COEN-4720 Embedded Systems

Lecture 7
I2C

Cris Ababei
Dept. of Electrical and Computer Engineering
MARQUETTE UNIVERSITY
BE THE DIFFERENCE.

Outline

- I2C
- CAN

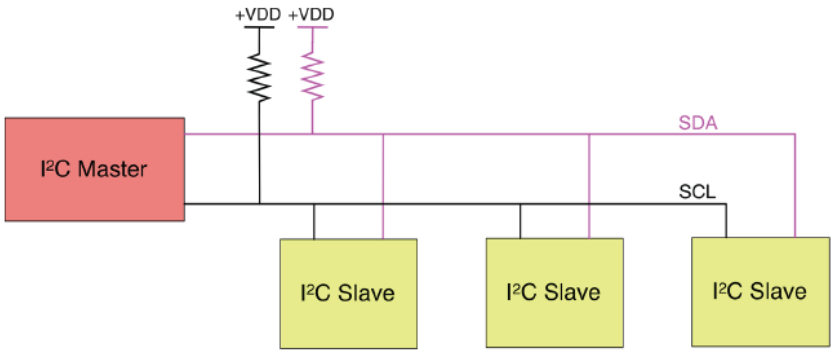
I2C

- I2C was created by Philips Semiconductors stands for Inter-Integrated Circuit (inside TV sets)
- It is a simple communication protocol
- Allows communication of data between I2C devices over two wires.
- It sends information serially using one line for data (SDA) and one for clock (SCL).
- To communicate, a master drives a clock signal on SCL while driving, or allowing a slave to drive SDA
  - Therefore, the bit-rate of a transfer is determined by the master.

3

3

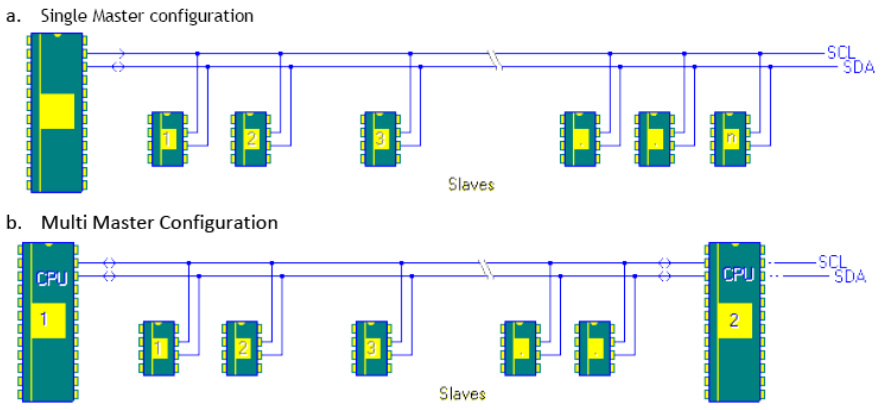
I2C Two Wire BUs



4

4

Single or Multi Master



5

5

Multi Master

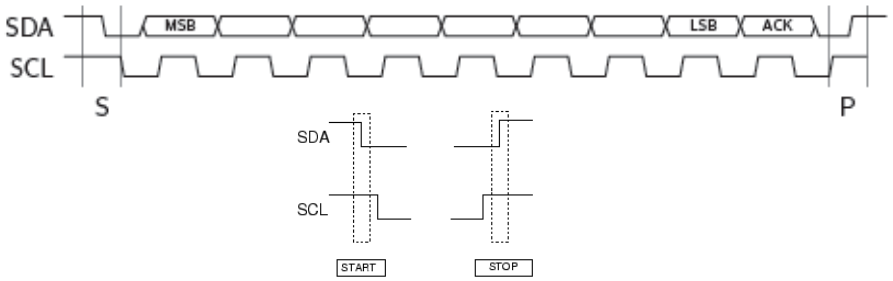
- Devices need to be able to cooperate. This means:
  a) Being able to follow arbitration logic. If two devices start to communicate at the same time the one writing more zeros to the bus (or the slower device) wins the arbitration and the other device immediately discontinues any operation on the bus.
  b) Bus busy detection. Each device must detect an ongoing bus communication and must not interrupt it. This is achieved by *recognizing traffic* and waiting for a stop condition to appear before starting to talk on the bus.
- More info:
  - http://www.i2c-bus.org/MultiMaster/

