AN3245
Application note

A hardware and software guide for the STMPE1801 Xpander Logic™ 18-bit enhanced port expander with keypad controller

Introduction

The STMPE1801 is a GPIO (general purpose input/output) port expander able to interface a main digital ASIC via a two-line bidirectional bus (I^2C). It offers the flexibility to be configured into a 10x8 matrix keypad controller for QWERTY keyboards and general purpose inputs/outputs.

This document highlights the guidelines and information complementary to the STMPE1801; *Xpander Logic^{TM} 18-bit enhanced port expander with keypad controller*, datasheet, which is necessary for the successful design of STMPE1801 in applications.

The first part of the document highlights information on the hardware including external components/connectivity, power, etc.

The second part of the document focuses on information regarding the software, in which programming sample codes are shown.

Contents

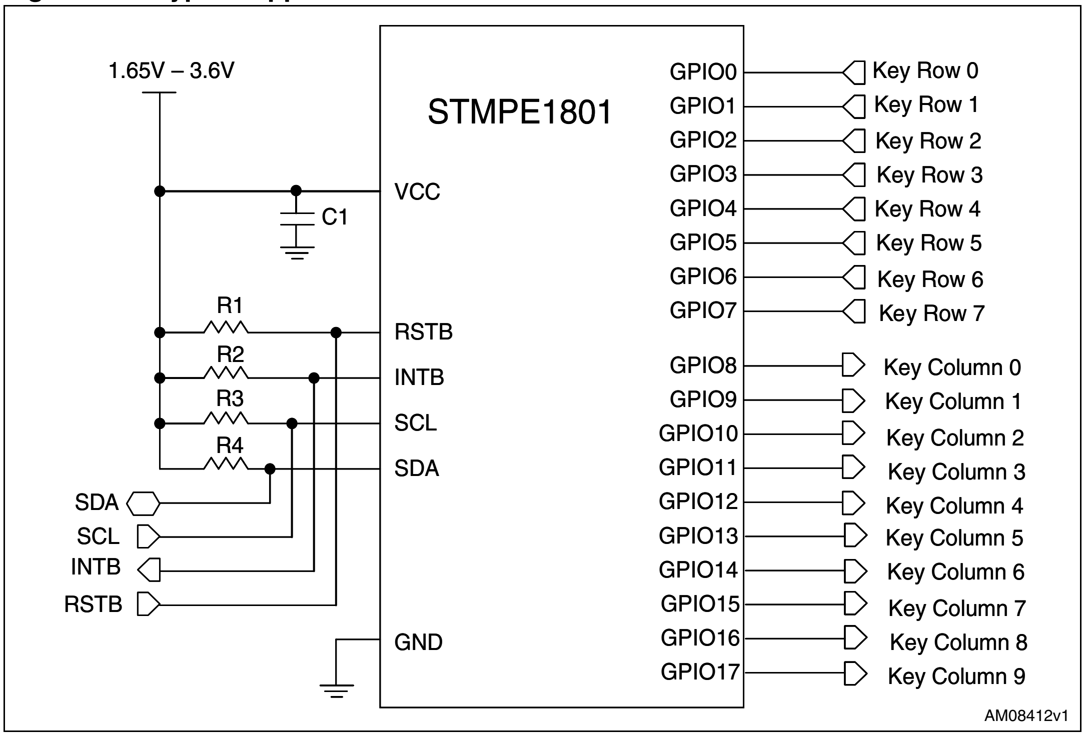
List of figures

1 Hardware

1.1 External components

1.1.1 Typical application circuit

Figure 1. Typical application schematic



Note: Recommended connection at RSTB is based on the use of Baseband/CPU GPIO as the control signal.

In a typical application, the following external components are required:

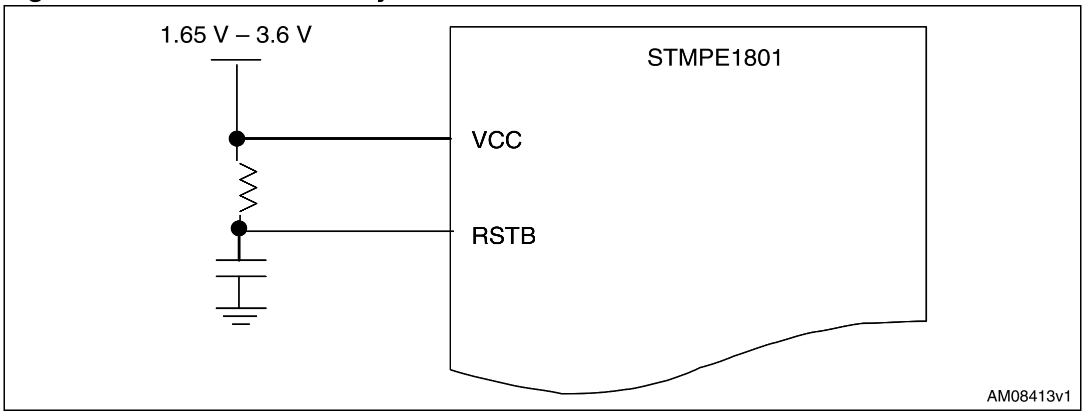
- R1: 10 kΩ pull-up resistor at RSTB
- R2: 2.2 kΩ - 10 kΩ pull-up resistor at INTB
- R3: 2.2 kΩ - 10 kΩ pull-up resistor at SCL (It is recommended that Vpullup ≥ VCC)
- R4: 2.2 kΩ - 10 kΩ pull-up resistor at SDA (It is recommended that Vpullup ≥ VCC)
- C1: 100 nF capacitor at VCC

1.1.2 RSTB pin recommendation

The following is a few examples of configurations at RSTB depending on applications.

