BR0101固件开发——模数转换器AD9715

（第五周）

第四周我们在等待新一批AD9715到货，同时也做了一些其他AD/DA电路设计等相关工作。

# 1 第四次上板调试的记录

在新的一批AD9715到货之后，我们进行了第四次上板调试，得到了如下的一些结果：

* 新的AD9715在高采样率数据输入的情况下FSADJx引脚电平一般不发生变化，保持在1.0V（如下图），只有DAC1的FSADJQ引脚电平会下降到0.2V左右，这与旧的AD9715在这种情况下FSADJx脚全部降到0V的现象不同，怀疑可能是旧的芯片存在故障；

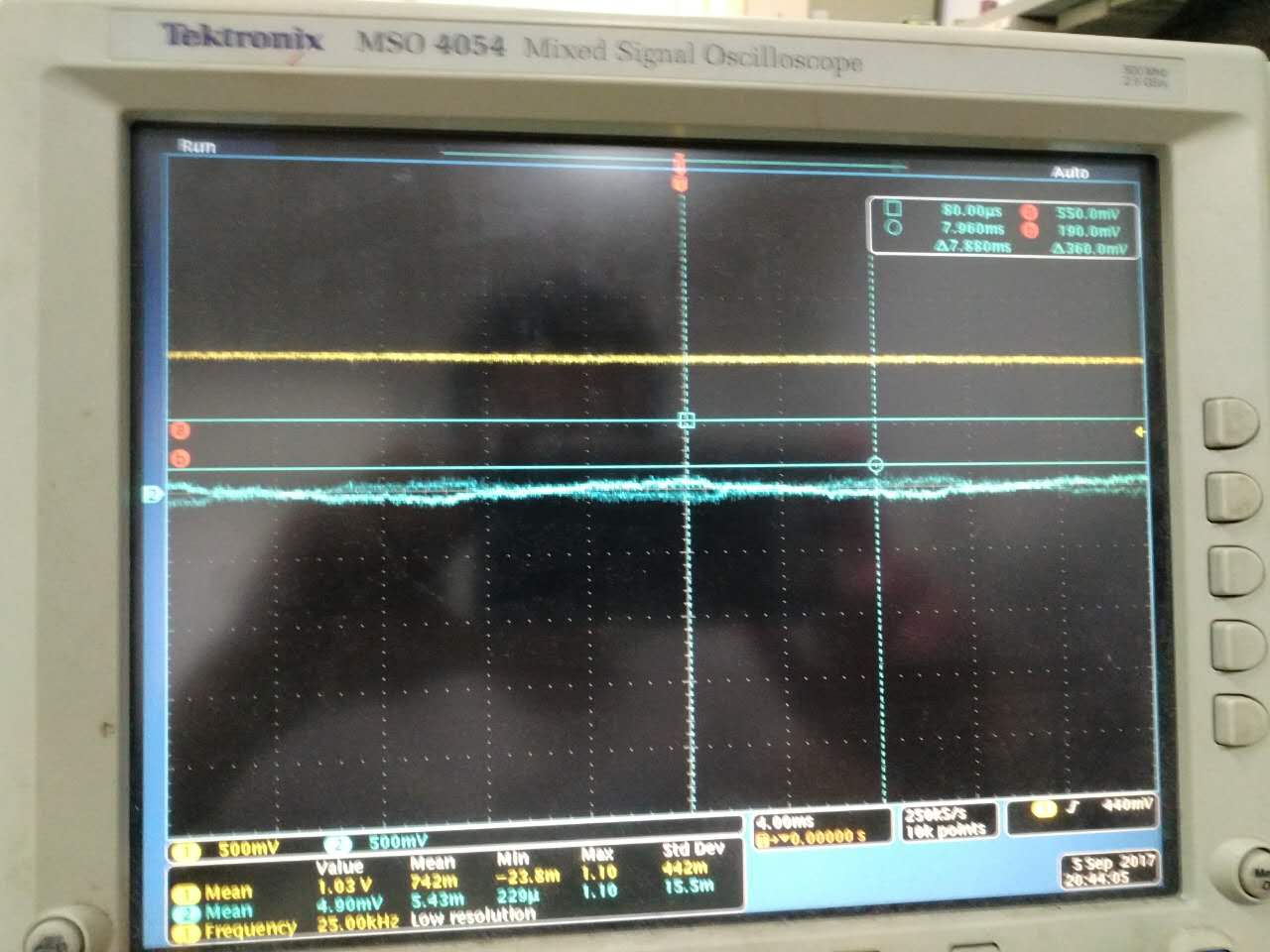


图 1 新的AD9715 FSADJx引脚电平测试波形图

* 新的芯片在高采样率情况下仍然无法输出正常的波形，采样率超过2KSPS，输出电压的范围就会大大减小；
* 新的芯片中辅助DAC部分仍然无法工作；
* 新的AD9715版本号寄存器中读出来的值仍然是4；

这些结果显示AD9715在高采样率情况下输出信号范围大幅度减小的问题仍没有得到解决。

# 2 信号跳变测试与第五次上板调试的成果

为了解决AD9715开发过程中遇到的问题，我们查阅了很多资料，也尝试在ADI官网的论坛Engineer Zone上面提问，还咨询了一些有经验的人。有人提出这个问题可能是因为电流型DAC的过饱和特性引起的，因此建议我们测试信号跳变的幅度和频率不同的情况下输出电压范围衰减的程度和时间。于是我们进行了程序设计和第五次上板调试。

## 2.1 信号跳变测试程序设计

我们计划设计一个测试程序来简单地控制信号跳变的幅度和频率，这就需要用到UART双向通信。为了简单起见，我们采用UART轮询的方法，实现了一个简单的UART交互程序。

