ANALOG DEVICES

AN-1367
应用笔记

One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106, U.S.A. • Tel: 781.329.4700 • Fax: 781.461.3113 • www.analog.com

ADE7953和ADuCM360之间的I²C接口

作者：Fermi Lim，Daniel Kim，Hariharan Mani

简介

本应用笔记说明了如何使用C语言来实现在ADE7953(从机)和ADuCM360(主机)之间实现I²C接口通信。ADE7953是一个单相电能计量IC，而ADuCM360是一个基于ARM® Cortex®-M3的微控制器。ADE7953包含8位、16位、24位以及32位长的寄存器。编写源代码时，必须确保依据寄存器的地址来识别读/写操作的数据大小。ADE7953数据手册中的寄存器清单包含全部相关信息。

此应用笔记介绍了ARM Cortex-M3内核和ADuCM360微控制器(MCU)，接着介绍了需要在ADuCM360中完成的初始化步骤，最后介绍了在ADE7953和ADuCM360之间如何实现I²C接口。在此应用笔记中所描述的示例代码也可以用在其他基于ARM Cortex-M内核的ADI处理器中。

用于建立I²C接口的完整源代码以可下载文件的形式(AN_1367_I2C_interface.zip)提供，网址为www.analog.com/ADE7953或www.analogcom/ADuCM360。

本应用笔记说明了如何在Visual C++ 2012集成开发环境(IDE)中实现源代码，并为ADE7953和ADuCM360之间的I²C通讯端口提供了有用的见解。

验证该源代码需用到下列设备和软件：

- ADE7953和ADuCM360评估模板 (EVM)
- 软件的开发环境：Keil MDK-ARM版本 4.72
- 终端模拟器：Tera Term版本4.79
- IDE：Visual C++ 2012
- 笔记本电脑 (PC)

整个实验环境设置如图1所示。两块评估模板以线相连，如图2所示。本设置中未考虑使用隔离接口，因此两块模板都以普通电源供电并共用一根地线。如果要隔离I²C接口，则必须考虑隔离器的传播延迟问题。

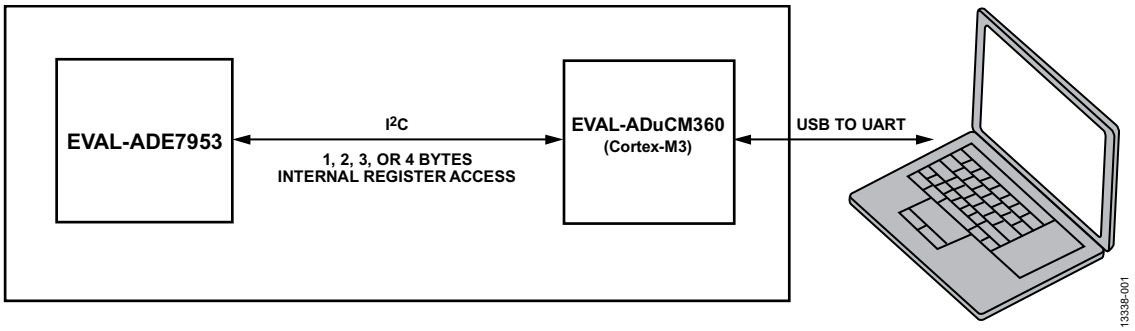


图1. 实验室设置

目录

修订历史

2015年9月—修订版0： 初始版

设置说明

终端模拟器软件Tera Term(版本4.79)被用于通过ADuCM360读取ADE7953寄存器的信息或向其写信息。这一通用的异步接收器/发射器(UART)通信接口被用在ADuCM360和电脑之间，用于发送和接收读/写的命令和数据。Tera Term(版本4.79)软件(包括为基于接口的UART通信完成源代码的示例)可从各种资源免费获取，其中包括OSDN公司。

图3显示的是终端窗口，通过此窗口，示例代码的获取和设置命令被用于访问ADE7953寄存器。这两个命令的语法是

- 设置REG_ADDR值
  设置命令采用特定值对位于REG_ADDR的寄存器进行写入。
- 获取START_REG_ADDR REGISTER_NUMBER
  获取命令读取从START_REG_ADDR地址开始、连续REGISTER_NUMBER数量的寄存器内容。待读取寄存器位置的数量由REGISTER_NUMBER决定。