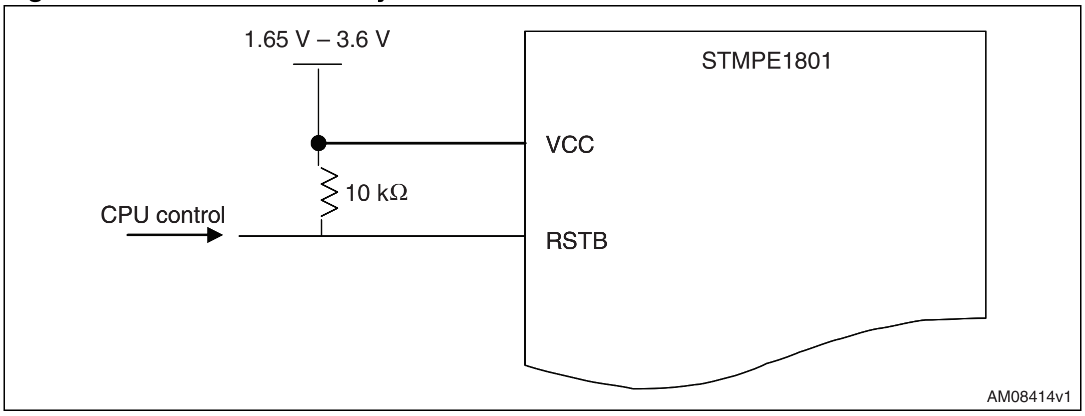
- If a reset delay is desired (recommended) upon power up, an RC delay can be connected to the RSTB as shown below.

Figure 2. RSTB connectivity recommendation 1



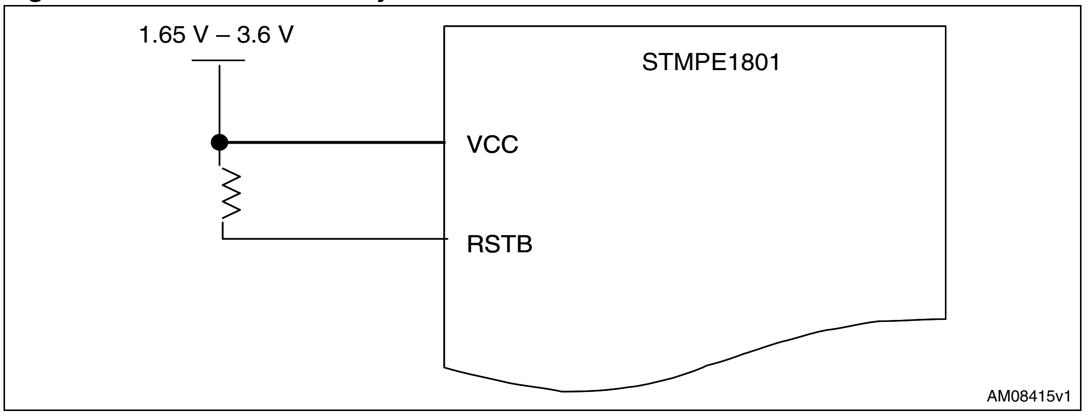
- If external reset assertion is required through CPU/baseband, RSTB can be connected to the GPIO. The diagram below shows the presence of a weak pull-up resistor assuming CPU/baseband control is open drain.

Figure 3. RSTB connectivity recommendation 2



- If reset delay and external reset assertion, as shown in case1 and case2 above, are not required, it is recommended to short the RSTB pin to VCC through a resistor.

Figure 4. RSTB connectivity recommendation 3



1.1.3 INTB pin recommendation

The INTB pin is programmable to active low or active high. When programmed to active low, a pull-up resistor of 2.2 kΩ -10 kΩ is required. When programmed to active high, a pull-down resistor of 2.2 kΩ - 10 kΩ is required.

If the INT signal is not in use, it is necessary to pull the INTB pin to VCC.

1.2 Power sequence (fail safe)

All GPIO pins of the STMPE1801 are NOT fail safe. This means that it is necessary to make sure that VCC supply to STMPE1801 is first turned on before driving the GPIO inputs.

All other pins except GPIO are with fail safe structures. It is possible to have these pins pull-up supply on (for SCL/SDA/INTB) or driven (for RSTB) before STMPE1801 VCC turns on.

