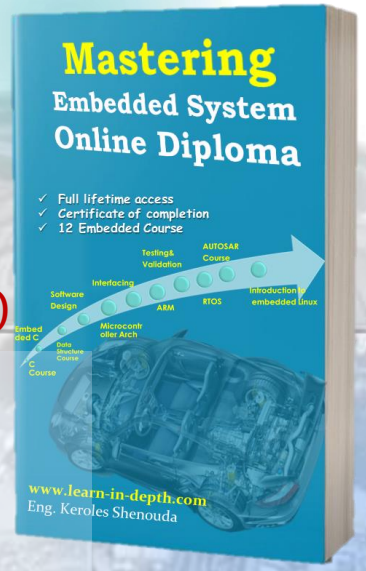
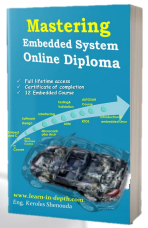
Mastering Embedded System

Online Diploma

- ✓ Full lifetime access
- ✓ Access on Android mobile and PC (Windows)
- ✓ Certificate of completion
- ✓ 12 Embedded Course





#LEARN_IN DEPTH

#Be_professional_in embedded_system

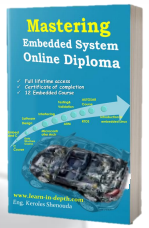
Unit 8 (MCU Interfacing) . lesson 5 SPI Controller (Read from TRM And Write a Driver)

- SPI Controller in STM32F103XX
- Read TRM (Technical Reference Manual)
  - SPI block diagram
  - Functional Description
  - SPI Communication Modes
  - Clock phase and clock polarity
  - Data frame format
  - SPI Clock Speed "SCLK"
  - Slave Master Management
- Write SPI Deriver From Scratch
  - Add the SPI in Device Header for SoC
  - Create SPI Configuration Structure
  - Implement SPI APIs
  - Handling SPI Interrupt
  - Create SPI Interrupt Callback for Application Layer

- SPI Lab1 :
  - UART > MCU > SPI
  - Debug SPI
  - Analyze SPI Signals
- SPI Lab2 :
  - Terminal1 <----> USART1 : MCU1 : (SPI1 Master) ---> (SPI2 Slave) :MCU2: USART2 ---> Terminal2

https://www.facebook.com/groups/ Eng. Keroles Shenouda

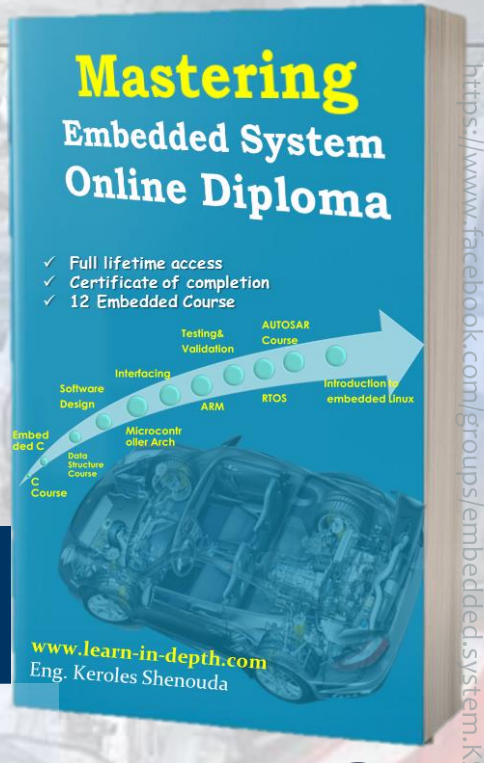
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

#Be_professional_in_embedded_system

SPI Controller in STM32F103XX



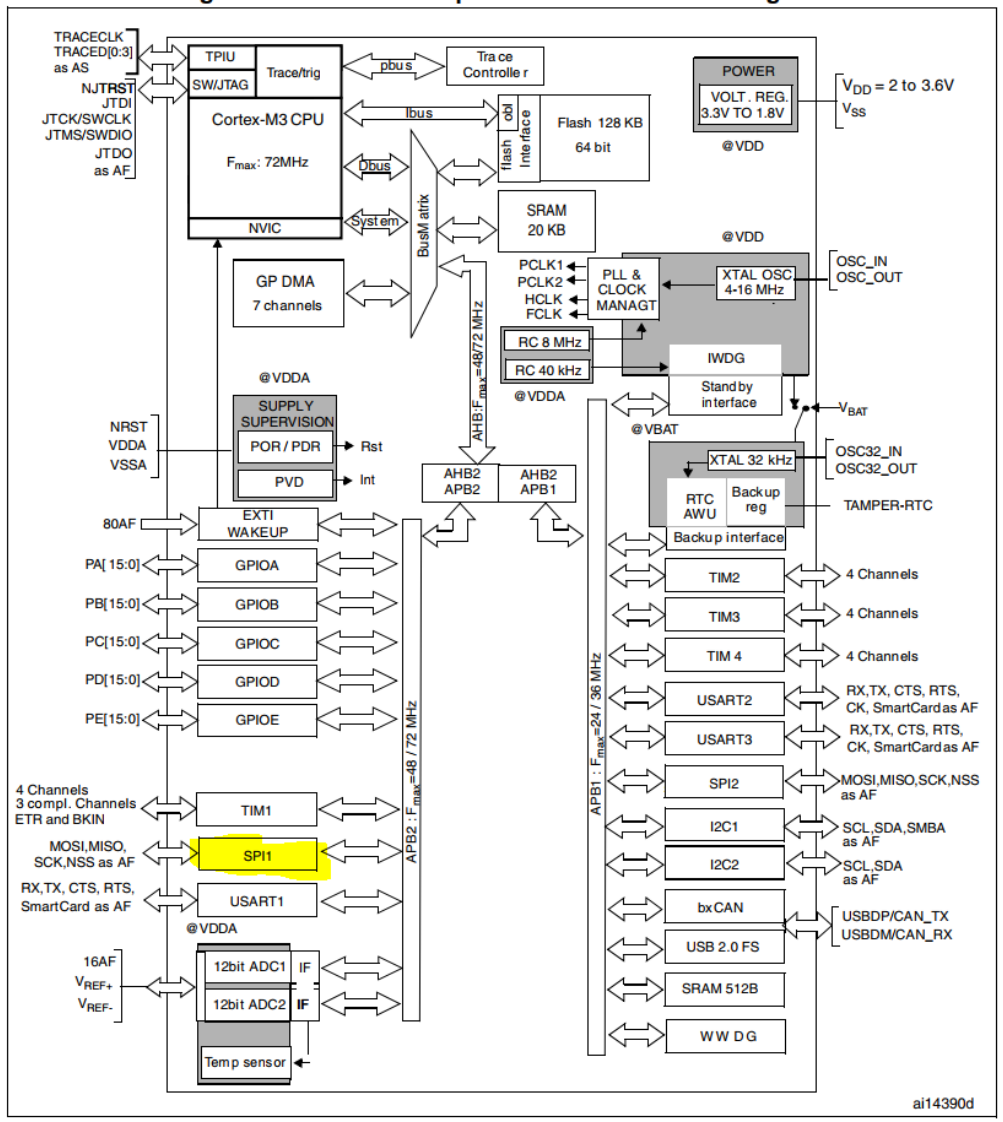
https://www.facebook.com/groups/embedded.system.KS/
Eng. Keroles Shenouda

LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

SPI Controller

Figure 1. STM32F103xx performance line block diagram



FOLLOW US
Press here
3
Mastering Embedded System Online Diploma
#LEARN_IN DEPTH
#Be_professional_in embedded_system
Eng. Keroles Shenouda
https://www.facebook.com/groups/embedded.system.KS/

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

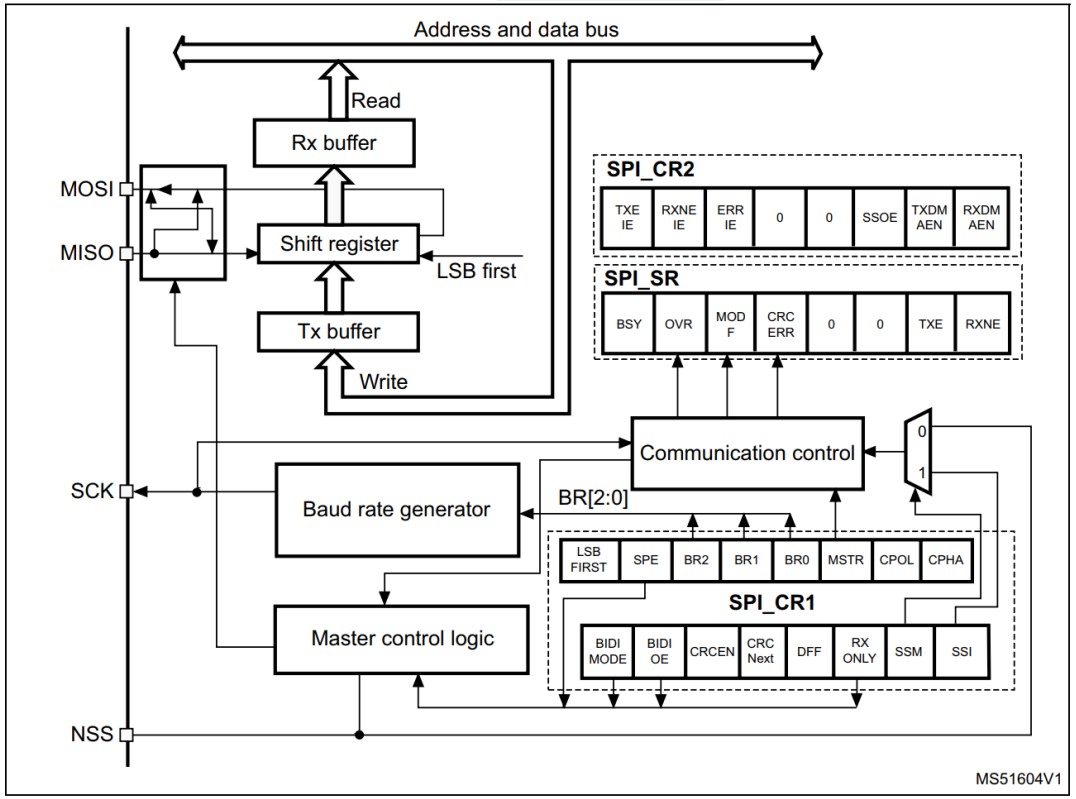
FOLLOW US
Press here
4
Mastering Embedded System Online Diploma
#LEARN_IN DEPTH
#Be_professional_in_embedded_system
Eng. Keroles Shenouda
https://www.facebook.com/groups/embedded.system.KS/

SPI block diagram

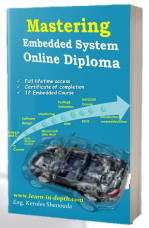
25.3.1 General description

The block diagram of the SPI is shown in *Figure 238*.

Figure 238. SPI block diagram



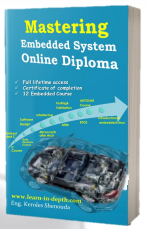
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Usually, the SPI is connected to external devices through four pins:

- MISO: Master In / Slave Out data. This pin can be used to transmit data in slave mode and receive data in master mode.
- MOSI: Master Out / Slave In data. This pin can be used to transmit data in master mode and receive data in slave mode.
- SCK: Serial Clock output for SPI masters and input for SPI slaves.
- NSS: Slave select. This is an optional pin to select a slave device. This pin acts as a 'chip select' to let the SPI master communicate with slaves individually and to avoid contention on the data lines. Slave NSS inputs can be driven by standard IO ports on the master device. The NSS pin may also be used as an output if enabled (SSOE bit) and driven low if the SPI is in master configuration. In this manner, all NSS pins from devices connected to the Master NSS pin see a low level and become slaves when they are configured in NSS hardware mode. When configured in master mode with NSS configured as an input (MSTR=1 and SSOE=0) and if NSS is pulled low, the SPI enters the master mode fault state: the MSTR bit is automatically cleared and the device is configured in slave mode (refer to *Section 25.3.10*).

