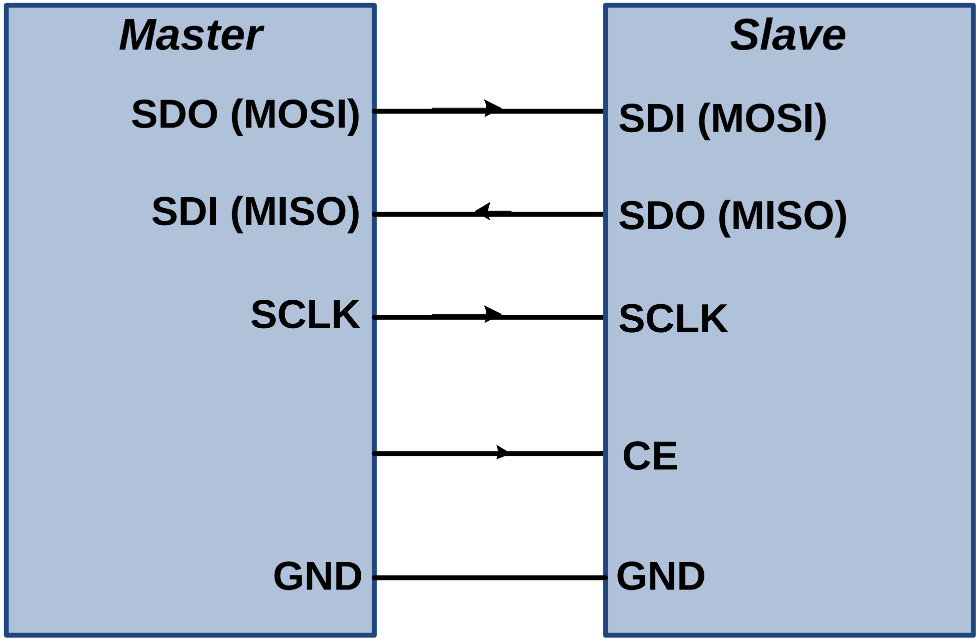
SPI

SPI Protocol

- Synchronous
- Full-duplex
- Serial
- Fast communication
- For short distances
- Pins
  - SDO (Data Out)
  - SDI (Data In)
  - SCLK (shift clock)
  - CE (chip enable)