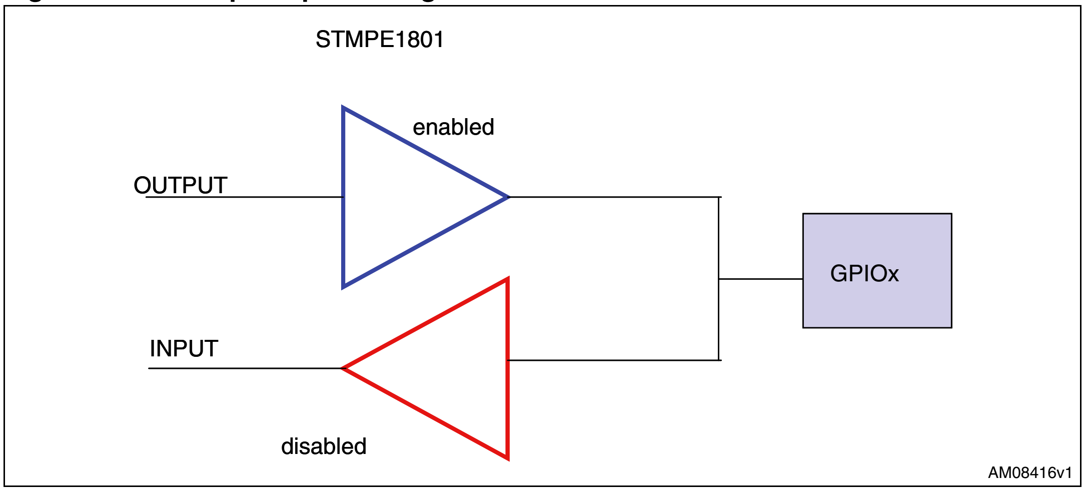
1.3 GPIO output configurations

The STMPE1801 provides push-pull type of GPIO output as is. If open drain GPIO outputs are required, it is configurable with a tweak to the software programming routine. See figures *5*, *6*, and *7*.

1.3.1 Push-pull

Set the GPIO to output state through the GPIO_SET_DIR (0x19 - 0x1B) register. Input path is disabled. Output path is enabled in push-pull configuration.

Figure 5. GPIO push-pull configuration

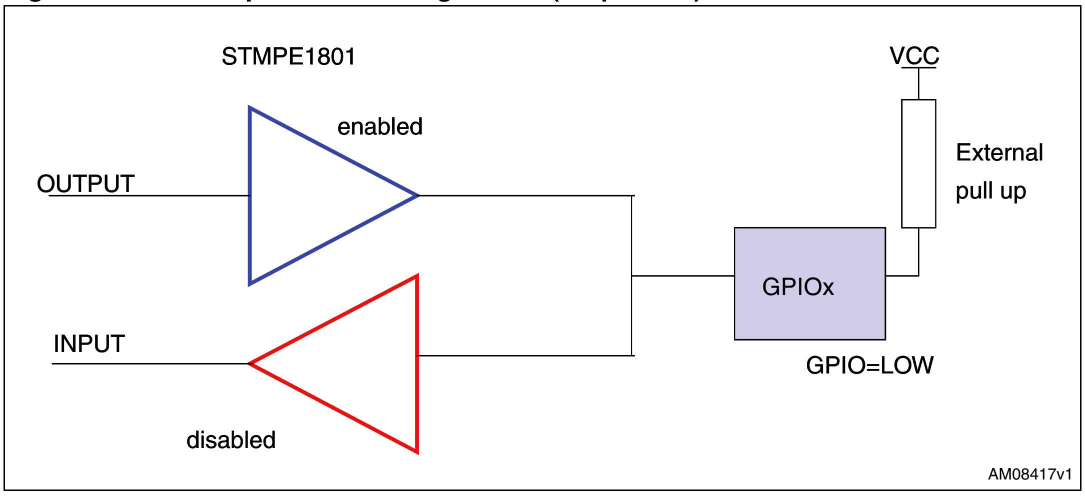


1.3.2 Open drain

GPIO output driven low by STMPE1801

Set the GPIO to output state through the GPIO_SET_DIR (0x19 - 0x1B) register. Input path is disabled. Set the output state to low through the GPIO_CLR (0x13 - 0x15) register. Output path is enabled and GPIO pin pulled low.

