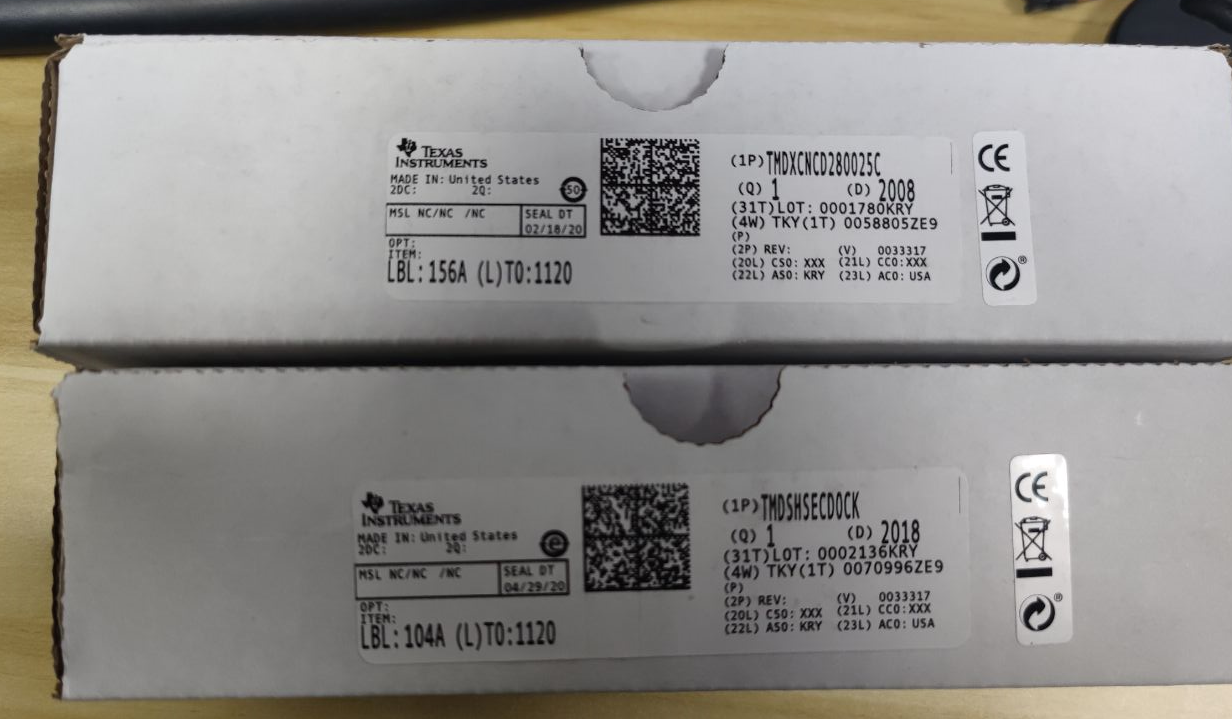
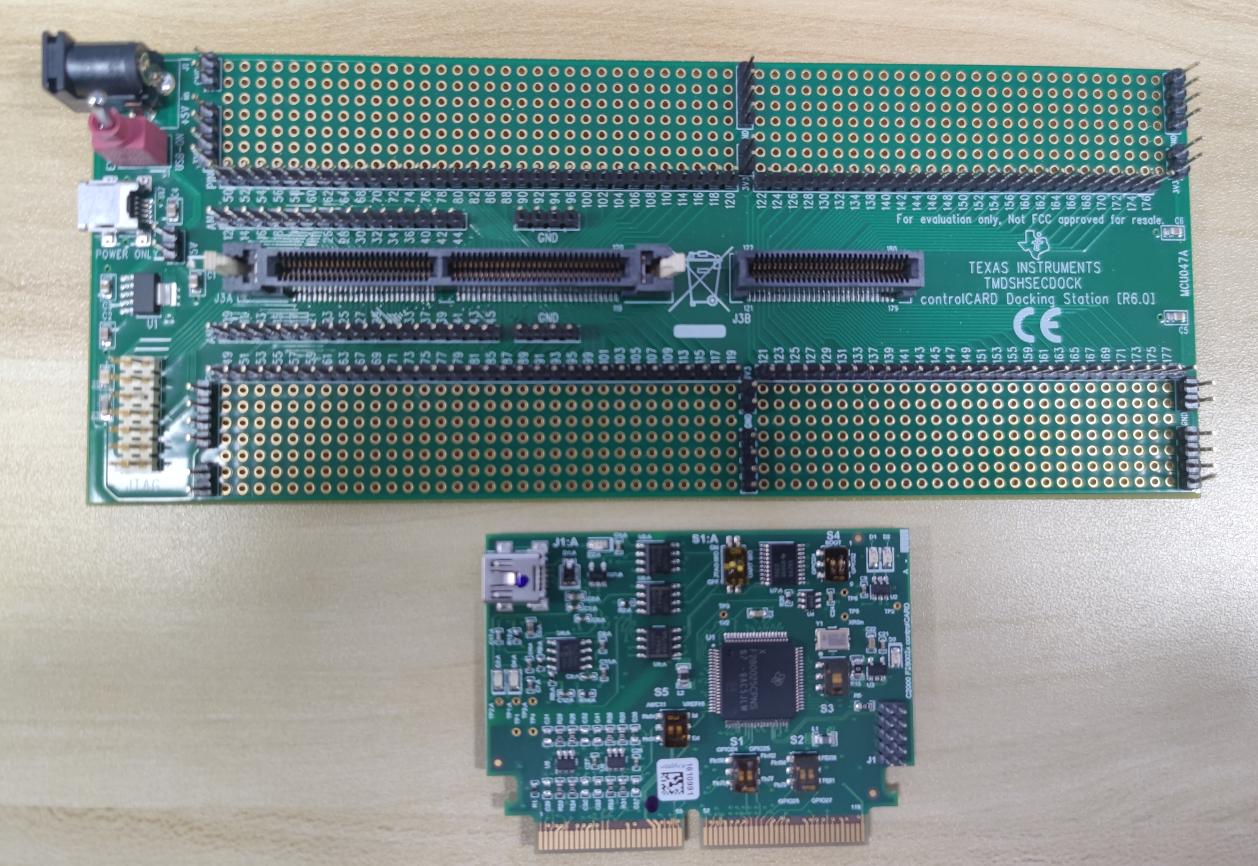
TMSXCNCD280025C测评报告