该源代码的独特性在于，即使不知道每个寄存器的数据大小，它也允许电脑与ADE7953通信，这是因为寄存器的查找表已被使用。

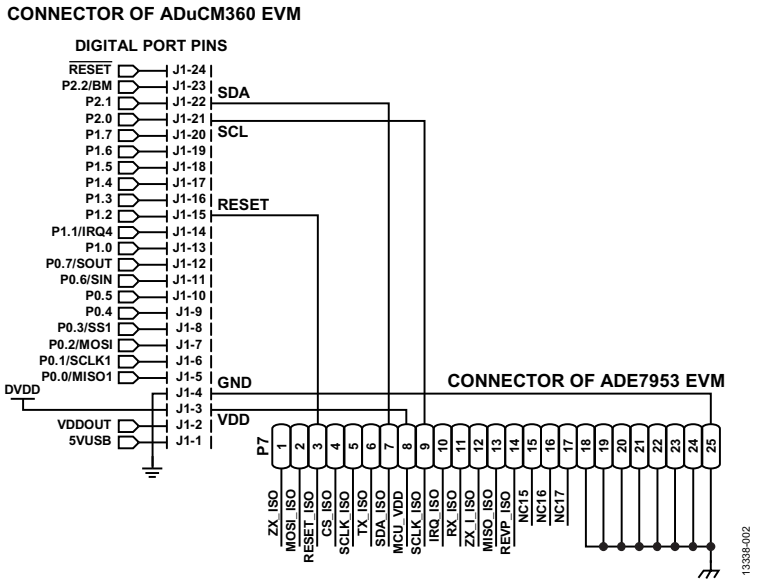


图2. ADuCM360 EVM与ADE7953 EVM之间的连接

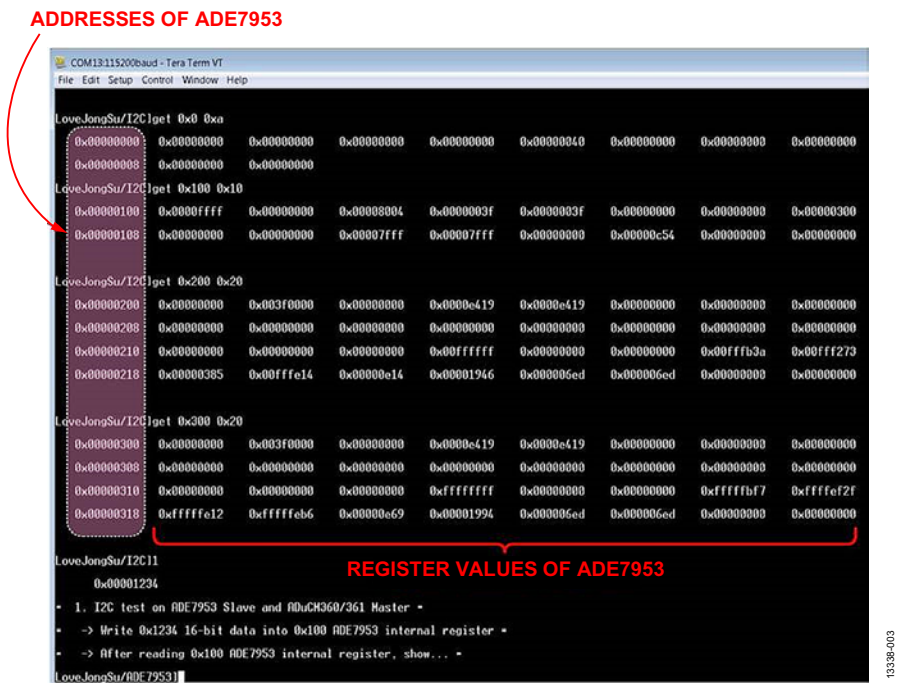


图3. 显示使用获取命令的终端窗口

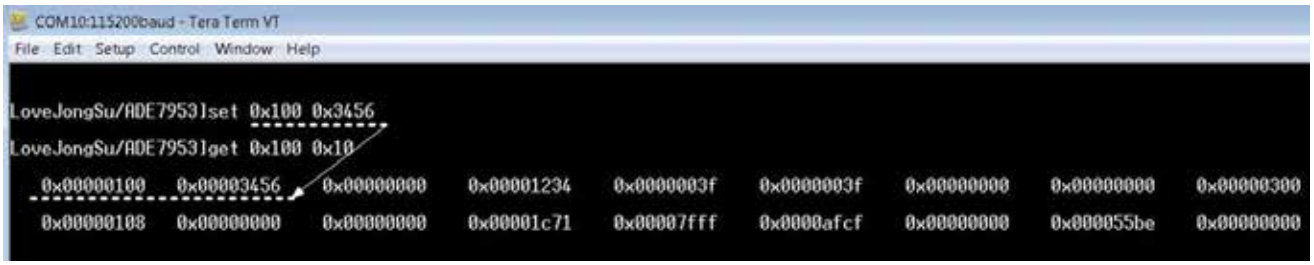


图4. 显示使用获取和设置命令的终端窗口

ARM CORTEX-M3内核与ADUCM360 MCU

ARM Cortex-M3内核是基于ARMv7-M架构。Cortex-M系列主要用于嵌入市场、所需的内核时钟频率低于300 MHz的应用中。大部分嵌入式应用(例如交流电动机控制和其他简单的家用电器)并不需要如内存管理单元(MMU)和NEON™一类的复杂功能，使Cortex-M系列成为不错的选择。Cortex-M内核的架构要比其他Cortex系列简单。

Cortex-M内核系列包含以下独特功能：

- 固定的存储图谱
- 通用嵌套矢量中断控制器(NVIC)
- 仅支持 Thumb®-2指令组

为一个Cortex-M系列MCU写的代码易于修改并可用于另一个Cortex-M系列MCU，即使它来自另一家供应商。而且，ARM免费为广泛的应用提供有用的Cortex微控制器软件接口标准(CMSIS)软件。

图5对Cortex-M系列内核的固定存储图谱与ADuCM360的固定存储图谱进行了比较。

如图5所示，ADuCM360包含适用于Cortex-M系列、基于固定存储图谱的8 kB SRAM和128 kB闪存盘/EE存储器。所有ADuCM360外围设备均被映射至0x40000000至0x4004FFFF地址范围。0xE000E000 - 0xE000EE000的地址范围适用于Cortex-M内存映射寄存器(MMR)；0x40000000 - 0x4004FFFF地址范围适用于ADuCM360 MMR。

注意以下有关NVIC的信息：

- Cortex-M内核系列只有一种核心中断，而较老的ARM内核有快速中断和正常中断两种中断。
- 两组寄存器可用：中断使能寄存器(ISER)和中断除能寄存器(ICER)。

一般来说，如UART和I²C一类的外围设备自身都具有寄存器，可以禁用/启用中断。根据ISER/ICER值，在每台外围设备传输的中断中，只有一种能到达MCU核心。ISER处于0xE000E100 - 0xE000E13C地址范围，ICER处于0xE000E180 - 0xE000E1BC地址范围。使用Cortex-M系列MCU时，请按需要更改ISER和ICER。在所有可用的IDE工具中，Keil MDK-ARM和IAR Embedded Workbench®最受欢迎。当可执行文件在指定的代码尺寸限制内时，这两种工具都可使用。Keil MDK-ARM IDE用于设置此应用笔记的实验环境。要在IAR Embedded Workbench IDE中使用来自可下载文件AN_1367_I2C_interface.zip中的代码，则必须更改**ADuCM360StartUp.s**文件。将IDE从Keil MDK-ARM转入IAR Embedded Workbench或从IAR Embedded Workbench转入Keil MDK-ARM时，只需更改汇编指令，无需更改指令组，这是因为汇编指令由汇编器决定，而指令组由MCU内核决定。

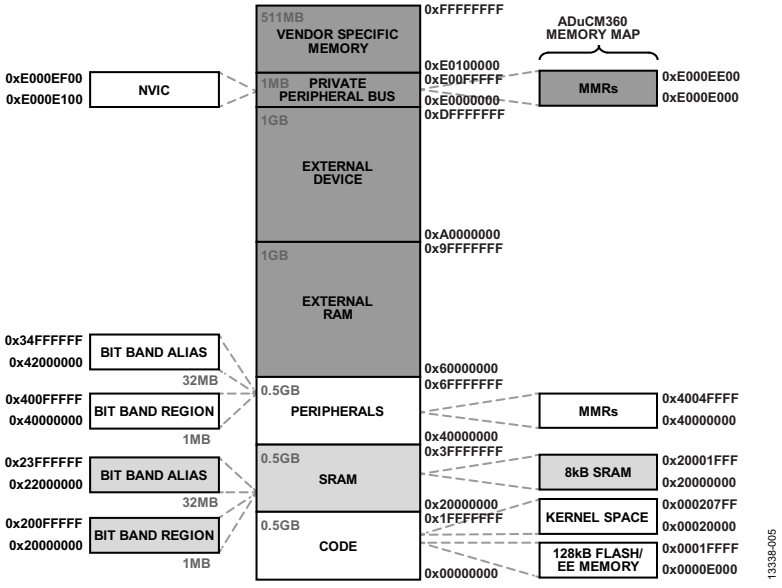


图5. Cortex-M和ADuCM360的存储图谱比较

图6和图7分别显示了Keil MDK-ARM和IAR Embedded Workbench IDE各自的工具链。在图6中，以灰色阴影显示armcc、armasm和armlink，表示它们分别构成C/C++编译器、汇编器和链接器。在图7中，同样以灰色阴影显示iccarm、iasmarm和ilinkarm，表示它们分别是编译器、汇编器和链接器。

