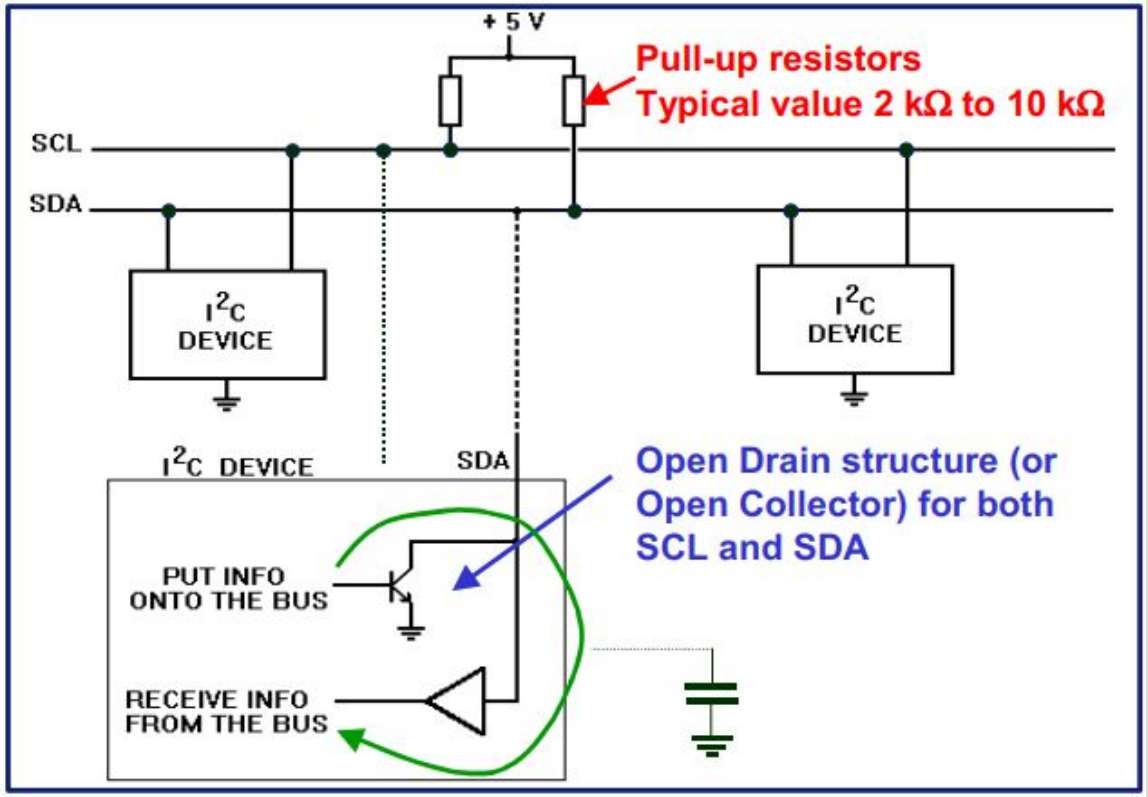
Передача данных

Институт Информационных Технологий

Челябинский Государственный Университет

I2C

- I2C = Inter-Integrated Circuit
- Разработана Philips в 80-х
- Простая двунаправленная шина из двух проводников:
  – линия данных(SDA)
  – линия тактов (SCL)
- Устройства делятся на два типа:
  – master
  – slave
- Устройство на шине идентифицируется с помощью собственного адресного кода
- Slave-устройство может быть:
  – только приемником данных
  – передатчиком с возможностью как получать, так и передавать данные



