TMS570LS MCU Driver Report for Analog Device LTC6812 IC

xEV & ESS Battery Team - Ankara

Mehmet Dinçer

mehmet.dincer@aspilsan.com

dmehmett38@gmail.com

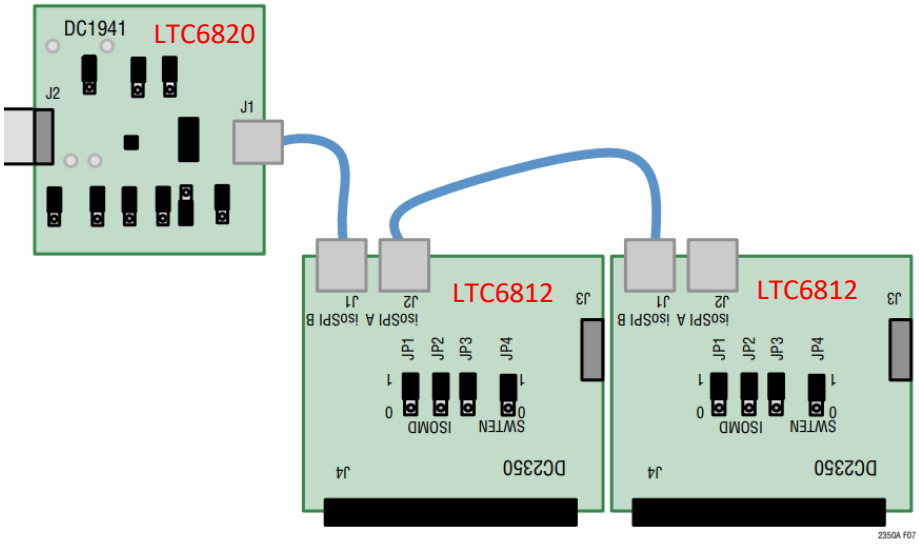
Şubat-2024

1.0- LTC6812

LTC6812-1, çok hücreli pil paketi izleme entegresidir ve toplam ölçüm hatası 2.2mV'nin altında 15 seri bağlı pil hücresini ölçer. 0V ile 5V arasındaki hücre ölçüm aralığı, LTC6812-1'i çoğu pil kimyası için uygun hale getirir. Birden fazla LTC6812-1 cihazı seri olarak bağlanabilir, bu da uzun, yüksek gerilimli pil dizilerinin aynı anda hücre izleme olanağı sağlar. Her bir LTC6812-1'in, yüksek hızlı, RF bağışıklıklı, uzun mesafe iletişimi için isoSPI arabirimi bulunmaktadır. Birden fazla cihaz, tek bir ana işlemci bağlantısıyla birbirine daisy chain olarak bağlanır. Bu daisy chain, iletişim yolu boyunca bir hatanın olması durumunda bile iletişim bütünlüğünü sağlamak üzere iki yönlü olarak çalıştırılabilir. LTC6812-1 doğrudan pil yığınından veya izole bir kaynaktan beslenebilir. LTC6812-1, her bir hücre için pasif dengeleme içerir ve her hücre için bireysel PWM görev döngüsü kontrolüne sahiptir. Diğer özellikler arasında bir kart üzerinde 5V regülatör, dokuz genel amaçlı I/O hatları ve 6µA'ye kadar düşen bir uyku modu bulunmaktadır.

2.0- ISO-SPI

LTC6820, tek bir örülü tel bağlantısı üzerinden iki izole cihaz arasında çift yönlü SPI iletişim sağlayan bir entegredir. Her LTC6820, mantık durumlarını sinyallere kodlar ve bu sinyalleri izolasyon bariyerini aşarak başka bir LTC6820'ye iletilir. Alıcı LTC6820, iletimi çözümler ve köle veri yolunu uygun mantık durumlarına sürer. İzolasyon bariyeri, basit bir darbe transformatörü kullanılarak geçilebilir, böylece yüksek izolasyon seviyeleri elde edilebilir. LTC6820, eşleştirilmiş kaynak ve batma akımlarını kullanarak diferansiyel sinyalleri sürer, bir transformatör merkez teli gerekliliğini ortadan kaldırır ve elektromanyetik girişimi azaltır. Alıcıda bulunan hassas pencere karşılaştırıcıları, diferansiyel sinyalleri tespit eder. Sürücü akımları ve karşılaştırıcı eşikleri, basit bir harici direnç bölücü kullanılarak ayarlanabilir, bu da sistemin istenen kablo uzunluklarına ve sinyal gürültü performansına göre optimize edilmesine imkân tanır. LTC6820, izole SPI iletişim çözümlerinde güvenilir ve yüksek performanslı bir seçenek sunar. İdeal kablolama uzunlukları ve istenilen sinyal-gürültü performansı için sistemi optimize etme esnekliği sağlar."

