.

Thread Tracking Toolkit

Show/Hide Account Look-Up Tool

Thread ID: 1350594

Account:

Region:

Company: Texas Instruments

Thread Status (Internal only)

Waiting for Customer

Priority

Normal

Assign

j-gundavarapu@ti.cc

Email addresses entered above will receive a one-time email notifying them of assignment to this thread **and** will be automatically email subscribed to all subsequent replies.

Notify

Email addresses/lists entered above will receive a one-time email notifying them of this thread.

Responsible Organization

--[ASM] ASM-IND

Notes:

[ZF 04-17-24] Assigning to Jagadish G {TOPIC} RM46/SPI

Submit Click "Submit" button to save any changes above.

RM46L852: Arm-based microcontrollers - INTERNAL forum

Alex Paul
139.187.34.35

Expert 3581 points
Texas Instruments

Part Number: RM46L852
Other Parts Discussed in Thread: TMS570LC4357, HALCOGEN

Hi experts,

My customer ABB motion just found the following issue in our HErcules Part:

We have a problem to send data from Hercules SPI slave to the master. Data from master works correctly and can be received without any issues (only a bit weird thing is that data is received correctly despite of POLARITY or PHASE setting, all 4 of them works in Hercules receive…).

The problem is that MSB bits of the byte(s) are consistently lost after the initial start byte.
If Hercules sends 0xFF 0xFF master sees FF 7F

If Hercules sends 0xFF 0x80 master sees FF 00

We have reduced the setup to simple test code, 2pcs of 0xFF is put to the SPI DAT0 buffer waiting for CS activation and then we scobe what happens in the bus

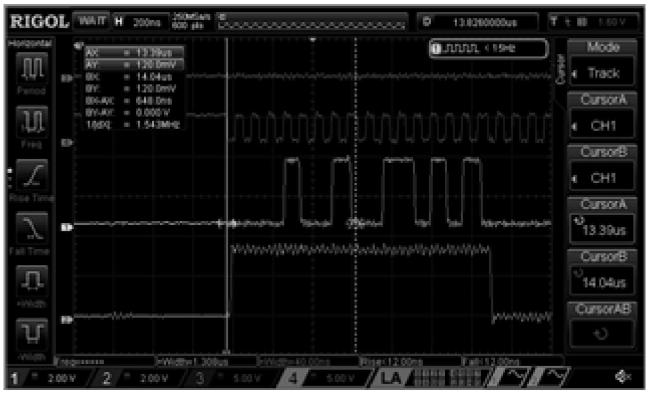
```
    // clear all buffers (SPIEN), also send PEND bit for TX in enable

     vSpiDisable( SPI_COMM_DEV );

     vSpiEnable( SPI_COMM_DEV );

#if defined(SSOM)

        CU_SPI_CS_LOW; // CS as LOW

#endif

        SPI_COMM_DEV->DAT0 = 0xFF;

        SPI_COMM_DEV->DAT0 = 0xFF; // at this point the FLG-register TXINTFLG goes to 0 == shift register and next byte waiting for sending



        dmaSetChEnable( (uint32)DMA_CH_COMM_RX, (uint32)DMA_HW ); // Enable DMA channel, RX first

        //dmaSetChEnable( (uint32)DMA_CH_COMM_TX, (uint32)DMA_HW ); // Enable DMA channel



#if defined(AFSO_42)

        (void)HAL_tDevWriteIntIo(SPI_ENA, IO_LOW); // allow master sending

#endif
```

TEAL line = Hercules SOMI (data to master)
Yellow = Hercules SIMO (data from master, not interesting but it’s data output/sampling logic can be seen)
Purple = CS
Blue = CLK

So master runs data so that data is updated on rising edge and latched in falling edge. By standard SPI definition this is mode 2 (CPOL=1, CPHA=0) since CLK idles in 3v3 state.

So we need/should to use Hercules POLARITY=1 setting since clock idles at 3v3. And just by looking standard SPI definition the POLARITY=1, PHASE=0 should be correct one (but read forward).

- With Hercules POLARITY=1 and PHASE =0 setting situation looks like this. Hercules does not arm the data to be ready for CLK when CS activates. So master sees 1st bit as 0 when it samples it at falling edge.