I2C

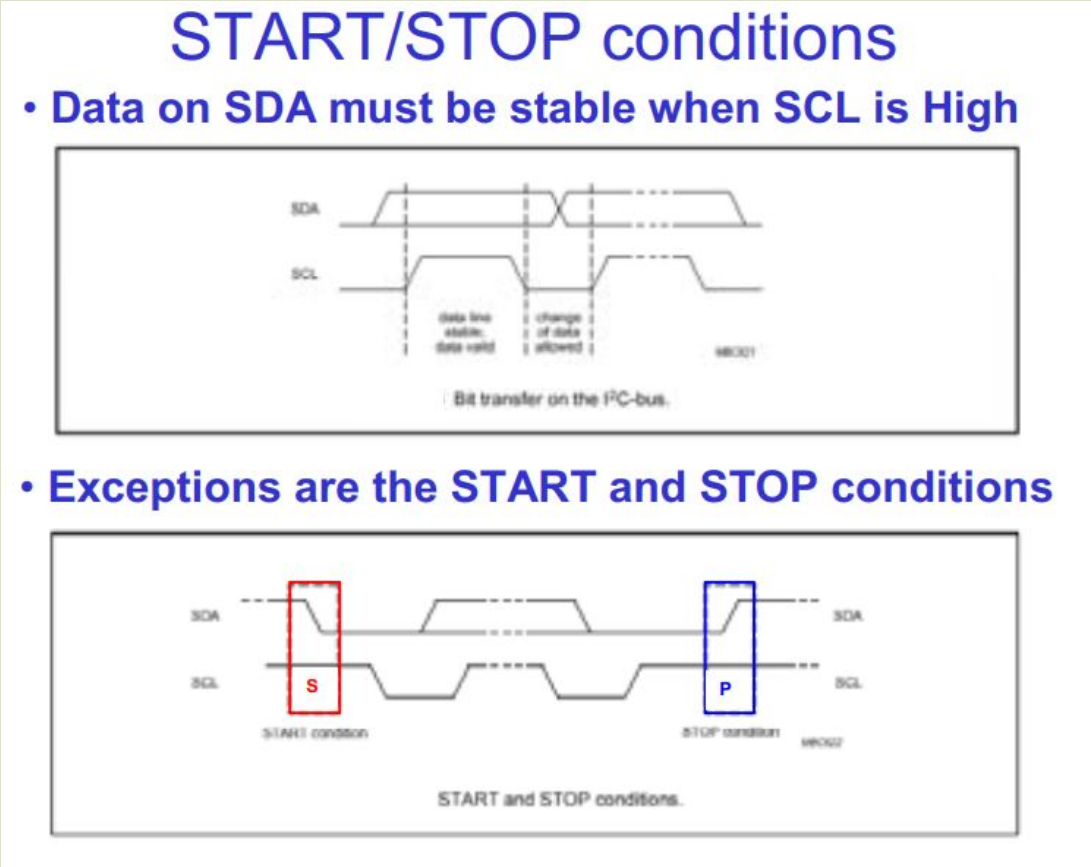
Состояние линии **SDA** не изменяется, если на **SCL** поддерживается **высокий** уровень за исключением начала и конца передачи:

START:

SDA переходит с **высокого** на **низкий** уровень в то время, когда на **SCL** поддерживается **высокий** уровень