Single master/ single slave application

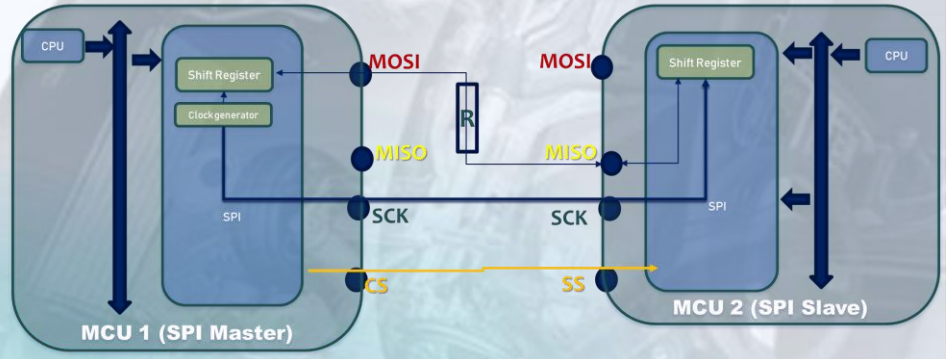


#LEARN_IN DEPTH

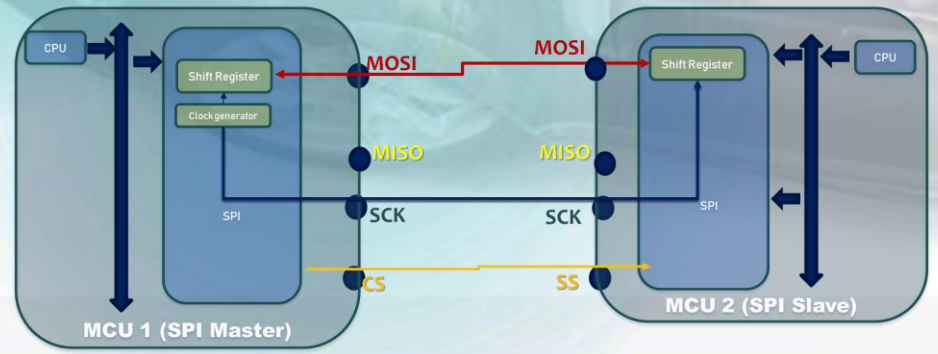
#Be_professional_in_embedded_system

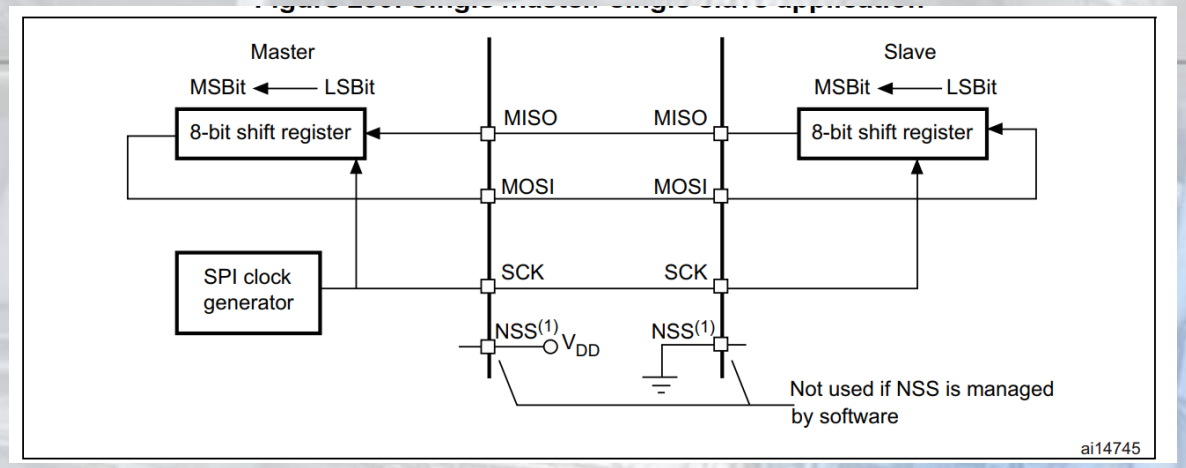
Eng. Keroles Shenouda
https://www.facebook.com/groups/embedded.system.KS/

Half-Duplex Communication

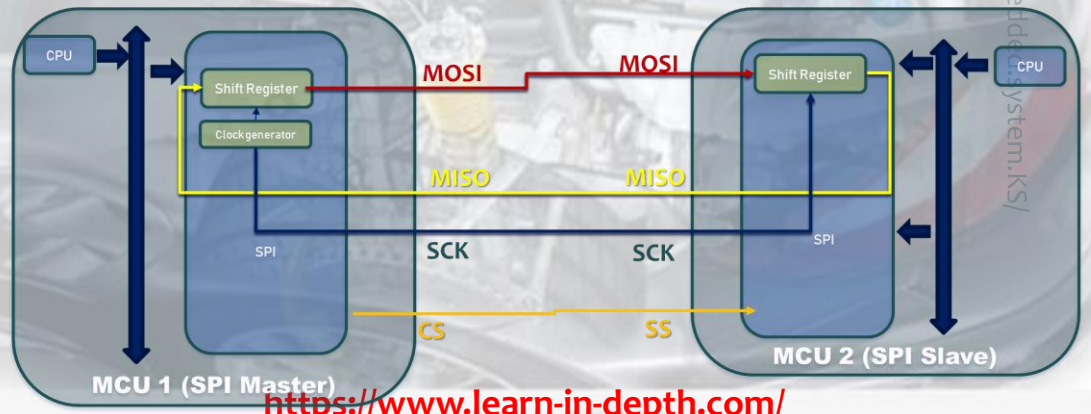


Simplex Communication
Master: Transmit only
Slave : receive only





Full-Duplex Communication

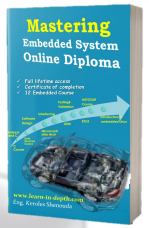


https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

Slave select (NSS) pin management

Software NSS management (SSM = 1)

FOLLOW US
Press here
10



#LEARN_IN DEPTH

#Be_professional_in_embedded_system

25.5.1 SPI control register 1 (SPI_CR1) (not used in I²S mode)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIDI MODE	BIDI OE	CRC EN	CRC NEXT	DFF	RX ONLY	SSM	SSI	*LSB FIRST*	SPE	BR [2:0]			MSTR	CPOL	CPHA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 8 **SSI:** Internal slave select

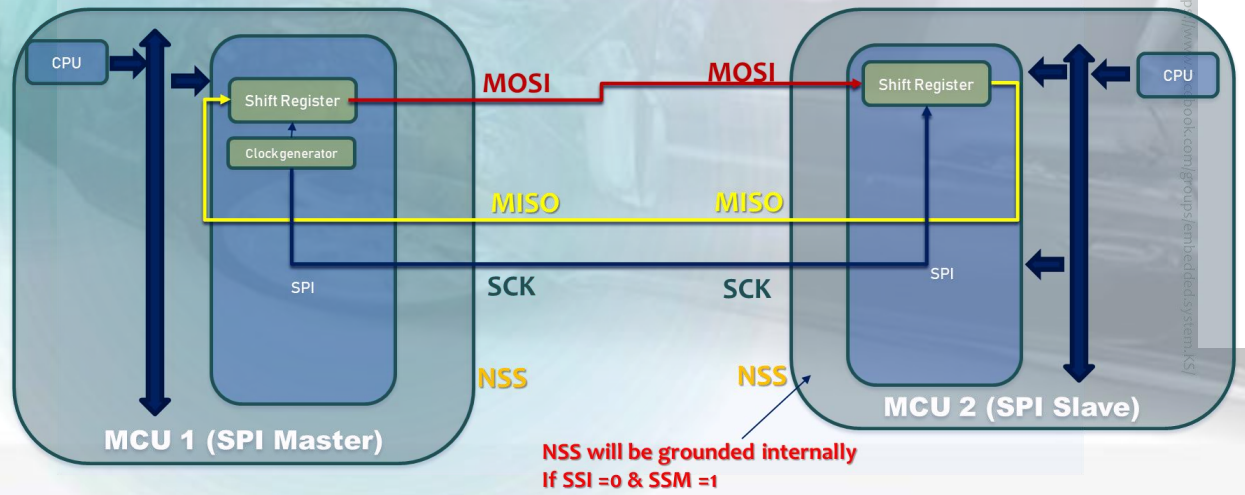
This bit has an effect only when the SSM bit is set. The value of this bit is forced onto the NSS pin and the IO value of the NSS pin is ignored.

Bit 9 **SSM:** Software slave management

When the SSM bit is set, the NSS pin input is replaced with the value from the SSI bit.

0: Software slave management disabled

1: Software slave management enabled



Slave select (NSS) pin management

Hardware or software slave select management can be set using the SSM bit in the SPI_CR1 register.

- Software NSS management (SSM = 1)

  The slave select information is driven internally by the value of the SSI bit in the SPI_CR1 register. The external NSS pin remains free for other application uses.

- Hardware NSS management (SSM = 0)

  Two configurations are possible depending on the NSS output configuration (SSOE bit in register SPI_CR2).

  – NSS output enabled (SSM = 0, SSOE = 1)

  This configuration is used only when the device operates in master mode. The NSS signal is driven low when the master starts the communication and is kept low until the SPI is disabled.

  – NSS output disabled (SSM = 0, SSOE = 0)

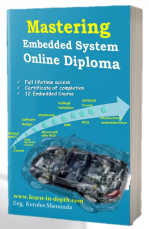
  This configuration allows multimaster capability for devices operating in master mode. For devices set as slave, the NSS pin acts as a classical NSS input: the slave is selected when NSS is low and deselected when NSS high.

https://www.facebook.com/groups/embedded.system.KS/
Eng. Keroles Shenouda

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

Slave select (NSS) pin management

Hardware NSS management (SSM = 0)



#LEARN_IN DEPTH

#Be_professional_in_embedded_system

Eng. Keroles Shenouda
https://www.facebook.com/groups/embedded.system.KS/

Slave select (NSS) pin management

Hardware or software slave select management can be set using the SSM bit in the SPI_CR1 register.

- Software NSS management (SSM = 1)

  The slave select information is driven internally by the value of the SSI bit in the SPI_CR1 register. The external NSS pin remains free for other application uses.

- Hardware NSS management (SSM = 0)

  Two configurations are possible depending on the NSS output configuration (SSOE bit in register SPI_CR2).

  – NSS output enabled (SSM = 0, SSOE = 1)

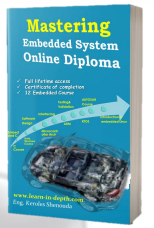
    This configuration is used only when the device operates in master mode. The NSS signal is driven low when the master starts the communication and is kept low until the SPI is disabled.

  – NSS output disabled (SSM = 0, SSOE = 0)

    This configuration allows multimaster capability for devices operating in master mode. For devices set as slave, the NSS pin acts as a classical NSS input: the slave is selected when NSS is low and deselected when NSS high.

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

FOLLOW US
Press here
17



#LEARN_IN DEPTH
#Be_professional_in_embedded_system

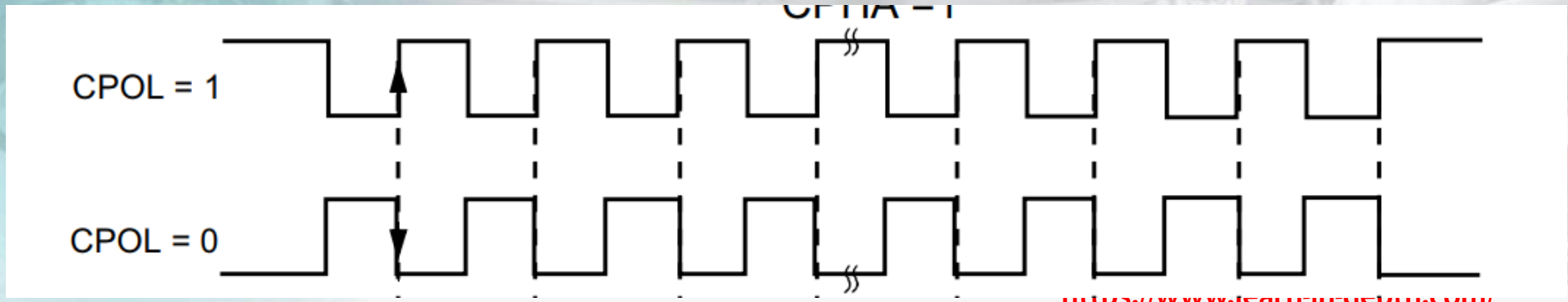
Clock phase and clock polarity

Clock phase and clock polarity

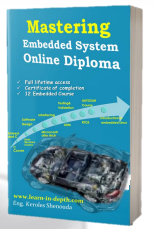
Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits in the SPI_CR1 register. The CPOL (clock polarity) bit controls the steady state value of the clock when no data is being transferred. This bit affects both master and slave modes. If CPOL is reset, the SCK pin has a low-level idle state. If CPOL is set, the SCK pin has a high-level idle state.

If the CPHA (clock phase) bit is set, the second edge on the SCK pin (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set) is the MSBit capture strobe. Data are latched on the occurrence of the second clock transition. If the CPHA bit is reset, the first edge on the SCK pin (falling edge if CPOL bit is set, rising edge if CPOL bit is reset) is the MSBit capture strobe. Data are latched on the occurrence of the first clock transition.

