Freescale Semiconductor
Application Note

Document Number: AN4600
Rev 0, 12/2012

Designing a Touch Panel using the Xtrinsic MPR121 Capacitive Touch Sensor Controller

By Kevin Jia, Applications Engineer

1 Introduction

This application note shows how to design a touch panel around the MPR121 capacitive touch sensor controller, using minimal hardware and software.

The function of a touch panel is something like the touch panel on a laptop—when you move the finger on the touch panel, the mouse cursor onscreen follows it.

With many consumer and industrial devices, a person uses a finger on a touch panel or pad, touch wheel, or slide bar to move the cursor on a display. This type of intuitive “finger” interface makes products easy to use and greatly improves the overall customer experience.

Contents

2 Touch Panel Demo Example

The following is an example of a demo board using

- a JM badge board
- and the MPR121 evaluation kit.

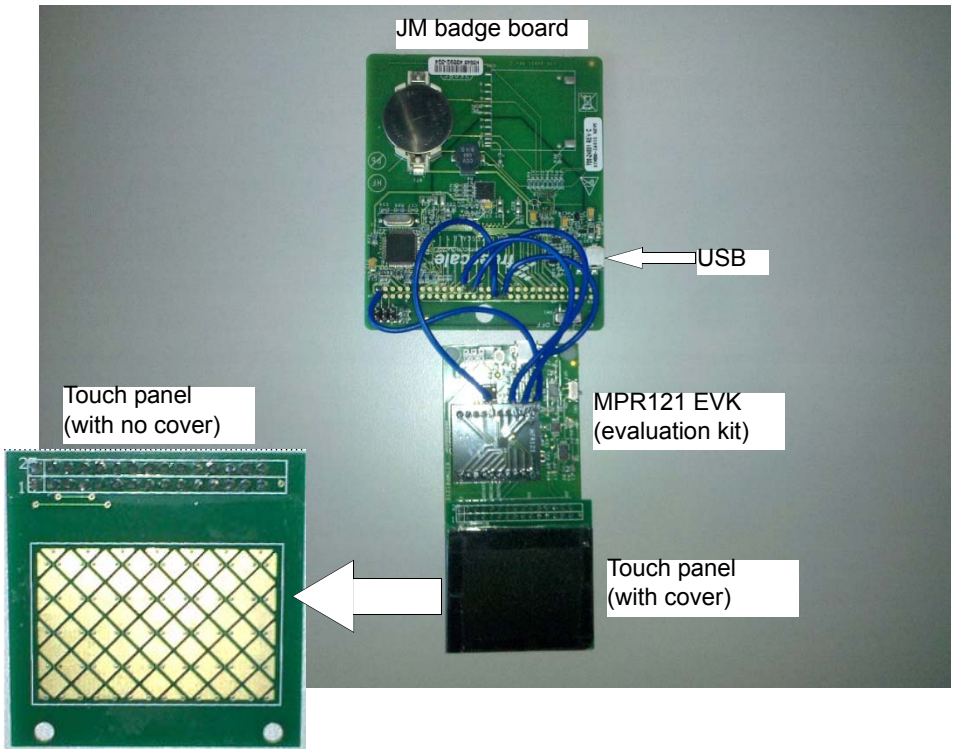


Figure 1. Touch panel demo assembly

Why choose a JM badge board as a motherboard?

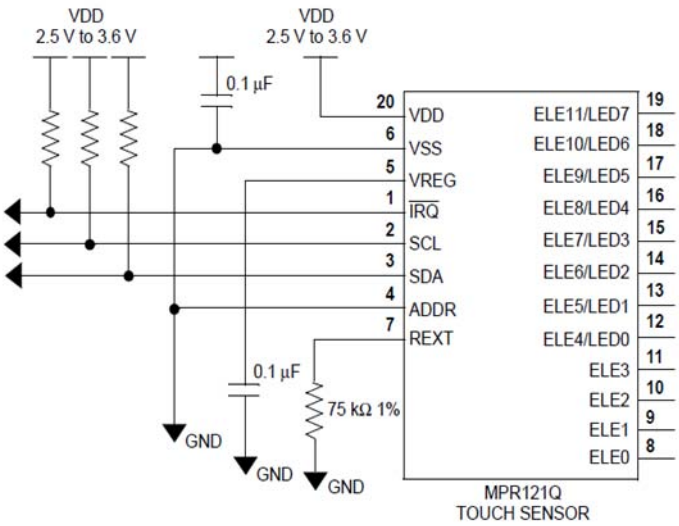
- To calculate the coordinates for a touch sensor, you want a powerful processor, like the MCF51JM128 (ColdFire v1 32-bit core).
- For convenience, you want a USB device interface with a HID mouse device driver already implemented on it, so that you don't have to implement it yourself.

However, the JM badge board does not have the MPR121 sensor on it (the JM badge board has an MPR08x, and eight electrodes are not enough for our application), so we run wires to the MPR121 EVK, to use the touch sensor on it.

Actually, you could make your own PCB to skip the operation above. If desired, we can provide the whole project code (include HID mouse device driver and touch sensor algorithm related code) for your reference. This application note provides the core algorithm code of the touch sensor; the HID device driver is not included here.

3 Controller Schematic

Using a 2.0 V to 3.6 V supply is recommended, as shown in Figure 2.