这个程序的原理是通过Microblaze循环向寄存器里写数值，控制DAC输出电压，从而形成在0和某个电压之间跳变的方波。跳变的幅度由变化的比特决定，跳变的频率由循环延时的次数决定。每次循环结束，尝试读取一次UART。因为UART接收函数XUartLite\_Recv()是非阻塞的，不会对波形输出产生影响。如果没有接收到命令，则继续循环；如果接收到0~9等字符，则将每次跳变的比特改为相应比特位，并继续循环；如果接收到“H”、“L”等频率控制命令，则将循环延时的次数改为相应的值，并继续循环；如果接收到“S”SPI写指令，则延时1000000次循环，再读取两个字节的数据，将这两个字节的数据写入固件寄存器，通过SPI发送出去。这样就实现了简单的交互控制功能。最后的交互界面如下图所示：



图 2 信号跳变测试程序UART交互界面

## 2.2 第五次上板测试的成果

### 2.2.1 信号跳变测试的结果

首先，我们利用上面的程序进行了信号跳变的测试，观察了AD9715在输入数字信号第9、第8、和第七比特分别跳变，采样率分别为450SPS、900SPS、1800SPS、3600SPS、450KSPS的情况下，输出信号的幅度范围和变化时间。得到部分结果如下：

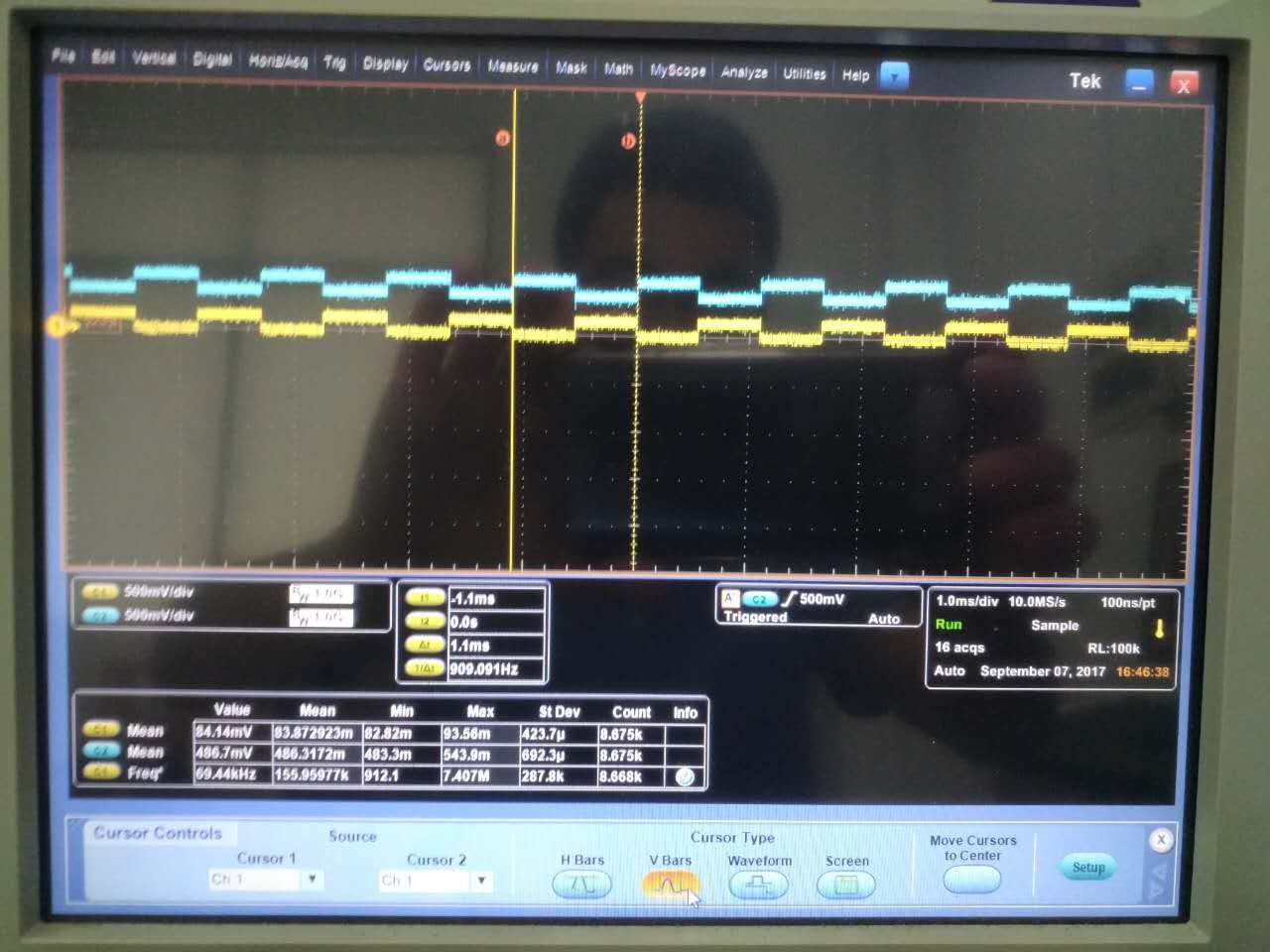


图 3 AD9715信号跳变测试结果波形图（1）



图 4 AD9715信号跳变测试结果波形图（2）

我们发现，在相同的采样率下，输出信号的衰减幅度和输入信号的跳变幅度有关，输入信号跳变幅度越大，输出信号的衰减幅度就越大。因此，第9比特跳变的信号最高采样率只有900SPS，第8比特跳变的信号的最高采样率为1800SPS，依此类推。信号衰减的时间大概在50~100ms左右。但是这些结论对解决问题没有起到太大帮助。

### 2.2.2 RCML的作用和影响

无奈之下，我们只能利用UART方便交互的特性，对AD9715的寄存器逐一进行读写，希望能找到一些解决问题的线索。在尝试到第6个寄存器（Reg 0x05, IRCML）的时候，不可思议的事情发生了：我们向这个寄存器写入0x08，输出信号的变化范围立刻恢复了正常，再也没有出现衰减的现象。在此基础上，我们又进行了200KSPS的矩形波和锯齿波测试，都得到了正常的波形：