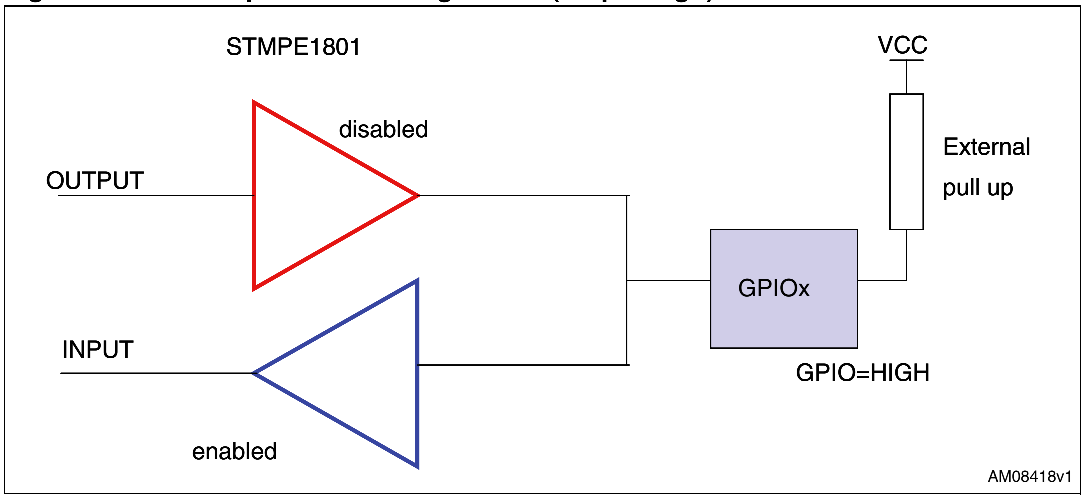
Figure 6. GPIO open drain configuration (output low)



GPIO output pulled high by external pull-up resistor

Set the GPIO to input state through the GPIO_SET_DIR (0x19 - 0x1B) register. Input path is enabled and output path disabled. GPIO is pulled high by external pull-up resistor.

Figure 7. GPIO open drain configuration (output high)



1.4 Keypad controller (KPC)

The integrated keypad controller (KPC) in the STMPE1801 is configurable to the following 3 types of keys:

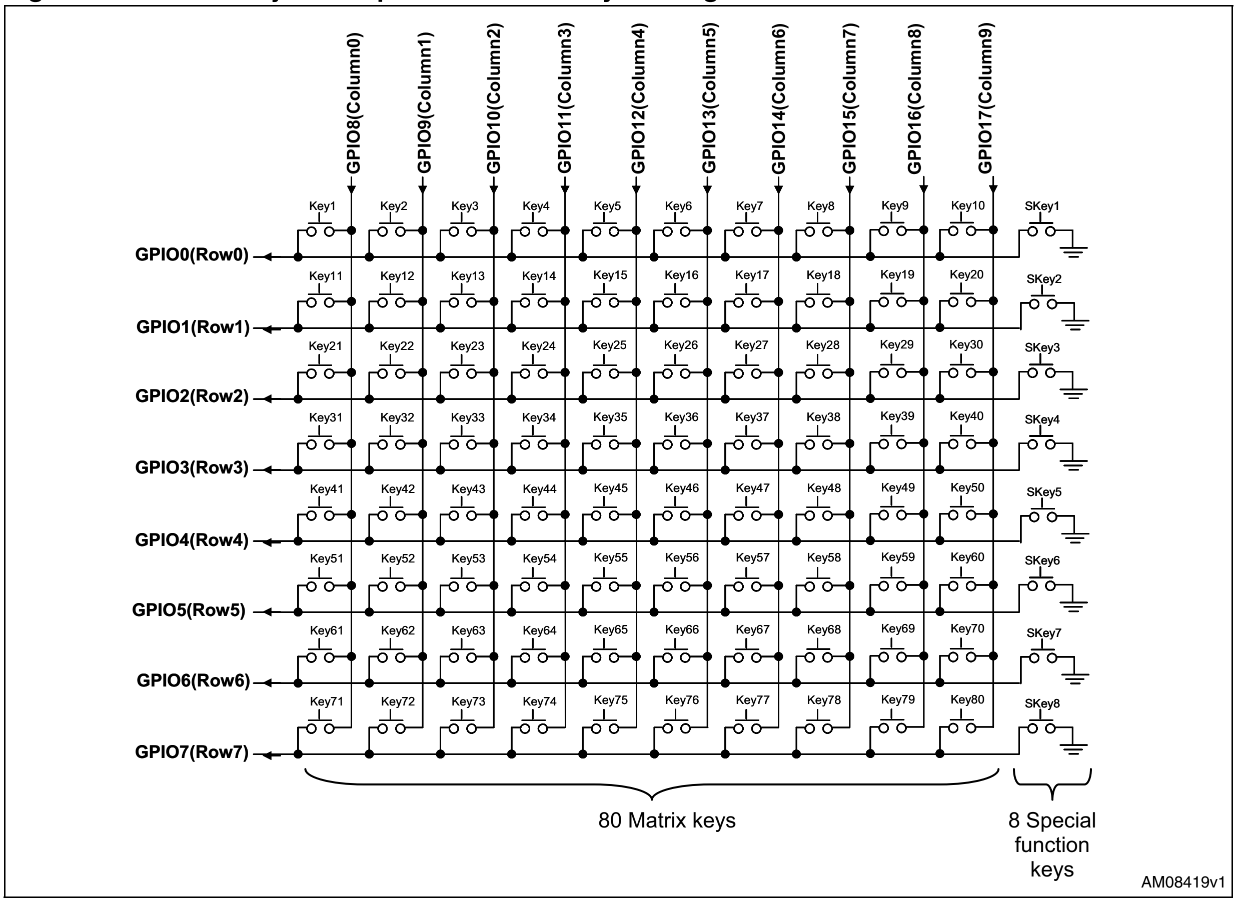
1. Up to 10x8 (80) matrix keys
2. Up to 8 special function keys
3. Up to 4 dedicated keys

1.4.1 Matrix keys and special function keys

The following is the configuration of the maximum 10x8 (80) matrix keys and 8 special function keys. When any of the special function keys are pressed, the matrix keys of the corresponding row are not detected. For example, when special function key 1 (SKey1) is pressed and held, all the matrix keys on row 0 (Key1- Key10) are not detected.

