Configuration of SPI Emulation by software

This SPI uses a Timer in output compare mode (Pulse Width Modulation), to generate the SCK signal to synchronize the communication, this can be configurate with TIMx to enable the internal counter, this depends directly on System Clock.

Also uses two general purpose input output pins to emulate MISO and MOSI signals, those signals are synchronized with DMA to register the data input and control the pin level in the case of data output, this without stop the normal function of the other modules, also generates an interruption service routine when the communication has finished.

**Configuration**

To include the SPI Emulation functionality to a project with STM32Cube, you just need to include the files “stm32f4xx\_hal\_spi\_master\_emul.c” and “stm32f4xx\_hal\_spi\_master\_emul.h”. These files contain all the configuration, functions and operation of SPI emulation module, these files are provided on this GitHub repository. Put attention in the following lines:

313-> TimHandle.Instance = TIM1; // select the correct TIMx  
325-> HAL\_TIM\_PWM\_ConfigChannel(&TimHandle, &sConfig, TIM\_CHANNEL\_1); // Channel selection  
343-> HAL\_TIM\_PWM\_ConfigChannel(&TimHandle, &sConfig1, TIM\_CHANNEL\_2); // channel selection

Also, you need to add the following code lines in file “main.h”, in the sections destinated to **user code addition**. On this file you can select the

main.h

/\* Definition for TIMx clock resources \*/

**#define** TIMx TIM1

**#define** TIMx\_CLK\_ENABLE() \_\_HAL\_RCC\_TIM1\_CLK\_ENABLE()

/\* Definition for TIMx Pins \*/

**#define** TIMx\_CHANNEL\_GPIO\_PORT() \_\_HAL\_RCC\_GPIOE\_CLK\_ENABLE()

**#define** GPIO\_PORT GPIOE

**#define** GPIO\_PIN\_CHANNEL GPIO\_PIN\_11

**#define** GPIO\_AF\_TIMx GPIO\_AF1\_TIM1

/\* Initialize GPIO and pin number for SPI Emulation \*/

**#define** SPI\_EMUL\_Clk\_PIN GPIO\_PIN\_11

**#define** SPI\_EMUL\_Clk\_PORT GPIOE

**#define** SPI\_EMUL\_TX\_PIN GPIO\_PIN\_10

**#define** SPI\_EMUL\_TX\_PORT GPIOC

**#define** SPI\_EMUL\_RX\_PIN GPIO\_PIN\_11

**#define** SPI\_EMUL\_RX\_PORT GPIOC

/\* Enable the clock for port SPI Emulation \*/

**#define** SPI\_EMUL\_Clk\_GPIO\_CLK\_ENABLE() \_\_GPIOE\_CLK\_ENABLE();

**#define** SPI\_EMUL\_TX\_GPIO\_CLK\_ENABLE() \_\_GPIOC\_CLK\_ENABLE();

**#define** SPI\_EMUL\_RX\_GPIO\_CLK\_ENABLE() \_\_GPIOC\_CLK\_ENABLE();

/\* Size of Trasmission buffer \*/

**#define** TXBUFFERSIZE (COUNTOF(aTxBuffer) - 1)

/\* Size of Receive buffer \*/

**#define** BUFFERSIZE TXBUFFERSIZE

/\* Exported macro ------------------------------------------------------------\*/

**#define** COUNTOF(\_\_BUFFER\_\_) (**sizeof**(\_\_BUFFER\_\_) / **sizeof**(\*(\_\_BUFFER\_\_)))

On file “stm32f7xx\_hal\_msp.c” you need to add the functions to configurate the peripheral involve in the function of SPI Emulator, those functions are described as follow:

stm32f7xx\_hal\_msp.c

/\* USER CODE BEGIN Includes \*/

**#include** "stm32f4xx\_spi\_master\_emul.h"

/\* USER CODE END Includes \*/

/\* USER CODE BEGIN TD \*/

**extern** SPI\_Emul\_HandleTypeDef SpiEmulHandle;

SPI\_Emul\_HandleTypeDef \*hspi;