图8显示了ADuCM360的架构。在ADuCM360的外围设备中，只有I²C、UART和GPIO用于与ADE7953和电脑进行通信。

首先，将UART配置为与电脑通信，然后将I²C配置为接口与ADE7953通信。

尽管本应用笔记的重点讲述I²C接口，但可下载文件AN_1367_I2C_interface.zip提供的示例代码中也包含可用于UART接口和GPIO的模块程序。

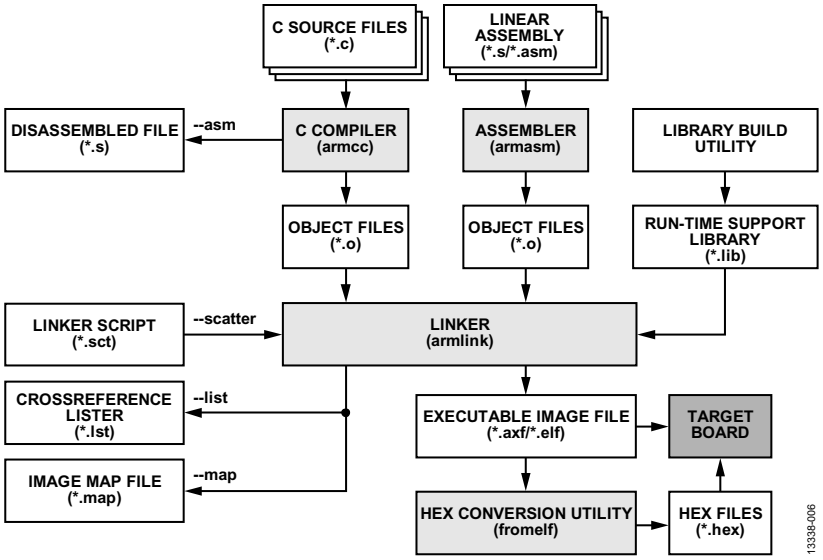


图6. Keil MDK-ARM工具链

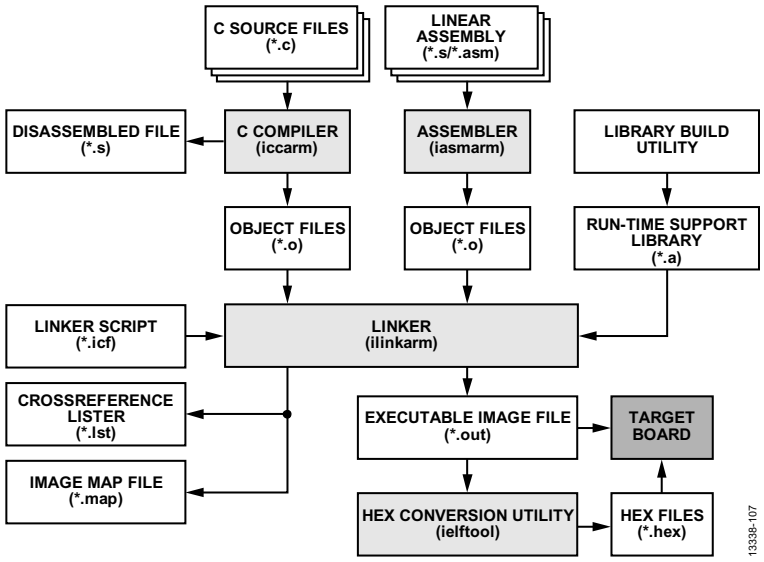


图7. IAR Embedded Workbench工具链

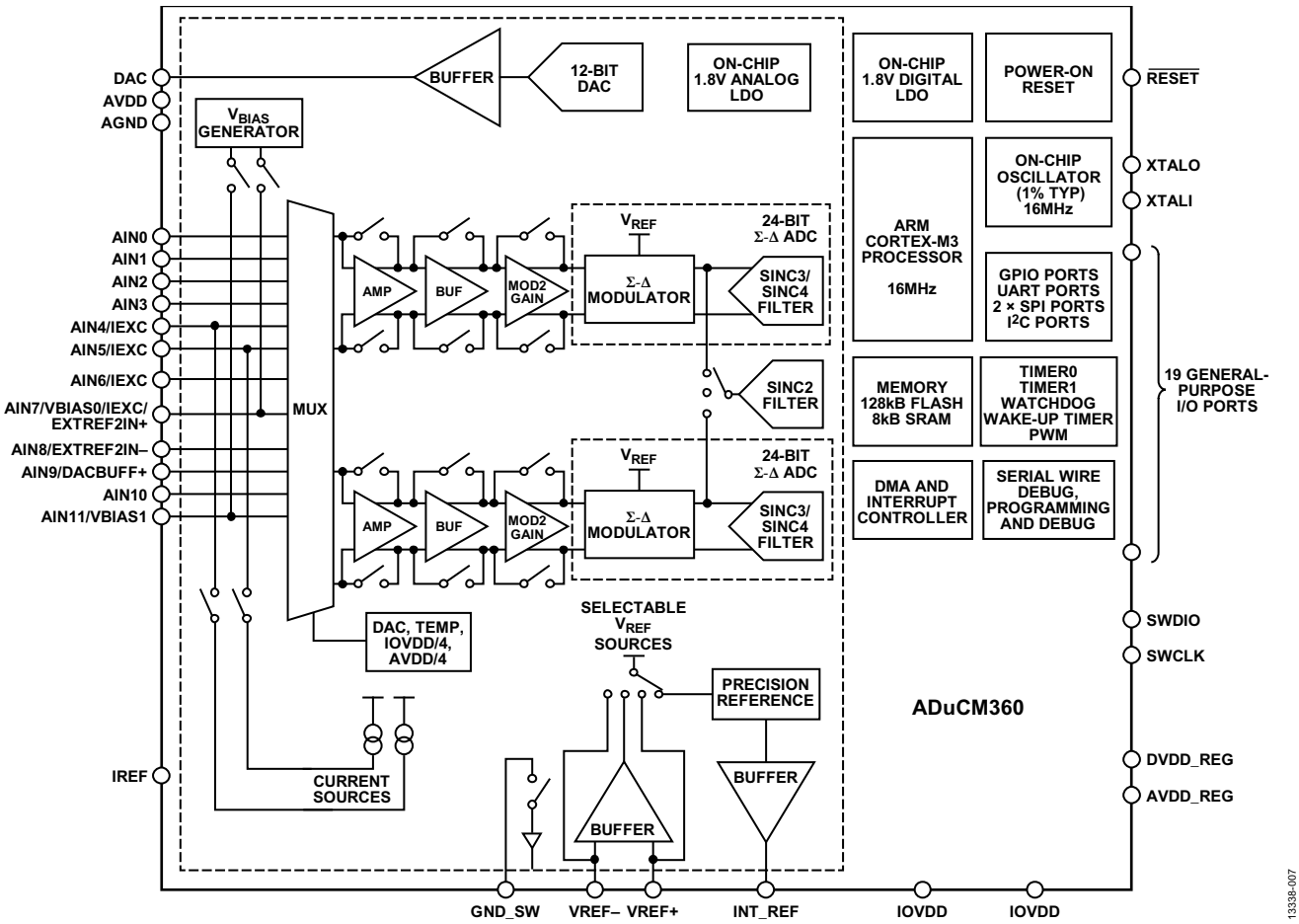


图8. ADuCM360框图

初始化适用于UART和I²C的ADUCM360

开发任意嵌入式C代码时，首先要禁用看门狗定时器看门狗定时器，然后按照需要为内核和外围设备配置时钟。

看门狗定时器在执行任何复位操作之后都会自动启用。要禁用ADuCM360的看门狗定时器，在T3CON[5]中写入0'b0，这是看门狗定时器的使能位。

```
// Step 1. Disable the watchdog timer.
  *pT3CON=0x0;
```

然后，配置UART、I²C和Cortex-M3内核的时钟。图9和图10显示了如何配置UART和I²C的时钟。将内核时钟配置为16 MHz，然后相应设置UART和I²C接口的控制寄存器。通过CLKCON1选择I²C系统分频比时，确保内核的时钟频率低于或等于I²C系统时钟的分频比，也就是说，HCLK≤I2CCLK。此情况适用于时钟可由CLKCON1划分的所有外围设备来说。如果外围时钟较慢，则连接至外围设备的时钟会被封闭，造成外围设备不工作。欲了解更多信息，请参考ADuCM360/ADuCM361硬件用户指南。

如图9所示，电脑的UART接口的设置传输速率为115,200 bps。因此，请相应配置Tera Term工具。参考名为"MyTera_Setup2014.INI"的Tera Term项目文件，该文件可从可下载文件AN_1367_I2C_interface.zip中获取。

