1.5. I/O
Serial Communication

Simplex

Half-Duplex

Duplex
Serial Communication

**Master-Slave**
- Master
- Slave

**Master-Multi-Slave**
- Master
- Slave
- Slave
- Slave

**Multi-(-)Master**
- Multi-Slave
Serial Communication

Synchronous

Master

Slave

Asynchronous

Master

Slave
## Some Bus Types

<table>
<thead>
<tr>
<th></th>
<th>Wires (+Gnd)</th>
<th>Directionality</th>
<th>Synchrony</th>
<th>Distance typ.</th>
<th>Speed typ.</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RS-232</strong></td>
<td>2/4 –7</td>
<td>full duplex</td>
<td>asynchronous +synchronous</td>
<td>10 m</td>
<td>115kbps / 1Mbps</td>
<td>Point-to-Point Interference prone</td>
</tr>
<tr>
<td><strong>RS-485</strong></td>
<td>2/4</td>
<td>half/full duplex</td>
<td>asynchronous</td>
<td>1000 m</td>
<td>Mbps</td>
<td>Differential Signalling</td>
</tr>
<tr>
<td><strong>SPI [aka SSP, Microwire]</strong></td>
<td>4 [-Vcc]</td>
<td>full duplex</td>
<td>synchronous</td>
<td>few cm</td>
<td>10 Mbps</td>
<td>Master-Multi-Slave with Slave select</td>
</tr>
<tr>
<td><strong>I²C [SMBus]</strong></td>
<td>2 [+Vcc]</td>
<td>half duplex</td>
<td>synchronous</td>
<td>few m</td>
<td>100kbps-3Mbps</td>
<td>Addressed Multi-Master</td>
</tr>
<tr>
<td><strong>1-Wire</strong></td>
<td>1</td>
<td>half duplex</td>
<td>time-slot based, synchronous</td>
<td>tens of m</td>
<td>15kbps / 125kbps</td>
<td>Master-Multi-Slave Parasitic power</td>
</tr>
<tr>
<td><strong>USB 2.0</strong></td>
<td>2 [+ Vcc]</td>
<td>half-duplex</td>
<td>asynchronous</td>
<td>few m</td>
<td>12Mbits / 480 MBits</td>
<td>isochronous/ bulk/ interrupt transfers</td>
</tr>
<tr>
<td><strong>USB 3.0</strong></td>
<td>2+4 [+DGnd + Vcc]</td>
<td>full-duplex</td>
<td>asynchronous</td>
<td>few m</td>
<td>5/10/20 GBits (USB 3.0/3.1/3.2)</td>
<td>Differential signalling</td>
</tr>
</tbody>
</table>
SPI

SCLK: Serial bit-rate Clock
MOSI: Master data Output, Slave data Input
MISO: Master data Input, Slave data Output
SS: Slave Select
SPI

- Four wire serial bus invented / named by Motorola
- Serial connection between two or more devices (microprocessors, D/A converters)
- Configurations
  - 1 Master, 1 Slave (single slave mode)
  - 1 Master, N Slaves (multiple slave mode)
- Synchronous bidirectional data transfer
- Data transfer initiated by Master
- Bandwidth some KBits/s up to several MBits/s
- Simple implementation in software
- Used in a variety of devices, such as memory (flash, EEPROM), LCD displays and in all MMC / SD cards
• Master configures the clock
• Master selects slave (SS), followed by waiting period (if required by slave)
Communication

- Master configures the clock
- Master selects slave (SS), followed by waiting period (if required by slave)
- Clock starts toggling in first active clock cycle
Full duplex data transmission in each cycle
  - Master sends bit over MOSI line, slave reads bit
Full duplex data transmission in each cycle
- Master sends bit over MOSI line, slave reads bit
- Slave sends bit over MISO line, master reads bit
- When no data is to be transmitted any more, master stops toggling the clock
Polarity

Polarity = 0
Polarity = 1
idle=low
idle=high

SCLK

Phase

Phase = 0

Phase = 1

SCLK

first edge

second edge

SPI – Data Transfer

- Master configures the clock
- Master selects slave (SS), followed by waiting period (if required by slave)
- Full duplex data transmission in each cycle
  - Master sends bit over MOSI line, slave reads bit
  - Slave sends bit over MISO line, master reads bit
- Two shift registers, one in slave, one in master for transfer
- When no data is to be transmitted any more, master stops toggling the clock