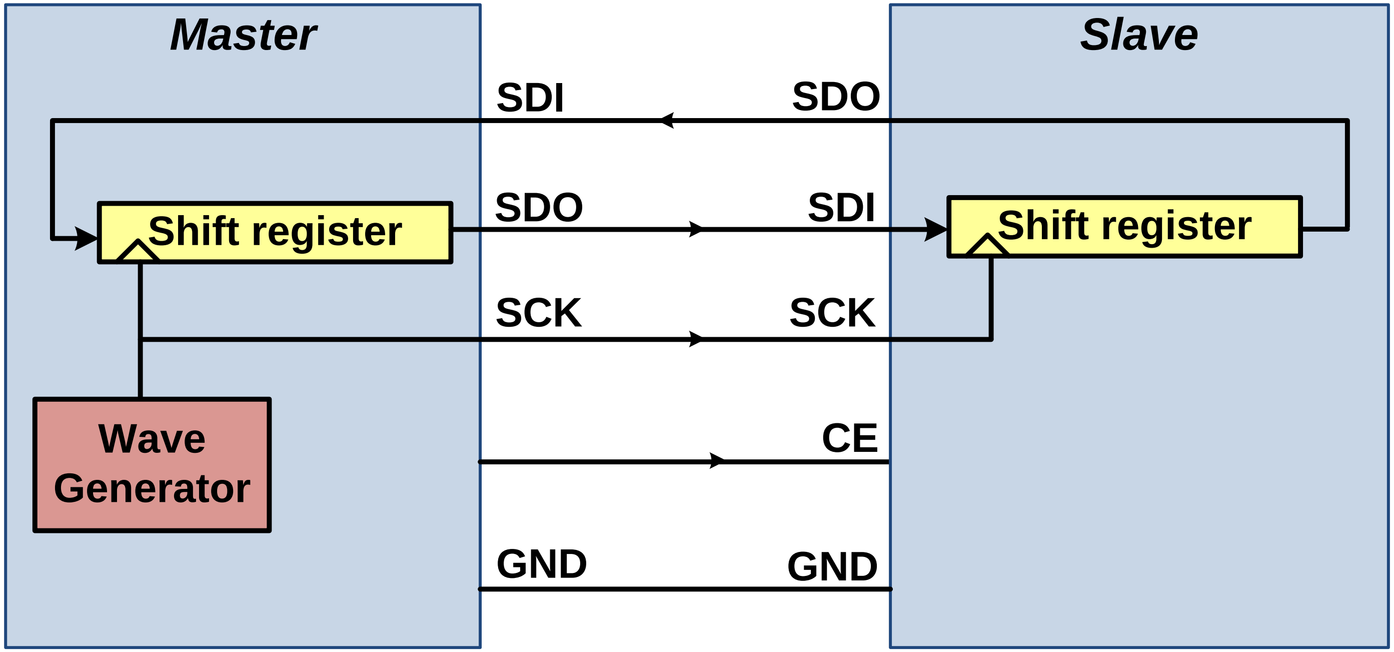


Master vs. Slave

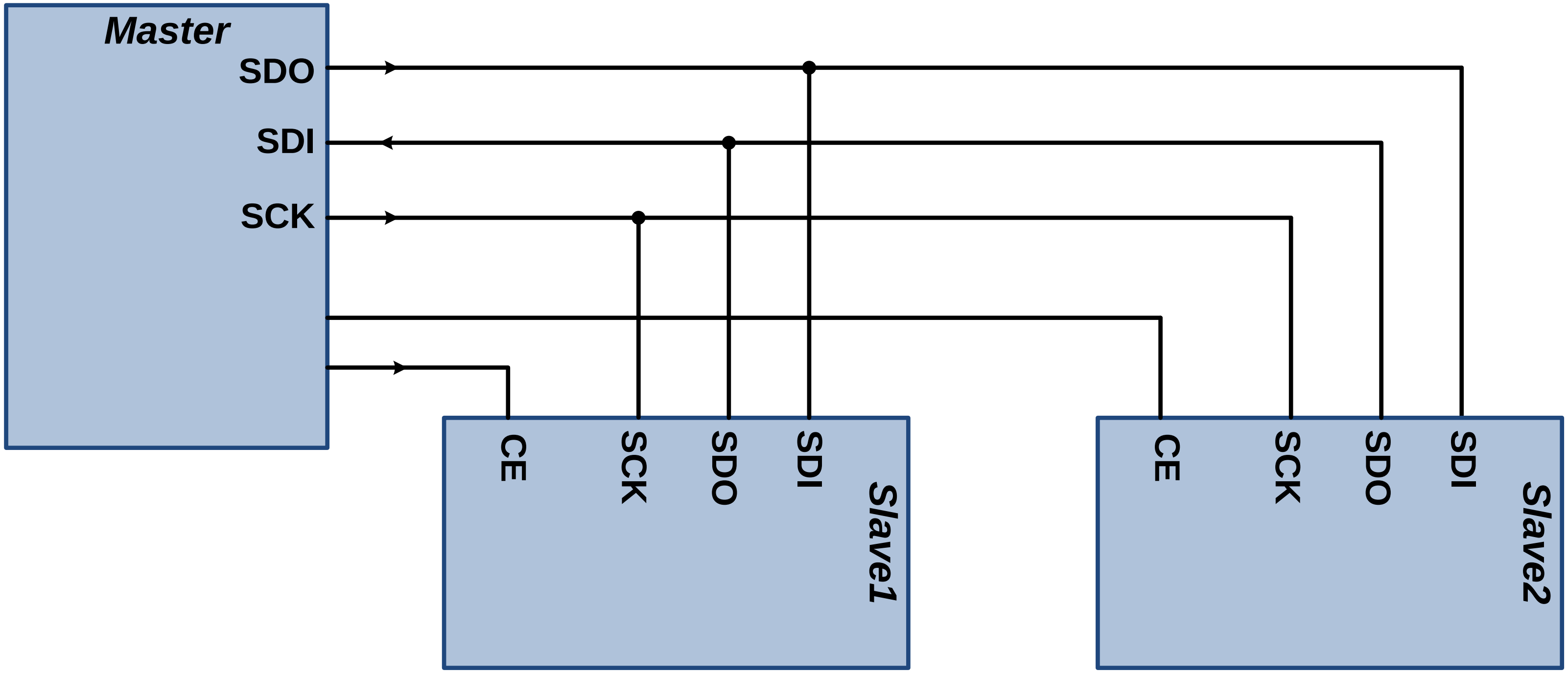
- Master begins the communication by pulling down the CE pin of slave.
- Master makes the clock for communication

SPI internal circuit

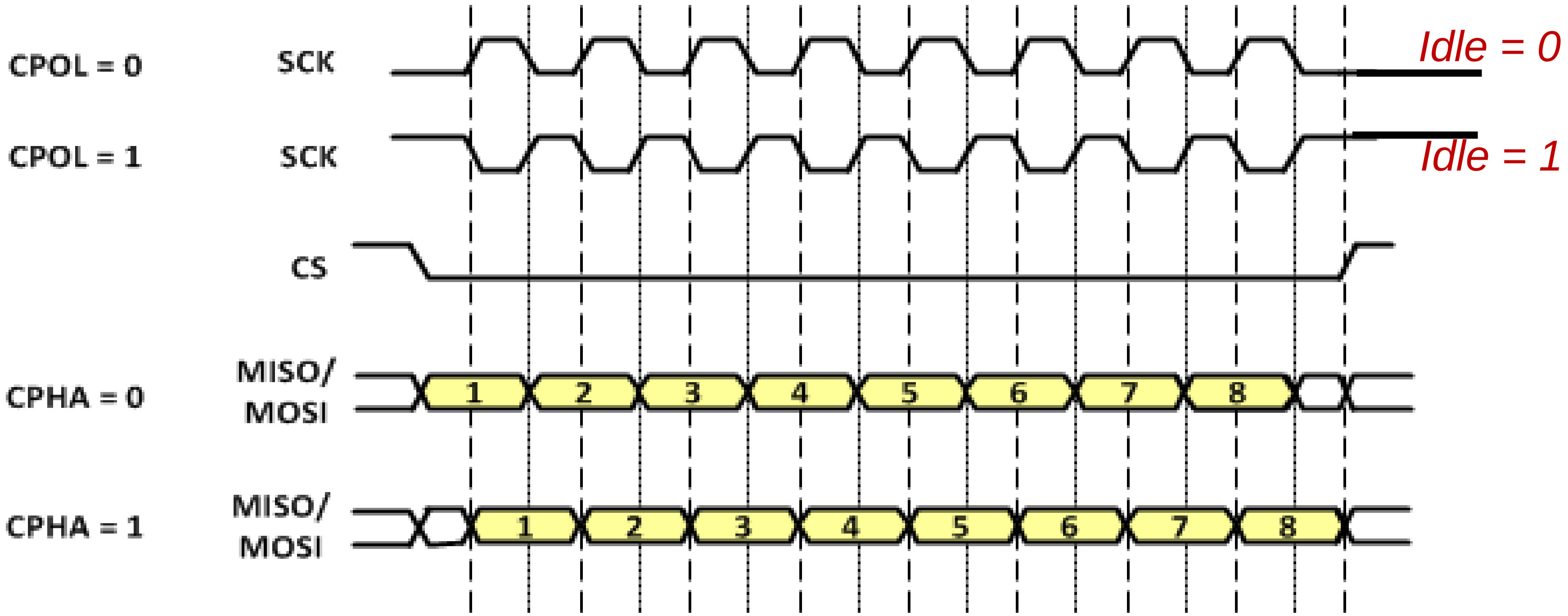
- A shift register in the master and another in the slave
- By each clock, a bit is shifted out from the master's shift register into the slave shift register and a bit is shifted from slave to master.