Please refer to *Section 1.4.4* for care and handling of ghost keys.

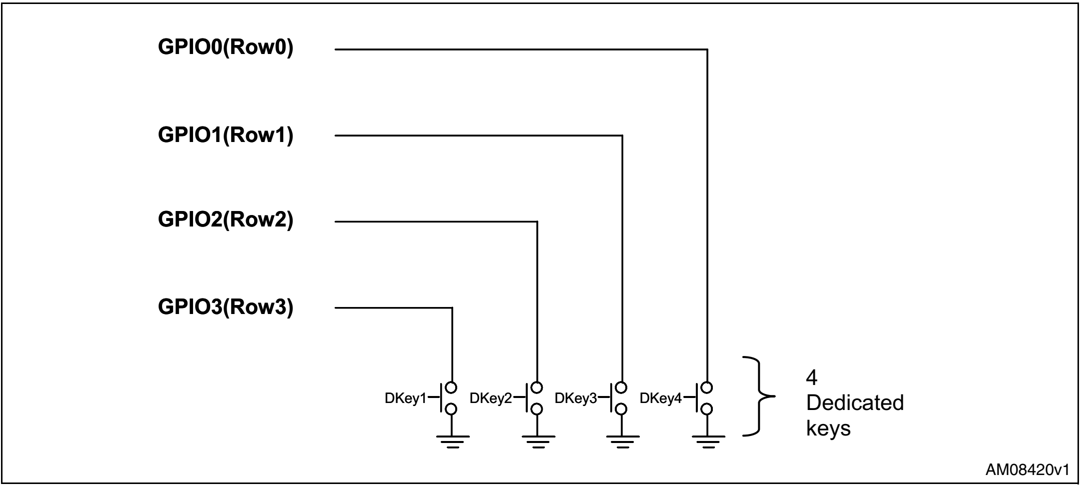
Figure 8. Matrix keys and special function keys configuration



1.4.2 Dedicated keys

The following is the configuration of the maximum 4 dedicated keys. Four of the row inputs (GPIO_0, GPIO_1, GPIO_2, and GPIO_3 pins) can be configured as dedicated keys through the setting of Dkey0~3 bits of the KPC_CTRL register. The dedicated keys function supports up to 4 simultaneous key presses in the applications.

Figure 9. Dedicated keys configurations



1.4.3 Combination keys

The 80 matrix keys can be programmed into a 3 combination key which when pressed simultaneously wakes up the keypad controller and sends an interrupt to the host system. This feature can be used to implement the <Ctrl> + <Alt> + <Delete> function.

Please refer to *Section 1.4.4* for care and handling of ghost keys.

1.4.4 Ghost keys handling

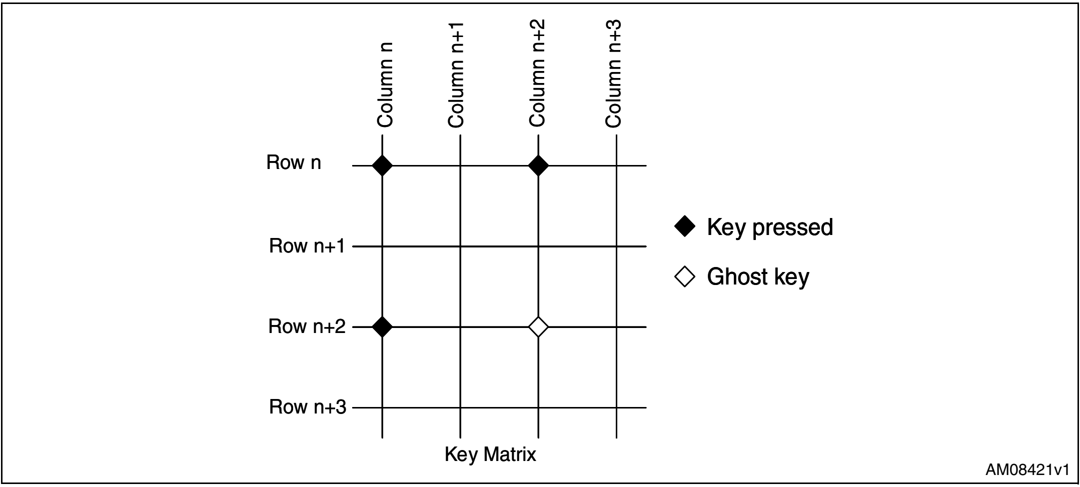
The ghost key is an inherent keypad matrix that is not equipped with a diode at each of the keys. While it is not possible to avoid ghost key occurrence, the STMPE1801 allows the detection of possible ghost keys through the capability of detecting 3 simultaneous key-presses in the key matrix.