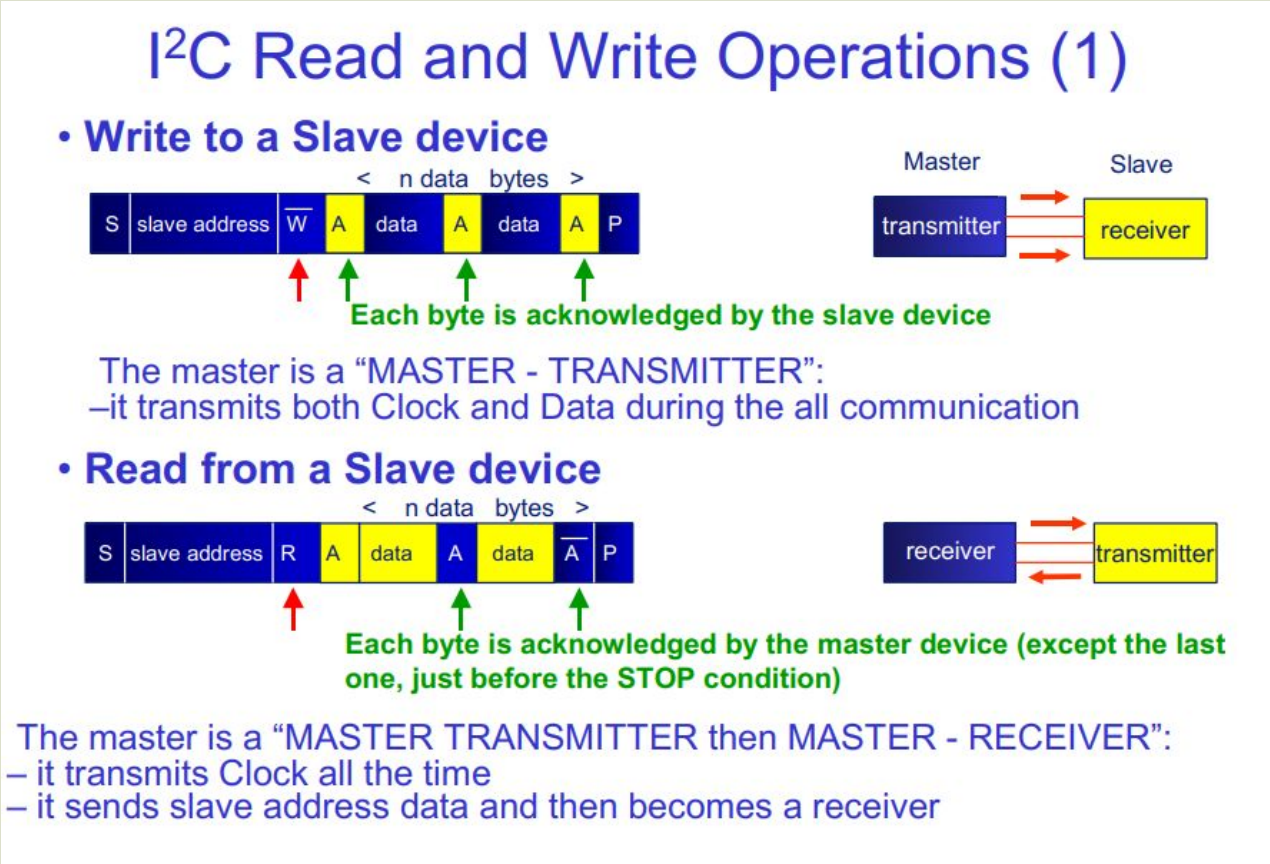
STOP:

SDA переходит с **низкого** на **высокий** уровень в то время, когда на **SCL** поддерживается **высокий** уровень



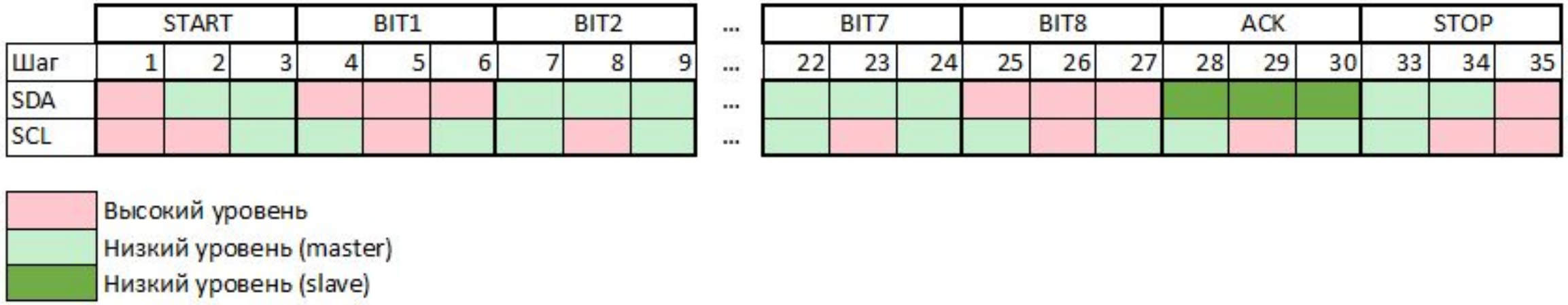
I2C

- Данные передаются блоками по 8 бит, 9-й бит подтверждает прием данных
- Первый блок представляет собой 7-битный адрес устройства + 1 бит направления передачи
- Биты передаются путем выставления **высокого** или **низкого** уровня на **SDA** в то время, пока на **SCL** поддерживается **низкий** уровень. Когда на **SCL** появляется **высокий** уровень, принимающее устройство должно считать бит
- Линией **SCL** управляет **master**. Однако **slave** может устанавливать **низкий** уровень на **SCL**, если он не успевает обработать бит



