**Freescale Semiconductor, Inc.**
Application Note

Document Number: AN4872
Rev. 0, 03/2014

# Using Four-Wire Interface I$^2$C on KE06

**by: Ben Wang**

## 1 Introduction

This application note describes the procedure to use the four-wire I$^2$C interface on KE06 of Kinetis E family. The sample code provided in this application note is tested on KE06 through I$^2$C master/slave communication between two evaluation boards.

**Contents**

## 2 KE06 I$^2$C overview

KE06 contains two inter-integrated circuit (I$^2$C) modules with SMBus feature. I$^2$C provides a method of communication between a number of devices. The interface operates at up to 100 kb/s with maximum bus loading and timing. The device is capable of operating at higher baud rates, up to a maximum of clock/20, with reduced bus loading. The maximum communication length and the number of devices that can be connected are limited by a maximum bus capacitance of 400 pF.

In addition, KE02 I$^2$C0 also provides four-wire interface option.

## 3 Four-wire interface feature

In most appliance applications there is heavy bus load and a lot of noise on the bus. Additional bus switch or line driver may be required in order to provide robust I$^2$C communication. This will add additional system BOM cost to the traditional two-wire I$^2$C communication. Instead of use traditional I2C SDA\SCL bidirectional, four-wire interface add an option for split them into input and output functionality. In four pin configurations, SDA_IN, SDA_OUT, SCL_IN, and SCL_OUT pins are present with properly inverted outputs. It allows customers to design their own line driver with minimum additional cost and can be used to improve the noise immunity of the I$^2$C bus. This change can be implemented with two transistors, six resistors, and two diodes.

### 3.1 Four-wire interface configuration

Four-wire interface is enabled when the KE06 SIM_SOPT1[I2C04WEN] bit is set. Input of SDA/SCL input is present in SDA_IN/SCL_IN(PTA2/PTA3 on KE06) bit and the output is present in SDA_OUT/SCL_OUT(PTA1/PTA0 on KE06) bit.

User can set SIM_SOPT1[I2C0OINV] bit after the four-wire interface feature is enabled, by settling SIM_SOPT1[I2C04WEN] bit, and the SDA_OUT/SCL_OUT will be inverted before output.

However, this feature is available only when I$^2$C0 pin-out is not remapped.

The following figure represents the I$^2$C0 four-wire interface diagram:

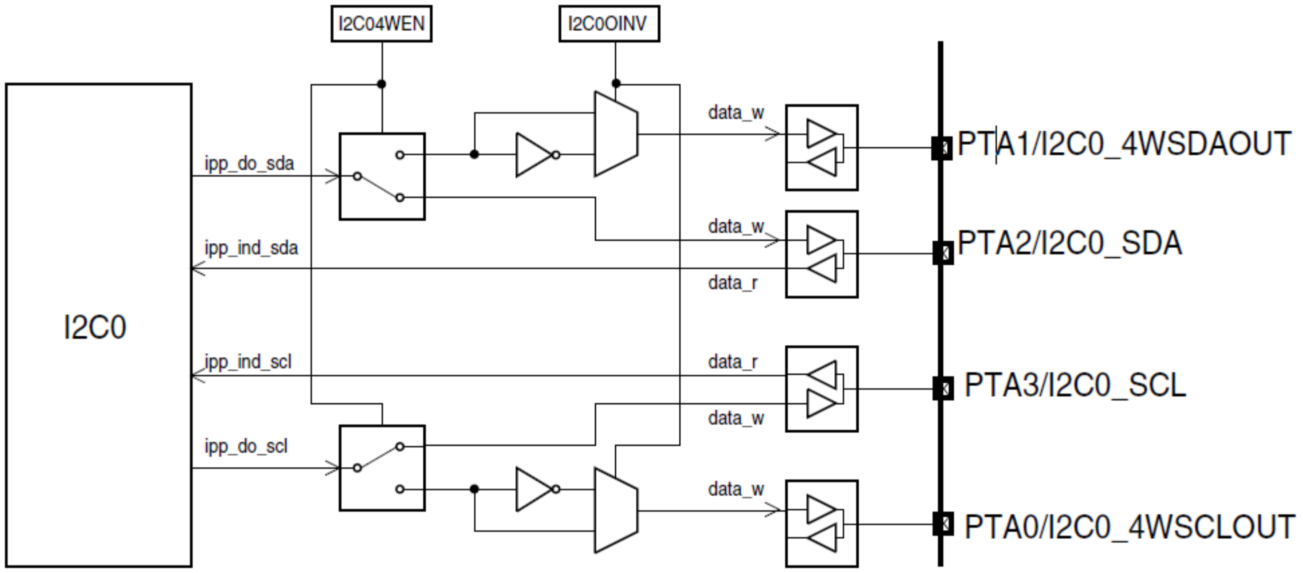

**Figure 1. I$^2$C0 four-wire interface diagram**

## 4 Four-wire interface test

Four-wire interface is tested on internal evaluation board. As discussed in the previous section, I$^2$C peripheral redesign is not required in four-wire I$^2$C, only a different interface is required.