The ghost key is only possible if 3 keys are pressed and held down together in a keypad matrix. If 3 keys are reported by the STMPE1801 keypad controller, it indicates a potential ghost key situation. The system may check for the possibility of a ghost key by analyzing the coordinates of the 3 keys. If the 3 keys form 3 corners of a rectangle, it may be a ghost key situation.

A ghost key may also occur in the “special function keys”. The keypad controller does not attempt to avoid the occurrence of ghost keys. However, the system should be aware that if more than one special function key is reported, then there is a possibility of ghost keys.

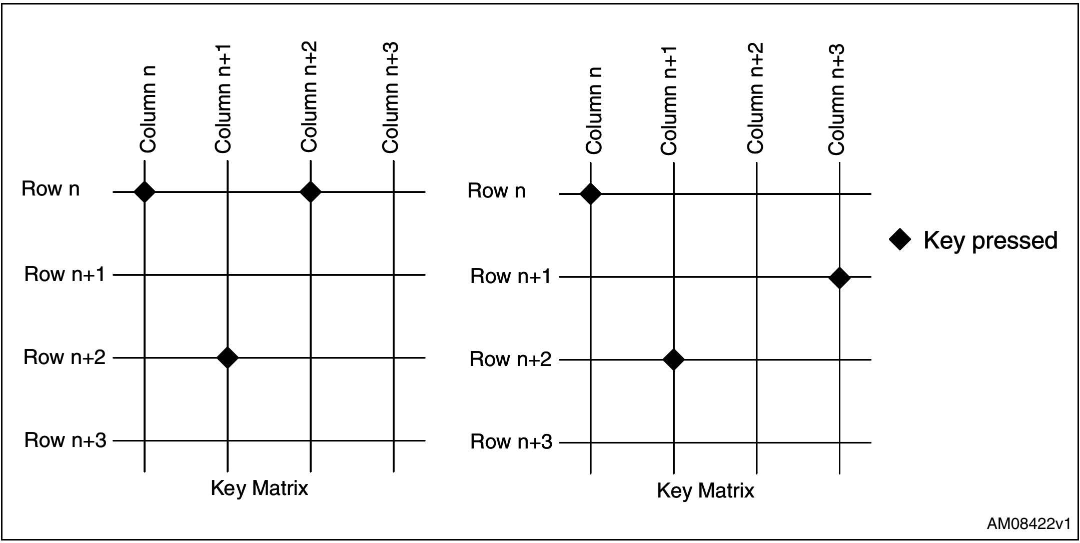
- Three simultaneous key presses with ghost key event
  - STMPE1801 stores and reports the 3 keys data
  - System host reads key coordinates that form a 90° corner of a rectangle. System host should be aware of the possible ghost key event in this condition and discard/ignore the key data.

Figure 10. Three simultaneous key presses with ghost key event



- Three simultaneous key presses without ghost key event
  - STMPE1801 stores and reports the 3 keys data
  - System host reads key coordinates that do not form a 90° corner of a rectangle. System host should recognize the key data as valid
  - For implementation of a function with 3 simultaneous key presses (e.g. <Ctrl> + <Alt> + <Del>) the user should avoid choosing key coordinates that can possibly result in a ghost key event

Figure 11. Three simultaneous key presses without ghost key event



1.5 Key column/row pins external capacitance

For enhancement of ESD performance, it is possible that external ESD protection diodes are connected to the key column and row pins in the application. This, at the same time, results in additional external capacitance loaded at the pins.

In order to ensure that the keypad controller scanning not be affected, the maximum recommended external capacitance at the pins is shown below.

- At VCC=1.8 V, recommended maximum external capacitance is 450 pF.
- At VCC=3.6 V, recommended maximum external capacitance is 900 pF.

These values are based on the calculation of the internal active pull-up resistance and the shortest KPC scanning time.

