SPI Flash device can be programmed by I2C Master device via I2C interface between the I2C Master device and IDT923X device as shown in above Figure. In this approach, the I2C Master will configure IDT923x via I2C interface to halt IDT923x M0 CPU execution first. And then the I2C Master will directly access IDT923x internal Flash controller peripheral HW to erase, program, or read the external SPI Flash device.

Figure 1. P923x Block Diagram for an I2C Master to program IDT923x SPI Flash

This Application note applies to the following IDT products: IDTP9235, IDTP9235A, IDTP9236A, IDTP9234, IDTP9242-R, IDTP9260, and IDTP9261.

IDTP923X I2C Slave Interface

In default, IDT923x I2C interface operates as I2C Slave. An external I2C Master can write/read IDT923x internal peripheral and memory through IDT923x I2C interface. The IDT923x I2C can be accessed with the following I2C write/read sequence.

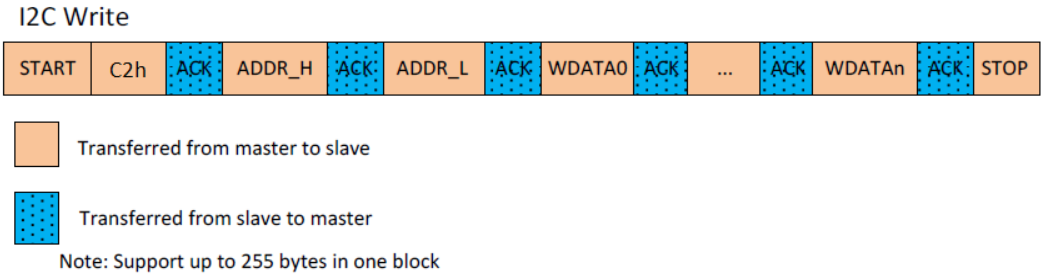


Figure 2. I2C Write sequence of P923x Product family

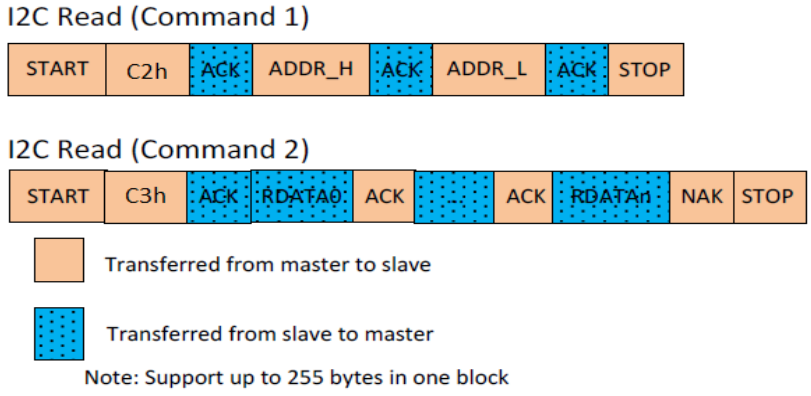


Figure 3. I2C Read sequence of P923x Product family

Flow Diagram for I2C Master to Program Flash Memory IC using SPI

The P923x family of products was designed to write to Flash Memory ICs with specific requirements. The P923x family of products can successfully write to the W25X40CLUXIG and the W25X40CLZPIG. Substitution of the Flash Memory IC should be completed with care and attention to the specifications. An IC that matches the properties of the above listed IC must be selected and the operation should be verified prior to going into production or commencing PCB builds of significant volume. A couple of key parameters required for correct operation are:

- Dual I/O Fast Read "with Continuous Read Mode". The P9235 uses M7-0 = 2XH for this mode.
- Fast Read Dual I/O (BBh)

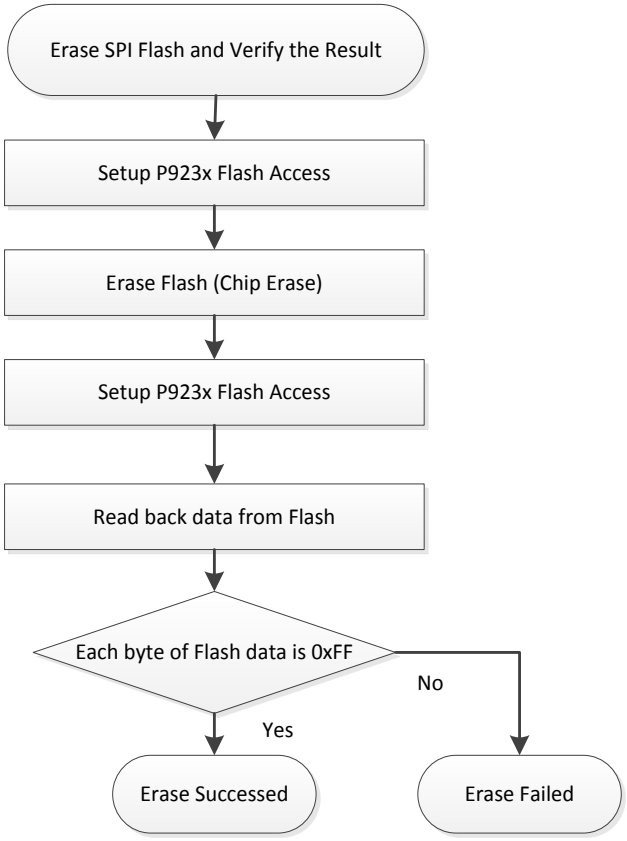


Figure 4. Erase SPI Flash and Verify Result.

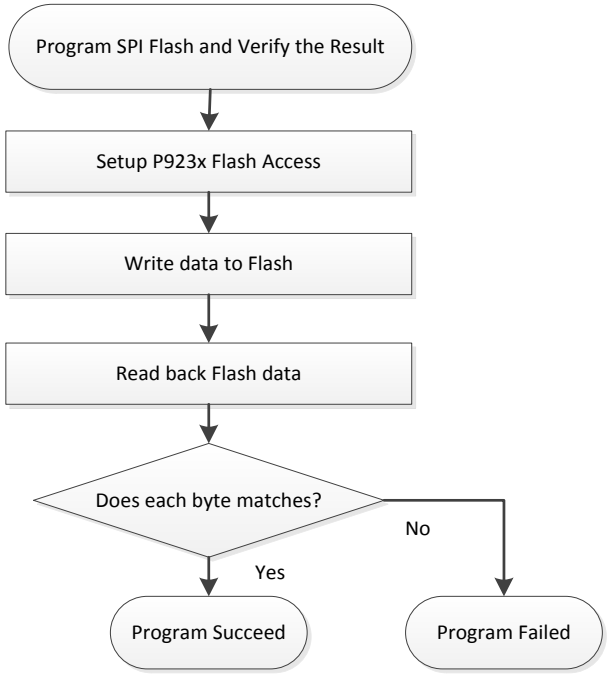


Figure 5. Program Flash and Verify Result.

Sample Code of I2C Master

Appendix A provides sample code for Erase Flash, Program Flash, and Reset P923x.
Appendix B provides I2C bus traffic captures and descriptions of Erase Flash, Program Flash, and Reset P923x processes.
Appendix C provides P923x device hardware register description related to programming Flash operations.