要使用Tera Term项目文件，请如图11所示，从**设置**菜单中选择**还原设置**。然后，从电脑的存储位置中选择**MyTera_Setup2014.INI**项目文件。

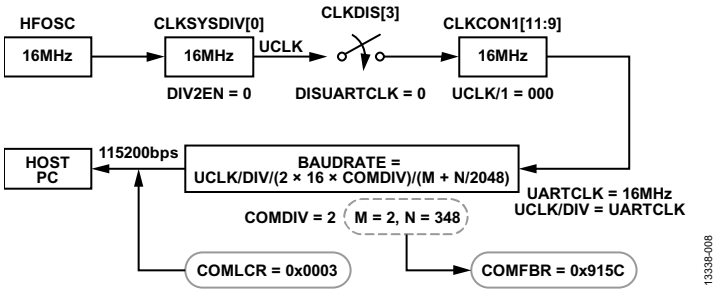


图9. UART的时钟设置

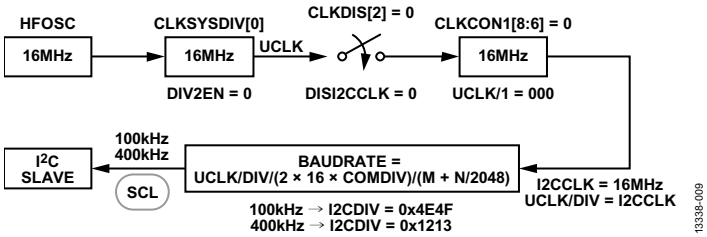


图10. I²C的时钟设置

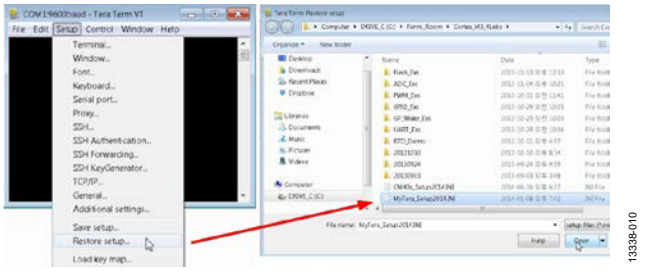


图11.Tera Term设置

从**ADuCM360_I2C_AppNote.c**文件(从可下载文件AN_1367_I2C_interface.zip)中提取的以下C代码说明了如何配置UART接口。代码中的所有变量表示的是与时钟、GPIO和UART相关的MMR。

```
// 第2步：配置时钟。
// UART速率为115,200 bps：
UCLK/DIV=16MHz。
*pCLKDIS=0x03F7；
*pCLKSYSDIV=0x00；
*pCLKCON0=0x0000；
*pCLKCON1=0x0000；
// 第3步：设置选定UART的行为。
*pCOMCON=0x00;// UART外围设备已启用。
// P0.2是UART的发射率；P0.1是UART的接受率。
*pGP0CON=0x003C；
*pCOMDIV=0x0002; // COMDIV = 2;
*pCOMFBR=0x915C; // Set to 115,200 bps
// WordLength = 8 bits, stop bit = 1 bit, no parity check.
*pCOMLCR=0x0003;
// COMTX and COMRX are enabled; COMIEN[1] = COMIEN
[0] = 1
*pCOMIEN=0x0003;
```

UART的中断使能寄存器COMIEN仅适用于UART外围设备的中断(欲了解更多信息，请参考ADuCM360/ADuCM361硬件用户指南)。对于中断将传输至Cortex-M3内核的UART发射器(Tx)/接收器(Rx)，如下所示设置相应的ISER：

```
// ISER = 0xE000E100; enabling UART interrupt.
write_reg(0xE000E100,0x00020000);
```