1、开箱简介

拿到手的开发板是分别装在两个盒子内的，盒子包装结实、实用。



打开包装看到的板子制作还是比较精良的，几个座子，开关，接头用的都是质量很不错的料，对于我们这种有质量洁癖的人来说，还是挺赏心悦目的。

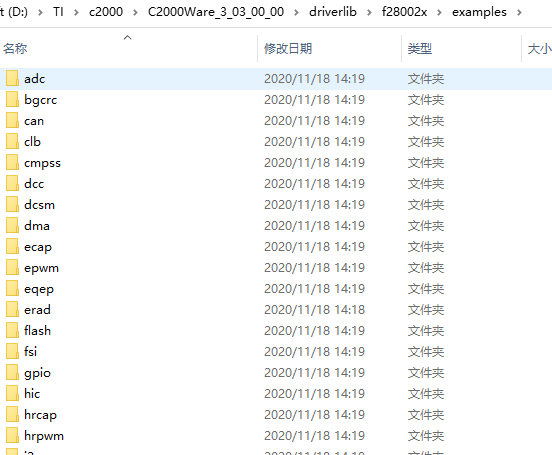


2、开发板资源

开发板的原理图，PCB图，都在C2000ware的文件夹内，



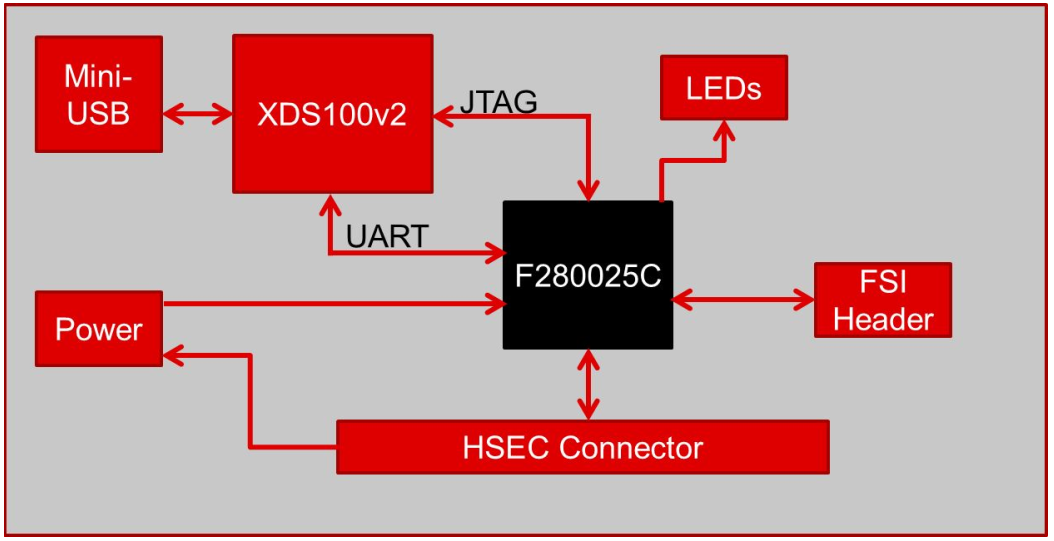
例程在C2000ware的driverlib中。



这个例程非常好用，提供了他所具有的模块的例程，而且每个模块的使用都提供了几种典型例子。

3、开发板框架

开发板的功能块如下，



HSEC引出的接口有EPWM接口、ADC采样口、串口、GPIO口等常用资源。本身板块设计比较紧凑，通过金手指把常用的资源都引出来，并通过拓展板拓展，对于调试人员来说，比较友好，便于做各种测试。

4、测评

本次测评结合我们正在开发的一个传感设备进行，使用到的模块有EPWM、ADC、SCI。EPWM用于控制LED灯以及泵，ADC用于采样光电管和压力传感器的模拟信号，280025C接受到信号后进行处理，并通过SCI串口上传到上位机上。基本的功能连接如下：