Appendix A:
GUI Code: Program IDT923x Flash via IDT USB-I2C dongle (For Reference Only)

A.1. Erase Flash

```
void ClearFLASH_F() //This is top level function to erase Flash
{
    int i;
    UInt16 StartAddr, CodeLen;
    UInt32 Addr_Offset;

    StartAddr = 0x800;     // (UInt16)num_StartAddr.Value;
    Addr_Offset = 0x00;    // (UInt32)num_OffsetAddr.Value;
    CodeLen = 0x100;       // (UInt16)num_CodeLen.Value;

    //Flash access
    num_StartAddr.Value = 0x800;
    num_OffsetAddr.Value = 0x00;
    num_CodeLen.Value = 0x100;

    //
    // Erase Flash
    //
    P923x_Flash_erase(SlaveAddress);
    Thread.Sleep(500);

    //
    // Verify Flash Erased
    //
    P923x_Flash_access(SlaveAddress);
    ReadMEM_RUN(StartAddr, Addr_Offset, CodeLen, 1);
    for (i = 0; i < CodeLen; i++)
    {
        if (MEMRead_Buffer[StartAddr + i] != 0xFF)
        {
            UpdateStatusLabel("Error Clearing", Color.Red);
            break;
        }
        else
            UpdateStatusLabel("Clear OK", Color.Green);
    }
}

public void P923x_Flash_erase(UInt16 SlvAddr)
{

    //
    // Setup P923x for Flash access
    //
    P923x_Flash_access(SlvAddr);

    //==== Send "Write Enable (0x06)" command to SPI Flash device ===
    // Write P923x TX_LEN register (0x1C0C), TX_RX_DATA0(0x1C0D), and TX_RX_DATA1(0x1C0E)
    // with 0x01, 0x06, 0xFF respectively, where 0x06 is Flash "Write Enable" Instruction
    // (refer to W25X40 datasheet).
    //
    I2C_WriteBuffer[0] = 0x01; //cmd length
    I2C_WriteBuffer[1] = 0x06; //write enable (Flash "Write Enable" Instruction)
    I2C_WriteBuffer[2] = 0xFF; //dummy
    FT_IIC_Write_Location(SlvAddr, 0x1C0C, 2, 3);

    //==== Send "Chip Erase (0xC7)" command to SPI Flash device ===
    // Write P923x TX_LEN register (0x1C0C), TX_RX_DATA0(0x1C0D), and TX_RX_DATA1(0x1C0E)
    // with 0x01, 0xC7, 0xFF respectively, where 0x06 is Flash "Write Enable" Instruction
    // (refer to W25X40 datasheet).
    //
```

```
        I2C_WriteBuffer[0] = 0x01; //cmd length
        I2C_WriteBuffer[1] = 0xc7; //chip erase (Flash "Chip Erase" Instruction)
        I2C_WriteBuffer[2] = 0xFF; //dummy
        FT_IIC_Write_Location(SlvAddr, 0x1C0C, 2, 3);

        //
        // Wait Flash Erase Complete
        //
        Thread.Sleep(500);
    }

    public void P923x_Flash_access(UInt16 SlvAddr)
    {
        FT_IIC_Speed_Set(100); //Configure I2C speed of I2C master to 100kHz

         //
         // Write P923x CORE_ACCKEY register (0x10DC) with 0x5A to disable write protection
         //
        I2C_WriteBuffer[0] = 0x5A; //Writekey=0x5A
        FT_IIC_Write_Location(SlvAddr, 0x10DC, 2, 1);

        //
        // Write P923x M0_CTRL register (0x1080) with 0x92 to hold M0/DMA/ACC in reset
        // Write P923x OTP_EXESEL1 register (0x1084) with 0xFF to specify OTP execution
        // Write P923x OTP_EXESEL2 register (0x1088) with 0xFF to specify OTP execution
        //
        I2C_WriteBuffer[0] = 0x92; //hold,reset M0, DMA ACC
        I2C_WriteBuffer[4] = 0xFF;
        I2C_WriteBuffer[8] = 0xFF;
        FT_IIC_Write_Location(SlvAddr, 0x1080, 2, 9);
        //==============================

        //
        // Disable Watchdog Timer (WDT) in case P923x OTP bootcode enables WDT.
        //
        I2C_WriteBuffer[0] = 0x5A;
        FT_IIC_Write_Location(SlvAddr, 0x5808, 2, 1);
        I2C_WriteBuffer[0] = 0x92;
        FT_IIC_Write_Location(SlvAddr, 0x5808, 2, 1);
        I2C_WriteBuffer[0] = 0x35;
        FT_IIC_Write_Location(SlvAddr, 0x5808, 2, 1);

        //
        // Write P923x SYS_CTRL register (0x101C) with 0x08
        // Write P923x CORE_ACCKEY register (0x10DC) with 0x5A to disable write protection
        // Write P923x M0_CTRL register (0x1080) with 0x12 to hold M0
        // Write P923x EXSPI_CONFIG register (0x10A) with 0x01 to enable APB_ACCESS
        //
        I2C_WriteBuffer[0] = 0x08; //Enable flash
        FT_IIC_Write_Location(SlvAddr, 0x101C, 2, 1);
        I2C_WriteBuffer[0] = 0x5A; //Writekey=0x5A
        FT_IIC_Write_Location(SlvAddr, 0x10DC, 2, 1);
        I2C_WriteBuffer[0] = 0x12; //Hold M0
        FT_IIC_Write_Location(SlvAddr, 0x1080, 2, 1);
        I2C_WriteBuffer[0] = 0x01; //enable APB access
        FT_IIC_Write_Location(SlvAddr, 0x10A0, 2, 1);

        //
        // Write P923x VCON_KEY register (0x586C) with 0x92 and then
        // Write P923x VCON_KEY register (0x586C) with 0x35 to enable vendor register writing
        // Write P923x LDO33_TRIM register (0x5848) with 0x00 to configure LDO33 to 3.3V
        //
        I2C_WriteBuffer[0] = 0x92; //Writekey
        FT_IIC_Write_Location(SlvAddr, 0x586C, 2, 1);
        I2C_WriteBuffer[0] = 0x35; //Writekey
        FT_IIC_Write_Location(SlvAddr, 0x586C, 2, 1);
        I2C_WriteBuffer[0] = 0x00; //3.3V
        FT_IIC_Write_Location(SlvAddr, 0x5848, 2, 1);


        //=========Set clock==================
        P923x_clock_set(SlvAddr);
        FT_IIC_Speed_Set(200);        //Configure I2C speed of I2C master to 200kHz
```

```
    //====================reset flash================
    //
    // Write P923x TX_LEN register (0x1C0C), TX_RX_DATA0(0x1C0D), and TX_RX_DATA1(0x1C0E)
    //   with 0x02, 0xFF, 0xFF respectively.
    //
    I2C_WriteBuffer[0] = 0x02; //TX_LEN = 2 bytes
    I2C_WriteBuffer[1] = 0xFF; //Flash "Continuous Read Mode Reset" Instruction (byte0)
    I2C_WriteBuffer[2] = 0xFF; //Flash "Continuous Read Mode Reset" Instruction (byte1)
    FT_IIC_Write_Location(SlvAddr, 0x1C0C, 2, 3);
}

public void P923x_clock_set(UInt16 SlvAddr)
{
    FT_IIC_Speed_Set(100);//Configure I2C speed of I2C master to 100kHz
    I2C_WriteBuffer[0] = 0x5A;
    FT_IIC_Write_Location(SlvAddr, 0x103C, 2, 1);
    I2C_WriteBuffer[0] = 0x05; //Set PLL output 60MHz
    FT_IIC_Write_Location(SlvAddr, 0x1018, 2, 1);
    I2C_WriteBuffer[0] = 0x01; //Set AHB clock, 7.5MHz
    FT_IIC_Write_Location(SlvAddr, 0x1014, 2, 1);
    I2C_WriteBuffer[0] = 0x05; //PRE_CLK 30MHz Select PLL
    FT_IIC_Write_Location(SlvAddr, 0x1010, 2, 1);
    FT_IIC_Speed_Set(200);
}

private bool FT_IIC_Write_Location(UInt16 I2C_Addr, UInt32 StartAddr, byte addrlen, byte bytes2write)
{
    UInt16 slv_address;
    bool ACK;
    int i, j, loop_cnt;
    UInt16 total_len;
    FT4222_STATUS ft4222Status = 0;
    UInt16[] sizeTransferred = new UInt16[2];
    byte[] u8wr_buf = new byte[bytes2write + addrlen];

    for (i = 0; i < addrlen; i++)
        u8wr_buf[addrlen - 1 - i] = (byte)(StartAddr >> (8 * i));
    for (i = 0; i < bytes2write; i++)
        u8wr_buf[addrlen + i] = I2C_WriteBuffer[i];
    total_len = addrlen;
    total_len += bytes2write;

    if ((BridgeConnected == true) & (_USBBridge == USB_BRIDGE.USB_FT4222)) //GUI only
    {
        ft4222Status = FT4222_I2CMaster_Write(ftHandle, I2C_Addr, u8wr_buf, total_len, sizeTransferred);

        if (ft4222Status != FT4222_STATUS.FT4222_OK)
            return false;
        else
            return true;
    }
    else
        return false;
}

private void ReadMEM_RUN(UInt16 StartAddr, UInt32 Addr_Offset, UInt16 CodeLen, UInt16 mem_view)
{
    int i, j;// = (1024 * 8);
    int err_flag = 0;
    UInt16 read_len;

    for (i = 0; i < CodeLen; )
    {
        if ((i + 64) < CodeLen)
            read_len = 64;
        else
            read_len = (UInt16)(CodeLen - i);

        if ((err_flag == 0) &&
            ReadI2C(SlaveAddress, (UInt32)(StartAddr + Addr_Offset + i), 2, read_len, 0))
        {
            for (j = 0; j < read_len; j++)
            {
```

```
                if ((StartAddr + i + j) < 32768)
                    MEMRead_Buffer[StartAddr + i + j] = _UsbInterface.I2C_ReadBuffer[j];
            }
            i += read_len;
        }
        else
        {
            err_flag = 1;
            UpdateStatusLabel("Load Failure - Check Slave ID", Color.Red); //GUI only
            break;
        }
    }
}

public bool ReadI2C(UInt16 SlvAddr, UInt32 rdaddr, byte addrlen, UInt16 bytes2read, byte DummyRdNum)
{
    if (addrlen > 4) addrlen = 1;
    for (int i = 0; i < bytes2read; i++)
        I2C_ReadBuffer[i] = 0;

    if ((rdaddr & 0x0F0000) == 0x000000) //GUI only
    { //read from flash

        //--read even byte--
        Thread.Sleep(10);
        P923x_Flash_Read(SlvAddr, rdaddr, bytes2read, 0);
        for (int i = 0; i < (bytes2read / 2); i++)
            Flash_Buffer[i * 2] = I2C_ReadBuffer[i * 2];

        //--read odd byte--
        Thread.Sleep(10);
        P923x_Flash_Read(SlvAddr, rdaddr, bytes2read, 1);
        for (int i = 0; i < (bytes2read / 2); i++)
            Flash_Buffer[i * 2 + 1] = I2C_ReadBuffer[i * 2];

        //--combine data--
        for (int i = 0; i < bytes2read; i++)
            I2C_ReadBuffer[i] = Flash_Buffer[i];
        return true;
    }
    else
        return false;
}

private bool P923x_Flash_Read(UInt16 I2C_Addr, UInt32 StartAddr, UInt16 bytes2read, byte hbyte)
{
    FT4222_STATUS ft4222Status = 0;
    byte[] rd_addr = new byte[2];
    byte[] u8rd_buf = new byte[bytes2read];
    UInt16[] sizeTransferred = new UInt16[2];

    if (hbyte == 0)
        StartAddr -= 1;

    I2C_WriteBuffer[0] = 0x1C;
    I2C_WriteBuffer[1] = 0x0C;
    I2C_WriteBuffer[2] = (byte)(bytes2read + 0x06); // cmd length
    I2C_WriteBuffer[3] = 0x0B; // fast read (Flash "Fast Read" Instruction)
    I2C_WriteBuffer[4] = (byte)((StartAddr >> 24) & 0x0ff); // addr[23:16]
    I2C_WriteBuffer[5] = (byte)((StartAddr >> 8) & 0x0ff); // addr[15:8]
    I2C_WriteBuffer[6] = (byte)(StartAddr & 0x0ff); // addr[7:0]
    I2C_WriteBuffer[7] = 0xff; //dummy
    I2C_WriteBuffer[8] = 0xff; //dummy
    I2C_WriteBuffer[9] = 0xff; //dummy

    ft4222Status = FT4222_I2CMaster_Write(ftHandle, I2C_Addr, I2C_WriteBuffer, 10, sizeTransferred);

    ft4222Status = FT4222_I2CMaster_Read(ftHandle, I2C_Addr, I2C_ReadBuffer, bytes2read,
sizeTransferred);
    return true;
}
```