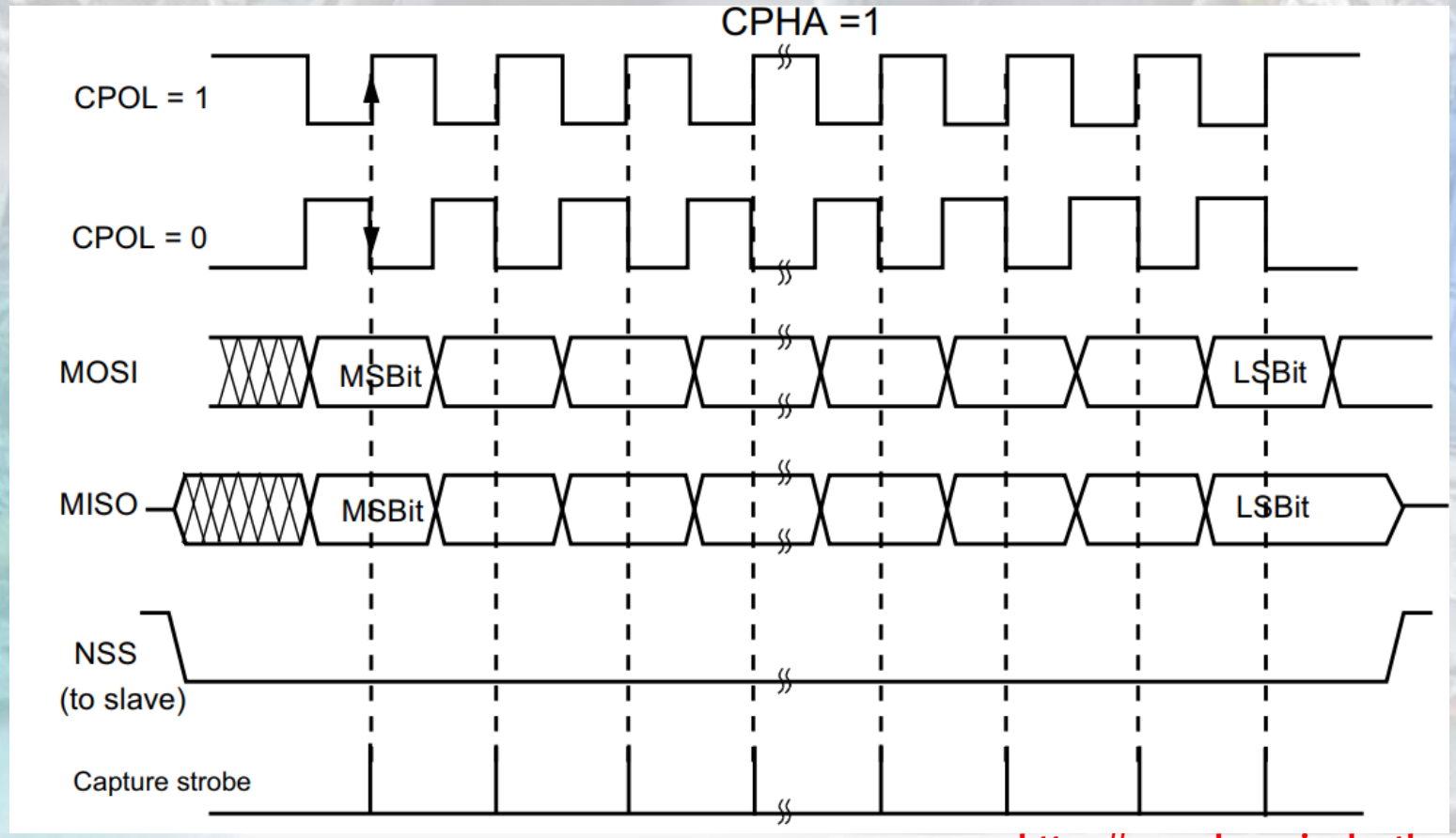
The combination of the CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edge.



https://www.facebook.com/groups/embedded.system.KS/

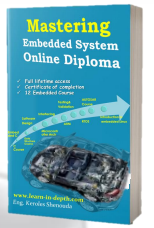


Data clock timing diagram

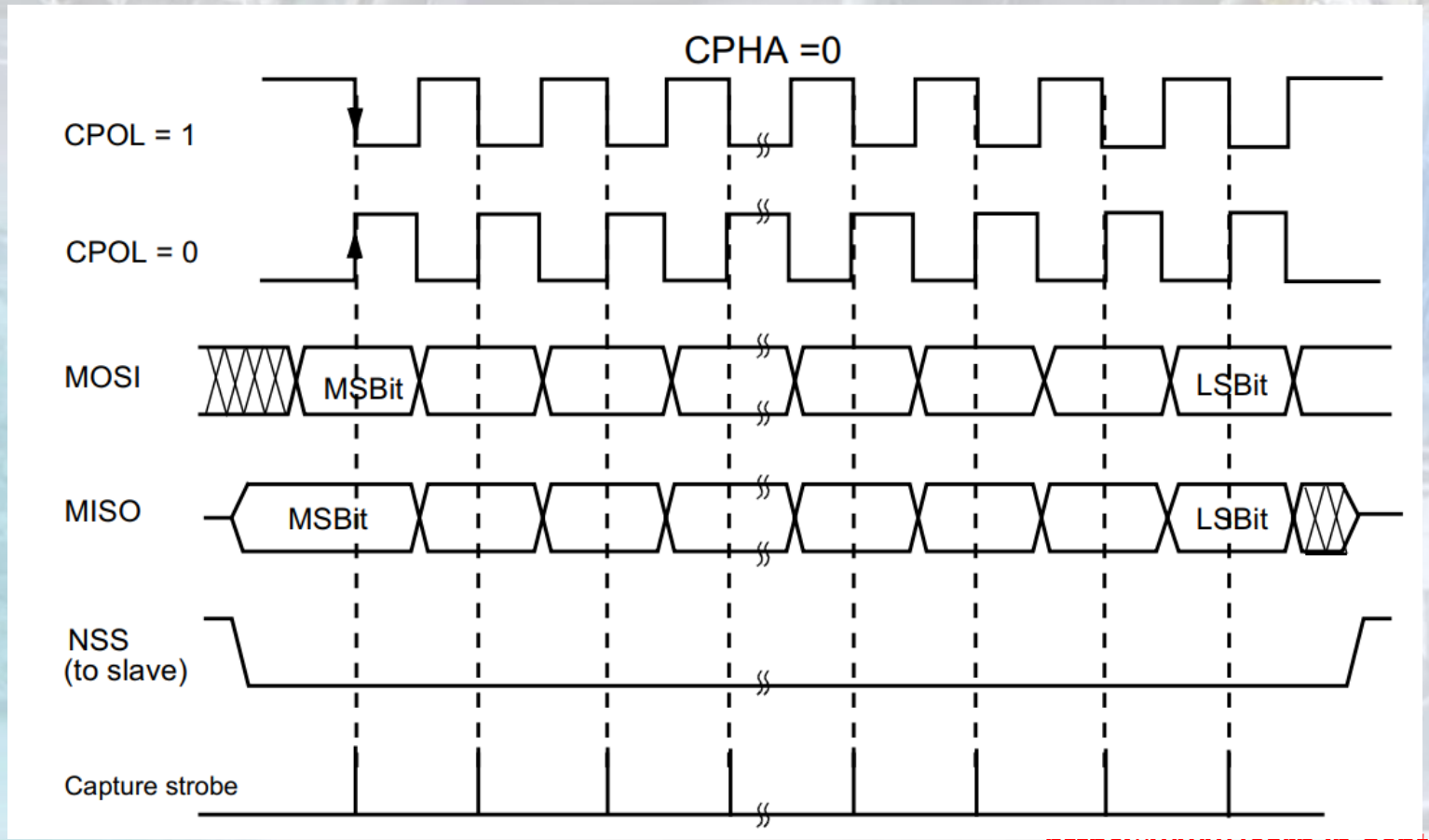


Eng. Keroles Shenouda
https://www.facebook.com/groups/embedded.system.KS/

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

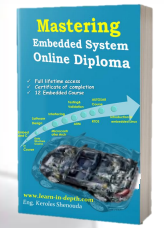


Data clock timing diagram



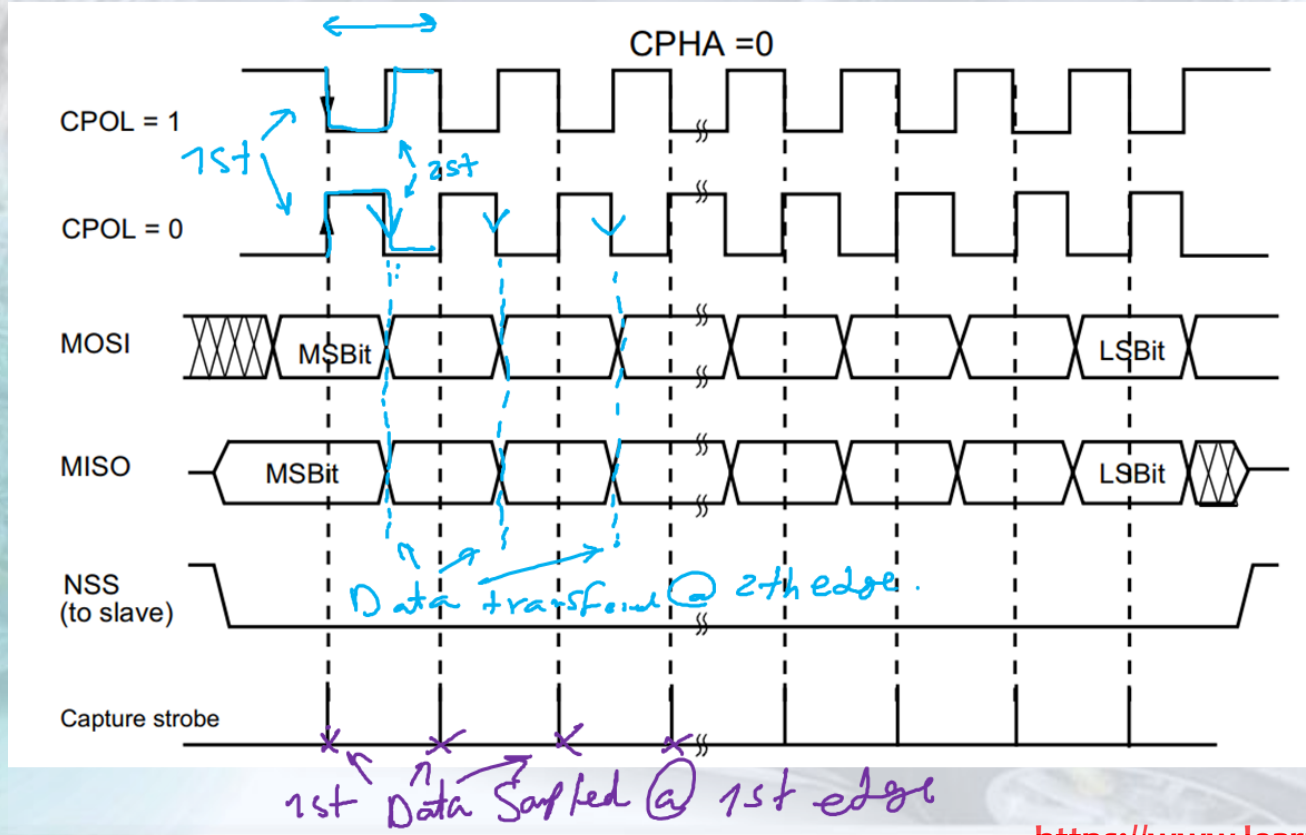
Eng. Keroles Shenouda
https://www.facebook.com/groups/embedded.system.KS/

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



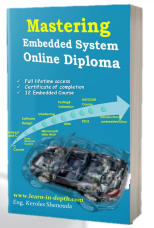
Eng. Keroles Shenouda
https://www.facebook.com/groups/embedded.system.KS/

Data clock timing diagram



https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

FOLLOW US
Press here
23



#LEARN_IN DEPTH
#Be_professional_in_embedded_system
Eng. Keroles Shenouda
https://www.facebook.com/groups/embedded.system.KS/

Clock phase and clock polarity

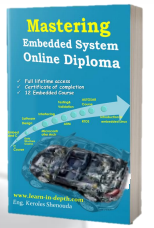
- **SPI_CR1**
  Master/Slave Should have the same Configuration

Bit1 **CPOL:** Clock polarity
0: CK to 0 when idle
1: CK to 1 when idle

Bit 0 **CPHA:** Clock phase
0: The first clock transition is the first data capture edge
1: The second clock transition is the first data capture edge

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

FOLLOW US
Press here
24



#LEARN_IN DEPTH
#Be_professional_in_embedded_system
Eng. Keroles Shenouda
https://www.facebook.com/groups/embedded.system.KS

Data frame format

Data frame format

Data can be shifted out either MSB-first or LSB-first depending on the value of the LSBFIRST bit in the SPI_CR1 Register.

Each data frame is 8 or 16 bits long depending on the size of the data programmed using the DFF bit in the SPI_CR1 register. The selected data frame format is applicable for transmission and/or reception.

25.5.1 SPI control register 1 (SPI_CR1) (not used in I^2S mode)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIDI MODE	BIDI OE	CRC EN	CRC NEXT	DFF	RX ONLY	SSM	SSI	*LSB FIRST*	SPE	BR [2:0]			MSTR	CPOL	CPHA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 7 **LSBFIRST:** Frame format

0: MSB transmitted first

1: LSB transmitted first

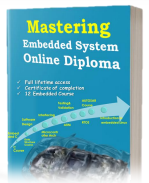
Bit 11 **DFF:** Data frame format

0: 8-bit data frame format is selected for transmission/reception

1: 16-bit data frame format is selected for transmission/reception

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

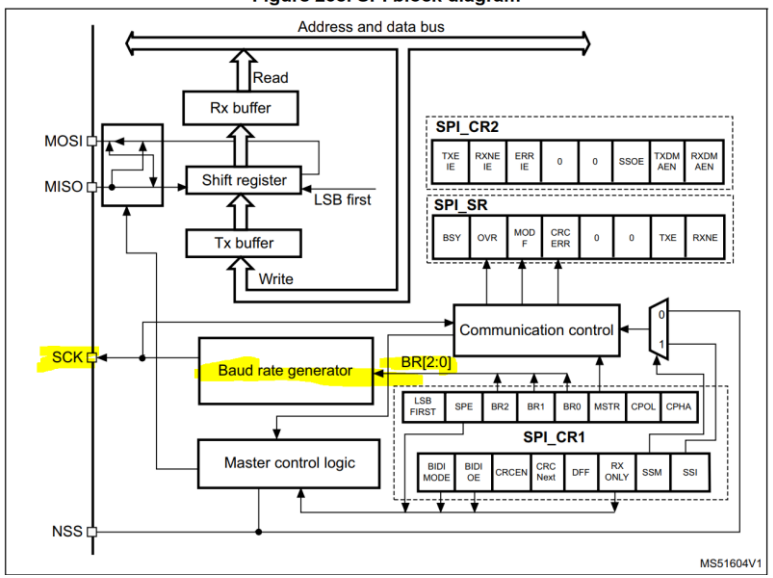
SCLK Calculation



Bits 5:3 **BR[2:0]:** Baud rate control

000: $f_{PCLK}/2$
001: $f_{PCLK}/4$
010: $f_{PCLK}/8$
011: $f_{PCLK}/16$
100: $f_{PCLK}/32$
101: $f_{PCLK}/64$
110: $f_{PCLK}/128$
111: $f_{PCLK}/256$