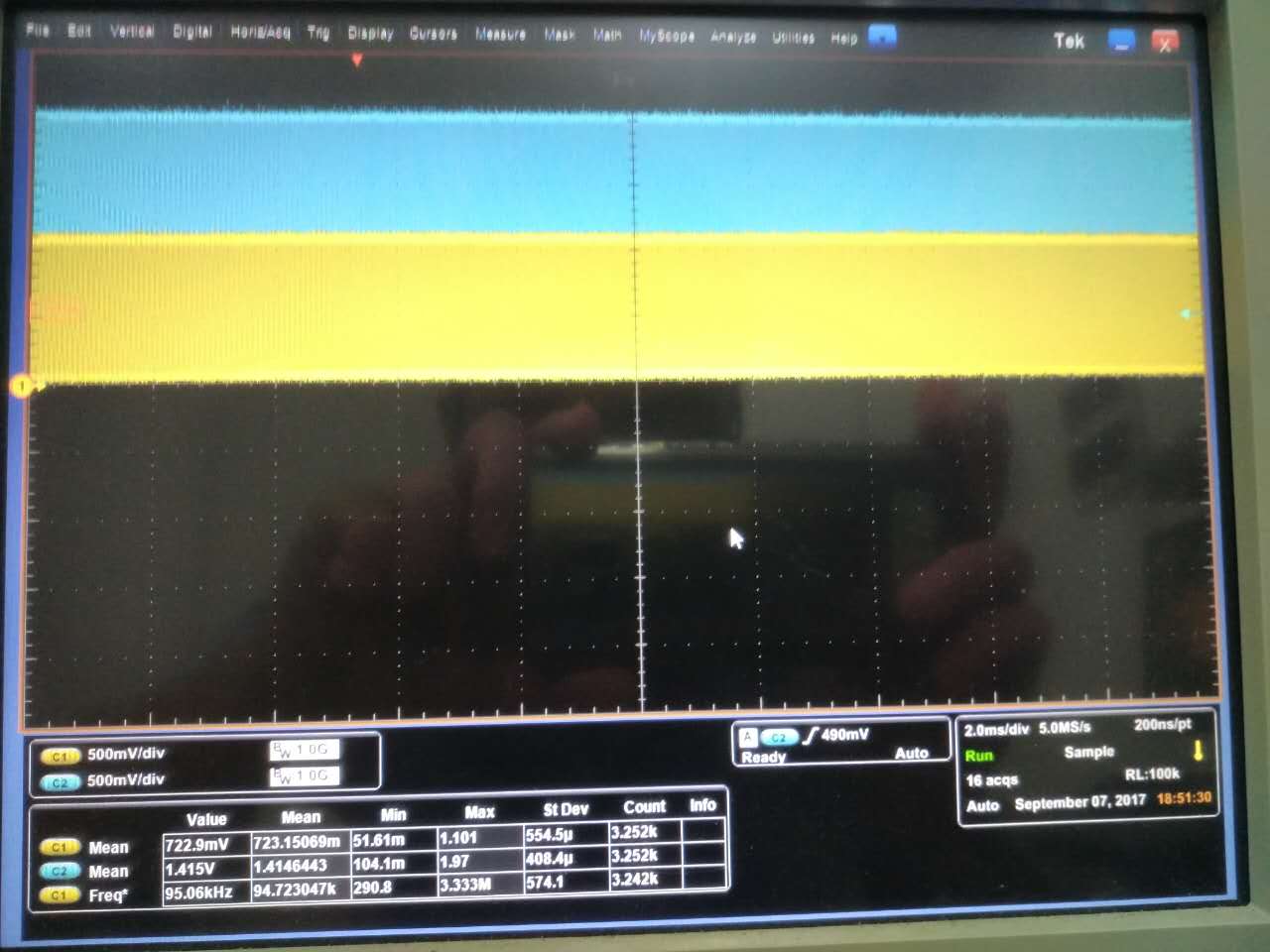


图 5 200KSPS矩形波测试波形图

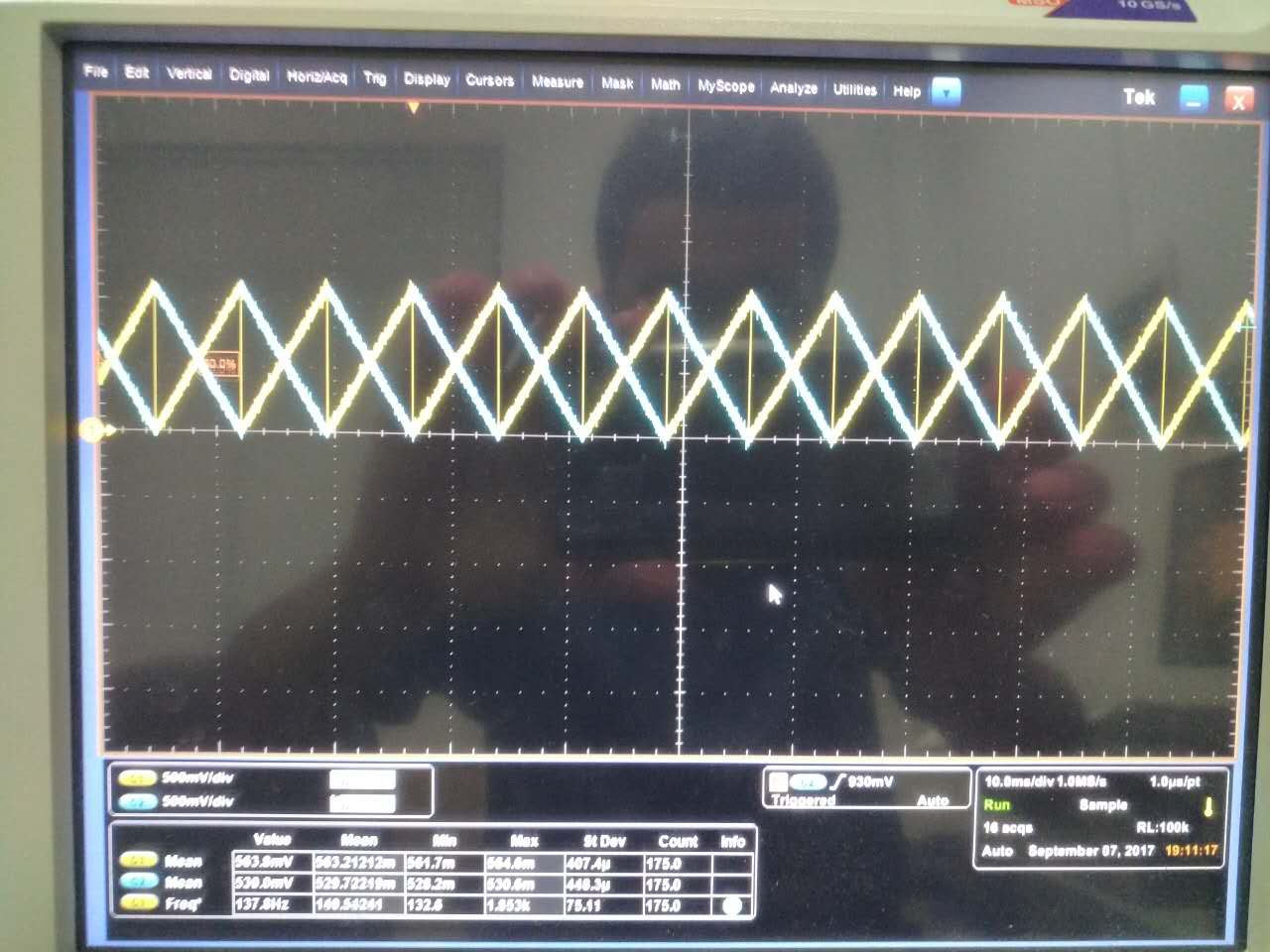


图 6 200KSPS锯齿波测试波形图

经过反复测试，我们确认引脚CML就是导致输出信号幅度范围衰减的原因。王老师为我们分析了其中的原理：根据数据手册，引脚CML起到调节共模电压的作用，AD9715是通过这个引脚来确定输出模拟地的位置，因此CML必须通过某种方式接地（如下图所示）。

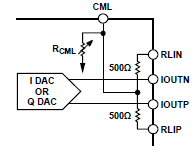


图 7 RCML的作用原理图[2]

可以通过使能内部的RCML，也可以通过外部的电阻接地。解决了这个问题，AD9715就可以正常工作了。

# 3 多种波形、频率控制与第六次上板调试的结果

为了实现多种波形的选择与频率控制等功能，我们对固件和程序又进行了一些修改。

## 3.1 多种波形、频率控制的固件设计

### 3.1.1 寄存器定义的修改

为了产生高频率的波形，必须要在固件层面给予支持，因为软件层面产生的波形频率不够高而且不够稳定。所以我们首先对固件的寄存器定义进行了修改，以提供对多种波形以及频率控制的支持，具体定义如下表所示：



图 8 DAC固件寄存器定义v2.1.0

其中，波形控制位的定义如下：



图 9 DAC固件寄存器波形控制位定义

### 3.1.2 波形数据产生器的实现

其次，我们需要实现不同的波形数据产生器。其中，方波和锯齿波比较简单。方波就是一个计数器加一个比较器；锯齿波就是一个计数器。而正弦波的产生就比较麻烦，为了简单起见，我们计划把正弦数据事先生成好，而不是通过实时计算得到。然而如何存储生成好的数据也是值得思考的问题。

