AN2807

STMPE811 reference code for touchscreen controller operation

Introduction

The purpose of this application note is to provide the reference code for the STMPE811 touchscreen controller operation.

The first chapter describes how to initilialize the STMPE811 device while the second part gives two practical methods to handle the data coming from the device.

For additional information, please refer to the STMPE811 datasheet.

1 Device initialization

In the startup of the system, the initialization of each hardware is performed.

For STMPE811 operating as touchscreen controller, the registers listed below need to be initialized:

1. SYS_CTRL_2 register (0x04), to turn on the necessary blocks.
   To turn on the touchscreen controller block, we need to turn on both the touchscreen controller and ADC blocks by writing 0x0C to this register.
2. INT_EN register (0x0A), to enable the required interrupt source.
   For the touchscreen function, only three interrupt sources need to be enabled: TOUCH_DETECT and FIFO_TH.
   - Write 0x03 to this register for group data reading.
   - Write 0x02 to this register for single data reading.
3. ADC_CTRL1 (0x20), to select sample time, bit number and ADC reference.
   a) Sample time: it is recommended to select 80 clock cycles for ADC conversion in order to get reasonably fast data rate and accurate reading.
   b) ADC bit number: 12-bit ADC is used in this document.
   c) Reference selection: in this document, internal reference is used.
4. ADC_CTRL2 (0x21), to select the ADC clock speed.
   3.25 MHz is the recommended value for touchscreen operation, although the device can be pushed to operate at 6.5 MHz. Write 0x01 for this setting.
5. GPIO_AF (0x17), to program the I/O pins to the correct function.
   By default, Power on reset (POR) sets the device in primary function (ADC/ touchscreen controller application), but it is a good practice to make sure the I/Os are in the correct mode. Write 0x00 to set all the I/O in ADC/touchscreen controller mode.
6. TSC_CFG (0x41), to set Average, Touch Detect Delay and Settling Time setting.
   a) Touch Detect dDelay and Settling Time: the setting for these two features depend on the filter capacitor used in the touch screen pins (X+, Y+, X-, Y-). For small screens, less than 6”, a filter capacitor of 1-2 nF is recommended to be placed in each of the touchscreen pins, while for bigger screens, 5-10 nF capacitors can be used.
   – In case of 1-2 nF capacitors:
   500 uS settling time and touch detect delay are recommended.
   – In case of 5-10 nF capacitors:
   1 mS settling time and touch detect delay are recommended.

   In this document, 2 nF filter is assumed. Hence 500 uS settling time and 500uS Touch Detect Delay are used.
   b) Average control: average of 2 or 4 is recommended. In this document, 4 point averaging is used. Write 0x9A to this register for the above mentioned setting.

7. FIFO_TH (0x4A), to set FIFO threshold level.
   FIFO threshold defines the minimum number of samples in the FIFO memory before it gives interrupt. The setting depends on the interrupt handling architecture to be used. Refer to *Section 1.1*. In overall there are two options for data fetching from FIFO memory:

- Single point reading.
  Read a single point data in each I^2C transaction to read data from FIFO memory. In this case the FIFO threshold is set to 1. Hence, in every data acquired, the device will issue FIFO threshold interrupt to inform host to read the data. Write 0x01 for this setting.
- Group points reading.
  Read a group of data in each I^2C transaction to read data from FIFO memory. In this case, the FIFO threshold may be set to recommended value of 5. When the number of data in FIFO memory is more than 5, it will issue FIFO threshold interrupt. Write 0x05 for this setting.

8. FIFO_CTRL_STA (0x4B), to reset FIFO.
   Write 0x01 to clear the FIFO memory content.
   Write 0x00 to put the FIFO back into operation mode.
9. TSC_FRACT_Z (0x56), to set the data format for Z value.
   Write 0x07 to set to recommended value (7 fractional part and 1 whole part).
10. TSC_I_DRIVE (0x58), to set the driving capability of the device for touchscreen controller pins.
    Two setting may be chosen, 20 mA or 50 mA maximum current setting.
    In this document, 50 mA setting is used. Write 0x01 for this setting
11. TSC_CTRL (0x40), to set tracking index, set touchscreen controller operation mode and enable the TSC.
    – No tracking Index will be used.
    – The device operate at X, Y, Z mode

Write 0x01 to write the setting and enable the touch screen controller.

