HY428 Embedded Systems
1) LEGO NXT Ports
2) I^2C

Lego NXT

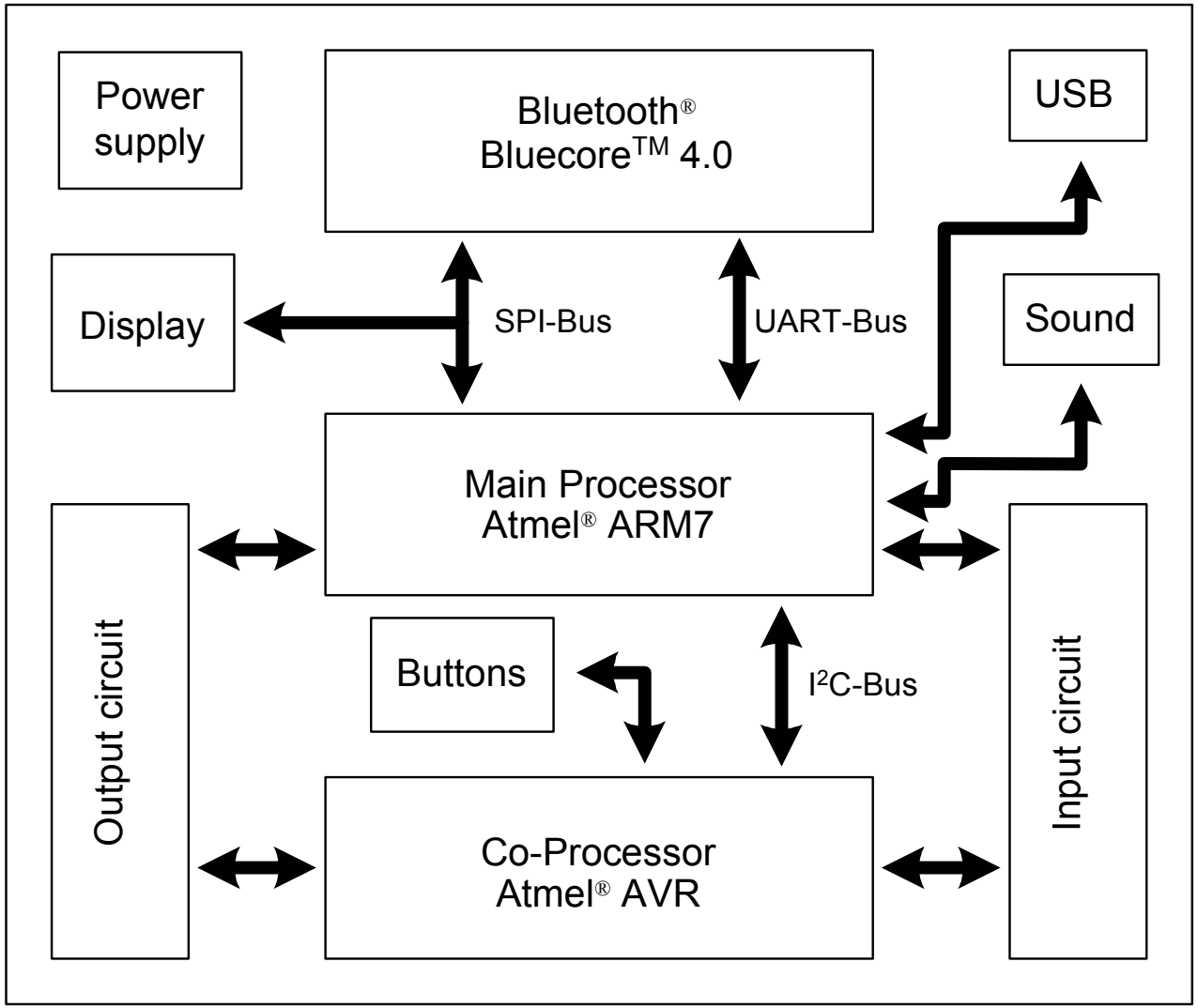
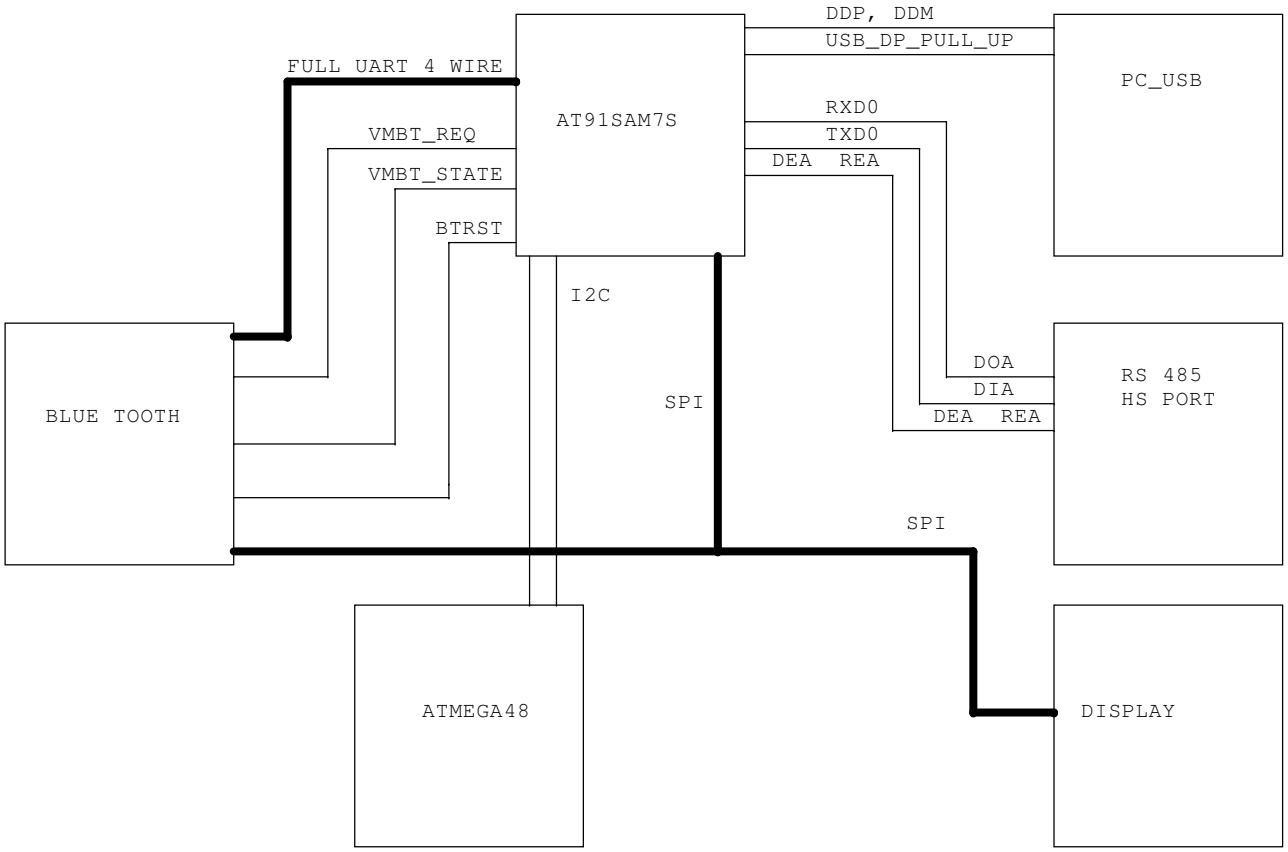