区分中断和例外情况非常重要。例外情况可归类为同步或异步。任何系统故障一般都可称为同步例外，而中断则是异步例外。在ADuCM360的中断矢量表中，UART排在第17位。因此，要启用UART中断，请将ISER的0xE000E100地址位置设置为0x00020000。同样，要启用I²C中断，请将ISER如下设置：

```
OldRegVal=read_reg(0xE000E100);
RegVal = OldRegVal | 0x00200000; // for I2C
write_reg(0xE000E100,RegVal); //ISER = 0xE000E100;
```

然后，初始化ADuCM360 I²C外围设备。ADE7953一直是从机，因此必须将ADuCM360配置为主机。

作为主装置，ADuCM360中可用的I²C模式包括

- 标准模式：100 kHz
  首先设置CLKSYSDIV[0] = 0，CLKCON1[8:6] = 000，然后设置I2CDIV[15:8] = 0x4E，I2CDIV[7:0] = 0x4F
- 快速模式：400 kHz
  首先设置CLKSYSDIV[0] = 0，CLKCON1[8:6] = 000，然后设置I2CDIV[15:8] = 0x12，I2CDIV[7:0] = 0x13

图12显示了SCL在标准模式下的生成方式。

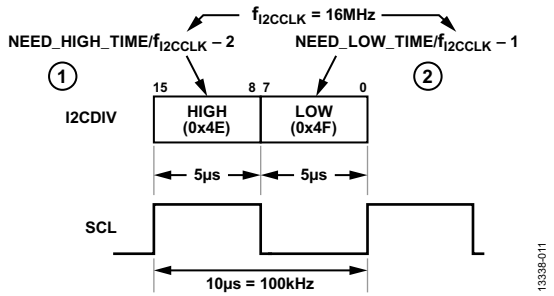


图12. 标准100 kHz SCL的生成方式

如图10和12所示，f_{I2CCLK}为16 MHz，如果CLKSYSDIV[0] = 0和CLKCON1[8:6] = 0被初始化，SCL的比特率f_{SCL}则使用以下公式予以计算：

$$f_{SCL} = \frac{f_{I2CCLK}}{(Low + High + 3)}[\text{Hz}]$$

其中：
*Low*指的是SCL的低速时段，这是由I2CDIV[7:0]设定的。
*High*指的是SCL的高速时段，这是由I2CDIV[15:8]设定的。

在标准模式中，I2CDIV[7:0]被设置为0x4F (0d'78)，I2CDIV[15:8]被设置为0x4E (0d'79)。使用这些数字就能获得100 kHz的SCL时钟频率。

为了在Visual C++中使用此代码，则增加以下说明：

```
#ifdef FermiEmulation_Mode
// to initialize all of the ADuCM360 register set.
   ADuCM360_RegsInit();
#endif
```

欲了解更多有关此示例代码的信息，请参考I²C接口：ADuCM360和ADE7953部分。此示例代码是基于超环架构的。因此，这种代码类型无法处理多任务。通过Tera Term工具使用菜单使得任何图形用户界面(GUI)元件可由Visual C++或Visual Basic轻松实现。大部分工厂自动化系统使用这一方法。

I²C接口：ADUCM360和ADE7953

通过SCL和SDA两点实现I²C接口。(注意，在此部分中，多功能引脚(例如P2.0/SCL/UARTCLK)都是因与某一功能(例如SCL)相关才被引入的。

- SCL：串行时钟引脚。只有主机才能生成I²C时钟；SCL可用在标准模式或快速模式中。
- SDA：串行数据引脚。

SCL和SDA均为双向，必须用一个上拉电阻器连接到正极电源。尽管在本例中SCL是由ADuCM360生成，但理论上，它可以是双向的，因为ADuCM360的I²C接口可以设置为从属接口。ADE7953的I²C接口只能被用作从属接口。

注意，ADE7953在SCL和SDA边缘之间最低要求有0.1 μs的延时；查看数据手册中的$t_{HD;DAT}$规格。ADE7953数据手册还列出了其他必须保持的正时规格。

图14所示为一个典型I²C传输序列。

ADuCM360有两个I²C接口：一个是通过P0.1/SCLK1/SCL/SIN与P0.2/MOSI1/SDA/SOUT引脚，另一个是通过 P2.0/SCL/UARTCLK与P2.1/SDA/UARTDCD引脚。在此例中，使用的是P2.0/SCL/UARTCLK和P2.1/SDA/UARTDCD引脚。

```
// P2.0/SCL/UARTCLK 被用于 SCL，P2.1/SDA/UARTDCD 被用于 SDA。
*pGP2CON |= 0x05;
// Master Enable, Tx/Rx Request Interrupt Enable.
*pI2CMCON = 0x131;
```

每个从设备都有一个I²C操作地址，主设备可利用此地址识别从设备，该地址由供应商设定。例如，ADE7953地址为0x38。

I²C寻址类型可为7位或10位。根据ADE7953的寻址类型，本例使用7位寻址。

如图13所示，ADuCM360的I²C接口具有2字节发射和接收FIFO方案。

核心时钟和外围时钟通常是不同的。如果数据从一个时钟域传输至另一个时钟域，则可能因通信过程中的错误出现压稳定性。为了避免这一情况，通常使用FIFO或双端口RAM。使用组合逻辑无法解决亚稳定性问题。