- A separate 0.1 µF decoupling ceramic capacitor on VREG to VSS is applied as a bypass cap for internal circuitry.
- Pullup resistors, typically 4.7 kΩ, are required on SDA, SCL, IRQ, and REXT.
- MPR121 has 12 input sensing channels ELE0 – ELE11 (pins 8 – 19). Pins 12 – 19 are multifunction pins.

Figure 2. MPR121 schematic using 2.0 V to 3.6 V power supply

The schematic for the sensor controller is simple—simply connect several resistors and capacitors. The sensor uses I^2C to communicate with the MCU. In this example, ELE0–ELE11 are used as touch panel inputs.

4 Touch Panel PCB Layout

For accurate coordinates, apply a matrix PCB layout method, see Figure 3.

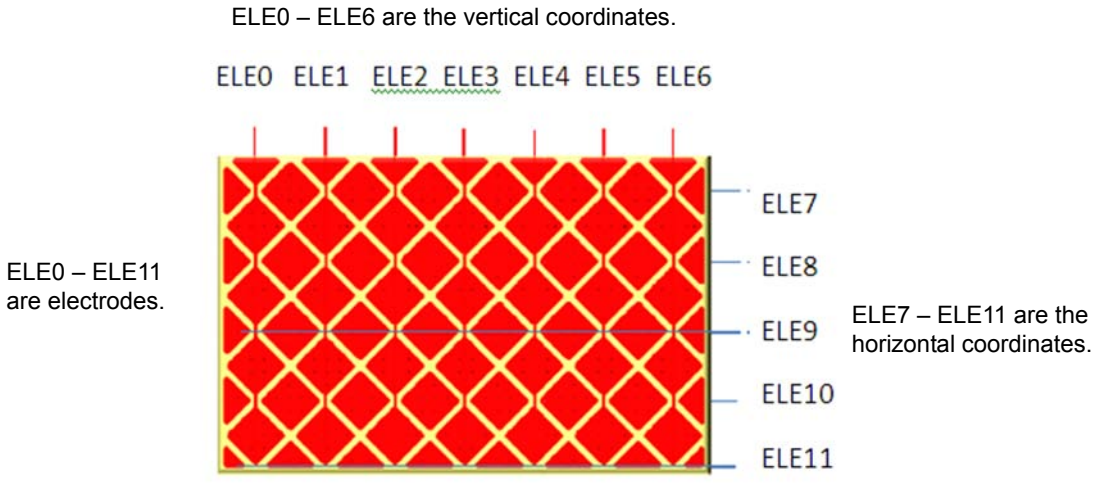


Figure 3. Track panel PCB layout

- The vertical and horizontal electrodes have no junctions.
- Blocks connect to each other in vertical or horizontal directions.
- The size of the block is determined by the expected finger size. When you touch the panel, your finger end should match four blocks for the sensor to achieve the best sensitivity and the data (vertical/horizontal) required for calculating the coordinate.
- If the block is too small, then the sensor will have low sensitivity, which can be worse if thick cover materials are used.

5 Board Layout Tips

We recommend:

- Block size of 3.5 – 4.0 mm
- 0.5 mm gap between blocks
- 1.6 mm touch panel thickness
- To reduce any interference, place the sensor circuit as far away from other circuits as possible.
- The electrode trace (the wire between an ELEx pin and the blocks) must be short and slim:
  - trace length should not exceed 7800 mil (200 mm)
  - trace width is recommended to be 5 mil (0.127 mm)
- There should be no high frequency traces near the electrode traces (such as I^2C, SPI, etc.). If you do have high frequency traces near the electrode traces, then route the traces on different layers and perpendicular to each other.
- When laying out the touch panel, route the blocks on the top layer, and route the traces on the bottom layer.
- If the board exhibits no strong RF disturbance, then a ground plane is not necessary. A ground plane will decrease the panel's sensitivity, but will improve the anti-interference capability. For safety, you can fill a 40% ground plane on the sensing area (the top layer); the bottom layer does not need a ground plane.
- If the board exhibits a strong RF disturbance, then fill a 40% ground plane on the top layer, plus a a 60% ground plane on the bottom layer.
- We recommend using ABS plastic cover materials (thickness ≤ 0.25 mm) and 3M 467MP adhesive (thickness ≤ 0.05 mm). The total thickness (plastic cover + adhesives) should be less than 0.3 mm.

6 About the Demo Software

Figure 4 shows the flowchart of a demo program; the source code follows in the next section (Section 7, "Demo Code").