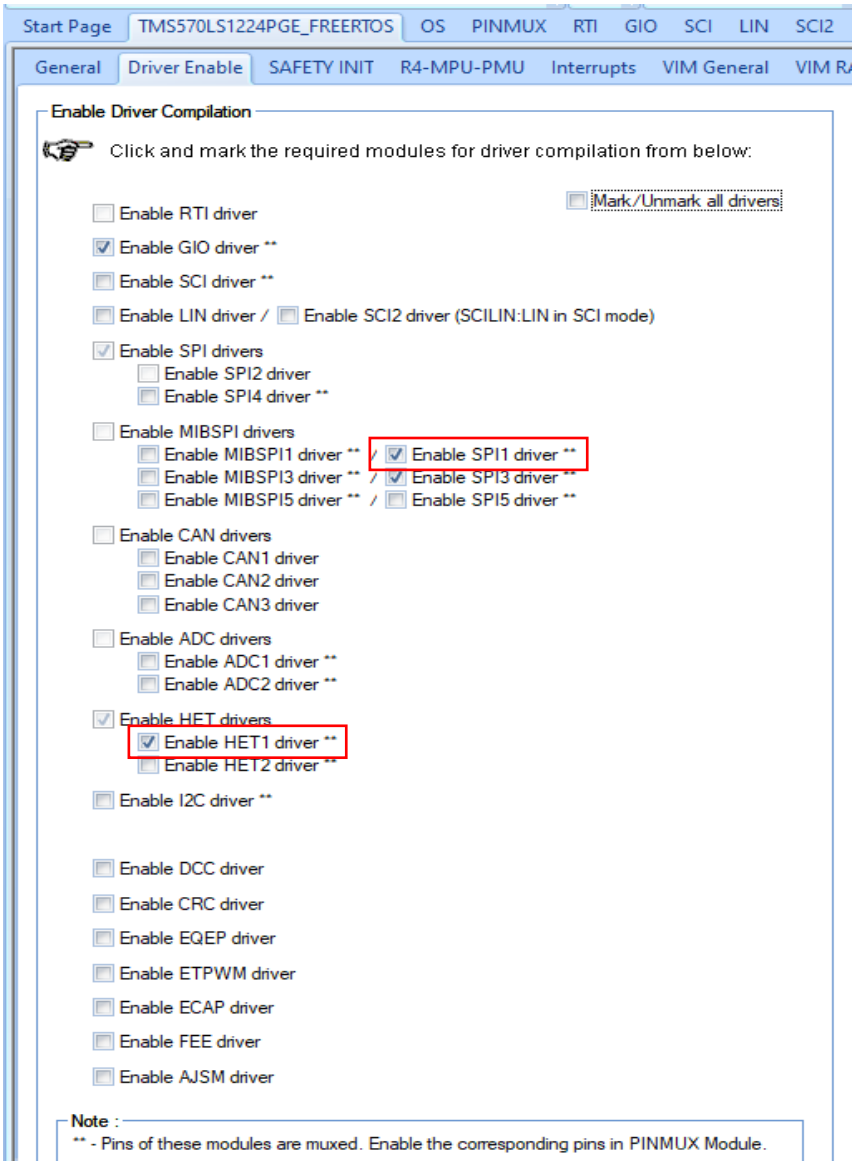


Şekil 1: Ltc6820 ve Ltc6812'nin oluşturduğu zincir yapısı

3.0- MCU Konfigürasyonu

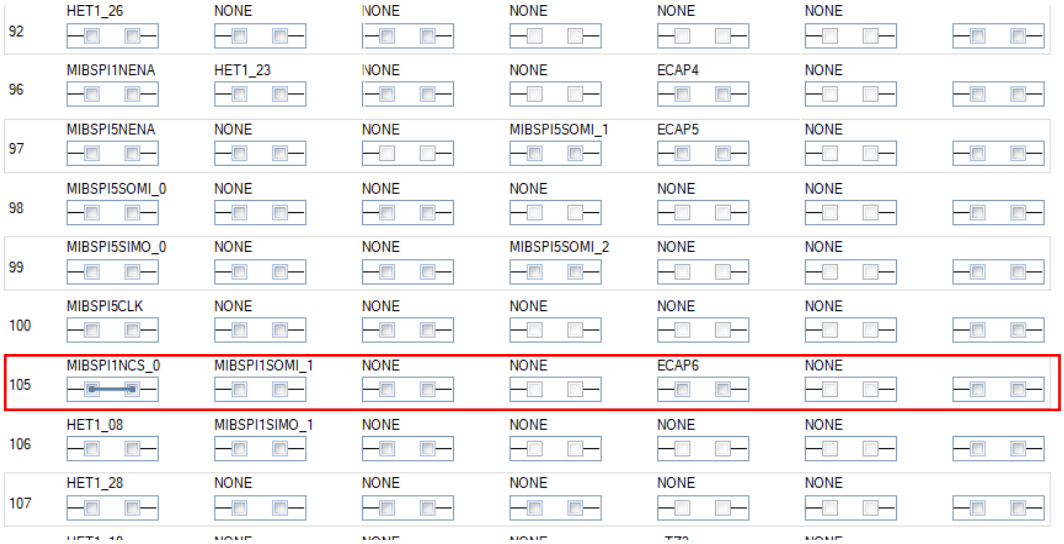
LTC6812 entegresi, SPI protokolü üzerinden 1MHz bant genişliğinde, 8-bit veri paketleri ile iletişim kurmaktadır. Bu ayarların yanı sıra, LTC6812 entegresinde milisaniye ve mikrosaniye beklemelerini gerçekleştirmek üzere HET-1 peripheralının PWM periyodu 10 mikrosaniye olarak belirlenmeli ve periyot tamamlama kesmeleri etkinleştirilmelidir.

