

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.

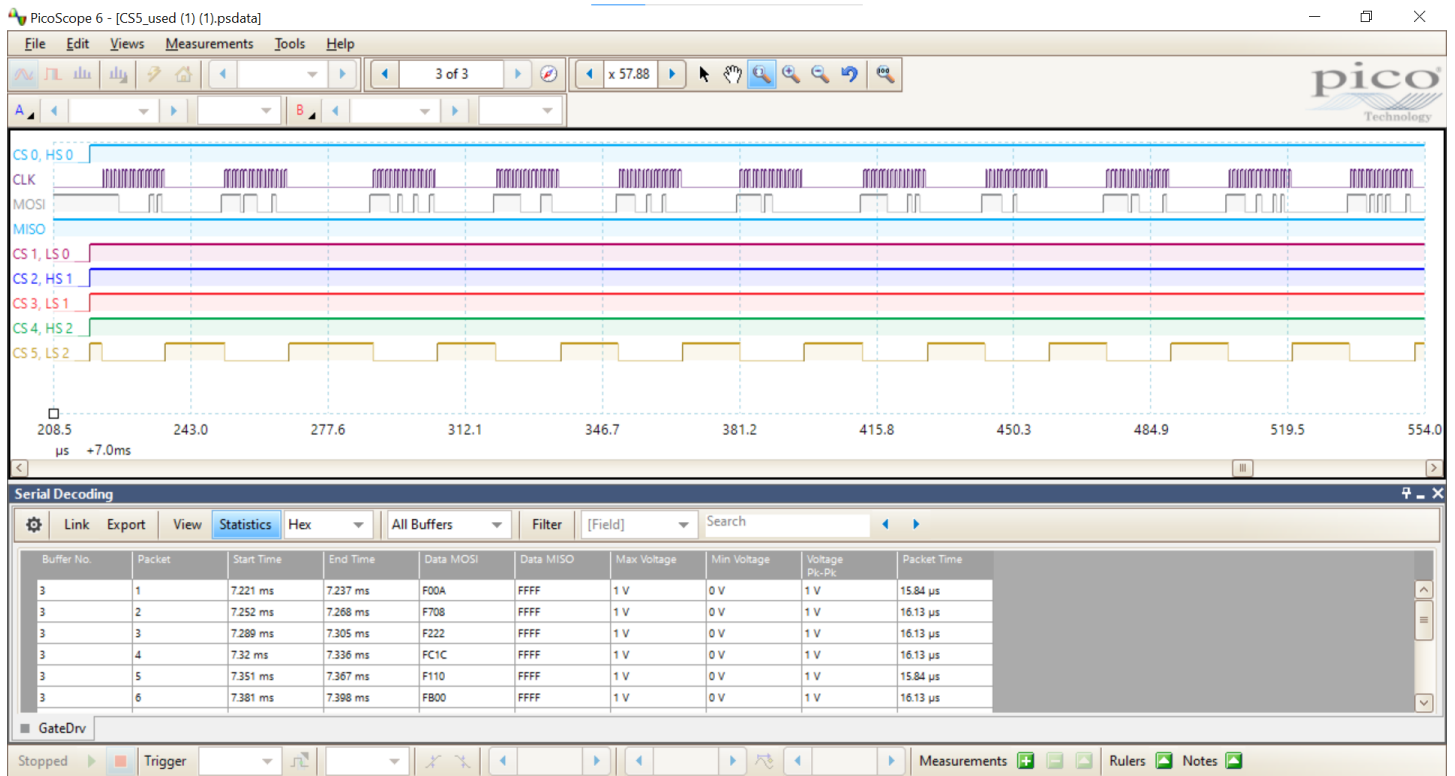


Figure 1

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

```

348 void UCC5870_DRVDIS (uint16 chipAddress)
349 {
350     /* Assign the Chip Address to Bits 12-15
351     and "0 0 0 0 0 0 0 0 1 0 1 0" to 11:0
352     */
353     Std_ReturnType ReadStatus ;
354     SpiWrite_Data_Buffer[1u] = (uint8) (0x00 + chipAddress);
355     SpiWrite_Data_Buffer[0u] = (uint8) (0x0A) ;
356     // SpiWrite_Data_Buffer[0] = ((chipAddress << 12) | (0x0c00 + regAddress));
357
358     Spi_SeqResultType setRg_Spi_stat = Spi_GetSequenceResult (SEQ_UCC5870_WRITE_DRV5);
359     if(Spi_stat == SPI_SEQ_OK)
360     {
361         /* Prepare for SPI transfer */
362         ReadStatus = Spi_SetupEB(CH_UCC5870_WRITE_DRV5 , &SpiWrite_Data_Buffer[0u] ,&SpiRead_Data_Buffer[0u],DataLength1);
363         // ReadStatus = Spi_WriteIB (4, &SpiWrite_Data_Buffer[0u] );
364         if ( ReadStatus == E_OK ) {
365             /* Send data over SPI */
366             ReadStatus = Spi_SyncTransmit(SEQ_UCC5870_WRITE_DRV5);
367             while (Spi_GetStatus() == SPI_BUSY)
368             {
369                 Spi_MainFunction_Handling();
370             }
371         }
372         else {
373             ReadStatus = E_NOT_OK ;
374         }
375     }
376     else
377     {
378         ReadStatus = E_NOT_OK;
379     }
380 }

```

Figure 3

```

6 void UCC5870_SWRESET (uint16 chipAddress)
7 {
8     /* Assign the Chip Address to Bits 12-15
9     and "0 1 1 0 0 0 0 1 0 0 0" to 11:0
10     */
11     Std_ReturnType ReadStatus ;
12     SpiWrite_Data_Buffer[1u] = (uint8) (0x07 + chipAddress);
13     SpiWrite_Data_Buffer[0u] = (uint8) (0x08) ;
14     Spi_SeqResultType setRg_Spi_stat = Spi_GetSequenceResult (SEQ_UCC5870_WRITE_DRV5);
15     if(Spi_stat == SPI_SEQ_OK)
16     {