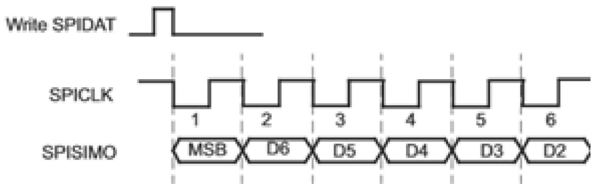


It technically matches to this one, and we can even see that outputting and latching wont match what master expects (master outputs in rising and this mode in falling)

Manual also nicely illustrates that after DAT write it requires CLK to get data into the line

With POLARITY=0, PHASE=0 setting se output of the data is even more delayed

Figure 27-10. Clock Mode with Polarity = 1 and Phase = 0

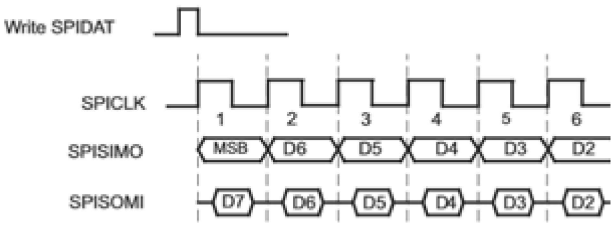


It yet again works like said, but the problem is again that data is not armed at CS, only when CLK edges are seen. Now the output&latch sequence is same

It is also worth to note that data goes to 0 after 16CLK. So even the 1st CLK is missed the data ends correctly.

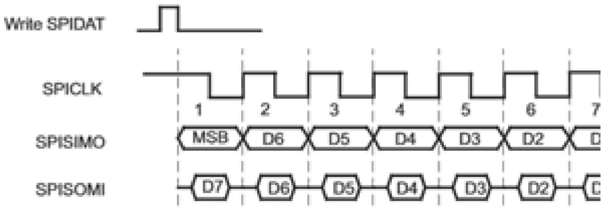
similarly here

Figure 27-8. Clock Mode with Polarity = 0 and Phase = 0

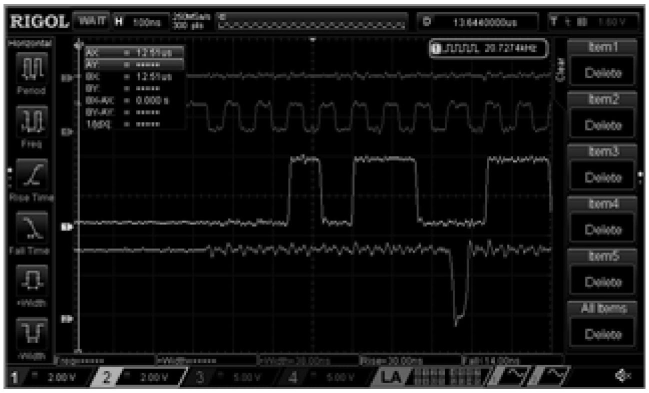


So this one would do what we want, data is armed on DAT write and outputted on rising & latched on falling just as master does:

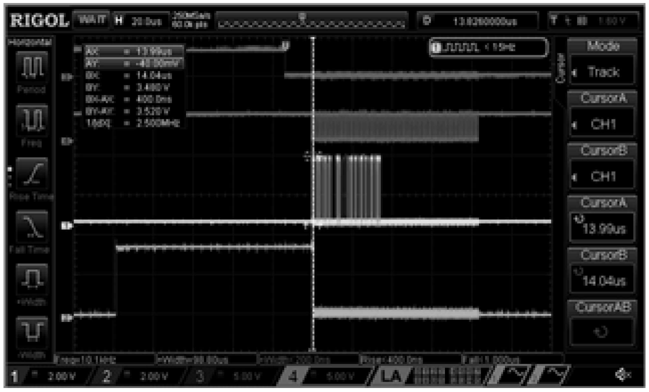
Figure 27-11. Clock Mode with Polarity = 1 and Phase = 1