Multi-slave communication

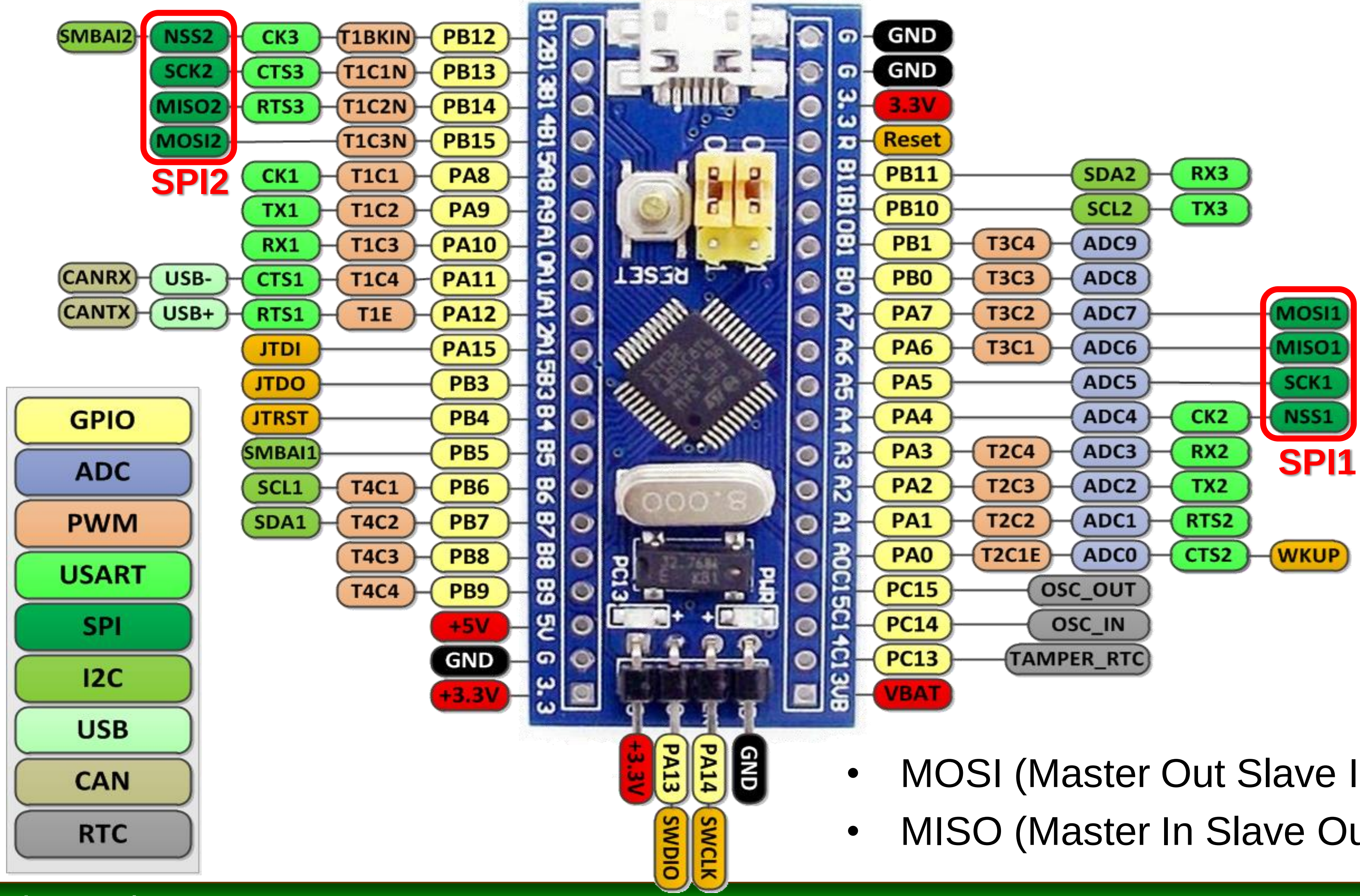


Polarity and Phase



CPOL	CPHA	Data Read and Change Time	SPI Mode
0	0	Read on rising edge, changed on a falling edge	0
0	1	Read on falling edge, changed on a rising edge	1
1	0	Read on falling edge, changed on a rising edge	2
1	1	Read on rising edge, changed on a falling edge	3