12. INT_STA (0x0B), to clear all the interrupt status.
    Write 0xFF to clear all the interrupt status
13. INT_CTRL (0x09), to set interrupt mode and enable interrupt.
    Write 0x01 to enable the interrupt.

1.1 Initialization summary

1. Write: register Add = 0x04, data = 0x0C
2. Write: register Add = 0x0A, data = 0x07
3. Write: register Add = 0x20, data = 0x49
   delay, 2mS
4. Write: register Add = 0x21, data = 0x01
5. Write: register Add = 0x17, data = 0x00
6. Write: register Add = 0x41, data = 0x9A (2 nF filter capacitor)
   - a. Write: register Add = 0x4A, data = 0x01 (for single point reading)
   - b. Write: register Add = 0x4A, data = 0x05 (for group points reading)
7. Write: register Add = 0x4B, data = 0x01
8. Write: register Add = 0x4B, data = 0x00
9. Write: register Add = 0x56, data = 0x07
10. Write: register Add = 0x58, data = 0x01
11. Write: register Add = 0x40, data = 0x01
12. Write: register Add = 0x0B, data = 0xFF
13. Write: register Add = 0x09, data = 0x01

2 Interrupt service routine (ISR)

There are two methods to handle the data coming from STMPE811.

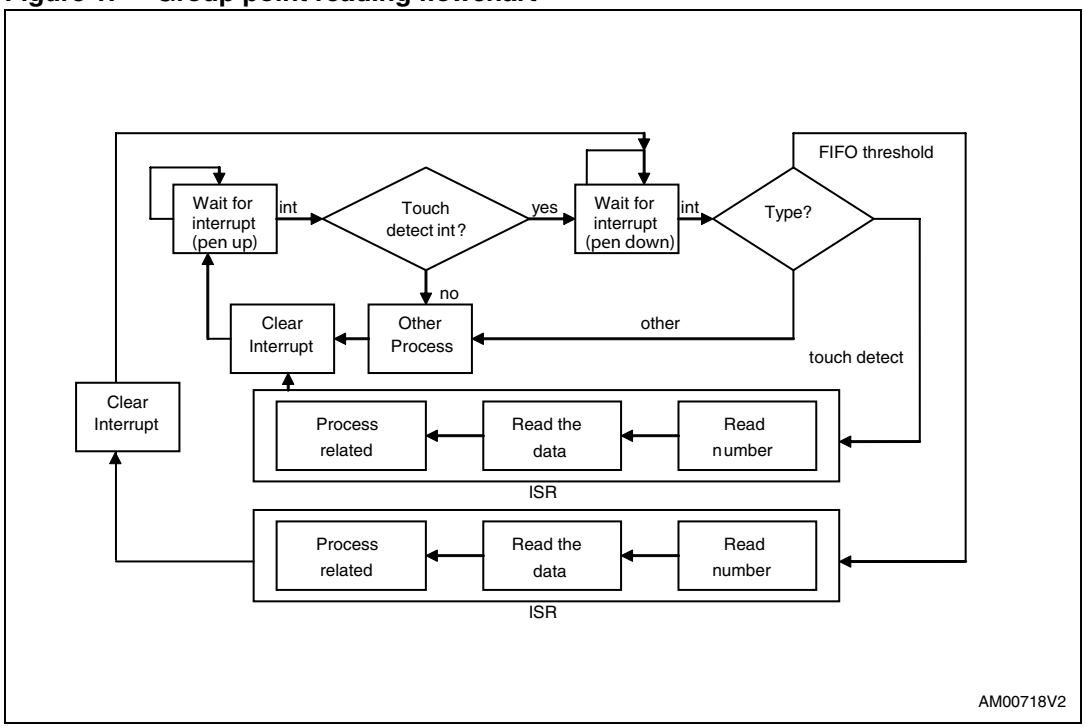
In applications requiring as many data as possible, the measurement time (provided by the settling time, touch detect delay and averaging) may be minimized and the data is read in group (more than one points data are read in one I^2C read FIFO transaction), in this document it is called “group points reading”.