/\* USER CODE END TD \*/

/\* USER CODE BEGIN 1 \*/

/\*\*

\* @brief TIM MSP Initialization

\* This function configures the hardware resources used in this example:

\* - Peripheral's clock enable

\* - Peripheral's GPIO Configuration

\* @param htim: TIM handle pointer

\* @retval None

\*/

**void** **HAL\_TIM\_PWM\_MspInit**(TIM\_HandleTypeDef \*htim) {

GPIO\_InitTypeDef GPIO\_InitStruct;

/\*##-1- Enable peripherals and GPIO Clocks #################################\*/

/\* TIMx Peripheral clock enable \*/

TIMx\_CLK\_ENABLE();

/\* Enable GPIO channels Clock \*/

TIMx\_CHANNEL\_GPIO\_PORT();

/\* Configure (TIMx\_Channel) in Alternate function, push-pull and 100MHz speed \*/

GPIO\_InitStruct.Pin = GPIO\_PIN\_CHANNEL;

GPIO\_InitStruct.Mode = GPIO\_MODE\_AF\_PP;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_HIGH;

GPIO\_InitStruct.Alternate = GPIO\_AF\_TIMx;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

HAL\_GPIO\_Init(GPIO\_PORT, &GPIO\_InitStruct);

}

/\*\*

\* @brief Initializes the SPI MSP.

\* This function configures the SPI resources used in this example:

\* @param htim: SPI handle pointer

\* @retval None

\*/

**void** **HAL\_SPI\_Emul\_MspInit**(SPI\_Emul\_HandleTypeDef \*hspi) {

GPIO\_InitTypeDef GPIO\_InitStruct;

/\*##-1- Enable peripherals and GPIO Clocks #######################\*/

/\* Enable clock for SPI Emul \*/

\_\_SPI\_EMUL\_CLK\_ENABLE();

/\* Enable GPIO clock \*/

SPI\_EMUL\_Clk\_GPIO\_CLK\_ENABLE();

SPI\_EMUL\_TX\_GPIO\_CLK\_ENABLE();

SPI\_EMUL\_RX\_GPIO\_CLK\_ENABLE();

/\* Initialize SPI Emulation port name \*/

SpiEmulHandle.ClkPortName = SPI\_EMUL\_Clk\_PORT;

SpiEmulHandle.TxPortName = SPI\_EMUL\_TX\_PORT;

SpiEmulHandle.RxPortName = SPI\_EMUL\_RX\_PORT;

/\*Initialize SPI Emulation pin number for Clk TX \*/

SpiEmulHandle.Init.ClkPinNumber = SPI\_EMUL\_Clk\_PIN;

SpiEmulHandle.Init.TxPinNumber = SPI\_EMUL\_TX\_PIN;

SpiEmulHandle.Init.RxPinNumber = SPI\_EMUL\_RX\_PIN;

/\* Configure GPIOA for SPI Emulation Clock \*/

GPIO\_InitStruct.Pin = SPI\_EMUL\_Clk\_PIN;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_PULLDOWN;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_HIGH;

HAL\_GPIO\_Init(SPI\_EMUL\_Clk\_PORT, &GPIO\_InitStruct);

/\* Configure GPIOC for SPI Emulation Tx \*/

GPIO\_InitStruct.Pin = SPI\_EMUL\_TX\_PIN;

GPIO\_InitStruct.Mode = GPIO\_MODE\_OUTPUT\_PP;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_HIGH;

HAL\_GPIO\_Init(SPI\_EMUL\_TX\_PORT, &GPIO\_InitStruct);

/\* Configure GPIOC for SPI Emulation Rx \*/

GPIO\_InitStruct.Pin = SPI\_EMUL\_RX\_PIN;

GPIO\_InitStruct.Mode = GPIO\_MODE\_INPUT;

GPIO\_InitStruct.Pull = GPIO\_NOPULL;

GPIO\_InitStruct.Speed = GPIO\_SPEED\_HIGH;

HAL\_GPIO\_Init(SPI\_EMUL\_RX\_PORT, &GPIO\_InitStruct);