SPI pins in STM32F10x

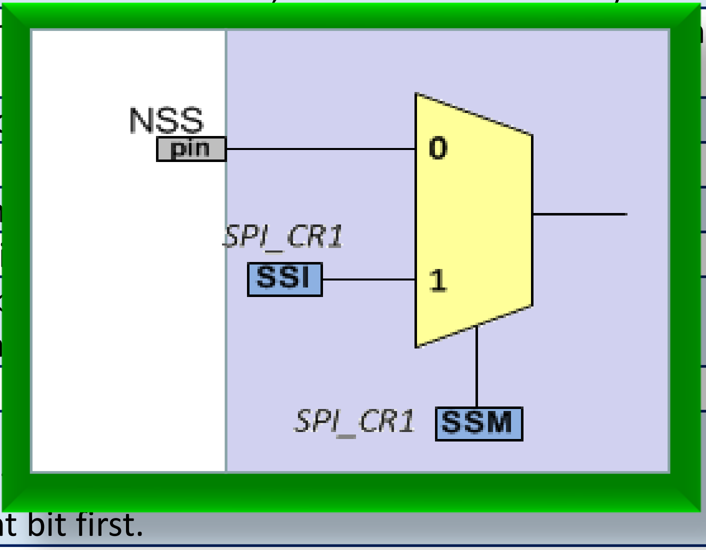


- MOSI (Master Out Slave In)
- MISO (Master In Slave Out)

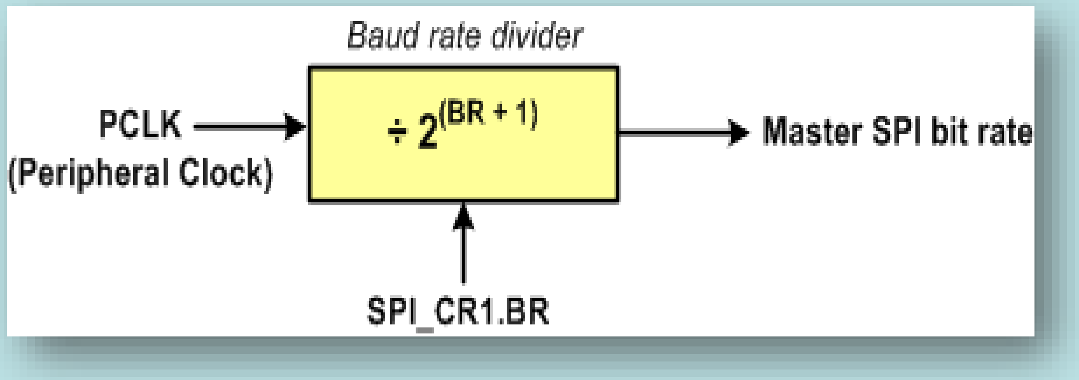
SPI_CR1:

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
BIDI MODE	BIDI OE	CRC EN	CRC NEXT	DFF	RX ONLY	SSM	SSI	LSB FIRST	SPE	BR			MSTR	CPOL	CPHA

Field	Bit	Descriptions
BIDIMODE	15	Bidirectional data mode enable (0: 2-line unidirectional, 1: 1-line bidirectional)
BIDIOE	14	When BIDIMODE=1 (1-line bidirection sfer (0: receive, 1: transmit)
CRCEN	13	Hardware CRC calculation enable (0: d
CRCNEXT	12	CRC transfer next (0: data transfer, 1:
DFF	11	Data Frame format (0: 8-bit data fram
RXONLY	10	Receive only (0: Both transmit & recei
SSM	9	Software Slave Management (0: NSS p If the bit is set, the SSI bit manages th
SSI	8	Internal Slave Select
LSBFIRST	7	LSB First Enable 1 = Data is transferred least significan 0 = Data is transferred most significant bit first.
SPE	6	SPI System Enable bit
B		
M		
C		
C		



BR	0	1	2	3	4	5	6	7
Speed	PCLK/2	PCLK/4	PCLK/8	PCLK/16	PCLK/32	PCLK/64	PCLK/128	PCLK/256



Example

- Assuming that the PCLK2 (peripheral clock for APB2) is 72MHz, find the SPI_CR1 value to initialize the SPI device as a master device, with bit rate of 2.25MHz, with active-high clock, sampling on rising edge, and 8-bit data MSB first. Make the NSS pin free

