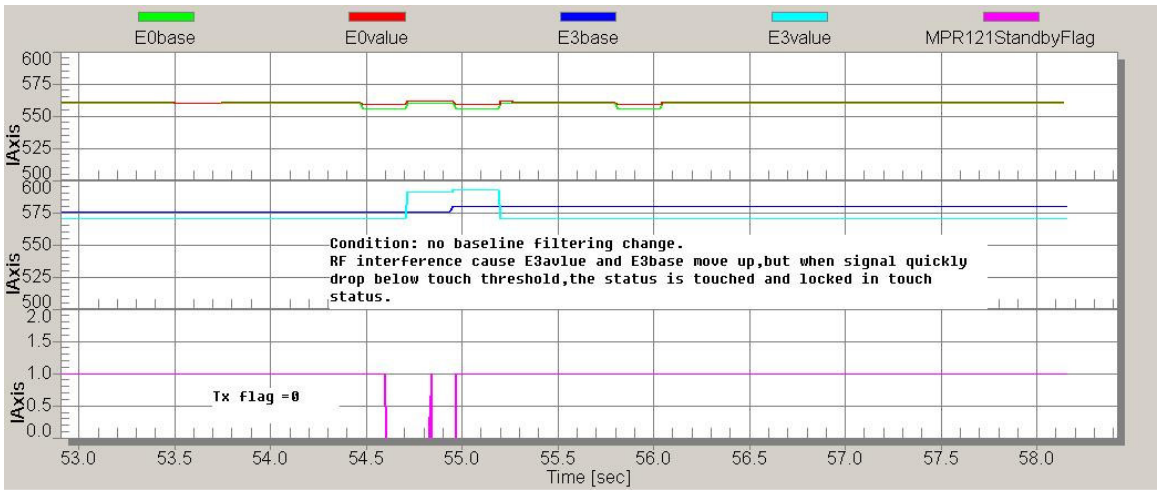


Figure 24: Output when baseline rising filter noise limit register 0x2D =0x00

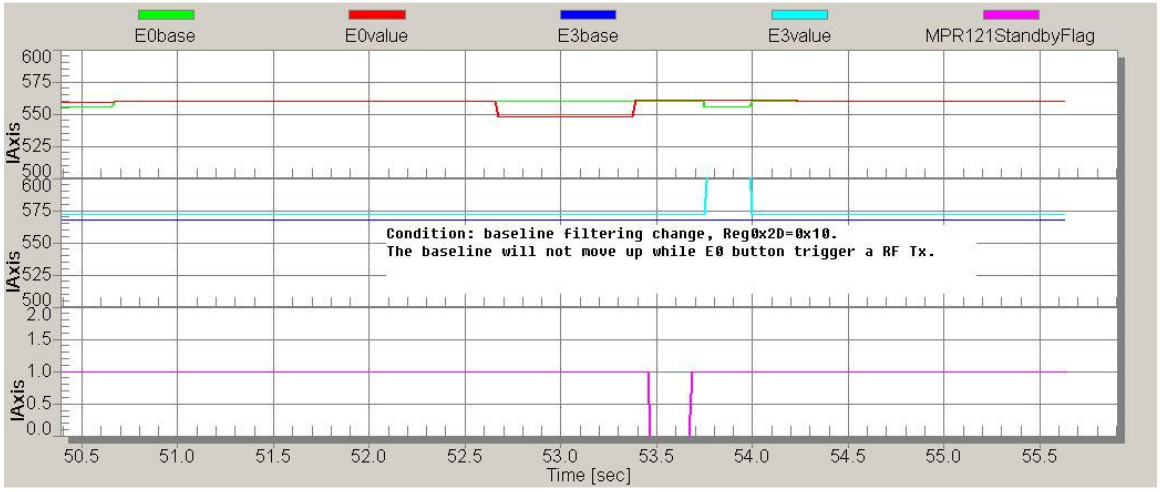


Figure 25: Output when baseline rising filter noise limit register 0x2D =0x10

4. Change the sampling interval. MPR121 has a register configurable sampling interval, which can have a big impact on average current consumption. A sampling interval at 16ms is adequate for most application requiring immediate touch response.

   For battery powered device such as a smart watch, lower power consumption is extremely important. To save the power as much as possible, the sampling interval can be set dynamically according to the user's activity level.

   For example, when there is no touch activity within period of 5 minutes, the MPR121 sampling interval can be set to 128ms by host to get an average current as low as 8uA so the whole system can go into low power hibernate mode. To wake up the system from hibernate by a touch, the user touches for a little bit long time (say approximately 1 second), then the host MCU can set the MPR121 into a higher sampling interval (for example 16ms) to detect user frequent touch activity, and the whole system goes into full power run mode.

5. Disable the proximity detection and LED function if not used, and do not enable unused electrode channel. This will help to avoid unnecessary power consumption.

6. Get touch signal output. For simple touch button application, after the MPR121 is properly configured and set to run mode, the host processor just need to read back the status register and all other registers are not cared.

   Along the touch/ release status output for each channel, MPR121 also provides 10 bit filtered output data which is a voltage represent of the sensed capacitance input on each channel, and 8 bit baseline data which is the internally filtered output of the previously mentioned 10 bit filtered data. Note the 10bit Filtered output data includes the DC component from background capacitance (the baseline value), the actual touch signal change should be calculated using below equation:

   Touch signal change =
   ((Filtered output data high byte*256+ low byte) >> 4)-Baseline Value

   This calculated data then can be used for slider, touch wheel and touchpad application interpolation calculation.

Conclusions

MPR121 is an easy to use touch sensor with intelligent touch detection engine for simple touch button design. PCB layout and structure design is the most important and fundamental part in the touch sensing design. With proper design, MPR121 can be successfully used to realize many functions like keypad matrix, slide bar, touch wheel, touchpad, finger gesture recognition, hands grip/ hold detection, and near proximity detection.