### 4.1 Hardware setup

Figure 1 represents the general four-wire I$^2$C interface connection.

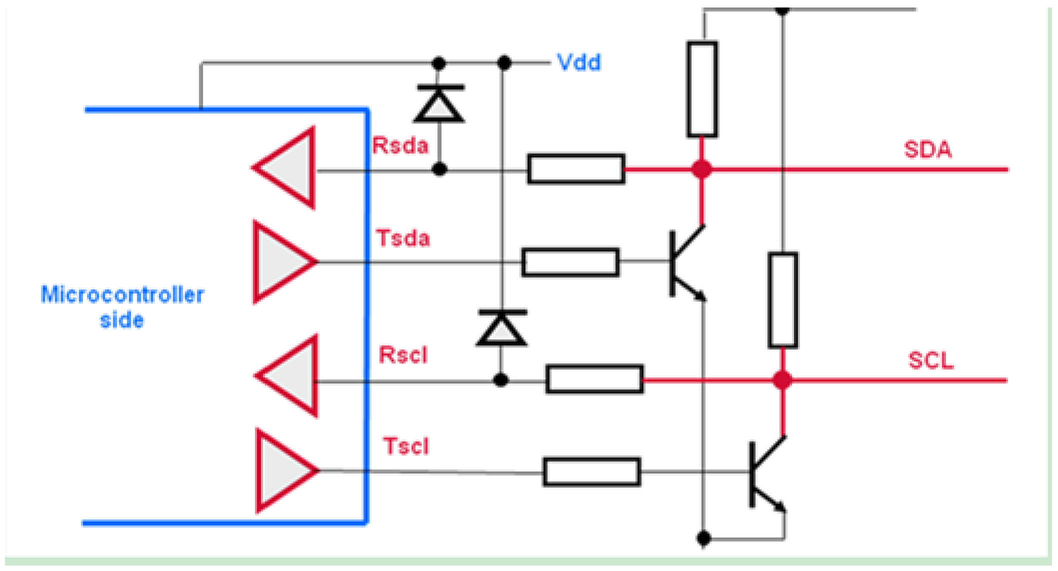

**Figure 2. Four-wire I$^2$C interface connection**

The following figure represents the actual four-wire I$^2$C connection on the EVB:

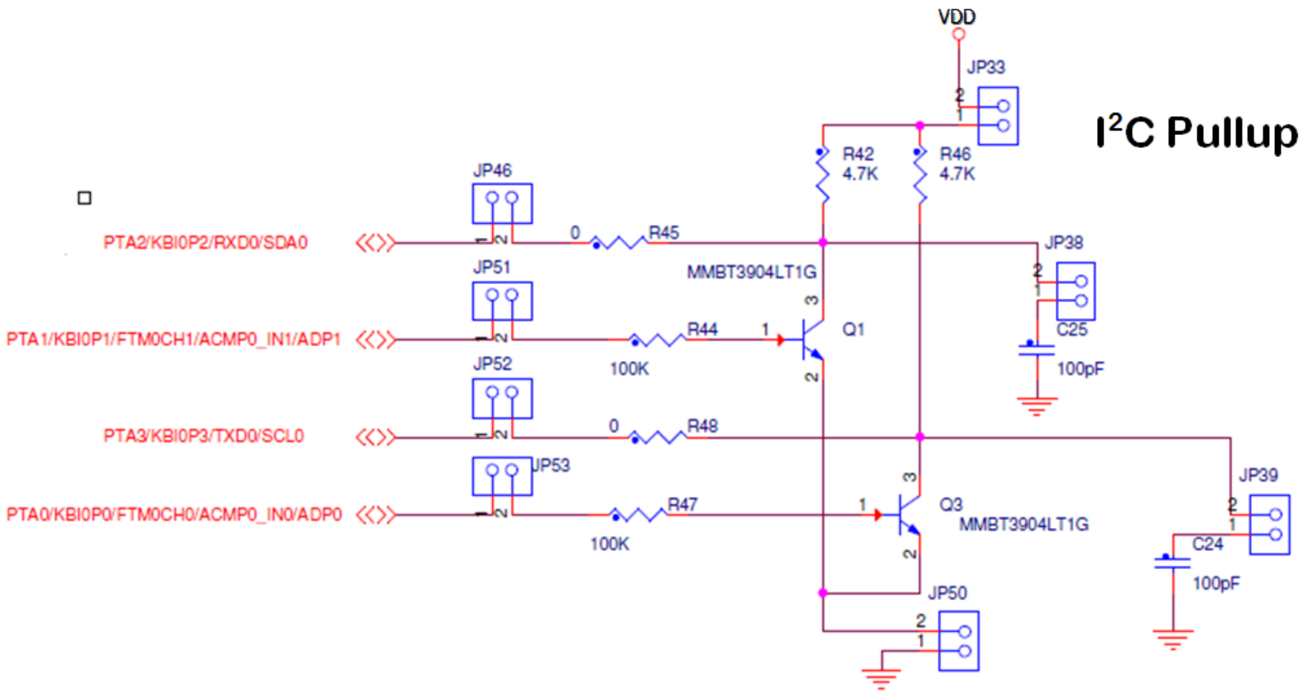

**Figure 3. I$^2$C0 four-wire hardware schematic**

## 4.2 Software setup

There are only two bits in SIM_SOPT1 register to set during four-wire interface test in software. The other I$^2$C master/slave configurations are same as two-wire mode.