Figure 238. SPI block diagram



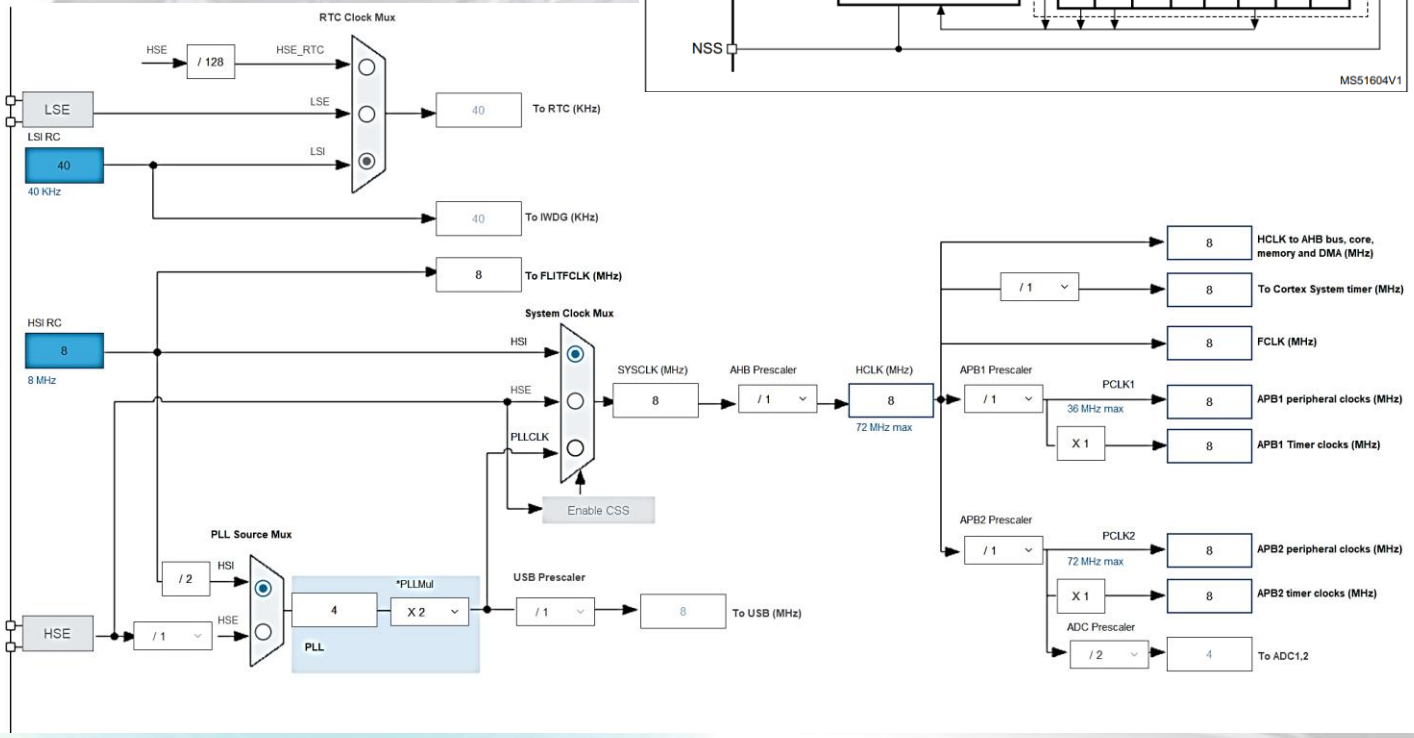
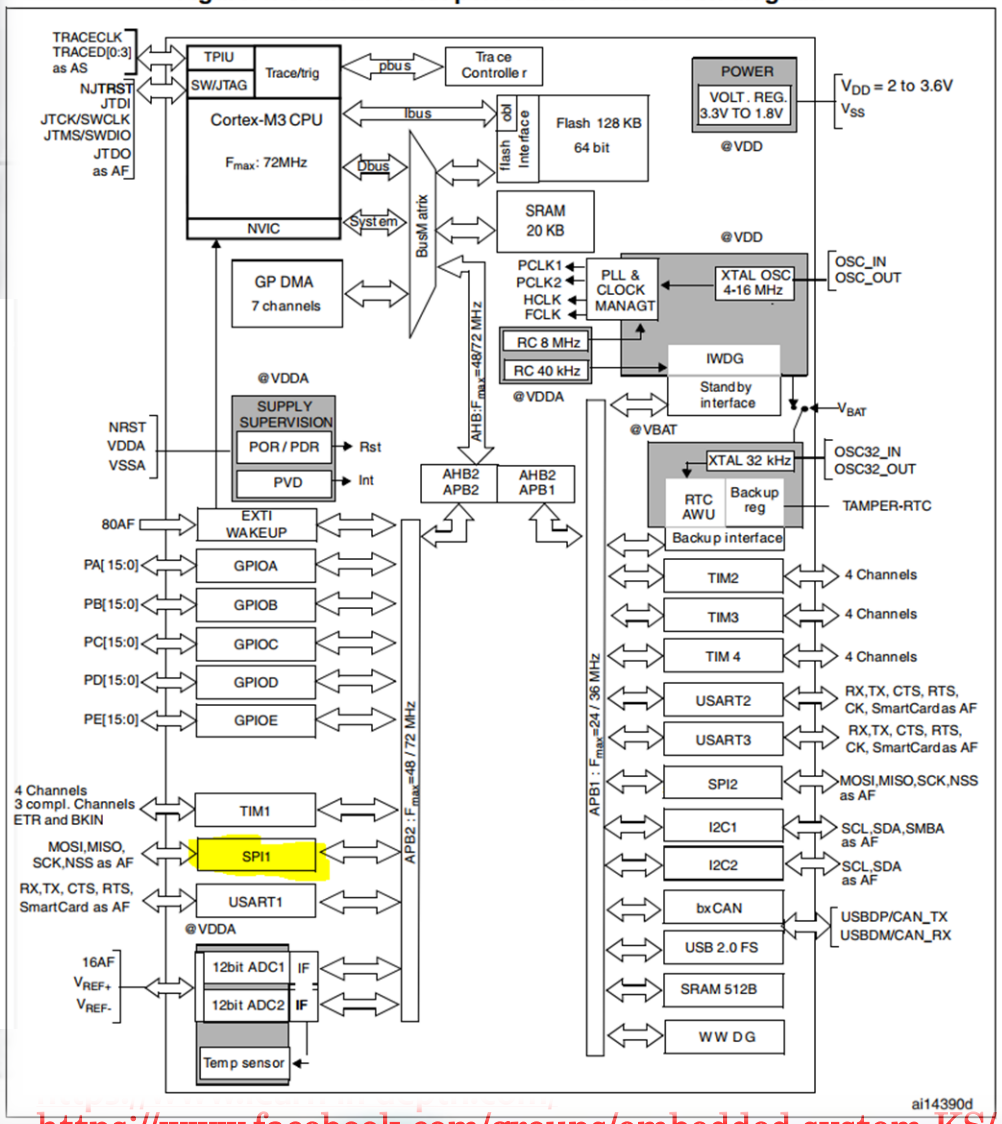
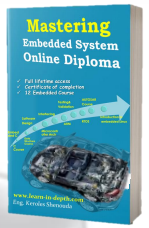


Figure 1. STM32F103xx performance line block diagram

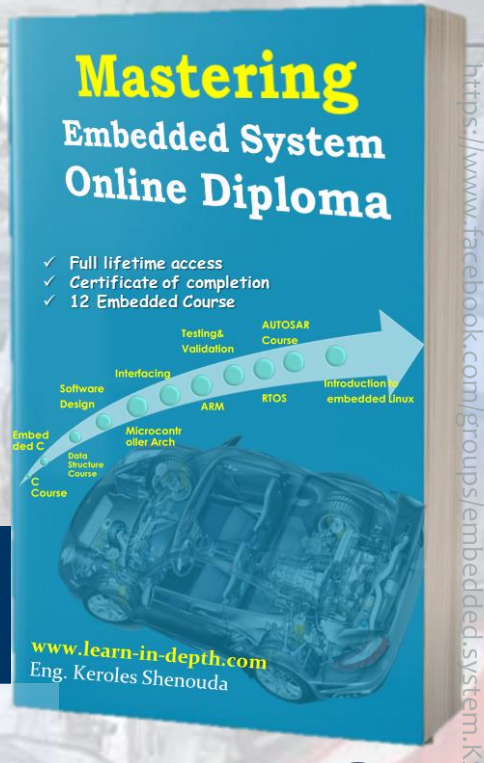


https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

#Be_professional_in_embedded_system

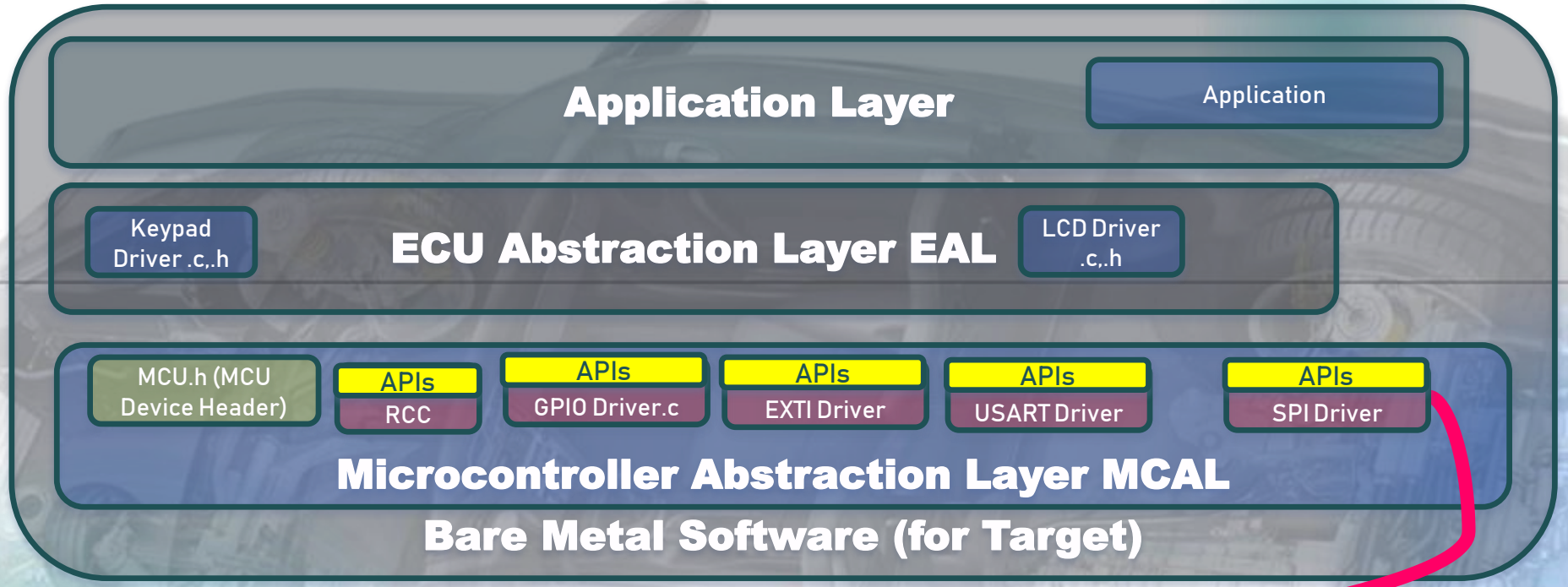


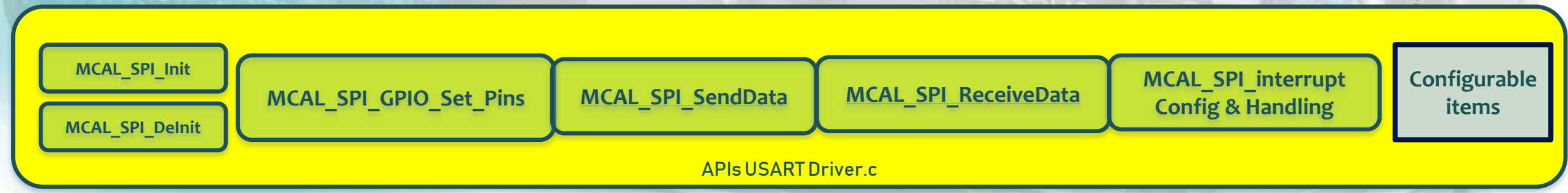
https://www.facebook.com/groups/embedded.system.KS/
Eng. Keroles Shenouda

Write SPI Deriver From Scratch

LEARN-IN-DEPTH
Be professional in embedded system

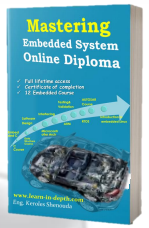
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/





FOLLOW US
Press here
28
Mastering Embedded System Online Diploma
#LEARN_IN DEPTH
#Be_professional_in_embedded_system
Eng. Keroles Shenouda
https://www.facebook.com/groups/embe