A.2. Program Flash

```
private void WriteMEM_RUN()//This is top level function to program Flash
{
    UInt16 StartAddr;
    UInt32 Addr_Offset;
    UInt16 EELength;
    int mismatch_cnt = 0;
    int error_addr = 0;
    int Byte_writed = 0;
    int i;

    StartAddr = (UInt16)num_StartAddr.Value;
    Addr_Offset = (UInt32)num_OffsetAddr.Value;
    EELength = (UInt16)num_CodeLen.Value;

    //
    // Program Flash
    //
    P923x_Flash_access(SlaveAddress);
    FLASH_write(StartAddr, 0, EELength);
    Byte_writed = EELenght;

    //
    // Verify Flash Programmed Successfully
    //
    ReadMEM_RUN(StartAddr, Addr_Offset, EELenght, 1);
    mismatch_cnt = 0;
    for (i = 0; i < EELength; i++)
    {
        if ((StartAddr + i) < 32768)
        {
            if (FWFile_Buffer[StartAddr + i] != MEMRead_Buffer[StartAddr + i])
            {
                mismatch_cnt += 1;
                error_addr = StartAddr + i;
            }
        }
    }
}

private void FLASH_write(UInt16 StartAddr, UInt32 Addr_Offset, UInt16 CodeLen)
{
    //page size=256
    int i, j;
    int page_len, write_len;
    byte wbyte_num;

    page_len = 256 - (StartAddr & 0x000000ff);
    if (CodeLen < page_len)
        write_len = CodeLen;
    else
        write_len = page_len;

    for (i = 0; i < write_len; )
    {
       //
       // Copy 128-byte data from FWFile_Buffer[] buffer to I2C_WriteBuffer[] buffer
       // The FWFile_Buffer[] buffer contains the P923x firmware image loaded from GUI.
       //
        wbyte_num = 0;
        for (j = 0; j < 128; j++)
        {
            if ((i + j) < write_len)
            {
                if ((StartAddr + i + j) < 32768)
                {
                    wbyte_num += 1;
                    I2C_WriteBuffer[j] = FWFile_Buffer[StartAddr + i + j];
                }
            }
            else break;
        }
```

```
        //
        // Program I2C_WriteBuffer[] buffer data to Flash
        //
        WriteI2C(SlaveAddress, (UInt32)(StartAddr + Addr_Offset + i), 2, wbyte_num);
        i = i + j;
    }

    for (i = write_len; i < CodeLen; )
    {
        wbyte_num = 0;
        for (j = 0; j < 128; j++)
        {
            if ((i + j) < CodeLen)
            {
                if ((StartAddr + i + j) < 32768)
                {
                    wbyte_num += 1;
                    I2C_WriteBuffer[j] = FWFile_Buffer[StartAddr + i + j];
                }
            }
            else break;
        }
        WriteI2C(SlaveAddress, (UInt32)(StartAddr + Addr_Offset + i), 2, wbyte_num);
        i = i + j;
    }
}

public bool WriteI2C(UInt16 SlvAddr, UInt32 waddr, byte addrlen, byte bytes2write)
{
    if ((waddr & 0x0F0000) == 0x000000) //yes for programming Flash
    {
        P923x_Flash_Write(SlvAddr, waddr, bytes2write);
        Thread.Sleep(5);
        return true;
    }
    return false;
}

private void P923x_Flash_Write(UInt16 SlvAddr, UInt32 wdaddr, byte bytes2write)
{

    //
    // Copy I2C_WriteBuffer[] buffer data to Flash_Buffer[] buffer
    //
    for (int i = 0; i < bytes2write; i++)
        Flash_Buffer[i] = I2C_WriteBuffer[i];

    //
    // Send Flash "Write Enable" command to Flash device
    //
    I2C_WriteBuffer[0] = 0x01; // cmd length
    I2C_WriteBuffer[1] = 0x06; // write enable (Flash "Write Enable" Instruction)
    I2C_WriteBuffer[2] = 0xFF; // dummy
    FT_IIC_Write_Location(SlvAddr, 0x1C0C, 2, 3);

        //
        // Send Flash "Page Program" command and Flash_Buffer[] buffer data to Flash device
        //
    I2C_WriteBuffer[0] = (byte)(bytes2write + 0x05);    // cmd length
    I2C_WriteBuffer[1] = 0x02; // fast write (Flash "Page Program" Instruction)
    I2C_WriteBuffer[2] = (byte)((wdaddr >> 24) & 0x0ff);        // addr[23:16]
    I2C_WriteBuffer[3] = (byte)((wdaddr >> 8) & 0x0ff);         // addr[15:8]
    I2C_WriteBuffer[4] = (byte)(wdaddr & 0x0ff);                // addr[7:0]
    for (int j = 0; j < bytes2write; j++)
        I2C_WriteBuffer[j + 5] = Flash_Buffer[j];

    I2C_WriteBuffer[bytes2write + 5] = 0xff;
    FT_IIC_Write_Location(SlvAddr, 0x1C0C, 2, (byte)(bytes2write + 0x06));
}
```