6

6

I2C Physical Protocol

- Communication between a master and a slave consists of a **sequence of transactions** where the master utilizes the SCL as a clock for serial data driven by the master or a slave on SDA
- When the master wishes to talk to a slave, it begins by issuing a start sequence on the I2C bus. A start sequence is one of two special sequences defined for the I2C bus, the other being the stop sequence. These are also referred to as **Start condition (S)** and **Stop condition (P)**

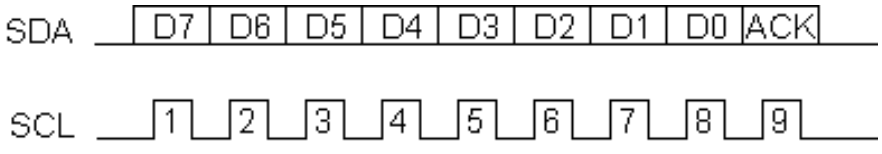


7

7

I2C Physical Protocol

- A transaction consists of a sequence of bytes.
- Each byte is sent as a sequence of 8 bits.
- Bits of each byte of data are placed on the SDA line starting with the MSB. The SCL line is then pulsed high, then low.
- For every 8 bits transferred, the device receiving the data sends back an acknowledge bit, so there are actually 9 SCL clock pulses to transfer each 8 bit byte of data.

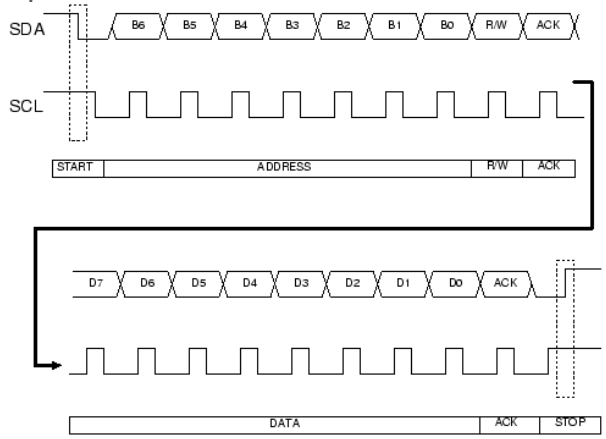


8

8

I2C Physical Protocol

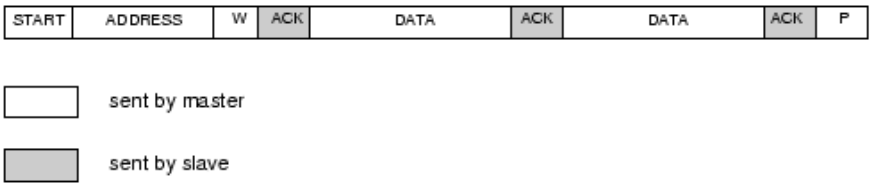
- If the receiving device sends back a low ACK bit, then it has received the data and is ready to accept another byte.
- If it sends back a high (Not Acknowledge, NACK) then it is indicating it cannot accept any further data and the master should terminate the transfer by sending a stop sequence.



9

9

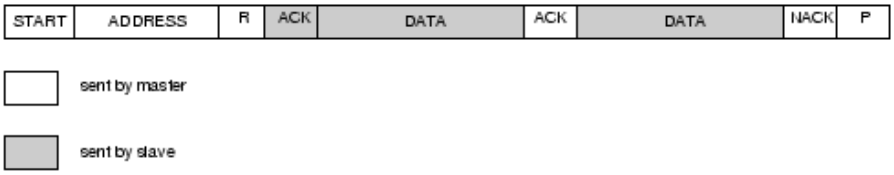
Data transfer from master to slave



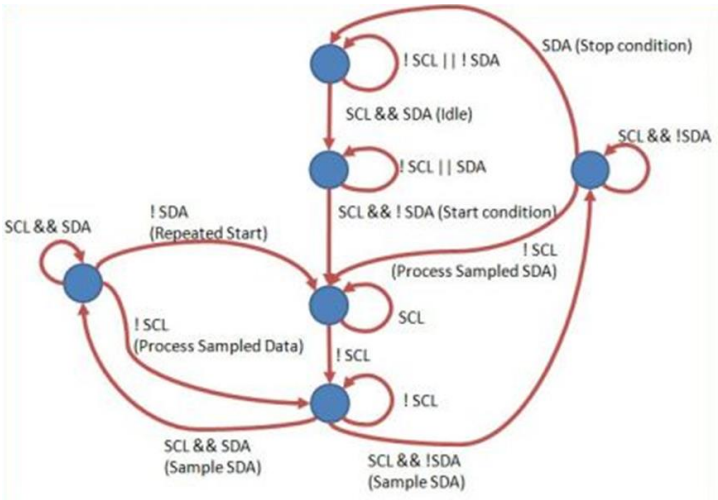
10

10

Data transfer from slave to master



Operation FSM



Operation FSM

- Six fundamental operations:
  1) Idle,
  2) Start condition,
  3) Sample SDA,
  4) Process Sampled SDA Bit,
  5) Stop condition, and
  6) Repeated Start condition