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Configurable items

Configurable items

IRQ Enable

SPI_Clock_Frequency
SPI Clock Speed

Slave Slect Management
SSM

Device_Mode
>Master
>Slave

Communication_Mode
>FD
>HF
>Simplex

Clock Polarity
CPOL
ClockPhase
CPHA

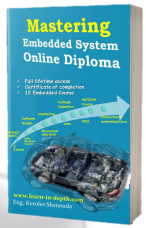
Frame_Format
MSB-first or LSB-first

DataSize
8 or 16 Bits

https://www.facebook.com/groups/embedded.system.KS/
Eng. Keroles Shenouda

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

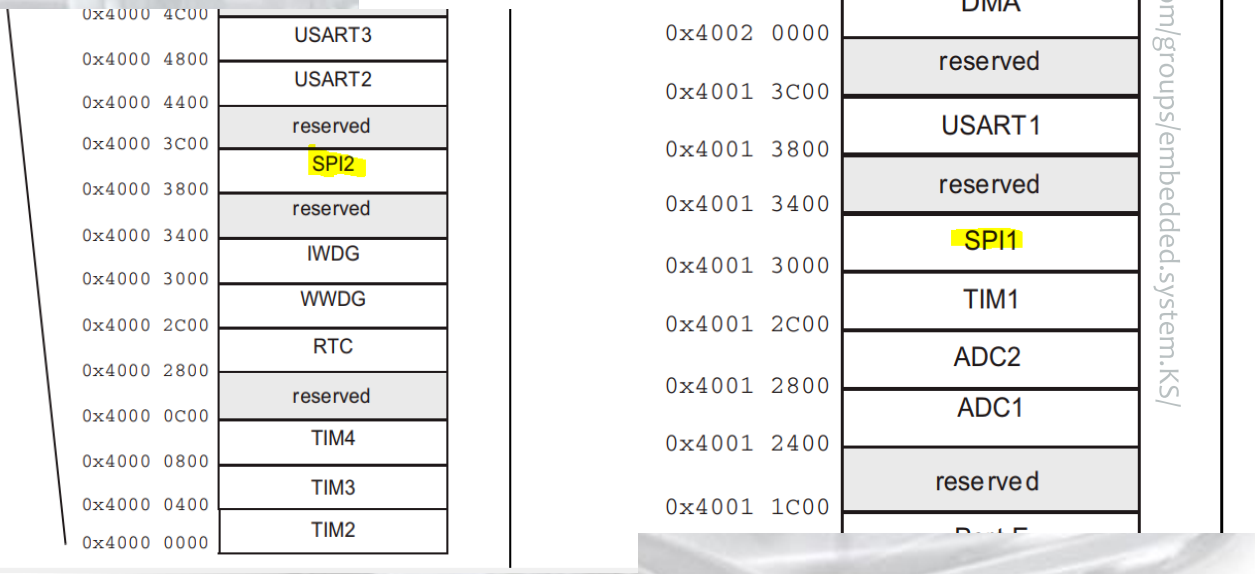
FOLLOW US
Press here
30



#LEARN_IN DEPTH
#Be_professional_in_embedded_system

1-add SPI registers on device header

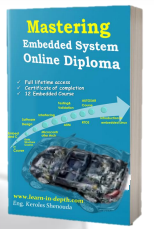
```
43 //------------------------------
44 //Base addresses for APB2 Peripherals
45 //------------------------------
46
47 //GPIO
48 //A,B fully included in LQFP48 Package
49 #define GPIOA_BASE            (Peripherals_BASE + 0x00010800UL)
50 #define GPIOB_BASE            (Peripherals_BASE + 0x00010C00UL)
51 //C,D Partial  included in LQFP48 Package
52 #define GPIOC_BASE            (Peripherals_BASE + 0x00011000UL)
53 #define GPIOD_BASE            (Peripherals_BASE + 0x00011400UL)
54 //EP not  included in LQFP48 Package
55 #define GPIOE_BASE            (Peripherals_BASE + 0x00011800UL)
56 //-------
57
58 #define AFIO_BASE             (Peripherals_BASE + 0x00010000UL)
59 #define EXTI_BASE             (Peripherals_BASE + 0x00010400UL)
60
61 #define USART1_BASE             (Peripherals_BASE + 0x00013800UL)
62
63
64 #define SPI1_BASE             (Peripherals_BASE + 0x00013000UL)
65
66
67 //------------------------------
68 //Base addresses for APB1 Peripherals
69 //------------------------------
70 #define USART2_BASE             (Peripherals_BASE + 0x00004400UL)
71 #define USART3_BASE             (Peripherals_BASE + 0x00004800UL)
72
73 #define SPI2_BASE             (Peripherals_BASE + 0x00003800UL)
```



https://www.facebook.com/groups/embedded.system.KS/
Eng. Keroles Shenouda

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

FOLLOW US
Press here
31



#LEARN_IN DEPTH
#Be_professional_in_embedded_system

add SPI info. on device header

```
162
163 typedef struct
164 {
165   volatile uint32_t SPI_CR1;
166   volatile uint32_t SPI_CR2;
167   volatile uint32_t SPI_SR;
168   volatile uint32_t SPI_DR;
169   volatile uint32_t SPI_CRCPR;
170   volatile uint32_t SPI_RXCRCR;
171   volatile uint32_t SPI_TXCRCR;
172   volatile uint32_t SPI_I2SCFGR;
173   volatile uint32_t SPI_I2SPR ;
174
175 } SPI_TypeDef;
```

```
203
204
205 #define SPI1                ((SPI_TypeDef *)SPI1_BASE)
206
207 //-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
```

Table 187. SPI register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SPI_CR1	Reserved																BIDIMODE	BIDIOE	CRCEN	CRCNEXT	DFF	RXONLY	SSM	SSI	*LSBFIRST*	SPE	BR [2:0]			MSTR	CPOL	CPHA
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	SPI_CR2	Reserved																								TXEIE	RXNEIE	ERRIE	Reserved			TXDMAEN	RXDMAEN
	Reset value																									0	0	0				0	0
0x08	SPI_SR	Reserved																								BSY	OVR	MODF	CRCERR	UDR	CHSIDE	TXE	RXNE
	Reset value																									0	0	0	0	0	0	1	0
0x0C	SPI_DR	Reserved																DR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	SPI_CRCPR	Reserved																CRCPOLY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
0x14	SPI_RXCRCR	Reserved																RxCRC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	SPI_TXCRCR	Reserved																TxCRC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	SPI_I2SCFGR	Reserved																				I2SMOD	I2SE	I2SCFG		PCMSYNC	Reserved	I2SSTD		CKPOL	DATLEN		CHLEN
	Reset value																					0	0	0	0	0		0	0	0	0	0	0
0x20	SPI_I2SPR	Reserved																								MCKOE	ODD	I2SDIV					
	Reset value																									0	0	0	0	0	0	1	0

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

FOLLOW US
Press here
32
Mastering Embedded System Online Diploma
#LEARN_IN DEPTH
#Be_professional_in_embedded_system
Eng. Keroles Shenouda
https://www.facebook.com/groups/embedded.system.KS/

add SPI info. on device header

8.3.7 APB2 peripheral clock enable register (RCC_APB2ENR)

Address: 0x18

Reset value: 0x0000 0000

Access: word, half-word and byte access

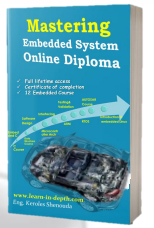
No wait states, except if the access occurs while an access to a peripheral in the APB2 domain is on going. In this case, wait states are inserted until the access to APB2 peripheral is finished.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART 1EN	Res.	SPI1 EN	TIM1 EN	ADC2 EN	ADC1 EN	Reserved		IOPE EN	IOPD EN	IOPC EN	IOPB EN	IOPA EN	Res.	AFIO EN
	rw		rw	rw	rw	rw			rw	rw	rw	rw	rw		rw

```
227
228 #define RCC_SPI1_CLK_EN()    ( RCC->APB2ENR |= (1<<12) )
229
```

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

FOLLOW US
Press here
33



#LEARN_IN DEPTH
#Be_professional_in_embedded_system

add SPI info. on device header

8.3.4 APB2 peripheral reset register (RCC_APB2RSTR)

Address offset: 0x0C

Reset value: 0x00000 0000

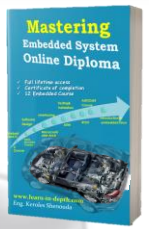
Access: no wait state, word, half-word and byte access

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	USART1 RST	Res.	SPI1 RST	TIM1 RST	ADC2 RST	ADC1 RST	Reserved		IOPE RST	IOPD RST	IOPC RST	IOPB RST	IOPA RST	Res.	AFIO RST
	rw		rw	rw	rw	rw			rw	rw	rw	rw	rw		rw

```
235
236 #define RCC_SPI1_Reset()     ( RCC->APB1RSTR |= (1<<12) )
237
238
```

https://www.facebook.com/groups/embedded.system.KS/
Eng. Keroles Shenouda

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



add SPI info. on device header

```
269 #define      SPI1_IRQ                35
270 #define      SPI2_IRQ                36
271
272
```

Table 61. Vector table for connectivity line devices (continued)

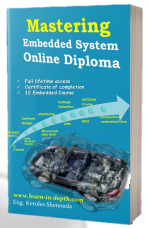
Position	Priority	Type of priority	Acronym	Description	Address
34	41	settable	I2C2_ER	I^2C2 error interrupt	0x0000_00C8
35	42	settable	SPI1	SPI1 global interrupt	0x0000_00CC
36	43	settable	SPI2	SPI2 global interrupt	0x0000_00D0

```
2
3 #define NVIC_IRQ35_SPI1_Enable      (NVIC_ISER1 |= 1<<( SPI1_IRQ - 32 )) //NVIC_ISER1 35-32
4 #define NVIC_IRQ35_SPI2_Enable      (NVIC_ISER1 |= 1<<( SPI2_IRQ - 32 )) //NVIC_ISER1 36-32
5
6 #define NVIC_IRQ35_SPI1_Disable     (NVIC_ICER1 |= 1<<( SPI1_IRQ- 32 )) //NVIC_ISER1 35-32
7 #define NVIC_IRQ35_SPI2_Disable     (NVIC_ICER1 |= 1<<( SPI2_IRQ- 32 )) //NVIC_ISER1 36-32
8
```

https://www.facebook.com/groups/embedded.system.KS/ Eng. Keroles Shenouda

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

FOLLOW US
Press here
35



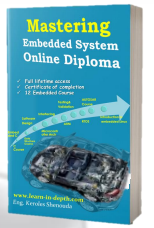
#LEARN_IN DEPTH
#Be_professional_in_embedded_system

2- add Configuration Structure

```
22 //Configuration structure
23 typedef struct
24 {
25
26     uint8_t                    Device_Mode;              //  Specifies the SPI operating mode  @ref SPI_Device_Mode
27
28     uint8_t                    Communication_Mode ;      // Specifies the SPI bidirectional mode state  @ref SPI_Communication_Mode
29
30     uint8_t                    Frame_Format;             // Specifies LSB or MSB    @ref SPI_Frame_Format
31
32     uint8_t                    DataSize ;                //  @ref SPI_DataSize
33
34     uint8_t                    CLKPolarity ;             // @ref SPI_CLKPolarity
35
36     uint8_t                    CLKPhase ;                //@ref SPI_CLKPhase
37
38     uint8_t                    NSS;                       //Specifies whether the NSS signal is managed by
39                                                          //hardware (NSS pin) or by software using the SSI bitenable or disable UART IRQ TX/RX
40                                                          //@ref SPI_NSS
41
42     uint8_t                    SPI_Clock_Frequency;      /*Specifies the Baud Rate prescaler value which will be
43                                                          used to configure the transmit and receive SCK clock.
44                                                          This parameter can be a value of @ref SPI_Clock_Frequency
45                                                          @note The communication clock is derived from the master
46                                                          clock. The slave clock does not need to be set.*/
47
48     void(* P_IRQ_CallBack)(void) ;                  //Set the C Function() which will be called once the IRQ  Happen
49
50 }SPI_Config;
51
```

https://www.facebook.com/groups/embedded.system.KS/
Eng. Keroles Shenouda

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Reference Macros

Bit 2 **MSTR:** Master selection

0: Slave configuration

1: Master configuration

25.5.1 SPI control register 1 (SPI_CR1) (not used in I²S mode)

Address offset: 0x00

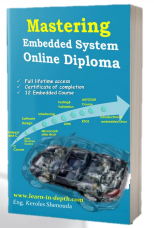
Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BIDI MODE	BIDI OE	CRC EN	CRC NEXT	DFF	RX ONLY	SSM	SSI	*LSB FIRST*	SPE	BR [2:0]			MSTR	CPOL	CPHA
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

```
60
61 //@ref SPI_Device_Mode
62 #define SPI_Device_Mode_SLAVE                (0x00000000U)
63 #define SPI_Device_Mode_MASTER               (0x1U<<2) //MSTR :1: Master configuration
64
```

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

FOLLOW US
Press here
37



#LEARN_IN DEPTH
#Be_professional_in_embedded_system

Reference Macros

Bit 15 **BIDIMODE:** Bidirectional data mode enable
0: 2-line unidirectional data mode selected
1: 1-line bidirectional data mode selected
Note: This bit is not used in I^2S mode

Bit 14 **BIDIOE:** Output enable in bidirectional mode
This bit combined with the BIDImode bit selects the direction of transfer in bidirectional mode
0: Output disabled (receive-only mode)
1: Output enabled (transmit-only mode)
Note: This bit is not used in I^2S mode.
In master mode, the MOSI pin is used while the MISO pin is used in slave mode.

