Hi,

I am trying to interface to the Heimann 16x16 sensor but having issue with the timings and synchronization. There is only one bi-directional data line as shown in the pinouts below. The device acts as a master spi device when the CONT line is 1 and the data is transmitted continuously. When the CONT line is 0, the device becomes a slave device and release the control of the clock and data line.

The code use for setting the the STM32 device to a slave device is shown below.

```
SPI_InitTypeDef spi_init_struct;
NVIC_InitTypeDef nvic_init_struct;

GPIO_InitStruct.GPIO_Pin = GPIO_Pin_9;
GPIO_InitStruct.GPIO_OType = GPIO_OType_PP;
GPIO_InitStruct.GPIO_PuPd = GPIO_PuPd_NOPULL;
GPIO_InitStruct.GPIO_Mode = GPIO_Mode_OUT;
GPIO_InitStruct.GPIO_Speed = GPIO_Speed_100MHz;
GPIO_Init(GPIOB, &GPIO_InitStruct);

GPIO_InitStruct.GPIO_Pin = GPIO_Pin_10 | GPIO_Pin_14;
GPIO_InitStruct.GPIO_PuPd = GPIO_PuPd_NOPULL;
GPIO_InitStruct.GPIO_OType = GPIO_OType_PP;
GPIO_InitStruct.GPIO_Mode = GPIO_Mode_AF;
GPIO_InitStruct.GPIO_Speed = GPIO_Speed_100MHz;
GPIO_Init(GPIOB, &GPIO_InitStruct);

GPIO_PinAFConfig(GPIOB, GPIO_PinSource10, GPIO_AF_SPI2);
GPIO_PinAFConfig(GPIOB, GPIO_PinSource14, GPIO_AF_SPI2);

RCC_APB1PeriphClockCmd(RCC_APB1Periph_SPI2, ENABLE);


SPI_StructInit(&spi_init_struct);
spi_init_struct.SPI_Direction = SPI_Direction_1Line_Rx;
spi_init_struct.SPI_Mode = SPI_Mode_Slave;
spi_init_struct.SPI_DataSize = SPI_DataSize_8b;
spi_init_struct.SPI_CPOL = SPI_CPOL_Low;
spi_init_struct.SPI_CPHA = SPI_CPHA_1Edge;
spi_init_struct.SPI_NSS = SPI_NSS_Soft;
spi_init_struct.SPI_BaudRatePrescaler = SPI_BaudRatePrescaler_64;
spi_init_struct.SPI_FirstBit = SPI_FirstBit_MSB;
spi_init_struct.SPI_CRCPolynomial = 0;
SPI_Init(SPI2, &spi_init_struct);

/* Configure the SPI interrupt priority */
nvic_init_struct.NVIC_IRQChannel = SPI2_IRQn;
nvic_init_struct.NVIC_IRQChannelPreemptionPriority = 0;
nvic_init_struct.NVIC_IRQChannelSubPriority = 0;
nvic_init_struct.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&nvic_init_struct);


/* Enable SPI */
SPI_Cmd(SPI2, ENABLE);
SPI_I2S_ITConfig(SPI2, SPI_I2S_IT_RXNE, ENABLE);
SPI_I2S_ITConfig(SPI2, SPI_I2S_IT_ERR, ENABLE);
SPI_Cmd(SPI2, ENABLE);
```

To set the clock for the sensor, the code is shown below.

```
TIM_TimeBaseInitTypeDef TIM_BaseStruct;

GPIO_InitTypeDef GPIO_InitStruct;

/* Clock for GPIOD */
```

```
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);

/* Alternating functions for pins */
GPIO_PinAFConfig(GPIOD, GPIO_PinSource13, GPIO_AF_TIM4);

/* Set pins */
GPIO_InitStruct.GPIO_Pin = GPIO_Pin_13;
GPIO_InitStruct.GPIO_OType = GPIO_OType_PP;
GPIO_InitStruct.GPIO_PuPd = GPIO_PuPd_NOPULL;
GPIO_InitStruct.GPIO_Mode = GPIO_Mode_AF;
GPIO_InitStruct.GPIO_Speed = GPIO_Speed_100MHz;
GPIO_Init(GPIOD, &GPIO_InitStruct);

/* Enable clock for TIM4 */
RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4, ENABLE);

TIM_BaseStruct.TIM_Prescaler = 0;
/* Count up */
TIM_BaseStruct.TIM_CounterMode = TIM_CounterMode_Up;

//TIM_BaseStruct.TIM_Period = 83; /* 10kHz PWM */
    TIM_BaseStruct.TIM_Period = 63; /* 10kHz PWM */
    //TIM_BaseStruct.TIM_Period = 41; /* 10kHz PWM */
TIM_BaseStruct.TIM_ClockDivision = TIM_CKD_DIV1;
TIM_BaseStruct.TIM_RepetitionCounter = 0;
/* Initialize TIM4 */
TIM_TimeBaseInit(TIM4, &TIM_BaseStruct);
/* Start count on TIM4 */
TIM_Cmd(TIM4, ENABLE);


TIM_OCInitTypeDef TIM_OCStruct;

/* Common settings */

/* PWM mode 2 = Clear on compare match */
/* PWM mode 1 = Set on compare match */
TIM_OCStruct.TIM_OCMode = TIM_OCMode_PWM2;
TIM_OCStruct.TIM_OutputState = TIM_OutputState_Enable;
TIM_OCStruct.TIM_OCPolarity = TIM_OCPolarity_Low;

//TIM_OCStruct.TIM_Pulse = 20; /* 50% duty cycle */
//TIM_OCStruct.TIM_Pulse = 43; /* 50% duty cycle */
TIM_OCStruct.TIM_Pulse = 32; /* 50% duty cycle */
TIM_OC2Init(TIM4, &TIM_OCStruct);
TIM_OC2PreloadConfig(TIM4, TIM_OCPreload_Enable);
```