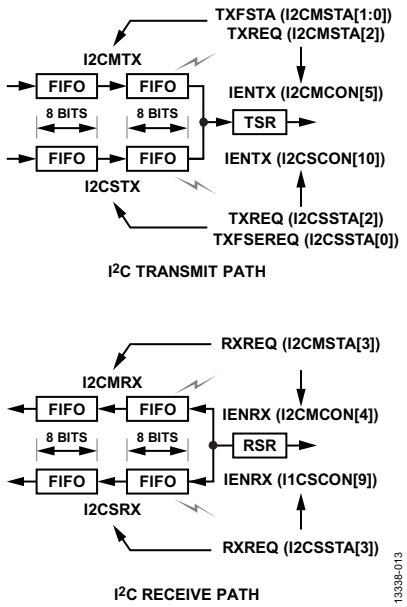


图13. I²C Tx/Rx的数据路径

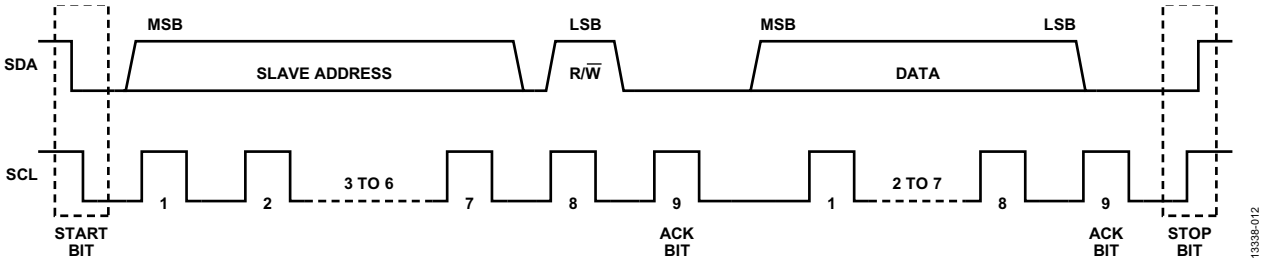


图14. 典型I²C传输序列

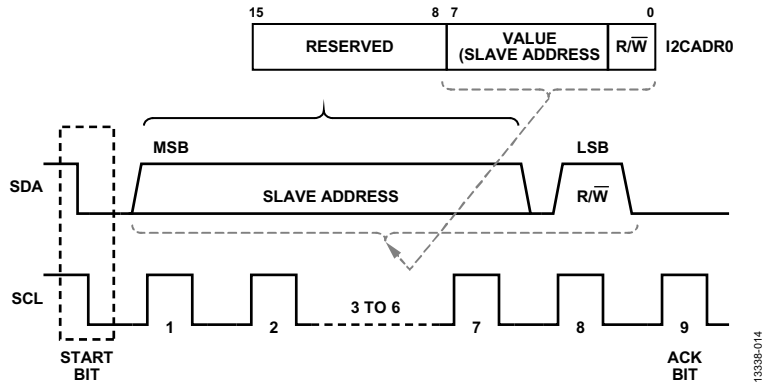


图15. 7位寻址模式

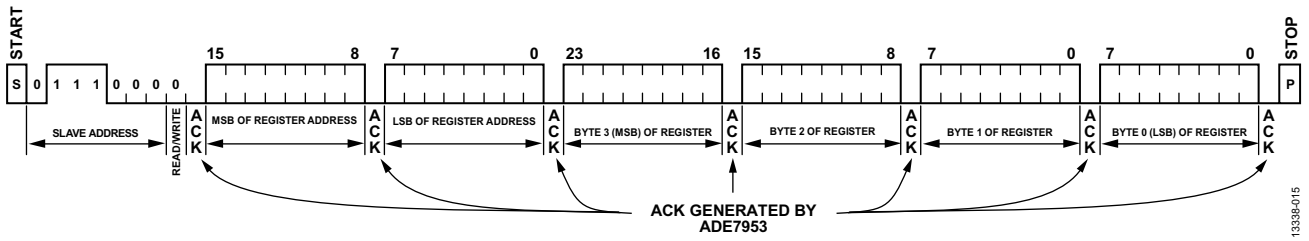


图16. ADE7953的I²C写入序列

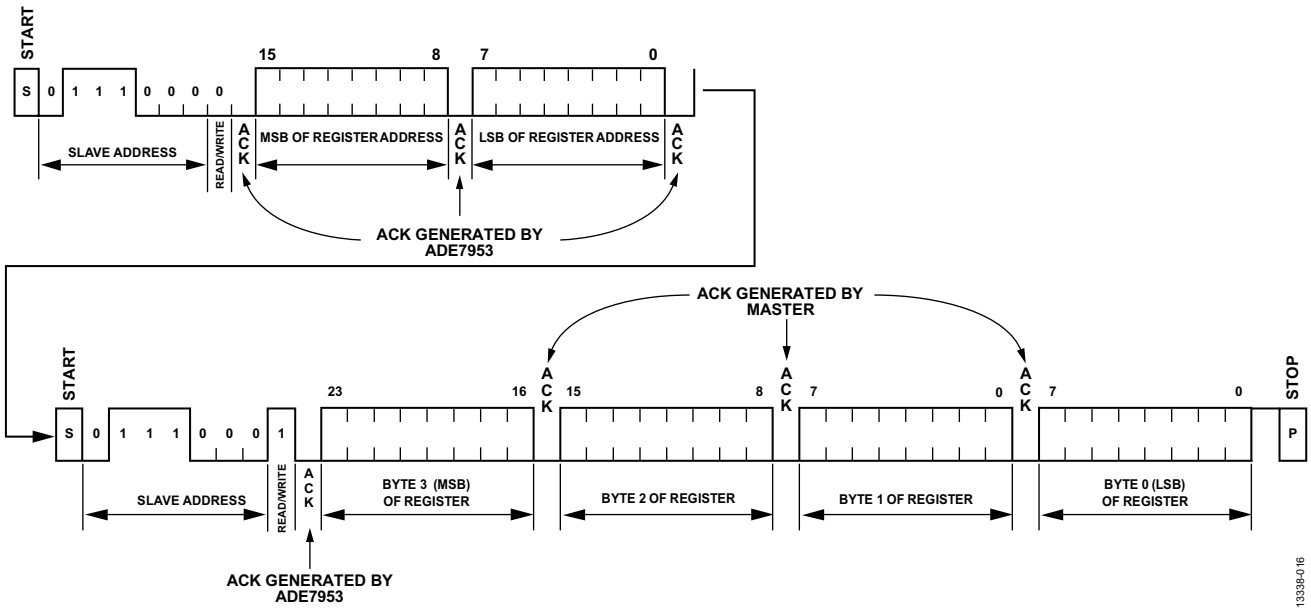


图17. ADE7953的I²C写入序列

图15显示了主设备如何将7位地址(Bits[7:1])和方向(读取和写入)位(Bit 0)发送至从设备。

I2CADR0寄存器保存从设备地址和方向位。

ADE7953的I²C接口支持的最高串行时钟频率为400 kHz。

图16和图17分别显示了ADE7953的I²C写入和读取操作。

ADE7953 I²C地址为0b'0111000x，x是个方向位。0表示写操作，1表示读操作。

当主设备发出一个7位设备地址且方向字节设为0(即0x70)时，启动ADE7953的写操作。之后是一个16位内部寄存器地址。每接收到一个字节，ADE7953就会向主机发出一个应答。主机随后发送寄存器数据，第一个是MSB。数据的长度可以是8位、16位、24位或32位，具体取决于寄存器本身。最后一个字节传输完成后，主机发出停止条件，总线返回空闲状态。

编写源代码时，重要的是要了解如何在目标设备上执行I²C。

下列代码片段代表的是ADE7953 I²C的写操作：

```
void I2C_Write_ADE7953(void)
{
    int i=0;
    if (((I2CMasterTxDat[0] >4) && (I2CMasterTxDat[0] <=
6))||((I2CMasterTxDat[0] >9))) {
        printf("The specified ADE7953 register address is out of
range\n");
    } else {
        ReadFlag=0;
        uiMasterTxIndex = 0;
        // Master case : send 1st data.
        *pI2CMTX =
I2CMasterTxDat[uiMasterTxIndex++];
        I2cMWrCfg(0x70);
        while (!ucComplete){}
        ucComplete = 0;
    }
}
```

