Code Details

Part Number: Gate Driver(UCC587- Q1)

Issue: We are sending the 16-bit commands through the MOSI line of the SPI channel, but we are not receiving any data back from the gate driver.

Clock Frequency: 1MHz

The data is transmitted on the rising edge of the clock.

Chip Select 5 is used for this instance.

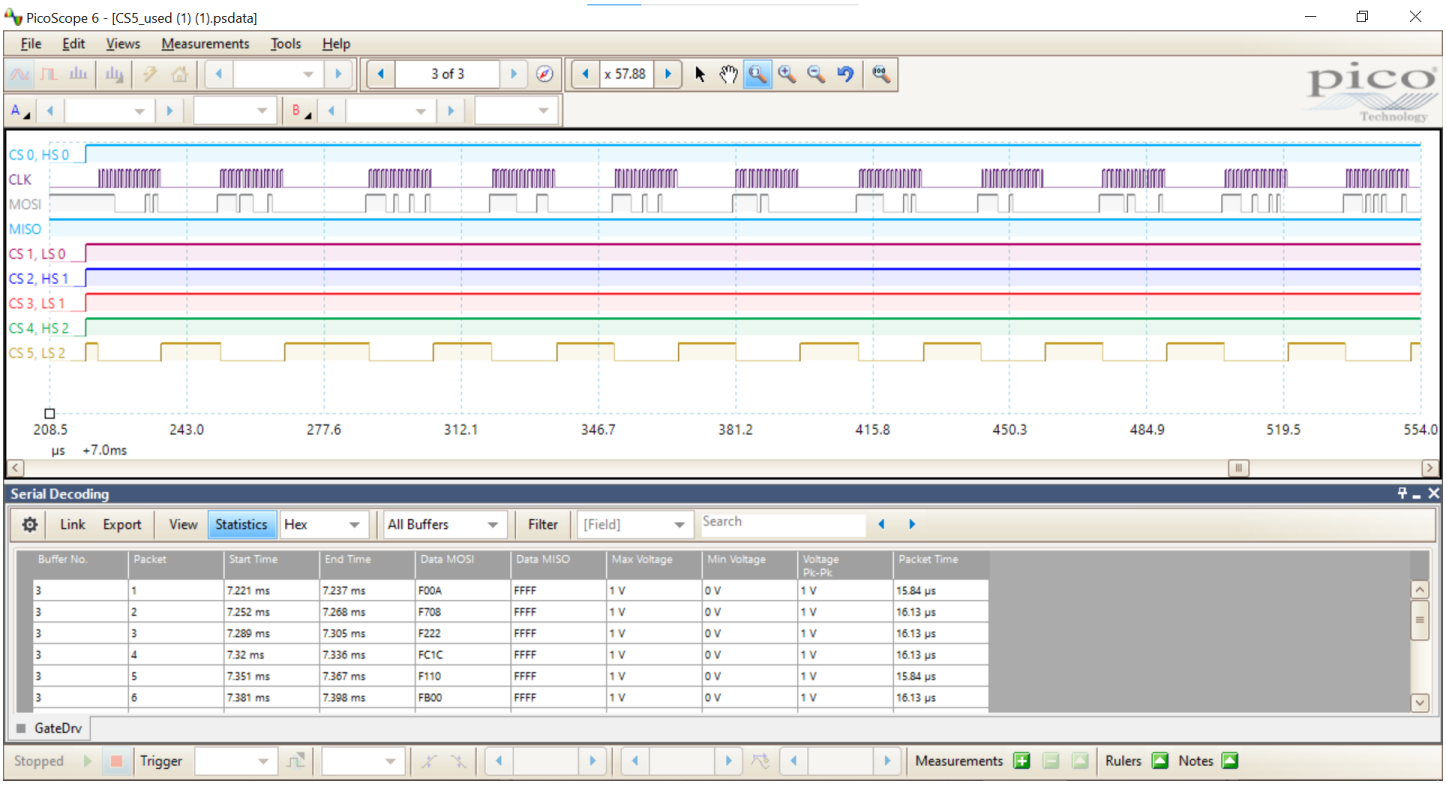

Figure 1

Code:

We are sending the following commands from MCU and trying to write and read 0x0002 to the gate driver's SPITEST register (Offset = 0x14).

```
All signals broadcasting
*/
UCC5870_DRVDIS (BROADCAST_SIGNAL); // F00A
UCC5870_SWRESET (BROADCAST_SIGNAL); //F708
UCC5870_CFGIN(BROADCAST_SIGNAL); /* to move from cfg1 to cfg2 */ // F222
UCC5870_WrReg(BROADCAST_SIGNAL,0x1C,0x1000);  //FC1C F110  FB00
UCC5870_SPITEST(BROADCAST_SIGNAL,0x0002); //
```

Figure 2

```
*/
void UCC5870_DRVDIS  (uint16 chipAddress)
{
/*  Assign the Chip Address to Bits 12-15
    and "0 0 0 0 0 0 0 0 1 0 1 0" to 11:0
*/
Std_ReturnType ReadStatus ;
    SpiWrite_Data_Buffer[1u] = (uint8)(0x00 + chipAddress);
    SpiWrite_Data_Buffer[0u] = (uint8)(0x0A) ;
    // SpiWrite_Data_Buffer[0]  = ((chipAddress << 12) | (0x0c00 + regAddress));

    Spi_SeqResultType setRg_Spi_stat = Spi_GetSequenceResult (SEQ_UCC5870_WRITE_DRV5);
    if(Spi_stat == SPI_SEQ_OK)
    {
        /* Prepare for SPI transfer */
        ReadStatus = Spi_SetupEB(CH_UCC5870_WRITE_DRV5 , &SpiWrite_Data_Buffer[0u] ,&SpiRead_Data_Buffer[0u],DataLength1);
        // ReadStatus = Spi_WriteIB (4, &SpiWrite_Data_Buffer[0u] );
        if ( ReadStatus == E_OK ){
            /* Send data over SPI */
            ReadStatus = Spi_SyncTransmit(SEQ_UCC5870_WRITE_DRV5);
/*      while (Spi_GetStatus() == SPI_BUSY)
        {
            Spi_MainFunction_Handling();
        } */
        }
        else {
            ReadStatus = E_NOT_OK ;
        }
    }
    else
    {
        ReadStatus = E_NOT_OK;
    }
}
```

Figure 3

```
*/
void UCC5870_SWRESET (uint16 chipAddress)
{
/*  Assign the Chip Address to Bits 12-15
    and "0 1 1 1 0 0 0 0 1 0 0 0" to 11:0
*/
    Std_ReturnType ReadStatus ;
    SpiWrite_Data_Buffer[1u] = (uint8)(0x07 + chipAddress);
    SpiWrite_Data_Buffer[0u] = (uint8)(0x08) ;
    Spi_SeqResultType setRg_Spi_stat = Spi_GetSequenceResult (SEQ_UCC5870_WRITE_DRV5);
    if(Spi_stat == SPI_SEQ_OK)
    {
        /* Prepare for SPI transfer */
        ReadStatus = Spi_SetupEB(CH_UCC5870_WRITE_DRV5 , &SpiWrite_Data_Buffer[0u] ,&SpiRead_Data_Buffer[0u],DataLength1);
        // ReadStatus = Spi_WriteIB (4, &SpiWrite_Data_Buffer[0u] );
        if ( ReadStatus == E_OK ){
            /* Send data over SPI */
            ReadStatus = Spi_SyncTransmit(SEQ_UCC5870_WRITE_DRV5);
        while (Spi_GetStatus() == SPI_BUSY)
        {
            Spi_MainFunction_Handling();
        }
        }
        else {
            ReadStatus = E_NOT_OK ;
        }
    }
    else
    {
        ReadStatus = E_NOT_OK;
    }
}
```

Figure 4

```
  void UCC5870_CFGIN (uint16 chipAddress)
{
/*  Assign the Chip Address to Bits 12 -15
 *  and " 0 0 1 0 0 0 1 0 0 0 1 0 " to 11:0
 *
 *
 */
    Std_ReturnType ReadStatus ;
    SpiWrite_Data_Buffer[1u] = (uint8)(0x02 + chipAddress);
    SpiWrite_Data_Buffer[0u] = (uint8)(0x22) ;

    Spi_SeqResultType Spi_stat = Spi_GetSequenceResult (SEQ_UCC5870_WRITE_DRV5);
    if(Spi_stat == SPI_SEQ_OK)
    {
        /* Prepare for SPI transfer */
        ReadStatus = Spi_SetupEB(CH_UCC5870_WRITE_DRV5 , &SpiWrite_Data_Buffer[0u] ,&SpiRead_Data_Buffer[0u],DataLength1);
        // ReadStatus = Spi_WriteIB (4, &SpiWrite_Data_Buffer[1u] );
        if ( ReadStatus == E_OK ){
            /* Send data over SPI */
            ReadStatus = Spi_SyncTransmit(SEQ_UCC5870_WRITE_DRV5);
        }
        else {
            ReadStatus = E_NOT_OK ;
        }
    }
    else
    {
        ReadStatus = E_NOT_OK;
    }
}
```