The SPI interrupt is shown below. The data is sent to the PC via the USART.

```
void SPI2_IRQHandler(void)
{
  /* SPI in Slave Receiver mode--------------------------------------- */
  if (SPI_I2S_GetITStatus(SPI2, SPI_I2S_IT_RXNE) == SET)
  {
        newData = SPI_I2S_ReceiveData(SPI2);
        Queue_Push(&thermopileQueue, &newData);
  }

  /* SPI Error interrupt---------------------------------------- */
  if (SPI_I2S_GetITStatus(SPI2, SPI_I2S_IT_OVR) == SET)
  {
    SPI_I2S_ReceiveData(SPI2);
    SPI_I2S_GetITStatus(SPI2, SPI_I2S_IT_OVR);
  }
}
```

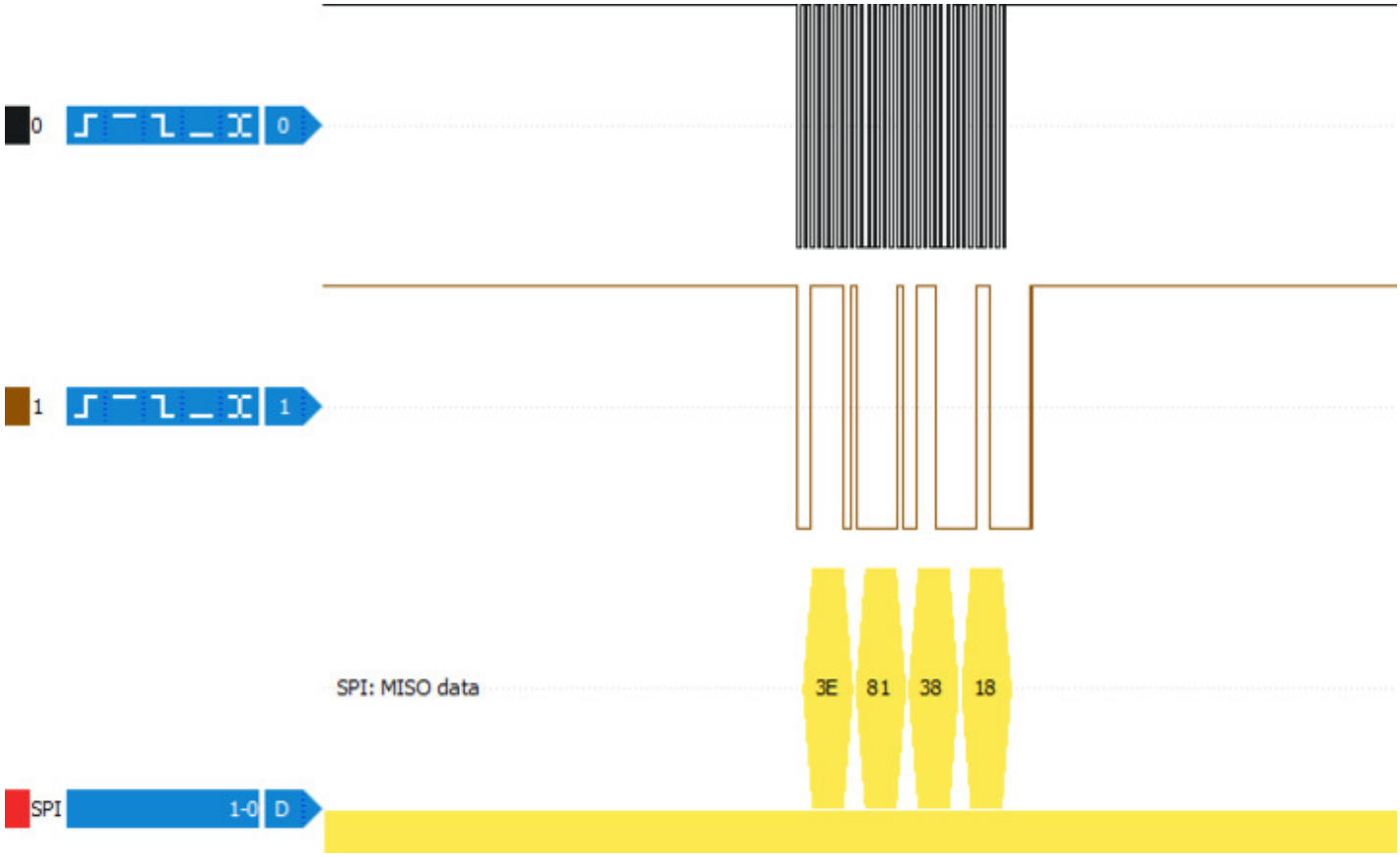
Nr.	*Name*	*Type*	*Description*
1	DATA_IO	digital I/O	Data
2	SCLK	digital I/O	Clock pulse SPI
3	VSS	Supply	0V
4	VDD	Supply	3.3V
5	CONT	digital Input	SPI switching see 5
6	MCLK	digital Input	Master Clock, typ. 1.877 MHz