3.1-

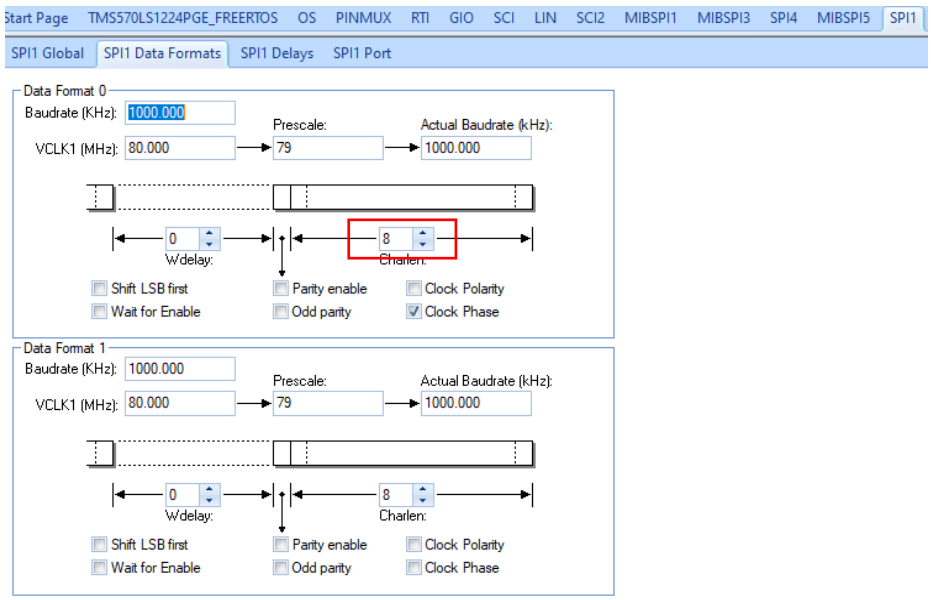


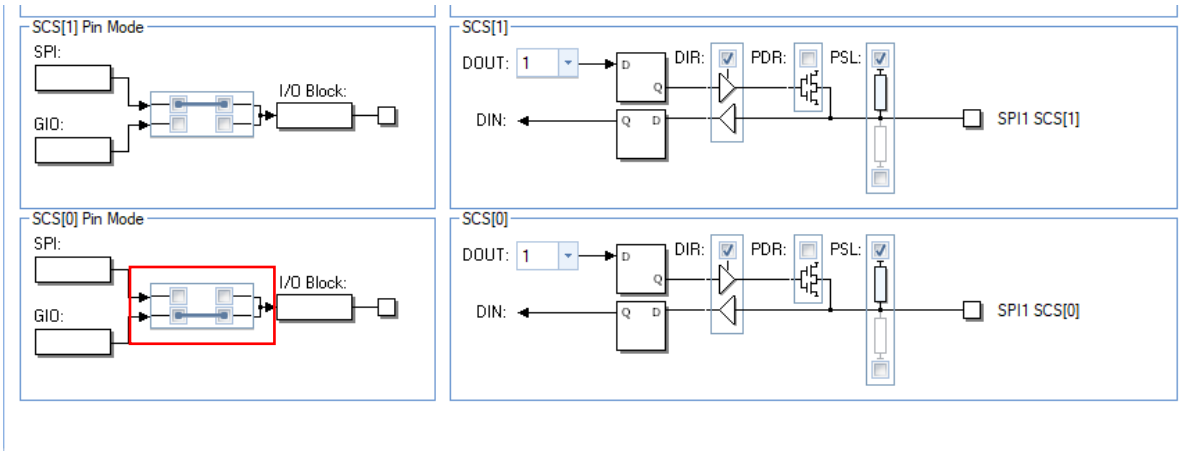
3.2-

İlgili spi ve het pinlerini seçin

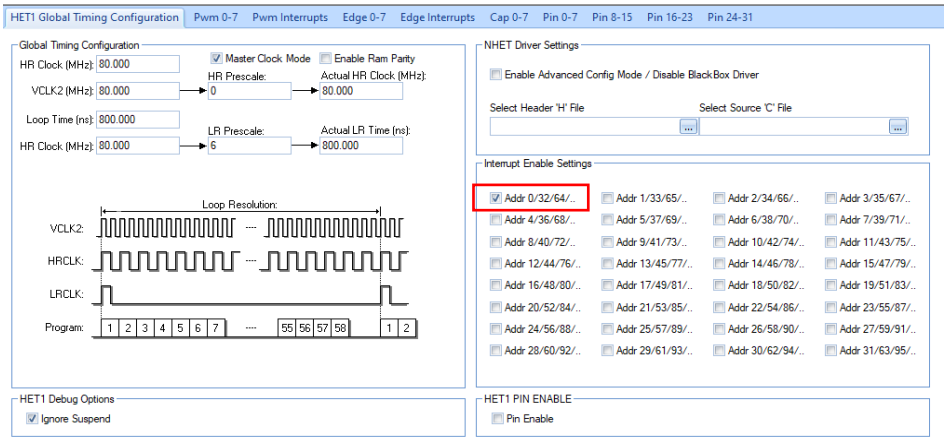


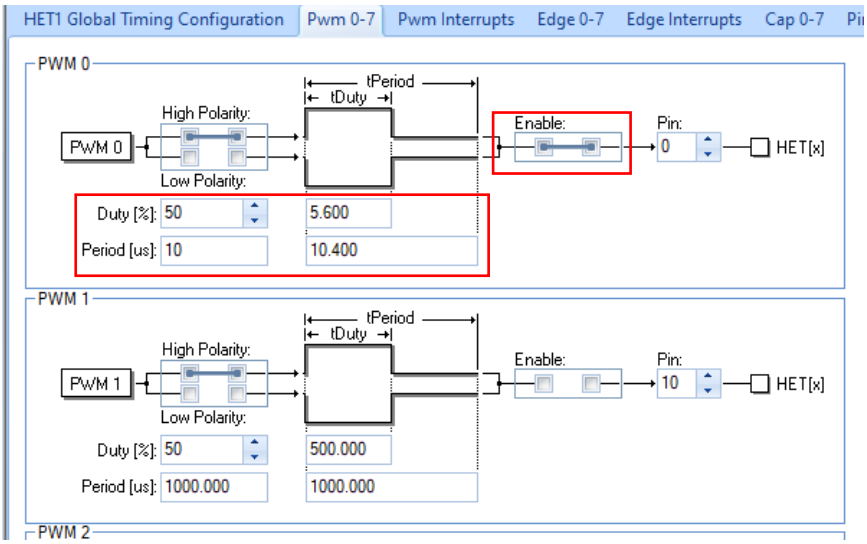
3.3-

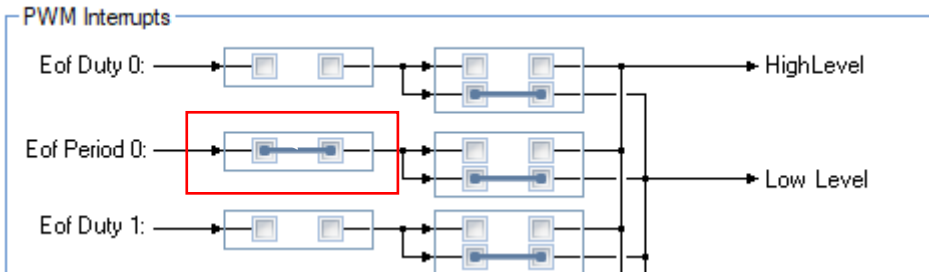




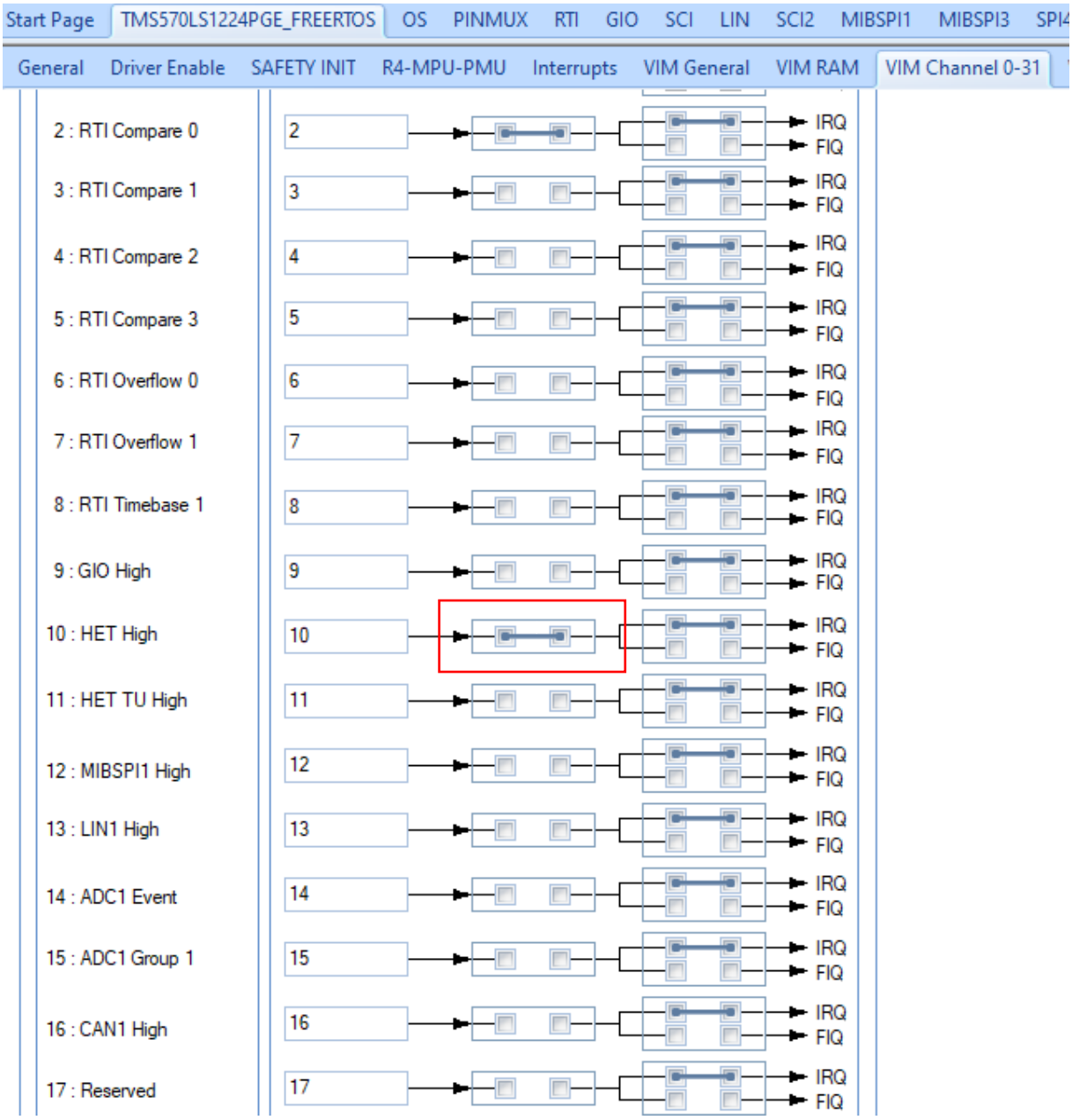
3.4-


HET1 Global Timing Configuration
Pwm 0-7
Pwm Interrupts
Edge 0-7
Edge Interrupts
Cap 0-7
Pin 0-7
Pin 8-15
Pin 16-23
