VL53L0 I2C Verification App Note

Version 0.1

Revision History

	Version	Reason/Changes
03 Jun 2016	0.1	First draft of the I2C verification

Commented [JEK1]:

Table of Contents

1. Overview

The API requires that the following functions be written for the host processor by the customer:

- *VL53L0X_WriteMulti()*
- *VL53L0X_ReadMulti()*
- *VL53L0X_WrByte()*
- *VL53L0X_WrWord()*
- *VL53L0X_WrDWord()*
- *VL53L0X_RdByte()*
- *VL53L0X_RdWord()*
- *VL53L0X_RdDWord()*

This document describes a technique to verify the correctness of these functions.

2. Prototypes

```
int rd_write_verification( uint8_t addr);
void i2c_test(void);
```

3. Main Function Call

Put this at the top of your code – just after reading the device ID etc.

```
I2c_test();
```

4. Test Code

```
void i2c_test(void)
{
    int err_count = 0;
    int expected_value = 0;

    uint8_t buff[4] = {0x11,0x22,0x33,0x44};
    uint8_t ChipID[4];
    int i=0;

    for (i=0; i<4; i++){ VL53L0_RdByte(&MyDevice, 0xC0+i, &ChipID[i]); }
    expected_value = ChipID[0]<<24 | ChipID[1]<<16 | ChipID[2]<<8 | ChipID[3];
    if(rd_write_verification(0xc0, expected_value) <0) err_count++;    // check the chip ID

    VL53L0_WriteMulti(&MyDevice, 0x4,  buff, 4);                      // check WriteMulti
    if(rd_write_verification(0x4, 0x11223344) <0) err_count++;

    VL53L0_WrDWord(&MyDevice, 0x4, 0xffeeddcc);                       // check WrDWord
    if(rd_write_verification(0x4, 0xffeeddcc) <0) err_count++;

    VL53L0_WrWord(&MyDevice, 0x4, 0x5566);                            // check WrWord
    VL53L0_WrWord(&MyDevice, 0x6, 0x7788);
    if(rd_write_verification(0x4, 0x55667788) <0) err_count++;

    for (i=0; i<4; i++){ VL53L0_WrByte (&MyDevice, 0x04+i, &buff[i]); }
    if(rd_write_verification(0x4,0x11223344) <0) err_count++;
    if(err_count>0)
    {
          printf("i2c test failed - please check it\n");
    }
}
int rd_write_verification( uint8_t addr, uint32_t expected_value)
{
    uint8_t bytes[4],mbytes[4];
    uint16_t words[2];
    uint32_t dword;
    VL53L0_ReadMulti(&MyDevice, addr, mbytes, 4);
    for (i=0; i<4; i++){ VL53L0_RdByte(&MyDevice, addr+i, &bytes[i]); }
    for (i=0; i<2; i++){ VL53L0_RdWord(&MyDevice, addr+i*2, &words[i]); }
    VL53L0_RdDWord(&MyDevice, addr, &dword);

    printf("expected   = %8x,\n",expected_value);
    printf("read_multi = %2x, %2x, %2x, %2x\n", mbytes[0],mbytes[1],mbytes[2],mbytes[3]);
    printf("read_bytes = %2x, %2x, %2x, %2x\n", bytes[0],bytes[1],bytes[2],bytes[3]);
    printf("read words = %4x, %4x\n",words[0],words[1]);
    printf("read dword = %8x\n",dword);

    if((mbytes[0]<<24 | mbytes[1]<<16 | mbytes[2]<<8 | mbytes[3]) != expected_value) return (-1);
    if((bytes[0]<<24 | bytes[1]<<16 | bytes[2]<<8 | bytes[3]) != expected_value) return (-1);
    if((words[0]<<16 | words[1]) != expected_value) return (-1);
    if(dword != expected_value) return(-1);
    return(0);

}
```

5. Theory of Operation

The first line reads the ModelID, ModuleType, RevID, and ModuleID starting at 0xC0. These 4 bytes are then combined in big-endian format to build the expected_value.

The `rd_write_verification()` function reads the same 4 bytes using all possible I3C access mechanisms and compares the result to the expected result.

The rest of the code executes the other write mechnisms and compares the read results to the expected.

6. Usage

Execute this code once per design to verify the correct operation of the I2C functions created by the user. After the verification passes, the code can be removed.