ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN-ĐIỆN TỬ
BỘ MÔN KỸ THUẬT ĐIỆN TỬ

Embedded System Design

Chapter 4: Hardware design for an embedded system

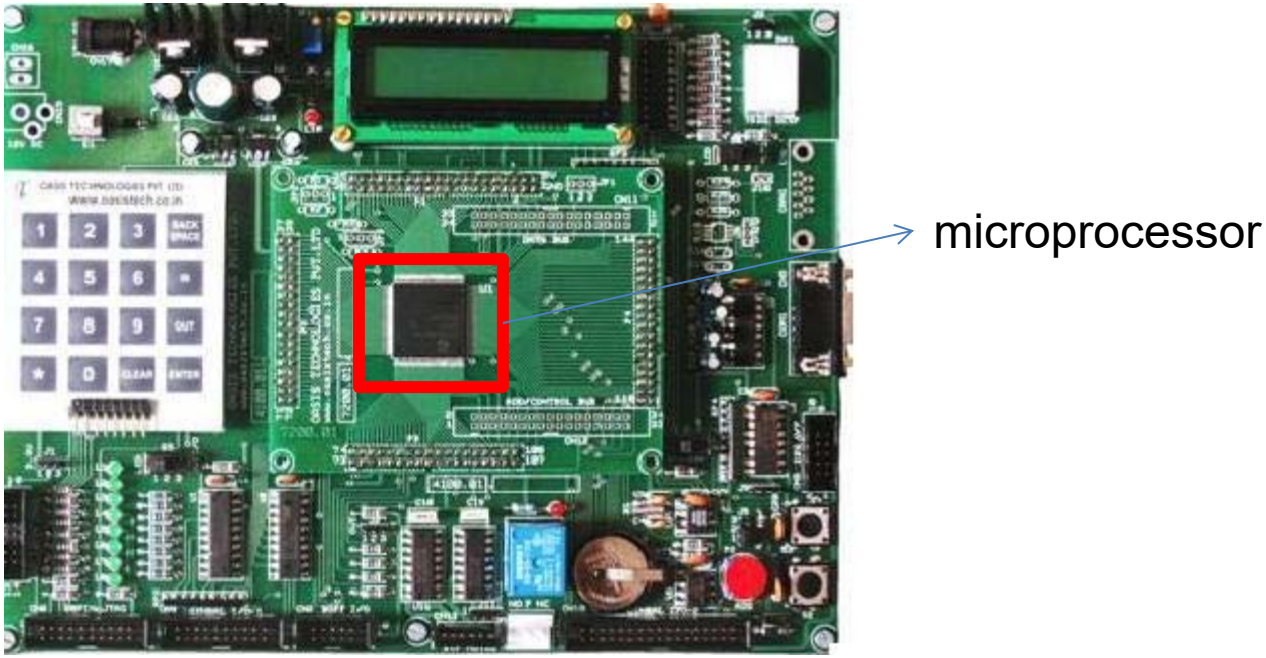
1. Hardware components
2. Design block diagrams
3. Micro controller
4. Board bus

1. Hardware components

1. Microprocessors/Microcontrollers
   - 8/16/32-bit microcontroller: PICs, ARMs
   - DSP
2. Peripherals
   - Input devices: button, switch, keyboard, mouse, touch-screen
   - Display devices: LED, text LCD, graphic LCD
   - Sensors: temperature, humidity, light, motion
   - Actuators: motor, solenoid, relay, FET, triac, SCR
   - Interfaces: UART, USB, I2C, SPI, Ethernet, Wifi, Bluetooth, Zigbee
3. Clock / reset circuits
4. Power supply
   - AC/DC adapters
   - Battery

1. Hardware components

- Microprocessors/Microcontrollers
  - Is a center processing unit
  - Control input devices, sensors, actuators, display devices
  - Process tasks, functions, and algorithms
  - Interface other systems



Example of an embedded system

1. Hardware components

- Microprocessors/Microcontrollers
  - The Intel MCS-51 (commonly referred to as 8051) is a Harvard architecture, CISC instruction set, single chip microcontroller (µC) series which was developed by Intel in 1980
  - PIC is a family of modified Harvard architecture microcontrollers made by Microchip Technology
  - ARM is a family of instruction set architectures for computer processors developed by British company ARM Holdings, based on a reduced instruction set computing (RISC) architecture.