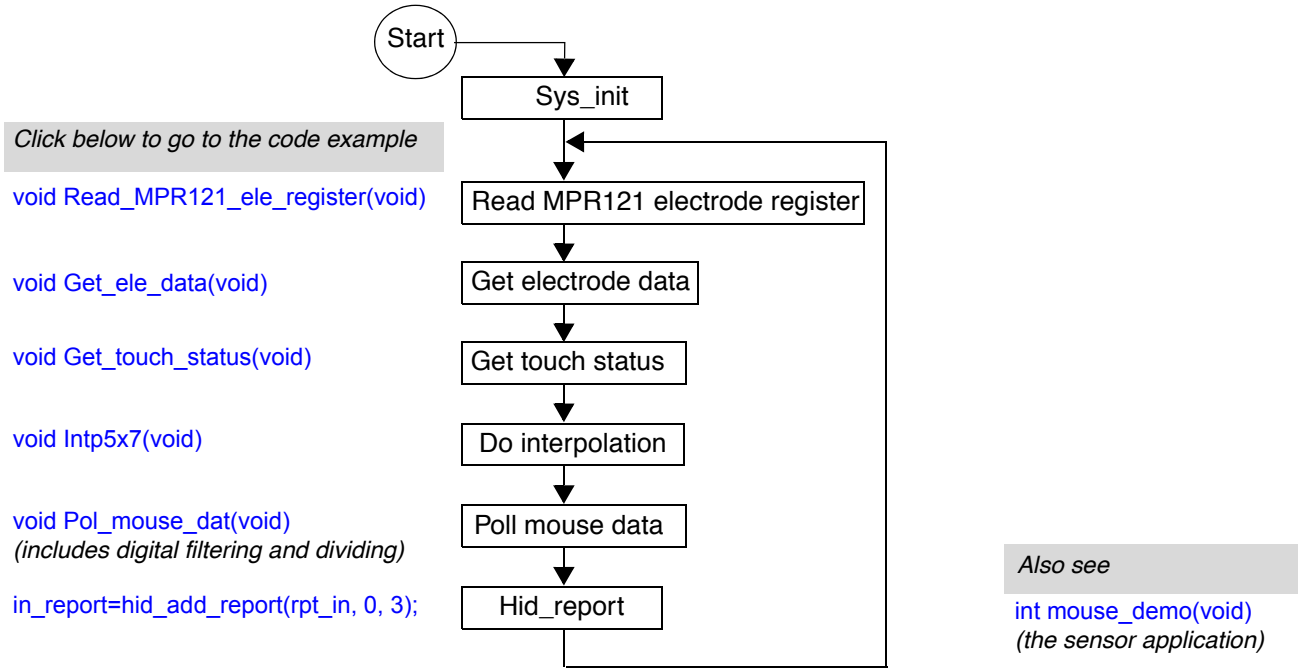


Figure 4. Program flowchart

MPR121 initialization: Before initializing the MPR121, know your application parameters:

- Touch button, slider bar, or touch panel?
- How many electrodes will be used?
- What is the required sensitivity?
- Which adhesive are you using, and what is its thickness?

This information helps to determine the sensor's configuration. For our finger tracking application (to emulate a mouse), we need a faster frequency response than a touch button, so we will use 12 electrodes. Thus, we configure the following registers:

- Electrode Configuration register (0x5E) = 0x0C (for 12 electrodes)
- AFE Configuration register (0x5C) = 0xC0 (take 34 samples, which is good for interference immunity)
- AFE Configuration2 register (0x5D) = 0x00 (SFI = 4, ESI = 1 ms, so the response time is 4 ms, which is the fastest response time)

Getting accurate coordinates using an interpolation algorithm: By using an interpolation algorithm, we can get the finger's coordinate accurately. For the horizontal coordinates, we use seven electrodes (ELE0 – ELE6). The formula is:

$$\mathbf{SumN} = 1 \times Dele(0) + 2 \times Dele(1) + 3 \times Dele(2) + \ldots\ldots + 7 \times Dele(6)$$

$$\textbf{SumD} = \text{Dele}(0) + \text{Dele}(1) + \text{Dele}(2) + \ldots\ldots + \text{Dele}(6)$$

Dele(n) (n = 0 to 6) represents the delta value (difference) of electrode-filtered data and its corresponding baseline value. After getting SumN and SumD, then:

$$\textbf{Coordinate} = \text{SumN} \div \text{SumD}$$

After getting the finger's horizontal coordinate, you use the same interpolation method to compute the vertical coordinate, using the other five electrodes (ELE7 – ELE11).

Smoothing sensor outputs using a digital filter: We need to use a digital filter to smooth the sensor's output. The first-order filter formula is:

$$FO(n) = A \times FO(n-1) + B \times SP(n)$$

$$A + B = 1$$

- FO(n) represents current filter output value.
- FO(n-1) represents previous filter output value.
- SP(n) represents current sample value.
- A and B are scale factors, which represent the weights of FO(n-1) and SP(n) in the first order filter FO(n).
  - The value of A determines the digital filter effect: the greater the value of A, the smoother the output (but this will sacrifice tracking performance, i.e., the system will respond more slowly), so you have to find a balance.
  - The value of B determines the digital filter response time: the greater the value of B, the faster the response time is.

Reducing cursor jitter: To reduce the noise that can cause the cursor to jitter when you touch the touch panel: after reading filtered 10-bit data value from the registers (0x04 – 0x1B), mask its lower two bits with 0xFFFC before passing it for consecutive calculations.

Implementing a divider in software: Some 8-bit MCUs have no hardware divider, so you may have implement the divider in software. See Section 7, “Demo Code”.

Data polling over USB: The polling time defines the frequency that the MCU polls data over the USB link. If we configure the sensor's response time to be T ms, then the polling time must satisfy 0.5 T, which is half the response time (according to Shannon Sampling Theory). This response performance will lose no frames and get the best performance.

7 Demo Code

The following code example is for a low cost, high performance mouse tracking demo, using the MPR121 device. You can copy the code directly out of this PDF to a text file.