Figure 5

```
void UCC5870_WrReg(uint16 chipAddress, uint16 regAddress, uint16  data)
{
/*
    Send the chip address to select the GD
    Write/Overwrite Use the WRH command to write the "high" byte of the register (bits 15:8)
    and the WRL command to write the "low" byte of the register (bits 7:0).
*/

    UCC5870_setReg(chipAddress, regAddress);
    UCC5870_WRDHcmd (chipAddress, data / 256 );
    UCC5870_WRDLcmd (chipAddress, data % 256 );

}
```

Figure 6

```
void UCC5870_setReg (uint16 chipAddress, uint16 regAddress){

 /*
  * add chip address to 12-15 bits
  * and '110000' for bits 11-5
  * and regAddress from 0-4 bits
  */
  Std_ReturnType ReadStatus ;
    SpiWrite_Data_Buffer[1u] = (uint8)(0x0C + chipAddress);
    SpiWrite_Data_Buffer[0u] = (uint8)(0x00 + regAddress) ;
 // SpiWrite_Data_Buffer[0]  = ((chipAddress << 12) | (0x0c00 + regAddress));

  Spi_SeqResultType setRg_Spi_stat = Spi_GetSequenceResult (SEQ_UCC5870_WRITE_DRV5);
  if(Spi_stat == SPI_SEQ_OK)
    {
        /* Prepare for SPI transfer */
        ReadStatus = Spi_SetupEB(CH_UCC5870_WRITE_DRV5 , &SpiWrite_Data_Buffer[0u] ,&SpiRead_Data_Buffer[0u],DataLength1);
        // ReadStatus = Spi_WriteIB (4, &SpiWrite_Data_Buffer[0u] );
        if ( ReadStatus == E_OK ){
            /* Send data over SPI */
            ReadStatus = Spi_SyncTransmit(SEQ_UCC5870_WRITE_DRV5);
/*      while (Spi_GetStatus() == SPI_BUSY)
        {
            Spi_MainFunction_Handling();
        } */
        }
        else {
            ReadStatus = E_NOT_OK ;
        }
    }
    else
    {
        ReadStatus = E_NOT_OK;
    }
}
```

Figure 7

```
void UCC5870_WRDLcmd (uint16 chipAddress, uint16 data){

 /*
  * add chip address to 12-15 bits
  * and '1011' for bits 8-11
  * and data from 0-7 bits
  */
  Std_ReturnType ReadStatus ;
    SpiWrite_Data_Buffer[1u] = (uint8)(0x0B + chipAddress);
    SpiWrite_Data_Buffer[0u] = (uint8)(0x00 + data) ;
    // SpiWrite_Data_Buffer[4]=((chipAddress << 12) | (0x0b00 + (data % 256 )));

  Spi_SeqResultType Spi_stat_WDHL = Spi_GetSequenceResult (SEQ_UCC5870_WRITE_DRV5);
  if(Spi_stat == SPI_SEQ_OK)
    {
        /* Prepare for SPI transfer */
        ReadStatus = Spi_SetupEB(CH_UCC5870_WRITE_DRV5 , &SpiWrite_Data_Buffer[0u] ,&SpiRead_Data_Buffer[0u],DataLength1);
        // ReadStatus = Spi_WriteIB (4, &SpiWrite_Data_Buffer[2u] );
        if ( ReadStatus == E_OK ){
            /* Send data over SPI */
            ReadStatus = Spi_SyncTransmit(SEQ_UCC5870_WRITE_DRV5);
    /*  while (Spi_GetStatus() == SPI_BUSY)
        {
            Spi_MainFunction_Handling();
        } */
        }
        else {
            ReadStatus = E_NOT_OK ;
        }
    }
    else
    {
        ReadStatus = E_NOT_OK;
    }

}
```