Pin 24-31
Global Timing Configuration
HR Clock (MHz): 80.000
Master Clock Mode
Enable Ram Parity
VCLK2 (MHz): 80.000
HR Prescale: 0
Actual HR Clock (MHz): 80.000
Loop Time (ns): 800.000
HR Clock (MHz): 80.000
LR Prescale: 6
Actual LR Time (ns): 800.000
Loop Resolution:
VCLK2:
HRCLK:
LRCLK:
Program:
NHET Driver Settings
Enable Advanced Config Mode / Disable BlackBox Driver
Select Header 'H' File
Select Source 'C' File
Interrupt Enable Settings
HET1 Debug Options
Ignore Suspend
HET1 PIN ENABLE
Pin Enable


HET1 Global Timing Configuration
Pwm 0-7
Pwm Interrupts
Edge 0-7
Edge Interrupts
Cap 0-7
PWM 0
tPeriod
tDuty
High Polarity:
Low Polarity:
PWM 0
Enable:
Pin:
0
HET[x]
Duty [%]: 50
5.600
Period [us]: 10
10.400
PWM 1
tPeriod
tDuty
High Polarity:
Low Polarity:
PWM 1
Enable:
Pin:
10
HET[x]
Duty [%]: 50
500.000
Period [us]: 1000.000
1000.000
PWM 2