Speed

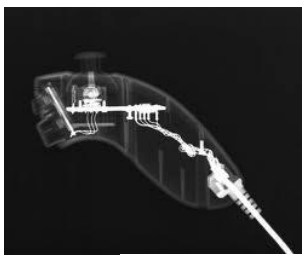
- Standard clock speeds:
  - 100kHz
  - 10kHz
- However, the standard lets us use clock speeds from zero to 100kHz
- A fast mode is also available (400kHz – Fast mode)

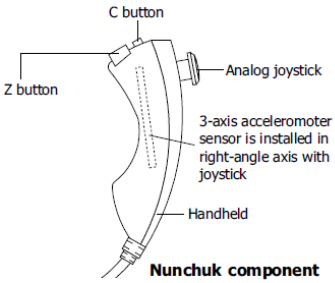
Examples of I2C devices

- Sensors: barometric, temperature, acceleration, compass
- Real-time clocks, DACs, keyboard
- Controllers: Wii NunChuck
- Memories
- Etc.

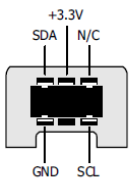
15

Wii NunChuck

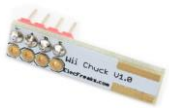




Nunchuk component



Pin assignment of Nunchuk connector. See from front view.



16

Wii NunChuck Internals

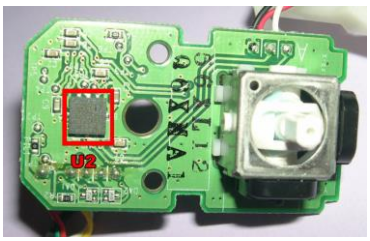
Function	Hardware	Circuit board surface and mounting
C	membrane switch	daughterboard, through-hole
Z	membrane switch	daughterboard, through-hole
Joystick X	axial potentiometer, 30KΩ	through-hole
Joystick Y	axial potentiometer, 30KΩ	through-hole
Accelerometer	ST 8XRJ 3L02AE 820 MLT	surface mount, top
Microcontroller	FNURVL 405 849KM (48-pin QFP)	surface mount, bottom

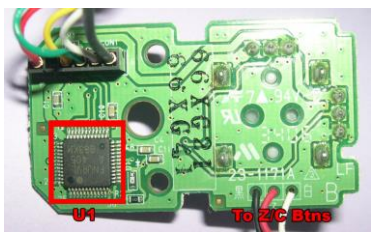
http://wiibrew.org/wiki/Nunchuck#Nunchuk

17

17

Wii NunChuck Internals





- Joystick: axial potentiometer, 30KΩ through-hole
- Accelerometer: ST 8XRJ 3L02AE 820 MLT surface mount, top
- Microcontroller: FNURVL(A)-405 849KM (48-pin QFP) surface mount, bottom

18

18

HAL_I2C Module

```
typedef struct {
  I2C_TypeDef                *Instance;  /* I²C registers base address     */
  I2C_InitTypeDef            Init;       /* I²C communication parameters   */
  uint8_t                    *pBuffPtr;  /* Pointer to I²C transfer buffer */
  uint16_t                   XferSize;   /* I²C transfer size              */
  __IO uint16_t              XferCount;  /* I²C transfer counter           */
  DMA_HandleTypeDef          *hdmatx;    /* I²C Tx DMA handle parameters   */
  DMA_HandleTypeDef          *hdmarx;    /* I²C Rx DMA handle parameters   */
  HAL_LockTypeDef            Lock;       /* I²C locking object             */
  __IO HAL_I2C_StateTypeDef  State;      /* I²C communication state        */
  __IO HAL_I2C_ModeTypeDef   Mode;       /* I²C communication mode         */
  __IO uint32_t              ErrorCode;  /* I²C Error code                 */
} I2C_HandleTypeDef;
```