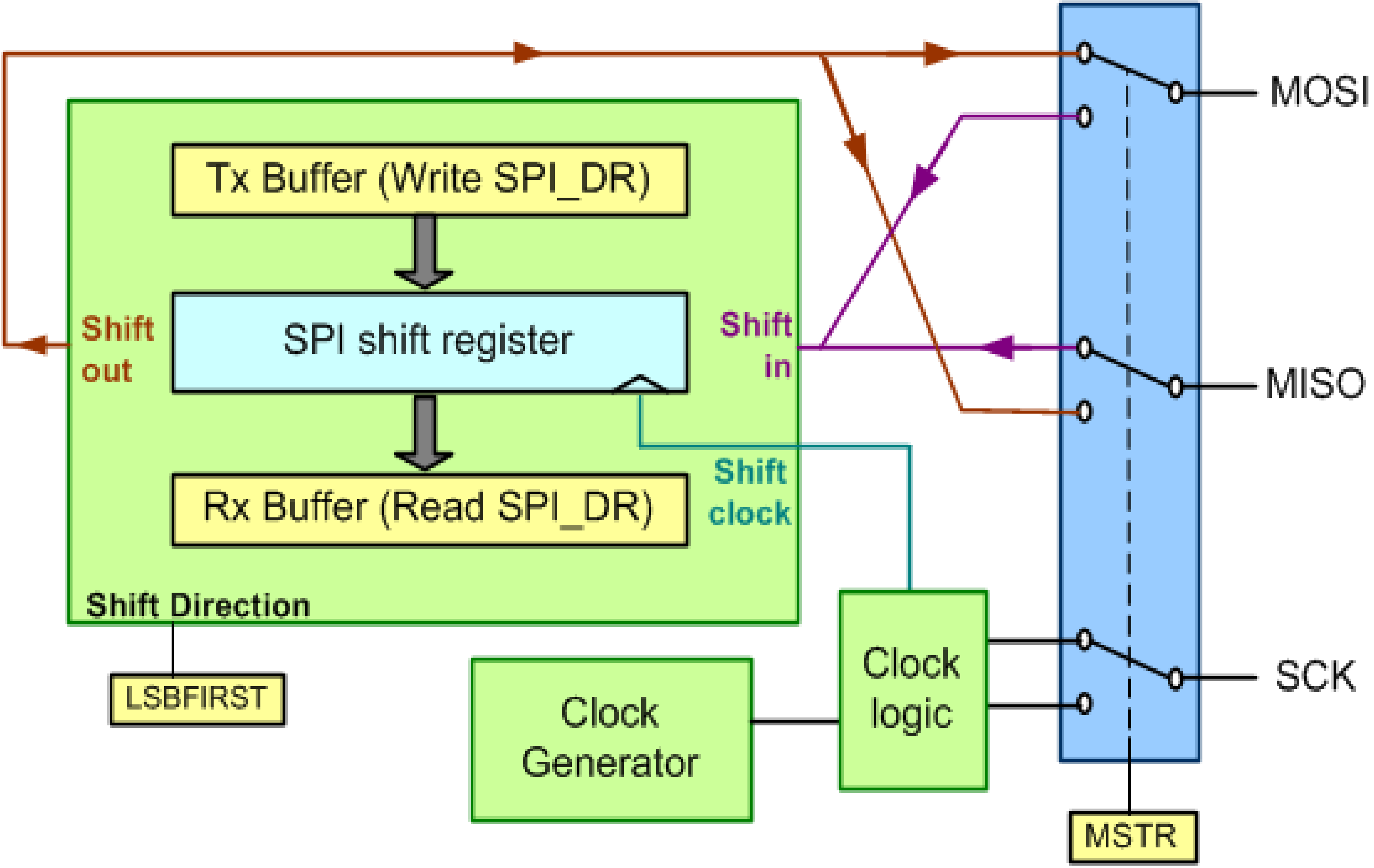
Solution:

72MHz / 2.25MHz = 64 = 2^6 ➔ BR = 5

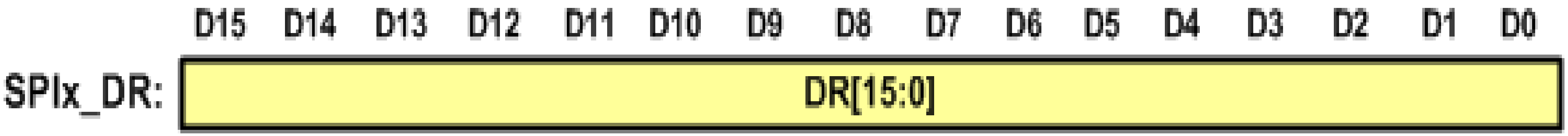
SPI_CR1	BIDIMODE	BIDIOE	CRCEN	CRCNEXT	DFF	RXONLY	SSM	SSI	LSBFIRST	SPE	BR	MSTR	CPOL	CPHA
	0	0	0	0	0	0	1	1	0	1	101	1	0	0

SPI_CR1 = 0x036C

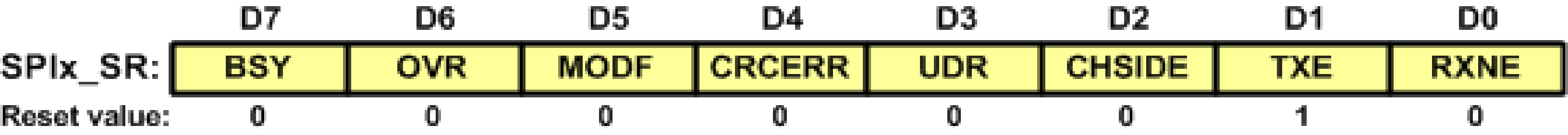
The MSTR bit and the role of pins



SPI Data Register (SPIx_DR)



	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
SPIx_DR:	DR[15:0]															



	D7	D6	D5	D4	D3	D2	D1	D0
SPIx_SR:	BSY	OVR	MODF	CRCERR	UDR	CHSIDE	TXE	RXNE
Reset value:	0	0	0	0	0	0	1	0

Field	Bit	Description
BSY	7	Busy flag (0: not busy, 1: busy) The bit is set when the SPI module is transferring data or TX buffer is not empty. The flag is set and cleared by hardware.
OVR	6	Overrun (0: no overrun, 1: overrun occurred) The flag sets when a new data is received and the previous data is not read.
MODF	5	Mode fault (0: no mode fault occurred, 1: mode fault occurred) If the STM32 is in master mode and the NSS pin is pulled down with another device, STM32 goes to slave mode and the flag is set.
CRCERR	4	CRC Error (For more information, see the user manual.)
UDR	3	Underrun (0: no underrun, 1: underrun occurred) The flag sets if the STM32 is in slave mode and a SCLK clock is appeared while we have not loaded data to the data register (SPI_DR).
CHSIDE	2	Channel side (It is not used in SPI mode.)
TXE	1	Transmit buffer empty (0: not empty, 1: empty) The flag is set if the SPI transmit buffer is empty and it is ready to send another data.
RXNE	0	Receive buffer not empty (0: empty, 1: not empty) The flag is set if a new data is received.