A.3. Reset P923x

P923x_reset() function below will disable P923x Watchdog timer if its input variable "type" sets to 1 and then reset P923x M0 CPU. Since P923x Watchdog timer must be disabled before programming Flash, it is recommended to call this function (type=1) after erasing Flash to disable watchdog timer and call this function (type= 0) again after programming Flash to restart P923x.

```
public void P923x_reset(UInt16 SlvAddr, UInt16 type)
{
    if(type == 1)
        hw_watchdog_clr();//Disable P923x Watchdog timer

    FT_IIC_Speed_Set(100);
    P923x_disable_PWM(SlvAddr);
    I2C_WriteBuffer[0] = 0x92;
    FT_IIC_Write_Location(SlvAddr, 0x586C, 2, 1);
    I2C_WriteBuffer[0] = 0x35; //Enable write key
    FT_IIC_Write_Location(SlvAddr, 0x586C, 2, 1);
    I2C_WriteBuffer[0] = 0x00; //Clear start flag
    FT_IIC_Write_Location(SlvAddr, 0x585C, 2, 1);
    I2C_WriteBuffer[0] = 0x5A; // Writekey= 0x5A
    FT_IIC_Write_Location(SlvAddr, 0x10DC, 2, 1);
    I2C_WriteBuffer[0] = 0x00;
    FT_IIC_Write_Location(SlvAddr, 0x108C, 2, 1);
    I2C_WriteBuffer[0] = 0x80; //reset M0
    FT_IIC_Write_Location(SlvAddr, 0x1080, 2, 1);
    Thread.Sleep(1);
}

public void hw_watchdog_clr()
{
    UInt16 timeout;
    I2C_ReadBuffer[0] = 0x10;
    timeout = 0;
    bool f4222state = true;

    //
    // This "while" loop disables P923x hardware watchdog timer
    //
    while (((I2C_ReadBuffer[0]&0x10) == 0x10) && f4222state && (timeout < 3))
    {
        // Write 0x92 followed by 0x35 to VCON_KEY register (0x586C) to enable
        // Power Control Block registers writable
        I2C_WriteBuffer[0] = 0x92;
        f4222state = FT_IIC_Write_Location(0x61, 0x586C, 2, 1);
        if (f4222state != true) continue;
        I2C_WriteBuffer[0] = 0x35; //Enable write key
        f4222state = FT_IIC_Write_Location(0x61, 0x586C, 2, 1);
        if (f4222state != true) continue;

        I2C_WriteBuffer[0] = 0x10; //hold M0
        f4222state = FT_IIC_Write_Location(0x61, 0x1080, 2, 1);
        if (f4222state != true) continue;
        I2C_WriteBuffer[0] = 0x55; //Enable UKEY
        f4222state = FT_IIC_Write_Location(0x61, 0x583C, 2, 1);
        if (f4222state != true) continue;
        I2C_WriteBuffer[0] = 0xA5;
        f4222state = FT_IIC_Write_Location(0x61, 0x583C, 2, 1);
        if (f4222state != true) continue;
        I2C_WriteBuffer[0] = 0x00; //clear PD enable
        f4222state = FT_IIC_Write_Location(0x61, 0x5804, 2, 1);
        if (f4222state != true) continue;

        //================Clear WDT==========
        I2C_WriteBuffer[0] = 0x5A; //clear wdt
        f4222state = FT_IIC_Write_Location(0x61, 0x5808, 2, 1);
        if (f4222state != true) continue;
        I2C_WriteBuffer[0] = 0x92; //clear wdt
        f4222state = FT_IIC_Write_Location(0x61, 0x5808, 2, 1);
        if (f4222state != true) continue;
        I2C_WriteBuffer[0] = 0x35; //clear wdt
        f4222state = FT_IIC_Write_Location(0x61, 0x5808, 2, 1);
        if (f4222state != true) continue;
        Thread.Sleep(1);
```

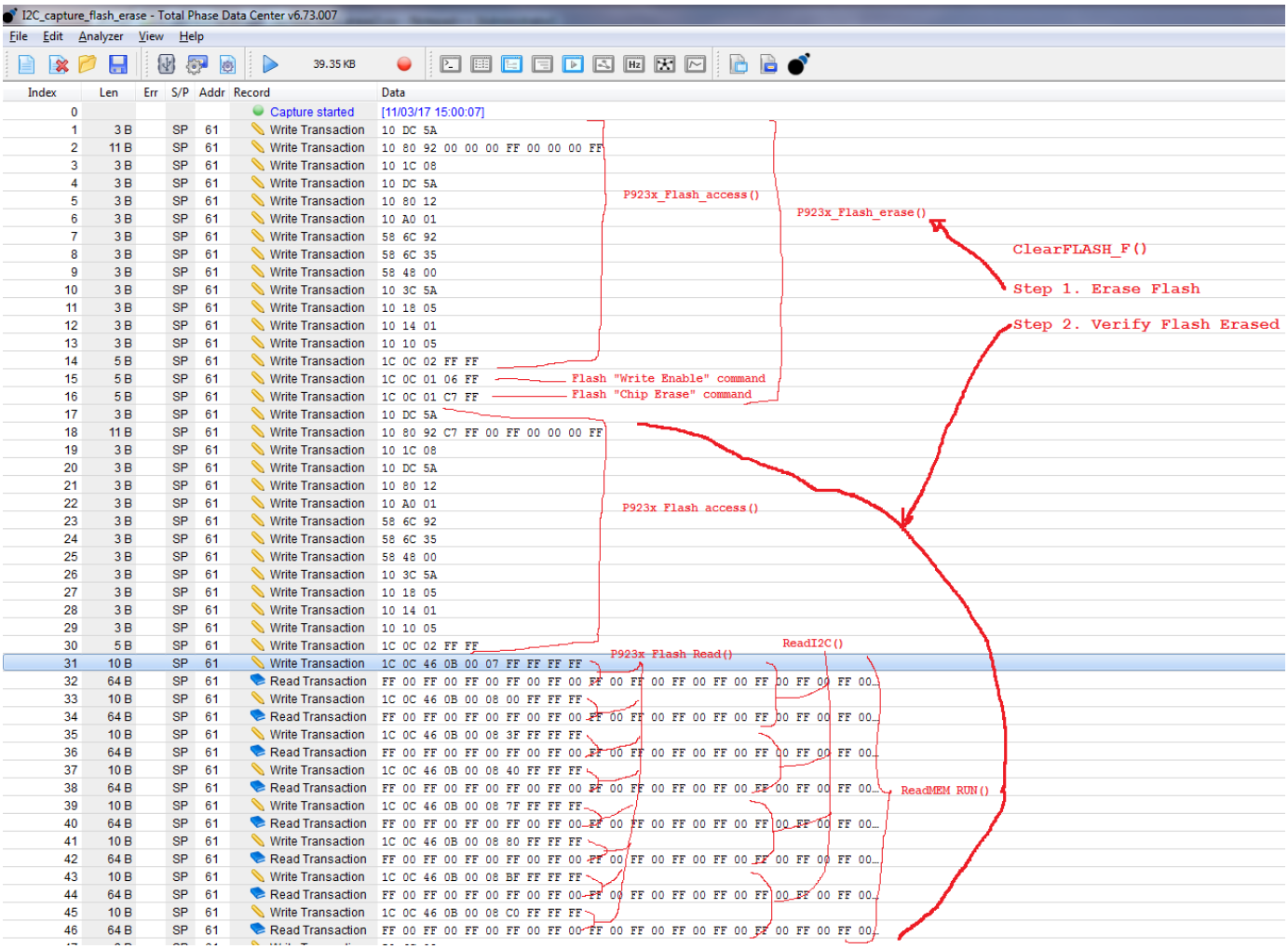
```
        f4222state = FT_IIC_Read_Location(0x61, 0x5808, 2, 1); //read data to I2C_ReadBuffer[]
        if (f4222state != true) continue;

        timeout += 1;
    }
}

public void P923x_disable_PWM(UInt16 SlvAddr)
{
    I2C_WriteBuffer[0] = 0x80; //reset FPWM
    FT_IIC_Write_Location(SlvAddr, 0x3400, 2, 1);
    I2C_WriteBuffer[0] = 0x80; //reset VPWM
    FT_IIC_Write_Location(SlvAddr, 0x3800, 2, 1);
}
```