Master code for I$^2$C four-wire communication is as follows:

**I2C_4wire_master.c:**

```
int main (void)
{
    I2C_ConfigType  sI2C_Config = {0};
    uint32_t i,j;

    for(i=0;i<0xfff;i++);

    printf("\nRunning the I2C_4wire_master project.\n");
    printf("\nThis test should be run on EVB.\n");

    UART_WaitTxComplete(TERM_PORT);

    /* Initialize I2C module with poll mode */
    sI2C_Config.u16F = 0x1F;
    sI2C_Config.sSetting.bIntEn = 0;
    sI2C_Config.sSetting.bI2CEn = 1;

    I2C_Init(I2C0,&sI2C_Config );
#ifdef I2C0_4WIRE_ENABLE
    SIM_Enable4WireI2C0();
#endif

#ifdef I2C0_4WIRE_OUT_INVERT
    SIM_EnableI2C0OuputInvertion();
#endif

    for(i=0;i<64;i++)
    {
        u8I2C_SendBuff[i] = i;
    }
         while(1)
         {

#ifdef I2C0_4WIRE_ENABLE
        printf("\nPress any key to make Master Send data to slave with four-wire mode.\n");
#else
        printf("\nPress any key to make Master Send data to slave with 2-wire mode.\n");
#endif
        I2C_MasterSendWait(I2C0,I2C_SLAVE_ADDRESS1,&u8I2C_SendBuff[0],64);
        I2C_MasterReadWait(I2C0,I2C_SLAVE_ADDRESS1,&u8I2C_ReceiveBuff[0],64);
        printf("Read data from I2C slave:\n");
        for(i=0;i<8;i++)
        {
            for(j=0;j<8;j++)
            {
                printf("0x%x,", u8I2C_ReceiveBuff[i*8+j]);
            }
            printf("\n");
        }
        for(i=0;i<64;i++)
        {
             u8I2C_SendBuff[i] += i;
        }
        for(i=0;i<0xfffff;i++);
         }
}
```

Slave code for I$^2$C four-wire communication is as follows:

**I2C_4wire_slave.c:**

```
int main (void)
{
    uint8_t         u8I2C_ReceiveLength;
    uint32_t        i;

    I2C_ConfigType  sI2C_Config = {0};

    printf("\nRunning the I2C_4wire_slave project.\n");

    UART_WaitTxComplete(TERM_PORT);

    /* initialize I2C0 global variable and call back function*/
    I2C0_InitGlobalVariable(  );

    sI2C_Config.u16Slt = 0;
    sI2C_Config.u16F = 0x1F;
    sI2C_Config.u16OwnA1 = I2C_SLAVE_ADDRESS;
    sI2C_Config.sSetting.bIntEn = 1;
    sI2C_Config.sSetting.bI2CEn = 1;

    I2C_Init(I2C0,&sI2C_Config);

#ifdef I2C0_4WIRE_ENABLE
    SIM_Enable4WireI2C0();
#endif

#ifdef I2C0_4WIRE_OUT_INVERT
    SIM_EnableI2C0OuputInvertion();
#endif

    for(i=0;i<64;i++)
    {
        u8I2C_SendBuff[i] = i;
    }
    u8I2C_SendBuff[0] = 0xa0;
    I2C0_SlaveSend(u8I2C_SendBuff,64);
         while(1)
         {

        u8I2C_ReceiveLength = I2C0_SlaveReceive(&u8I2C_ReceiveBuff[0]);
        I2C0_SlaveSend(&u8I2C_ReceiveBuff[0],64);

        if( u8I2C_ReceiveLength )
        {
            printf("I2C received data:\n");
            for(i=0;i<u8I2C_ReceiveLength;i++)
            {
                          if( (i%8) == 0 )
                {
                   printf("\n");
                }
                printf("0x%x,", u8I2C_ReceiveBuff[i]);
            }
            printf("\n");
        }
        for(i=0;i<0xfffff;i++);
         }
}
```

## 5 References

The following references are available on freescale.com:

- KE06 Reference Manual
- KE06 Data Sheet

## 6 Rivision history

| Revision number | Date | Substantial changes |
|---|---|---|
| 0 | 03/2014 | Initial release |

**How to Reach Us:**

**Home Page:**
freescale.com

**Web Support:**
freescale.com/support

Document Number: AN4872
Rev. 0
03/2014