/\*##-1- Enable peripherals and GPIO Clocks #######################\*/

/\* Enable clock for SPI Emul \*/

\_\_SPI\_EMUL\_CLK\_ENABLE();

}

/\* USER CODE END 1 \*/

On file “main.c” you can configurate and produce an example as follow:

main.c

/\* USER CODE BEGIN Includes \*/

**#include** "stm32f4xx\_spi\_master\_emul.h"

/\* USER CODE END Includes \*/

/\* USER CODE BEGIN PTD \*/

SPI\_Emul\_HandleTypeDef SpiEmulHandle;

/\* USER CODE END PTD \*/

/\* USER CODE BEGIN PV \*/

/\* Buffer used for transmission \*/

uint8\_t aTxBuffer[] = "\*\*\*\* SPI \*\*\*\*\n";

// uint8\_t aTxBuffer[] = "aaaaaaaaaaaaaa";

uint8\_t outBuffer[4][8] = { "fabian ", "andres ", "castano ", "usuga " };

// uint8\_t outBuffer[4][8] = {"aaaaaaaa","aaaaaaaa","aaaaaaaa","aaaaaaaa"};

uint8\_t inData[8];

uint8\_t cont = 0;

/\* Buffer used for reception \*/

uint8\_t aRxBuffer[BUFFERSIZE];

/\* USER CODE END PV \*/

/\* USER CODE BEGIN PFP \*/

**void** **HAL\_SPI\_Emul\_TxCpltCallback**(SPI\_Emul\_HandleTypeDef \*hspi);

**static** uint16\_t **Buffercmp**(uint8\_t \*pBuffer1, uint8\_t \*pBuffer2,

uint16\_t BufferLength);

**volatile** **int** RestTx = 0;

/\* USER CODE END PFP \*/

/\*\*

\* @brief The application entry point.

\* @retval int

\*/

**int** **main**(**void**) {

/\* USER CODE BEGIN 1 \*/

/\* USER CODE END 1 \*/

/\* MCU Configuration--------------------------------------------------------\*/

/\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/

HAL\_Init();

/\* USER CODE BEGIN Init \*/

/\* USER CODE END Init \*/

/\* Configure the system clock \*/

SystemClock\_Config();

/\* USER CODE BEGIN SysInit \*/

/\* USER CODE END SysInit \*/

/\* Initialize all configured peripherals \*/

MX\_GPIO\_Init();

// MX\_ETH\_Init();

MX\_USART3\_UART\_Init();

MX\_USB\_OTG\_FS\_PCD\_Init();

// MX\_TIM1\_Init();

/\* USER CODE BEGIN 2 \*/

SpiEmulHandle.Init.Mode = SPI\_EMUL\_MODE\_MASTER;

SpiEmulHandle.Init.Direction = SPI\_EMUL\_DIRECTION\_TX\_RX;

SpiEmulHandle.Init.DataSize = SPI\_EMUL\_DATASIZE\_8BIT;

SpiEmulHandle.Init.CLKPolarity = SPI\_EMUL\_POLARITY\_LOW;

SpiEmulHandle.Init.CLKPhase = SPI\_EMUL\_PHASE\_1EDGE;

SpiEmulHandle.Init.SPI\_Clk = 1000000;

SpiEmulHandle.Init.FirstBit = SPI\_EMUL\_FIRSTBIT\_MSB;

**if** (HAL\_SPI\_Emul\_Init(&SpiEmulHandle) != *HAL\_OK*) {

Error\_Handler();

}

/\*## Start the transmission process #####################################\*/

/\* While the SPI Emulation in reception process, user can transmit data through

"aTxBuffer" buffer \*/

**if** (HAL\_SPI\_Emul\_TransmitReceive\_DMA(&SpiEmulHandle, (uint8\_t\*) aTxBuffer,

(uint8\_t\*) aRxBuffer, TXBUFFERSIZE) != *HAL\_OK*) {

Error\_Handler();

HAL\_GPIO\_WritePin(LD2\_GPIO\_Port, LD2\_Pin, *SET*);

