CENTRO UNIVERSITARIO DE AUTOMACIÓN Y ROBÓTICA

Facultad Regional Córdoba

TECNICAS DIGITALES III

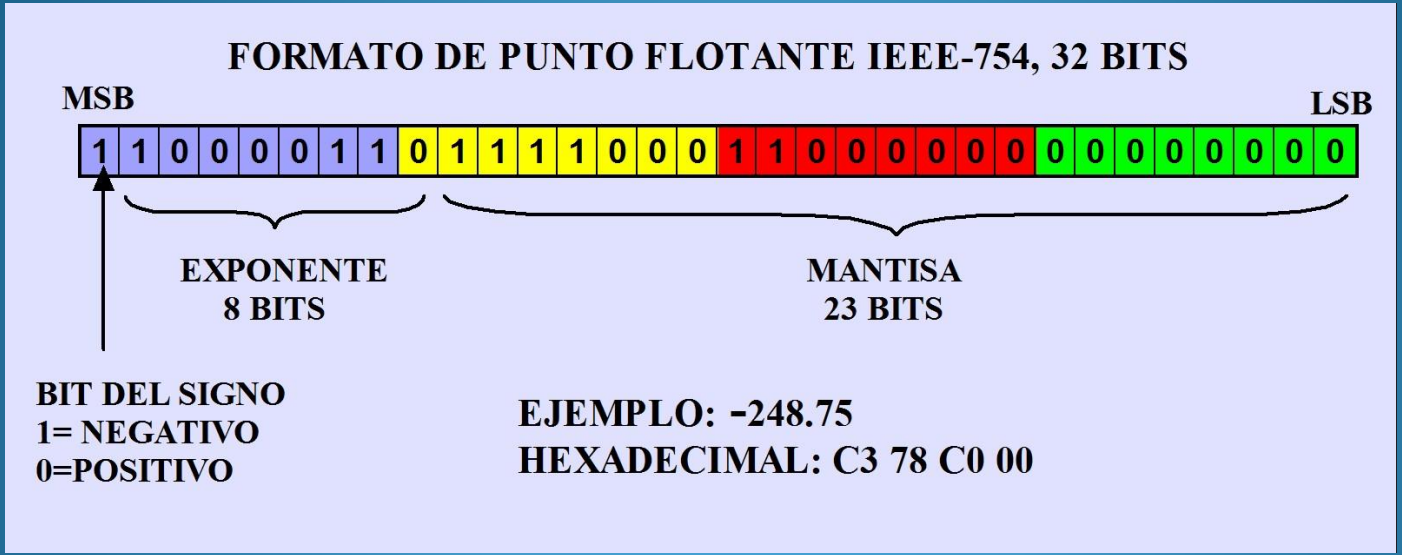
Agenda

- **Uso de uniones y estructuras para transferencia de datos en puertos serie.**
- **Comunicaciones digitales.**
- **Comunicación CAN**

Uniones y estructuras

```
// Definición de la estructura
struct Persona {
    char nombre[50];
    int edad;
    float altura;
};
```

Punto flotante - IEEE 754



Ejemplo Float

0x3f9d70a4

3	f	9	d	7	0	a	4
0 0 1 1	1 1 1 1	1 0 0 1	1 1 0 1	0 1 1 1	0 0 0 0	1 0 1 0	0 1 0 0
0	01111111	00111010111000010100100					

sign	exponent	mantissa
+1	127	1.00111010111000010100100 (binary)
+1 *	2^(127 - 127) *	1.23000001907348633
+1 *	1.00000000 *	1.23000001907348633
1.23		

Float value: 1.23 Convert to hex

https://gregstoll.com/~gregstoll/floattohex/

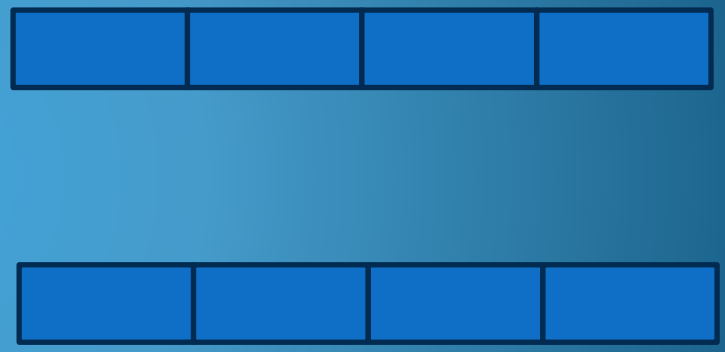
Codificación en exceso del exponente

En el caso del formato "float" de 32 bits en IEEE 754, el exponente se codifica en exceso 127. Esto significa que el exponente real se obtiene restando 127 del valor almacenado en los bits del exponente. Por ejemplo:

- Si el campo de exponente tiene el valor binario 01111111 (equivalente a 127 en decimal), el exponente real es 127 - 127 = 0.
- Si el campo de exponente tiene el valor binario 10000000 (equivalente a 128 en decimal), el exponente real es 128 - 127 = 1.

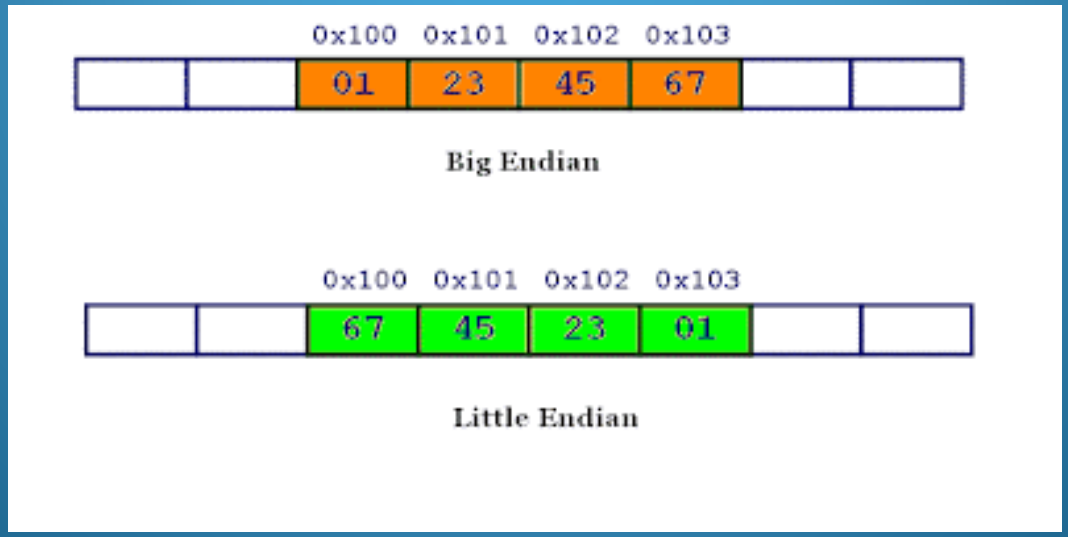
Estructura Float y char

```
struct Datos {
    float numero;
    char cadena[4];
};
```



Union Float y char

```
union FloatOrString {
    float numero;
    char cadena[4];
};
```



Union Float y char

- En una arquitectura little-endian, el primer byte de la cadena de caracteres (**char[0]**) corresponderá al byte menos significativo (menos peso) del **float**.
- En una arquitectura big-endian, el primer byte de la cadena de caracteres (**char[0]**) corresponderá al byte más significativo (más peso) del **float**.

Union Float y char

- En una arquitectura little-endian, el primer byte de la cadena de caracteres (**char[0]**) corresponderá al byte menos significativo (menos peso) del **float**.
- En una arquitectura big-endian, el primer byte de la cadena de caracteres (**char[0]**) corresponderá al byte más significativo (más peso) del **float**.

Union Float y char

- La arquitectura x86 y x86-64 utiliza little-endian.
- La arquitectura ARM puede ser configurada como little-endian o big-endian, dependiendo de la configuración.*
- La arquitectura PowerPC suele ser big-endian.
- La arquitectura MIPS puede configurarse en ambos modos, dependiendo de la implementación.
- El ESP32, basado en el núcleo Xtensa, utiliza una arquitectura little-endian

*al compilar con GCC (el compilador C de GNU), puedes usar la opción **-mbig-endian** o **-mlittle-endian** para especificar la endianness que deseas.

Acceder a cada elemento de la unión o estructura

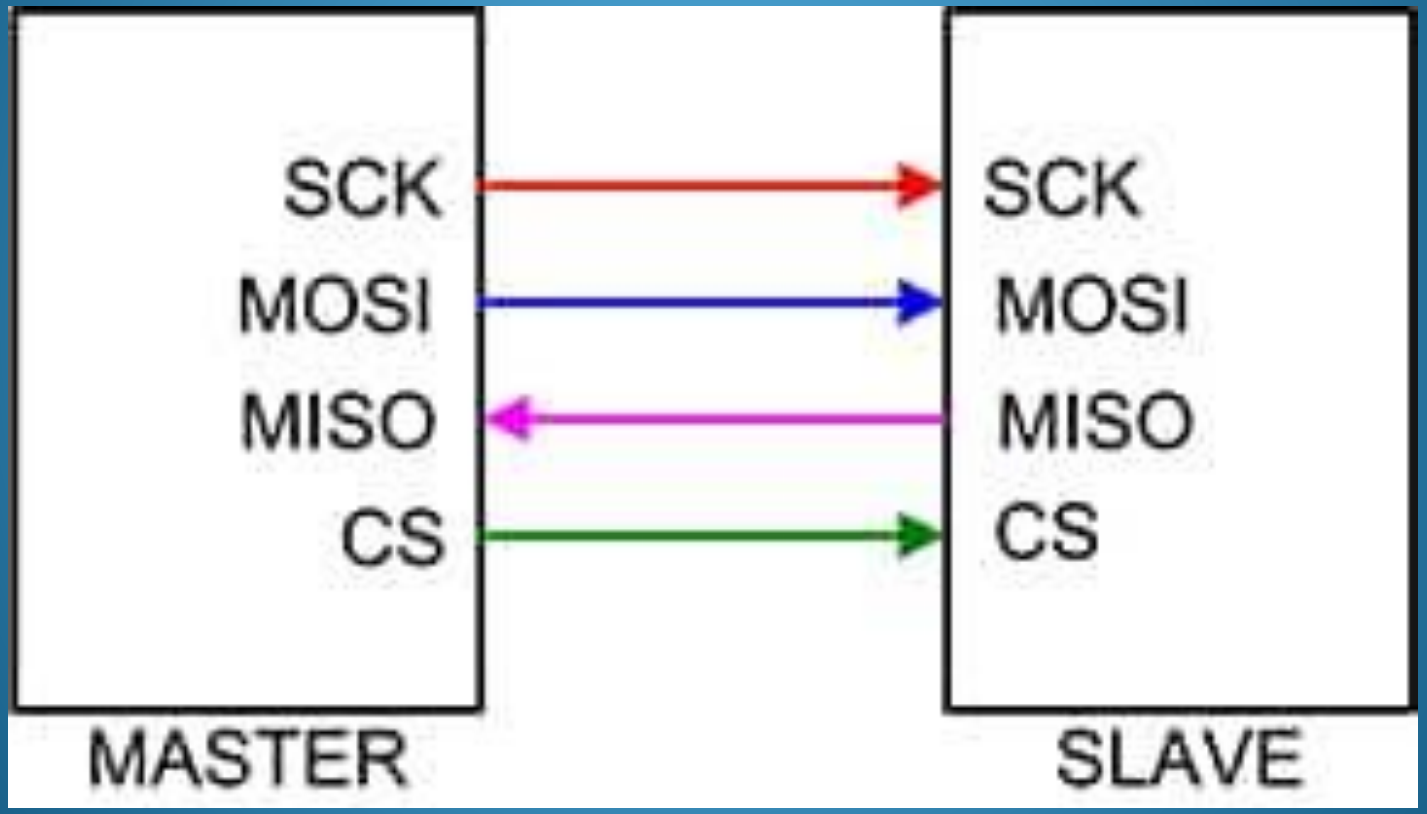
Acceso a la unión o estructura

```
valor.cadena
```

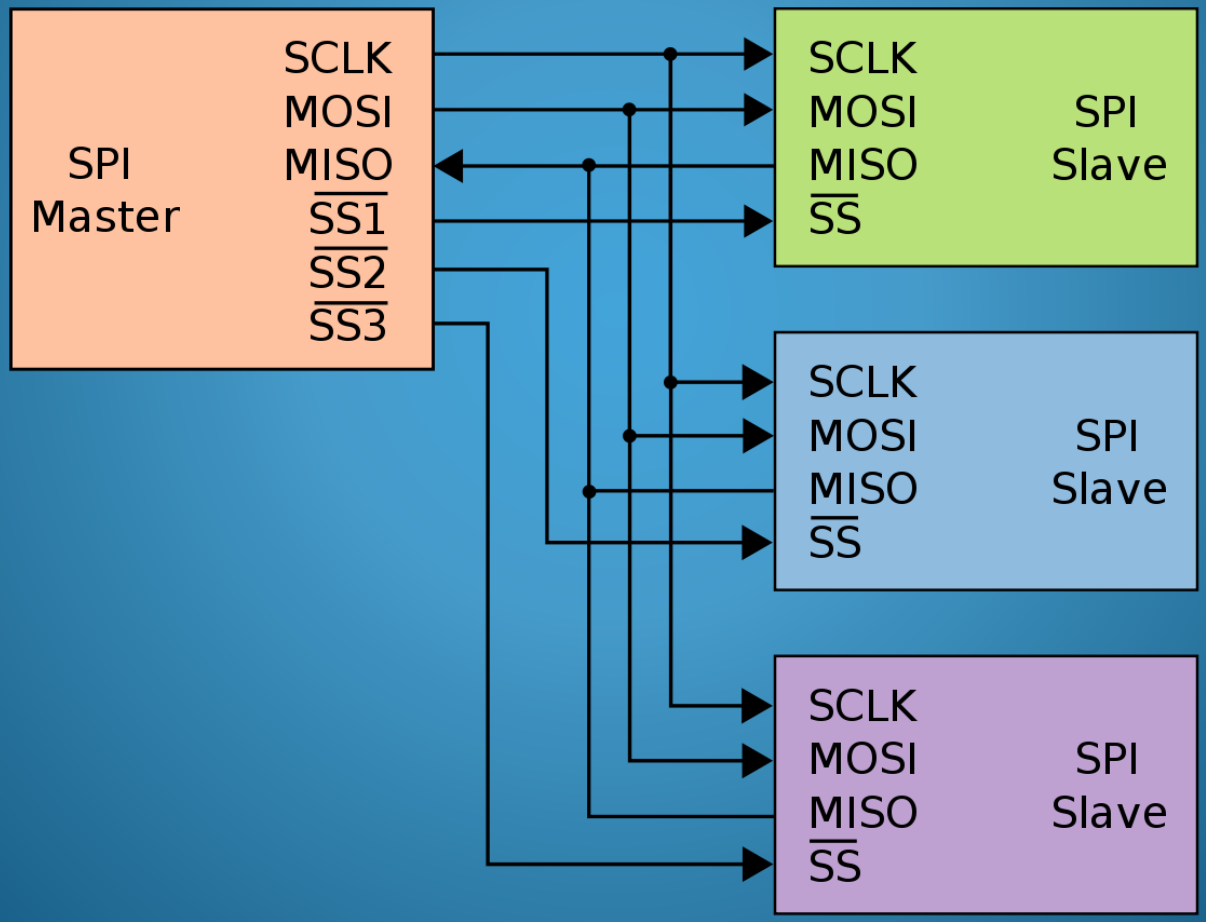
Acceso a la unión o estructura cuando la misma fue pasada a una función como un puntero ej función(&pPersona)

```
pPersona->nombre
```