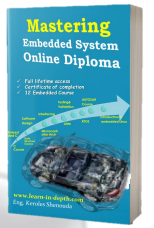
Bit 10 **RXONLY:** Receive only
This bit combined with the BIDImode bit selects the direction of transfer in 2-line unidirectional mode. This bit is also useful in a multislave system in which this particular slave is not accessed, the output from the accessed slave is not corrupted.
0: Full duplex (Transmit and receive)
1: Output disabled (Receive-only mode)

```
64
65 //@ref SPI_Communication_Mode
66 #define SPI_DIRECTION_2LINES                    (0x00000000U)
67 #define SPI_DIRECTION_2LINES_RXONLY             (0x1U<<10) //Bit 10 RXONLY: Receive only
68 #define SPI_DIRECTION_1LINE_receive_only        ((0x1U<<15) ) //Bit 15 BIDIMODE: Bidirectional data mode enable
69 #define SPI_DIRECTION_1LINE_transmit_only       ((0x1U<<15) | (0x1U<<14)) //Bit 15 BIDIMODE: Bidirectional data mode enable &  Bit 14
70
71
72
```

https://www.facebook.com/groups/embedded.system.KS/
Eng. Keroles Shenouda

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

FOLLOW US
Press here
38



#LEARN_IN DEPTH
#Be_professional_in_embedded_system

Reference Macros

Bit 7 **LSBFIRST:** Frame format

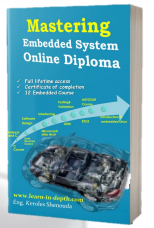
0: MSB transmitted first

1: LSB transmitted first

```
71
72 //@ref SPI_Frame_Format
73 #define SPI_Frame_Format_MSB_transmitted_first          (0x00000000U)  //Bit 7 LSBFIRST: Frame format
74 #define SPI_Frame_Format_LSB_transmitted_first          (0x1U<<7)       //Bit 7 LSBFIRST: Frame format
75
76
```

Eng. Keroles Shenouda
https://www.facebook.com/groups/embedded.system.KS/

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Reference Macros

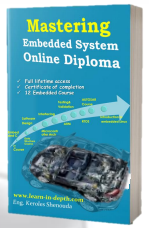
Bit 11 **DFF:** Data frame format

0: 8-bit data frame format is selected for transmission/reception

1: 16-bit data frame format is selected for transmission/reception

```
75
76
77 //@ref SPI_DataSize
78 #define SPI_DataSize_8BIT                (0x00000000U)
79 #define SPI_DataSize_16BIT               (0x1U<<11)    //Bit 11 DFF: Data frame format
80
81
```

https://www.facebook.com/groups/embedded.system.KS/
Eng. Keroles Shenouda



Reference Macros

Serial peripheral interface (SPI)

Bit1 **CPOL:** Clock polarity
0: CK to 0 when idle
1: CK to 1 when idle
Note: This bit should not be changed when communication is ongoing. It is not used in I^2S mode

Bit 0 **CPHA:** Clock phase
0: The first clock transition is the first data capture edge
1: The second clock transition is the first data capture edge
Note: This bit should not be changed when communication is ongoing. It is not used in I^2S mode

```
/** @defgroup SPI_Clock_Polarity SPI Clock Polarity
 * @{
 */
#define SPI_CLKPolarity_LOW_when_idle                (0x00000000U)
#define SPI_CLKPolarity_HIGH_when_idle               (0x1U<<1) //Bit1 CPOL: Clock polarity
/**
  * @}
  */
/** @defgroup SPI_Clock_Phase SPI Clock Phase
  * @{
  */
#define SPI_Clock_Phase_1EDGE_first_data_capture_edge                (0x00000000U)
#define SPI_Clock_Phase_2EDGE_first_data_capture_edge                (0x1U<<0) //Bit 0 CPHA: Clock phase

```

Eng. Keroles Shenouda https://www.facebook.com/groups/embedded.system.KS/

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

FOLLOW US
Press here
41
Mastering Embedded System Online Diploma
#LEARN_IN DEPTH
#Be_professional_in_embedded_system

Reference Macros

Bit 9 **SSM:** Software slave management

When the SSM bit is set, the NSS pin input is replaced with the value from the SSI bit.

0: Software slave management disabled

1: Software slave management enabled

Note: This bit is not used in I^2S mode

Bit 8 **SSI:** Internal slave select

This bit has an effect only when the SSM bit is set. The value of this bit is forced onto the NSS pin and the IO value of the NSS pin is ignored.

Note: This bit is not used in I^2S mode

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **SSOE:** SS output enable

0: SS output is disabled in master mode and the cell can work in multimaster configuration

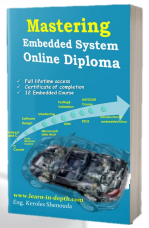
1: SS output is enabled in master mode and when the cell is enabled. The cell cannot work in a multimaster environment.

Note: This bit is not used in I^2S mode

```
97  //@ref SPI_NSS  SPI Slave Select Management
98  #define SPI_NSS_SLAVE_SOFTWARE                  (0x1U<<9)  //Bit 9 SSM: Software slave management
99  #define SPI_NSS_HARD_INPUT                      (0x00000000U)
100 #define SPI_NSS_Master_HARD_OUTPUT              (0x1U << 2) //Bit 2 SPI_CR2_SSOE: SS output enable
101
```

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/
Eng. Keroles Shenouda https://www.facebook.com/groups/embedded.system.KS/

FOLLOW US
Press here
42



#LEARN_IN DEPTH
#Be_professional_in_embedded_system
Eng. Keroles Shenouda https://www.face

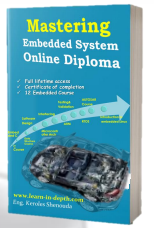
Reference Macros

Bits 5:3 **BR[2:0]: Baud rate control**

000: $f_{PCLK}/2$
001: $f_{PCLK}/4$
010: $f_{PCLK}/8$
011: $f_{PCLK}/16$
100: $f_{PCLK}/32$
101: $f_{PCLK}/64$
110: $f_{PCLK}/128$
111: $f_{PCLK}/256$

```
102 //@ref SPI_BAUDRATEPRESCALER
103 #define SPI_BAUDRATEPRESCALER_2          (0x00000000U)
104 #define SPI_BAUDRATEPRESCALER_4          (0b001U << 3 )
105 #define SPI_BAUDRATEPRESCALER_8          (0b010U << 3)
106 #define SPI_BAUDRATEPRESCALER_16         (0b011U << 3)
107 #define SPI_BAUDRATEPRESCALER_32         (0b100U << 3)
108 #define SPI_BAUDRATEPRESCALER_64         (0b101U << 3)
109 #define SPI_BAUDRATEPRESCALER_128        (0b110U << 3)
110 #define SPI_BAUDRATEPRESCALER_256        (0b111U << 3)
111
```

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Reference Macros

25.5.2 SPI control register 2 (SPI_CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TXEIE	RXNEIE	ERRIE	Res.	Res.	*SSOE*	TXDMAEN	RXDMAEN
								rw	rw	rw			rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **TXEIE:** Tx buffer empty interrupt enable
0: TXE interrupt masked
1: TXE interrupt not masked. Used to generate an interrupt request when the TXE flag is set.

Bit 6 **RXNEIE:** RX buffer not empty interrupt enable
0: RXNE interrupt masked
1: RXNE interrupt not masked. Used to generate an interrupt request when the RXNE flag is set.

Bit 5 **ERRIE:** Error interrupt enable
This bit controls the generation of an interrupt when an error condition occurs (CRCERR, OVR, MODF in SPI mode and UDR, OVR in I^2S mode).
0: Error interrupt is masked
1: Error interrupt is enabled

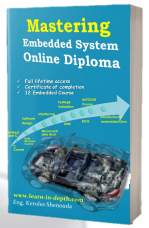
```
114
115 //@ref UART_IRQ_Enable_define
116 #define SPI_IRQ_Enable_NONE                (uint32_t)(0)
117 #define SPI_IRQ_Enable_TXEIE               (uint32_t) (1<<7) //SPI_CR2 Bit 7 TXEIE: Tx buffer empty interrupt enable
118 #define SPI_IRQ_Enable_RXNEIE              ((uint32_t)(1<<6)) //SPI_CR2 Bit 6 RXNEIE: RX buffer not empty interrupt enable
119 #define SPI_IRQ_Enable_ERRIE               (uint32_t) (1<<5) //SPI_CR2 Bit 5 ERRIE: Error interrupt enable
120
121     /*
```

https://www.facebook.com/groups/embedded.system.KS/ Eng. Keroles Shenouda

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

Init

FOLLOW US
Press here
44



#LEARN_IN DEPTH
#Be_professional_in_embedded_system

25.3.2 Configuring the SPI in slave mode

In the slave configuration, the serial clock is received on the SCK pin from the master device. The value set in the BR[2:0] bits in the SPI_CR1 register, does not affect the data transfer rate.

Note: *It is recommended to enable the SPI slave before the master sends the clock. If not, undesired data transmission might occur. The data register of the slave needs to be ready before the first edge of the communication clock or before the end of the ongoing communication. It is mandatory to have the polarity of the communication clock set to the steady state value before the slave and the master are enabled.*

Follow the procedure below to configure the SPI in slave mode:

Procedure

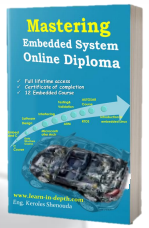
1. Set the DFF bit to define 8- or 16-bit data frame format
2. Select the CPOL and CPHA bits to define one of the four relationships between the data transfer and the serial clock (see *Figure 240*). For correct data transfer, the CPOL and CPHA bits must be configured in the same way in the slave device and the master device.
3. The frame format (MSB-first or LSB-first depending on the value of the LSBFIRST bit in the SPI_CR1 register) must be the same as the master device.
4. In Hardware mode (refer to *Slave select (NSS) pin management*), the NSS pin must be connected to a low level signal during the complete byte transmit sequence. In NSS software mode, set the SSM bit and clear the SSI bit in the SPI_CR1 register.
5. Clear the MSTR bit and set the SPE bit (both in the SPI_CR1 register) to assign the pins to alternate functions.

In this configuration the MOSI pin is a data input and the MISO pin is a data output.

Eng. Keroles Shenouda
https://www.facebook.com/groups/embedded.system.KS/

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

FOLLOW US
Press here
45



#LEARN_IN DEPTH
#Be_professional_in_embedded_system

GPIO

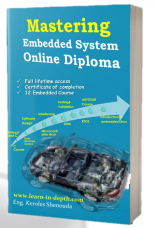
G3	M3	14	H3	20	29	11	PA4	I/O	-	PA4	SPI1_NSS(9)/ USART2_CK(9)/ ADC12_IN4
H3	K4	15	F4	21	30	12	PA5	I/O	-	PA5	SPI1_SCK(9)/ ADC12_IN5
J3	L4	16	G4	22	31	13	PA6	I/O	-	PA6	SPI1_MISO(9)/ ADC12_IN6/ TIM3_CH1(9)
K3	M4	17	H4	23	32	14	PA7	I/O	-	PA7	SPI1_MOSI(9)/ ADC12_IN7/ TIM3_CH2(9)
K8	L12	25	H8	33	51	-	PB12	I/O	FT	PB12	SPI2_NSS/ I2C2_SMBAI/ USART3_CK(9)/ TIM1_BKIN(9)	-
J8	K12	26	G8	34	52	-	PB13	I/O	FT	PB13	SPI2_SCK/ USART3_CTS(9)/ TIM1_CH1N (9)	-
H8	K11	27	F8	35	53	-	PB14	I/O	FT	PB14	SPI2_MISO/ USART3_RTS(9) TIM1_CH2N (9)	-
G8	K10	28	F7	36	54	-	PB15	I/O	FT	PB15	SPI2_MOSI/ TIM1_CH3N(9)	-

Table 25. SPI

SPI pinout	Configuration	GPIO configuration
SPIx_SCK	Master	Alternate function push-pull
	Slave	Input floating
SPIx_MOSI	Full duplex / master	Alternate function push-pull
	Full duplex / slave	Input floating / Input pull-up
	Simplex bidirectional data wire / master	Alternate function push-pull
	Simplex bidirectional data wire/ slave	Not used. Can be used as a GPIO
SPIx_MISO	Full duplex / master	Input floating / Input pull-up
	Full duplex / slave (point to point)	Alternate function push-pull
	Full duplex / slave (multi-slave)	Alternate function open drain
	Simplex bidirectional data wire / master	Not used. Can be used as a GPIO
	Simplex bidirectional data wire/ slave (point to point)	Alternate function push-pull
	Simplex bidirectional data wire/ slave (multi-slave)	Alternate function open drain
SPIx_NSS	Hardware master /slave	Input floating/ Input pull-up / Input pull-down
	Hardware master/ NSS output enabled	Alternate function push-pull
	Software	Not used. Can be used as a GPIO

Eng. Keroles Shenouda https://www.facebook.com/groups/embedded.system.KS/

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



Eng. Keroles Shenouda
https://www.facebook.com/groups/embedded.system.KS/

25.3.3 Configuring the SPI in master mode

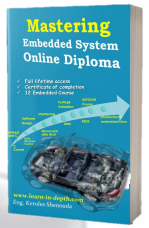
In the master configuration, the serial clock is generated on the SCK pin.

Procedure

1. Select the BR[2:0] bits to define the serial clock baud rate (see SPI_CR1 register).
2. Select the CPOL and CPHA bits to define one of the four relationships between the data transfer and the serial clock (see *Figure 240*).
3. Set the DFF bit to define 8- or 16-bit data frame format
4. Configure the LSBFIRST bit in the SPI_CR1 register to define the frame format.
5. If the NSS pin is required in input mode, in hardware mode, connect the NSS pin to a high-level signal during the complete byte transmit sequence. In NSS software mode, set the SSM and SSI bits in the SPI_CR1 register. If the NSS pin is required in output mode, the SSOE bit only should be set.
6. The MSTR and SPE bits must be set (they remain set only if the NSS pin is connected to a high-level signal).

In this configuration the MOSI pin is a data output and the MISO pin is a data input.