PWM Interrupts
Eof Duty 0:
Eof Period 0:
Eof Duty 1:
HighLevel
Low Level

3.5-


Start Page
TMS570LS1224PGE_FREERTOS
OS
PINMUX
RTI
GIO
SCI
LIN
SCI2
MIBSPI1
MIBSPI3
SPI4
General
Driver Enable
SAFETY INIT
R4-MPU-PMU
Interrupts
VIM General
VIM RAM
VIM Channel 0-31
2 : RTI Compare 0
3 : RTI Compare 1
4 : RTI Compare 2
5 : RTI Compare 3
6 : RTI Overflow 0
7 : RTI Overflow 1
8 : RTI Timebase 1
9 : GIO High
10 : HET High
11 : HET TU High
12 : MIBSPI1 High
13 : LIN1 High
14 : ADC1 Event
15 : ADC1 Group 1
16 : CAN1 High
17 : Reserved
IRQ
FIQ

4.0- Kod Yapısı

Kaynak kodları, donanımı soyutlamaya yönelik olarak geliştirildi. Bu geliştirme, Ltc682x yapısı çerçevesi üzerinden gerçekleştirildi. Bu yapı, her türlü değeri içeren Register'ların yazılması ve okunması için kullanıldı. Hücre gerilimleri, GPIO pinlerinden okunan gerilim değerleri, hücrelerin yüksek veya alçak voltaj bayrakları, sıcaklık ve anlık durum değerleri bu yapı içerisinde depolanmaktadır.

LTC682x Enum'ı sayesinde, işlev sonuçlarına bağlı olarak LTC_OK, LTC_WRONG_CRC, LTC_OPEN_WIRE gibi durumları döndererek, anlık durum takibi sağlanması amaçlanmıştır.

4.1- Kod Entegrasyonu

İlgili kaynak kodlar projeye eklendikten ve IDE'ye path olarak tanımlandıktan sonra main.c dosyası içerisinde void ltcInit(spiBASE_t * spiReg) fonksiyonunu oluşturunuz. Bu fonksiyon sistemdeki slave sayısını ve her bir slave de yer alacak hücre sayısı, sistemin çalışma modu… ve pinlerin pull konfigürasyonlarının yapılmasını sağlar.

Slave sayısı, cell sayısı ve gpioPull konfigürasyonları dışındaki ayarları değiştirmek bazı fonksiyonlarda hatalara yol açabilir.

```
/* USER CODE BEGIN (4) */

void ltcInit(spiBASE_t * spiReg)
{
    uint8_t i = 0;
    for(; i < SLAVE_NUMBER; i++)
    {
        ltcBat[i].batConf.adcopt = false;          //ADC conversion mode selection, if 1->14kHz, 3kHz, 1kHz or 2kHz, 0-> 27kHz, 7kHz, 422H
        ltcBat[i].batConf.refon = true;            //referances remain powered on until watchog timeout

        ltcBat[i].batConf.gioAPullOffPin = GPIO_5 | GPIO_4 | GPIO_3;   // selected pin's pull down off
        ltcBat[i].batConf.gioBPullOffPin = GPIO_8 | GPIO_7 | GPIO_6;   // selected pin's pull down off

        ltcBat[i].batConf.numberOfCell = 13;                    //!< cell number in a slave can assign difference cell with array
        ltcBat[i].batConf.numberOfSlave = SLAVE_NUMBER;         //!< number of slave
    }

    AE_ltcInit(spiReg, ltcBat);

    AE_delayMs(2);  // t-wakeup time
}
```

Bu ayarlar yapıldıktan sonra ltcInit(spiRegx) fonksiyonu int main () fonksiyonu içerisinde çağırmak driver'ı kullanıma hazır hala getirir.

4.3- Kod Kullanımı

Kaynak kodlar geliştirilirken her bir fonksiyon main () fonksiyonu içerisinde #if … #endif blokları arasında denenmiş, gerekli açıklamalar yapılmıştır. V1.0 bölümünde çevresel birimlerle ilgili, V1.1 bölümünde ltc681x ile ilgili kodlar bulunmaktadır.

4.3.1- Hücre Gerilimlerini Okuma

```
144 #if 0   // read the cell voltage
145         AE_ltcStartCellAdc(&ltcBat, MODE_7KHZ, true, CELL_ALL);
146         //!< check adcMeasure duration is completed
147         while(!AE_ltcAdcMeasureState());
148         status = AE_ltcReadCellVoltage(&ltcBat);
149 #endif
```

Şekilde gösterildiği gibi, hücre gerilimlerini okumak için önce hücrelerin ADC örnekleme işlemi başlatılır. Ardından, ADC okumasının tamamlanmasını beklenir. Bu işlemleri takiben hücre okuma işlemi gerçekleştirilir. LTC681x'ten okunan PEC değeri ile hesaplanan PEC değeri eşleştiğinde fonksiyon, LTC_OK durumunu döndürür; farklı olduğunda ise LTC_WRONG_CRC durumunu döndürür. Fonksiyona verilen MODE_7KHZ değişkeni ADC'nin frekansını, true parametresi deşarj'a izin verir, CELL_ALL parametresi hangi hücrelerin okunacağını temsil eder.