Table 1. Demo functions

Function (click to go there)	Description
void MPR121_init(void)	MPR121 and sensor initialization
void Read_MPR121_ele_register(void)	Read registers
void Get_ele_data(void)	Get coordinate data
void Get_touch_status(void)	Get touch status
void Intp5x7(void)	Interpolate coordinates
INT32 FilterXY(INT32 prev,INT32 spl,UINT8 m)	Digital filter
INT32 GetPosXY(INT32 fz,INT32 fm)	Data divider
void Pol_mouse_dat(void)	Poll for data over USB
int mouse_demo(void)	Mouse tracking demo

Example 1. Mouse tracking demo using MPR121 device

```
#define TouchThre 10//15//30//10
#define ReleaThre 8//8//25//8

void MPR121_init(void)
{
  //Reset MPR121 if not reset correctly
  IIC_ByteWrite(0x80,0x63);  //Soft reset
  IIC_ByteWrite(0x5E,0x00);  //Stop mode

  //touch pad baseline filter
  //rising
  IIC_ByteWrite(0x2B,0x01); //0xFF// MAX HALF DELTA Rising
  IIC_ByteWrite(0x2C,0x01); //0xFF// NOISE HALF DELTA Rising
  IIC_ByteWrite(0x2D,0x00); // //0 NOISE COUNT LIMIT Rising
  IIC_ByteWrite(0x2E,0x00); // DELAY LIMIT Rising
  //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(0x32,0x02); // //2//DELAY LIMIT Falling
  //touched
  IIC_ByteWrite(0x33,0x00); // Noise half delta touched
  IIC_ByteWrite(0x34,0x00); // Noise counts touched
  IIC_ByteWrite(0x35,0x00); //Filter delay touched

  //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

  //AFE configuration
  IIC_ByteWrite(0x5D,0x00);
  IIC_ByteWrite(0x5C,0xC0);

  //Auto configuration
  IIC_ByteWrite(0x7B,0xCB);
  IIC_ByteWrite(0x7D,0xE4);
  IIC_ByteWrite(0x7E,0x94);
  IIC_ByteWrite(0x7F,0xCD);
  IIC_ByteWrite(0x5E,0x0C);
}

void Read_MPR121_ele_register(void)
{
  // read the register before 0x2B
  reading = u8VI2CRead(iBatAddress, (UINT8 *)(&readingArray), 0x2B);
}

void Get_ele_data(void)
{
    UINT8 i;
    UINT16 tmp_sig,tmp_bas;
    for (i=0; i<13; i++)
    {
tmp_sig=(((UINT16)readingArray[0x04+2*i])|(((UINT16)readingArray[0x04+1+2*i])<<8))&0xFFFC;
      tmp_bas=((UINT16)readingArray[0x1e + i])<<2;
      ele_delta[i]=abs((INT16)(tmp_sig-tmp_bas));
    }
}

void Get_touch_status(void)
{
  CurrTouchStatus.Reg0=readingArray[0x00];
```

```
  CurrTouchStatus.Reg1=readingArray[0x01];

  if (((CurrTouchStatus.Reg0 & 0xff) != 0) || ((CurrTouchStatus.Reg1 & 0x1f)!= 0) )
  {
      CurrTouchStatus.Touched=1;
  }else
  {
      CurrTouchStatus.Touched=0;
  }
}

void Intp5x7(void)
{
  UINT8 i;
  SampSumX=0;
  SampSX=0;
  SampSumY=0;
  SampSY=0;
  for(i=0;i<12;i++)
  {
    if(i<7)
    {
      SampSumX+=(i+1)*ele_delta[i];
      SampSX+=ele_delta[i];
    }
    else
    {
      SampSumY+=(i-6)*ele_delta[i];
      SampSY+=ele_delta[i];
    }
  }
}

INT32 FilterXY(INT32 prev,INT32 spl,UINT8 m)
{
  //X=6/8*X'+2/8*X''
  if(m==1)       return prev-(prev>>2)+(spl>>2);
  //  4/8    4/8
  else if(m==2)  return (prev>>1)+(spl>>1);
  //  5/8    3/8
  else if(m==3)  return prev-(prev>>1)+(prev>>3)+(spl>>2)+(spl>>3);
  //  7/8    1/8
  else if(m==4)  return prev-(prev>>3)+(spl>>3);
}

// divider
INT32 GetPosXY(INT32 fz,INT32 fm)
{
   UINT8 i;
   UINT32 w=0;
   UINT16 q=0,b=0;
   UINT8  s=0,g=0;
   if(fz==0||fm==0)  return 0;
   for(i=0;i<5;i++)
   {
      if(fz<fm)
      {
```

```
            if(i==0)   w=0;
            if(i==1)   q=0;
            if(i==2)   b=0;
            if(i==3)   s=0;
            if(i==4)   g=0;

            fz=(fz<<3)+(fz<<1);
            continue;
         }
         while(1)
         {
            fz-=fm;
            if(i==0)  ++w;
            if(i==1)  ++q;
            if(i==2)  ++b;
            if(i==3)  ++s;
            if(i==4)  ++g;

            if(fz<fm)
            {
               fz=(fz<<3)+(fz<<1);
               break;
            }
         }