Sending and Receiving as Master

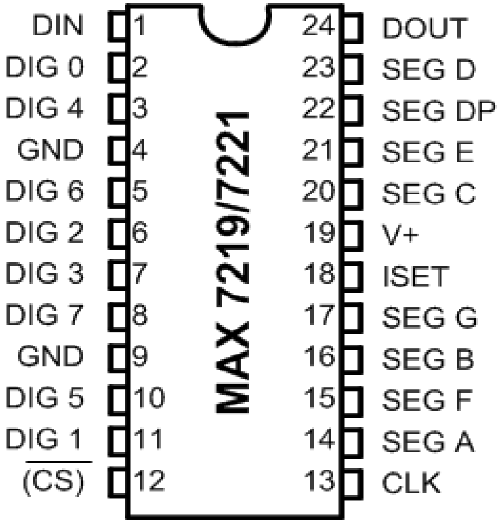
1. Enable the clocks for SPI and GPIO.
2. Initialize MOSI and SCK as alternate function output push-pull (CNFx = 10) and make MISO an input pin.
3. Initialize SPI_CR1 with proper value:
4. Consider a GPIO pin for the CE pin
5. Make low the CE pin of the desired slave.
6. Load SP_DR to send data.
7. Monitor the SPI_SR register until the TXE is set (or RXNE is set).
8. Read SP_DR to get the received data.
9. Repeat steps 6 to 8 until all data are transferred.
10. Make the CE pin of the slave device high.

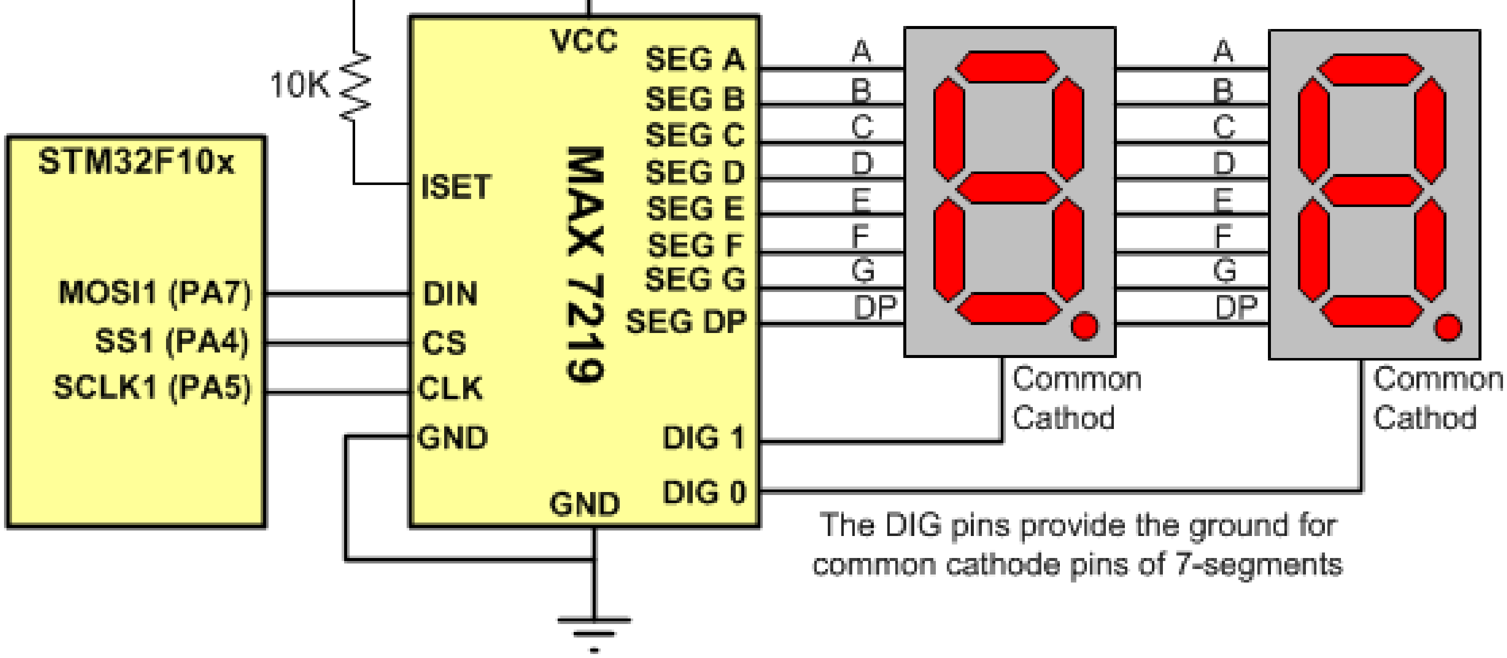
Sending ‘A’ to ‘Z’ via SPI

```
#include <stm32f10x.h>

void spi1_init(void);
uint8_t spi1_transfer(uint8_t d);

int main( ) {
  RCC->APB2ENR |= 0xFC;            /* enable clocks for GPIO */

  spi1_init(); /* initialize the SPI module */

  /*--- make the NSS pin of the slave low if needed ---*/

  for(char c = 'A'; c <= 'Z'; c++) /* send characters ‘A’ to ‘Z’ */
    spi1_transfer(c); /* send c through SPI */

  /*--- make the NSS pin high ---*/

  while(1)
  {  }
}

/* The function initializes the SPI module */
```