- No acknowledgement mechanism
- No device interrupts
Programming SPI

1. Bit-Banging
1. Bit-Banging

FOR i := 7 TO 0 BY -1 DO
    IF ODD(ASH(data,-i)) THEN
        Platform.WriteBits(Platform.GPSET0, MOSI);
    ELSE
        Platform.WriteBits(Platform.GPCLR0, MOSI);
    END;
    Kernel.MicroWait(HalfClock);
    Platform.WriteBits(Platform.GPSET0, CLOCK);
    Kernel.MicroWait(HalfClock);
    Platform.WriteBits(Platform.GPCLR0, CLOCK);
END;
Programming SPI

2. Using a Controller

Master

SPI Controller
2. Using a Controller

(* start transition *)
Platform.SetBits(Platform.SPI_CS, {TA});

REPEAT UNTIL TXD IN Platform.ReadBits(Platform.SPI_CS);

Platform.WriteWord(Platform.SPI_FIFO, data);
junk := Platform.ReadWord(Platform.SPI_FIFO);

REPEAT UNTIL DONE IN Platform.ReadBits(Platform.SPI_CS);

(* transfer inactive *)
Platform.ClearBits(Platform.SPI_CS, {TA});
BCM 2835 Registers

**CS -- Control and Status**
- Chip Select
- FIFO Status
- Transfer Progress
- Interrupts
- Polarity & Phase

**FIFO Register**
- Data
  - Write: TX Fifo
  - Read: RX Fifo

**CLK**
- Clock Divider

**Other**
- DMA Control
- Special Mode Control
MAX7219 8-Digit LED Display Driver

Max7219 Specification, p.5
MAX7219 8-Digit LED Display Driver

![MAX7219 8-Digit LED Display Driver Diagram](image)

**Figure 1. Timing Diagram**

**Table 1. Serial-Data Format (16 Bits)**

<table>
<thead>
<tr>
<th>D15</th>
<th>D14</th>
<th>D13</th>
<th>D12</th>
<th>D11</th>
<th>D10</th>
<th>D9</th>
<th>D8</th>
<th>D7</th>
<th>D6</th>
<th>D5</th>
<th>D4</th>
<th>D3</th>
<th>D2</th>
<th>D1</th>
<th>D0</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ADDRESS | MSB | DATA | LSB

Max7219 Specification, p.6
MMC and SD Cards

- Low cost memory system for persistent data on “solid state mass storage“ (for example flash memory cards)
- Separate bus system
  - 1 master, N slaves (cards)
  - typically 1 master for one card
- Serial & synchronous transfer of commands and data
  - Sequential read/ write
  - Block read/ write
MMC System Interaction

Host (BCM 2835)
- CPU Core
- EMMC Controller
- GPIO Pins

Bidirectional Data Channels

SD Card
- SD Card Interface Controller
- Memory
- SD Card Pins

CLK
CMD
DAT

Command and Response Channel
Clock for synchronous transfer

Synchronous transfer

Memory

CPU Core

EMMC Controller

GPIO Pins

Bidirectional Data Channels

SD Card

MMC System Interaction
SD Card

Memory Core Interface

Card Interface Controller

Power On Detection

reset

reset

DAT0 DAT1 CLK CMD DAT2 DAT3 V_{DD}

SD Mode vs SPI Mode
SD Mode vs SPI Mode
Example: Block Read/Write Operation (SPI mode)

- **Read**

- **Write**

SanDisk SD Card Product Manual 2.2
SD Memory Card State Diagram Example
(Card Identification)
RS232

Terminal [DTE]

UART

Data Set [DCE]
(Modem)

UART

TxD

Rx&D

GND

+ 

RTS/RTR

CTS

if Hardware Flow Control
RS232 Signalling

- Start bit: starts the local clock
- 8 data bits: (+parity, if applicable)
- 1-2 stop bit(s)

Sampling in the middle of bit intervals
UART

Universal Asynchronous Receiver/ Transmitter

- Serial transmission of individual bits in byte packets (lowest significant bit first)
- Configurable
  - Number of data bits per byte: 5, 6, 7, 8
  - Parity: odd, even, none
  - Number of stop bits: 1, 1.5, 2
  - Transfer rate in bps (bits per second): 75, 110, 300, ..., 115200

PROCEDURE UARTHandler( uart: Uart );
VAR pending: SET;
BEGIN
    pending := 
      Platform.ReadBits(Platform.UART_MIS);
    IF (Platform.RXMIS IN pending) OR (Platform.RTIM IN pending) THEN
        EmptyFIFO( uart );
    END;
    IF Platform.TXMIS IN pending THEN
        FillFIFO( uart );
    END;
    Kernel.EnableIRQ( 
      Platform.UartInstallIrq , TRUE );
END UARTHandler;