19

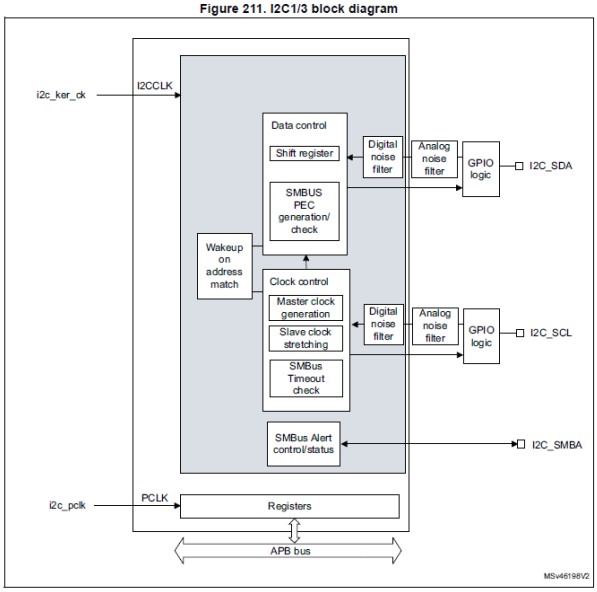
Struct I2C_InitTypeDef

```
typedef struct {
  uint32_t ClockSpeed;      /* Specifies the clock frequency */
  uint32_t DutyCycle;       /* Specifies the I²C fast mode duty cycle. */
  uint32_t OwnAddress1;     /* Specifies the first device own address. */
  uint32_t OwnAddress2;     /* Specifies the second device own address if dual addressing
                               mode is selected */
  uint32_t AddressingMode;  /* Specifies if 7-bit or 10-bit addressing mode is selected. */
  uint32_t DualAddressMode; /* Specifies if dual addressing mode is selected. */
  uint32_t GeneralCallMode; /* Specifies if general call mode is selected. */
  uint32_t NoStretchMode;   /* Specifies if nostretch mode is selected. */
} I2C_InitTypeDef;
```


20

STM32L0x3

Figure 211. I2C1/3 block diagram



Source: Datasheet

Outline

- I2C
- CAN

CAN

- Controller Area Networking (CAN) is a multi-master broadcast serial bus standard for connecting electronic control units (ECUs).
- Each node is able to send and receive messages, but not simultaneously.
  - A message consists primarily of an ID (identifier), which represents the priority of the message, and up to eight data bytes.
  - It is transmitted serially onto the bus.
- The devices that are connected by a CAN network are typically sensors, actuators, and other control devices. These devices are not connected directly to the bus, but through a host processor and a CAN controller.

23

23

CAN communication protocol

- The CAN communication protocol is a carrier-sense, multiple-access protocol with collision detection and arbitration on message priority (CSMA/CD+AMP)
- CSMA means that each node on a bus must wait for a prescribed period of inactivity before attempting to send a message
- CD+AMP means that collisions are resolved through a bit-wise arbitration, based on a preprogrammed priority of each message in the identifier field of a message

24

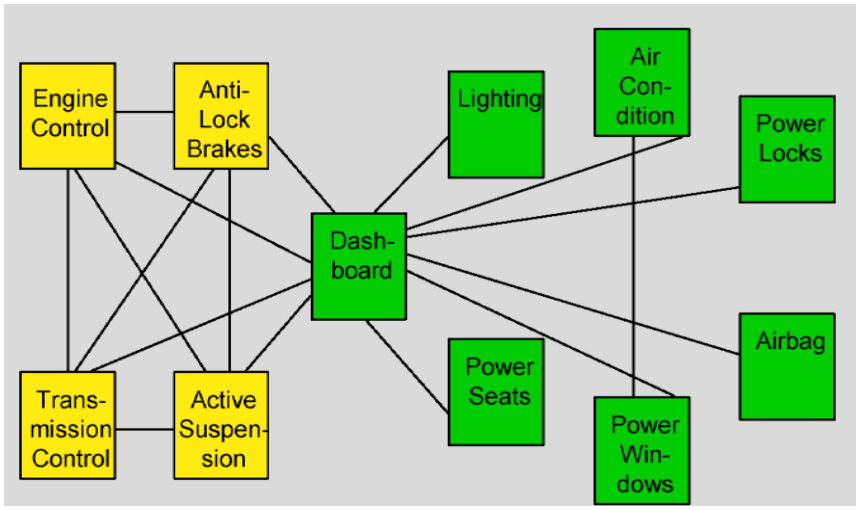
24

CAN

- The CAN bus was defined in the 1980's by Bosch, initially for use in automotive applications. It has been found to be very useful in a wide variety distributed industrial systems
- Characteristics:
  - Uses a single terminated twisted pair cable
  - Is multi master
  - Maximum Signal frequency used is 1 Mbit/sec
  - Length is typically 40m at 1Mbit/sec up to 10km at 5Kbits/sec
  - Has high reliability with extensive error checking
  - Typical maximum data rate achievable is 40KBytes/sec
  - Maximum latency of high priority message <120 µsec at 1Mbit/sec