Appendix B:
I2C Bus Traffic Captures with Beagle I2C/SPI Protocol Analyzer (For Reference Only)

B.1. I2C Bus Traffic Capture of "ClearFLASH_F()" Execution



Index	Len	Err	S/P	Addr	Record	Data
0					Capture started	[11/03/17 15:00:07]
1	3 B		SP	61	Write Transaction	10 DC 5A
2	11 B		SP	61	Write Transaction	10 80 92 00 00 00 FF 00 00 00 FF
3	3 B		SP	61	Write Transaction	10 1C 08
4	3 B		SP	61	Write Transaction	10 DC 5A
5	3 B		SP	61	Write Transaction	10 80 12
6	3 B		SP	61	Write Transaction	10 A0 01
7	3 B		SP	61	Write Transaction	58 6C 92
8	3 B		SP	61	Write Transaction	58 6C 35
9	3 B		SP	61	Write Transaction	58 48 00
10	3 B		SP	61	Write Transaction	10 3C 5A
11	3 B		SP	61	Write Transaction	10 18 05
12	3 B		SP	61	Write Transaction	10 14 01
13	3 B		SP	61	Write Transaction	10 10 05
14	5 B		SP	61	Write Transaction	1C 0C 02 FF FF
15	5 B		SP	61	Write Transaction	1C 0C 01 06 FF
16	5 B		SP	61	Write Transaction	1C 0C 01 C7 FF
17	3 B		SP	61	Write Transaction	10 DC 5A
18	11 B		SP	61	Write Transaction	10 80 92 C7 FF 00 FF 00 00 00 FF
19	3 B		SP	61	Write Transaction	10 1C 08
20	3 B		SP	61	Write Transaction	10 DC 5A
21	3 B		SP	61	Write Transaction	10 80 12
22	3 B		SP	61	Write Transaction	10 A0 01
23	3 B		SP	61	Write Transaction	58 6C 92
24	3 B		SP	61	Write Transaction	58 6C 35
25	3 B		SP	61	Write Transaction	58 48 00
26	3 B		SP	61	Write Transaction	10 3C 5A
27	3 B		SP	61	Write Transaction	10 18 05
28	3 B		SP	61	Write Transaction	10 14 01
29	3 B		SP	61	Write Transaction	10 10 05
30	5 B		SP	61	Write Transaction	1C 0C 02 FF FF
31	10 B		SP	61	Write Transaction	1C 0C 46 0B 00 07 FF FF FF FF
32	64 B		SP	61	Read Transaction	FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00...
33	10 B		SP	61	Write Transaction	1C 0C 46 0B 00 08 00 FF FF FF
34	64 B		SP	61	Read Transaction	FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00...
35	10 B		SP	61	Write Transaction	1C 0C 46 0B 00 08 3F FF FF FF
36	64 B		SP	61	Read Transaction	FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00...
37	10 B		SP	61	Write Transaction	1C 0C 46 0B 00 08 40 FF FF FF
38	64 B		SP	61	Read Transaction	FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00...
39	10 B		SP	61	Write Transaction	1C 0C 46 0B 00 08 7F FF FF FF
40	64 B		SP	61	Read Transaction	FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00...
41	10 B		SP	61	Write Transaction	1C 0C 46 0B 00 08 80 FF FF FF
42	64 B		SP	61	Read Transaction	FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00...
43	10 B		SP	61	Write Transaction	1C 0C 46 0B 00 08 BF FF FF FF
44	64 B		SP	61	Read Transaction	FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00...
45	10 B		SP	61	Write Transaction	1C 0C 46 0B 00 08 C0 FF FF FF
46	64 B		SP	61	Read Transaction	FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00 FF 00...

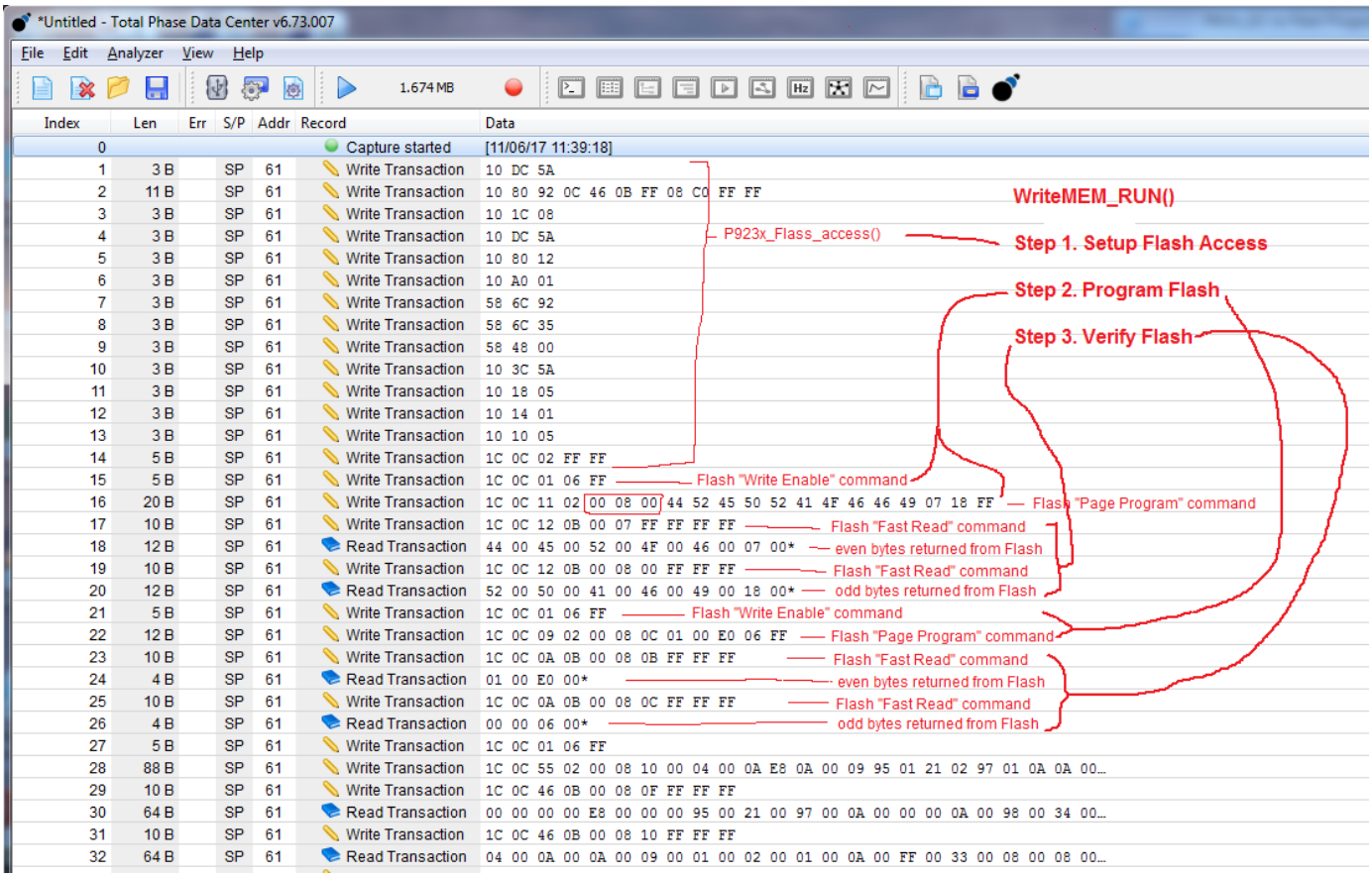
Notes:

- Line of Index 14: 1C 0C 02 FF FF --- Send Flash "Continuous Read Mode Reset" command to Flash device
  - 0x1C0C is P923x "TX_LEN" register address. (Refer to External Flash Interface of Appendix C.)
  - 0x02 means to write 2-byte data 0xFFFF from P923x "TX_RX_DATA" buffer to Flash device.
  - 0xFFFF is Flash "Continuous Read Mode Reset" Instruction (Refer to "W25X40CL" Flash Datasheet).