      }
      // y.yyyy*10000
      w=(w<<13)+(w<<10)+(w<<9)+(w<<8)+(w<<4);
      q=(q<<9)+(q<<8)+(q<<7)+(q<<6)+(q<<5)+(q<<3);
      b=(b<<6)+(b<<5)+(b<<2);
      s=(s<<3)+(s<<1);
      return w+q+b+s+g;
}

void Pol_mouse_dat(void)
{
  if(CurrTouchStatus.Touched==1)      // pressed
  {
      // get CurSumX,CurSumY,SX,SY
      CurSumX = FilterXY(PrevSumX,SampSumX,1);
      CurSX   = FilterXY(PrevSX,SampSX,1);
      CurSumY = FilterXY(PrevSumY,SampSumY,1);
      CurSY   = FilterXY(PrevSY,SampSY,1);
      CurPosX = GetPosXY(CurSumX,CurSX);
      CurPosY = GetPosXY(CurSumY,CurSY);
      // G Filter
      #if FILTER==G
      CurPosX = FilterXY(PrevPosX,CurPosX,2);
      CurPosY = FilterXY(PrevPosY,CurPosY,2);
      CurDX = CurPosX-PrevPosX;
      CurDY = CurPosY-PrevPosY;
      #endif
      // D Filter
      #if FILTER==D
      SamDX = CurPosX-PrevPosX;
      SamDY = CurPosY-PrevPosY;
      CurDX = FilterXY(PrevDX,SamDX,1);
      CurDY = FilterXY(PrevDY,SamDY,1);
```

```
 #endif
 // A Filter
 #if FILTER==A
 CurPosX = FilterXY(PrevPosX,CurPosX,3);
 CurPosY = FilterXY(PrevPosY,CurPosY,3);
 SamDX = CurPosX-PrevPosX;
 SamDY = CurPosY-PrevPosY;
 CurDX = FilterXY(PrevDX,SamDX,3);
 CurDY = FilterXY(PrevDY,SamDY,3);
 #endif
if(PreTouchStatus.Touched==0)  // fast track when finger just pressed
{
   #if FILTER==D||FILTER==A
   SndFlg=1;
   #endif
   PrevSumX=SampSumX;
   PrevSX=SampSX;
   PrevSumY=SampSumY;
   PrevSY=SampSY;
   PrevPosX=GetPosXY(PrevSumX,PrevSX);
   PrevPosY=GetPosXY(PrevSumY,PrevSY);
}
else        // when finger pressed for some time
{
  if(FstFlg++>=3)  // ingored first three samples for avoid cursor's tremble
  {
    FstFlg=3;
    // for debug
    /*
    if((CurSumX!=PrevSumX)||(CurSX!=PrevSX)||(CurSumY!=PrevSumY)||(CurSY!=PrevSY))
    {
      SendChar((char)(CurSumX>>8));
      SendChar((char)(CurSumX&0x00ff));
      SendChar((char)(CurSX>>8));
      SendChar((char)(CurSX&0x00ff));
      SendChar((char)(CurSumY>>8));
      SendChar((char)(CurSumY&0x00ff));
      SendChar((char)(CurSY>>8));
      SendChar((char)(CurSY&0x00ff));
      SendChar('\r');
      SendChar('\n');
    }        */
    if((CurPosX!=PrevPosX)||(CurPosY!=PrevPosY))
    {
      // for debug
      /*
      SendChar((char)(CurPosX>>24));
      SendChar((char)((CurPosX&0x00ff0000)>>16));
      SendChar((char)((CurPosX&0x0000ff00)>>8));
      SendChar((char)(CurPosX&0x000000ff));

      SendChar((char)(CurPosY>>24));
      SendChar((char)((CurPosY&0x00ff0000)>>16));
      SendChar((char)((CurPosY&0x0000ff00)>>8));
      SendChar((char)(CurPosY&0x000000ff));
      SendChar('\r');
      SendChar('\n');   */
```

```
// delta counts < threshold?ingored
if(((CurDX<S_X)&&(CurDX>=0))||((CurDX>-S_X)&&(CurDX<=0)))
{
   // send 0
   DX=0;
}
else
{
   // every 50 delta counts corresponds to 1 amplify factor
   // amplify factors could be adjusted by cursor's move performance
   if(CurDX>0)
   {
      if(CurDX>=S_X&&CurDX<S_X+50)                DX=2;    // 2
      else if(CurDX>=S_X+50&&CurDX<S_X+100)       DX=2;    // 2
      else if(CurDX>=S_X+100&&CurDX<S_X+150)      DX=2;    // 2
      else if(CurDX>=S_X+150&&CurDX<S_X+200)      DX=2;    // 2
      else if(CurDX>=S_X+200&&CurDX<S_X+250)      DX=3;    // 3
      else if(CurDX>=S_X+250&&CurDX<S_X+300)      DX=3;    // 3
      else if(CurDX>=S_X+300&&CurDX<S_X+350)      DX=3;    // 3
      else if(CurDX>=S_X+350&&CurDX<S_X+400)      DX=3;    // 3
      else if(CurDX>=S_X+400&&CurDX<S_X+450)      DX=4;    // 4
      else if(CurDX>=S_X+450&&CurDX<S_X+500)      DX=4;    // 4
      else if(CurDX>=S_X+500&&CurDX<S_X+550)      DX=4;    // 4
      else if(CurDX>=S_X+550&&CurDX<S_X+600)      DX=4;    // 5