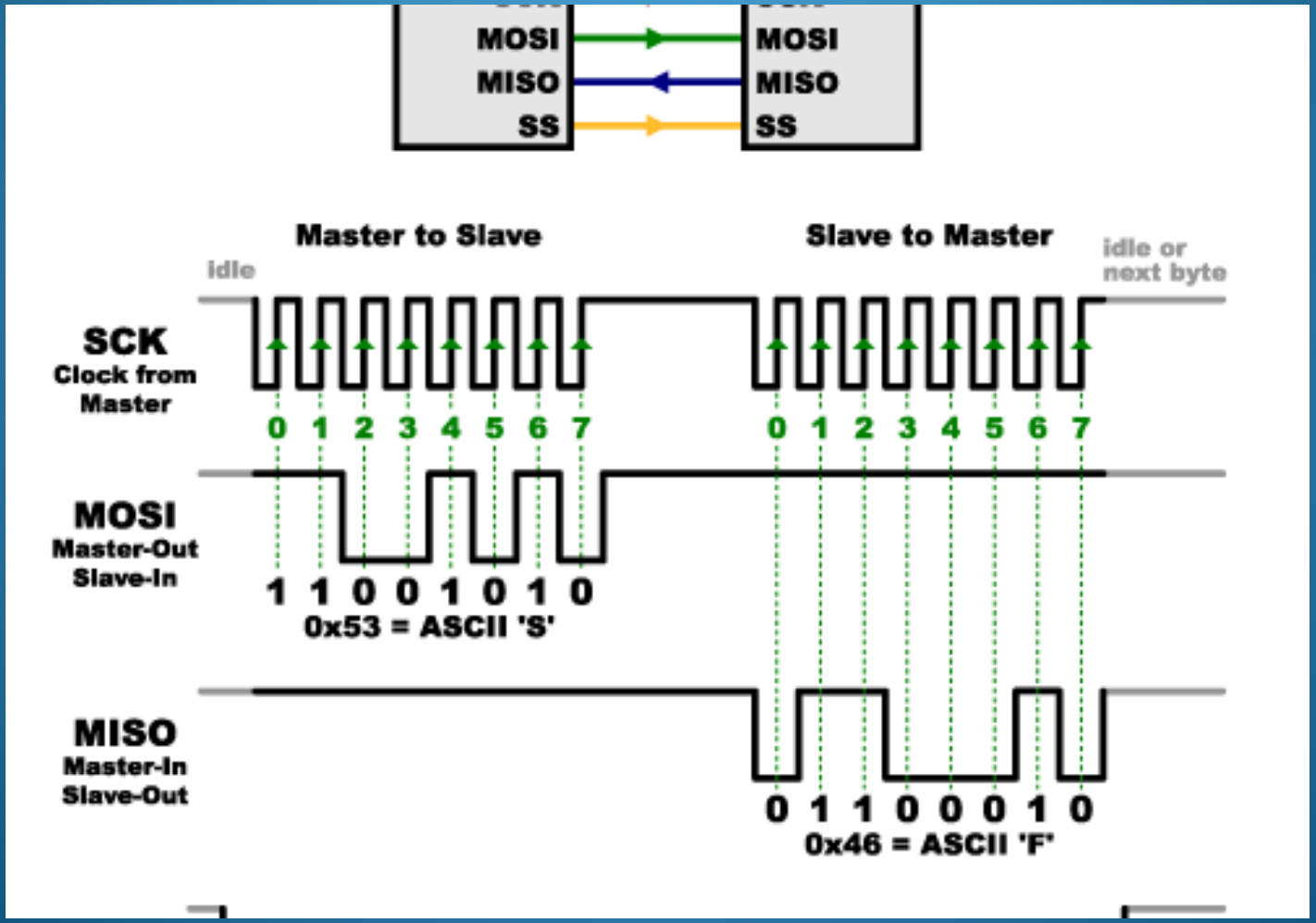
Comunicaciones a nivel IC
SPI



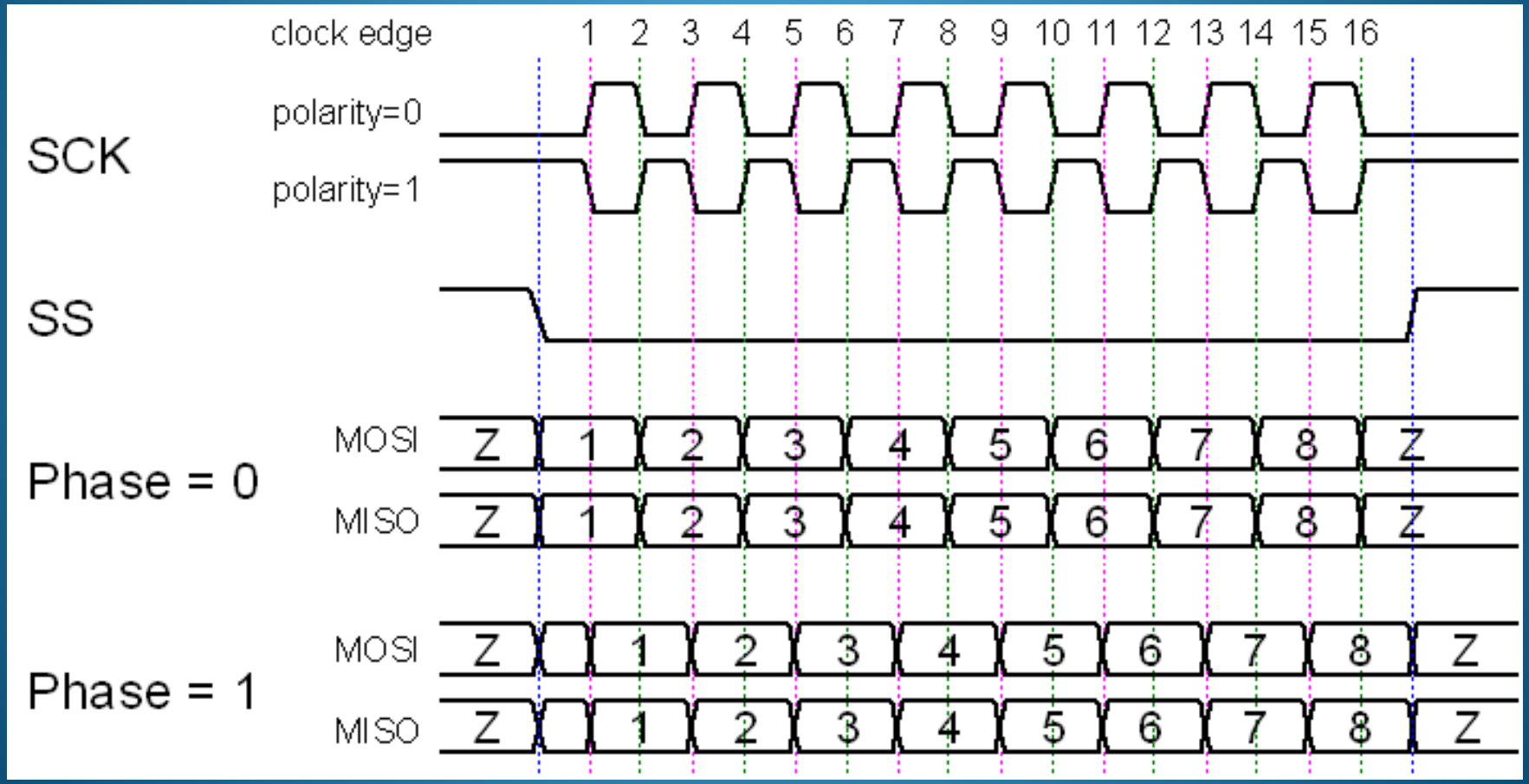
Comunicaciones a nivel IC
SPI



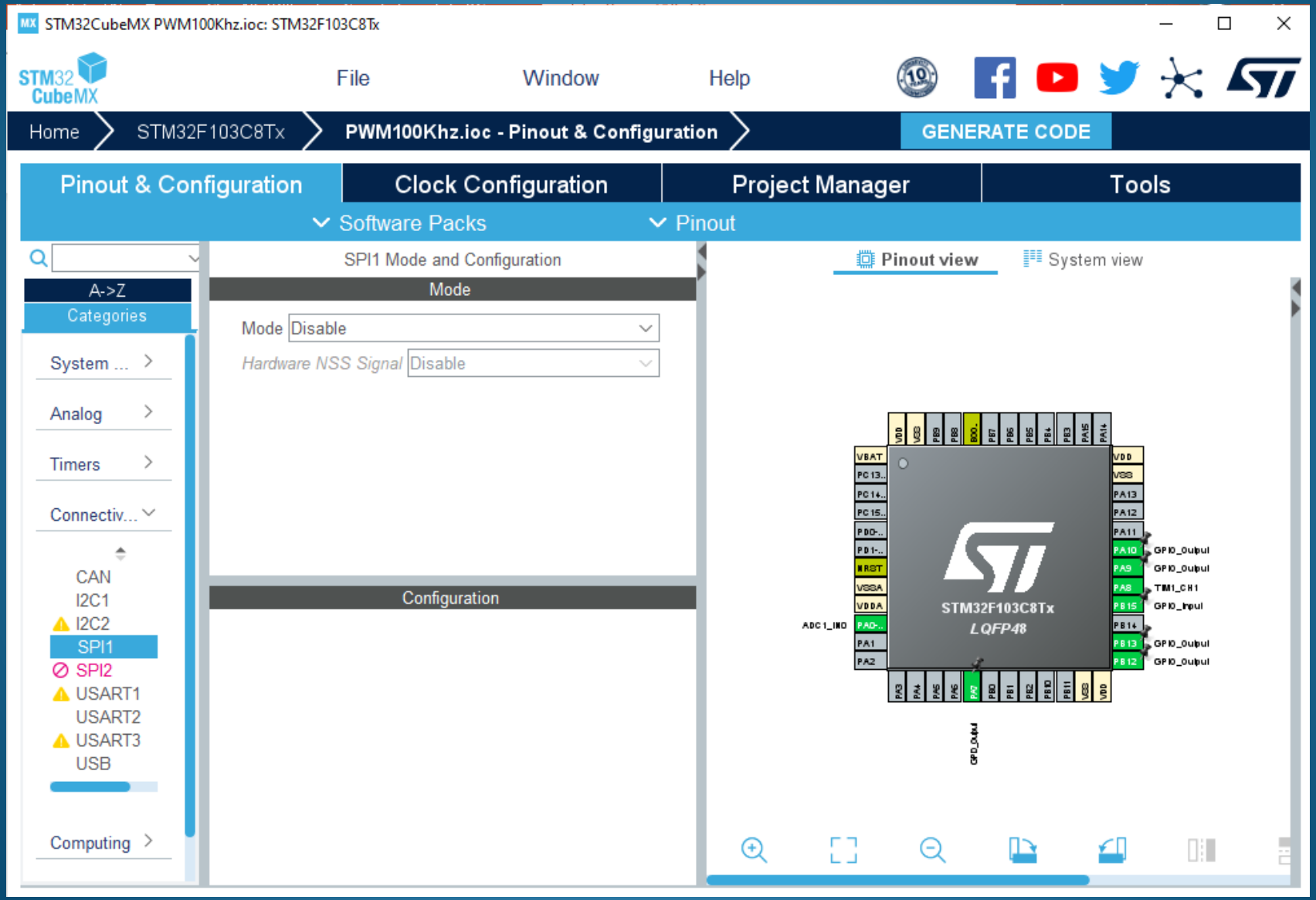
Comunicaciones a nivel IC
SPI



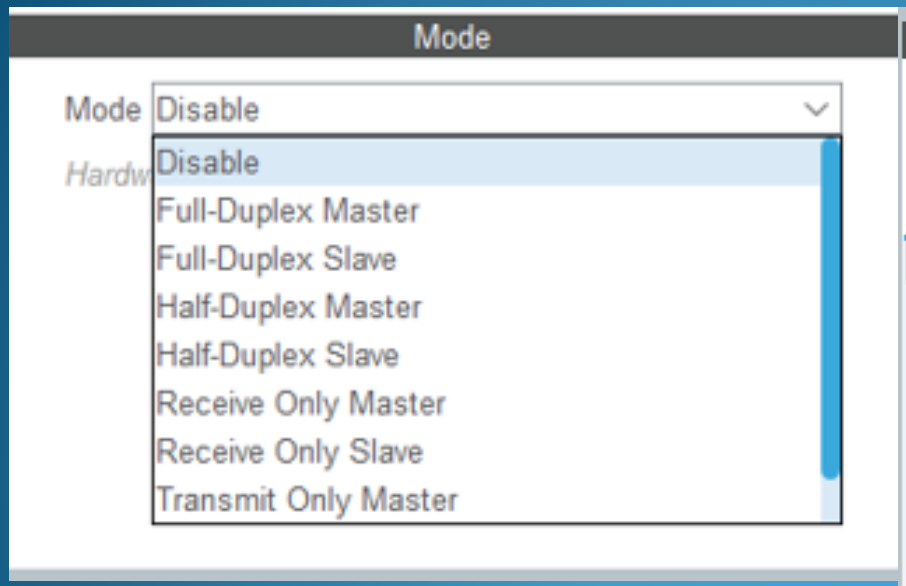
Comunicaciones a nivel IC
SPI



Registros del STM32



Registros del STM32



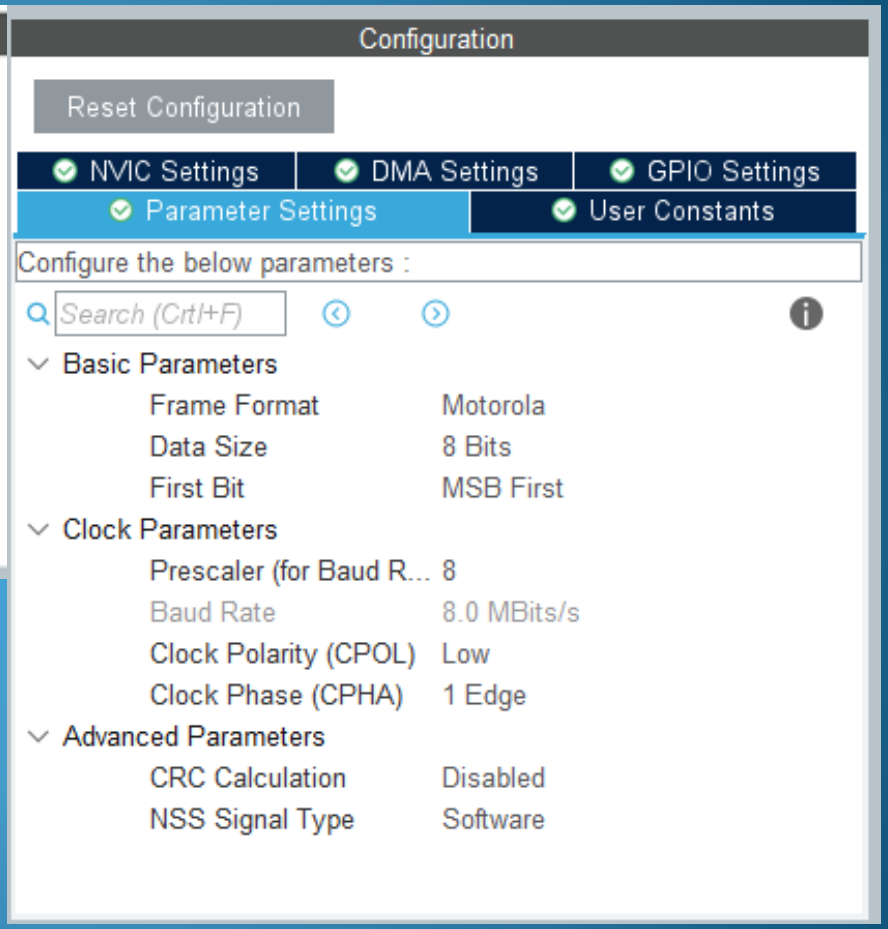


Figure 4. STM32F103xx performance line LQFP100 pinout

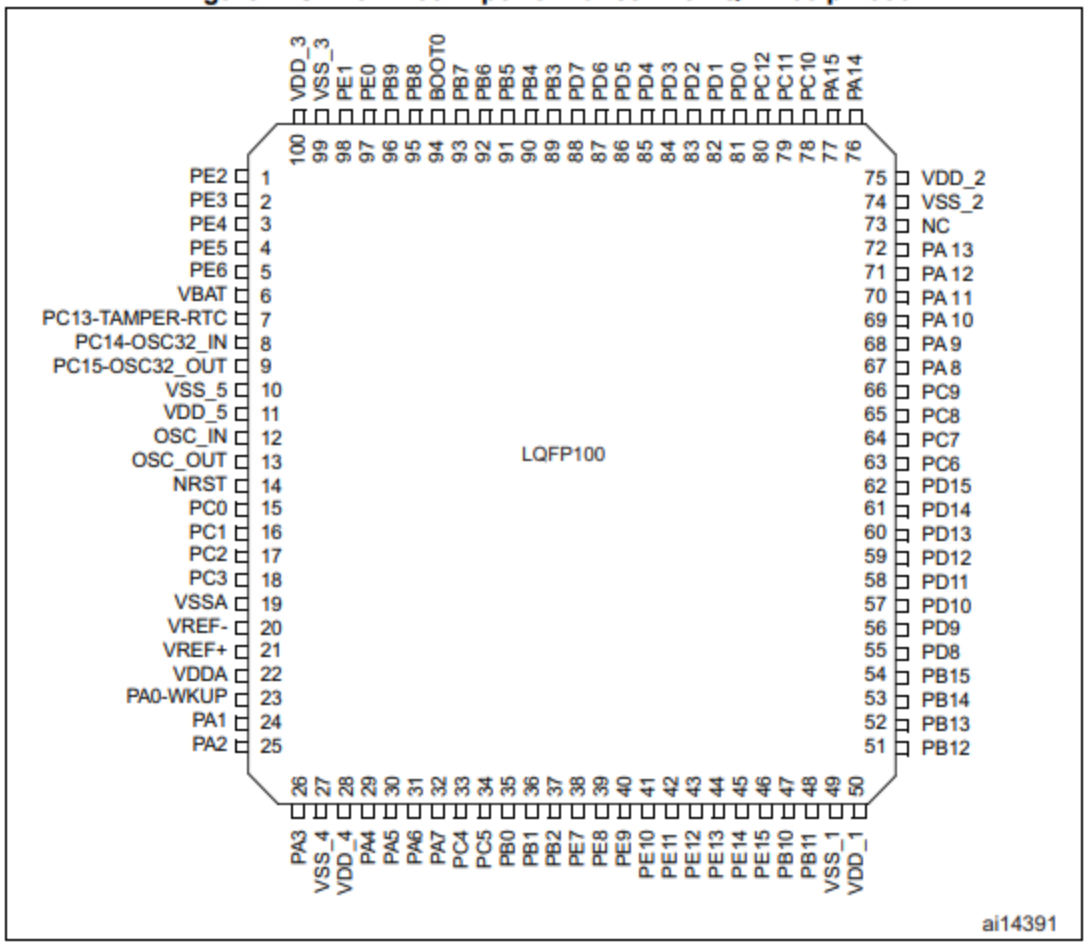


Table 5. Medium-density STM32F103xx pin definitions (continued)