MAX7219/MAX7221

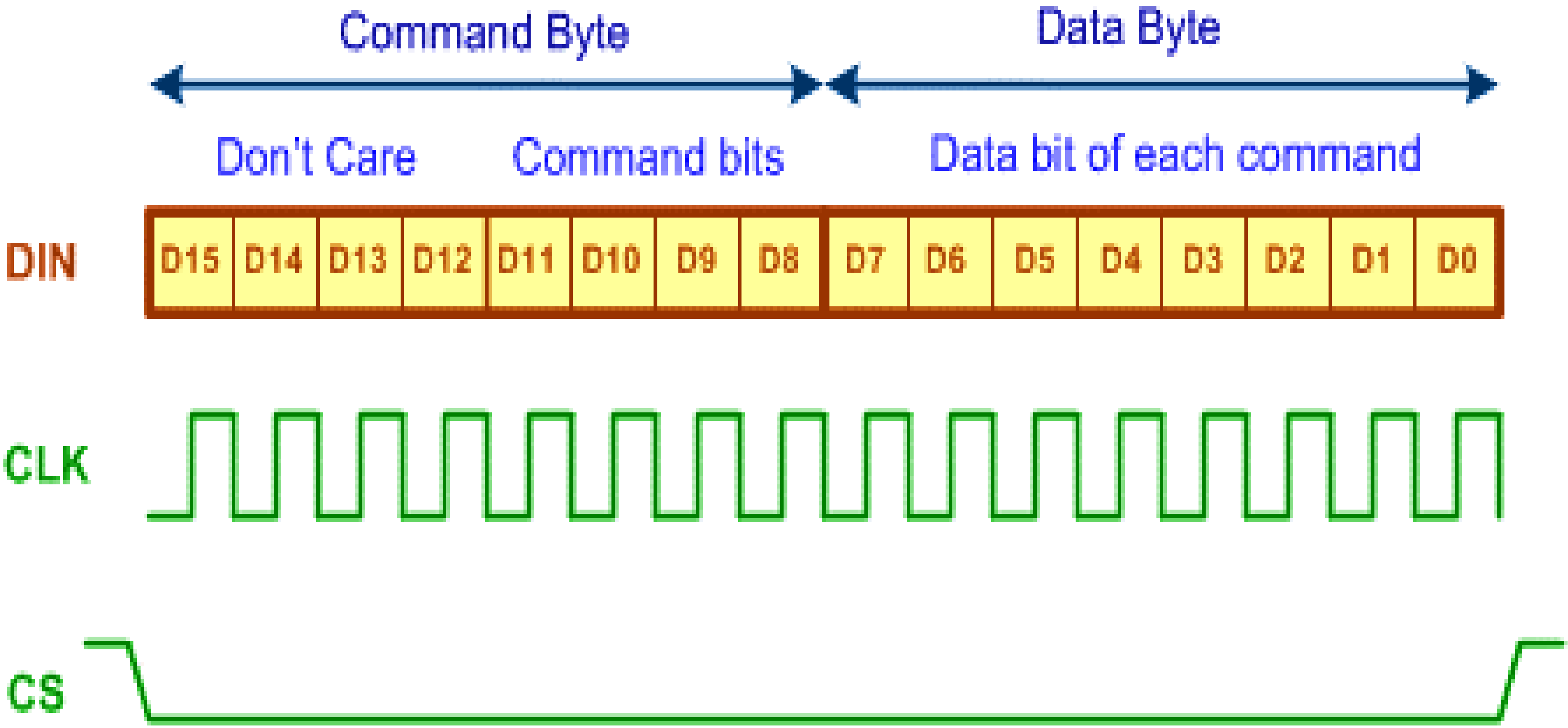
- Drives 8-digit common cathode 7-segment
- It has SPI bus





The DIG pins provide the ground for common cathode pins of 7-segments

MAX7219/MAX7221 Packet Format



Command Byte

Data Byte

Don't Care

Command bits

Data bit of each command

DIN	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

CLK

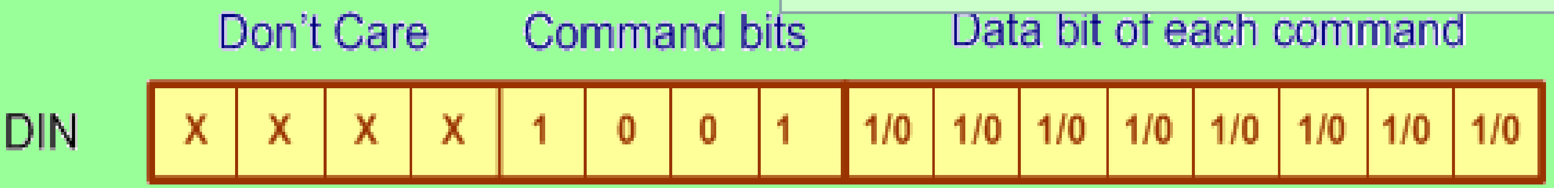
CS

List of Commands in MAX7221/MAX7219

Command	D15-12	D11
No operation	X	0
Set value of digit 0	X	0
Set value of digit 1	X	0
Set value of digit 2	X	0
Set value of digit 3	X	0
Set value of digit 4	X	0
Set value of digit 5	X	0
Set value of digit 6	X	0
Set value of digit 7	X	1
Set decoding mode	X	1

Set Scan limit (XB)

Data	Meaning
0	Scan digit 0
1	Scan digits 0 and 1
2	Scan digits 0, 1, and 2
3	Scan digits 0 to 3
4	Scan digits 0 to 4
5	Scan digits 0 to 5
6	Scan digits 0 to 6
7	Scan digits 0 to 7



	Don't Care				Command bits				Data bit of each command							
DIN	X	X	X	X	1	0	0	1	1/0	1/0	1/0	1/0	1/0	1/0	1/0	1/0

0: bypass decoding
1: decode

Digits are designated as 0-7 to drive total of eight 7-segment LEDs.