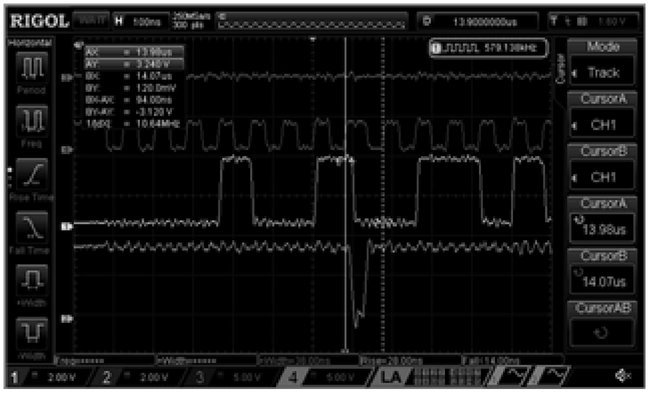
With POLARITY=1, PHASE=1 it looks like this. Data is “armed” to the line before the clock starts running but then something weird happens in byte boundary.



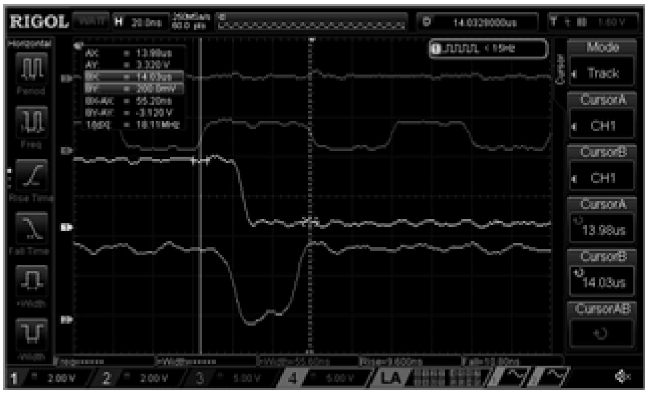
It is also worth to note that in PHASE=1 mode this does not honor/listen CS at all, data looks to be outputted whenever DAT is written despite of CS state (how this would work in multi slave chain???).



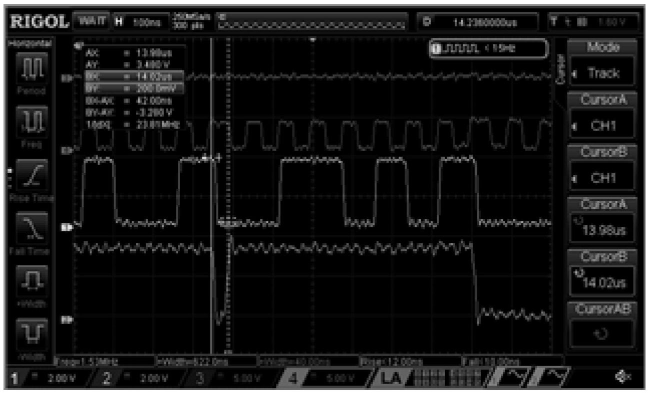
Closeup of the byte boundary. Master sent data 09 35 in this case. 1st cursor marks the data update point for the MSB of the 2nd byte. Hercules starts to output 0 (why, if I had written 0xFF 0xFF) and then surprisingly near falling edge starts correct the line into proper state



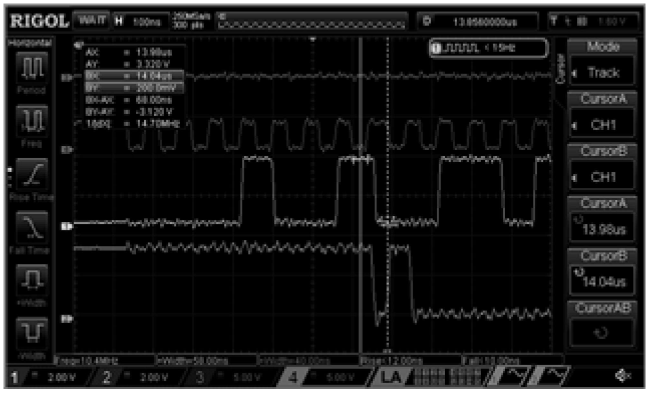
Basically it looks the that “correction” starts even before the falling edge since on edge it is already in 1 state



We can also the data the data ends correctly and Hercules starts to output 0 exactly as 3rd byte MSB bit point in rising edge.