Pins							Pin name	Type[1]	I / O Level[2]	Main function[3] (after reset)	Alternate functions[4]	
LFBGA100	UFBG100	LQFP48/UFQFPN48	TFBGA64	LQFP64	LQFP100	VFQFPN36					Default	Remap
H9	J12	-	-	-	57	-	PD10	I/O	FT	PD10	-	USART3_CK
G9	J11	-	-	-	58	-	PD11	I/O	FT	PD11	-	USART3_CTS
K10	J10	-	-	-	59	-	PD12	I/O	FT	PD12	-	TIM4_CH1 / USART3_RTS
J10	H12	-	-	-	60	-	PD13	I/O	FT	PD13	-	TIM4_CH2
H10	H11	-	-	-	61	-	PD14	I/O	FT	PD14	-	TIM4_CH3
G10	H10	-	-	-	62	-	PD15	I/O	FT	PD15	-	TIM4_CH4

Figure 238. SPI block diagram

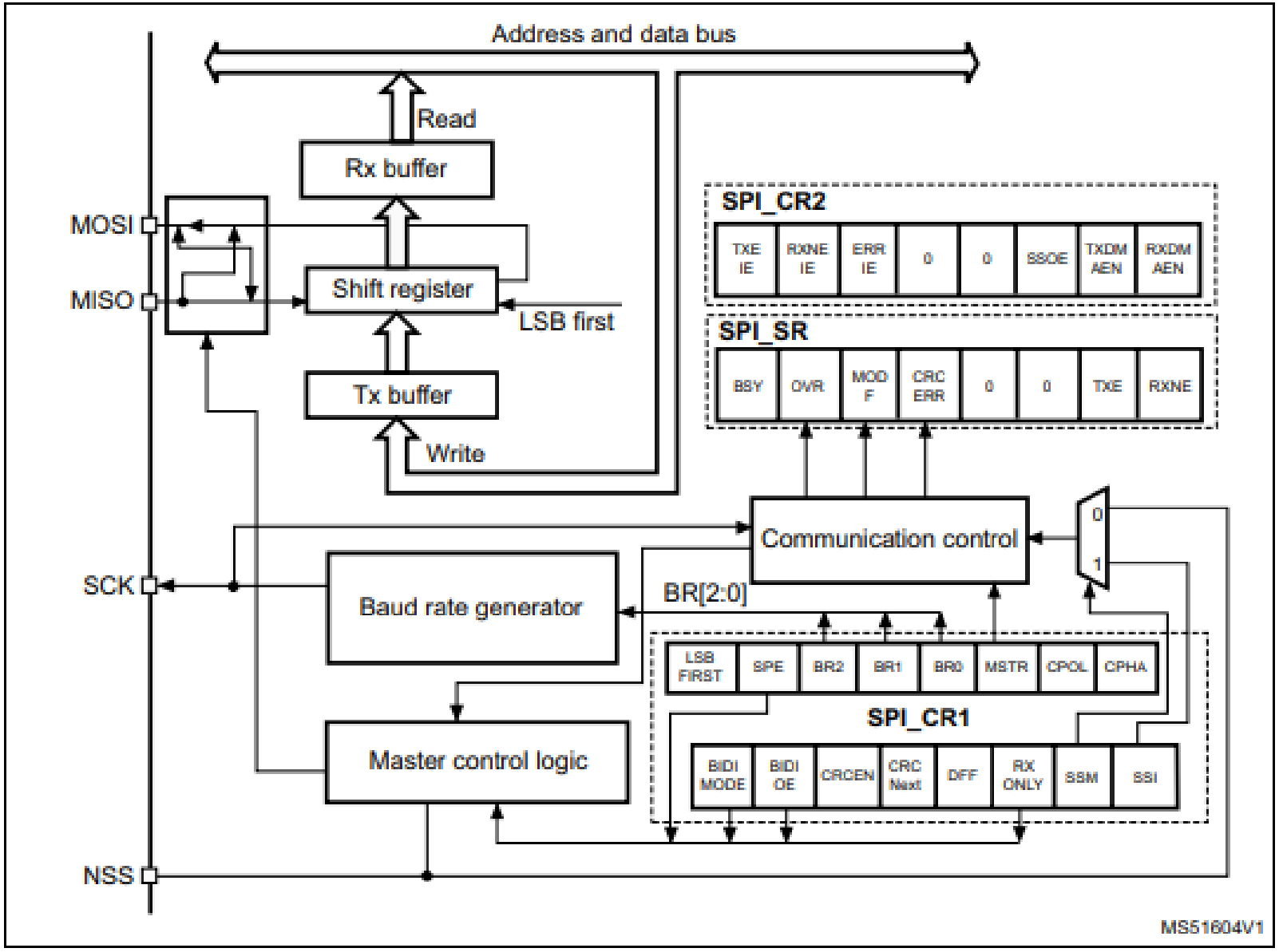


Figure 239. Single master/ single slave application

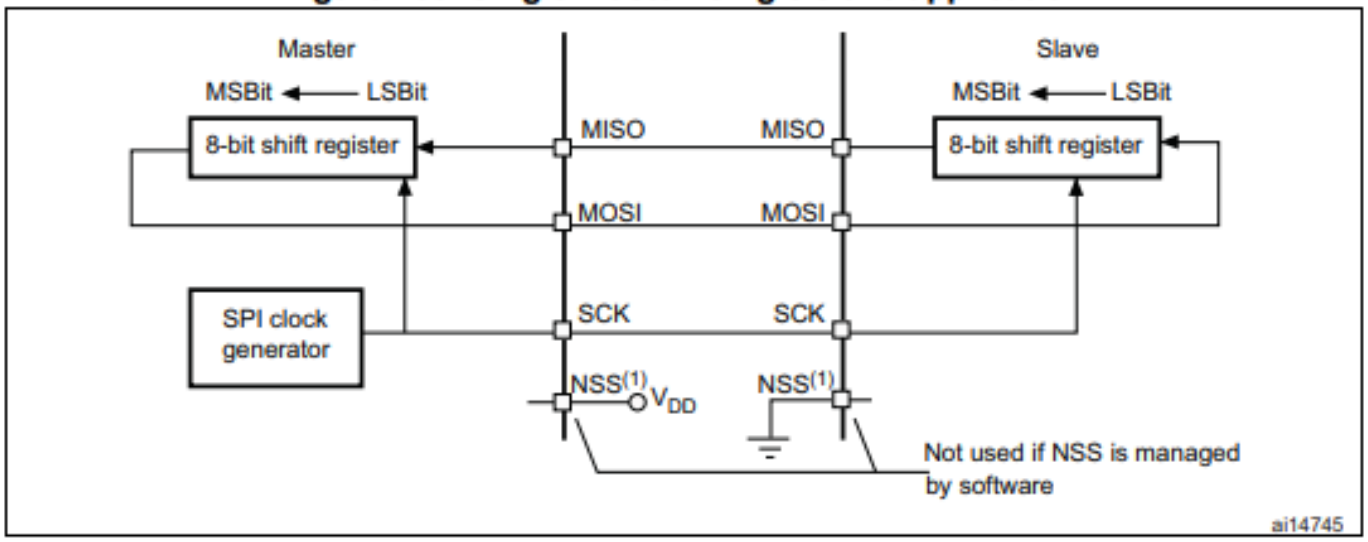
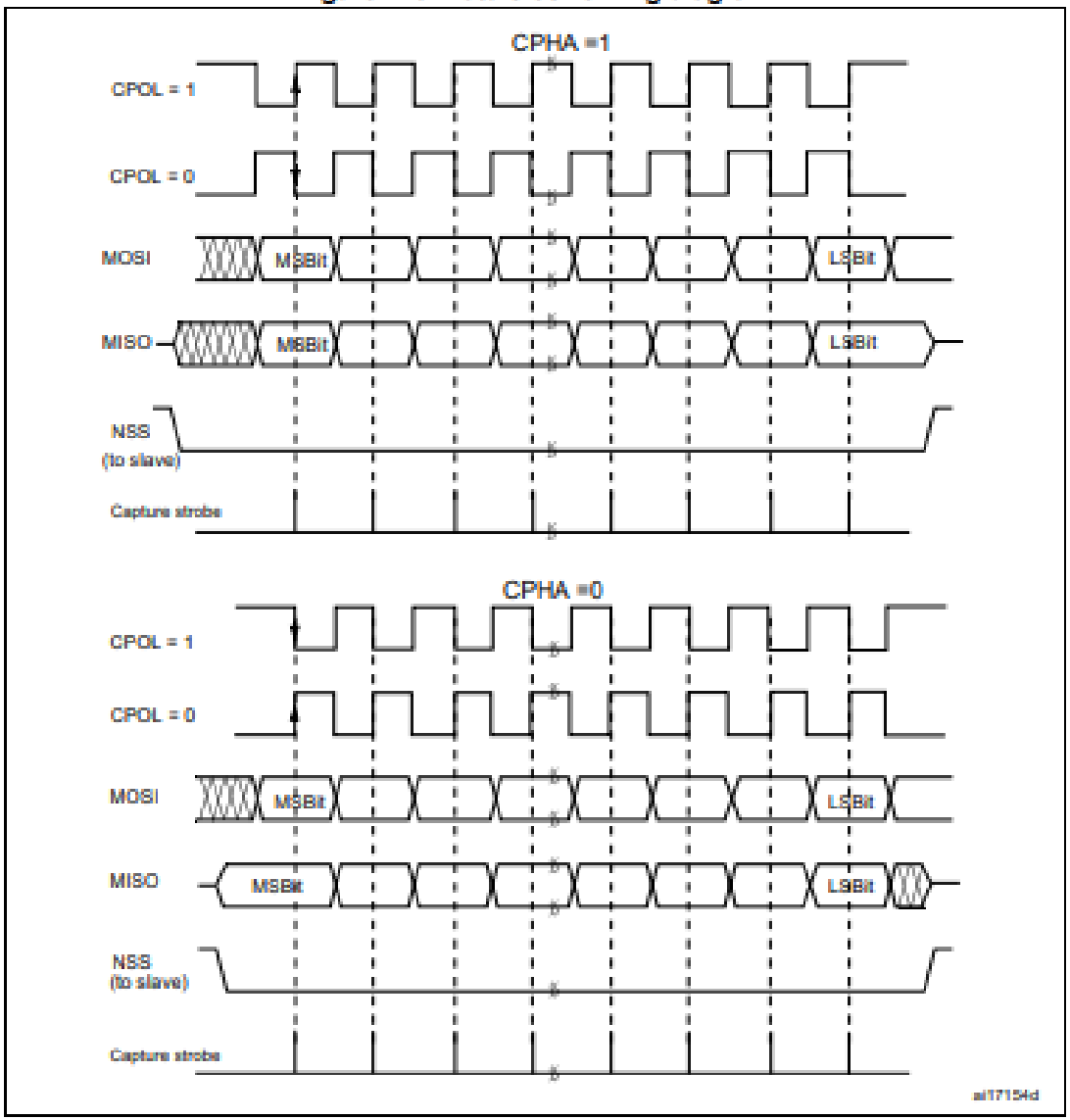


Figure 240. Data clock timing diagram



1. These timings are shown with the LSBFIRST bit reset in the SPI_CR1 register.

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 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 13 **CRCEN**: Hardware CRC calculation enable

0: CRC calculation disabled

1: CRC calculation enabled

Note: This bit should be written only when SPI is disabled (SPE = '0') for correct operation.

It is not used in I^2S mode.

Bit 12 **CRCNEXT**: CRC transfer next

0: Data phase (no CRC phase)

1: Next transfer is CRC (CRC phase)

Note: When the SPI is configured in full duplex or transmitter only modes, CRCNEXT must be written as soon as the last data is written to the SPI_DR register.

When the SPI is configured in receiver only mode, CRCNEXT must be set after the second last data reception.

This bit should be kept cleared when the transfers are managed by DMA.

It is not used in I^2S mode.

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

Note: This bit should be written only when SPI is disabled (SPE = '0') for correct operation.

It is not used in I^2S 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)

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

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

Bit 7 **LSBFIRST:** Frame format

0: MSB transmitted first

1: LSB transmitted first

Note: This bit should not be changed when communication is ongoing. It is not used in I^2S mode

Bit 6 **SPE:** SPI enable

0: Peripheral disabled

1: Peripheral enabled

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

When disabling the SPI, follow the procedure described in Section 25.3.8.

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: $f_{PCLK}/256$

Note: These bits should not be changed when communication is ongoing. They are not used in I^2S mode.

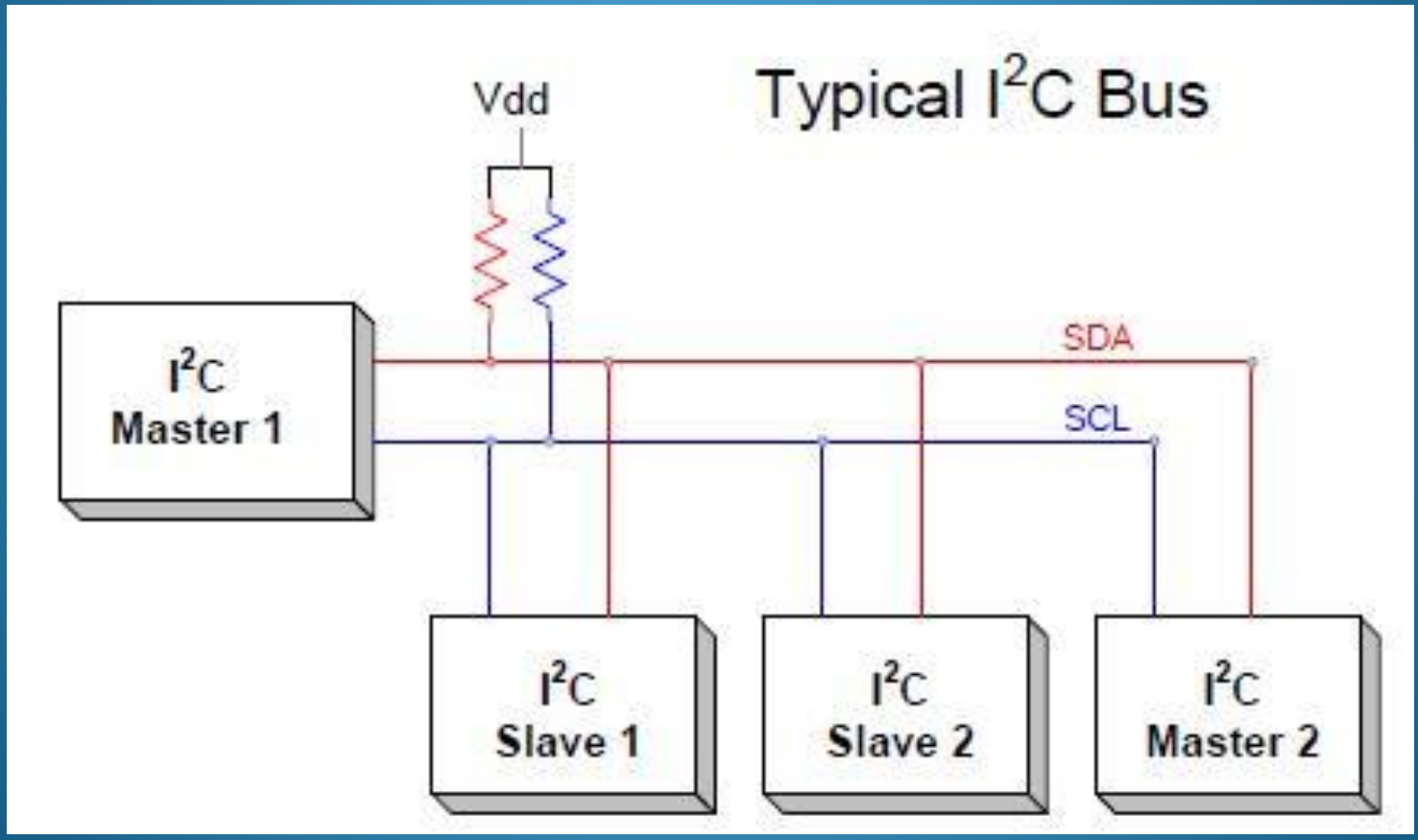
Bit 2 **MSTR:** Master selection

0: Slave configuration

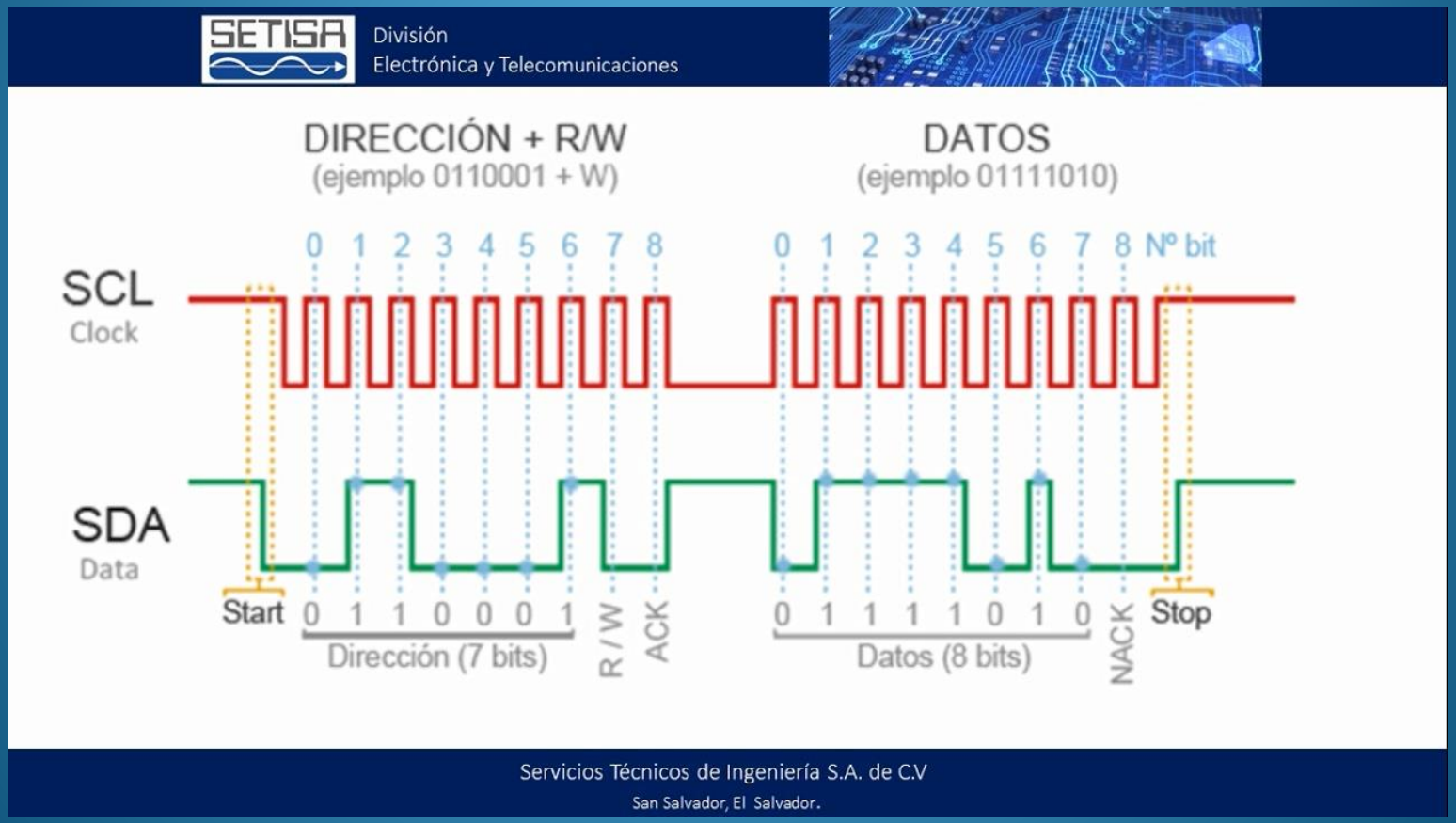
1: Master configuration

Note: This bit should not be changed when communication is ongoing. It is not used in I^2S mode.