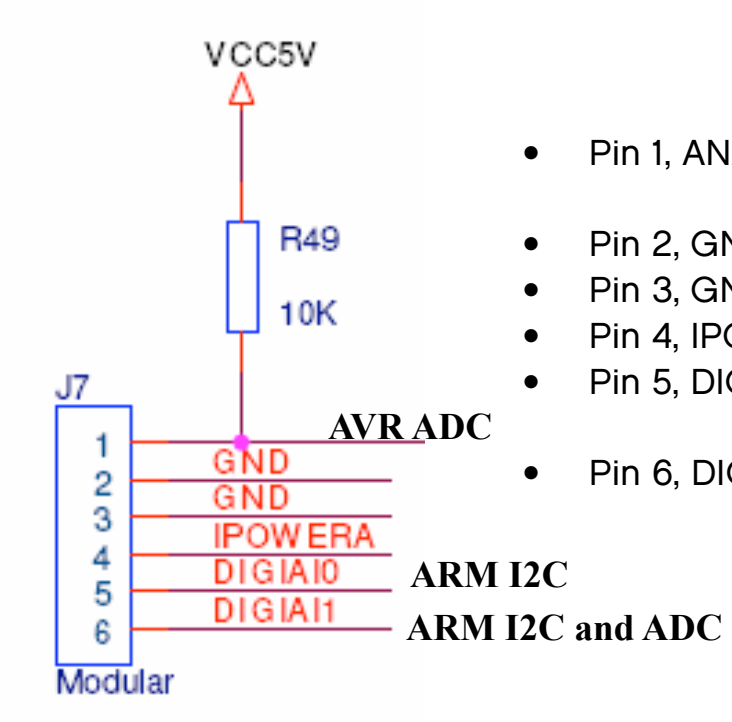
Figure 1: Hardware block diagram of the NXT brick

Lego NXT

COMMUNICATION BLOCK

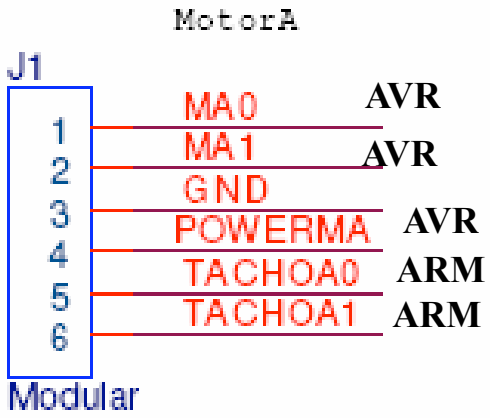


Lego NXT Input Ports (1,2,3,4)



- Pin 1, ANA — Analog input and possible current output signal
- Pin 2, GND — Ground signal
- Pin 3, GND — Ground signal
- Pin 4, IPOWERA — 4.3 Volt output supply
- Pin 5, DIGIAI0 — Digital I/O pin connected to the ARM7 processor
- Pin 6, DIGIAI1 — Digital I/O pin connected to the ARM7 processor

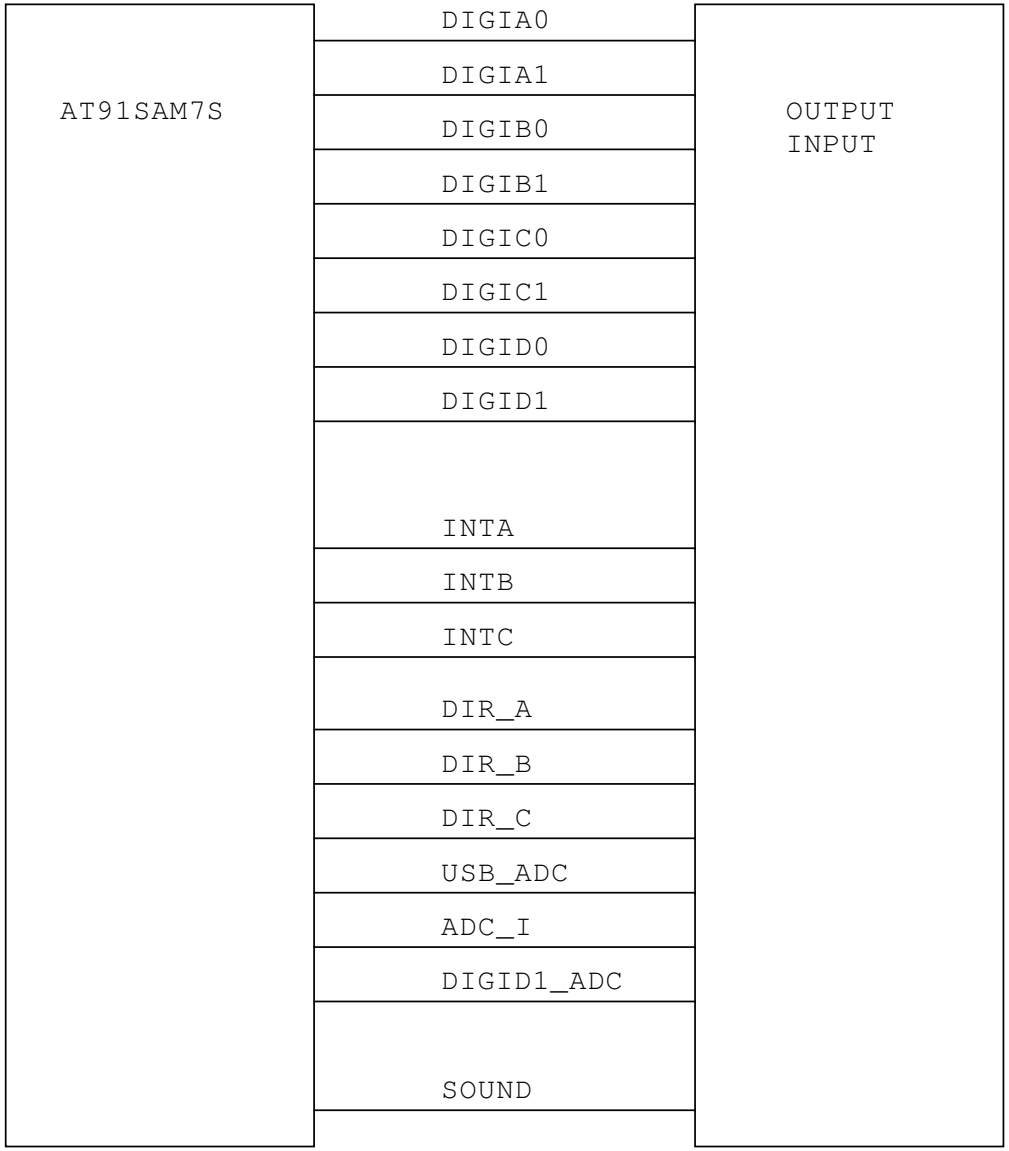
Lego NXT Output Ports (A,B,C)