I2C

Пример: Передать байт **10011001** на шину (в реальности передача 1 байта не имеет смысла, будет интерпретировано как чтение 0 байт с устройства с адресом **1001100**)



BIT1 – старший бит

BIT8 – младший бит

I2C

Пример: Передать **10011111 00100001** устройству с адресом **0101011**)

START 010101100 **10011111**0 **00100001**0 **STOP**

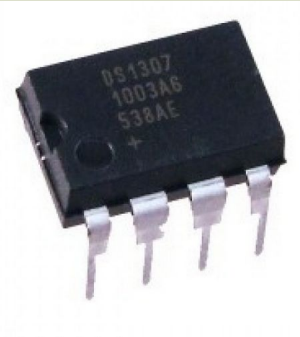
Пример: Получить 2 байта от устройства с адресом **0101011**)

START 010101110 **10011111**0 **00100001**1 **STOP**

Взаимодействие с устройством (I2C)

Устройство может выглядеть как набор пронумерованных ячеек памяти (регистров)

Пример: **DS1307** – часы реального времени. Имеют фиксированный адрес **1101000** на шине I2C



ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00h	CH	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12	10 Hour	10 Hour	Hours				Hours	1–12 +AM/PM 00–23
		24	PM/ AM							
03h	0	0	0	0	0	DAY			Day	01–07
04h	0	0	10 Date		Date				Date	01–31
05h	0	0	0	10 Month	Month				Month	01–12
06h	10 Year				Year				Year	00–99
07h	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08h–3Fh									RAM 56 x 8	00h–FFh

Взаимодействие с DS1307 (I2C)

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00h	CH	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12	10 Hour	10 Hour	Hours				Hours	1–12 +AM/PM 00–23
		24	PM/ AM							
03h	0	0	0	0	0	DAY			Day	01–07
04h	0	0	10 Date		Date				Date	01–31
05h	0	0	0	10 Month	Month				Month	01–12
06h	10 Year				Year				Year	00–99
07h	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08h–3Fh									RAM 56 x 8	00h–FFh

Устройство помнит текущую позицию, откуда считываются или записываются данные. При очередной операции чтения этот адрес сдвигается на соответствующее количество байт.

При записи: первый байт – адрес, куда записывать

Записать, что сейчас 15 часов:

START 110100000 000000100 000101010 STOP

Взаимодействие с DS1307 (I2C)

ADDRESS	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0	FUNCTION	RANGE
00h	CH	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12	10 Hour	10 Hour	Hours				Hours	1–12 +AM/PM 00–23
		24	PM/ AM							
03h	0	0	0	0	0	DAY			Day	01–07
04h	0	0	10 Date		Date				Date	01–31
05h	0	0	0	10 Month	Month				Month	01–12
06h	10 Year				Year				Year	00–99
07h	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08h–3Fh									RAM 56 x 8	00h–FFh

Устройство помнит текущую позицию, откуда считываются или записываются данные. При очередной операции чтения этот адрес сдвигается на соответствующее количество байт.

При записи: первый байт – адрес, куда записывать

Узнать, какой сейчас год:

START 110100000 000001100 STOP

START 110100010 001101111 STOP

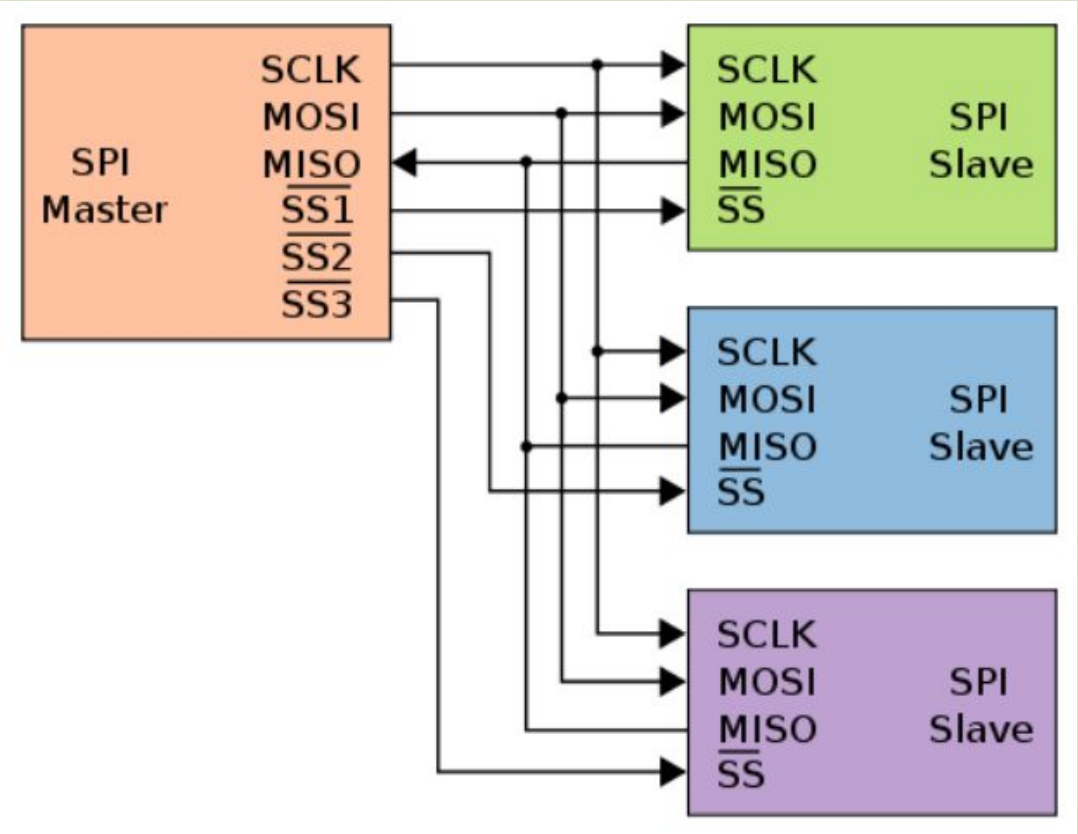
Сейчас 2037 год

SPI

- **SPI** (*Serial Peripheral Interface* — последовательный синхронный стандарт передачи данных в режиме полного дуплекса