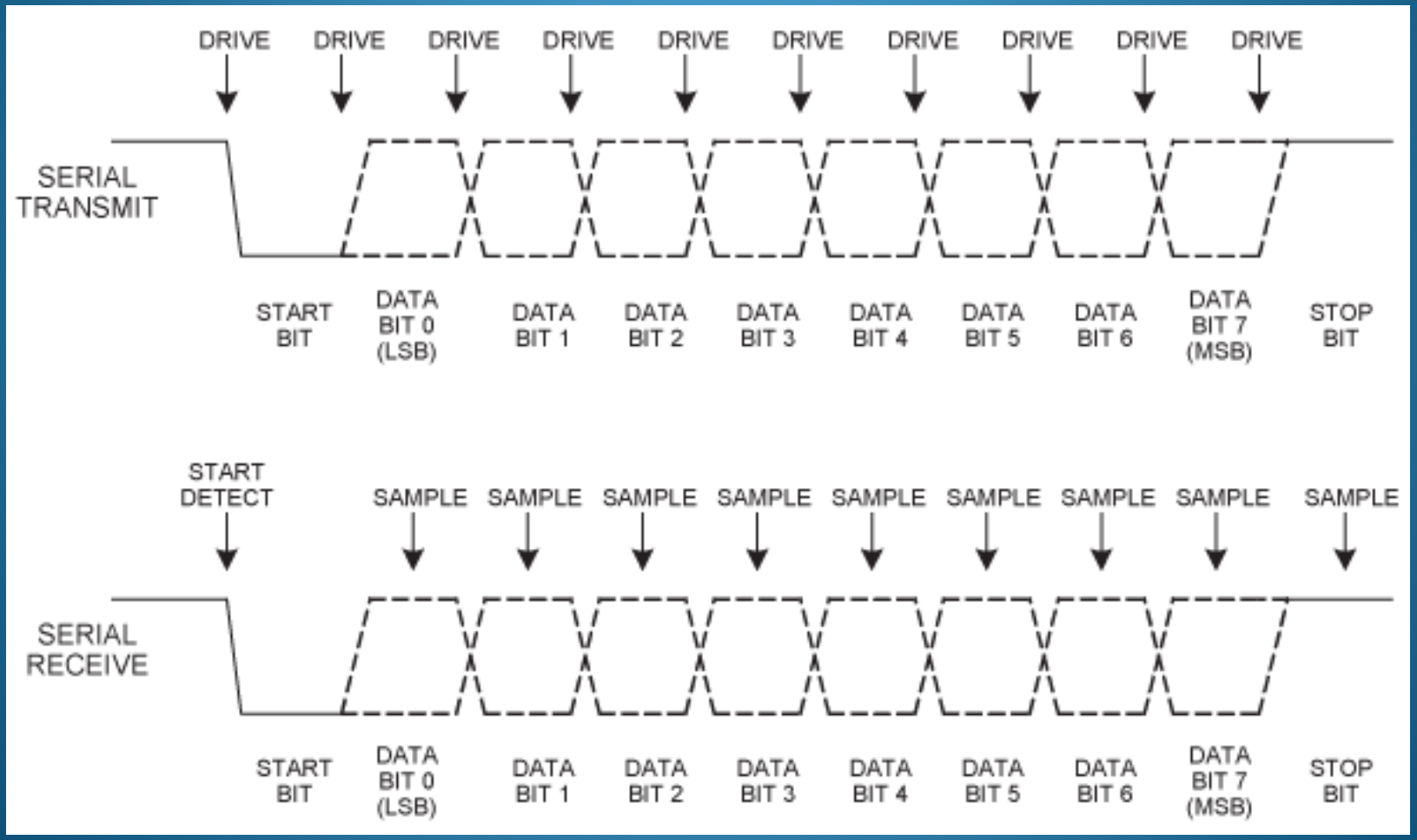
Comunicaciones a nivel IC
I²C



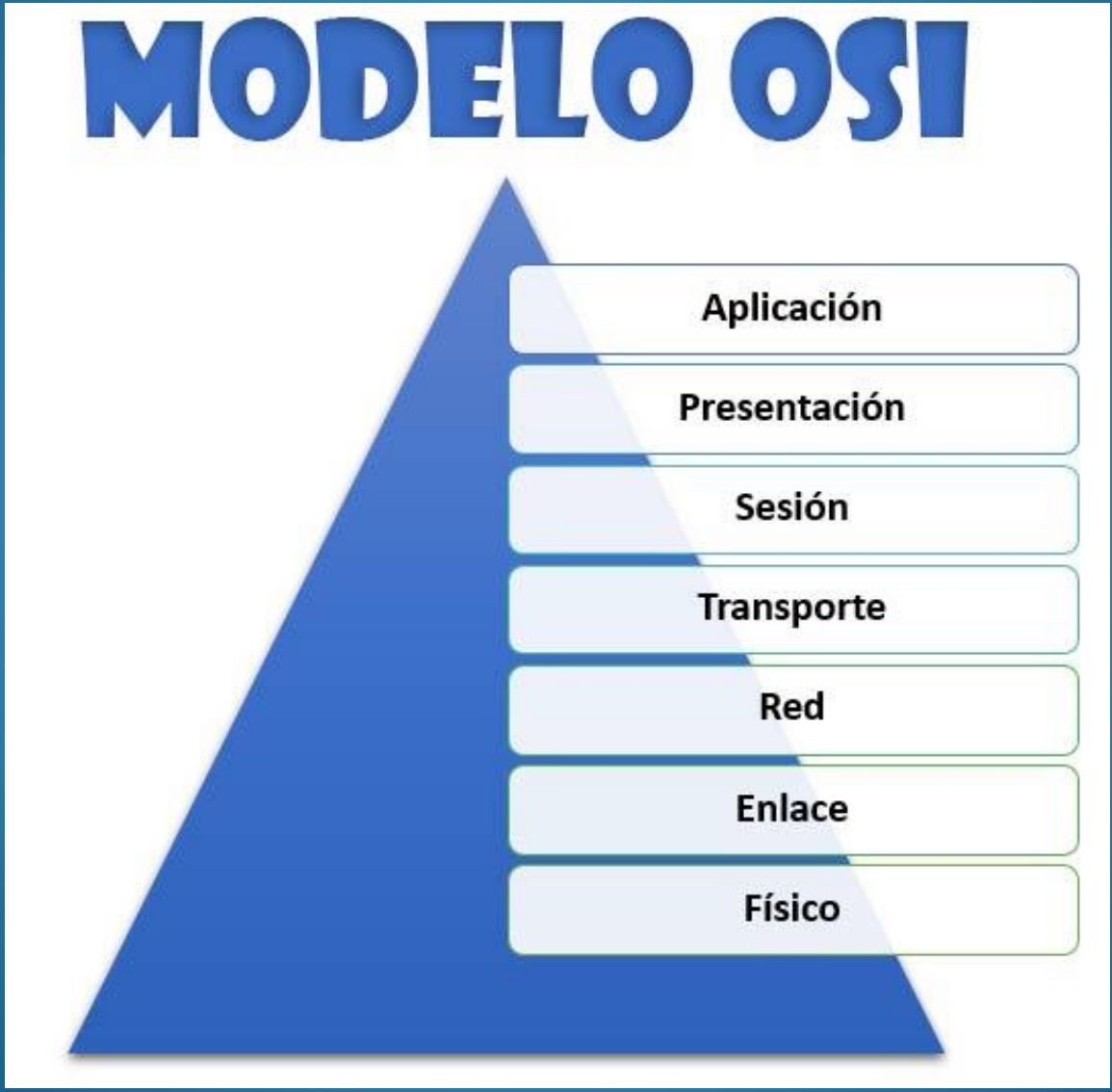
Comunicaciones a nivel IC
I^2C



Comunicaciones a nivel IC
USART



Modelo OSI


MODELO OSI
Aplicación
Presentación
Sesión
Transporte
Red
Enlace
Físico

Esquema General MODBUS

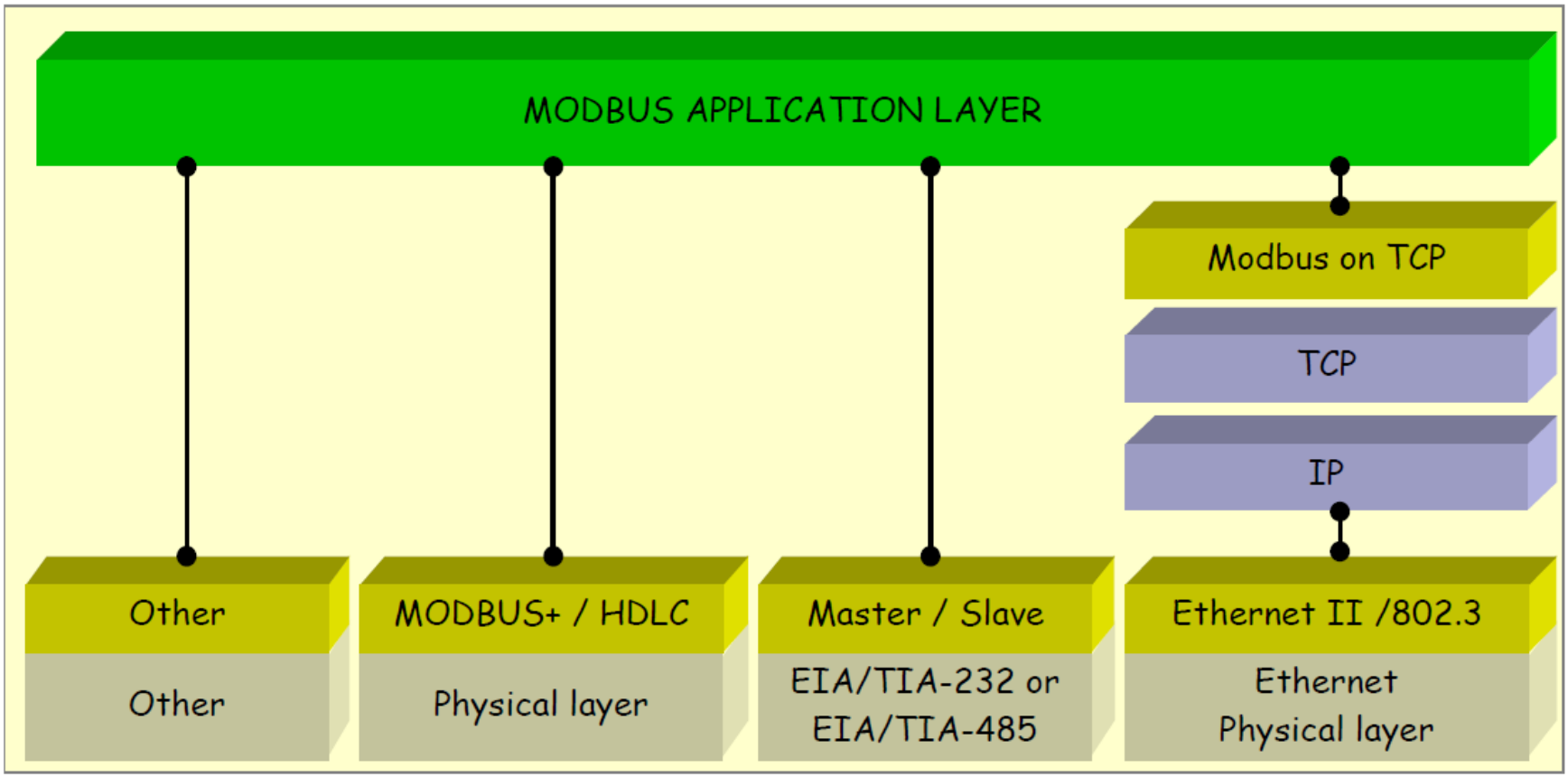


Figure 1: MODBUS communication stack

Esquema General

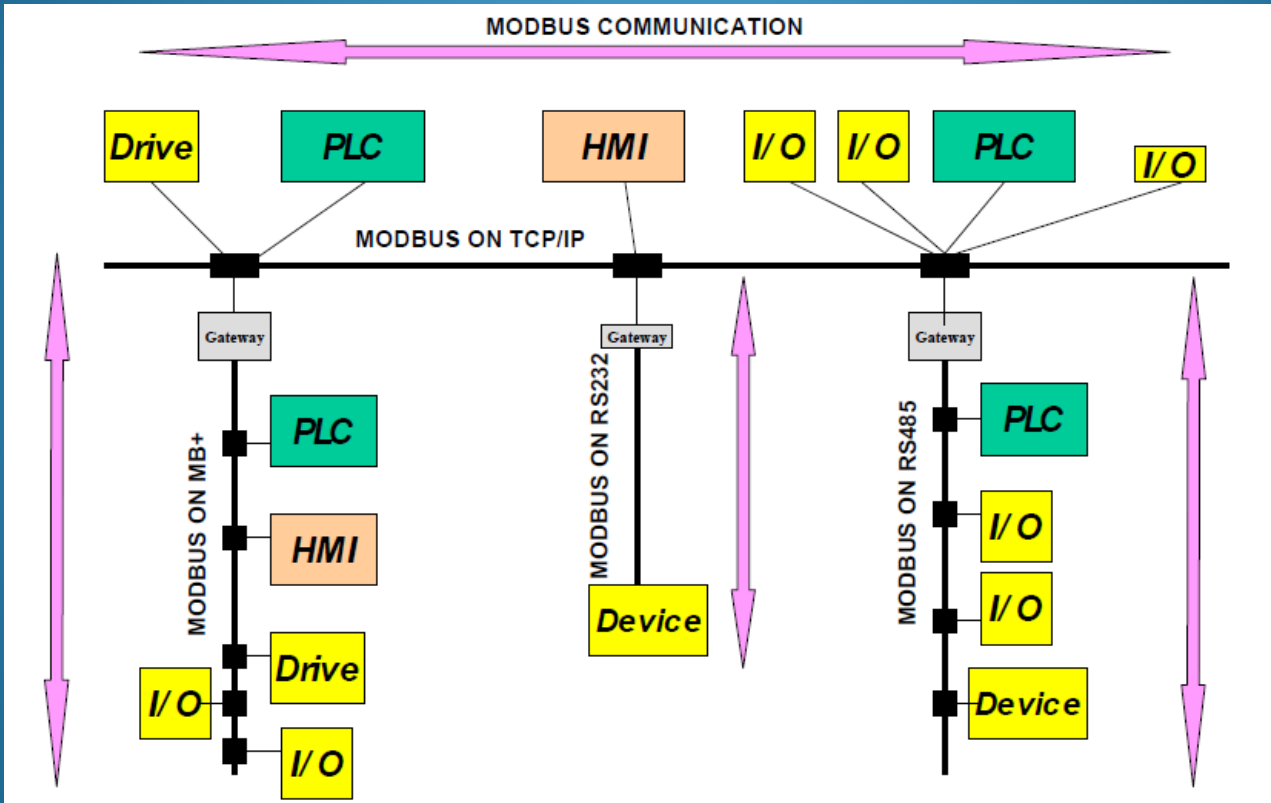
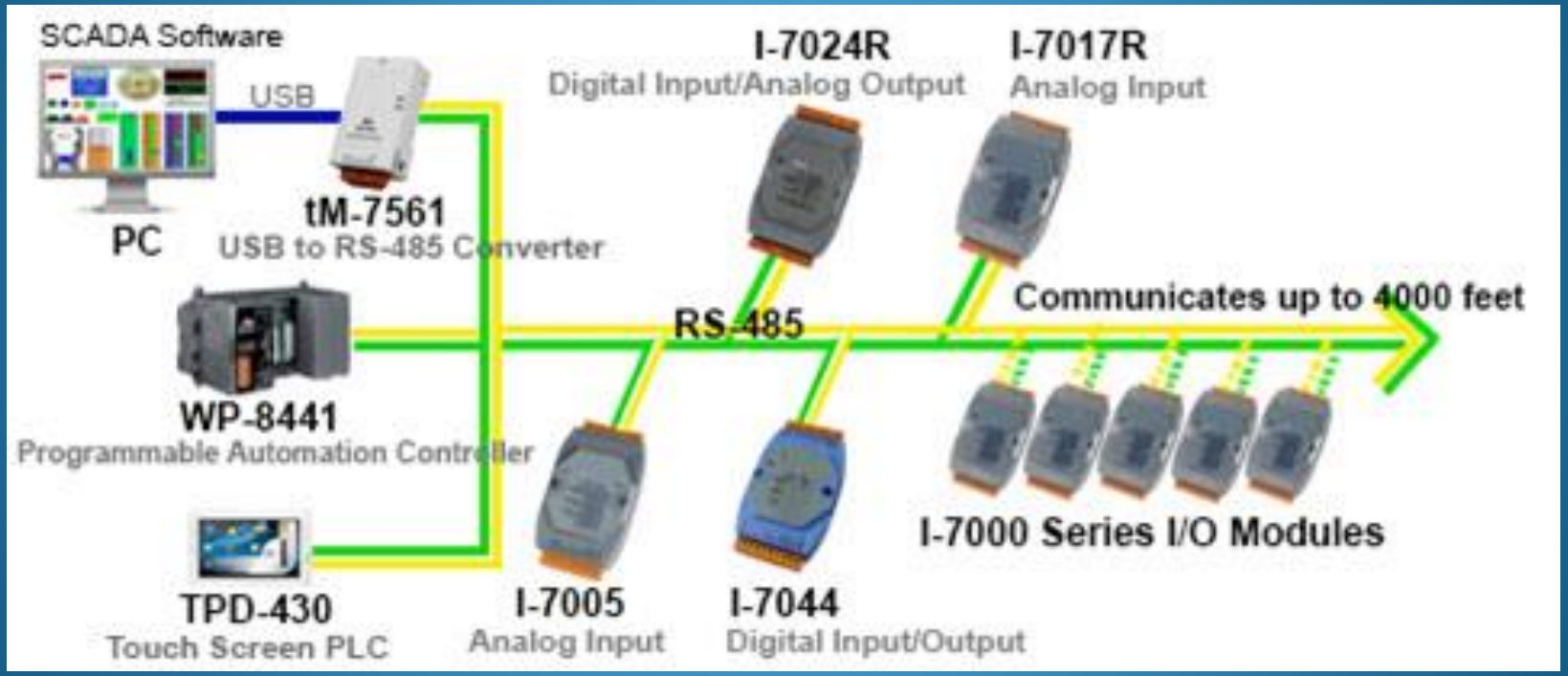
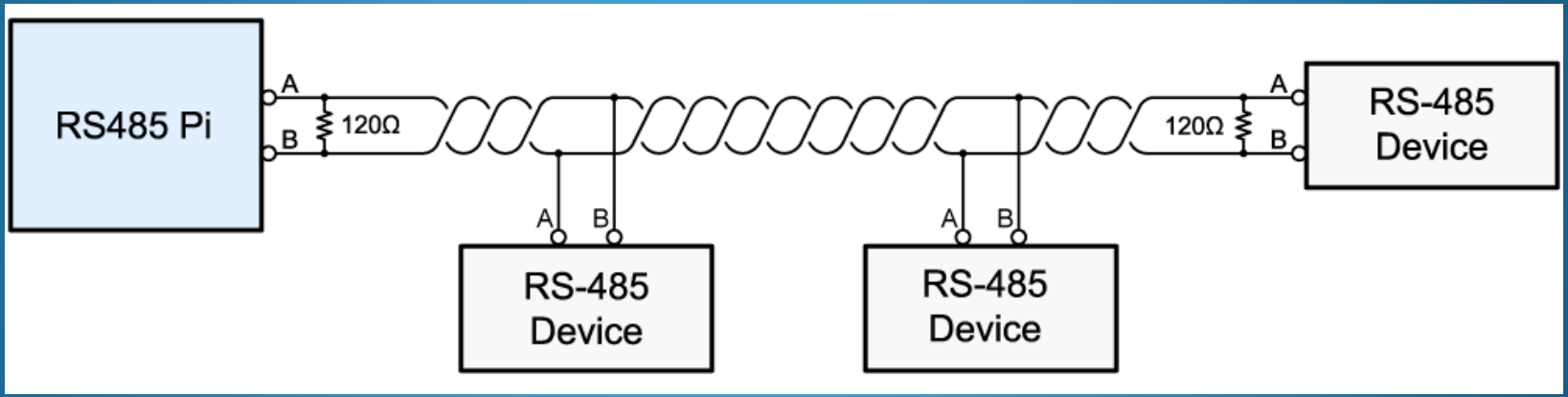