4.3.2- GPIO Gerilimlerini Okuma

```
151 #if 0   // GPIO voltage Measure
152         AE_ltcStartGpioAdc(&ltcBat, MODE_7KHZ, GPIO_ALL);
153         while(!AE_ltcAdcMeasureState());
154         status = AE_ltcReadGPIOVoltage(&ltcBat);
155 #endif
```

Şekilde gösterildiği gibi, GPIO gerilimlerini okumak için önce GPIO ADC örnekleme işlemi başlatılır. Ardından, ADC okumasının tamamlanmasını beklenir. Bu işlemleri takiben GPIO okuma işlemi gerçekleştirilir. LTC681x'ten okunan PEC değeri ile hesaplanan PEC değeri eşleştiğinde fonksiyon, LTC_OK durumunu döndürür; farklı olduğunda ise LTC_WRONG_CRC durumunu döndürür.

4.3.3- Status Register-A Okuma

```
157 #if 0   //read status registerA
158         AE_ltcStartStatusAdc(&ltcBat, MODE_7KHZ, CHST_ALL);
159         //!< check adcMeasure duration is completed
160         while(!AE_ltcAdcMeasureState());
161         status = AE_ltcReadStatusRegA(&ltcBat);
162 #endif
```

Önceki başlıklar için yapılan açıklamalar bu kısım için de geçerlidir. NOTE: status register-B için farklı bir istisna geçerlidir.

4.3.4- Status Register-B Okuma

```
#if 0   // when under and over limits are exceeded for cellx, x.th flag is raise
        underVoltage[0] = 3.0f;     // undervoltage value for slave 1
        overVoltage[0] = 4.1f;      // overvoltage value for slave 1
        underVoltage[1] = 3.0f;     // undervoltage value for slave 2
        overVoltage[1] = 3.2f;      // overvoltage value for slave 2

        AE_ltcSetUnderOverVoltage(ltcBat, underVoltage, overVoltage);

        status = AE_ltcUnderOverFlag(ltcBat);

        if(ltcBat[0].statusRegB.CellOverFlag.cell1)
        {
            //cell1 over the overVoltage limit
        }

        if(ltcBat[0].statusRegB.CellUnderFlag.cell1)
        {
            //cell1 under the underVoltage limit
        }

#endif
```

Eğer hücre gerilimleri, atanmış olan eşik değerlerinin üzerinde veya altında ise, status register-B içerisinde bulunan under voltage ve over voltage flag'leri kalkar. AE_ltcUnverOverFlag() fonksiyonu, flag durumlarını okur ve daha sonrasında bu flag'ler kullanılabilir hale gelir.

4.3.5- Hücre eşik değerlerini atama

```
#if 0   // when under and over limits are exceeded for cellx, x.th flag is raise
        underVoltage[0] = 3.0f;     // undervoltage value for slave 1
        overVoltage[0] = 4.1f;      // overvoltage value for slave 1
        underVoltage[1] = 3.0f;     // undervoltage value for slave 2
        overVoltage[1] = 3.2f;      // overvoltage value for slave 2

        AE_ltcSetUnderOverVoltage(ltcBat, underVoltage, overVoltage);

```

underVoltage dizisi her bir slave hücresinin alt eşik değerini tutar. Hücre bu değerin altında olursa underVoltageFlag register'ından kendisine ait flag'i kalkacaktır.

overVoltage dizisi her bir slave hücresinin üst eşik değerini tutar. Hücre bu değerin üzerinde olursa overVoltageFlag register'ından kendisine ait flag'i kalkacaktır.

4.3.6- Hücre içi sıcaklık okuma

```
#if 0   //internal die temperature
        AE_ltcStartStatusAdc(&ltcBat, MODE_7KHZ, CHST_ALL);
        while(!AE_ltcAdcMeasureState());
        AE_ltcReadStatusRegA(&ltcBat);

        if(ltcBat.statusRegA.internalDieTemp > 150)    //configuration register is reset
        {
            /*the thermal shutdown circuit trips and resets the Configuration
             * Register Groups (except the MUTE bit) and turns off all discharge switches.*/
        }
#endif
```

Status Reg-A içerisinde iç sıcaklık değerini tutan bir değişken vardır, sıcaklık 150 dereceui aşarsa IC kendisini sıfırlar.

4.3.6- Okunan Adc verilerini silme

```
199 #if 0   // close the adc, LTC6812-1 has 3 clear ADC commands: CLRCELL, CLRAUX and CLRSTAT
200         status = AE_ltcClearCellAdc(&ltcBat);
201         status = AE_ltcClearGpioAdc(&ltcBat);
202         status = AE_ltcClearStatusAdc(&ltcBat);
203 #endif
```

Bu fonksiyonlar adc ile okunup kaydedilen hücre, gpio status verilerini temizler ve 0xFF değerini atar.

