Using M258 to Drive STK3420 Sensor

Example Code Introduction for 32-bit NuMicro® Family

Document Information

Application	This example code demonstrates gesture sensing using the M258 series microcontroller (MCU) to drive the STK3420 sensor via I²C.
BSP Version	M251_M252_M254_M256_M258_Series_BSP_CMSIS_V3.00.005
Hardware	NuMaker-M258KE V1.2

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1. Overview

As touchless control devices and interactive technology become increasingly widespread, light sensing, distance sensing, and gesture recognition technologies are progressively playing crucial roles in microcontroller application domain. Light sensing that detects light intensity is used in environmental monitoring, lighting systems, etc. Distance sensing accurately measures the distance between objects and sensors, widely applicable in automation and robotics. Gesture recognition allows devices to identify human gestures, applicable in interactive technology, smart home devices, etc.

This example code utilizes Sensortek's “Proximity with Ambient Light Sensor” chip - STK3420, demonstrating how to get gesture values from this sensor using the I²C protocol.

1.1 Principle

The STK3420 sensor integrates a photodiode, ADC, and a built-in 940nm infrared LED, utilizing a noise cancellation scheme to mitigate environmental infrared interference. The STK3420’s internal block diagram is shown in Figure 1-1. It is operated by using its photodiode to distinguish object movements; when combined with software algorithms, it accurately detects 3D gestures (up/down/left/right/far/near). However, this example code only identifies up/down/left/right gestures.

For external communication, it interfaces via I²C, with the STK3420's address set as a 7-bit value of 0x58. All operations are accessed (Wrote/Read) via instructions to registers. The register map is shown in Figure 1-2, while the settings and values configured in this example code are described in Table 1-1. For further detailed configurations, please refer to the STK3420’s Datasheet.