Figure 2: Example of MODBUS Network Architecture

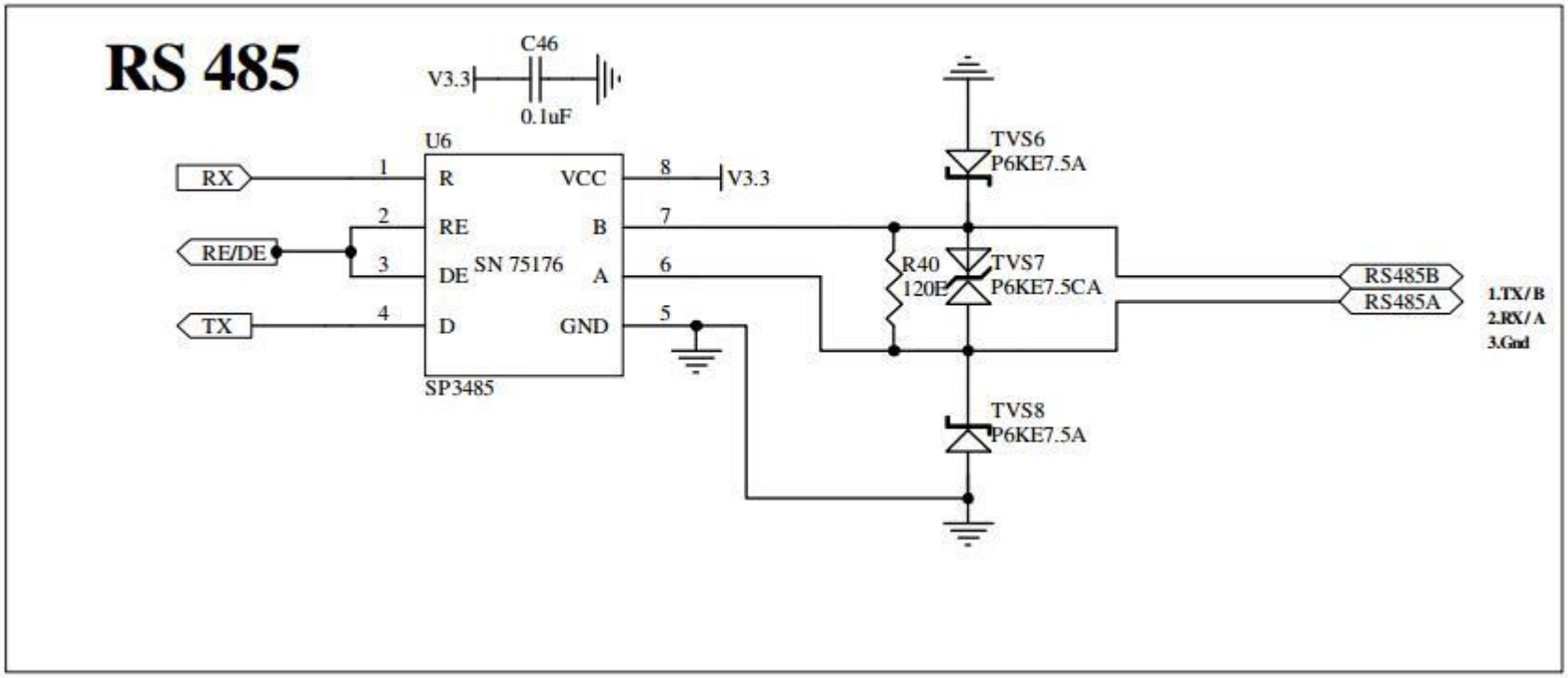
Esquema General



Interfaz RS485



DS75176


RS 485
C46
V3.3
0.1uF
U6
RX
RE/DE
TX
1
2
3
4
R
RE
DE
D
SN 75176
VCC
B
A
GND
8
7
6
5
V3.3
SP3485
TVS6
P6KE7.5A
R40
120E
TVS7
P6KE7.5CA
TVS8
P6KE7.5A
RS485B
RS485A
1.TX/B
2.RX/A
3.Gnd

Comunicación

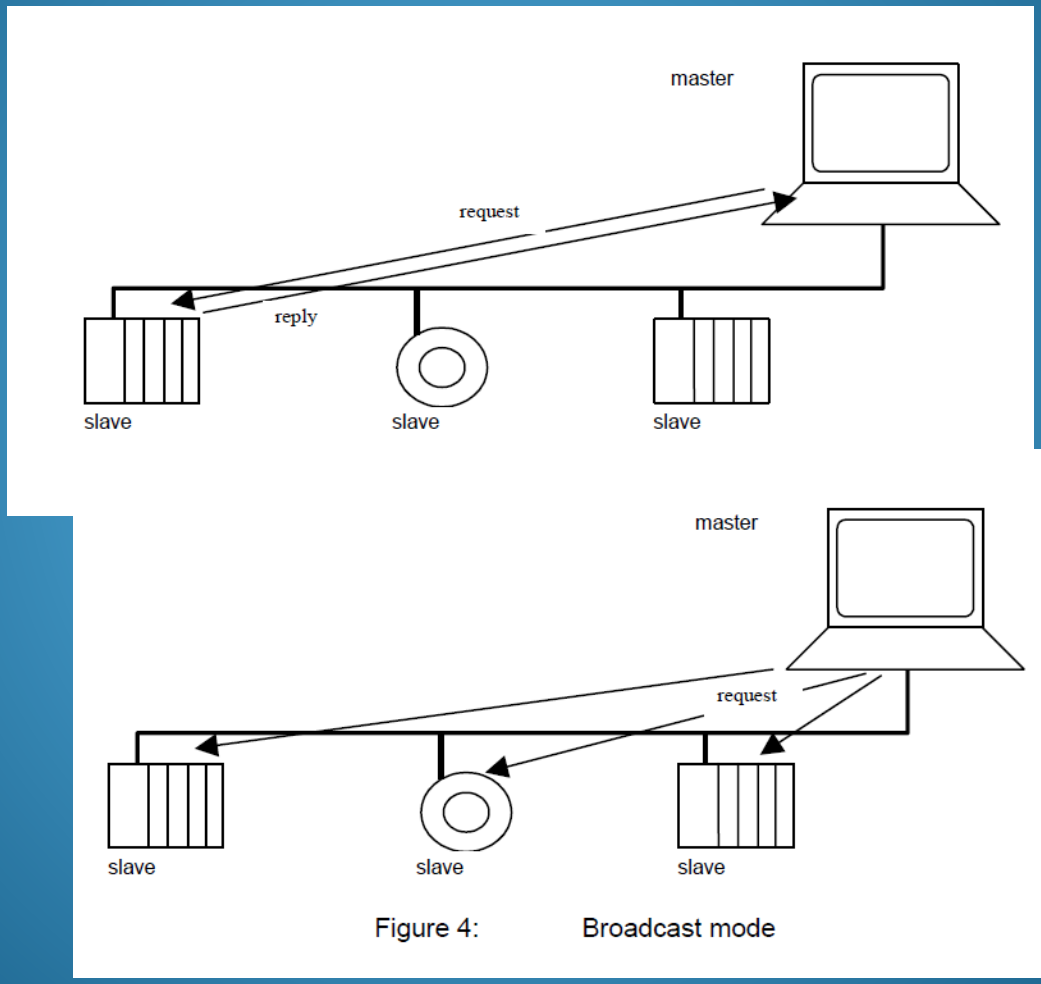


Figure 4: Broadcast mode

Comunicación

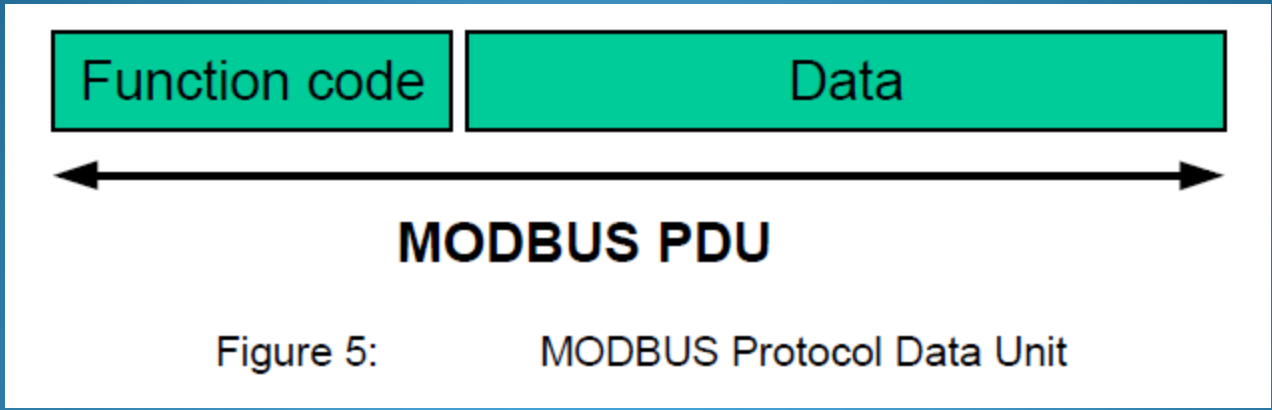
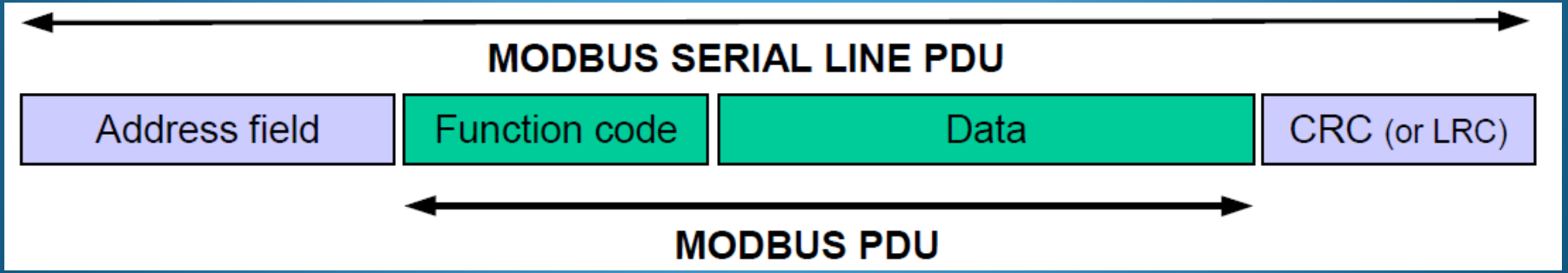