      else if(CurDX>=S_X+600&&CurDX<S_X+650)      DX=5;    // 5
      else if(CurDX>=S_X+650&&CurDX<S_X+700)      DX=5;    // 5
      else if(CurDX>=S_X+700&&CurDX<S_X+750)      DX=5;    // 6
      else if(CurDX>=S_X+750&&CurDX<S_X+800)      DX=5;    // 6
      else if(CurDX>=S_X+800&&CurDX<S_X+850)      DX=6;    // 6
      else if(CurDX>=S_X+850&&CurDX<S_X+900)      DX=6;    // 7
      else if(CurDX>=S_X+900&&CurDX<S_X+950)      DX=6;    // 7
      else if(CurDX>=S_X+950&&CurDX<S_X+1000)     DX=6;    // 8
      else if(CurDX>=S_X+1000&&CurDX<S_X+1050)    DX=7;    // 8
      else if(CurDX>=S_X+1050&&CurDX<S_X+1100)    DX=7;    // 9
      else if(CurDX>=S_X+1100&&CurDX<S_X+1150)    DX=7;   // 10
      else if(CurDX>=S_X+1150&&CurDX<S_X+1200)    DX=8;   // 11
      else if(CurDX>=S_X+1200&&CurDX<S_X+1250)    DX=8;   // 12
      else if(CurDX>=S_X+1250&&CurDX<S_X+1300)    DX=9;   // 13
      else if(CurDX>=S_X+1300&&CurDX<S_X+1350)    DX=9;   // 14
      else if(CurDX>=S_X+1350&&CurDX<S_X+1400)    DX=10;   // 15
      else if(CurDX>=S_X+1400&&CurDX<S_X+1450)    DX=10;   // 16
      else if(CurDX>=S_X+1450&&CurDX<S_X+1500)    DX=11;   // 17
      else if(CurDX>=S_X+1500&&CurDX<S_X+1550)    DX=11;   // 18
      else if(CurDX>=S_X+1550&&CurDX<S_X+1600)    DX=12;   // 19
      else if(CurDX>=S_X+1600&&CurDX<S_X+1650)    DX=13;   // 20
      else if(CurDX>=S_X+1650&&CurDX<S_X+1700)    DX=14;   // 21
      else if(CurDX>=S_X+1700&&CurDX<S_X+1750)    DX=15;   // 22
      else                                        DX=18;   // 25
   }
   else
   {
      if(CurDX+S_X<=0&&CurDX+S_X+50>0)               DX=-2;   // -2    2
      else if(CurDX+S_X+50<=0&&CurDX+S_X+100>0)      DX=-2;   // -2    2
      else if(CurDX+S_X+100<=0&&CurDX+S_X+150>0)     DX=-2;   // -2    2
      else if(CurDX+S_X+150<=0&&CurDX+S_X+200>0)     DX=-2;   // -3    3
      else if(CurDX+S_X+200<=0&&CurDX+S_X+250>0)     DX=-3;   // -3    3
```

```
        else if(CurDX+S_X+250<=0&&CurDX+S_X+300>0)      DX=-3;   // -3    3
        else if(CurDX+S_X+300<=0&&CurDX+S_X+350>0)      DX=-3;   // -3    3
        else if(CurDX+S_X+350<=0&&CurDX+S_X+400>0)      DX=-3;   // -3    4
        else if(CurDX+S_X+400<=0&&CurDX+S_X+450>0)      DX=-4;   // -4    4
        else if(CurDX+S_X+450<=0&&CurDX+S_X+500>0)      DX=-4;   // -4    4
        else if(CurDX+S_X+500<=0&&CurDX+S_X+550>0)      DX=-4;   // -4    4
        else if(CurDX+S_X+550<=0&&CurDX+S_X+600>0)      DX=-4;   // -5
        else if(CurDX+S_X+600<=0&&CurDX+S_X+650>0)      DX=-5;   // -5
        else if(CurDX+S_X+650<=0&&CurDX+S_X+700>0)      DX=-5;   // -5
        else if(CurDX+S_X+700<=0&&CurDX+S_X+750>0)      DX=-5;   // -6
        else if(CurDX+S_X+750<=0&&CurDX+S_X+800>0)      DX=-5;   // -6
        else if(CurDX+S_X+800<=0&&CurDX+S_X+850>0)      DX=-6;   // -6
        else if(CurDX+S_X+850<=0&&CurDX+S_X+900>0)      DX=-6;   // -7
        else if(CurDX+S_X+900<=0&&CurDX+S_X+950>0)      DX=-6;   // -7
        else if(CurDX+S_X+950<=0&&CurDX+S_X+1000>0)     DX=-6;   // -8
        else if(CurDX+S_X+1000<=0&&CurDX+S_X+1050>0)    DX=-7;   // -8
        else if(CurDX+S_X+1050<=0&&CurDX+S_X+1100>0)    DX=-7;   // -9
        else if(CurDX+S_X+1100<=0&&CurDX+S_X+1150>0)    DX=-7;  // -10
        else if(CurDX+S_X+1150<=0&&CurDX+S_X+1200>0)    DX=-8;  // -11
        else if(CurDX+S_X+1200<=0&&CurDX+S_X+1250>0)    DX=-8;  // -12
        else if(CurDX+S_X+1250<=0&&CurDX+S_X+1300>0)    DX=-9;  // -13
        else if(CurDX+S_X+1300<=0&&CurDX+S_X+1350>0)    DX=-9;  // -14
        else if(CurDX+S_X+1350<=0&&CurDX+S_X+1400>0)    DX=-10;  // -15
        else if(CurDX+S_X+1400<=0&&CurDX+S_X+1450>0)    DX=-10;  // -16
        else if(CurDX+S_X+1450<=0&&CurDX+S_X+1500>0)    DX=-11;  // -17