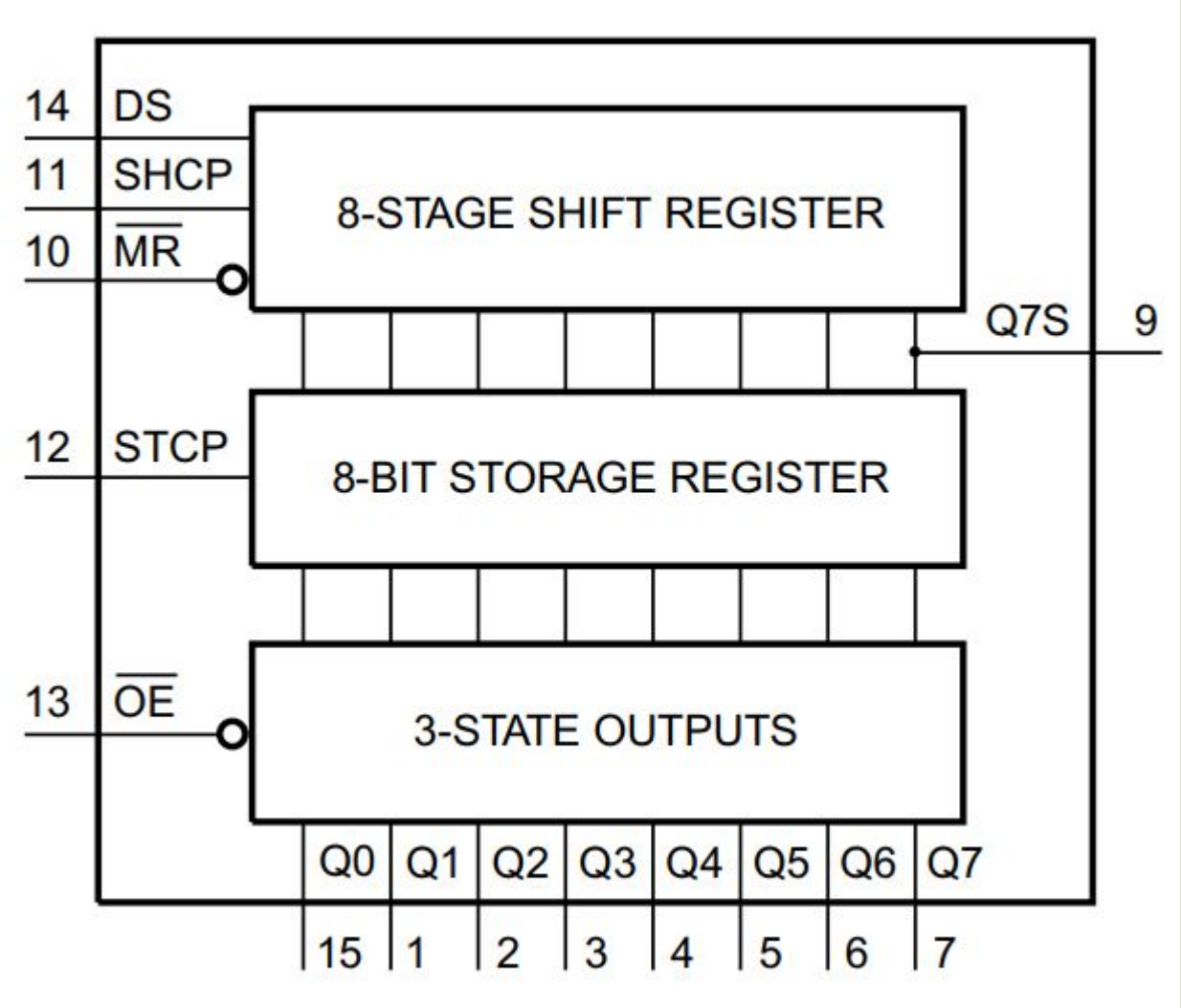
В SPI используются четыре цифровых сигнала:

- **MOSI**(*Master Out Slave In*) — служит для передачи данных от ведущего устройства ведомому.
- **MISO**(*Master In Slave Out*) — служит для передачи данных от ведомого устройства ведущему.
- **SCLK** — последовательный тактовый сигнал. Служит для передачи тактового сигнала для ведомых устройств.
- **CS** или **SS** — выбор микросхемы, выбор ведомого (*Chip Select, Slave Select*).



- **Не нужно передавать адрес**
- **Нет подтверждений**

74HC595



8-bit serial-in, serial or parallel-out shift register with output latches; 3-state