前面代码中的假设条件是基于ADE7953地址范围。寄存器地址的MSB决定地址是否超出范围。参考ADE7953数据手册和ADE7953评估板用户指南了解更多信息。

I2cMWrCfg(0x70)程序如图15和图16所示，向ADE7953发送0x01110000 (0x70)。

每收到一个字节，ADE7953都会向主机应答，ADuCM360都会生成一个I²C接口，以调用在**ADuCM360_Test_Lib.h**文件中找到的I2C0_Master_Int_Handler函数，这会改变前面代码的ucComplete值。如果ucComplete值改变，I2C_Write_ADE7953()会退出当型循环并继续向ADE7953发送一个字节。

往任何ADE7953寄存器中写入时，必须确保数据长度匹配寄存器的描述。从ADE7953 IC中回读信息时，这一观念也同样适用。I2C0_Master_Int_Handler函数在写操作和读操作中都会考虑寄存器数据的长度。

在代码中存在错误，尤其是针对初始项目或使用新IC时，使用Visual C++调试程序帮助调试。使用Visual C++ 调试程序也能帮助用户理解代码是如何工作的。

一般来说，嵌入式代码涉及物理地址，但Visual C++只能解读由操作系统(OS)的MMU分配的虚拟地址。程序运行时，虚拟地址可能改变。Cortex-M系列微控制器不包含一个MMU；因此，基于Cortex-M内核的MCU不能被轻易迁移到(例如)Windows® CE或Linux® OS中。要学习如何利用Visual C++与嵌入式C代码，请参考利用Visual C++仿真嵌入式C代码章节。”

利用Visual C++仿真嵌入式C代码

在Visual C++中完成整个C代码之后，要为ADuCM360生成一张可执行的图形，请在Keil MDK-ARM或IAR Embedded Workbench IDE中执行重建。因为示例代码中的以下C/C++预编译器指令，这种重建是可行的：

```
#ifdef FermiEmulation_Mode
// to initialize all of the ADuCM360 register set.
    ADuCM360_RegsInit();
#endif
```

在使用Visual C++ 2012调试程序工具时，以下这种逐步进行的步骤有助于理解调试过程。注意，以下方法可用于任何Visual Studio版本。

1. 如图18所示，创建一个新项目。

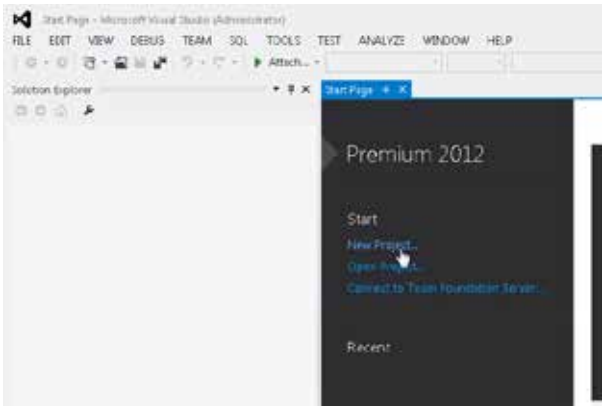


图18. 创建新项目

2. 如图19所示，选择**Empty Project**。

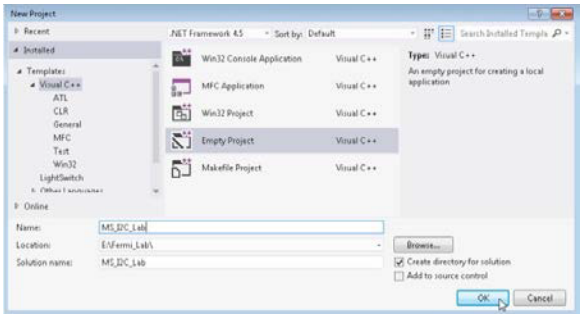
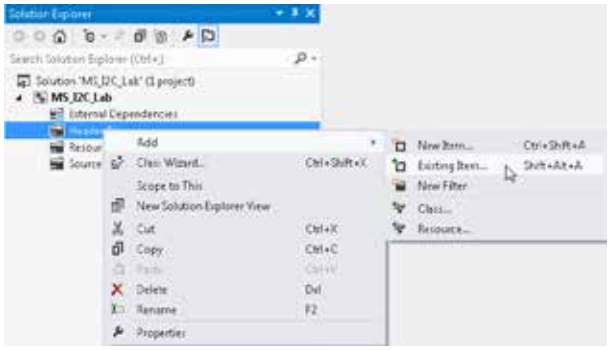