Pin 1, MA0	PWM output signal for the actuators
Pin 2, MA1	PWM output signal for the actuators
Pin 3, GND	Ground signal related to the output supply
Pin 4, POWERMA	4.3 Volt output supply
Pin 5, TACHOA0	Input value that includes Schmitt trigger functionality
Pin 6, TACHOA1	Input value that includes Schmitt trigger functionality

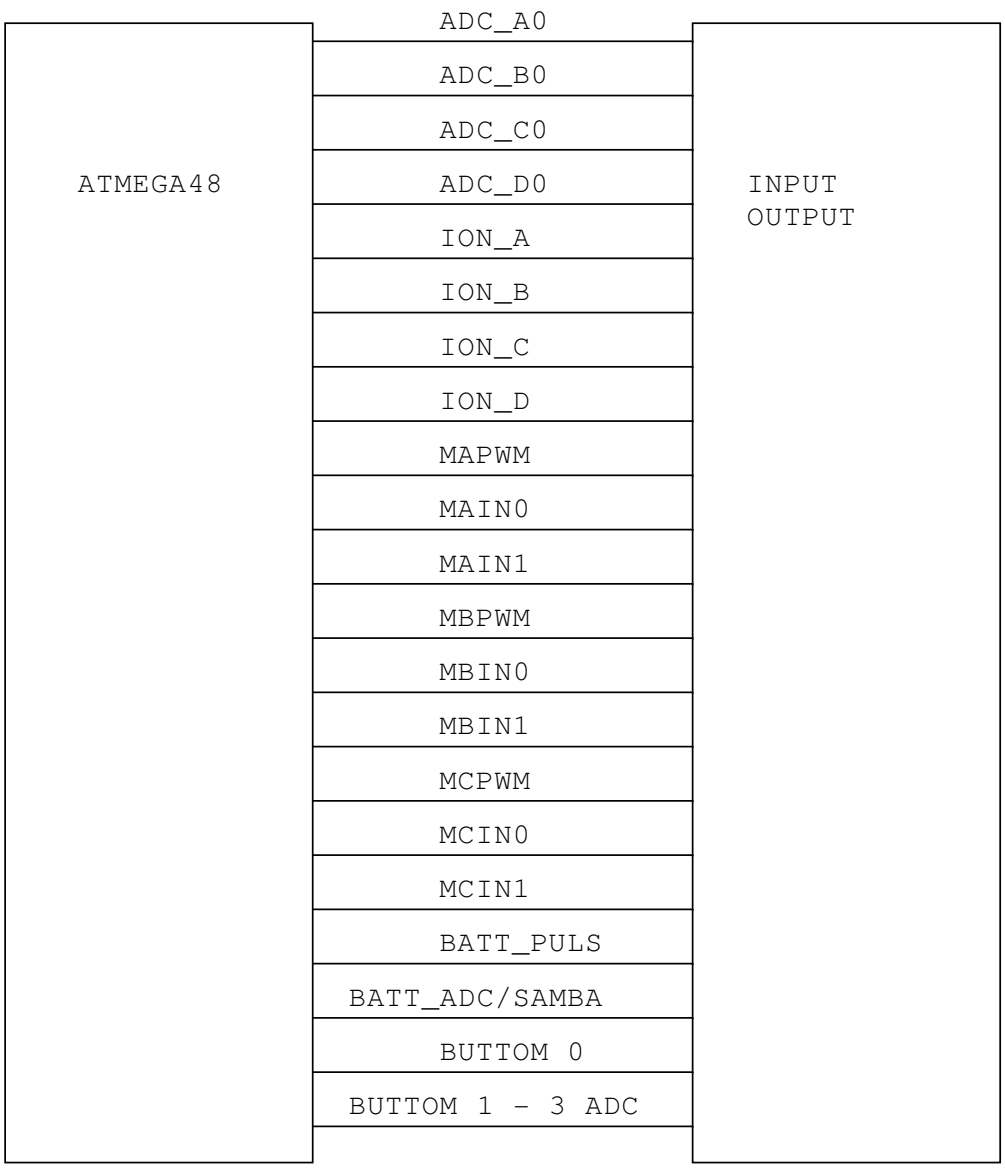
ARM

ARM BLOCK



AVR

AVR BLOCK



HY428 Embedded Systems
1) LEGO NXT Ports
2) I^2C

I2C

- I2C is a bus
- It uses two wires
  - One for clock
  - One for data
- It allows multiple masters/slaves
  - Only one master at a time
  - What if multiple devices try to become master?
- Master says
  - Which slave
  - Read/write
  - Multiple byte transfers
- Slave simply responds to master requests
- Read/write transaction
  - Address and data
  - Address is 7 bits
  - +1 bit for R/W = 8 bits
  - Data: read/write bytes
- How do you indicate traffic from idle?
  - Need start/stop signals
- What if things go wrong?
  - Need ack/nack
- Difference from SPI?
  - Let's come back to this…

Master - slave

- Only master can read/write
  - Others (slaves) respond
- Master always writes/reads 8 bits (multiples)
- A transactions consists of
  - Address (slave to read/write)
  - Data – possibly multiple bytes

Bit Transfers

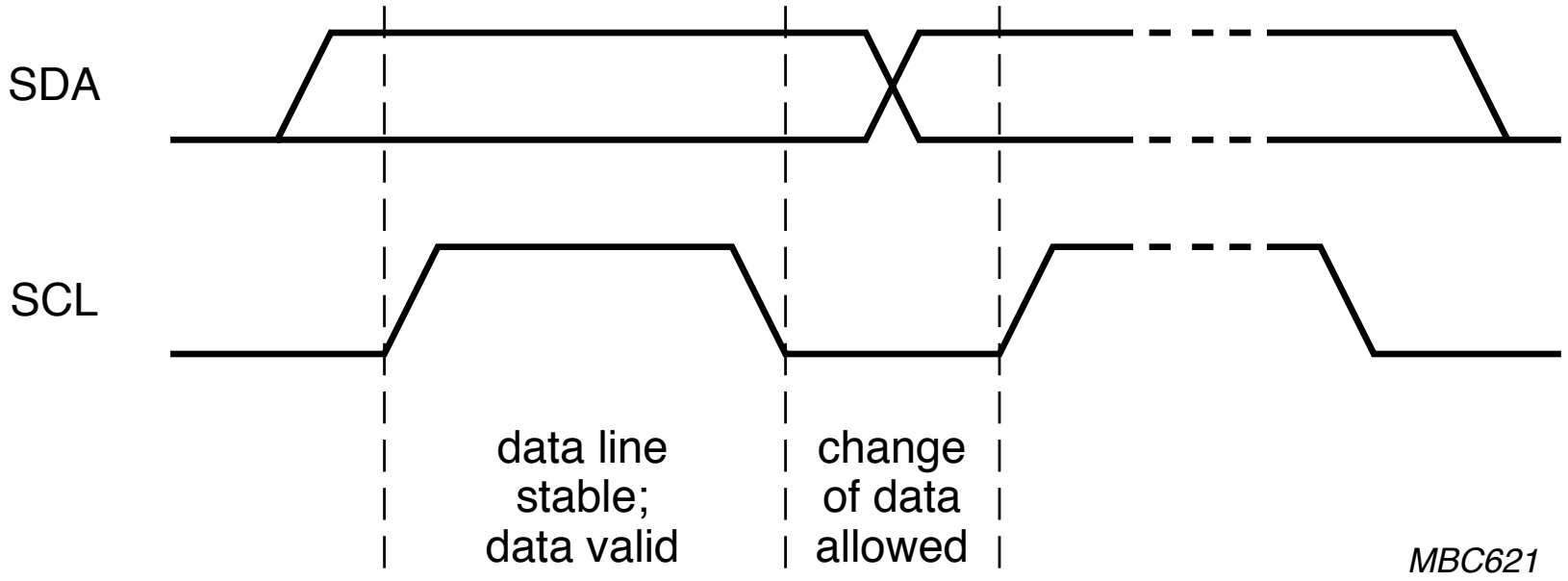


Fig.4 Bit transfer on the I^2C-bus.