网上提到的一种方法是用Verilog声明内存，再用生成的数据文件初始化内存。不过这种方法稍微有点麻烦。我们这次直接采用的是数据查找表的方法。

用下面的MatLab程序可以生成正弦函数的Verilog数据查找表，一共256个数据点，数据变化的范围为0到1023。

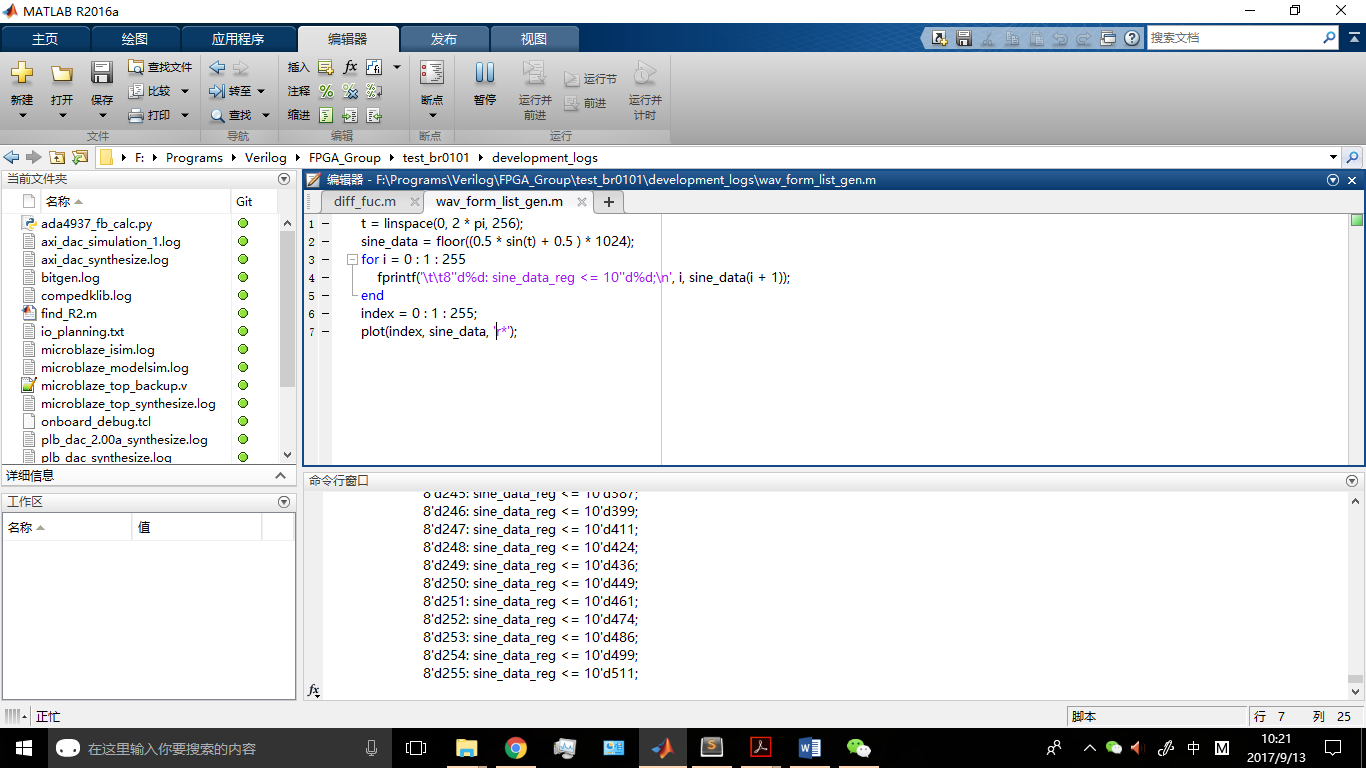


图 10 利用MATLAB程序产生波形数据

将信号index输入波形产生器模块，就可以得到相应的波形数据。

### 3.1.3 波形的选择与控制

最后我们需要实现对波形的选择与控制。波形的选择实际上就是对DAC输入数据的选择，只需要在输入信号前面加上数据选择器就可以。

另外，为了实现频率的控制，我们借鉴了DDS（Direct Digital Synthesizer）的算法，通过步长来控制相位的变化，进而控制频率。我们用4个比特代表步长，可以从1变化到16。这样每个周期index增大一个步长，再去查表，得到就是下一个波形数据。根据估算，这种方法下可调整的频率范围为97.7kHz到1.56MHz。但是在高频率下，由于步长太大，波形会出现明显的失真。