FOLLOW US
Press here
47

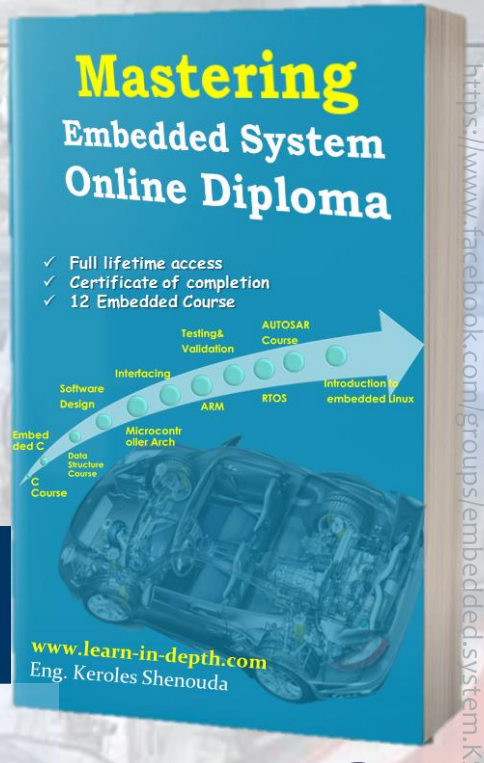


#LEARN_IN DEPTH

#Be_professional_in_embedded_system

Eng. Keroles Shenouda

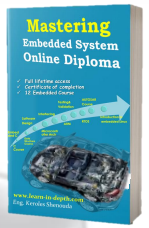
LAB1: SPI Debug/Analyze SPI Master



https://www.facebook.com/groups/embedded.system.KS/

LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

#Be_professional_in_embedded_system

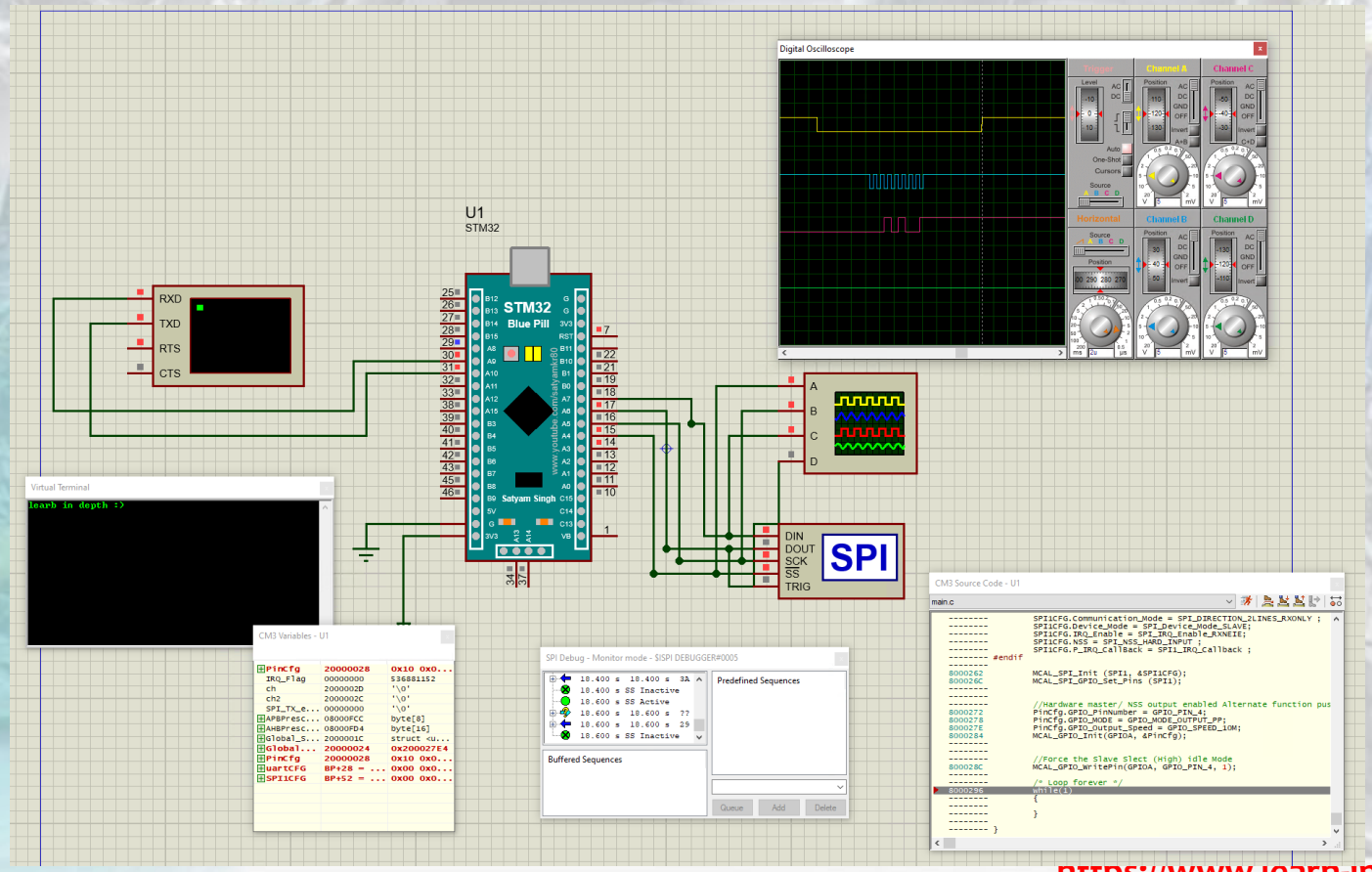
LAB1

```
64
65
66  }
67
68  void UART_IRQ_Callback (void)
69  {
70  #ifdef MCU_Act_As_Master
71    MCAL_UART_ReceiveData(USART1, &ch, disable);
72    MCAL_UART_SendData(USART1, &ch, enable);
73    //Send to SPI
74    MCAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, 0);
75    MCAL_SPI_TX_RX  (SPI1, &ch,Pollingenable);
76    MCAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, 1);
77  #endif
78
79
80  }
```

https://www.learn-in-depth.com/ Eng. Keroles

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

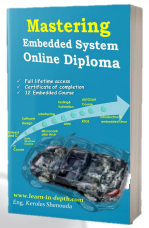
lab1



https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

FOLLOW US
Press here
49
#LEARN_IN DEPTH
#Be_professional_in embedded_system
Eng. Keroles Shenouda
https://www.facebook.com/groups/embedded.system.KS/

FOLLOW US
Press here
50



#LEARN_IN DEPTH

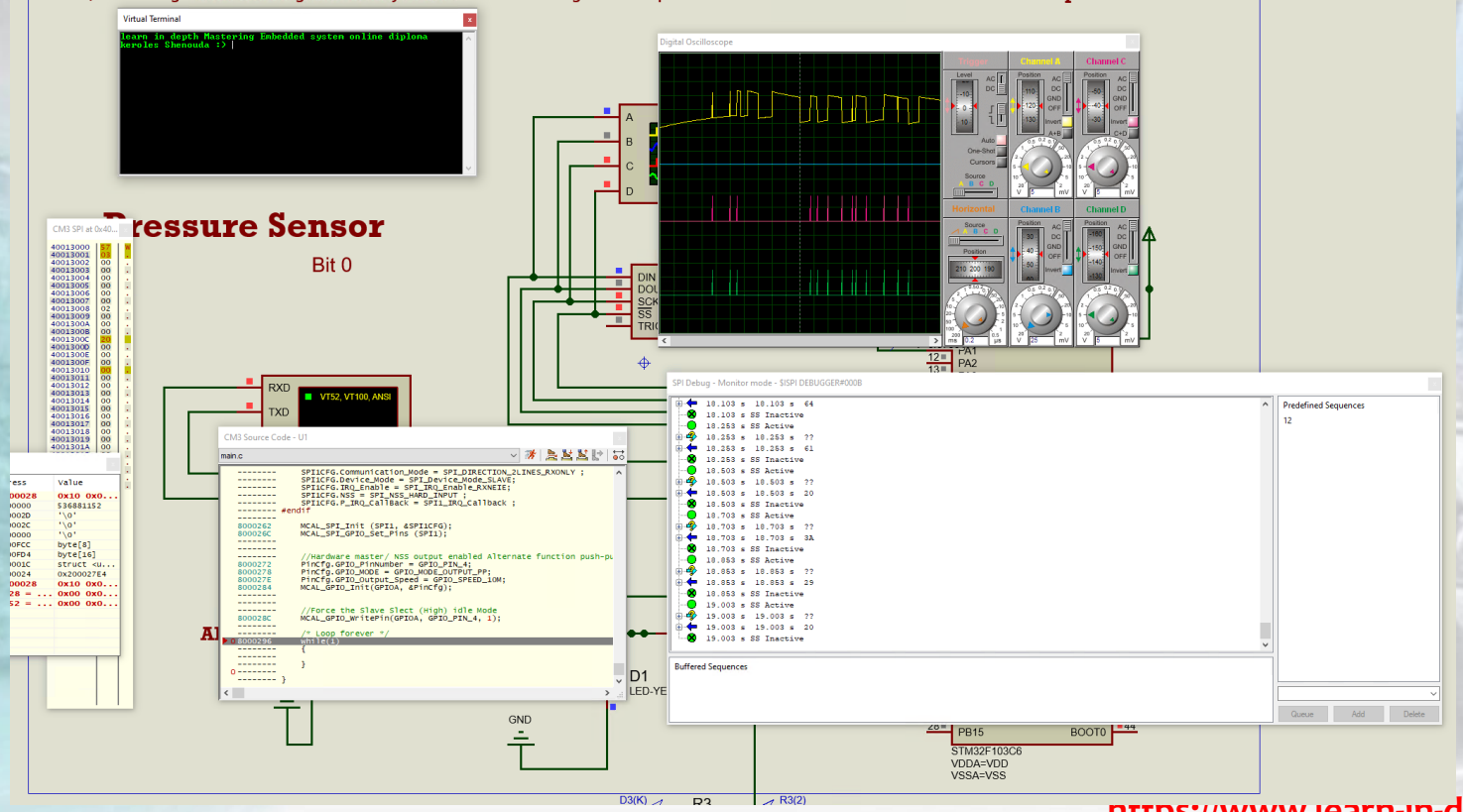
#Be_professional_in_embedded_system

Eng. Keroles Shenouda
https://www.facebook.com/groups/embedded.system.KS/

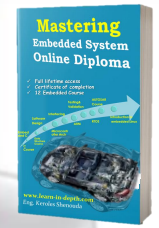
Write your OWN Linker & Startup & Makefile
write your algorithm according to:
SYSML/UML Design Flows and Diagrams which you are created according to the Requirements

Mastering Embedded System Online Diploma (KS)

www.learn-in-depth.com



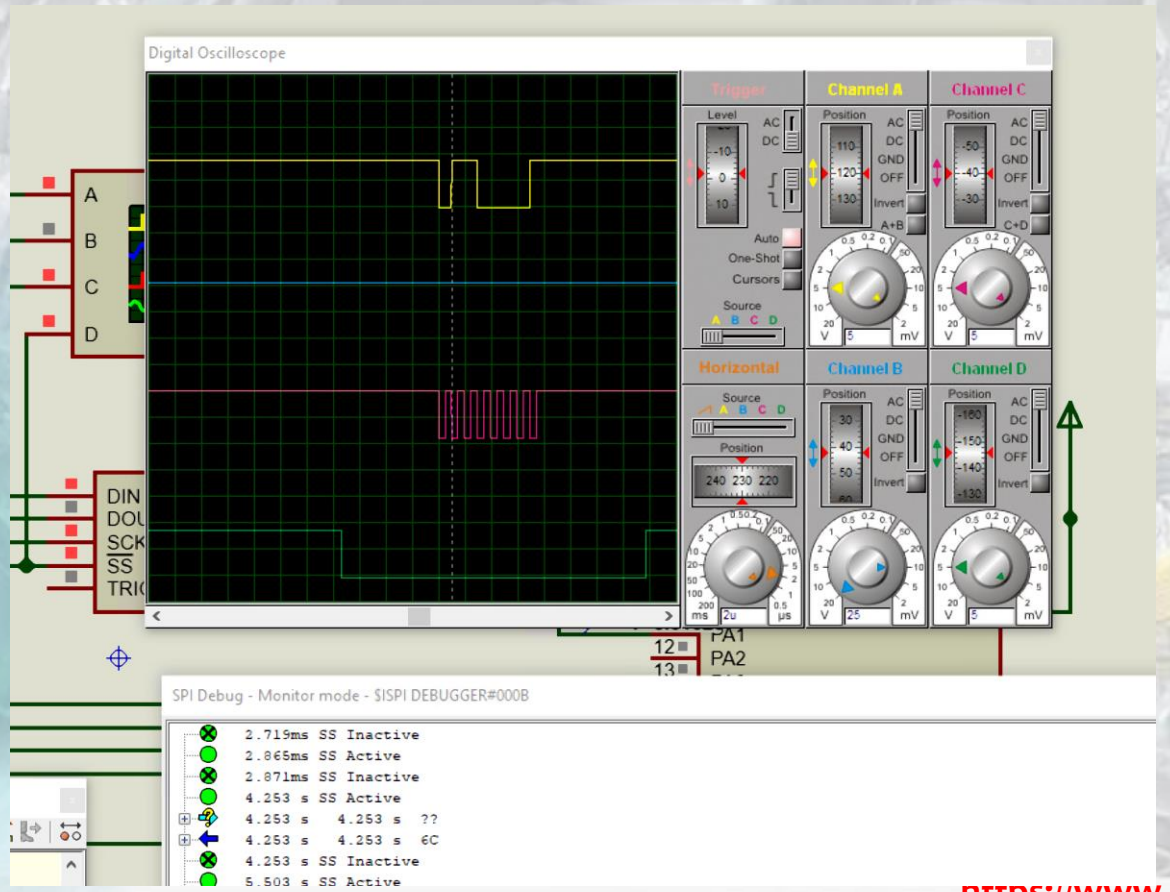
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