START & STOP: special ops

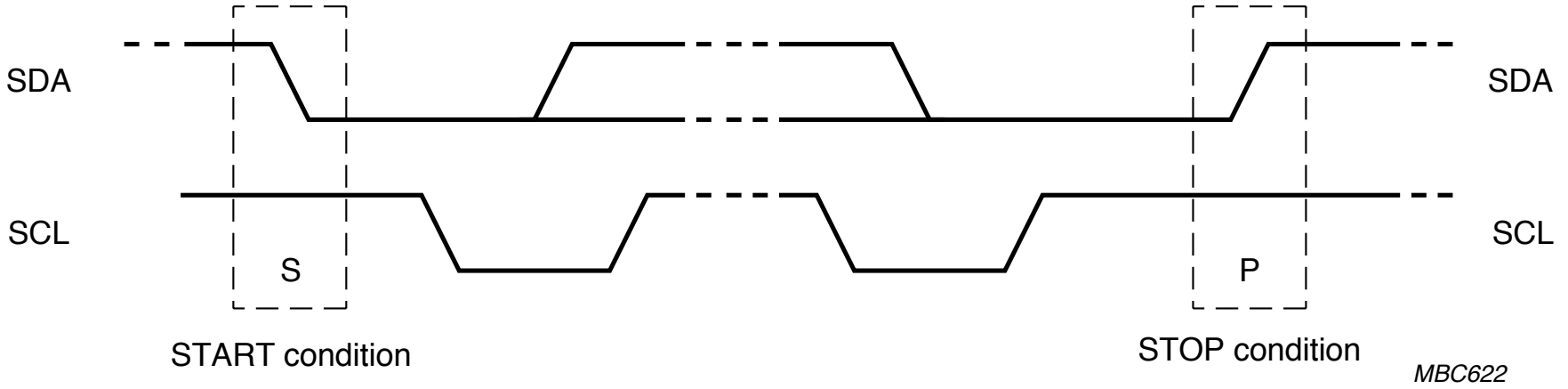


Fig.5 START and STOP conditions.

Acknowledgements

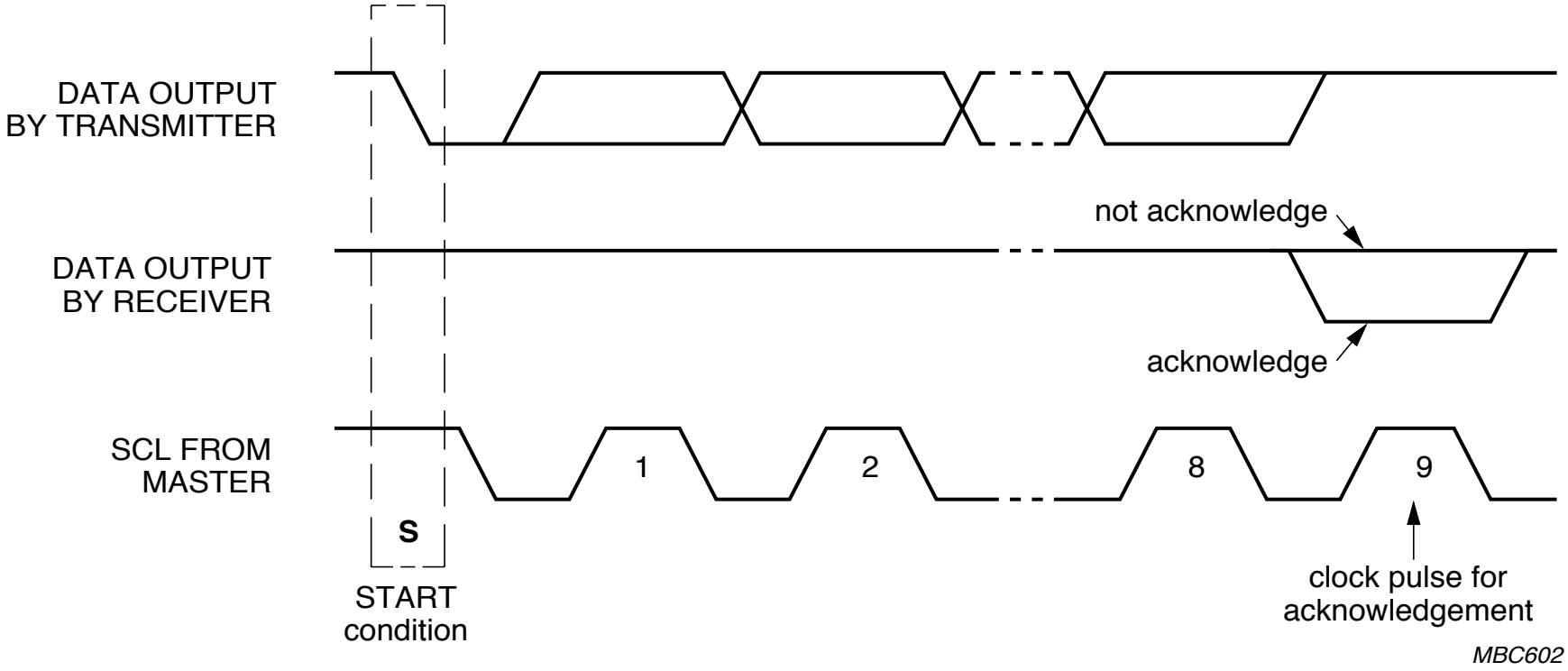


Fig.7 Acknowledge on the I^2C-bus.