It does not matter do we write 0x80 or what, the behavior is same if then MSB bit is 1. Here is FF 80 write.



So how to say, everything would work OK with POL=1, PHASE=1 setting if we do not send a value over 127dec. All MSB bits after the 1st byte will get eaten away by that weird state change in between bytes.

We have used this same protocol before successfully so that Hercules has been master. But now based on above screenshots is it obvious that with any of settings this cannot work when Hercules is slave. We are normally using the DMA and I started even to wonder that cannot DMA move data to the DAT0 so that in between bytes it would start output 0 and then data comes to BUF and shift register, but this cannot be the problem for 2 reason:

- There is data always in shift register so DMA has 1 byte time to move next byte into it
- since we can manually prefill 2 bytes and we got those bytes ouput it is obvious that those 2 bytes has been there since CLK starts and still the odd in between bytes 0 value appears.

There cannot be this kind of a problem in this SPI peripheral so only option is that some settings are not correct? But I cannot find anything meaningful which could affect to this one, also tried to read data sheet SLAVE timings (table 7-36 and 7-35) but no luck in that sector either.

the only ? part was this one. How ever I tried like 7 different PS value in between 0..255 and observed behavior stays exactly same so looks like in principle this PS setting does not have any effect to outside functionality.
(5) When the SPI is in Slave mode, the following must be true:

For PS values from 1 to 255: tc(SPC)S ≥ (PS +1)tc(VCLK) ≥ 40ns, where PS is the prescale value set in the SPIFMTx.[15:8] register bits.

For PS values of 0: tc(SPC)S = 2tc(VCLK) ≥ 40ns.

For the slave the tc is min 40ns this 25MHz master clock.

VCLK of the SPI module is 110MHz so ~9ns.

2*9ns >= 40 // not true so cannot use PS = 0 value

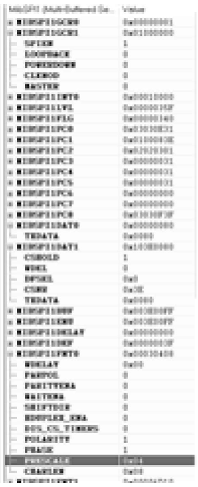
(4+1)*9ns >= 40
master CLK is ~80ns => 12,5MHz so PS should be
4*9=36 // not ok
5*9=45 // ok

6*9=54 // ok

7*9=63 // ok

8*9=72 // ok
9*9=81 // not ok
meaning PS values in between 5-1 and 9-1 => 4 to 8 should be acceptable
This also means that maximum supported CLK is not 40ns/25MHz because of PS. 45ns = 22,2MHz with this VCLK (110Mhz) setting

Here should be all key settings (DMAREQ_EN is set in INT0)



```
MibSPI1 (Multi-Buffered Se...   Value
⊞ MIBSPI1GCR0                   0x00000001
⊟ MIBSPI1GCR1                   0x01000000
   SPIEN                        1
   LOOPBACK                     0
   POWERDOWN                    0
   CLKMOD                       0
   MASTER                       0
⊞ MIBSPI1INT0                   0x00010000
⊞ MIBSPI1LVL                    0x0000035F
⊞ MIBSPI1FLG                    0x00000140
⊞ MIBSPI1PC0                    0x01010E31
⊞ MIBSPI1PC1                    0x0100003E
⊞ MIBSPI1PC2                    0x01010001
⊞ MIBSPI1PC3                    0x00000031
⊞ MIBSPI1PC4                    0x00000031
⊞ MIBSPI1PC5                    0x00000031
⊞ MIBSPI1PC6                    0x00000000
⊞ MIBSPI1PC7                    0x00000000
⊞ MIBSPI1PC8                    0x01010F3F
⊟ MIBSPI1DAT0                   0x00000000
   TXDATA                       0x0000
⊟ MIBSPI1DAT1                   0x103E0000
   CSHOLD                       1
   WDEL                         0
   DFSEL                        0x0
   CSNR                         0x3E
   TXDATA                       0x0000
⊞ MIBSPI1BUF                    0x003E00FF
⊞ MIBSPI1EMU                    0x003E00FF
⊞ MIBSPI1DELAY                  0x00000000
⊞ MIBSPI1DEF                    0x0000003F
⊟ MIBSPI1FMT0                   0x00030408
   WDELAY                       0x00
   PARPOL                       0
   PARITYENA                    0
   WAITENA                      0
   SHIFTDIR                     0
   HDUPLEX_ENA                  0
   DIS_CS_TIMERS                0
   POLARITY                     1
   PHASE                        1
   PRESCALE                     0x04
   CHARLEN                      0x08
⊞ MIBSPI1FMT1                   [illegible]
```