25

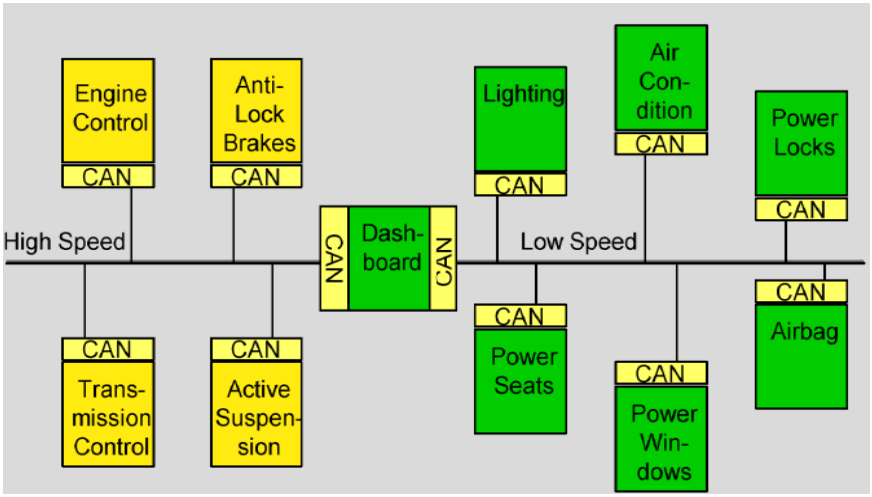
Before CAN



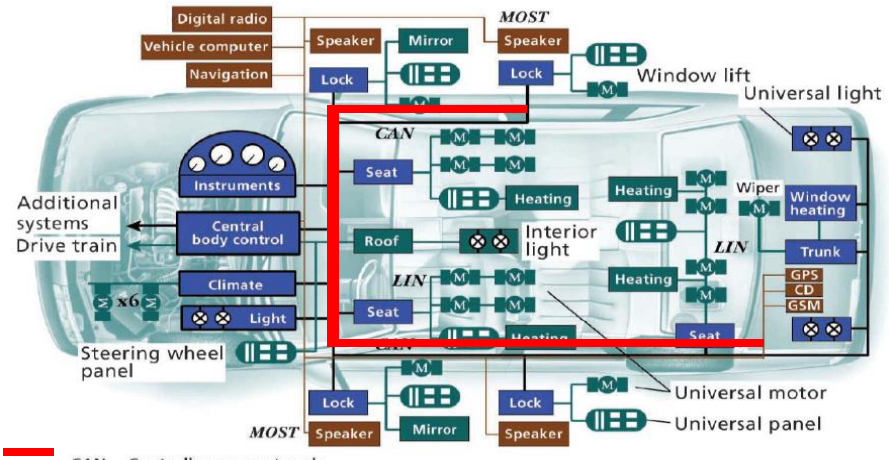
26

26

With CAN



CAN is central to automotive networks



CAN Controller area network
GPS Global Positioning System
GSM Global System for Mobile Communications
LIN Local interconnect network
MOST Media-oriented systems transport

CAN terminology

- Entities on the network are called nodes - are not given specific addresses
- Nodes - depending on their function - transmit specific messages and look for specific message
- Messages themselves have an identifier which also determines the messages' priority

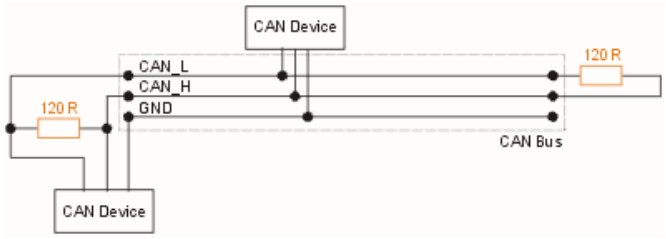
29

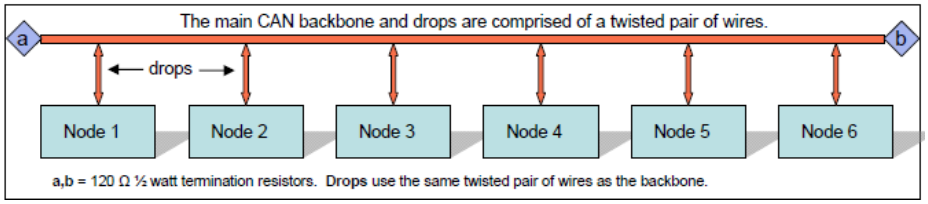
CAN Node Requires:

- Host processor
  - The host processor decides what received messages mean and which messages it wants to transmit itself.
  - Sensors, actuators and control devices can be connected to the host processor.
- CAN controller (hardware with a synchronous clock)
  - Receiving: the CAN controller stores received bits serially from the bus until an entire message is available, which can then be fetched by the host processor (usually after the CAN controller has triggered an interrupt).
  - Sending: the host processor stores its transmit messages to a CAN controller, which transmits the bits serially onto the bus.
- Transceiver
  - Receiving: it adapts signal levels from the bus to levels that the CAN controller expects and has protective circuitry that protects the CAN controller.
  - Transmitting: it converts the transmit-bit signal received from the CAN controller into a signal that is sent onto the bus.

30

CAN devices

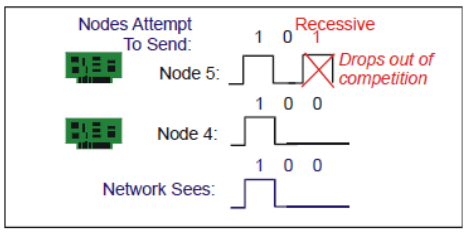




31

31

CAN – Bit dominance



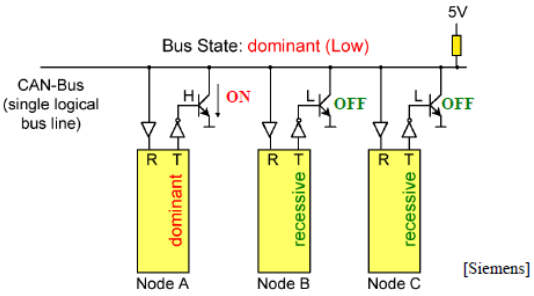
- **Operation**
  - Each node is assigned a unique identification number
  - All nodes wishing to transmit compete for the channel by transmitting a binary signal based on their identification value
  - A node drops out the competition if it detects a dominant state while transmitting a passive state
  - Thus, the node with the lowest identification value wins

32

32

CAN – Bit dominance

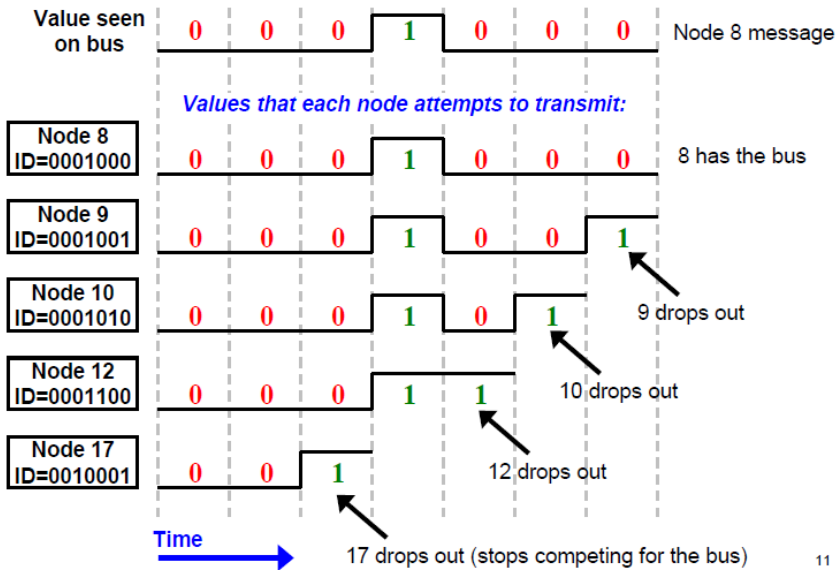
- **CAN uses the idea of recessive and dominant bits**
  - Wired "OR" design
  - Bus floats high unless a transmitter pulls it down (dominant)
  - (Other bus wire in differential transmission floats low and transmitter pulls up)
- **High is "recessive" value**
  - Sending a "1" can't override the value seen on the bus
- **Low is "dominant" value**
  - Sending a "0" forces the bus low no matter what another node is sending



[Siemens]