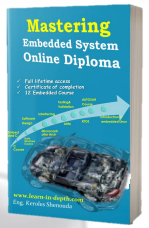
#LEARN_IN DEPTH

#Be_professional_in_embedded_system

Eng. Keroles Shenouda
https://www.facebook.com/groups/embedded.system.KS/

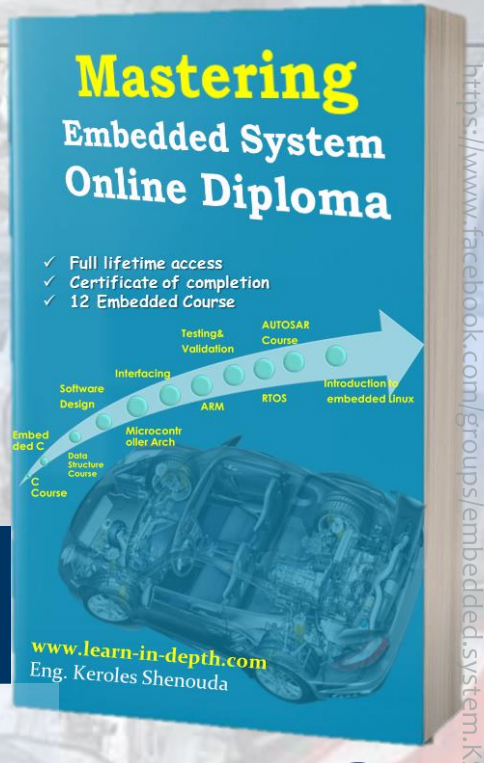


nttps://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



#LEARN_IN DEPTH

#Be_professional_in_embedded_system



https://www.facebook.com/groups/embedded.system.KS/
Eng. Keroles Shenouda

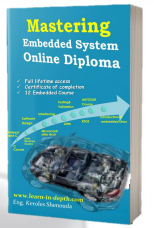
Terminal1 <----> USART1 : MCU1 : (SPI1 Master) ---> (SPI2 Slave) :MCU2: USART2 ---> Terminal2

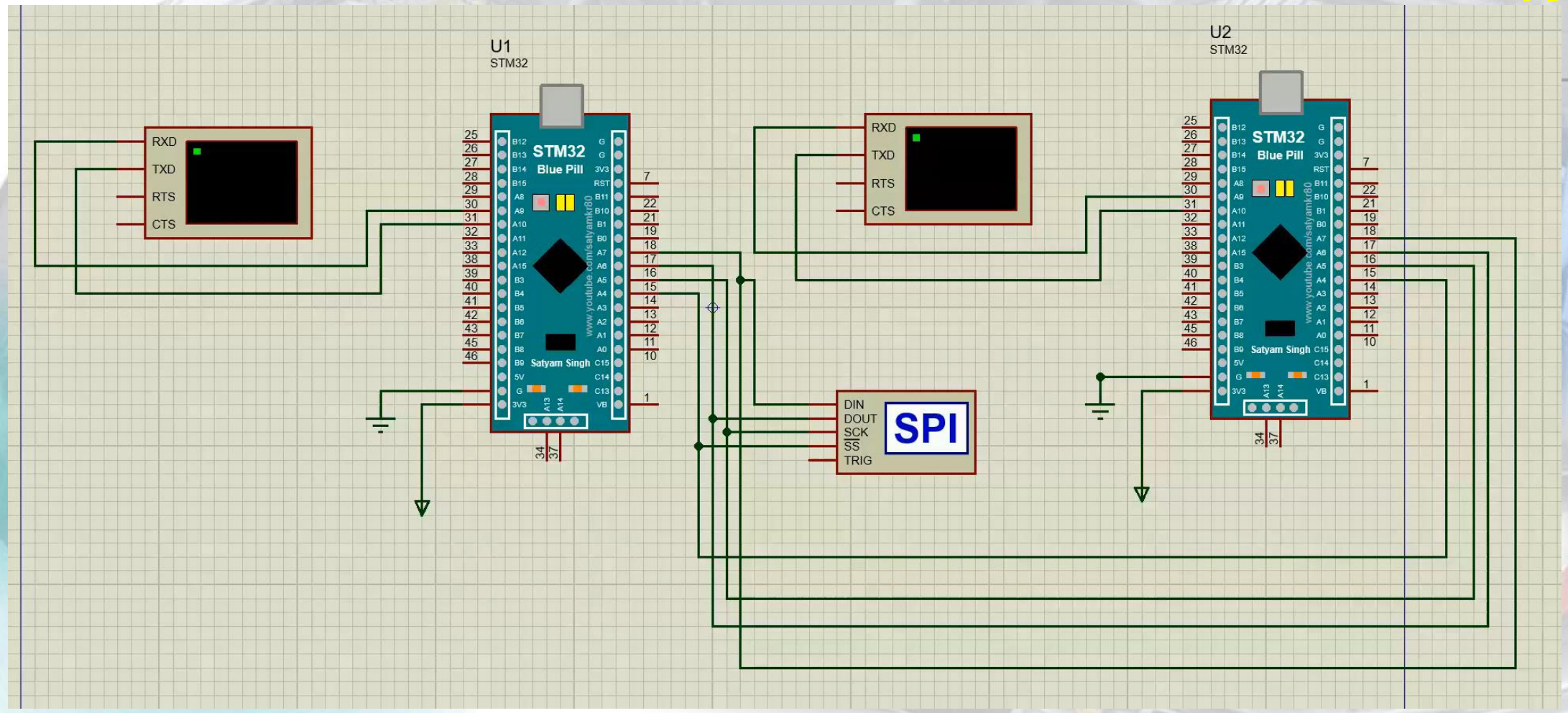
LAB2

LEARN-IN-DEPTH
Be professional in embedded system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

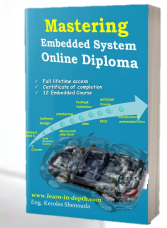
Terminal1 <----> USART1 : MCU1 : (SPI1 Master) ---> (SPI2 Slave) :MCU2: USART2 ---> Terminal2





Eng. Keroles Shenouda
https://www.facebook.com/groups/embedded.system.KS/

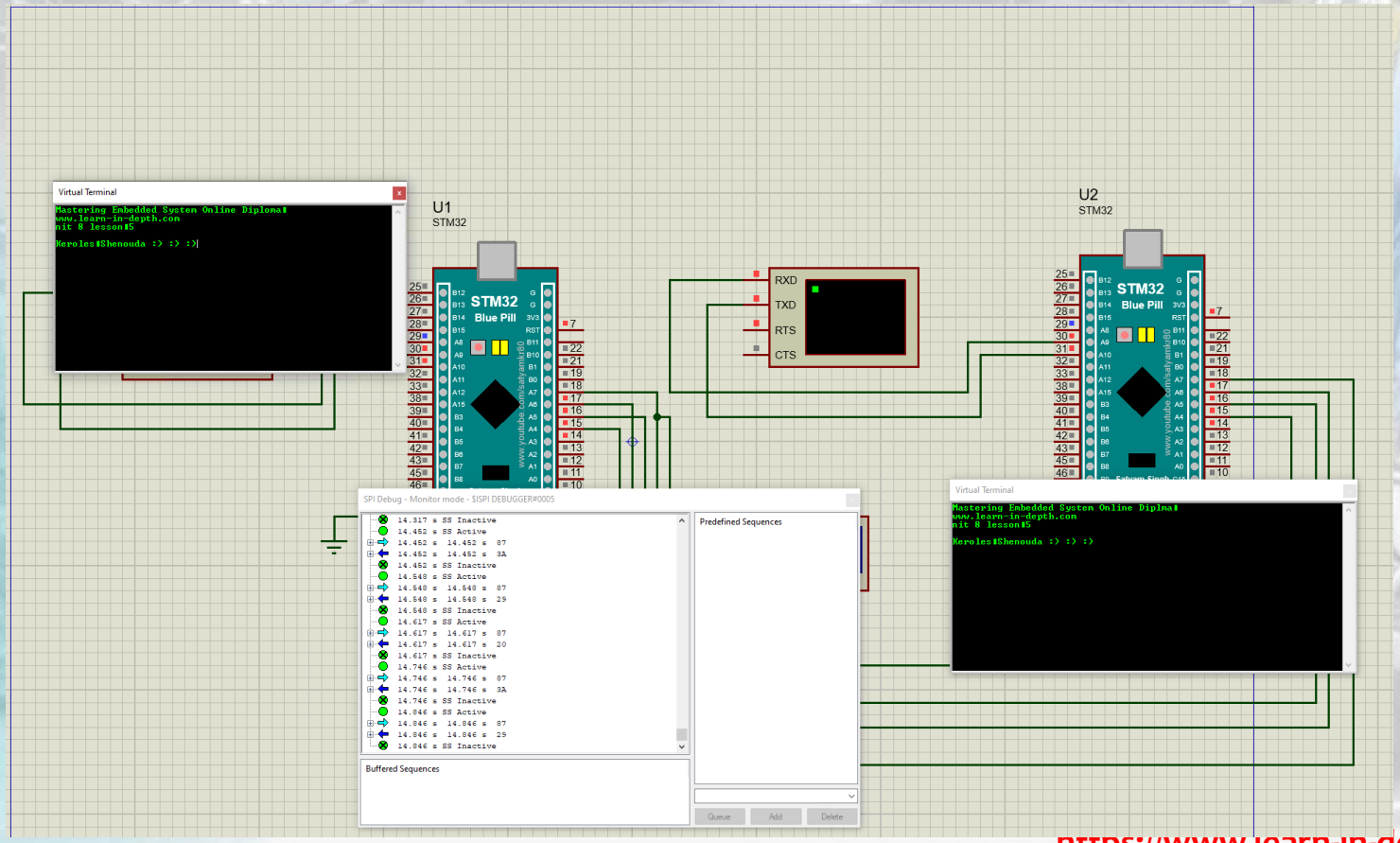
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



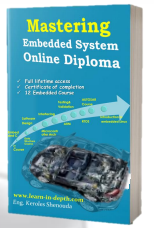
#LEARN_IN DEPTH

#Be_professional_in_embedded_system

Eng. Keroles Shenouda
https://www.facebook.com/groups/embedded.system.KS/



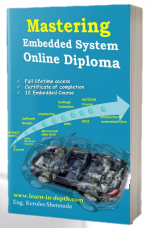
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



References

- https://docs.google.com/viewer?a=v&pid=sites&srcid=ZmtlLnV0bS5teXxyaWR6dWFuLXMtd2Vic2l0ZXxneDo2ODU0NzlkM2JkOTg4MjRk
- http://www.avrprojects.net/index.php/avr-projects/sensors/38-humidity-and-temperature-sensor-dht11?showall=&start=1
- http://www.cse.wustl.edu/~lu/cse467s/slides/dsp.pdf
- http://www.avr-tutorials.com/
- **Microprocessor: ATmega32 (SEE3223-10)**
  http://ridzuan.fke.utm.my/microprocessor-atmega32-see3223-10
- **http://circuitdigest.com/article/what-is-the-difference-between-microprocessor-and-microcontroller**
- **http://cs4hs.cs.pub.ro/wiki/roboticsisfun/chapter2/ch2_7_programming_a_microcontroller**
- Embedded Systems with ARM **Cortex-M Microcontrollers in** Assembly Language and C Dr. Yifeng Zhu Third edition June 2018

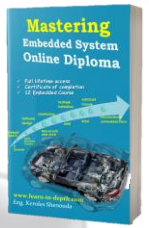
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



References

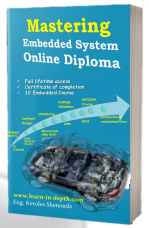
- http://techdifferences.com/difference-between-interrupt-and-polling-in-os.html
- http://www.bogotobogo.com/Embedded/hardware_interrupt_software_interrupt_latency_irq_vs_fiq.php
- Preventing Interrupt Overload Presented by Jiyong Park Seoul National University, Korea 2005. 2. 22. John Regehr, Usit Duogsaa, School of Computing, University.
- First Steps Embedded Systems Byte Craft Limited reference
- COMPUTER ORGANIZATION AND ARCHITECTURE DESIGNING FOR PERFORMANCE EIGHTH EDITION **William Stallings**
- **Getting Started with the Tiva™ TM4C123G LaunchPad Workshop**

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/



References

- Tiva™ TM4C123GH6PM Microcontroller DATA SHEET
- Interrupts and Exceptions COMS W6998 Spring 2010
- THE **AVR MICROCONTROLLER**. AND EMBEDDED SYSTEMS Using Assembly and C. Muhammad Ali **Mazidi**.
- http://embedded-lab.com/blog/tinkering-ti-msp430f5529/27/



References

- https://docs.google.com/viewer?a=v&pid=sites&srcid=ZmtlLnV0bS5teXxyaWR6dWFuLXMtd2Vic2l0ZXxneDo2ODU0NzlkM2JkOTg4MjRk
- http://www.avrprojects.net/index.php/avr-projects/sensors/38-humidity-and-temperature-sensor-dht11?showall=&start=1
- http://www.cse.wustl.edu/~lu/cse467s/slides/dsp.pdf
- http://www.avr-tutorials.com/
- Microprocessor: ATmega32 (SEE3223-10) http://ridzuan.fke.utm.my/microprocessor-atmega32-see3223-10
- http://circuitdigest.com/article/what-is-the-difference-between-microprocessor-and-microcontroller
- AVR Microcontroller and Embedded Systems: Using Assembly and C (Pearson Custom Electronics Technology) 1st Edition https://www.amazon.com/AVR-Microcontroller-Embedded-Systems-Electronics/dp/0138003319

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

FOLLOW US
Press here
60

#LEARN_IN DEPTH

#Be_professional_in embedded_system

https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/