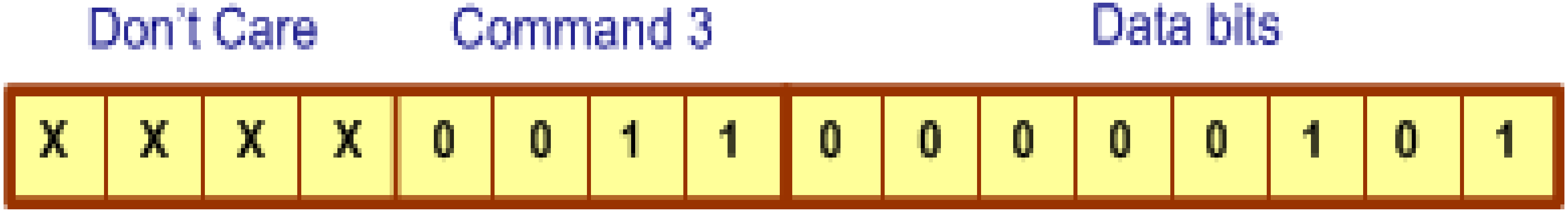
Example

- What sequence of bytes should be sent to the MAX7219 in order to enable the decoding function for digit 0 and digit 2, and disable the decoding function for other digits?

Don't Care				Command 9				Data bits							
X	X	X	X	1	0	0	1	0	0	0	0	0	1	0	1

Example

- what sequence of numbers should be sent to the MAX7219 in order to write 5 on digit 2?



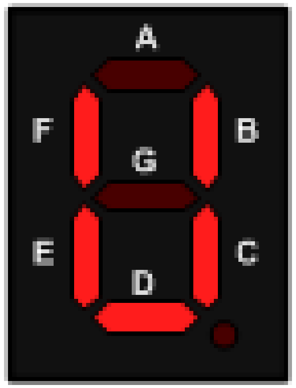
Don't Care				Command 3				Data bits							
X	X	X	X	0	0	1	1	0	0	0	0	0	1	0	1

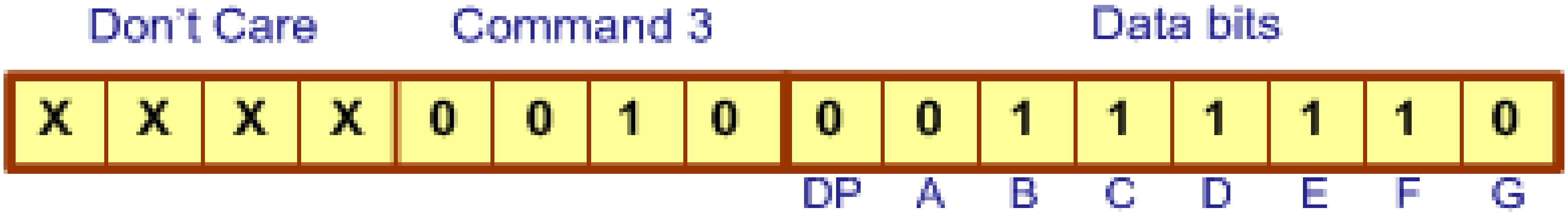
Example: Display 49 using 8-bit transfer

```
#include <stm32f10x.h>

void spi1_init(void);
uint8_t spi1_transfer(uint8_t d);
void max7219_send(uint8_t cmd, uint8_t data);

int main( ) {
        RCC->APB2ENR |= 0xFC;               /* enable clocks for GPIO */

        spi1_init(); /* initialize the SPI module */

        max7219_send(0x09, 0xFF);           /* enable decoding for all digits */
        max7219_send(0x0B, 1);              /* 2 (1+1) digits */
        max7219_send(0x0C, 0x01);           /* turn on */

        max7219_send(0x01, 9);   /* show 9 on digit 1 */
        max7219_send(0x02, 4);   /* show 4 on digit 2 */

        while(1) { }
}

/* The function initializes the SPI module */
```

Example

- After disabling the decoder, what sequence of numbers should be sent to the MAX7219 in order to write U on digit 1?





Don't Care				Command 3				Data bits							
X	X	X	X	0	0	1	0	0	0	1	1	1	1	1	0
								DP	A	B	C	D	E	F	G

Example: Display 2U

```
#include <stm32f10x.h>

void spi1_init(void);
uint8_t spi1_transfer(uint8_t d);
void max7219_send(uint8_t cmd, uint8_t data);

int main( ) {
  RCC->APB2ENR |= 0xFC;            /* enable clocks for GPIO */
  spi1_init(); /* initialize the SPI module */

  max7219_send(0x09, 0x02);  /* enable decoding for digit2 and disable for digit1  */
  max7219_send(0x0B, 1);           /* 2 (1+1) digits */
  max7219_send(0x0C, 0x01);        /* turn on */

  max7219_send(0x01, 0x3E);        /* show U on digit 1 */
  max7219_send(0x02, 2); /* show 2 on digit 2 */

  while(1) {
  }
}
```