33

CAN – Bit dominance



34

Signal characteristics

- CAN may be implemented over a number of physical media (most common is a twisted pair of wires) so long as the drivers are open-collector and each node can hear itself and others while transmitting (this is necessary for its message priority and error handling mechanisms)
- The most popular transceiver chips:
  - Philips 82C251
  - TJA1040 (on MCB1700 evaluation board)
- It is necessary to terminate the bus at both ends with 120 Ohms
  - prevent reflections
  - unload the open collector transceiver drivers

35

35

CAN message format

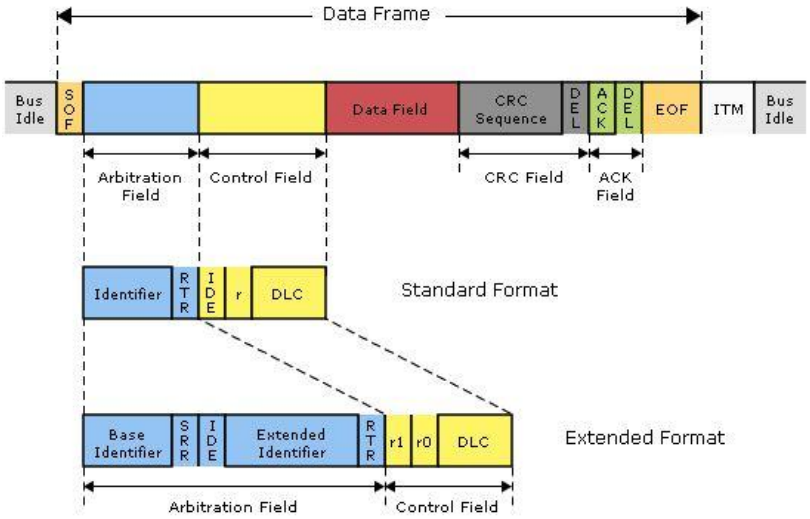
SOF	MESSAGE ID	RTR	CONTROL	DATA	CRC	ACK	EOF

- Start of frame (SOF)
- Message Identifier (MID)
  - the Lower the value the Higher the priority of the message
  - its length is either 11 or 29 bits long depending on the standard being used
- Remote Transmission Request (RTR)=0
- Control field (CONTROL)
  - specifies the number of bytes of data to follow (0-8)
- Data Field (DATA) length 0 to 8 bytes
- CRC field containing a fifteen bit cyclic redundancy check code
- Acknowledge field (ACK)
  - an empty slot which will be filled by any and every node that receives the frame
  - it does NOT say that the node you intended the data for got it, just that at least one node on the whole network got it.
- End of Frame (EOF)

36

36

CAN message format



See nice presentations at:

- *http://marco.guardigli.it/2010/10/hacking-your-car.html*
- *https://www.linkedin.com/pulse/automotive-can-bus-system-explained-kiril-mucevski*

37

37

Remote frames

- Frames that are used to request that a particular message be put on the network - a node somewhere on the network has to be set up to recognize the request, get the data and put out a Message frame.
- This mechanism is used in polled networks.
- The fields are:
  - Start of frame (SOF)
  - Message Identifier (MID) either 11 or 29 bits long depending on the chosen mode.
  - Remote Transmission Request (RTR)=1
  - Control field (CTRL) this specifies the number of bytes of data expected to be returned (0-8).
  - CRC field containing a fifteen bit cyclic redundancy check code.
  - Acknowledge field (ACK) an empty slot which will be filled by any and every node that receives the frame it does NOT say that the node you intended the data for got it, just that at least one node on the whole network got it.
  - End of Frame (EOF)

38

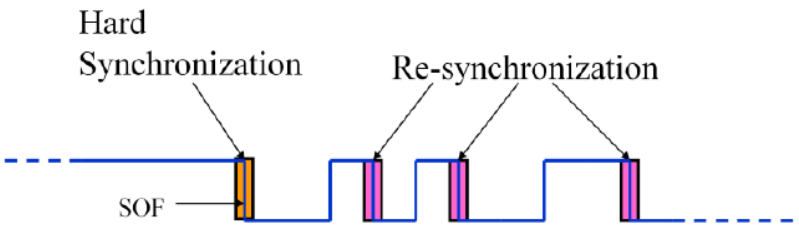
38

Error checking

- CAN is a very reliable system with multiple error checks
- Stuffing error - a transmitting node inserts a high after five consecutive low bits (and a low after five consecutive high). A receiving node that detects violation will flag a bit stuffing error.
- Bit error - A transmitting node always reads back the message as it is sending. If it detects a different bit value on the bus than the one it sent, and the bit is not part of the arbitration field or in the acknowledgement field, an error is detected.
- Checksum error - each receiving node checks CAN messages for checksum errors.
- Frame error - There are certain predefined bit values that must be transmitted at certain points within any CAN Message Frame. If a receiver detects an invalid bit in one of these positions a Form Error (sometimes also known as a Format Error) will be flagged.
- Acknowledgement Error - If a transmitter determines that a message has not been ACKnowledged then an ACK Error is flagged.