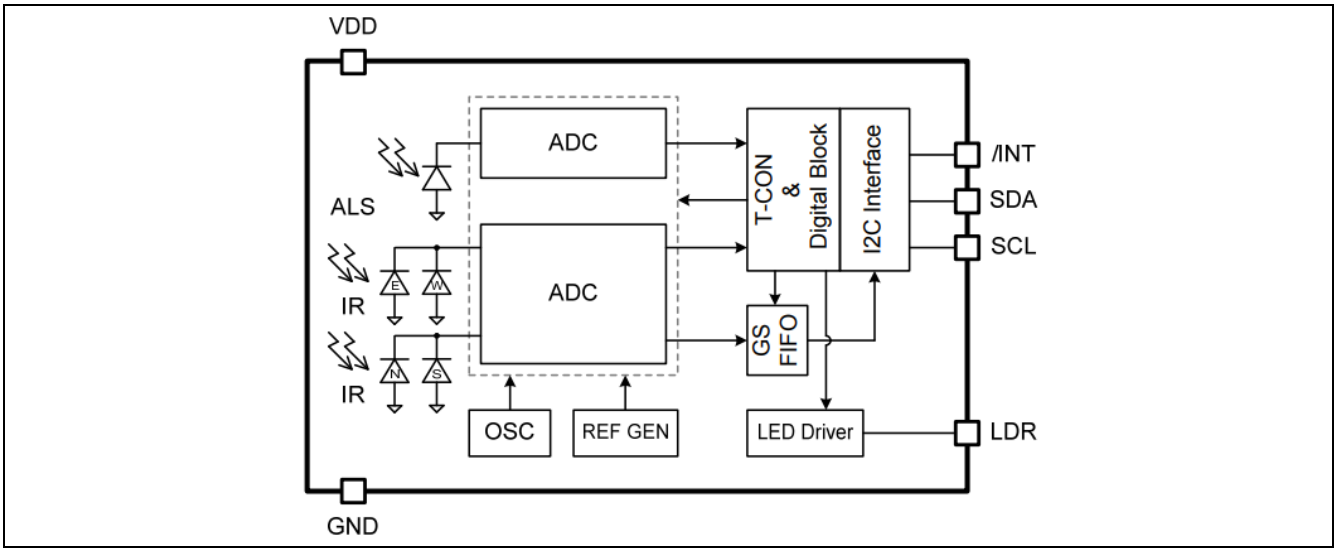

Figure 1-1 Block Diagram of STK3420

ADDR	REG NAME	BIT								Default
		7	6	5	4	3	2	1	0	
0x00	STATE	EN_IRS		EN_ALS_RO	EN_PS_RO	EN_WAIT_ALS	EN_WAIT_PSGS	EN_ALS	EN_PS	0x00
0x01	PS/GSCTRL1	PRST_PS[1:0]				IT_PSGS[3:0]				0x01
0x02	ALSCTRL1	PRST_ALS[1:0]		GAIN_ALS[1:0]		IT_ALS[3:0]				0x34
0x03	LEDCTRL	IRDR_LED[1:0]								0x40
0x04	INT	INT_CTRL			EN_GS_INT	EN_ALS_INT		PS_MODE	EN_PS_INT	0x00
0x05	WAIT1_PSGS	WAIT1_PSGS[7:0]								0x00
0x06	THDH1_PS	THDH_PS[15:8]								0xFF
0x07	THDH2_PS	THDH_PS[7:0]								0xFF
0x08	THDL1_PS	THDL_PS[15:8]								0x00
0x09	THDL2_PS	THDL_PS[7:0]								0x00
0x0A	THDH1_ALS	THDH_ALS[15:8]								0xFF
0x0B	THDH2_ALS	THDH_ALS[7:0]								0xFF
0x0C	THDL1_ALS	THDL_ALS[15:8]								0x00
0x0D	THDL2_ALS	THDL_ALS[7:0]								0x00
0x10	FLAG	FLG_ALS_DR	FLG_PS_DR	FLG_ALS_INT	FLG_PS_INT	FLG_GS_INT		FLG_IRS_DR	FLG_NF	0x01
0x11	DATA1_PS	DATA_PS[15:8]								
0x12	DATA2_PS	DATA_PS[7:0]								
0x13	DATA1_ALS	DATA_ALS[15:8]								
0x14	DATA2_ALS	DATA_ALS[7:0]								
0x17	DATA1_IRS	DATA_IRS[15:8]								
0x18	DATA2_IRS	DATA_ IRS[7:0]								
0x19	ALSCTRL2	ITER_IRS[1:0]		GAIN_IRS[1:0]		IT_IRS[3:0]				0x74
0x1B	WAIT_ALS	WAIT_ALS[7:0]								0x00
0x1C	WAIT2_PS		WAIT2_PS[6:0]							0x00
0x1D	PS/GSCTRL2		EN_INTELLIGENT_PRST					EN_GSNS	EN_GSEW	0xF0
0x1E	GSFLAG	FLG_GS_FIFO_OV			GS_FIFO_LEN[4:0]					
0x1F	GSFIFOCTRL					GS_FIFO_THD[3:0]				0x08
0x20	DATA1_GSE	DATA_GSE[15:8]								
0x21	DATA2_GSE	DATA_GSE[7:0]								
0x22	DATA1_GSW	DATA_GSW[15:8]								
0x23	DATA2_GSW	DATA_GSW[7:0]								
0x24	DATA1_GSN	DATA_GSN[15:8]								
0x25	DATA2_GSN	DATA_GSN[7:0]								
0x26	DATA1_GSS	DATA_GSS[15:8]								
0x27	DATA2_GSS	DATA_GSS[7:0]								
0x3E	PDT_ID	PDT_ID[7:0]								0x50
0x3F	Reserved	Reserved								
0x80	SOFT_RESET	Write any to soft reset								
0x94	MISC1						LED_DIV2			0xC3

Figure 1-2 STK3420 Register List

Reg Addr	Reg Name	Value	Description
0x1d	PS/GSCTRL2	0xf3	Enable EWNS gesture detection and PS intelligent persistence.
0x5	WAIT1_PSGS	0x6	Set waiting state period. (6+1)*780 us = 5.46 ms
0x1	PS/GSCTRL1	0x2	Set gesture refreshing time to 390 us, and PS persistence times to 1.
0x95	Analog Gain	0x90	To perform gesture well, the analog gain should be set to 2 times, and the register value is 0x90. Note: This register is not mentioned in Datasheet.
0x3	LEDCTRL	0xc0	Set LED current sink to 150 mA.
0x94	MISC1	0x35	Set LED current control (LED_DIV2) to 1.
0x2	ALSCTRL1	0x2	Set ALS persistence times to 1, ALS gain to 1, and ALS refreshing time to 50 ms.
0x0	STATE	0x4	Enable PS/GS waiting state.