In applications which need just a reasonable number of points, the measurement time can be adjusted so that it can produce a single sample in a period of time. The FIFO threshold can be written as 0, so that STMPE811 issues an interrupt in each sample acquired. It is advisable to ensure that the latency of the INT_STA register is less than the measurement time. This is called “single point reading”.

2.1 Group point reading

The flowchart for this system type is shown in *Figure 1*.

Figure 1. Group point reading flowchart



2.1.1 Group point reading code example

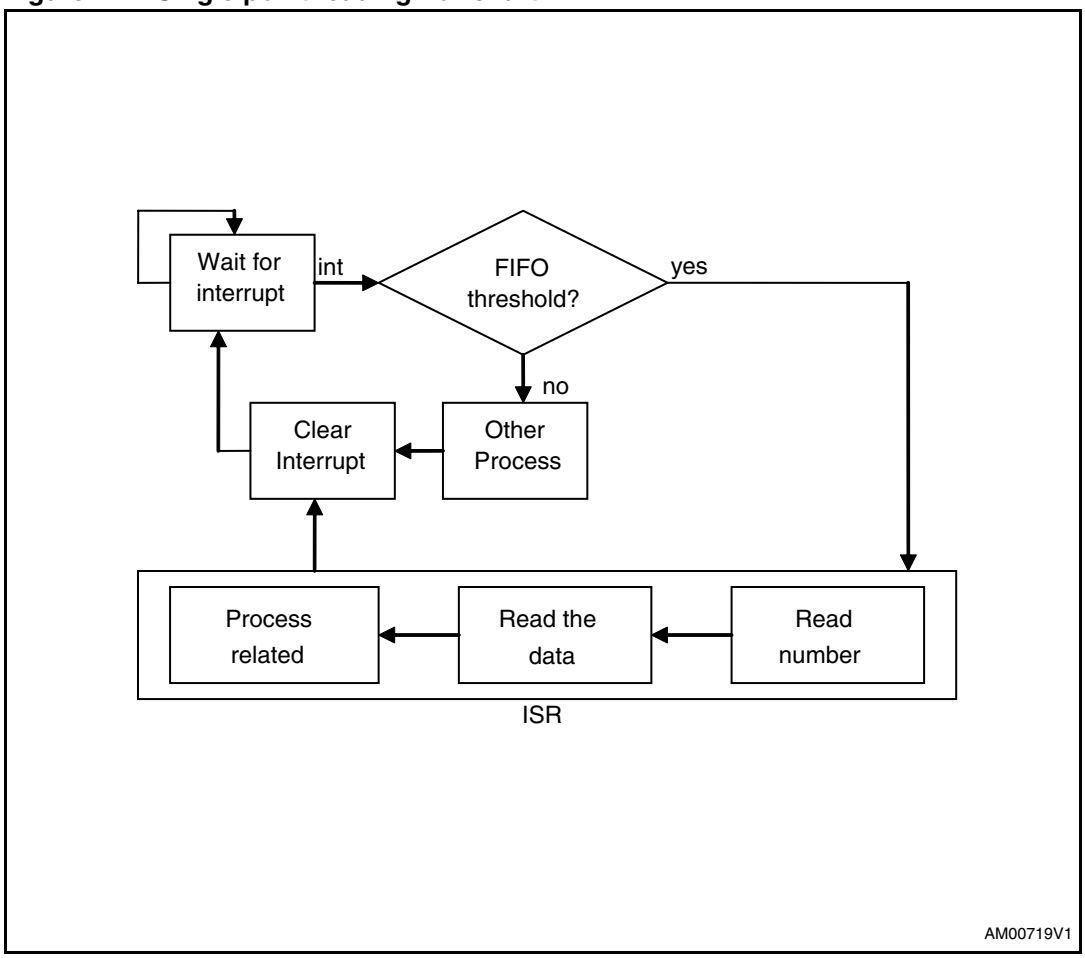
```
//Group Point Reading
void processInterrupt()
{
   int PenDown = 0;
   byte INT = 0;
   I2Creadarray(0x82, 0x0B, 0x01, INT); //read STMPE811 interrupt status
      if(PenDown==0) {
      if((INT & 0x01) == 0x01) PenDown=1;//Touch Detect?
      I2Cwritearray(0x82,0x0B, 0x01, 0x01);//clear touch detect interrupt
} else {
if((INT & 0x02) == 0x02) {
ReadGroup( );
I2Cwritearray(0x82, 0x0B, 0x01, 0x02);//clear interrupt
} else if((INT & 0x01) = 0x01) {
ReadGroup( );
PenDown=0;
I2Cwritearray(0x82, 0x0B, 0x01, 0x01);//clear interrupt
      }
}
}

void ReadGroup( ) {
byte Size;//number of points available in FIFO
I2Creadarray(0x82, 0x4C, 0x01, Size);//read number of points

   int Number;
   Number= Size*4; //4 bytes data for each point

byte Data[Number-1];
byte X[Size-1];
byte Y[Size-1];
byte Z[Size-1];

   I2Creadarray(0x82, 0xD7, Number, Data[0]);//read the data

   for(i=0;i<(Size-1);i++) {
      X[i] = data[(i*4)+0] < 4;
           temp = (data[(i*4)+1] & 0xF0) > 4;
```

```
            X[i] = (X[i] + temp);
            Y[i] = (data[(i*4)+1] & 0x0F) < 8;
            Y[i] = (Y[i] | data[(i*4)+2]);
            Z[i] = data[(i*4)+3];
    }
    //do necessary process with the data here
}
```