Figure 5: MODBUS Protocol Data Unit



Comunicación

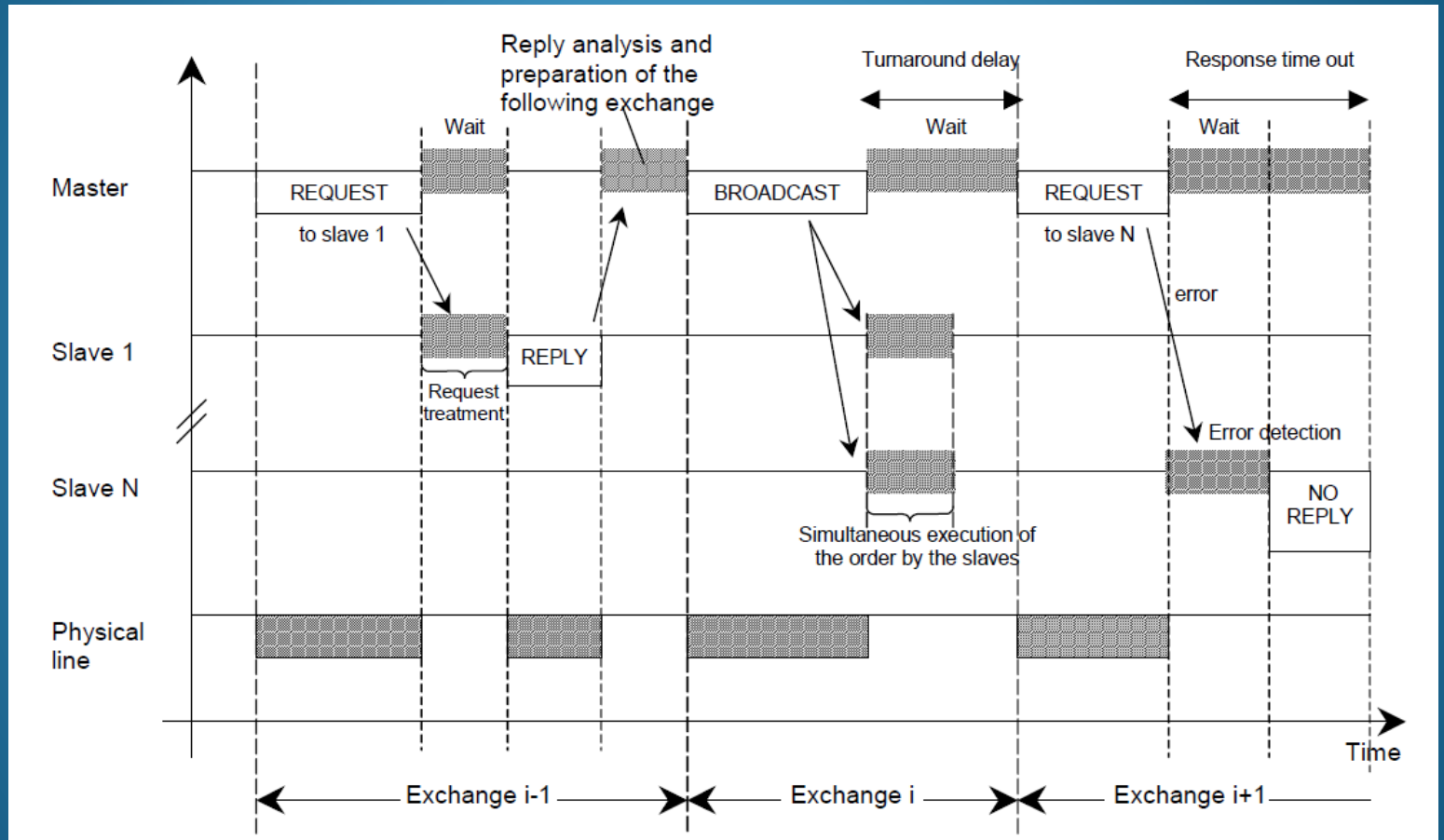
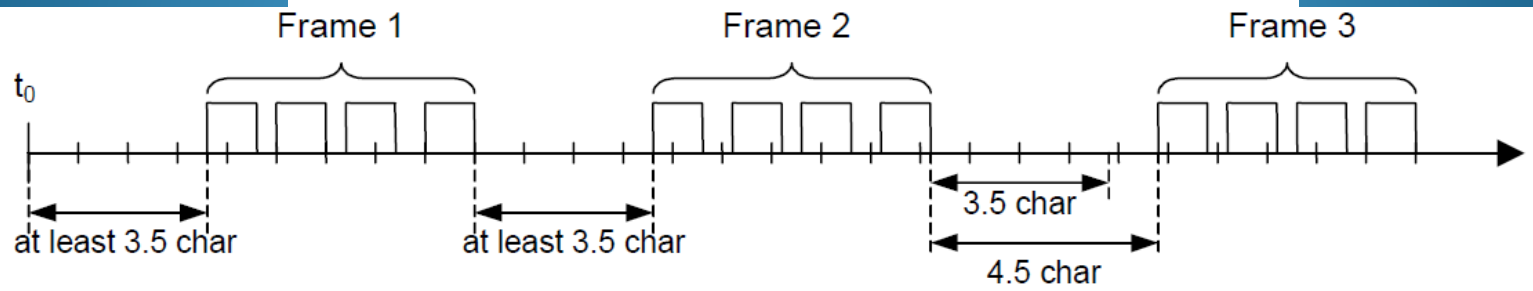


Figure 9: Master / Slave scenario time diagram

Comunicación

Slave Address	Function Code	Data	CRC
1 byte	1 byte	0 up to 252 byte(s)	2 bytes CRC Low, CRC Hi

Figure 12: RTU Message Frame



MODBUS message

Start	Address	Function	Data	CRC Check	End
≥ 3.5 char	8 bits	8 bits	N x 8 bits	16 bits	≥ 3.5 char

Figure 13: RTU Message Frame

Serial BUS

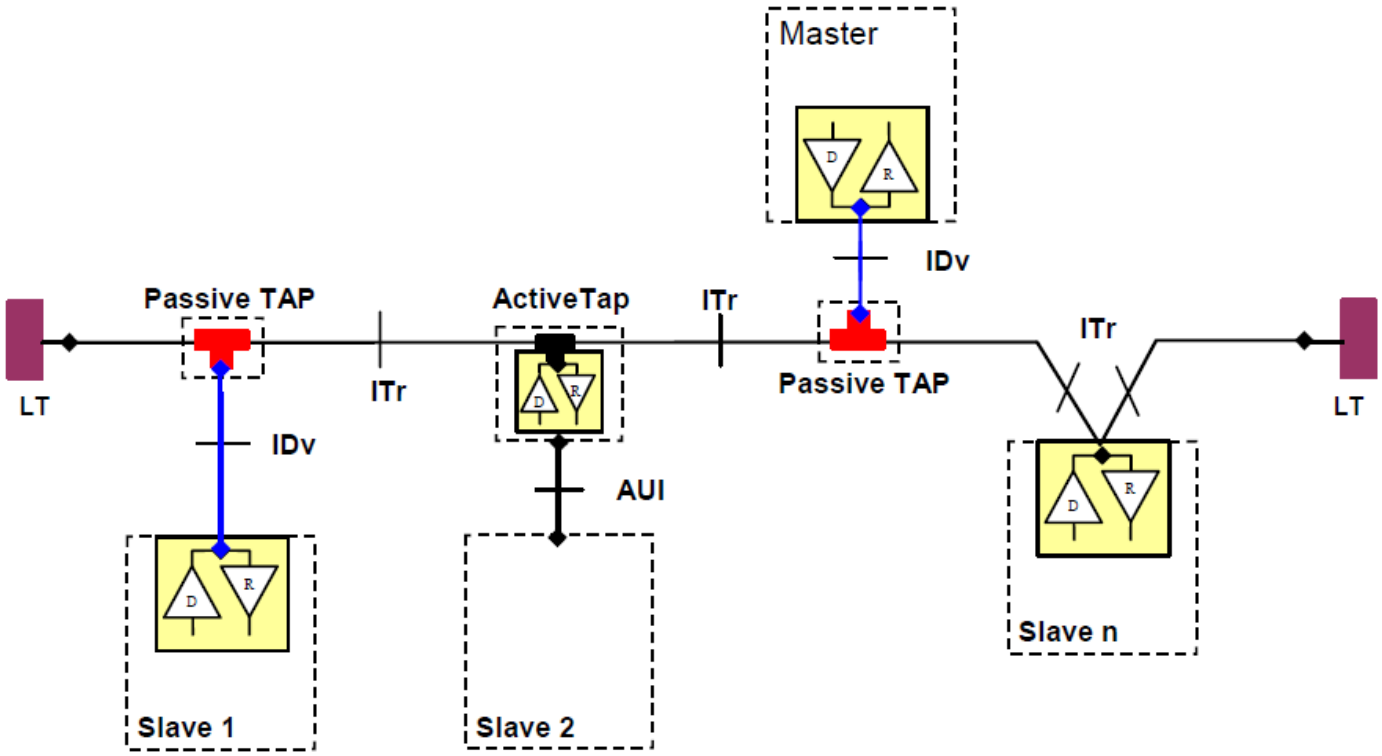


Figure 19 : Serial bus infrastructure

Registros Generales

Commonly used public function codes				
Code	Hex	Function	Type	
01	01	Read Coils	Single Bit Access	Data Access
02	02	Read Discrete Inputs		
05	05	Write Single Coil		
15	0F	Write Multiple Coils		
03	03	Read Holding Registers	16 bit Access	
04	04	Read Input Register		
06	06	Write Single Register		
16	10	Write Multiple Registers		
22	16	Mask Write Register		
23	17	Read/Write Multiple Registers		
24	18	Read FIFO queue		
20	14	Read File Record	File record access	
21	15	Write File Recore		
07	07	Read Exception Status	Diagnostics	
08	08	Diagnostic		
11	0B	Get Com event counter		
12	0C	Get Com Event Log		
17	11	Report Server ID		

Registros Generales

				Function Codes			
				code	*Sub code*	*(hex)*	*Section*
Data Access	**Bit access**	Physical Discrete Inputs	Read Discrete Inputs	**02**		**02**	6.2
		Internal Bits Or Physical coils	Read Coils	**01**		**01**	6.1
			Write Single Coil	**05**		**05**	6.5
			Write Multiple Coils	**15**		**0F**	6.11
	16 bits access	Physical Input Registers	Read Input Register	**04**		**04**	6.4
		Internal Registers Or Physical Output Registers	Read Holding Registers	**03**		**03**	6.3
			Write Single Register	**06**		**06**	6.6
			Write Multiple Registers	**16**		**10**	6.12
			Read/Write Multiple Registers	**23**		**17**	6.17
			Mask Write Register	**22**		**16**	6.16
			Read FIFO queue	**24**		**18**	6.18
	File record access		Read File record	**20**		**14**	6.14
			Write File record	**21**		**15**	6.15
Diagnostics			Read Exception status	**07**		**07**	6.7
			Diagnostic	**08**	**00-18,20**	**08**	6.8
			Get Com event counter	**11**		**OB**	6.9
			Get Com Event Log	**12**		**0C**	6.10
			Report Slave ID	**17**		**11**	6.13
			Read device Identification	**43**	**14**	**2B**	6.21
Other			Encapsulated Interface Transport	**43**	**13,14**	**2B**	6.19

01 (0x01) Read Coils

Request

Function code	1 Byte	**0x01**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of coils	2 Bytes	1 to 2000 (0x7D0)

Response

Function code	1 Byte	**0x01**
Byte count	1 Byte	**N***
Coil Status	**n** Byte	n = N or N+1

***N** = Quantity of Outputs / 8, if the remainder is different of 0 ⇒ N = N+1

Error

Function code	1 Byte	**Function code + 0x80**
Exception code	1 Byte	01 or 02 or 03 or 04

Here is an example of a request to read discrete outputs 20–38:

Request		**Response**	
Field Name	*(Hex)*	*Field Name*	*(Hex)*
Function	**01**	Function	**01**
Starting Address Hi	**00**	Byte Count	**03**
Starting Address Lo	**13**	Outputs status 27-20	**CD**
Quantity of Outputs Hi	**00**	Outputs status 35-28	**6B**
Quantity of Outputs Lo	**13**	Outputs status 38-36	**05**

03 (0x03) Read Holding Registers

Request

Function code	1 Byte	**0x03**
Starting Address	2 Bytes	0x0000 to 0xFFFF
Quantity of Registers	2 Bytes	1 to 125 (0x7D)

Response

Function code	1 Byte	**0x03**
Byte count	1 Byte	2 x **N***
Register value	**N*** x 2 Bytes	

***N** = Quantity of Registers

Error

Error code	1 Byte	**0x83**
Exception code	1 Byte	01 or 02 or 03 or 04

Here is an example of a request to read registers 108 – 110:

Request		**Response**	
Field Name	*(Hex)*	*Field Name*	*(Hex)*
Function	**03**	Function	**03**
Starting Address Hi	**00**	Byte Count	**06**
Starting Address Lo	**6B**	Register value Hi (108)	**02**
No. of Registers Hi	**00**	Register value Lo (108)	**2B**
No. of Registers Lo	**03**	Register value Hi (109)	**00**
		Register value Lo (109)	**00**
		Register value Hi (110)	**00**
		Register value Lo (110)	**64**

05 (0x05) Write Single Coil

Request

Function code	1 Byte	**0x05**
Output Address	2 Bytes	0x0000 to 0xFFFF
Output Value	2 Bytes	0x0000 or 0xFF00

Response

Function code	1 Byte	**0x05**
Output Address	2 Bytes	0x0000 to 0xFFFF
Output Value	2 Bytes	0x0000 or 0xFF00

Error

Error code	1 Byte	**0x85**
Exception code	1 Byte	01 or 02 or 03 or 04

Here is an example of a request to write Coil 173 ON:

Request		**Response**	
Field Name	*(Hex)*	*Field Name*	*(Hex)*
Function	**05**	Function	**05**
Output Address Hi	**00**	Output Address Hi	**00**
Output Address Lo	**AC**	Output Address Lo	**AC**
Output Value Hi	**FF**	Output Value Hi	**FF**
Output Value Lo	**00**	Output Value Lo	**00**

Redes de datos CAN

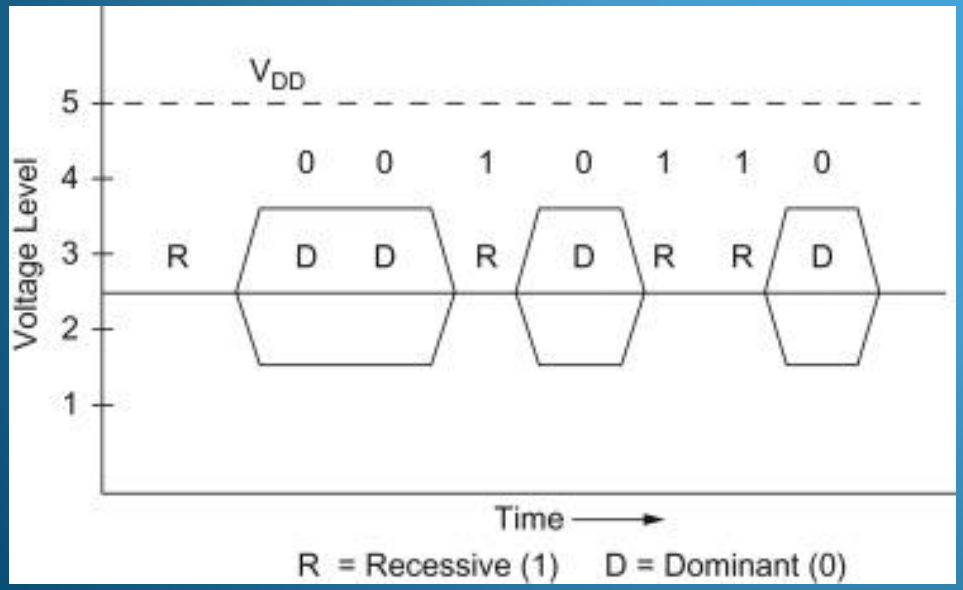
Capa física: Se encarga de las conexiones físicas de la red, especifica los niveles o características eléctricas de las señales así como la codificación, decodificación, temporización y sincronización de los bits del frame.

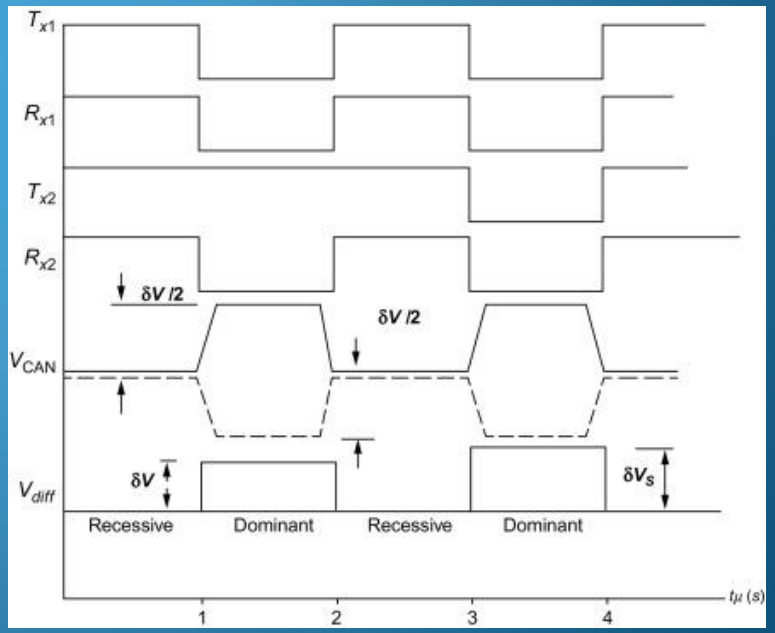
Capa de enlace de datos: Controla el flujo de información entre los nodos de la red. Es decir, se encarga de la transmisión de los bits en frames de información, se ocupa de que los mensajes lleguen a su destino sin errores, controla las secuencias de transmisión, los acuses de recibido y si en determinado caso no se recibió bien un mensaje se encarga de retransmitirlo.

http://catarina.udlap.mx/u_dl_a/tales/documentos/lmt/pacheco_h_je/capitulo2.pdf

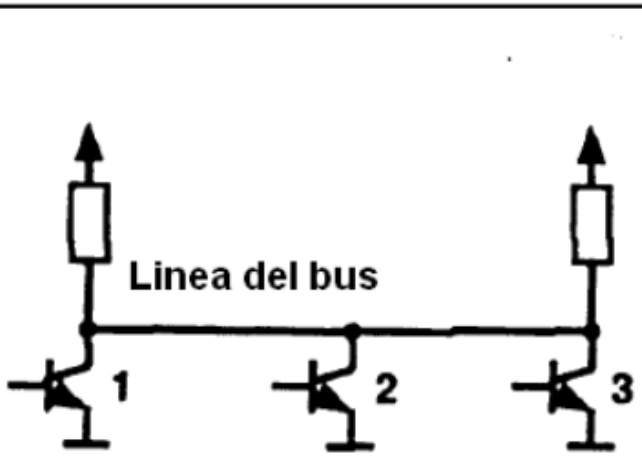
Redes de datos CAN

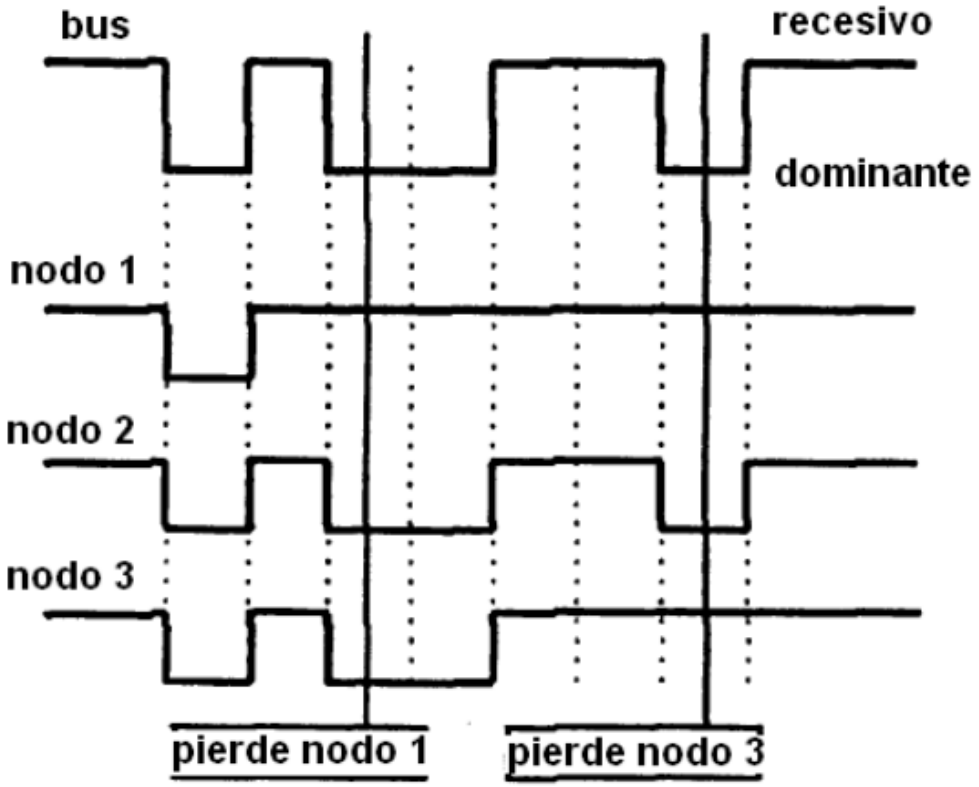
Bit dominante: Un bit dominante es un “0” lógico presente en algún nodo de la red. Es nombrado bit dominante debido a que hace desaparecer los “1” lógicos en la red.
Bit recesivo: Un bit recesivo es un “1” lógico.