## 3.2 多种波形、频率控制的软件设计

为了对多种波形、频率控制的功能进行测试，我们还需要设计相应的驱动程序、测试程序和用户界面。

### 3.2.1 用户界面与UART中断的应用

我们采用UART中断来实现与用户的交互，因为中断的方式更加灵活，能够更方便地接收命令。

在Microblaze系统中，提供了很多与UART中断相关的驱动程序，我们需要用它们来初始化和使用UART中断。初始化的过程如下所示：

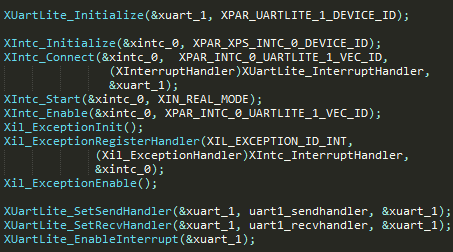


图 11 UART中断初始化的过程

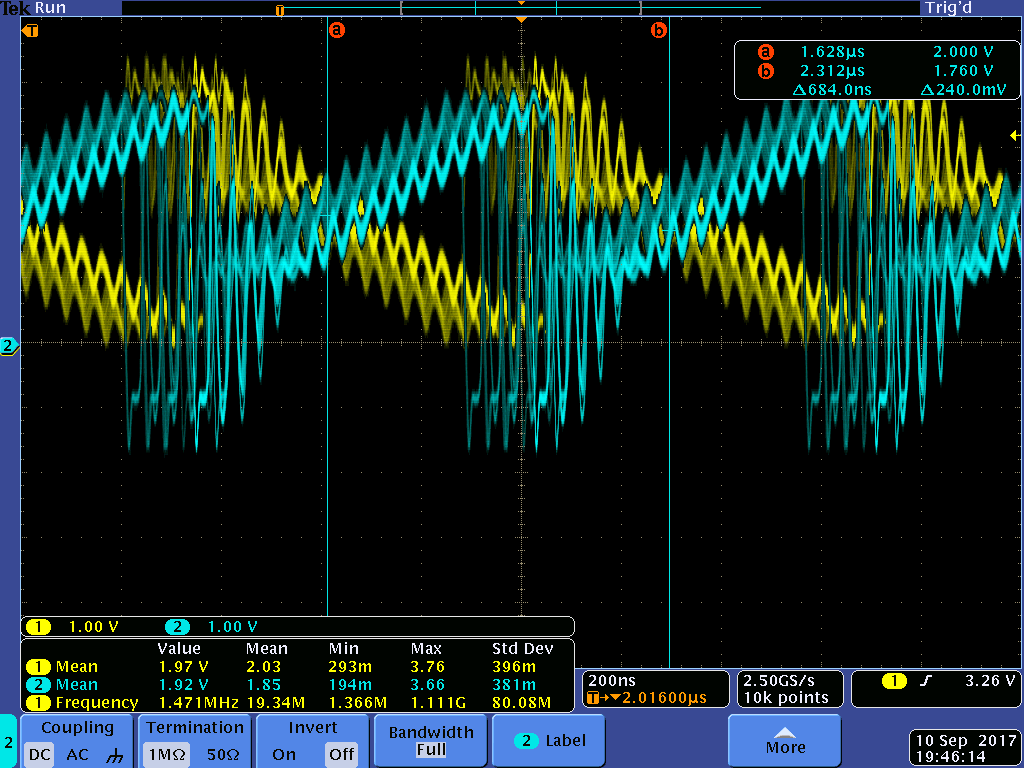
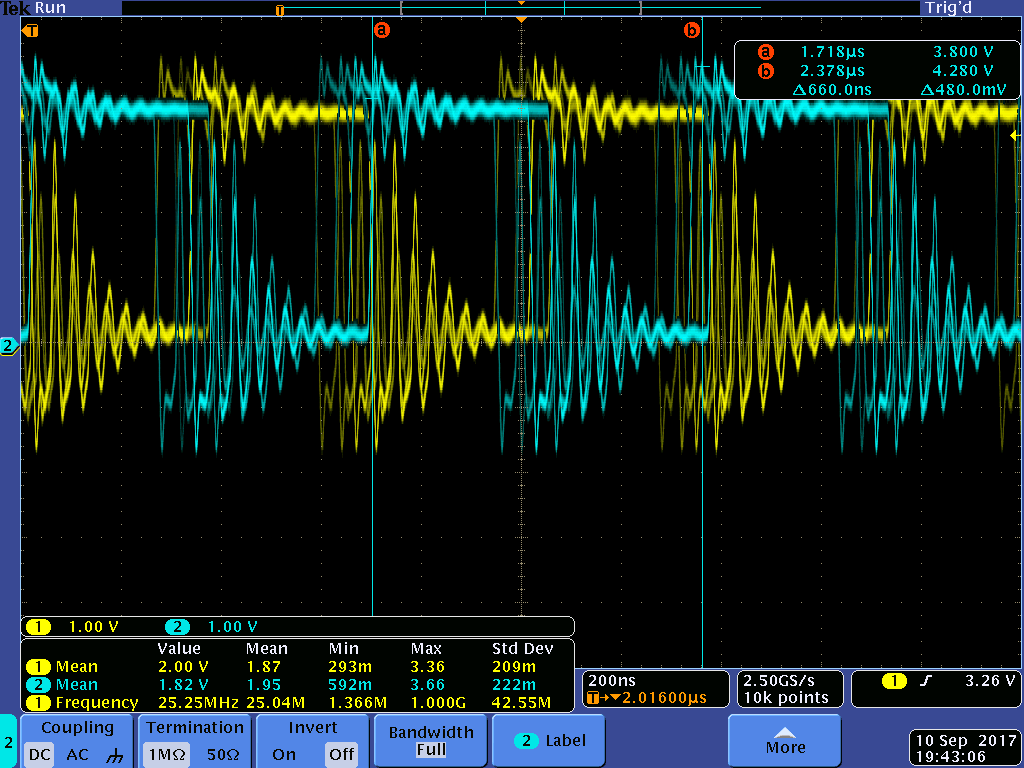
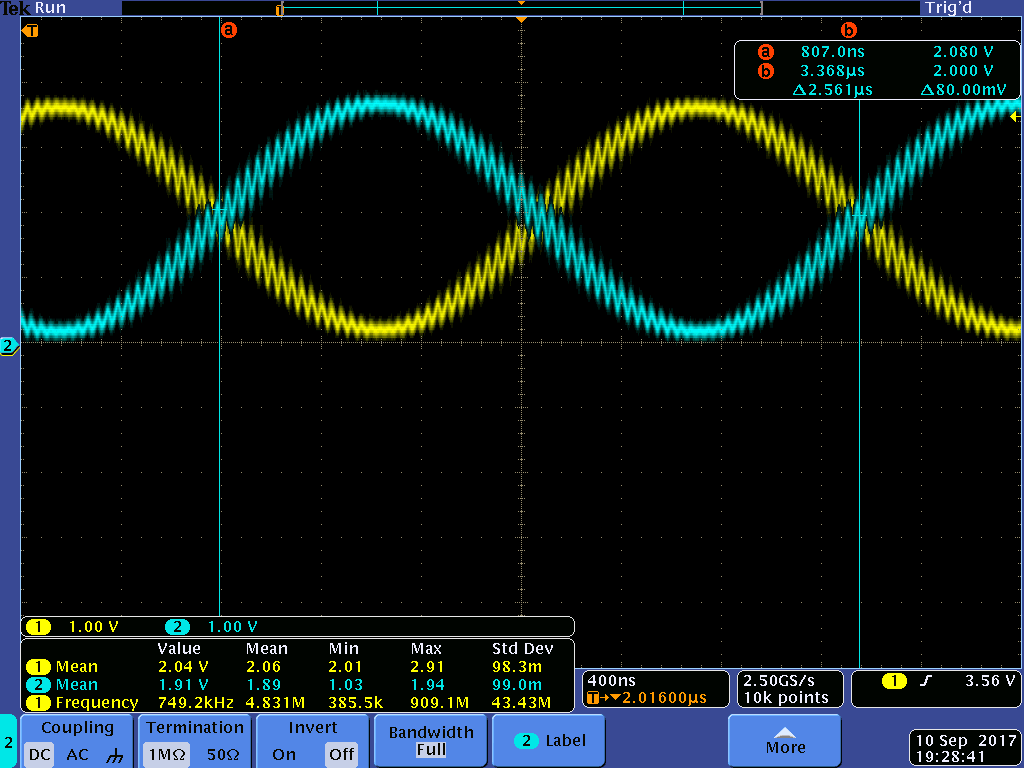
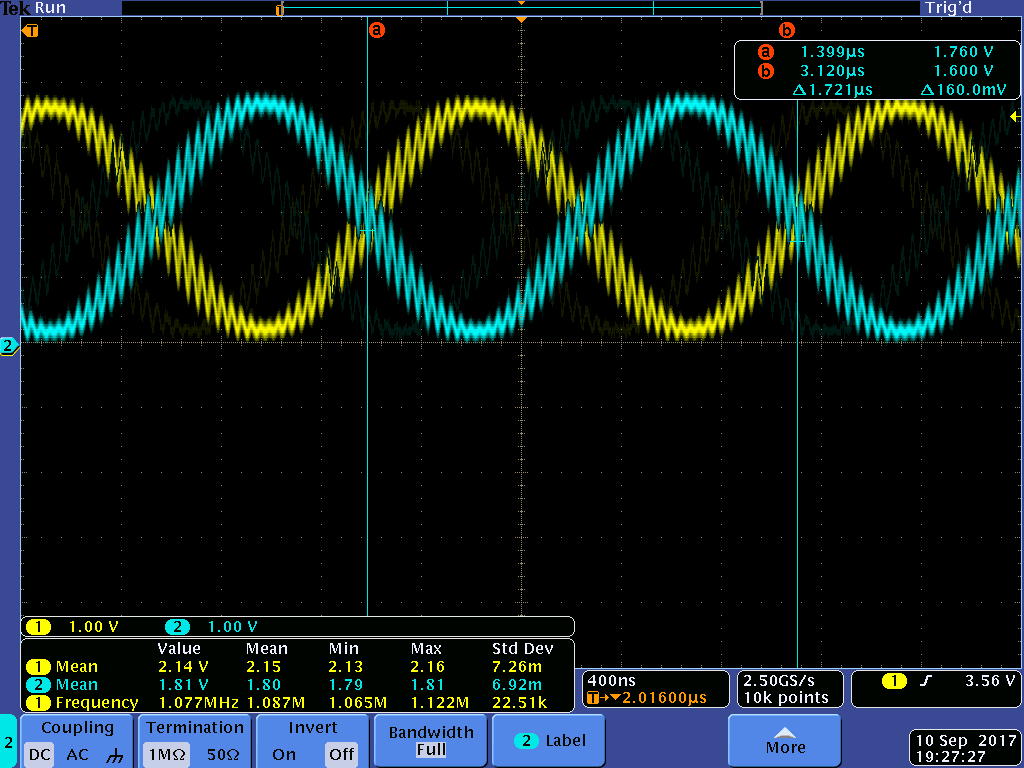
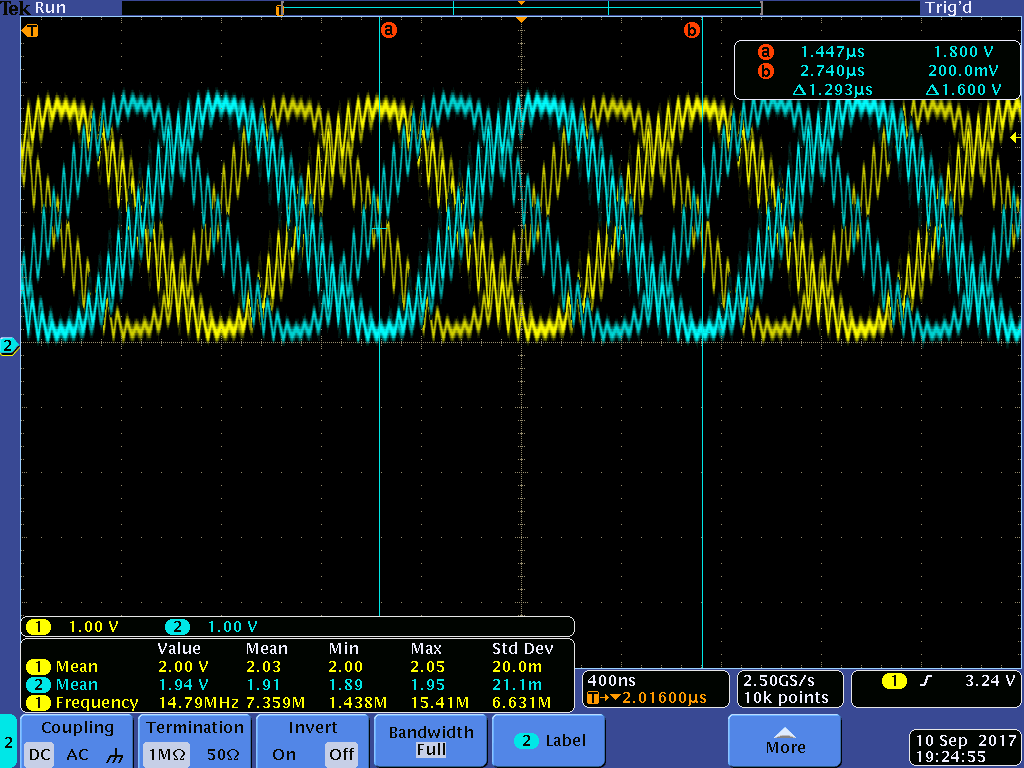
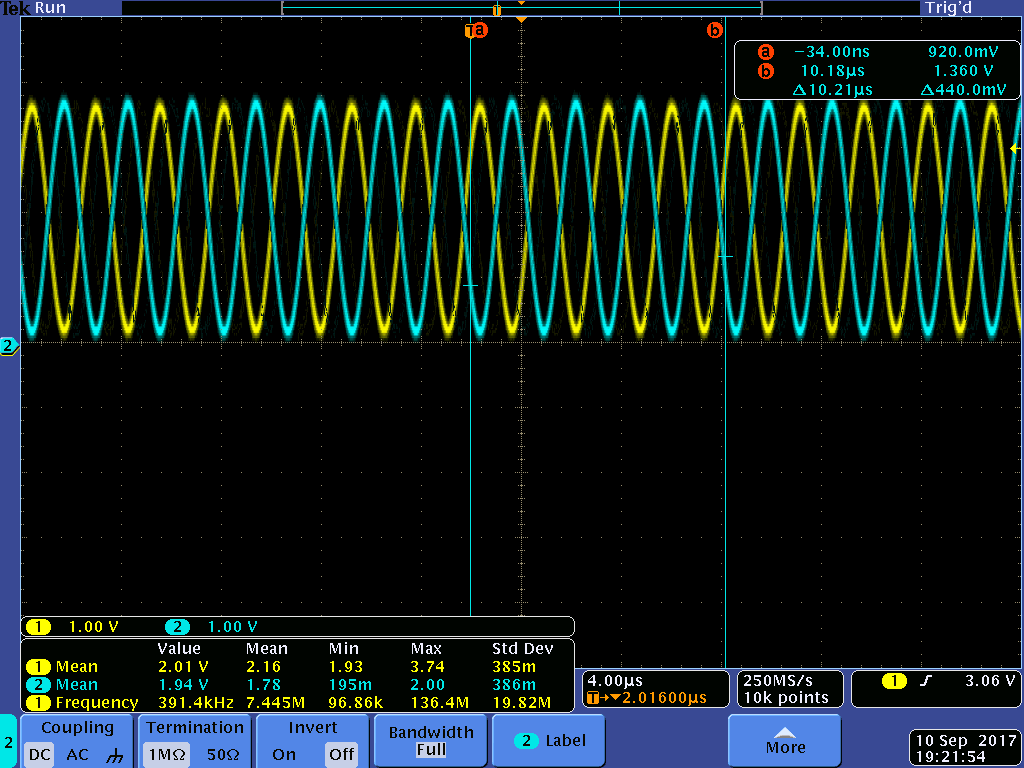
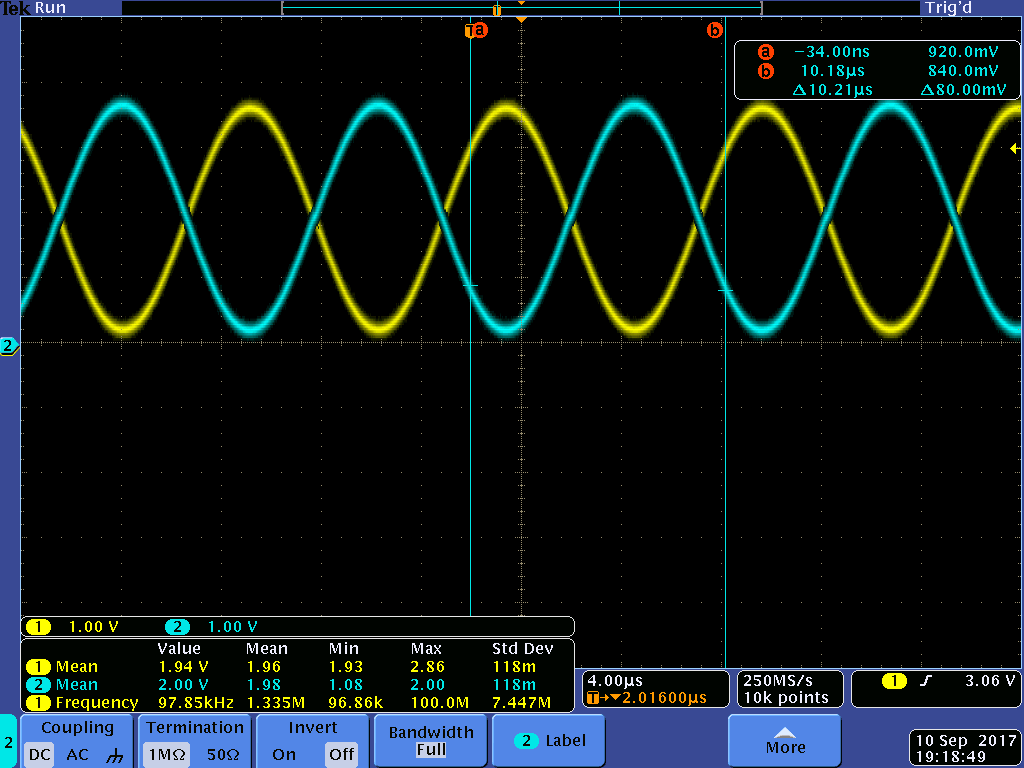
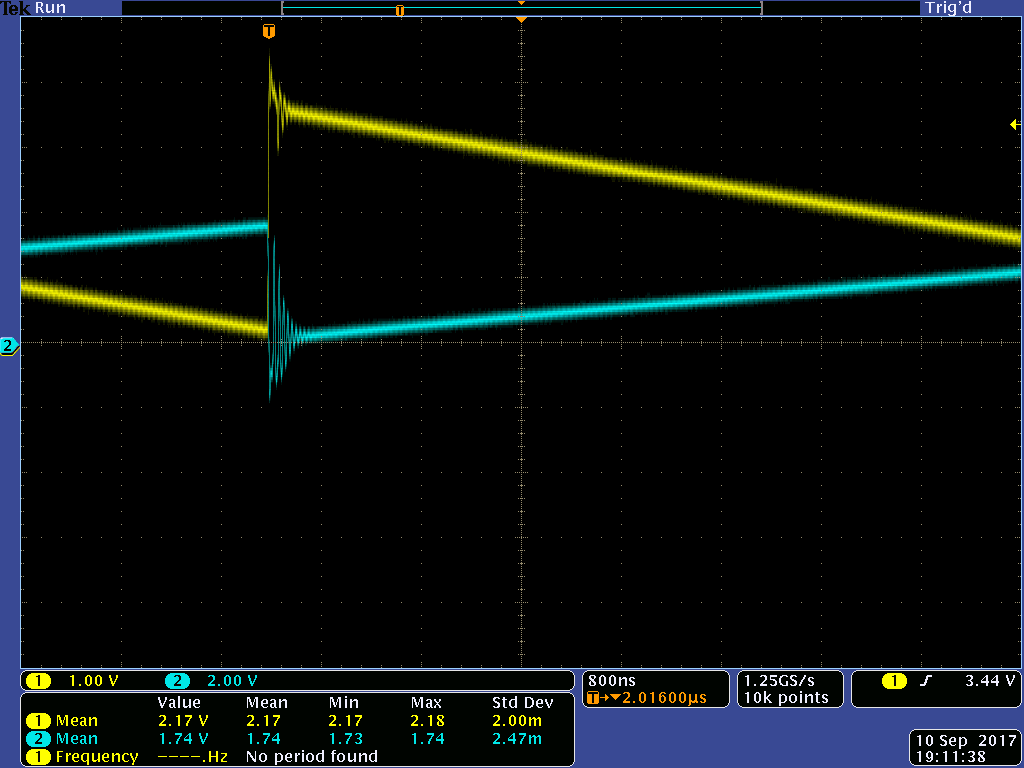
主要的步骤是初始化UART，初始化中断控制器，将中断控制器与UART连接起来，启动中断控制器，将中断控制器中控制UART的部分使能，初始化异常处理，注册中断处理器，使能异常处理，配置接收和发送中断处理程序，最后使能UART中断。

在接收中断处理程序中，可以根据接收到的字符对固件和DAC的状态进行调整，这样就实现了简单的用户界面。

### 3.2.2 用UART控制波形和频率

在UART接收中断处理程序中，如果接收到“R”、“S”、“C”、“D”等字符，就控制固件输出相应的波形；如果接收到“+”、“-”等字符，就控制固件输出信号的频率增大或减小；如果接收到以“;”结尾的16进制数据，就控制固件通过SPI传递出去。这样就完成了UART对波形和频率的控制。

## 3.3 第六次调试的结果



# 参考资料

1. Lingjun Zhu, [Low Sampling Rate of AD9715](https://ez.analog.com/thread/98659)
2. ADI, [AD9715 Datasheet](http://www.analog.com/media/en/technical-documentation/data-sheets/AD9714_9715_9716_9717.pdf)