}

**while** (\_\_HAL\_SPI\_EMUL\_GET\_FLAG(&SpiEmulHandle, SPI\_EMUL\_FLAG\_TC) != *SET*) {

}

/\*##-4- Compare the sent and received buffers ##############################\*/

**if** (Buffercmp((uint8\_t\*) aTxBuffer, (uint8\_t\*) aRxBuffer, BUFFERSIZE)) {

/\* Transfer error in transmission process \*/

Error\_Handler();

HAL\_GPIO\_WritePin(LD3\_GPIO\_Port, LD3\_Pin, *SET*);

}

HAL\_GPIO\_WritePin(LD1\_GPIO\_Port, LD1\_Pin, *SET*);

HAL\_Delay(200);

/\* USER CODE END 2 \*/

/\* Infinite loop \*/

/\* USER CODE BEGIN WHILE \*/

**while** (1) {

/\* USER CODE END WHILE \*/

/\* USER CODE BEGIN 3 \*/

**if** (SpiEmulHandle.State == *HAL\_SPI\_EMUL\_STATE\_READY*) {

SpiEmulHandle.TxXferCount = 0;

SpiEmulHandle.RxXferCount = 0;

HAL\_GPIO\_WritePin(CS\_EM\_GPIO\_Port, CS\_EM\_Pin, 0);

**if** (HAL\_SPI\_Emul\_TransmitReceive\_DMA(&SpiEmulHandle,

(uint8\_t\*) outBuffer[cont], (uint8\_t\*) inData, 8)

!= *HAL\_OK*) {

Error\_Handler();

HAL\_GPIO\_WritePin(LD2\_GPIO\_Port, LD2\_Pin, *SET*);

}

**while** (\_\_HAL\_SPI\_EMUL\_GET\_FLAG(&SpiEmulHandle, SPI\_EMUL\_FLAG\_TC)

!= *SET*) {

}

HAL\_UART\_Transmit(&huart3, outBuffer[cont], 8, HAL\_MAX\_DELAY);

// HAL\_Delay(100);

HAL\_UART\_Transmit(&huart3, inData, 8, HAL\_MAX\_DELAY);

cont++;

**if** (cont >= 4)

cont = 0;

HAL\_Delay(2000);

}

}

/\* USER CODE END 3 \*/

}

/\* USER CODE BEGIN 4 \*/

**void** **HAL\_SPI\_Emul\_TxCpltCallback**(SPI\_Emul\_HandleTypeDef \*hspi) {

HAL\_GPIO\_WritePin(CS\_EM\_GPIO\_Port, CS\_EM\_Pin, 1);

}

/\*\*

\* @brief Compares two buffers.

\* @param pBuffer, pBuffer1: buffers to be compared.

\* @param BufferLength: buffer's length

\* @retval PASSED: pBuffer identical to pBuffer1

\* FAILED: pBuffer differs from pBuffer1

\*/

**static** uint16\_t **Buffercmp**(uint8\_t \*pBuffer1, uint8\_t \*pBuffer2,

uint16\_t BufferLength) {

**while** (BufferLength--) {

**if** ((\*pBuffer1) != \*pBuffer2) {

**return** BufferLength;

}

pBuffer1++;

pBuffer2++;