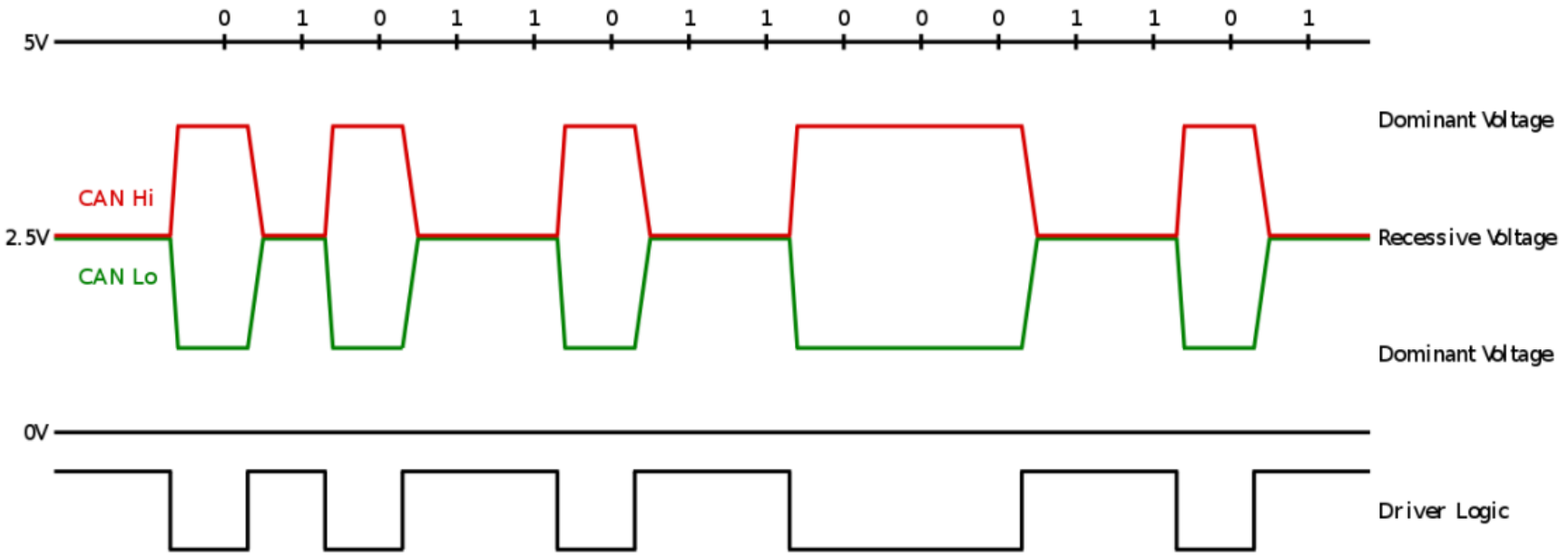


Arbitraje No destructivo

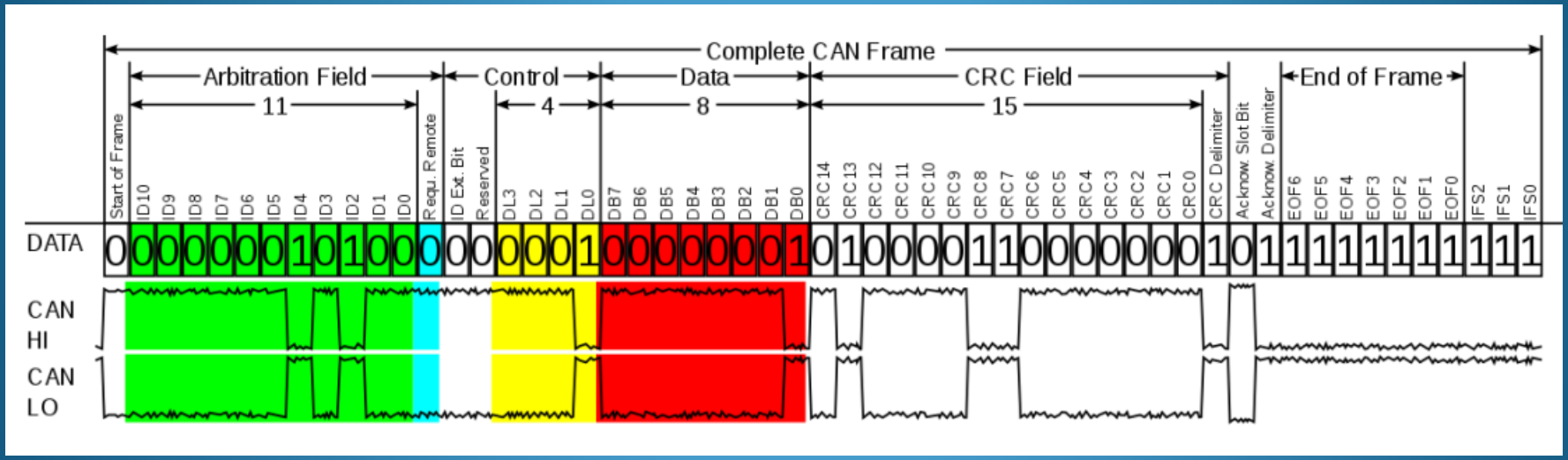




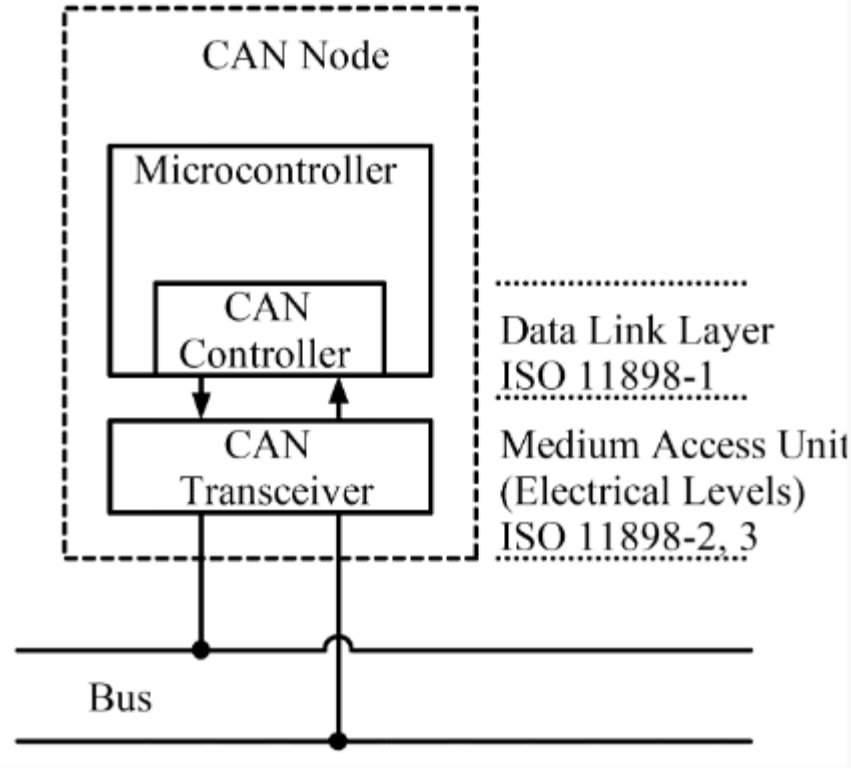
Niveles de tensor



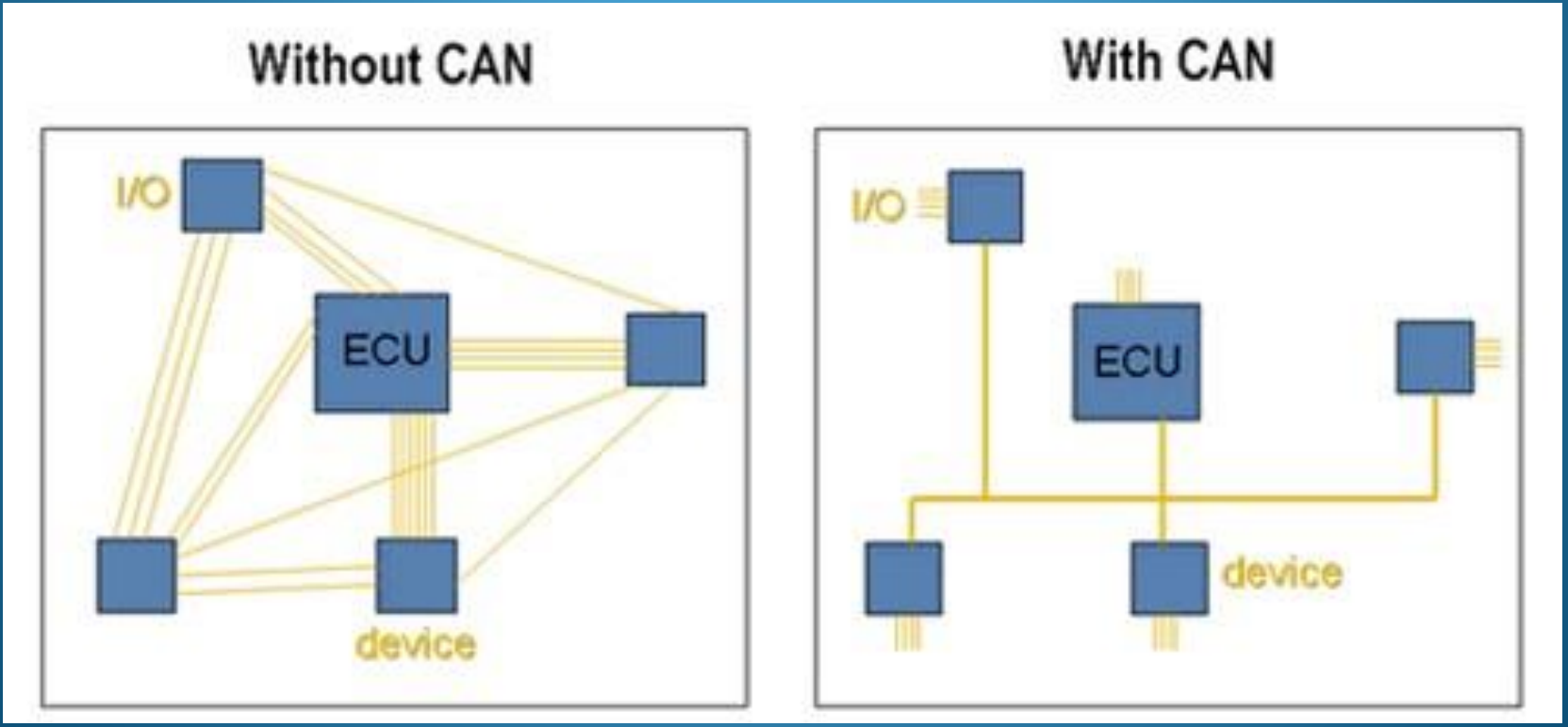
Trama CAN


Complete CAN Frame
Arbitration Field
11
Control
4
Data
8
CRC Field
15
End of Frame
Start of Frame
ID10 ID9 ID8 ID7 ID6 ID5 ID4 ID3 ID2 ID1 ID0
Requ. Remote
ID Ext. Bit
Reserved
DL3 DL2 DL1 DL0
DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0
CRC14 CRC13 CRC12 CRC11 CRC10 CRC9 CRC8 CRC7 CRC6 CRC5 CRC4 CRC3 CRC2 CRC1 CRC0
CRC Delimiter
Acknow. Slot Bit
Acknow. Delimiter
EOF6 EOF5 EOF4 EOF3 EOF2 EOF1 EOF0
IFS2 IFS1 IFS0
DATA
0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 1 1 1 1 1 1 1 1 1 1
CAN HI
CAN LO

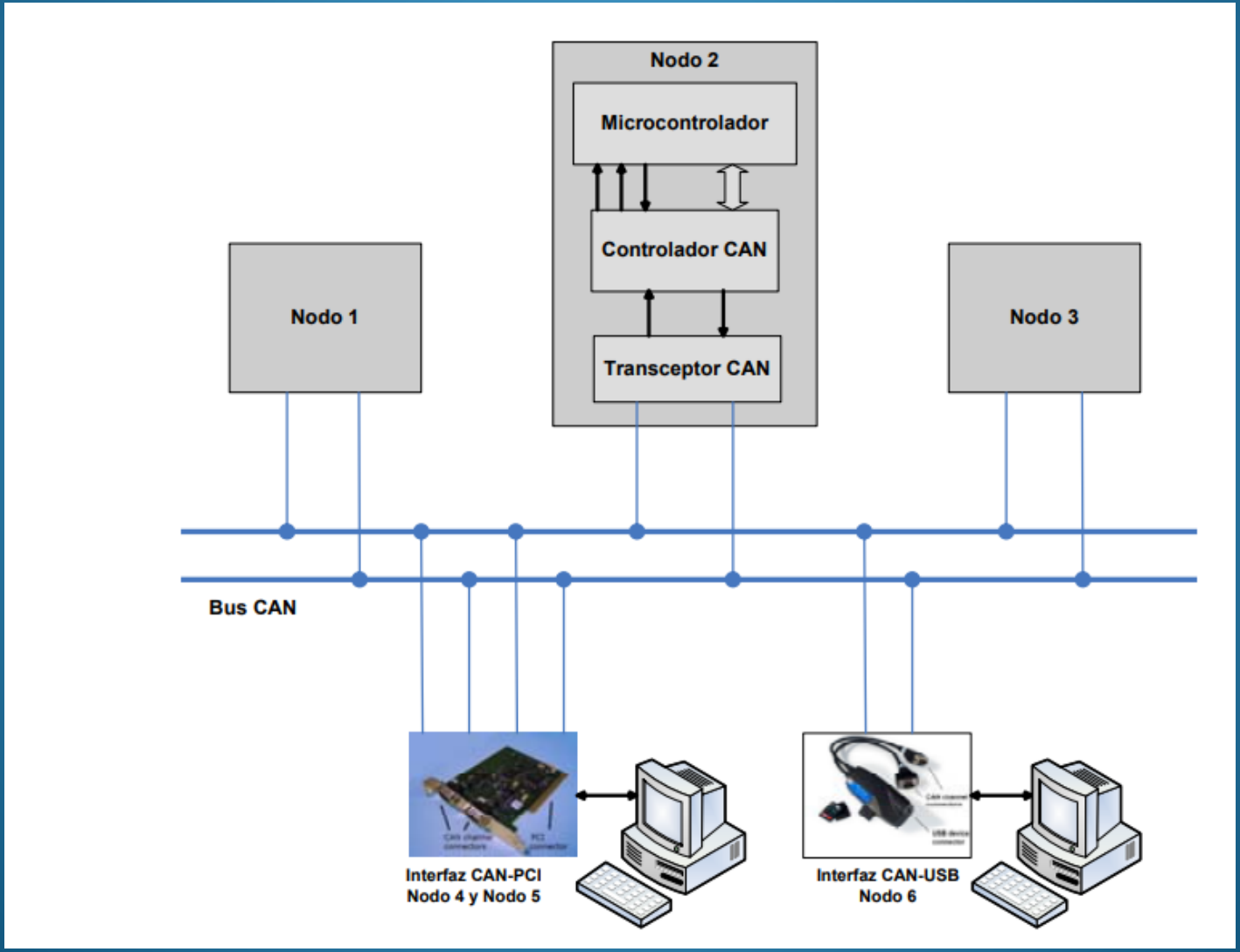
Redes de datos CAN



Redes de datos CAN
Porque la CAN?