- What does an ACK/NACK mean for master/slave, transmitter/receiver?
  - Slave-receiver: ack/nack data and if nack master should reset
  - Master-receiver: ack=send more, nack=stop sending

Data Transfer Example

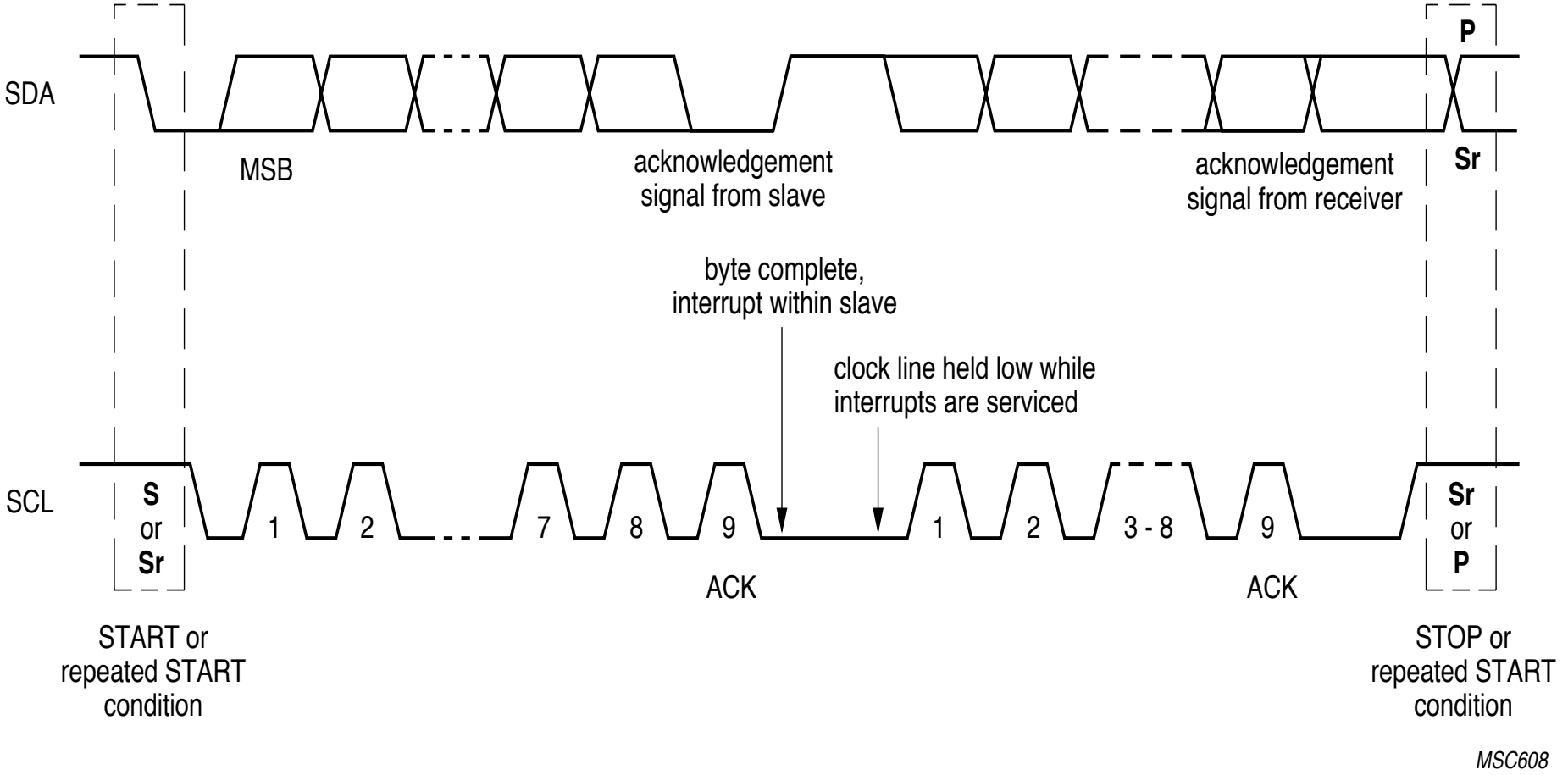


Fig.6 Data transfer on the I^2C-bus.

Multi-master Clock Synchronization

Problem: When multiple master devices on bus, how do they agree on a common clock?