        else if(CurDX+S_X+1500<=0&&CurDX+S_X+1550>0)    DX=-11;  // -18
        else if(CurDX+S_X+1550<=0&&CurDX+S_X+1600>0)    DX=-12;  // -19
        else if(CurDX+S_X+1600<=0&&CurDX+S_X+1650>0)    DX=-13;  // -20
        else if(CurDX+S_X+1650<=0&&CurDX+S_X+1700>0)    DX=-14;  // -21
        else if(CurDX+S_X+1700<=0&&CurDX+S_X+1750>0)    DX=-15;  // -22
        else                                            DX=-18;  // -25
    }
}
if(((CurDY<S_Y)&&(CurDY>=0))||((CurDY>-S_Y)&&(CurDY<=0)))
{
    // send 0
    DY=0;
}
else
{
    if(CurDY>0)
    {
        if(CurDY>=S_Y&&CurDY<S_Y+50)                DY=2;      // 2
        else if(CurDY>=S_Y+50&&CurDY<S_Y+100)       DY=2;      // 2
        else if(CurDY>=S_Y+100&&CurDY<S_Y+150)      DY=2;      // 2
        else if(CurDY>=S_Y+150&&CurDY<S_Y+200)      DY=2;      // 2
        else if(CurDY>=S_Y+200&&CurDY<S_Y+250)      DY=3;      // 3
        else if(CurDY>=S_Y+250&&CurDY<S_Y+300)      DY=3;      // 3
        else if(CurDY>=S_Y+300&&CurDY<S_Y+350)      DY=3;      // 3
        else if(CurDY>=S_Y+350&&CurDY<S_Y+400)      DY=3;      // 3
        else if(CurDY>=S_Y+400&&CurDY<S_Y+450)      DY=4;      // 4
        else if(CurDY>=S_Y+450&&CurDY<S_Y+500)      DY=4;      // 4
        else if(CurDY>=S_Y+500&&CurDY<S_Y+550)      DY=4;      // 4
        else if(CurDY>=S_Y+550&&CurDY<S_Y+600)      DY=4;      // 5
        else if(CurDY>=S_Y+600&&CurDY<S_Y+650)      DY=5;      // 5
        else if(CurDY>=S_Y+650&&CurDY<S_Y+700)      DY=5;      // 5
        else if(CurDY>=S_Y+700&&CurDY<S_Y+750)      DY=5;      // 6
```

```
    else if(CurDY>=S_Y+750&&CurDY<S_Y+800)      DY=5;      // 6
    else if(CurDY>=S_Y+800&&CurDY<S_Y+850)      DY=6;      // 6
    else if(CurDY>=S_Y+850&&CurDY<S_Y+900)      DY=6;      // 7
    else if(CurDY>=S_Y+900&&CurDY<S_Y+950)      DY=6;     // 7
    else if(CurDY>=S_Y+950&&CurDY<S_Y+1000)     DY=6;     // 8
    else if(CurDY>=S_Y+1000&&CurDY<S_Y+1050)    DY=7;     // 8
    else if(CurDY>=S_Y+1050&&CurDY<S_Y+1100)    DY=7;     // 9
    else if(CurDY>=S_Y+1100&&CurDY<S_Y+1150)    DY=7;     // 10
    else if(CurDY>=S_Y+1150&&CurDY<S_Y+1200)    DY=8;     // 11
    else if(CurDY>=S_Y+1200&&CurDY<S_Y+1250)    DY=8;     // 12
    else if(CurDY>=S_Y+1250&&CurDY<S_Y+1300)    DY=9;     // 13
    else if(CurDY>=S_Y+1300&&CurDY<S_Y+1350)    DY=9;     // 14
    else if(CurDY>=S_Y+1350&&CurDY<S_Y+1400)    DY=10;     // 15
    else if(CurDY>=S_Y+1400&&CurDY<S_Y+1450)    DY=10;     // 16
    else if(CurDY>=S_Y+1450&&CurDY<S_Y+1500)    DY=11;     // 17
    else if(CurDY>=S_Y+1500&&CurDY<S_Y+1550)    DY=11;     // 18
    else if(CurDY>=S_Y+1550&&CurDY<S_Y+1600)    DY=12;     // 19
    else if(CurDY>=S_Y+1600&&CurDY<S_Y+1650)    DY=13;     // 20
    else if(CurDY>=S_Y+1650&&CurDY<S_Y+1700)    DY=14;     // 21
    else if(CurDY>=S_Y+1700&&CurDY<S_Y+1750)    DY=15;     // 22
    else                                        DY=18;     // 25
}
else
{
    if(CurDY+S_Y<=0&&CurDY+S_Y+50>0)                DY=-2;     // -2
    else if(CurDY+S_Y+50<=0&&CurDY+S_Y+100>0)       DY=-2;     // -2
    else if(CurDY+S_Y+100<=0&&CurDY+S_Y+150>0)      DY=-2;     // -2
    else if(CurDY+S_Y+150<=0&&CurDY+S_Y+200>0)      DY=-2;     // -2
    else if(CurDY+S_Y+200<=0&&CurDY+S_Y+250>0)      DY=-3;     // -3
    else if(CurDY+S_Y+250<=0&&CurDY+S_Y+300>0)      DY=-3;     // -3
    else if(CurDY+S_Y+300<=0&&CurDY+S_Y+350>0)      DY=-3;     // -3
    else if(CurDY+S_Y+350<=0&&CurDY+S_Y+400>0)      DY=-3;     // -3
    else if(CurDY+S_Y+400<=0&&CurDY+S_Y+450>0)      DY=-4;     // -4
    else if(CurDY+S_Y+450<=0&&CurDY+S_Y+500>0)      DY=-4;     // -4