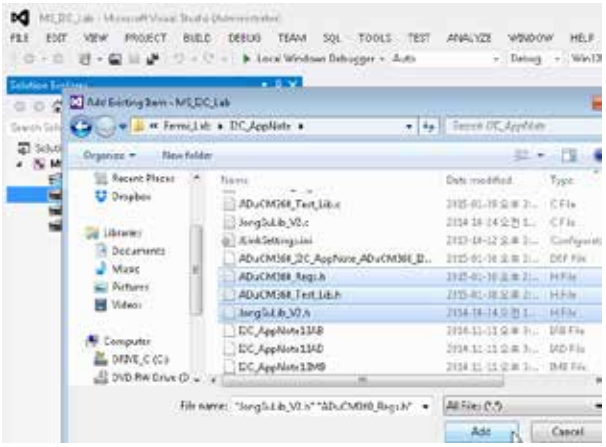
图19. 选择Empty Project

3. 如图20和图21所示，将所有样本文件和现有项目添加到该项目中。



13338-019

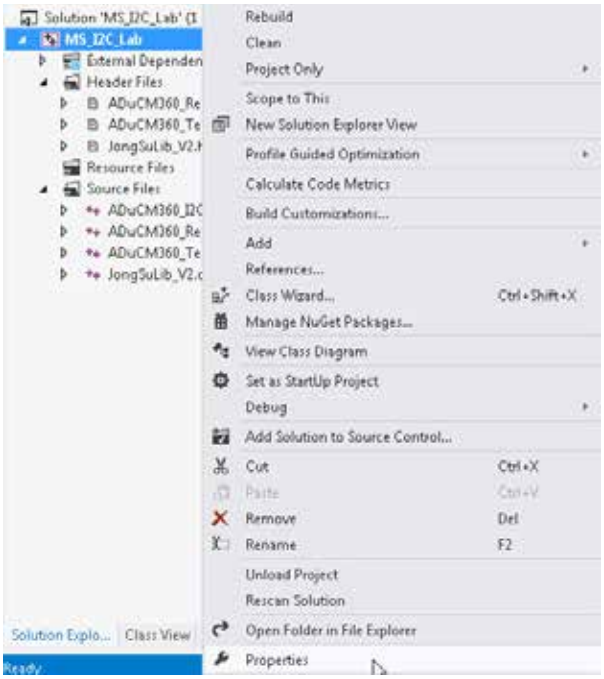
图20. 添加样本文件



13338-020

图21. 添加现有项目

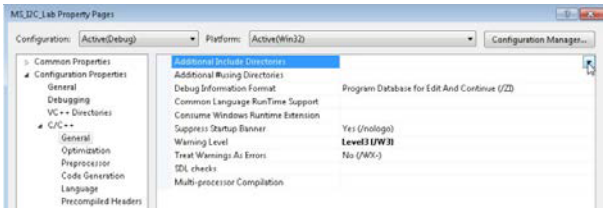
4. 如图22至图25所示，更改某些属性。



13338-021

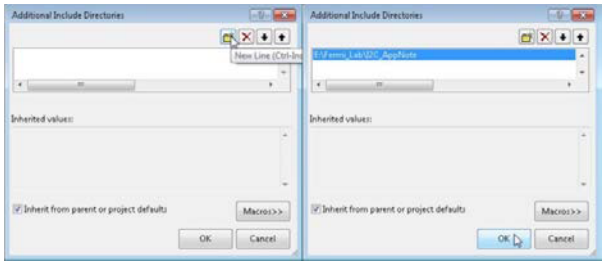
*图22. 选择**Properties***

5. 在C/C++下选择包含示例代码的文件夹。点击**Additional Include Directories**。参考图23和图24。



13338-022

图23. 搜索示例代码



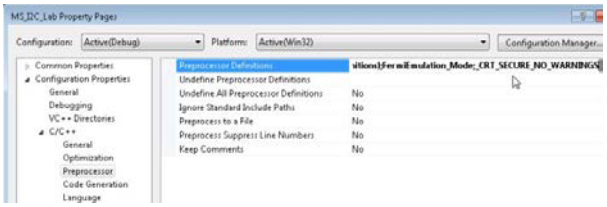
13338-023

图24. 选择包括Visual Studio示例代码的文件夹

6. 如图25所示，仅对Visual C++编译器定义**FermiEmulation_Mode**。

   这一步骤基于哪个Visual C++代码可编程并与嵌入式C代码配合使用来预定义**FermiEmulation_Mode**。

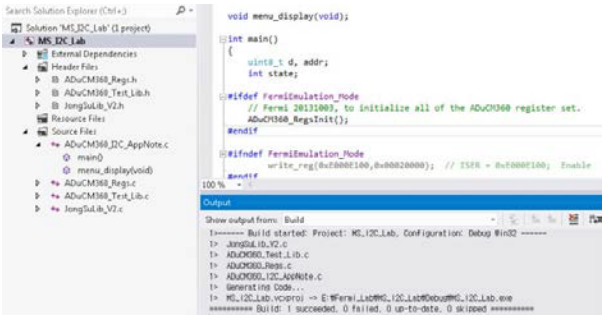
   使用其他Visual Studio版本时，需要搜索图22至图25中所示的相应选项，但步骤不变。仅为.NET添加**_CRT_SECURE_NO_ WARNINGS**(参考图25)。



13338-024

图25. 定义代码

7. 如图26所示，编译和链接Visual C++下的所有示例代码。



13338-025

图26. 编译和链接所有示例代码

如图26所示，代码中没有错误和警告。

图27显示了如何利用Visual C++仿真嵌入式代码。

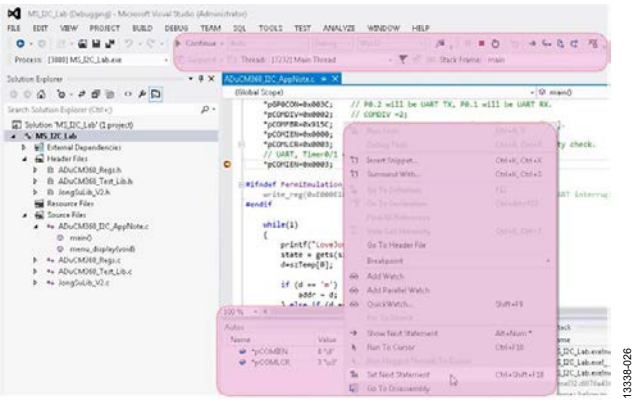


图27.仿真嵌入式代码

要了解Visual C++如何工作，请仔细分析**ADuCM360_Regs.c**和**ADuCM360_Regs.h**。这些文件定义与ADuCM360有关的真实地址，并启用MMU以依照真实地址分配虚拟地址，以便Visual C++能够除去这些文件的错误。注意，这些文件也可以通过Keil MDK-ARM或IAR Embedded Workbench进行调试。如果使用Keil MDK-ARM或IAR Embedded Workbench进行调试，则无需使用FermiEmulation_Mode。

这种调试方法非常简单，可用在所有类型的处理器中。使用Visual C++调试程序的优点在于易于分析I2C0_Master_Int_Handler函数和设置/获取命令。

I2C0_Master_Int_Handler函数的以下命令是用于同步：

```
#ifndef FermiEmulation_Mode
        __asm{ DSB}
    __asm{ DSB}
    __asm{ nop}
    __asm{ nop}
    __asm{ nop}
    __asm{ nop}
#endif
```

下载示例代码中的printf函数旨在使用Visual C++时缩小代码尺寸；因此，如果使用Keil MDK-ARM或IAR Embedded Workbench运行库，则代码尺寸可能更大。在**JongSuLib_V2.c**库文件中可以找到压缩printf函数，它还包含许多有用的其他函数，例如寄存器读/写函数。名称为**ADuCM360_Test_Lib.c**的有用库文件包含支持ADuCM360外围设备(如I^2C、SPI、UART和ADCs/DAC)的函数。这两种通用库在需要时也可导入其他项目。

结论

本应用笔记说明了如何实现I²C C语言代码，来通过ADuCM360监控和修改改ADE7953的寄存器。建议使用Visual C++来开发代码，因为使用此工具很容易进行编码和调试。然后使用Keil MDK-ARM或IAR Embedded Workbench来编译和链接C代码，无需额外更改，即可生成可执行文件。

I²C指最初由Philips Semiconductors(现为NXP Semiconductors)开发的一种通信协议。