DAT1: CSNR does not have any effect. Similar in between 0 is seen with any value. Also with PHASE=1 the data is outputted similarly way before CS despite is the CSNR 0, 1, 0x3E. We have muxedfor the SPI use only CS_0 all others are GIO. With that 0x3E value in master mode the CS_0 (SPI_CS_0 = 0xFEU) was controlled propely. It is weird that in spi.h the value is 0xFE when in TRM table 27-24 is says that CS_0 CSNR value is 1. So look like somewhere something is bit inverted…

For me it also looks that Hercules POLARITY and PHASE settings does not match to standard SPI definition, looks like POL=1, PHA=1 is mod2 and not mod3 as one would figure from binary combination of the digits.

I'll want to that, that the behavior is exactly same with TX DMA enabled we send

FF FE FD FC FB …42. 84…. FB FC FD FE FF

and master sees

7F 7E 7D 7C 7B …02. 04…. 7B 7C 7D 7E 7F
So there is nothing wrong in the protocol logic/DMA usage, it is just that observed bus state with various POL/PHASE settings which causes the "data corruption" it cannot work with these settings.

In overall following observation we made, please advice especially on item 1

1. MSB bits are lost
2. POL/PHASE does not effect to receiving at all
3. DAT1:CSNR definitions in spi.h and TRM are conflicting
4. FMTn: PRESCALER looks to have no effect what ever what it is

Would nice if you could help here.

Regards,

Alex

17 days ago

jagadish gundavarapu 2409:40f0:2f:82ce:70a5:5f12:7117:b482 15 days ago TI__Mastermind 40145 points

Hi Alex,

Can you please ask customer to verify below thread:

(+) TMS570LS3137: MibSPI Slave -> Master transfer 1st value missing first bit - Arm-based microcontrollers forum - Arm-based microcontrollers - TI E2E support forums

Ask them to try the workaround i gave in above thread once, i felt it is very similar issue.

—

Thanks & Regards,

Jagadish.

Alex Paul 139.187.34.35 11 days ago in reply to jagadish gundavarapu TI__Expert 3581 points

Hi Jagadish,

This does not solve the issue at all.

My customer assumes that there is something going wrong with the HW or HW-drivers.

He put some effort in there to debug the issue.

I forwarded you the E-Mail chain.

Regards,

Alex

jagadish gundavarapu 139.187.34.35 *3 days ago in reply to Alex Paul* TI__Mastermind 40145 points

Hi Alex Paul,

I think customer tested with two RM46 boards one as SPI master and other one as SPI slave.

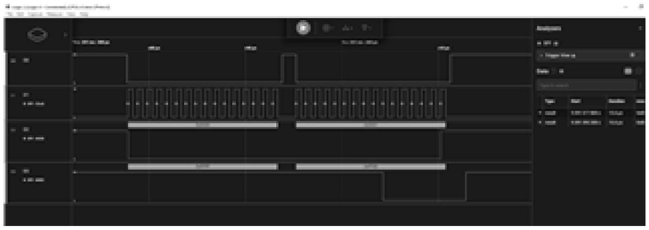
As i don't have two RM46 boards, i tested with one TMS570LC4357 as master and one RM46 as slave.

Here i didn't get any issues in communication.

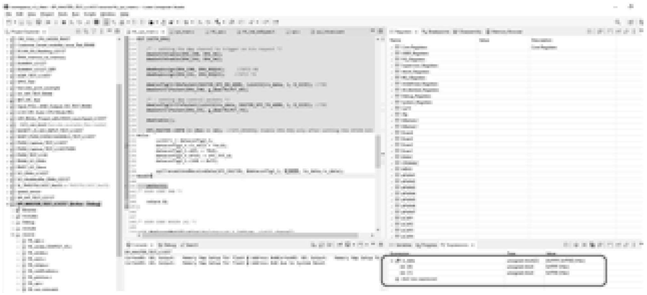
Here i am sending two characters of 0x0000 and 0x0001 from master to slave, similarly i am sending two characters of 0xFFFF and 0xFF80 from the slave.