- Slaves receive their clock from master
- SCL is wired AND

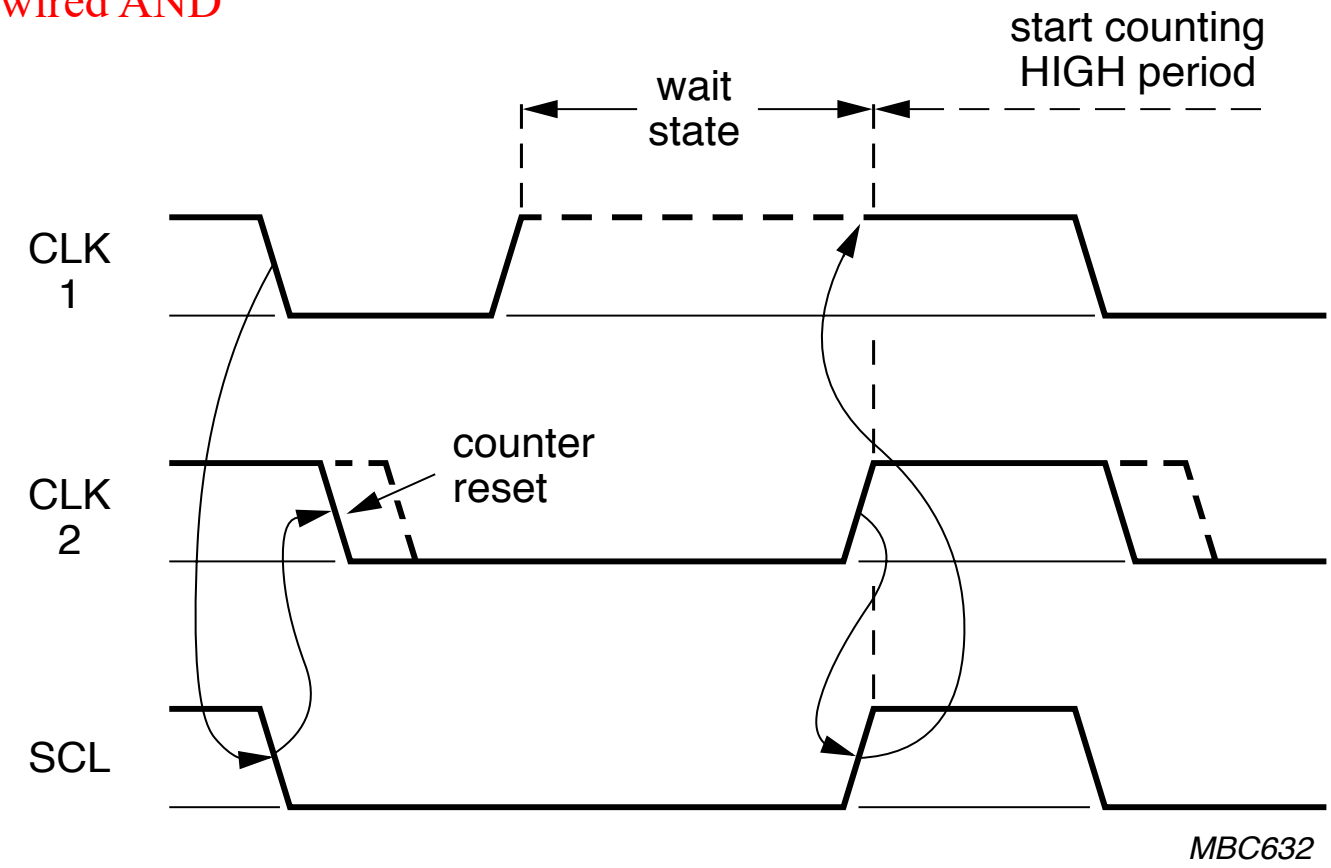
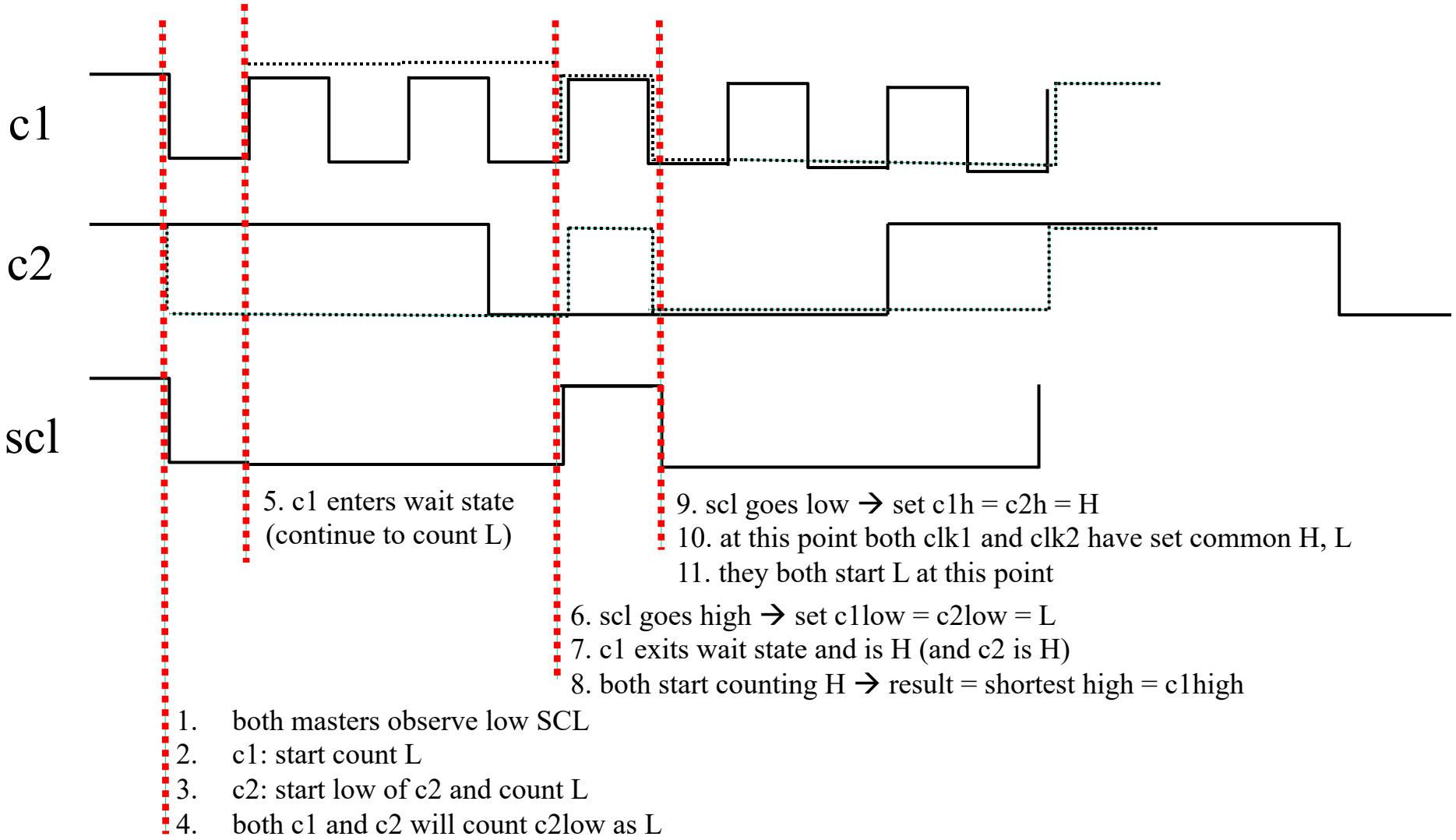


Fig.8 Clock synchronization during the arbitration procedure.

Clock Synchronization (scenario: fast+slow)



Clock Synchronization

- Wired AND:
  - When SCL goes low (even if device is high) device starts counting low
- Wait state:
  - When a low clock drives the SCL line the high devices go in wait state (so high period is “postponed”) and continues to count low
- Wired AND:
  - When the last low device goes high, all others exit the wait state and start counting high
  - When any device goes low, everyone stops counting high
- At this point all have counted low and high and have a common clock
- I2C clock has the longest low period and shortest high period of all devices connected to the I2C
- What if there are two devices with “non-overlapping” high pulses?
  - Low SCL causes all high devices to go in a wait state - the world stops and nothing happens for them. At some point when the low device becomes high they will also be high and everyone will start counting high
  - Invariant: if you go high, you never go low until after everyone else has gone high => This means that everyone will see (and count) first low and then high pulse of common clock