2 Software

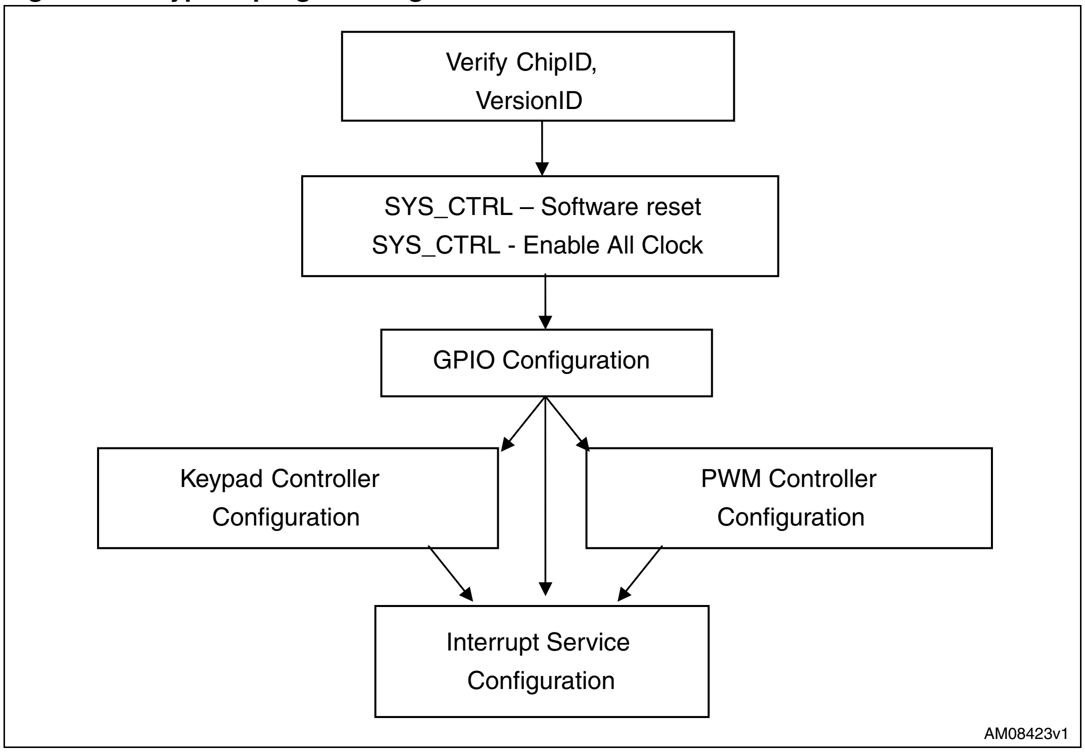
2.1 I^{2}C

2.1.1 I^{2}C initialization

It is recommended to insert the software reset as the first command during initialization before the starting of the I^{2}C transaction.

2.2 Programming guide

Figure 12. Typical programming flow


Verify ChipID, VersionID
SYS_CTRL – Software reset
SYS_CTRL - Enable All Clock
GPIO Configuration
Keypad Controller Configuration
PWM Controller Configuration
Interrupt Service Configuration
AM08423v1

2.2.1 Initialization

The following is an example for device initialization based on standard implementation for keypad, GPIO, and hotkey.

- Keypad initialization (e.g. 10x8 keypad initialization)

```
WriteRegister(SYS_CTRL, 0x80);//Issue SW reset
WriteRegister(KPC_ROW,0xFF); //Activate all rows
WriteRegister(KPC_COL_LOW,0xFF); //Activate all columns
WriteRegister(KPC_COL_HIGH,0x03);
WriteRegister(KPC_CTRL_LOW,0xF0);
```

```
//Set scan count, no dedicated keys
WriteRegister(INT_EN_MASK_LOW,0x06);
//Enable KPC FIFO overflow and KPC Interrupt
ReadRegister(INT_STA_LOW);
//Clear Interrupt Status register
WriteRegister(INT_CTRL_LOW,0x01);
//Active Low, Level Interrupt, Enable Global Interrupt
```

- GPIO initialization (e.g. GPIO low to output, GPIO MID to input)

```
WriteRegister(SYS_CTRL, 0x80);//Issue SW reset
WriteRegister(GPIO_SET_DIR_LOW,0xFF); //Set port to OUTPUT
WriteRegister(GPIO_SET_DIR_MID,0x00); //Set port to INPUT
WriteRegister(GPIO_SET_DIR_HIGH,0x03); //IO 16/17 UNUSED,Set to
OUTPUT
WriteRegister(INT_EN_MASK_LOW,0x08); //Enable GPIO Interrupt
WriteRegister(INT_EN_GPIO_MASK_MID,0xFF); //Enable 8-15 Input
Interrupt
ReadRegister(INT_STA_GPIO);
//Clear all status in GPIO status register
ReadRegister (INT_STA_LOW);
//Clear Interrupt Status register
WriteRegister(INT_CTRL_LOW,0x01);
//Active Low, Level Interrupt, Enable Global Interrupt
```

- Hotkey initialization (e.g. Setting GPIO15 as hotkey)

```
WriteRegister(GPIO_SET_DIR_MSB,0x80); //Set GPIO15 to INPUT
WriteRegister(GPIO_SET_RE_MSB,0x80);
//Set GPIO15 as RISING EDGE trigger
WriteRegister(INT_CTRL,0x01);
//Active Low, Level Interrupt, Enable Global Interrupt
```

2.2.2 Interrupt handling

The following is a sample for the device interrupt/hotkey serving routine based on standard implementation. It is necessary to ensure that the host and STMPE1801 interrupt type (active high/low and level/edge detect) are consistent.