Figure 8

```
121  void UCC5870_WRDHcmd (uint16 chipAddress, uint16 data){
122
123   /*
124    * add chip address to 12-15 bits
125    * and '1010' for bits 8-11
126    * and data from 0-7 bits
127    */
128    Std_ReturnType ReadStatus ;
129      SpiWrite_Data_Buffer[1u] = (uint8)(0x01 +chipAddress);
130      SpiWrite_Data_Buffer[0u] = (uint8)(0x00 + data) ;
131    // SpiWrite_Data_Buffer[2u]  = ((chipAddress << 12) | (0x0a00 + ( data / 256  )));
132
133    Spi_SeqResultType Spi_stat = Spi_GetSequenceResult (SEQ_UCC5870_WRITE_DRV5);
134    if(Spi_stat == SPI_SEQ_OK)
135      {
136          /* Prepare for SPI transfer */
137          ReadStatus = Spi_SetupEB(CH_UCC5870_WRITE_DRV5 , &SpiWrite_Data_Buffer[0u] ,&SpiRead_Data_Buffer[0u],DataLength1);
138          // ReadStatus = Spi_WriteIB (4, &SpiWrite_Data_Buffer[1u] );
139          if ( ReadStatus == E_OK ){
140              /* Send data over SPI */
141              ReadStatus = Spi_SyncTransmit(SEQ_UCC5870_WRITE_DRV5);
142          }
143          else {
144              ReadStatus = E_NOT_OK ;
145          }
146      }
147      else
148      {
149          ReadStatus = E_NOT_OK;
150      }
151  }
```

Figure 9

```
Std_ReturnType UCC5870_RdReg(uint16 chipAddress, uint16 regAddress)
{


    /*
    * RD_Reg_data = ((CA << 12) | (0x0100 + RA));
    * UCC5870_NOP(chipAddress)
    * Select the GD and its Register to
    * read what is received back (SPI_RXBUF)
    *
*/

    SpiWrite_Data_Buffer[1u] = (uint8)(0x01 + chipAddress);
    SpiWrite_Data_Buffer[0u] = (uint8)(0x00 + regAddress) ;
    /* SpiWrite_Data_Buffer[2u] = (uint8)(0xFF);
    SpiWrite_Data_Buffer[3u] = (uint8)(0xFF); */
    Std_ReturnType ReadStatus= E_NOT_OK;
    Spi_SeqResultType Spi_stat_RdReg = Spi_GetSequenceResult (SEQ_UCC5870_READ_DRV5);
    if(Spi_stat == SPI_SEQ_OK)
    {
        /* Prepare for SPI transfer */
        ReadStatus |= Spi_SetupEB(CH_UCC5870_READ_DRV5, &SpiWrite_Data_Buffer[0u] ,&SpiRead_Data_Buffer[0u],DataLength1);
        // ReadStatus = Spi_WriteIB (4, &SpiWrite_Data_Buffer[0u] );
        ReadStatus |= Spi_SyncTransmit(SEQ_UCC5870_READ_DRV5);
        /* Wait till the transmission is completed */
/*      while (Spi_GetStatus() == SPI_BUSY)
        {
            Spi_MainFunction_Handling();
        } */
//  UCC5870_NOP(chipAddress);
    }
    else
```

Figure 10

```
117  void UCC5870_SPITEST( uint16 chipAddress , uint16 Data){
118
119  /* Spi_Init(Spi_Config);
120   Std_ReturnType Init_chk = Spi_InitCheck(Spi_Config); */
121   //uint16 TEST_Reg_Addr =  ;
122
123   UCC5870_WrReg(chipAddress, 0x14, Data);
124   UCC5870_RdReg(chipAddress, 0x14) ; /* For reading the status reg for cfg mode */
125   UCC5870_NOP(BROADCAST_SIGNAL);
126   Spi_stat = Spi_GetSequenceResult( SEQ_UCC5870_READ_DRV5 ) ;
127
128  }
```

Figure 11

```
#define SEQ_UCC5870_WRITE_DRV5          (14)
#define SEQ_UCC5870_READ_DRV5           (17)
#define CH_UCC5870_READ_DRV5            (15)
#define CH_UCC5870_WRITE_DRV5           (9)
```

Figure 12

```
#define BROADCAST_SIGNAL                ((uint8)0xF0)
#define CA0                             (0x10)
#define DataLength1                     (1)
```

Figure 13