17         /* Prepare for SPI transfer */
18         ReadStatus = Spi_SetupEB(CH_UCC5870_WRITE_DRV5 , &SpiWrite_Data_Buffer[0u] ,&SpiRead_Data_Buffer[0u],DataLength1);
19         // ReadStatus = Spi_WriteIB (4, &SpiWrite_Data_Buffer[0u] );
20         if ( ReadStatus == E_OK ) {
21             /* Send data over SPI */
22             ReadStatus = Spi_SyncTransmit(SEQ_UCC5870_WRITE_DRV5);
23             while (Spi_GetStatus() == SPI_BUSY)
24             {
25                 Spi_MainFunction_Handling();
26             }
27         }
28         else {
29             ReadStatus = E_NOT_OK ;
30         }
31     }
32     else
33     {
34         ReadStatus = E_NOT_OK;
35     }
36 }

```

Figure 4

```

348 void UCC5870_CFGIN (uint16 chipAddress)
349 {
350     /* Assign the Chip Address to Bits 12 -15
351     and " 0 0 1 0 0 0 1 0 0 0 1 0" to 11:0
352     */
353     Std_ReturnType ReadStatus ;
354     SpiWrite_Data_Buffer[1u] = (uint8) (0x02 + chipAddress);
355     SpiWrite_Data_Buffer[0u] = (uint8) (0x22) ;
356
357     Spi_SeqResultType Spi_stat = Spi_GetSequenceResult (SEQ_UCC5870_WRITE_DRV5);
358     if(Spi_stat == SPI_SEQ_OK)
359     {
360         /* Prepare for SPI transfer */
361         ReadStatus = Spi_SetupEB(CH_UCC5870_WRITE_DRV5 , &SpiWrite_Data_Buffer[0u] ,&SpiRead_Data_Buffer[0u],DataLength1);
362         // ReadStatus = Spi_WriteIB (4, &SpiWrite_Data_Buffer[1u] );
363         if ( ReadStatus == E_OK ) {
364             /* Send data over SPI */
365             ReadStatus = Spi_SyncTransmit(SEQ_UCC5870_WRITE_DRV5);
366             while (Spi_GetStatus() == SPI_BUSY)
367             {
368                 Spi_MainFunction_Handling();
369             }
370         }
371         else {
372             ReadStatus = E_NOT_OK ;
373         }
374     }
375     else
376     {
377         ReadStatus = E_NOT_OK;
378     }
379 }