- Line of Index 15: 1C 0C 01 06 FF --- Send Flash "Write Enable" command to Flash device
  - 0x1C0C is P923x "TX_LEN" register address. (Refer to External Flash Interface of Appendix C.)
  - 0x01 means to write 1-byte data 0x06 from P923x "TX_RX_DATA" buffer to Flash device.
  - 0x06 is Flash "Write Enable" Instruction (Refer to "W25X40CL" Flash Datasheet). I2C writes it to P923x "TX_RX_DATA" buffer.
  - 0xFF is dummy byte which won't be sent to Flash.

- Line of Index 16: 1C 0C 01 C7 FF --- Send Flash "Chip Erase" command to Flash device
  - 0x1C0C is P923x "TX_LEN" register address. (Refer to External Flash Interface of Appendix C.)
  - 0x01 means to write 1-byte data 0xC7 from P923x "TX_RX_DATA" buffer to Flash device.
  - 0xC7 is Flash "Chip Erase" Instruction (Refer to "W25X40CL" Flash Datasheet). I2C writes it to P923x "TX_RX_DATA" buffer.
  - 0xFF is dummy byte which won't be sent to Flash.

B.2. I2C Bus Traffic Capture of "WriteMEM_RUN()" Execution



Index	Len	Err	S/P	Addr	Record	Data
0					Capture started	[11/06/17 11:39:18]
1	3 B		SP	61	Write Transaction	10 DC 5A
2	11 B		SP	61	Write Transaction	10 80 92 0C 46 0B FF 08 C0 FF FF
3	3 B		SP	61	Write Transaction	10 1C 08
4	3 B		SP	61	Write Transaction	10 DC 5A
5	3 B		SP	61	Write Transaction	10 80 12
6	3 B		SP	61	Write Transaction	10 A0 01
7	3 B		SP	61	Write Transaction	58 6C 92
8	3 B		SP	61	Write Transaction	58 6C 35
9	3 B		SP	61	Write Transaction	58 48 00
10	3 B		SP	61	Write Transaction	10 3C 5A
11	3 B		SP	61	Write Transaction	10 18 05
12	3 B		SP	61	Write Transaction	10 14 01
13	3 B		SP	61	Write Transaction	10 10 05
14	5 B		SP	61	Write Transaction	1C 0C 02 FF FF
15	5 B		SP	61	Write Transaction	1C 0C 01 06 FF
16	20 B		SP	61	Write Transaction	1C 0C 11 02 00 08 00 44 52 45 50 52 41 4F 46 46 49 07 18 FF
17	10 B		SP	61	Write Transaction	1C 0C 12 0B 00 07 FF FF FF FF
18	12 B		SP	61	Read Transaction	44 00 45 00 52 00 4F 00 46 00 07 00*
19	10 B		SP	61	Write Transaction	1C 0C 12 0B 00 08 00 FF FF FF
20	12 B		SP	61	Read Transaction	52 00 50 00 41 00 46 00 49 00 18 00*
21	5 B		SP	61	Write Transaction	1C 0C 01 06 FF
22	12 B		SP	61	Write Transaction	1C 0C 09 02 00 08 0C 01 00 E0 06 FF
23	10 B		SP	61	Write Transaction	1C 0C 0A 0B 00 08 0B FF FF FF
24	4 B		SP	61	Read Transaction	01 00 E0 00*
25	10 B		SP	61	Write Transaction	1C 0C 0A 0B 00 08 0C FF FF FF
26	4 B		SP	61	Read Transaction	00 00 06 00*
27	5 B		SP	61	Write Transaction	1C 0C 01 06 FF
28	88 B		SP	61	Write Transaction	1C 0C 55 02 00 08 10 00 04 00 0A E8 0A 00 09 95 01 21 02 97 01 0A 0A 00…
29	10 B		SP	61	Write Transaction	1C 0C 46 0B 00 08 0F FF FF FF
30	64 B		SP	61	Read Transaction	00 00 00 00 E8 00 00 00 95 00 21 00 97 00 0A 00 00 00 0A 00 98 00 34 00…
31	10 B		SP	61	Write Transaction	1C 0C 46 0B 00 08 10 FF FF FF
32	64 B		SP	61	Read Transaction	04 00 0A 00 0A 00 09 00 01 00 02 00 01 00 0A 00 FF 00 33 00 08 00 08 00…

Notes:

- Line of Index 16: 1C 0C 11 02 00 08 00 44 52 45 50 52 41 4F 46 46 49 07 18 FF --- Send Flash "Page Program" command to Flash device
  - 0x1C0C is P923x "TX_LEN" register address. (Refer to External Flash Interface of Appendix C.)
  - 0x11 means 17-byte data 02 00 08 … 18 FF in P923x "TX_RX_DATA" buffer. (Refer to External Flash Interface of Appendix C.)
  - 0x02 is Flash "Page Program" Instruction (Refer to "W25X40CL" Flash Datasheet)
  - 0x000800 is 3-byte address of Flash "Page Program" command.
  - 44 52 45 50 52 41 4F 46 46 49 07 18 FF are 13 bytes of data to be written to Flash

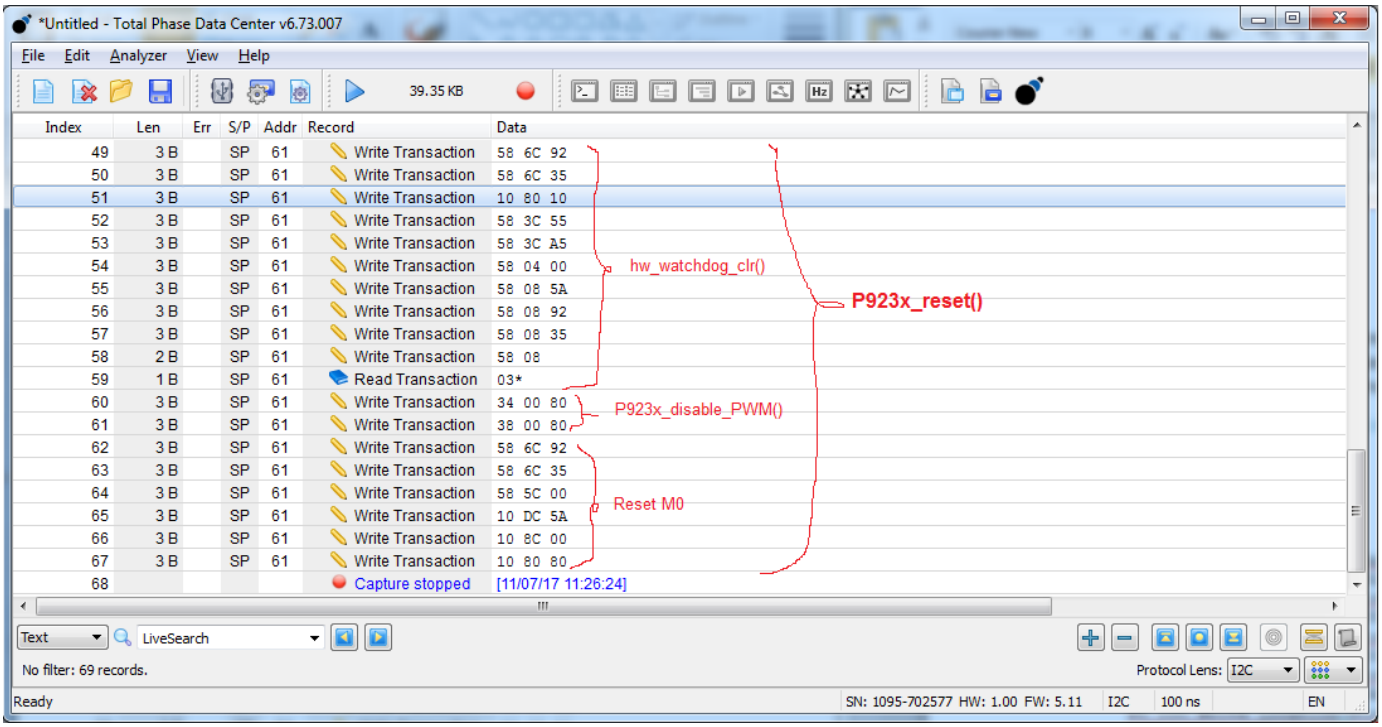
- Line of Index 17: 1C 0C 12 0B 00 07 FF FF FF --- Send Flash "Fast Read" command to Flash device to read even numbers of bytes
  - 0x1C0C is P923x "TX_LEN" register address. (Refer to External Flash Interface of Appendix C.)
  - 0x12 means to read 12 bytes (18 – 5 = 12. The size of "Fast Read" command is 5 bytes) of data from Flash device. (Refer to External Flash Interface of Appendix C.)
  - 0B 00 07 FF FF is 5-byte Flash "Fast Read" command (Refer to "W25X40CL" Flash Datasheet): 1 byte Instruction (0x0B), 3-byte address (0x0007FF), and a dummy byte (0xFF).
  - FF FF are dummy bytes which won't be sent to Flash