    else if(CurDY+S_Y+500<=0&&CurDY+S_Y+550>0)      DY=-4;     // -4
    else if(CurDY+S_Y+550<=0&&CurDY+S_Y+600>0)      DY=-4;     // -5
    else if(CurDY+S_Y+600<=0&&CurDY+S_Y+650>0)      DY=-5;     // -5
    else if(CurDY+S_Y+650<=0&&CurDY+S_Y+700>0)      DY=-5;     // -5
    else if(CurDY+S_Y+700<=0&&CurDY+S_Y+750>0)      DY=-5;     // -6
    else if(CurDY+S_Y+750<=0&&CurDY+S_Y+800>0)      DY=-5;     // -6
    else if(CurDY+S_Y+800<=0&&CurDY+S_Y+850>0)      DY=-6;     // -6
    else if(CurDY+S_Y+850<=0&&CurDY+S_Y+900>0)      DY=-6;     // -7
    else if(CurDY+S_Y+900<=0&&CurDY+S_Y+950>0)      DY=-6;    // -7
    else if(CurDY+S_Y+950<=0&&CurDY+S_Y+1000>0)     DY=-6;    // -8
    else if(CurDY+S_Y+1000<=0&&CurDY+S_Y+1050>0)    DY=-7;    // -8
    else if(CurDY+S_Y+1050<=0&&CurDY+S_Y+1100>0)    DY=-7;    // -9
    else if(CurDY+S_Y+1100<=0&&CurDY+S_Y+1150>0)    DY=-7;    // -10
    else if(CurDY+S_Y+1150<=0&&CurDY+S_Y+1200>0)    DY=-8;    // -11
    else if(CurDY+S_Y+1200<=0&&CurDY+S_Y+1250>0)    DY=-8;    // -12
    else if(CurDY+S_Y+1250<=0&&CurDY+S_Y+1300>0)    DY=-9;    // -13
    else if(CurDY+S_Y+1300<=0&&CurDY+S_Y+1350>0)    DY=-9;    // -14
    else if(CurDY+S_Y+1350<=0&&CurDY+S_Y+1400>0)    DY=-10;    // -15
    else if(CurDY+S_Y+1400<=0&&CurDY+S_Y+1450>0)    DY=-10;    // -16
    else if(CurDY+S_Y+1450<=0&&CurDY+S_Y+1500>0)    DY=-11;    // -17
    else if(CurDY+S_Y+1500<=0&&CurDY+S_Y+1550>0)    DY=-11;    // -18
    else if(CurDY+S_Y+1550<=0&&CurDY+S_Y+1600>0)    DY=-12;    // -19
    else if(CurDY+S_Y+1600<=0&&CurDY+S_Y+1650>0)    DY=-13;    // -20
```

```
                else if(CurDY+S_Y+1650<=0&&CurDY+S_Y+1700>0)    DY=-14;    // -21
                else if(CurDY+S_Y+1700<=0&&CurDY+S_Y+1750>0)    DY=-15;    // -22
                else                                            DY=-18;    // -25
              }
            }
          }
          else
          {
             // send 0
             DX=0;
             DY=0;
          }
        }
        else
        {
           //send 0
           DX=0;
           DY=0;
        }
        PrevSumX=CurSumX;
        PrevSX=CurSX;
        PrevSumY=CurSumY;
        PrevSY=CurSY;
        PrevPosX=CurPosX;
        PrevPosY=CurPosY;
        #if FILTER==D||FILTER==A
        if(SndFlg==1)
        {
           SndFlg=0;
           PrevDX=SamDX;
           PrevDY=SamDY;
        }
        else
        {
           PrevDX=CurDX;
           PrevDY=CurDY;
        }
        #endif
      }
      PreTouchStatus.Touched=1;
  }
  else          // unpressed
  {
     FstFlg=0;
     PreTouchStatus.Touched=0;
     DX=0;
     DY=0;
  }
}

int mouse_demo(void)
{
  int x=0;
  int y=0;
  dword xinit,yinit;
  const int delta=1;
  hcc_u8 in_report;
```

```
  PTAD=0x01;
  PTED=~(0x01);
  set_mode(dm_mouse);
  /* Value for accelerometer when held level */
  xinit=32000;
  yinit=32000;
  HID_init(0, 0);
  in_report=hid_add_report(rpt_in, 0, 3);
  LED_GRN = OFF;
  while(!device_stp)
  {
    LED_RED = ON;
    Read_MPR121_ele_register();  // read reg 0x00 - 0x2b value to readingArray[]
    LED_RED = OFF;
    Get_ele_data();              // get signal, baseline, delta
    Get_touch_status();      // read touch status reg 0x00, 0x01, determin touched or not
    Intp5x7();                        // interpolation algorithm
    Pol_mouse_dat();
    hid_process();
    if (!hid_report_pending(in_report))
    {
      DIR_REP_X(hid_report) = (hcc_u8)(DX);
      DIR_REP_Y(hid_report) = (hcc_u8)(DY);
      hid_write_report(in_report, (hcc_u8*)hid_report);
    }
    busy_wait();
    {
      long stackSize = stack_size(0x88);
    }
  }
  return(0);
}
```

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: AN4600
Rev 0
12/2012