DS – вход данных

SHCP – вход синхронизации для регистра сдвига

STCP - вход синхронизации для регистра хранения

MR – сброс

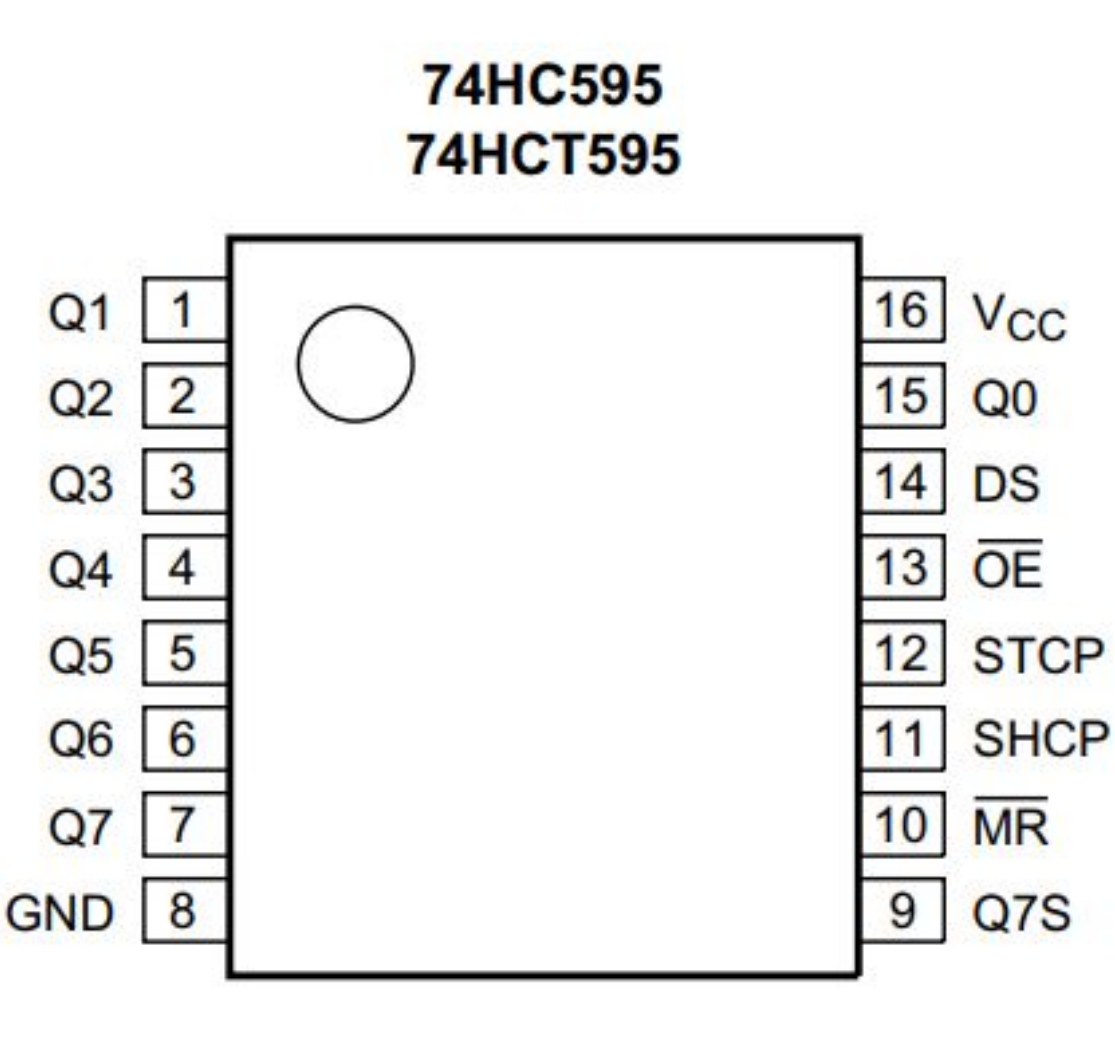
OE – ВКЛ/ВЫКЛ выходы

Q7S – выход данных

74HC595

Control				Input	Output		Function
SHCP	STCP	$\overline{OE}$	$\overline{MR}$	DS	Q7S	Qn	
X	X	L	L	X	L	NC	a LOW-level on $\overline{MR}$ only affects the shift registers
X	↑	L	L	X	L	L	empty shift register loaded into storage register
X	X	H	L	X	L	Z	shift register clear; parallel outputs in high-impedance OFF-state
↑	X	L	H	H	Q6S	NC	logic HIGH-level shifted into shift register stage 0. Contents of all shift register stages shifted through, e.g. previous state of stage 6 (internal Q6S) appears on the serial output (Q7S).
X	↑	L	H	X	NC	QnS	contents of shift register stages (internal QnS) are transferred to the storage register and parallel output stages
↑	↑	L	H	X	Q6S	QnS	contents of shift register shifted through; previous contents of the shift register is transferred to the storage register and the parallel output stages

[1] H = HIGH voltage state;
L = LOW voltage state;
↑ = LOW-to-HIGH transition;
X = don't care;
NC = no change;
Z = high-impedance OFF-state.

74HC595

```
void SendByte595(unsigned char b)
{
    int i;
    for(i=0;i<8;i++)
    {
        if(0x80&(b<<i))HC595_DS=1;
        else HC595_DS=0;
        HC595_SH=1;
        HC595_SH=0;
    }
}

void Append595()
{
    HC595_ST=1;
    HC595_ST=0;
}
```