- Line of Index 18: 44 00 45 00 52 00 4F 00 46 00 07 00 --- data read from Flash
  - 44 45 52 4F 46 07 are the even numbers of data returned from the Flash read
  - The odd numbers of data (zeros) should be ignored.

- Line of Index 19: 1C 0C 12 0B 00 08 00 FF FF --- Send Flash "Fast Read" command to Flash device to read odd numbers of bytes
  - 0x1C0C is P923x "TX_LEN" register address. (Refer to External Flash Interface of Appendix C.)
  - 0x12 means to read 12 bytes (18 – 5 = 12. The size of "Fast Read" command is 5 bytes) of data from Flash device. (Refer to External Flash Interface of Appendix C.)
  - 0B 00 08 00 FF is 5-byte Flash "Fast Read" command (Refer to "W25X40CL" Flash Datasheet): 1 byte Instruction (0x0B), 3-byte address (0x000800), and a dummy byte (0xFF).
  - FF FF are dummy bytes which won't be sent to Flash
  -

- Line of Index 20: 52 00 50 00 41 00 46 00 49 00 18 00 --- data read from Flash
  - 52 50 41 46 49 18 are the odd numbers of data returned from the Flash read
  - The even numbers of data (zeros) should be ignored.

B.3. I2C Bus Traffic Capture of "P923x_reset()" Execution



Appendix C:

P923X Peripheral Registers related to Programming Flash

Configure Register

Debug-access base address: 0x1000

System Clock Select Register (SYS_CKSEL)

Address offset: 0x0010 (Protected write)

Bit	Label	Default	Type	Description
15:3	RSV	0x0	RO	Reserved
2:1	SRC_PREDIV	0x0	RW	0x0: PRE_CLK=SRC_CLK 0x1: PRE_CLK=SRC_CLK/2 0x2: PRE_CLK=SRC_CLK/4 0x3: PRE_CLK=SRC_CLK/8
0	SRC_CLKSEL	0x0	RW	0: SRC_CLK = OSC6M 1: SRC_CLK = PLL_CLK

System Clock Divider Register (SYS_CKDIV)

Address offset: 0x0014 (Protected write)

Bit	Label	Default	Type	Description
15:8	RSV	0x0	RO	Reserved
7	CLKGEN_MODE	0x0	RW	
6:0	AHB_CDIV[6:0]	0xA4	RW	When bit7=0 AHB_CLK=PRE_CLK/(AHB_CDIV +1) When bit7=1 AHB_CLK=PRE_CLK/(AHB_CDIV[3:0] +1) Clock high pulse width = (AHB_CDIV[6:4] +1)

PLL Control Register (PLL_CONTROL)

Address offset: 0x0018 (Protected write)

Bit	Label	Default	Type	Description
15:3	RSV	0x0	RO	Reserved
2	PLL_EN	0x0	RW	1: PLL enable 0: PLL power down
1:0	PLL_ODIV	0x0	RW	0x0: PLL_CLK=VCO/2=120MHz 0x1: PLL_CLK=VCO/4=60MHz 0x2: PLL_CLK=VCO/3=80MHz 0x3: PLL_CLK=VCO/6=40MHz Note: VCO=240MHz

System Control Register (SYS_CTRL)

Address offset: 0x001C (Protected write)

Bit	Label	Default	Type	Description
15:4	RSV	0x0	RO	Reserved
3	SRC_PREDIV_SYNC	0x1	RW	1: SRC_PREDIV always updated 0: SRC_PREDIV updated only after ongoing flash access finished
2	FLASH_SPI_PD	0x1	RW	0: SPI clock = PRE_CLK 1: SPI clock stopped Default is unexpected “1” !!!

1	RSV	0x1	RW	
0	RSV	0x0	RW	

Configure Access Key Register (SYS_ACCKEY)

Address offset: 0x003C

Bit	Label	Default	Type	Description
7:1	RSV	0x0	RO	
0	WRITE_PROTECT	0x0	RW	Write 8'h5A to enable writing registers in offset 0x00 ~ 0x34, write other value to disable writing

Core Configure Register

Debug-access base address: 0x1000

M0 control Register (M0_CTRL)

Address offset: 0x0080

Bit	Label	Default	Type	Description
15:8	RSV	0x0	RO	Reserved
7	M0_SWRST	0x0	RWSC	M0 Soft reset
6:5	RSV	0x0	RO	Reserved
4	M0_HOLD	0x0	RW	1: hold M0 execute
3:2	RSV	0x0	RW	Reserved
1	OTP_SEL_MODE	0x0	RW	0: Full OTP range selected by latch status of pin GPIOA6 (1 select OTP, 0 select FLASH) 1: OTP selection controlled by register OTP_EXESEL1 and 2
0	OTP_DMA_ACC_EN	0x0	RW	

OTP Execute Select 1(OTP_EXESEL1)

Address offset: 0x0084 (Protected write)

Bit	Label	Default	Type	Description
15:8	RSV	0x0	RO	Reserved
7:1	CODE_OTP_SEL [7:1]	0x0	RW	Specify each 2K code location. "1" selects OTP; "0" selects external flash. For example, codes from 2K to 4K are in OTP @ bit1 = 1; codes from 2K to 4K are in flash @ bit1 = 0.
0	FLASH_BOOT_SEL	0x0	RO	Invert value of latched GPIOA6. Specify the location of the code from 0K to 2K. "1" means boot from FLASH; "0" selects OTP.

OTP Execute Select 2(OTP_EXESEL2)

Address offset: 0x0088 (Protected write)

Bit	Label	Default	Type	Description
15:8	RSV	0x0	RO	Reserved
7:0	CODE_OTP_SEL [15:8]	0x0	RW	Codes above 30K are in OTP when CODE_OTP_SEL[15] = 1, else codes above 30K are in flash.