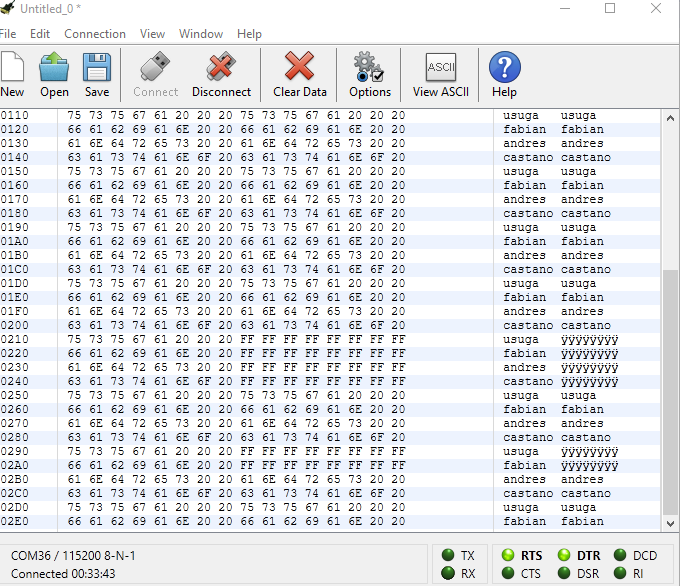
}

**return** 0;

}

/\* USER CODE END 4 \*/

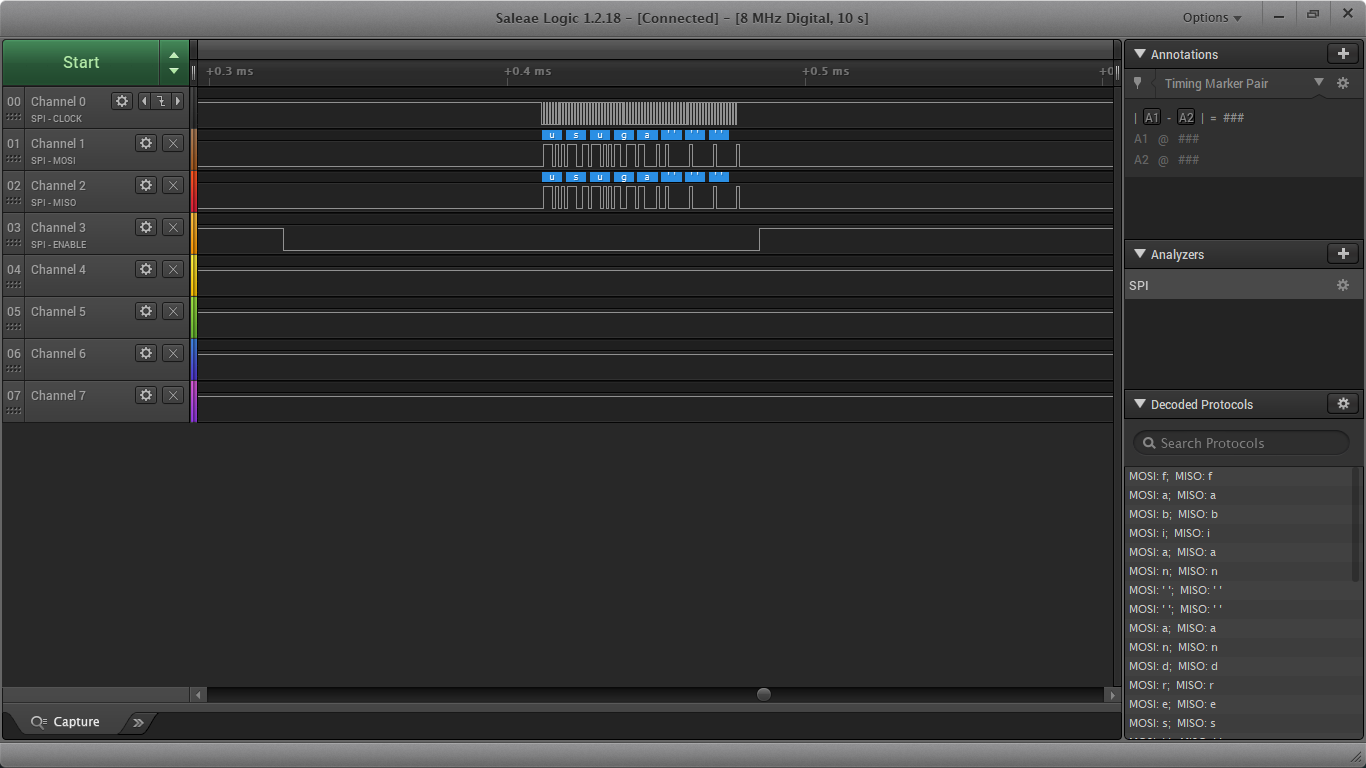
**Test**



Purpose lost

Received

Sent



SCK

MOSI

MISO

CS

Logic Analyzer lost