2.2 Single point reading

The single point reading flowchart is illustrated in *Figure 2*.

Figure 2. Single point reading flowchart



2.2.1 Single point reading code example

```
//Single Point Reading
void processInterrupt()
{
   I2Creadarray(0x82, 0x0B, 0x01, INT); //read STMPE811 interrupt status

   if((INT && 0x02) == 0x02) {
      ReadSingle( );
      I2Cwritearray(0x82, 0x0B, 0x01, 0x02);//clear interrupt
   }
}

void ReadSingle( ) {
   byte Size;//number of points available in the FIFO
   I2Creadarray(0x82, 0x4C, 0x01, Size);//read number of data

   int Number;
   Number = Size*4;//4 bytes data for each point

byte data[Number-1];
byte X[Size-1];
byte Y[Size-1];
byte Z[Size-1];

   I2Creadarray(0x82, 0xD7, Number, Data[0]);//read the data

   for(i=0;i<Size;i++) {
      X[i] = data[(i*4)+0] < 4;
      temp = (data[(i*4)+1] & 0xF0) > 4;
      X[i] = (X[i] + temp);
      Y[i] = (data[(i*4)+1] & 0x0F) < 8;
      Y[i] = (Y[i] | data[(i*4)+2]);
      Z[i] = data[(i*4)+3];
   }
   //…do necessary process with the data here
}
```

Note: *I^2C function*
I^2C write array (slave, sub addr, number, variable) ' write to device
I^2C read array (slave, subaddr, number, variable) ' read from device
Slave = slave address.
SubAddr = register address
Number = number of data to be read/written
Variable = variable that store data being read/written

3 Revision history

Table 1. Document revision history

Date	Revision	Changes
15-Sep-2008	1	Initial release.