SRAM Execute Configure Register (SRAMEXE_CONFIG)

Address offset: 0x008C (Protected write)

Bit	Label	Default	Type	Description
15:8	RSV	0x0	RO	reserved
7	SRAM_EXE_EN	0x0	RW	1:Enable code mapping to SRAM When address matched, access from SRAM instead of OTP and FLASH
6	RSV	0x0	RW	reserved
5:4	SRAM_EXE_SPACE	0x0	RW	0x0: mapping whole 4K code to SRAM Haddr[14:12]==SRAM_EXECFG[3:1] Mapping: OTP:[SRAM_EXECFG[3:1]*4K, SRAM_EXECFG[3:1]*4K_4K) → SRAM: [0K, 4K) Not used!! 0x1: mapping 3.5K code to SRAM Haddr[14:12]==SRAM_EXECFG[3:1]; SRAM_addr[12:9] += 4'h1; (haddr[11:9] < 3'h7) Mapping: OTP: [SRAM_EXECFG[3:1]*4K, SRAM_EXECFG[3:1]*4K+3.5K) → SRAM: [0.5K, 4K) 0x2: Mapping 3K code to SRAM Haddr[14:12]==SRAM_EXECFG[3:1]; SRAM_addr[12:10] += 3'h1; (haddr[11:10] < 2'h3) Mapping: OTP: [SRAM_EXECFG[3:1]*4K, SRAM_EXECFG[3:1]*4K+3K) → SRAM: [1K,4K) 0x3:Mapping 2K code to SRAM Haddr[14:11]==SRAM_EXECFG[3:0]; SRAM_addr[12:11] += ~ Haddr[11] ;(haddr[11] = 1'h0) Mapping: OTP: [SRAM_EXECFG[3:2]*8K, SRAM_EXECFG[3:2]*8K+2K) → SRAM: [2K, 4K)
3:0	SRAM_EXECFG	0x0	RW	

EXSPI Configure Register (EXSPI_CONFIG)

Address offset: 0x00A0 (Protected write) (TBD)

Bit	Label	Default	Type	Description
15:6	RSV	0x0	RO	reserved
6	SRC_PREDIV_SYNC			SRC_PREDIV?
5:4	SDI_BUF_MODE	0x0	RW	00: No latch inserted 01: neg edge latch inserted 10: pos egde latch inserted 11: both edge latched inserted
3	DELAYED_LATCH	0x0	RW	
2	PRE_FETCH_EN	0x0	RW	
1	BLOCK_READ_EN	0x0	RW	
0	APB_ACC_EN	0x0	RW	0: CODE_FETCH MODE 1: APB_ACCESS Enable

Configure Access Key Register (CORE_ACCKEY)

Address offset: 0x00DC

Bit	Label	Default	Type	Description
7:1	RSV	0x0	RO	
0	WRITE_PROTECT	0x0	RW	Write 8'h5A to disable write protect, Write other value to enable write protect

FPWM Registers

Debug-access base address: 0x3400

FPWM Reset register (FPWM_RSTEN)

Address offset: 0x0000

Bit	Label	Default	Type	Description
15:8	RSV	0	RO	Reserved
7	PWM_SRST	0	RWSC	Software reset
6	RSV	0	RO	Reserved
5	PWM_GINT_EN	0	RW	0: disable, 1: global interrupt enable
4	PWM_MOD_EN_S	0	RW	0:disable,1:Modulation enable
3	PWM_FRAC_EN_S	0	RW	0: disable, 1: enable
2	PWM_FRAC_TYPE_S	0	RW	0: period, 1:duty
1	PWM_DITH_EN_S	0	RW	0:dither disable,1:dither enable
0	PWM_EN	0	RW	0: disable, 1:enable

FPWM Control Rregister (FPWM_SYNC)

Address offset: 0x0008

Bit	Label	Default	Type	Description
15:8	RSV	0	RO	Reserved
7	PWM_PARAM_SYNC	0	RWSC	Register with name *_S will synchronous load with PWM period
6:0	RSV	0	RO	Reserved

Power Control Registers

Debug-access base address: 0x5800

Power Down Control Register (POWER_DOWN)

Address offset: 0x04 (UKEY)

Bit	Label	Default	Type	Description
7	DIRECT_PD_EN	0	RWSC	Enter power down when this register
6:5	RSV	0	RO	Reserved
4	REF_BIAS_PD	0	RW	Set 1b to power-down reference/bias
3	BUCK_PD	0	RW	Set 1b to power-down buck 5v
2	LDO33_PD	0	RW	Set 1b to power-down LDO33
1	LDO18_PD	0	RW	Set 1b to power-down LDO18
0	OSC6M_PD	0	RW	Set 1b to power down OSC6M

Watchdog Enable/Kick Register (WDG_EN)

Address offset: 0x08 (UKEY)

- Enable Watchdog
  Write 0x5A and 0xA5 to enable watchdog.
- Disable Watchdog
  Write 0x5A, 0x92 and 0x35 to disable watchdog.
- Kick Watchdog
  When WDG_CTRL.KICK_MODE is "0", write 0x5A and 0xAA to kick watchdog to reset the watchdog timer.
  When WDG_CTRL.KICK_MODE is "1", write 0xAA to kick watchdog to reset the watchdog timer.

For read-out, the bits are as below:

Bit	Label	Default	Type	Description
7:5	RSV	0	R	Reserved
4	WD_EN	0	R	"1" indicates the watchdog is enabled.
3:2	RSV	0	R	Reserved
1	WD_RST_FLAG	0	R	
0	WD_INT_FLAG	0	R	Cleared by watchdog disabling

LDO33 Trim Register (LDO33_TRIM)

Address offset: 0x48 (VKEY)

Bit	Label	Default	Type	Description
7:6	LDO33_TRIM	0	RW	00:3.3v, 01:3.6v, 10:3.9v, 11:4.2v.
5:0	RXDET_CNT		RW	Rx detection count

User Key Register (USR_KEY)

Address offset: 0x3C

Bit	Label	Default	Type	Description
7:0	**USR_KEY**	0x0	RW	Write 0x55 followed by 0xA5 to set UCON_STS.UKEY to enable user register. This register always can be written.

Test 5V Register (TEST_5V)

Address offset: 0x5C (VKEY)

Bit	Label	Default	Type	Description
7:6	RSV	0	RO	Reserved.
5	TM	0	RW	Write 1b to set 5v test mode. In 5v test mode, BIAS/BUCK_5V/LDO18/LDO33/OSC6M can be enabled / disabled by writing registers.
4	START_FLAG	0	RW	Power up flag
3:0	STATUS_FLAG	0	RW	System Status flag

Vendor Control Key Register (VCON_KEY)

Address offset: 0x6C

Bit	Label	Default	Type	Description
7:0	**VENDOR_KEY**	0x0	W	Write 0x92 followed by 0x35 to set VCON_STS.VKEY to enable vendor register writing. This register always can be written.

External Code Control Register (External Flash Interface)

Debug-access base address: 0x1C00

Test Summary: Access external flash via I2C. For single byte access, the unexpected “00” byte will be sent before sending the real data. A workaround is to use I2C block write to access external flash.

APB_ACC_EN bit of EXSPI_CONFIG should set first before operation

Tx length Register (TX_LEN)

Address offset: 0x000C

Bit	Label	Default	Type	Description
7:0	TX_LENGTH	0x0	RW	N bytes

Data Register (TX_RX_DATA)

Address offset: 0x000D~0xff

Bit	Label	Default	Type	Description
7:0	TX_DATA	0x0	RW	TX/RX data (N-byte data)

Revision History

Revision Date	Description of Change
V1.0, August 1, 2017	Initial release of AN-977: P923x Programming with I2C to Flash Memory using SPI.
V1.1, August 1, 2017	Added ReadI2C() function and P923x_Flash_Read() function in Appendix A. Added some comments to source code in Appendix A. Changed “Page Erase” to Chip Erase” in Figure 4.
V1.2, Nov. 6, 2017	Added A.3. Section: Reset P923x() in Appendix A. Added Appendix B. Added Appendix C.
V1.3, Nov. 7, 2017	Updated P923x_reset() function in Appendix A. Added P923x register description in Appendix C for P923x_reset() function.
V1.4, Nov. 21, 2017	Removed unused function FT_IIC_Set_StartAddr() in Appendix A.
V1.5, Nov. 29, 2017	Added code of disabling WDT in P923x_Flash_access() function in Appendix A.

Corporate Headquarters
6024 Silver Creek Valley Road
San Jose, CA 95138
www.IDT.com

Sales
1-800-345-7015 or 408-284-8200
Fax: 408-284-2775
www.IDT.com/go/sales

Tech Support
www.IDT.com/go/support