Table 1-1 Register Setting and Description

The STK3420 utilizes a 128-byte FIFO to store 16 sets of gesture values. Each set of gesture data comprises output data of E/W/N/S within each ADC conversion cycle. As shown in Figure 1-3, the FIFO data can be accessed by reading the DATA_GSE/W/N/S registers. Once these values are obtained, they are processed using Sensortek's library, allowing for precise gesture identification.

ADDR	REG NAME	BIT								Default
		7	6	5	4	3	2	1	0	
0x20	DATA1_GSE	DATA_GSE[15:8]								
0x21	DATA2_GSE	DATA_GSE[7:0]								
0x22	DATA1_GSW	DATA_GSW[15:8]								
0x23	DATA2_GSW	DATA_GSW[7:0]								
0x24	DATA1_GSN	DATA_GSN[15:8]								
0x25	DATA2_GSN	DATA_GSN[7:0]								
0x26	DATA1_GSS	DATA_GSS[15:8]								
0x27	DATA2_GSS	DATA_GSS[7:0]								

Figure 1-3 Register for Gesture Data

Note: The images in this document are sourced from the STK3420's Datasheet.

1.2 Demo Result

The compilation and programming results of this example code are shown in Figure 1-4.

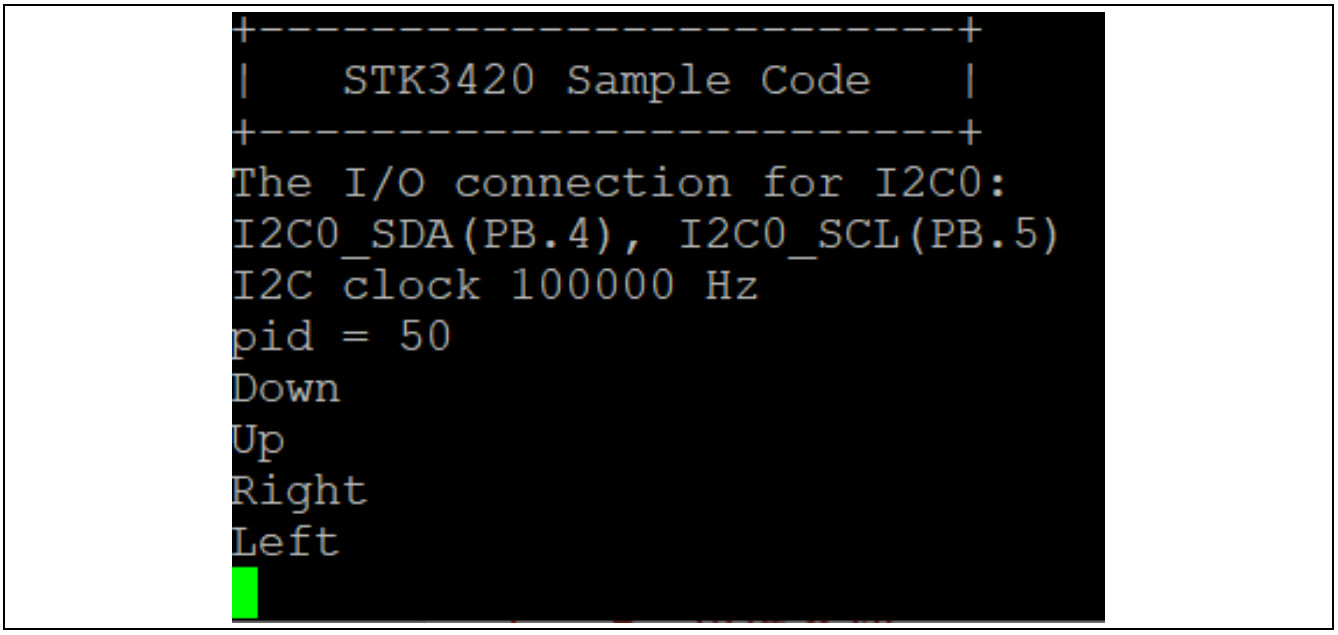

Figure 1-4 Compilation and Programming Results

2. Code Description

The main program is entered through *main.c* and consists of four parts as follows:

- Clock initialization and pin definition: SYS_Init
- Peripheral initialization and open: UART0_Init, I2C0_Init
- Register setting for STK3420: STK3420_start
- Gesture calculation: read DATA_GSE/W/N/S register, STK_Gesture.lib

For the detailed register settings, please refer to Table 1-1 and STK3420's Datasheet.

```
void STK3420_start(void)
{
    I2C_WriteByteOneReg(I2C0, STK342X_SLAVE_ADDR, 0x80, 0x00);
    CLK_SysTickDelay(20000);   // wait STK3420 IC Reset

    I2C_ReadByteOneReg(I2C0, STK342X_SLAVE_ADDR, 0x3E, &PID);
    printf("pid = %x\n", PID);
    //enable EWNS gesture detection and PS intelligent persistence
    I2C_WriteByteOneReg(I2C0, STK342X_SLAVE_ADDR, 0x1d, 0xf3);
    //set wait state period (6+1)*780 us = 5.46 ms
    I2C_WriteByteOneReg(I2C0, STK342X_SLAVE_ADDR, 0x5,  0x6);
    //set gesture refresh time to 390 us, PS persistence setting to 1 times
    I2C_WriteByteOneReg(I2C0, STK342X_SLAVE_ADDR, 0x1,  0x2);
    I2C_WriteByteOneReg(I2C0, STK342X_SLAVE_ADDR, 0x95, 0x90);//set analog gain
    //set LED current sink to 150mA
    I2C_WriteByteOneReg(I2C0, STK342X_SLAVE_ADDR, 0x3,  0xc0);
    I2C_WriteByteOneReg(I2C0, STK342X_SLAVE_ADDR, 0x94, 0x35);//set LED_DIV2 to 1
    /*ALS setting: set ALS persistence to 1 times, ALS gain to 1,
    and ALS refresh time to 50 ms*/
    I2C_WriteByteOneReg(I2C0, STK342X_SLAVE_ADDR, 0x2,  0x2);
    I2C_WriteByteOneReg(I2C0, STK342X_SLAVE_ADDR, 0x0,  0x4);//enable PS/GS wait state
    memset(&xtGes, 0, sizeof(tfGesture));
    Gesture_Init(&xtGes);
}
```

After system initialization and sensor activation settings, the system enters a loop to read the E/W/N/S values stored in the FIFO. These values are then processed through algorithms within the *Stk_Gesture.lib*, which leads to precise result of the gesture direction.

```
int32_t main(void)
{
    int i = 0;
    /* Unlock protected registers */
    SYS_UnlockReg();

    /* Init System, IP clock and multi-function I/O */
    SYS_Init();

    /* Init UART0 for printf */
    UART0_Init();
```

```
    /* Lock protected registers */
    SYS_LockReg();

    printf("+-------------------------+\n");
    printf("|   STK3420 Sample Code   |\n");
    printf("+-------------------------+\n");

    printf("The I/O connection for I2C0:\n");
    printf("I2C0_SDA(PB.4), I2C0_SCL(PB.5)\n");

    /* Init I2C0 */
    I2C0_Init();

    STK3420_start();

    while(1) {
        uint8_t fifo_cnt = 0;
        uint8_t ges_data[8];
        uint16_t gse = 0;
        uint16_t gsw = 0;
        uint16_t gsn = 0;
        uint16_t gss = 0;

        // Read FIFO Frame Cnt
        I2C_ReadByteOneReg(I2C0, STK342X_SLAVE_ADDR, 0x1E, &fifo_cnt);

        if(fifo_cnt) {
            for(i = 0; i < (fifo_cnt & 0x1F); i++) {
                // Read FIFO Data
                I2C_ReadMultiBytesOneReg(I2C0, STK342X_SLAVE_ADDR, 0x20, ges_data, 8);
                gse = (ges_data[0] << 8 | ges_data[1]);
                gsw = (ges_data[2] << 8 | ges_data[3]);
                gsn = (ges_data[4] << 8 | ges_data[5]);
                gss = (ges_data[6] << 8 | ges_data[7]);
                int act = iGesture_Do(&xtGes, gse, gsw, gss, gsn);
                switch(act) {
                    case GESTURE_EVENT_UP:
                        printf("Up\n");
                        break;
                    case GESTURE_EVENT_DOWN:
                        printf("Down\n");
                        break;
                    case GESTURE_EVENT_LEFT:
                        printf("Left\n");
                        break;
                    case GESTURE_EVENT_RIGHT:
                        printf("Right\n");
                        break;
                    default:
                        break;
                }
            }
        }
```