Synchronization

- No clock sent on bus (each node has it own oscillator and internal CAN clock)
- Receivers synchronize on recessive-to-dominant transitions
  - Hard synchronization occurs at SOF and resets clock bit
  - Re-synchronization occurs at recessive-to-dominant (1-to-0) edges and adjusts the clock bit as necessary



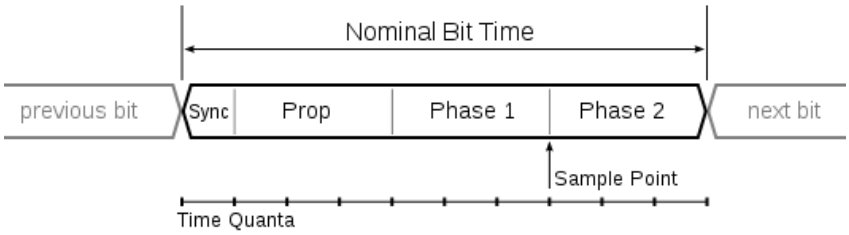
Bit timing

- Synchronization is done by dividing each bit of the frame into a number of segments:
  - Synchronization
  - Propagation,
  - Phase 1 and Phase 2
- The length of each phase segment can be adjusted based on network and node conditions.
- The sample point falls between phase buffer segment 1 and phase buffer segment 2, which helps facilitate continuous synchronization.
- Continuous synchronization in turn enables the receiver to be able to properly read the messages.

41

41

Bit timing



http://en.wikipedia.org/wiki/CAN_bus

42

42

Abstraction or protocol layers

- CAN standardizes only the lower layers
- The CAN protocol, like many networking protocols, can be decomposed into the following abstraction layers:

Application Layer
Object Layer - Message Filtering - Message and Status Handling
Transfer Layer - Fault Confinement - Error Detection and Signalling - Message Validation - Acknowledgment - Arbitration - Message Framing - Transfer Rate and Timing
Physical Layer - Signal Level and Bit Representation - Transmission Medium

43

43

Examples of CAN interfaces

- National Instruments controller area network (CAN) interfaces
- PEAK CAN Controllers
  - The Peak range of CAN interfaces provides simple and cost effective connections between PCs and CAN-networks and includes routers, extenders and adapters to the many CAN variants.

44

44

Summary

Comm. method	Shares clock	Num. of wires	Speed	Dist	Pros	Cons
UART	No	2	115Kbits/sec max	Medium, long	Simple; Widely supported; Large range of physical standard interfaces (TTL, RS-232, RS-422, RS-485);	It's asynchronous; Requires reasonable clock accuracy at both ends;
CAN	No	3	1 Mbits/sec	Long: 40m (1Mbit/sec) up to 10km (5Kbits/sec)	Highly reliable; Reduces amount of wiring; Multi-master capability;	Complex;
I2C	Yes	2	100Kbits/sec 400Kbits/sec fast mode	Short, medium (< 6")	Simple; Multi-master capability; Only 2 wires to support multiple devices; Robust in noisy or power-up/down situations;	More complex protocol than SPI; Harder to level-shift or opto-isolate due to bidirectional lines; Need for pull-up resistors can reduce power efficiency in some cases;
SPI	Yes	4	10-20Mbits/sec	Short	Fast, easy, simple; A lot of support; Self clocking; Flexible data word sizes;	Multiple devices need multiple select lines; No acknowledgement ability; No inherent arbitration; No flow control; Single master only;

45

45

Credits and References

- [1] Carmine Noviello, Mastering STM32, Second Edition, 2022. (Required, Book 1). Available to purchase online.
- [2] Joseph Yiu, The Definitive Guide to ARM Cortex-M0 and Cortex-M0+ Processors, 2nd Ed., 2015. (Book 2). Can be found online.
- http://www.best-microcontroller-projects.com/i2c-tutorial.html
- http://www.robot-electronics.co.uk/acatalog/I2C_Tutorial.html
- http://marco.guardigli.it/2010/10/hacking-your-car.html
- http://www.ni.com/white-paper/2732/en

46

46