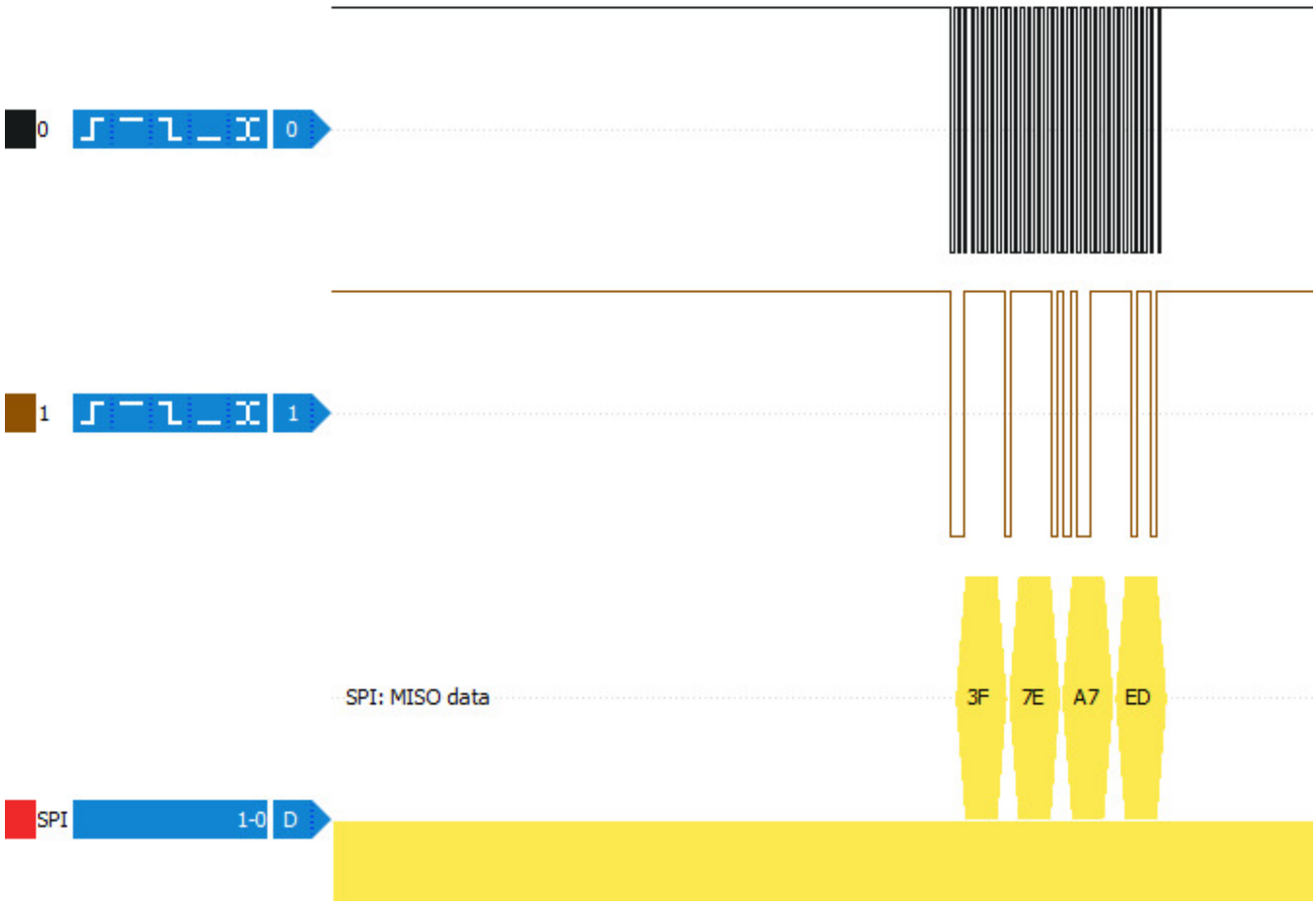
Read Register

31	*30*	*29*	*28*	*27*	*26*	*25*	*24*	*23….12*	*11….0*
L4	L3	L2	L1	L0	C3	C2	C1	ADC value1 (correspondent C0=1)	ADC value0 (correspondent C0=0)

L0…4: Address for the row of the Thermopiles; L4 = 1 corresponds to the reference row
C1…3: Address for the column without LSB
Format ADC-values: unsigned

The logic analyser capture below shows the address increment.





Can anyone see anything wrong with the code and why the data is not sync?

Paul