Arbitration

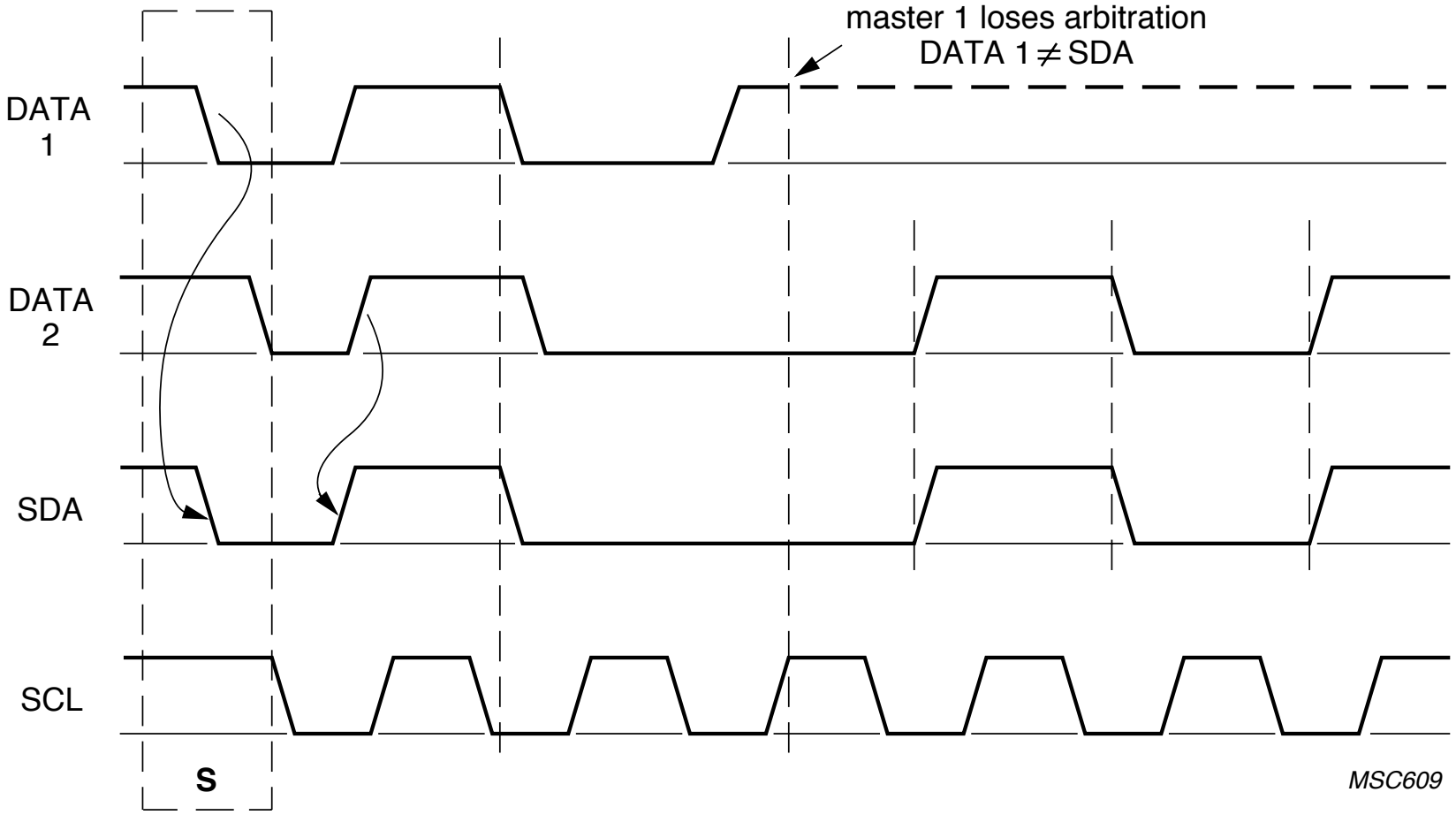


Fig.9 Arbitration procedure of two masters.

A Complete Data Transfer

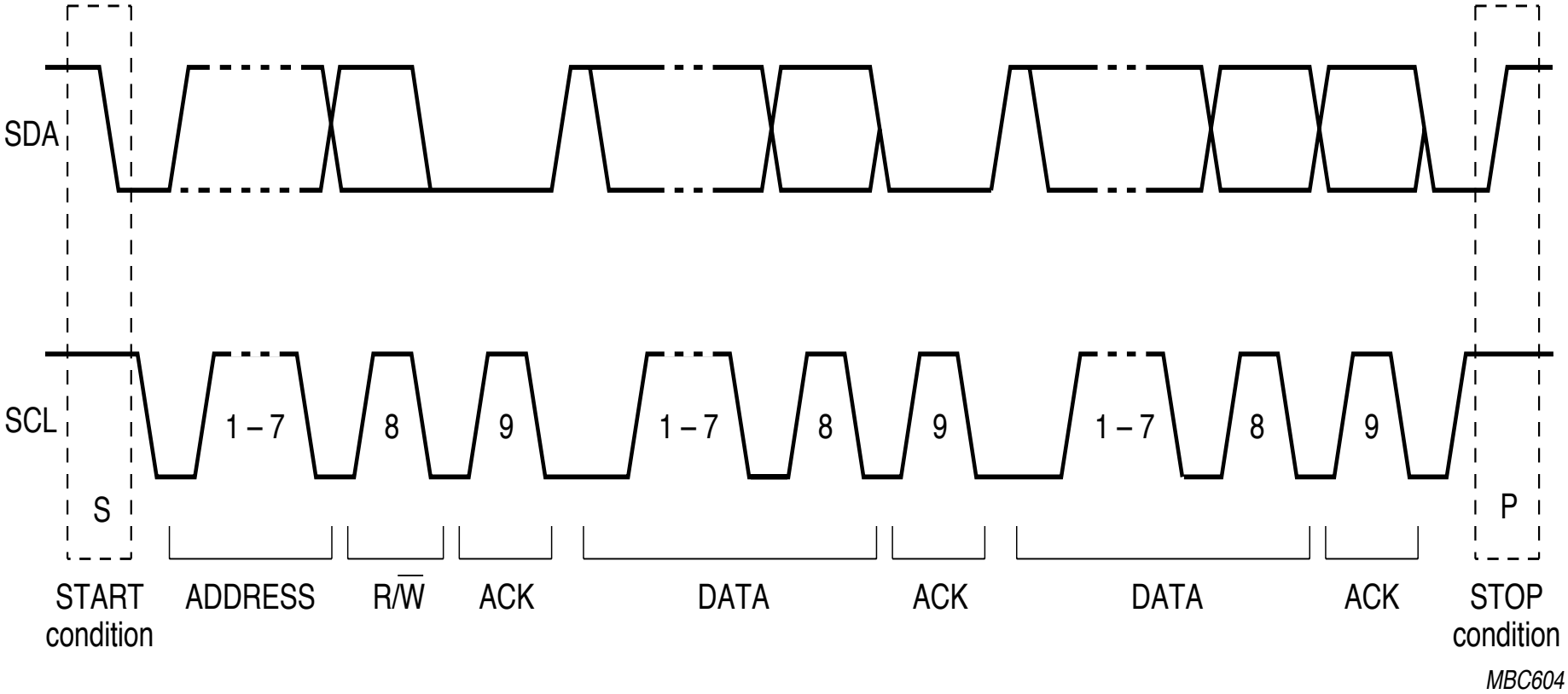


Fig.10 A complete data transfer.