4.3.7- GPIO3 pinine bağlı NTC okuma (DC2350 geliştirme kartı kullanıldı)

```
222 #if 0   // read GPIO3 temperature on development board
223         AE_ltcTemperature(ltcBat);
224 #endif
```

GPIO-3 pinine bağlı NTC sıcaklığını okur, Kart üzerinde R = 10K, NTC = 10K olarak ayarlamıştır. Sizde farklı bir ayar varsa ltc681x\temperature\temperature.h dosyasında yer alan macrolardan değiştirebilirsiniz.

4.3.7- Balance ayarları****

```
94  #if 0    // read the lowest cell voltage and balance the other cell up to this level
95      AE_ltcStartCellAdc(ltcBat, MODE_7KHZ, false, CELL_ALL);
96      //!< check adcMeasure duration is completed
97      while(!AE_ltcAdcMeasureState());
98      status = AE_ltcReadCellVoltage(ltcBat);
99
100     AE_ltcMinCellVolt(ltcBat);
101     underVoltage[0] = ltcBat[0].minCellVolt;
102     underVoltage[1] = ltcBat[1].minCellVolt;
103     overVoltage[0] = 4.2f;
104     overVoltage[1] = 4.2f;
105
106     AE_ltcPreBalance(ltcBat, DIS_5_MIN, underVoltage, overVoltage, DCC_ALL);
107     AE_ltcStartPwm(ltcBat, S_PIN_ALL, PWM_DUTY_LEVEL_10);
108 #endif
```

LTC6812 entegresiyle, iki farklı yöntemle dengeleme işlemi gerçekleştirilebilir. İlk yöntem, while(1) döngüsünün dışında yalnızca bir kez tanımlanır. Bu yöntemde, DIS_X_MIN parametresi ile belirlenen süre boyunca dengeleme işlemi gerçekleştirilir. Dengeleme işlemi, 30 saniyelik periyotlara bölünmüştür ve AE_ltcStartPwm() fonksiyonuna iletilen PWM_DUTY_LEVEL_X parametresi, bu sürenin içinde dengeleme ve bekleme sürelerini belirtir. Örneğin, PWM_DUTY_LEVEL_10 için yaklaşık olarak 20 saniye dengeleme yapılır ve ardından 10 saniye bekleme süresi uygulanır. DIS_X_MIN süresi boyunca herhangi bir okuma veya yazma işlemi, dengeleme işleminin sona ermesine neden olur. Bu dengeleme

yöntemi, sistem ilk kez dengeleme yapmak istendiğinde kullanılabilir, ancak diğer durumlar için uygun değildir.

```
#if 1   // balance in polling mode              @refgroup balance
    AE_ltcStartCellAdc(ltcBat, MODE_7KHZ, false, CELL_ALL);
    //!< check adcMeasure duration is completed
    while(!AE_ltcAdcMeasureState());
    status = AE_ltcReadCellVoltage(ltcBat);

   AE_ltcMinCellVolt(ltcBat);    //assign the minimum cell voltage to the ltcBat[x].minCellVoltage variable

   minCellVoltages[0] = ltcBat[0].minCellVolt;
   minCellVoltages[1] = ltcBat[1].minCellVolt;
#endif


    while(1)
    {
#if 1   //balance in polling    before this function call @refgroup balance section to take min cell voltages

        AE_ltcBalance(ltcBat, minCellVoltages, minBalanceVoltages);
#endif
```

İkinci yöntemde, her bir hücre tek tek okunur ve slave bloğu içindeki en düşük voltajlı hücre seçilir. Slave'lerin en düşük hücre voltaj değerleri ve minimum deşarj voltajı (2.8V) AE_ltcBalance() fonksiyonuna parametre olarak iletilir. AE_ltcBalance() fonksiyonu her döngüde overVoltage flag'lerini kontrol eder; voltajı min voltajdan daha yüksek olan hücrelerin flag değeri 1 olur. Flag değeri 1 olan hücrelerin deşarj pinleri açılır, tüm hücreler min voltaj değerine ulaşana kadar bu işlem devam eder.

(Not 1: Ledlerin 4.5 saniyede bir ripple yapmasının sebebi entegrenin watchdog timer'ı ile ilgilidir.

Not 2: Sönen bir ledin diğer döngüde başlamasının sebebi voltaj okumasındaki küçük sapmalardır. Örneğin, min voltaj değeri 3.5V ise ve bir hücrenin voltajı 3.505V olarak kabul edilsin. Bir döngüde bu değer 3.49V olarak okunurken, diğer döngüde 3.502 olarak okunabilir.)