- Interrupt service routine

```
ON_INTERRUPT
INT_STATUS = ReadRegister(INT_STA_LOW); //Read Interrupt Status
register
// GPIO Interrupt
```

```
If( (INT_STATUS & 0X08) == 0X08) //check for gpio interrupt
{
  GPIO_INT_STATUS = ReadRegister(INT_STA_GPIO, 3); // Read gpio int
mask
  If((GPIO_INT_STATUS[1] & 0x80) == 0x80) // Check for int on
GPIO15
  {
    // Handle GPIO interrupt
  }
}
// Keypad Interrupt Handling Routine
If(  (INT_STATUS & 0x02) == 0X02) //KPC Interrupt
{
  ReadArray(KPC_DATA, 5);
  // System handles the key data.

}
If(  (INT_STATUS&0x04) == 0X04) //KPC FIFO Overflow
{
  // SYSTEM HANDLES DATA OVERFLOW
}
```

2.2.3 Keypad press/release code

- Key release (key up) code

Table 1. **Key release code**

	Col 0	Col 1	Col 2	Col 3	Col 4	Col 5	Col 6	Col 7	Col 8	Col9
Row 0	0x80	0x88	0x90	0x98	0xA0	0xA8	0xB0	0xB8	0xC0	0xC8
Row 1	0x81	0x89	0x91	0x99	0xA1	0xA9	0xB1	0xB9	0xC1	0xC9
Row 2	0x82	0x8A	0x92	0x9A	0xA2	0xAA	0xB2	0xBA	0xC2	0xCA
Row 3	0x83	0x8B	0x93	0x9B	0xA3	0xAB	0xB3	0xBB	0xC3	0xCB
Row 4	0x84	0x8C	0x94	0x9C	0xA4	0xAC	0xB4	0xBC	0xC4	0xCC
Row 5	0x85	0x8D	0x95	0x9D	0xA5	0xAD	0xB5	0xBD	0xC5	0xCD
Row 6	0x86	0x8E	0x96	0x9E	0xA6	0xAE	0xB6	0xBE	0xC6	0xCE
Row 7	0x87	0x8F	0x97	0x9F	0xA7	0xAF	0xB7	0xBF	0xC7	0xCF

- Key press (key down) code

Table 2. **Key press code**

	Col 0	Col 1	Col 2	Col 3	Col 4	Col 5	Col 6	Col 7	Col 8	Col9
Row 0	0x00	0x08	0x10	0x18	0x20	0x28	0x30	0x38	0x40	0x48
Row 1	0x01	0x09	0x11	0x19	0x21	0x29	0x31	0x39	0x41	0x49
Row 2	0x02	0x0A	0x12	0x1A	0x22	0x2A	0x32	0x3A	0x42	0x4A
Row 3	0x03	0x0B	0x13	0x1B	0x23	0x2B	0x33	0x3B	0x43	0x4B
Row 4	0x04	0x0C	0x14	0x1C	0x24	0x2C	0x34	0x3C	0x44	0x4C
Row 5	0x05	0x0D	0x15	0x1D	0x25	0x2D	0x35	0x3D	0x45	0x4D
Row 6	0x06	0x0E	0x16	0x1E	0x26	0x2E	0x36	0x3E	0x46	0x4E
Row 7	0x07	0x0F	0x17	0x1F	0x27	0x2F	0x37	0x3F	0x47	0x4F

2.2.4 Key lock with combination keys

STMPE1801 provides key lock features when a combination key is defined. Without a programmed combination key, this feature is not active. Once activated all subsequent key presses are ignored. Key lock remains until a combination keys press is detected.

Key lock is enabled when:

1. Bit[1] of the KPC_CMD register is written '1' and
2. All keys are released.

Read back of '1' at bit [1] of the KPC_CMD register only means that the key lock command is active. It doesn't mean that the device has already entered into key lock mode.

Read back of '0' at bit [1] of the KPC_CMD register means that the key lock command is not active or the device has exited from key lock mode.

2.3 Hotkey de-bounce

It is possible that a signal from a mechanical connector (e.g. a 3.5 mm earphone jack) is connected to the STMPE1801 input as a hotkey input. In such a case, excessive noise is expected from the hotkey input due to the mechanical movement.

STMPE1801 provides a programmable hotkey de-bounce ranging from 30 µs to 210 µs (SYS_CTRL Register 0x02). If this is not sufficient, it is also possible to connect the noise input to one of the 4 dedicated keys (GPIO_0, GPIO_1, GPIO_2, and GPIO_3 pins) to which de-bounce can be programmed from 10 ms to 127 ms (KPC_CTRL_MID Register 0x34).

If the above de-bounce is still not sufficient, the host can implement the filtering or de-bouncing by polling the status bit in the interrupt status register (0x09).

2.4 Power modes transition

Table 3 highlights the states of keypad FIFO, GPIO, Interrupt, and PWM during the various power modes:

- Operational mode
- Hibernation mode
- Reset

Table 3. Power modes vs. GPIO/Keypad/Interrupt data

	Operational mode	Hibernation mode	Reset (HW, SW, POR, general call)
Keypad configuration and FIFO data	Active	Sustained (FIFO is read and cleared prior to entering hibernation mode)	Cleared
GPIO configuration, pin state, and data	Active	Sustained	Cleared
Interrupt configuration and data	Active	Sustained	Cleared

3 Revision history

Table 4. Document revision history

Date	Revision	Changes
03-Nov-2010	1	Initial release.