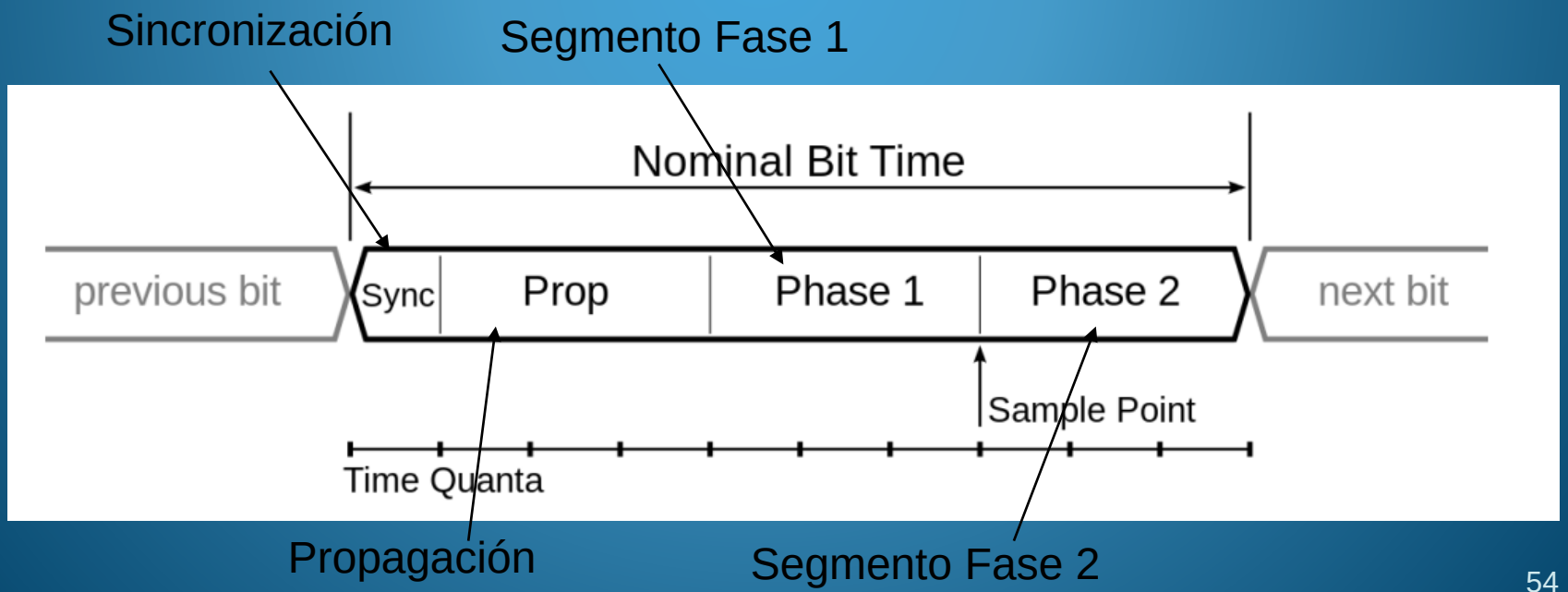


Red CAN

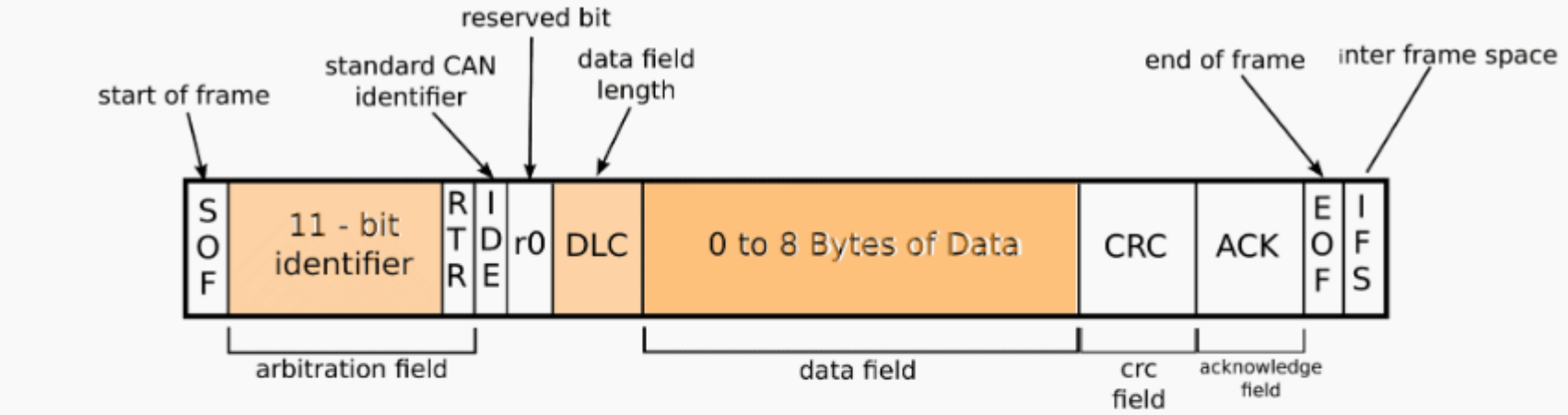

Nodo 2
Microcontrolador
Controlador CAN
Transceptor CAN
Nodo 1
Nodo 3
Bus CAN
Interfaz CAN-PCI
Nodo 4 y Nodo 5
Interfaz CAN-USB
Nodo 6

Trama CAN

El controlador de bus CAN espera que una transición del bus de recesivo a dominante ocurra en un determinado intervalo de tiempo. Si la transición no ocurre en el intervalo esperado, el controlador reajusta la duración del siguiente bit en consecuencia. Dicho ajuste se lleva a cabo dividiendo cada bit en intervalos o cuantos de tiempo y asignando los intervalos a los cuatro segmentos de cada bit: sincronización, propagación, segmento de fase 1 y segmento de fase 2.



Protocolo CAN



Marco estándar y extendido de la arquitectura de mensajes de datos CAN

Protocolo CAN

Campo	Bits	Descripción
SOF	1	El único comienzo dominante del cuadro. Este bit marca el comienzo de un mensaje. Sincroniza los nodos después de un período de inactividad.
Identifier	11	El campo de datos del identificador CAN de 11 bits establece la prioridad del mensaje. Los valores más bajos significan prioridades más altas.
RTR	1	Solicitud de transmisión remota. Este bit es dominante cuando otro nodo solicita información. Todos los nodos recibirán la solicitud, pero el identificador determina el nodo deseado.
IDE	1	El bit de extensión de identificador indica que se está transmitiendo un identificador CAN estándar (no uno extendido).
R0	1	Reservado para uso futuro.
DLC	4	El código de longitud de datos contiene el número de bytes en la transmisión.
Data	0 - 64	Los datos reales que se transmiten.
CRC	16	La verificación de redundancia cíclica (CRC) de 16 bits (15 bits más delimitador) contiene la suma de verificación (número de bits transmitidos) de los datos de la aplicación anterior para transmitir la detección de errores.
ACK	2	Cuando un nodo recibe un mensaje con éxito, ACKnowledings sobrescribe este bit con un bit dominante. Por otro lado, si un nodo encuentra un error en un mensaje, permite que este bit permanezca recesivo e ignora el mensaje. La ranura ACK y el delimitador ACK tienen cada uno un bit de longitud.

Mensaje CAN

EOF	7	End Of Frame es un campo de 7 bits que denota el final de cada trama CAN (mensaje).
IFS	3+	El espacio entre cuadros (IFS) es el tiempo que el controlador necesita para mover un cuadro (mensaje) a su posición en el área de búfer. Tenga en cuenta que IFS contiene un mínimo de tres bits recesivos (1) consecutivos. Después de que hayan pasado tres bits recesivos, cuando se detecta un bit dominante, se convierte en el bit SOF de la siguiente trama.

Variantes del BUS CAN

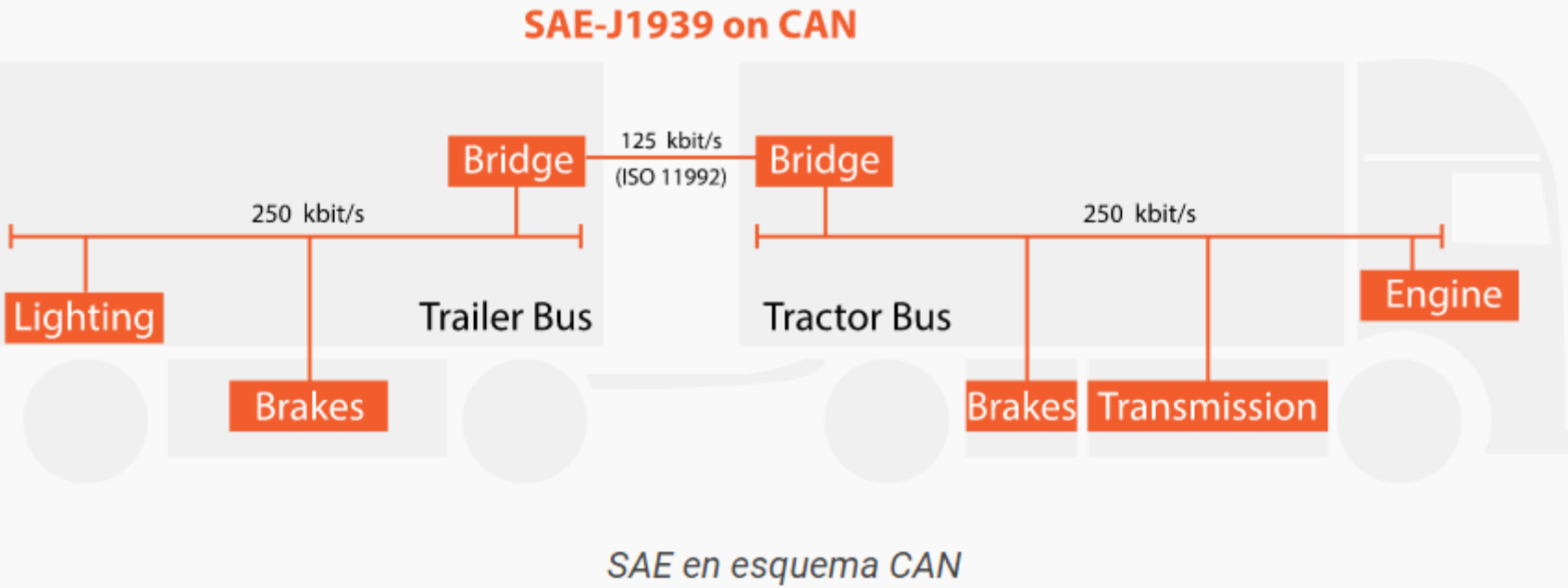
- CAN de baja velocidad
- CAN de alta velocidad
- CAN FD (CAN con velocidad de datos flexible)

125Kbps

1Mbps

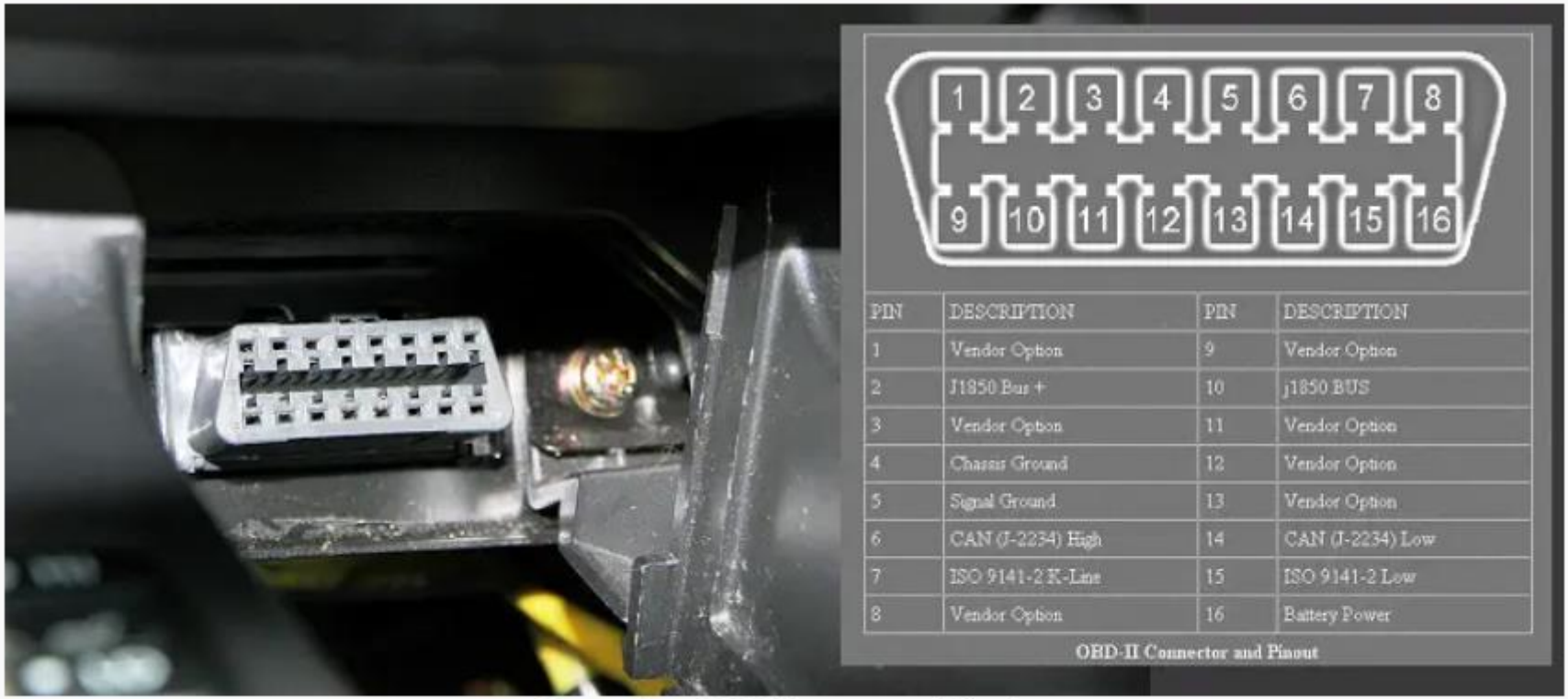
8 Mbps

Estándares y protocolos CAN



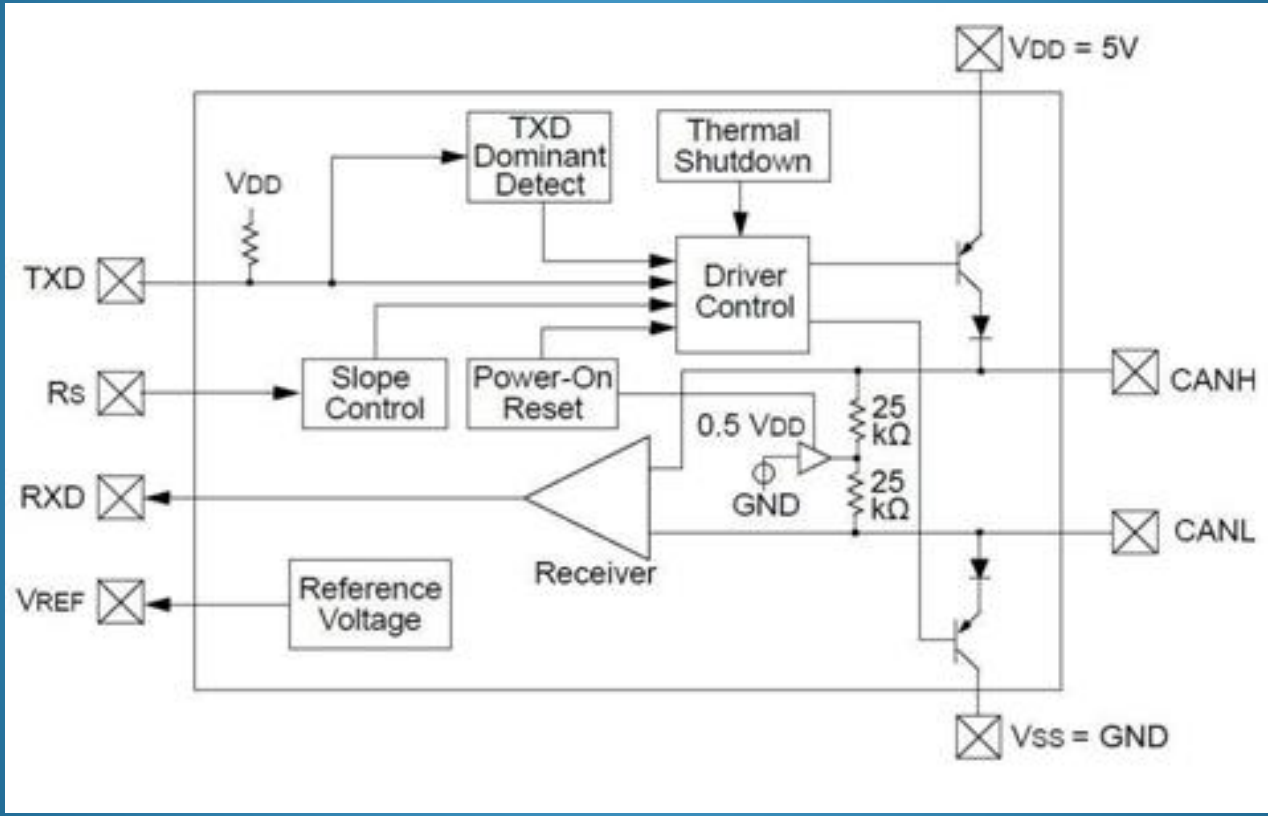
SAE en esquema CAN

ODBII



Conector OBD II en un vehículo

TRANSEIVER


VDD = 5V
TXD Dominant Detect
Thermal Shutdown
VDD
TXD
Driver Control
RS
Slope Control
Power-On Reset
CANH
0.5 VDD
25 kΩ
25 kΩ
GND
RXD
CANL
Receiver
VREF
Reference Voltage
VSS = GND

TRANSEIVER