3. Software and Hardware Requirements

3.1 Software Requirements

- BSP version
  - M251_M252_M254_M256_M258_Series_BSP_CMSIS_V3.00.005
- IDE version
  - Keil uVersion 5.38

3.2 Hardware Requirements

- Circuit components
  - NuMaker-M258KE V1.2
- Pin Connect
  - Please refer to the Pin description in Table 3-1 and the connection in Figure 3-1.

STK3420	Description	M258KE3AE
VDD	Power supply for 2.4V to 3.6V.	VDD
SDA	Data line for I²C interface.	I²C0_SDA Pin 2 (PB.4)
SCL	Serial clock for I²C interface.	I²C0_SCL Pin 1 (PB.5)
LEDA	The anode of the embedded IR LED, connected to power.	VDD
GND	Ground.	VSS

Table 3-1 Pin Description

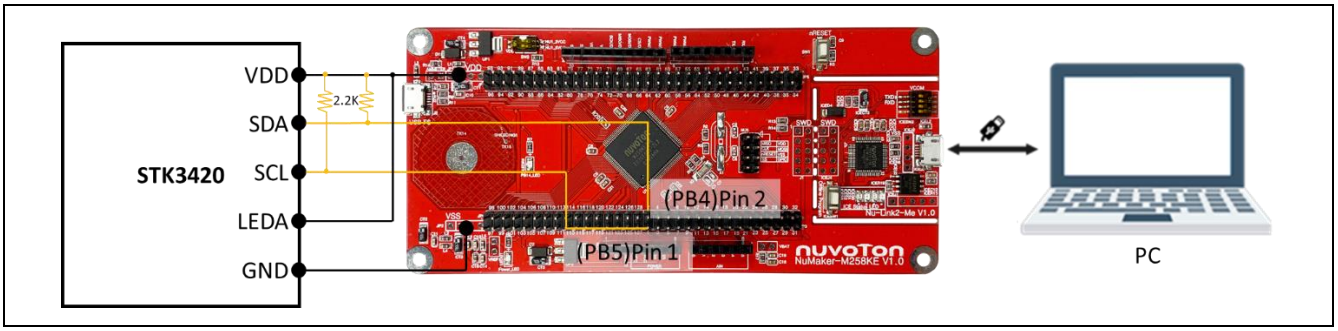

Figure 3-1 Pin Connect

4. Directory Information

The directory structure is shown below.

EC_M258_Gesture_Sensor_STK3420_V1.00	
Library	Sample code header and source files
CMSIS	Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
Device	CMSIS compliant device header file
StdDriver	All peripheral driver header and source files
SampleCode	
ExampleCode	
Project	Source file of example code

Figure 4-1 Directory Structure

5. Example Code Execution

1. Browse the sample code folder as described in the Directory Information section and double-click *M258_Gesture_Sensor_STK3420.uvprojx*.
2. Enter Keil compile mode.
   - Build
   - Download
   - Start/Stop debug session
3. Enter debug mode.
   - Run

6. Revision History

Date	Revision	Description
2023.12.01	1.00	Initial version.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.