1. Output when CPOL=0 and CPHA=0:

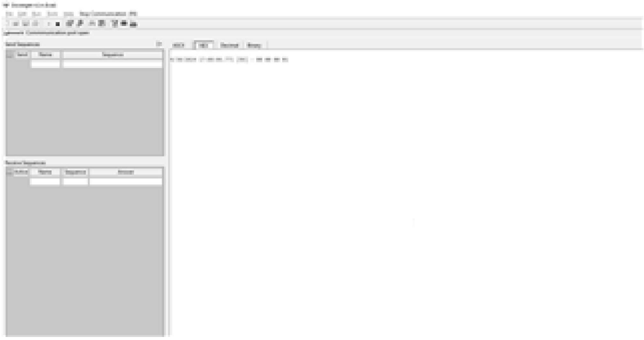
Waveforms between two:



Master receiving data on debug:

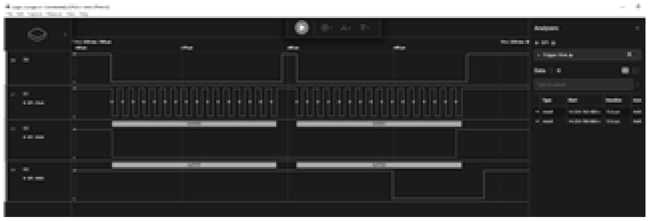


Slave receiving data on serial port:

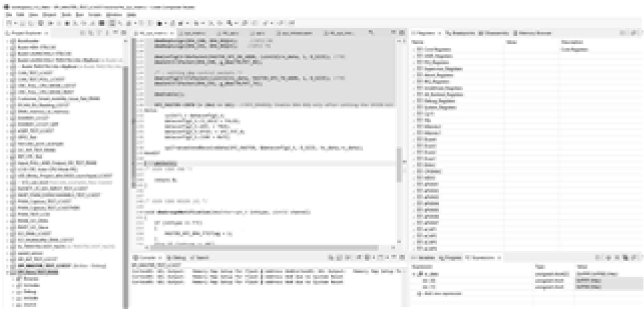


1. Output when CPOL=1 and CPHA=0:

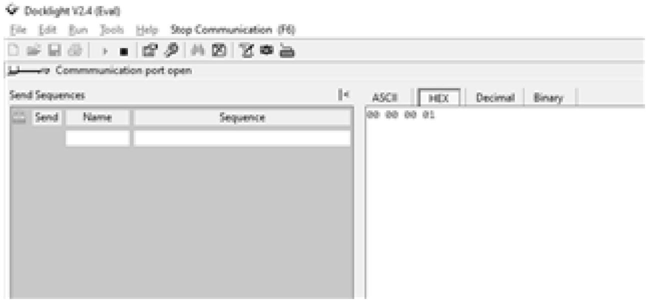
Waveforms between two:



Master receiving data on debug:



Slave receiving data on serial port:



Commmunication port open

```
00 00 00 01
```

Here are my testing codes for the reference:

SPI_MASTER_TEST_LC4357.zip

SPI_Slave_TEST_RM46.zip

Here i am suggesting customer to first compare my codes and his codes. If there were any difference (HALCoGen settings or code) between them then customer can change his codes according to mine once and can do the test. If customer facing issues, even after making changes also then we will setup a live debugging session and work on the issue again.

--
Thanks & regards,
Jagadish.

.

About TI

Quick links

Buying

Connect with us

Texas Instruments has been making progress possible for decades. We are a global semiconductor company that designs, manufactures, tests and sells analog and embedded processing chips. Our products help our customers efficiently manage power, accurately sense and transmit data and provide the core control or processing in their designs.

Accessibility | Cookie policy | Privacy policy | Terms of sale | Terms of use | Trademarks

Website feedback

Previewing Staged Changes