```

Figure 5

```

159 //
160 void UCC5870_WrReg(uint16 chipAddress, uint16 regAddress, uint16 data)
161 {
162     /*
163     /*
164     /* Send the chip address to select the GD
165     /* Write/Overwrite Use the WRH command to write the "high" byte of the register (bits 15:8)
166     /* and the WRL command to write the "low" byte of the register (bits 7:0).
167     */
168     /*
169     UCC5870_setReg(chipAddress, regAddress);
170     UCC5870_WRDHcmd (chipAddress, data / 256 );
171     UCC5870_WRLcmd (chipAddress, data % 256 );
172     */
173 }
174

```

Figure 6

```

76 void UCC5870_setReg (uint16 chipAddress, uint16 regAddress){
77
78     /*
79     /* add chip address to 12-15 bits
80     /* and '110000' for bits 11-5
81     /* and regAddress from 0-4 bits
82     */
83     Std_ReturnType ReadStatus ;
84     SpiWrite_Data_Buffer[1u] = (uint8)(0x0C + chipAddress);
85     SpiWrite_Data_Buffer[0u] = (uint8)(0x00 + regAddress) ;
86     // SpiWrite_Data_Buffer[0] = ((chipAddress << 12) | (0x0c00 + regAddress));
87
88     Spi_SeqResultType setRg_Spi_stat = Spi_GetSequenceResult (SEQ_UCC5870_WRITE_DRV5);
89     if(Spi_stat == SPI_SEQ_OK)
90     {
91         /* Prepare for SPI transfer */
92         ReadStatus = Spi_SetupEB(CH_UCC5870_WRITE_DRV5 , &SpiWrite_Data_Buffer[0u] ,&SpiRead_Data_Buffer[0u],DataLength1);
93         // ReadStatus = Spi_WriteIB (4, &SpiWrite_Data_Buffer[0u] );
94         if ( ReadStatus == E_OK ){
95             /* Send data over SPI */
96             ReadStatus = Spi_SyncTransmit(SEQ_UCC5870_WRITE_DRV5);
97             /* while (Spi_GetStatus() == SPI_BUSY)
98             {
99                 Spi_MainFunction_Handling();
100             } */
101         }
102         else {
103             ReadStatus = E_NOT_OK ;
104         }
105     }
106     else
107     {
108         ReadStatus = E_NOT_OK;
109     }
110 }

```

Figure 7

```

162 void UCC5870_WRLcmd (uint16 chipAddress, uint16 data){
163
164     /*
165     /* add chip address to 12-15 bits
166     /* and '1011' for bits 8-11
167     /* and data from 0-7 bits
168     */
169     Std_ReturnType ReadStatus ;
170     SpiWrite_Data_Buffer[1u] = (uint8)(0x0B + chipAddress);
171     SpiWrite_Data_Buffer[0u] = (uint8)(0x00 + data) ;
172     // SpiWrite_Data_Buffer[4]=((chipAddress << 12) | (0x0b00 + (data % 256 )));
173
174     Spi_SeqResultType Spi_stat_WDHL = Spi_GetSequenceResult (SEQ_UCC5870_WRITE_DRV5);
175     if(Spi_stat == SPI_SEQ_OK)
176     {
177         /* Prepare for SPI transfer */
178         ReadStatus = Spi_SetupEB(CH_UCC5870_WRITE_DRV5 , &SpiWrite_Data_Buffer[0u] ,&SpiRead_Data_Buffer[0u],DataLength1);
179         // ReadStatus = Spi_WriteIB (4, &SpiWrite_Data_Buffer[2u] );
180         if ( ReadStatus == E_OK ){
181             /* Send data over SPI */
182             ReadStatus = Spi_SyncTransmit(SEQ_UCC5870_WRITE_DRV5);
183             /* while (Spi_GetStatus() == SPI_BUSY)
184             {
185                 Spi_MainFunction_Handling();
186             } */
187         }
188         else {
189             ReadStatus = E_NOT_OK ;
190         }
191     }
192     else
193     {
194         ReadStatus = E_NOT_OK;
195     }
196 }
197

```

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