4.1 EPWM测试

需要引两路EPWM输出，分别设置为EPWM1\_A和EPWM1\_B，对应板块HSEC的管脚是49与51。

**GPIO\_setPadConfig**(0, GPIO\_PIN\_TYPE\_STD);

**GPIO\_setPinConfig**(GPIO\_0\_EPWM1\_A);

**GPIO\_setPadConfig**(1, GPIO\_PIN\_TYPE\_STD);

**GPIO\_setPinConfig**(GPIO\_1\_EPWM1\_B);

设置默认的EPWM参数如下，频率为1k，占空比分别为50%，EPWM的计数模式为加减模式。

EPWM\_SignalParams pwmSignal1 =

{1000, 0.5f, 0.5f, **false**, DEVICE\_SYSCLK\_FREQ,

*EPWM\_COUNTER\_MODE\_UP\_DOWN*, *EPWM\_CLOCK\_DIVIDER\_1*,

*EPWM\_HSCLOCK\_DIVIDER\_1*};

4.2 ADC测试

ADC需要采集一个光电管的信息还有气泵充气的气室压力，选择A0和A1通道作为采样通道，对应HSEC管脚为pin9和pin15。直接采用库函数进行调用，关键的配置函数如下：

ADC\_forceSOC(ADCA\_BASE, *ADC\_SOC\_NUMBER0*);

ADC\_forceSOC(ADCA\_BASE, *ADC\_SOC\_NUMBER1*);

adcAResult0 = ADC\_readResult(ADCARESULT\_BASE, *ADC\_SOC\_NUMBER0*);

adcAResult1 = ADC\_readResult(ADCARESULT\_BASE, *ADC\_SOC\_NUMBER1*);

用示波器测量压力传感器的输出电压值并同时观察ADC值变化，评估该功能是否正常。

4.3 SCI通信

配置为SCIA，对于HSEC接口为67与69。配置的串口波特率为9600，如下所示，

**#define** mySCI0\_BASE SCIA\_BASE

**#define** mySCI0\_BAUDRATE 9600

由于上位机是有特定的接受协议，因此传输协议也按照上位机的协议进行设置。

**void** **SCI\_transmit\_frame**(uint32\_t base)

{

**unsigned** **char** send\_data[43];

**unsigned** **char** \*send\_msg;

**int** i;

**int** length;//length of frame

**int** check;

length = 43;

check =0;

send\_data[0] = 0xb6; //head

send\_data[1] = 0xb6; //head

send\_data[2] = length>>8; //high 8 bit of length of frame

send\_data[3] = length; //low 8 bit of length of frame

send\_data[4] = 0x01; //protocal version

send\_data[5] = 0x02; //modular type

**for**(i=0;i<7;i++)

{

**if**(i == 5)

f2c(PPG1\_ADC\_filter, send\_data+6+4\*i);

**else** **if** (i == 6)

f2c(PPG2\_ADC\_filter, send\_data+6+4\*i);

**else**

f2c(ADC\_val[i], send\_data+6+4\*i);

}

send\_data[34] = pos\_pump\_status;

send\_data[35] = neg\_pump1\_status;

send\_data[36] = neg\_pump2\_status;

send\_data[37] = ch1\_air\_in\_status;

send\_data[38] = ch1\_air\_out\_status;

send\_data[39] = ch2\_air\_in\_status;

send\_data[40] = ch2\_air\_out\_status;

send\_data[41] = check;

send\_data[42] = 0x6b;

// for(i=0;i<36;i++)

// {

// SCI\_writeCharNonBlocking(base, send\_data[i]);

// }

send\_msg = send\_data;

SCI\_writeCharArray(base, (uint16\_t\*)send\_msg, 43);

}

只上传PPG1\_ADC\_filter、ADC\_val[2]，其余的变量值均为0。

上位机接受到的数据并绘制成图如下所示，控制气泵的PWM掌控比从0增加到1再从1减少到0，可发现压力传感器采集到的压力值也随着变化，而该气室包裹这人体组织，LED灯照射被包裹的人体组织，用光电管接受输出的光，光也随着气室压力的变化而变化。





实验连接图如下



5、总结

由于只是简单测试各个模块的功能，所以并没有写很复杂的程序，只是把自己功能块堆在一次做一个简单的测试，后续随着项目的推荐，我会往上面集成更多的功能，届时如果有机会再分享。

这个板块设计得比较紧凑，使用也比较方便，DSP的pin脚虽然不多，但是功能丰富，对于我们这种带有多种外设的应用场合，还是比较实用的，具有比较高的性价比。

该DSP芯片的功能还算是比较丰富，几个最常用的功能是都有，其中比较亮眼的应该是CLB模块，这个模块这次测评并没有使用，在完成上述的基本测试后，后序如果我还会抽空测试一下该板卡的CLB模块。

在开发过程中，我的最大的感受是，借助TI提供的库函数，能够很方便的在短时间内实现你要的驱动功能，这个库函数省去了我们开发者大量的写寄存器的时间，能够运用好，对于加快我们的开发进程有非常大的帮助。