Example 1

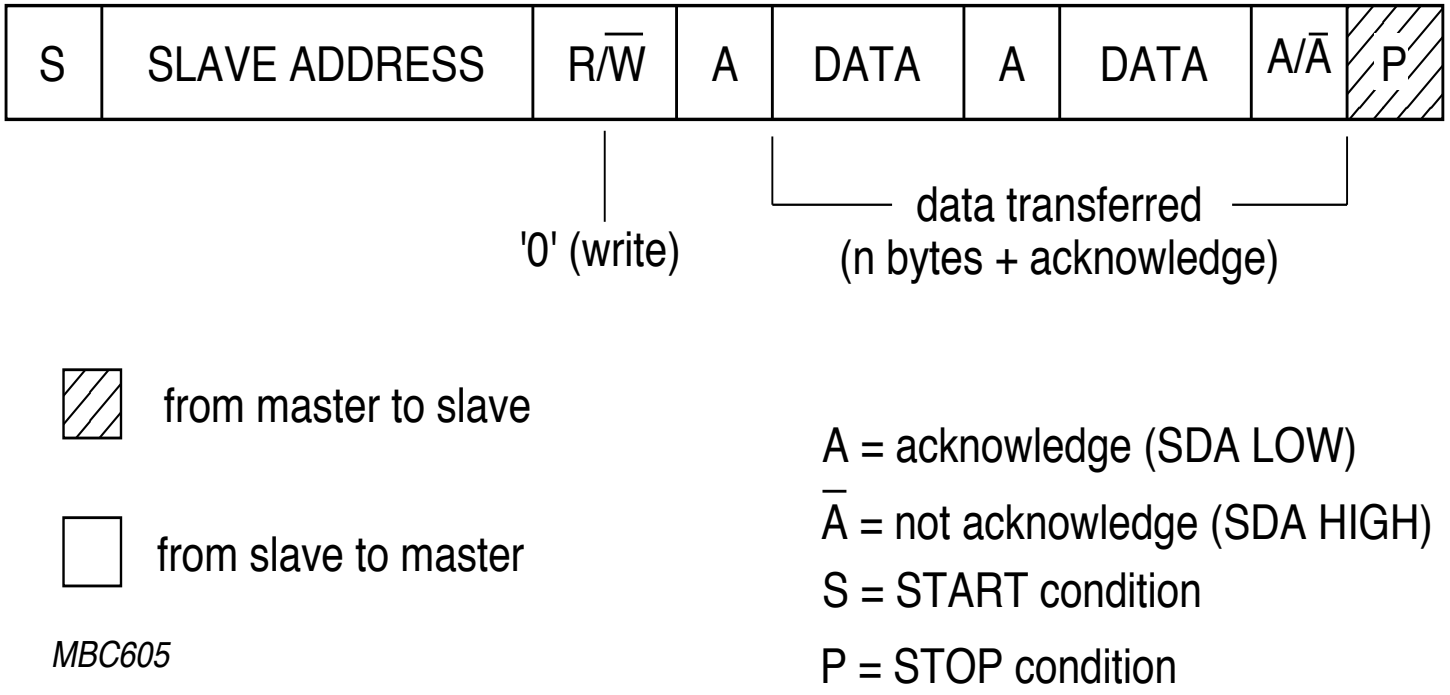


Fig.11 A master-transmitter addressing a slave receiver with a 7-bit address. The transfer direction is not changed.

Example 2

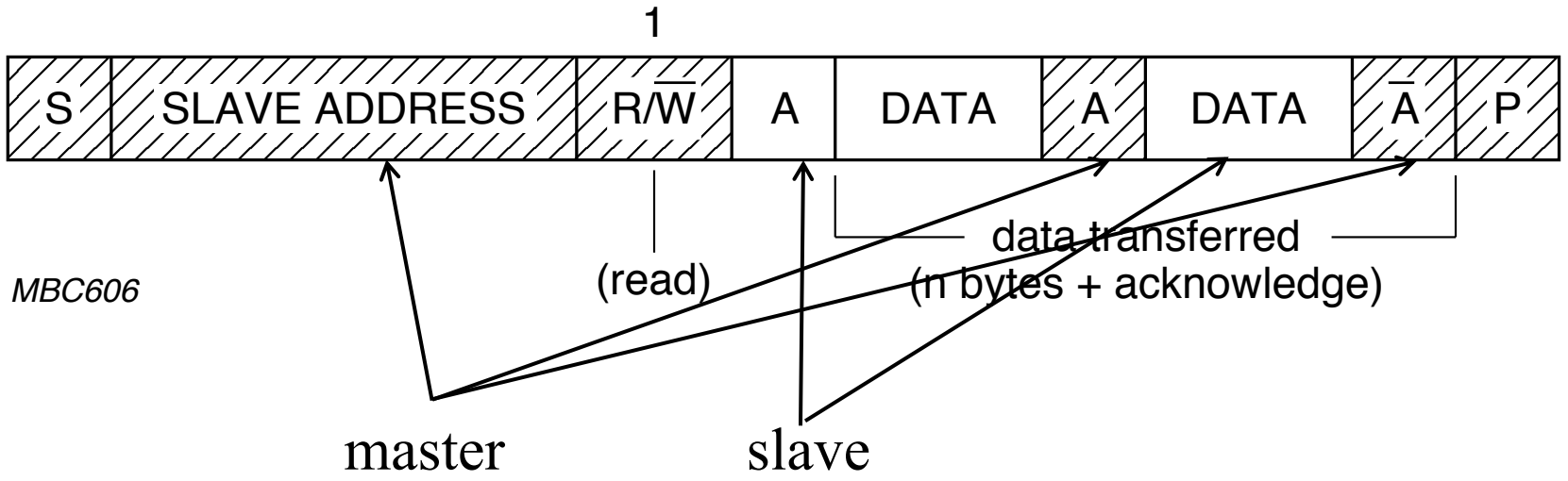


Fig.12 A master reads a slave immediately after the first byte.

Example 3

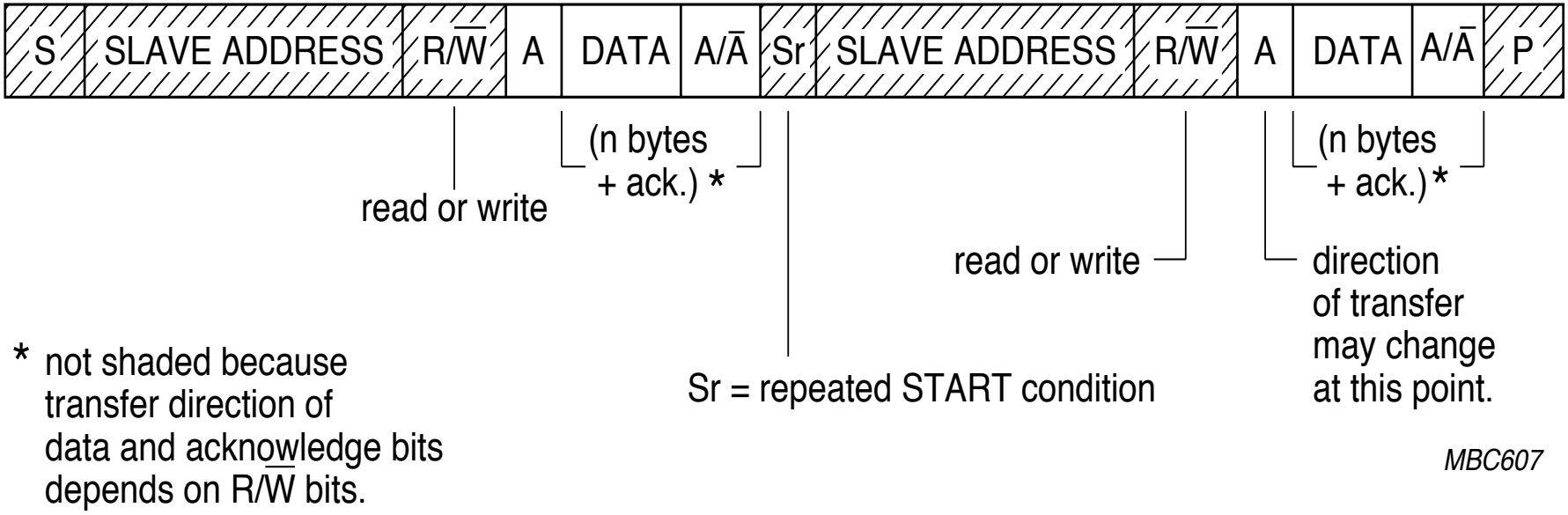


Fig.13 Combined format.

- Direction of transfer can change at any time during a transfer by
- Repeating START+Address
- no stop, no re-arbitration required

Summary: How to perform a transfer?

- Who is the master
- How to agree on clocks
- When a new transaction (read/write) starts
- How to read/write one or multiple bytes
- When something went wrong