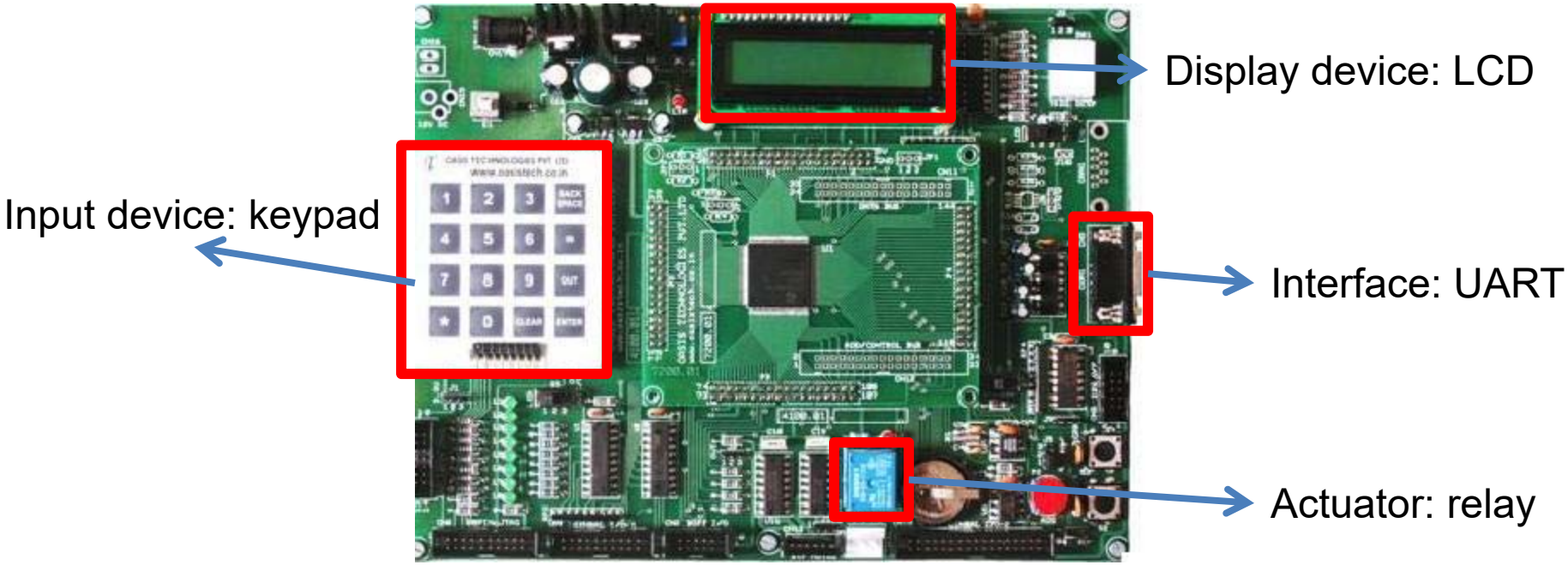
8051

PIC

ARM

1. Hardware components

- Peripherals
  - Input devices: button, switch, keyboard, mouse, touch-screen
  - Display devices: LED, text LCD, graphic LCD
  - Sensors: temperature, humidity, light, motion
  - Actuators: motor, solenoid, relay, FET, triac, SCR
  - Interfaces: UART, USB, I2C, SPI, Ethernet, Wifi, Bluetooth, Zigbee

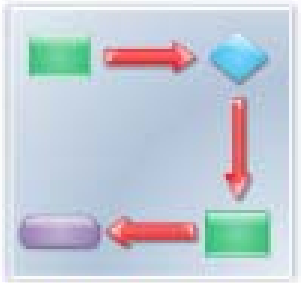


2. Design block diagram

- Block diagram
  - Is a diagram of a system in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks
  - Use to model the system graphically and show the relationships in the process.
  - presents a quick overview of major process steps and key process participants, as well as the relationships and interfaces.

Block 2D

Block 3D

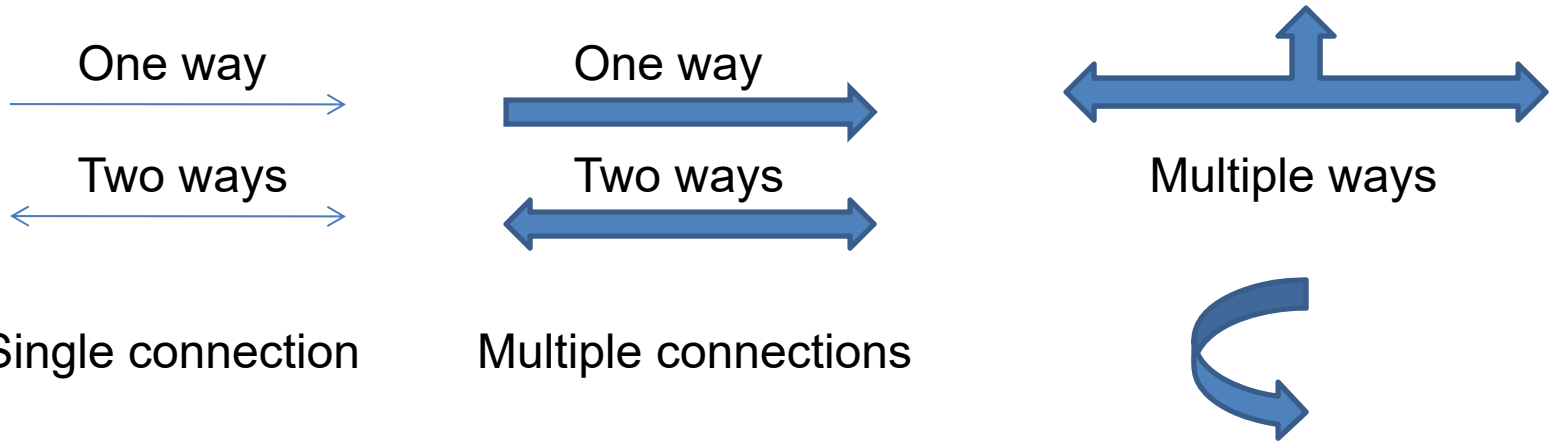


Highlight Shapes

2. Design block diagram

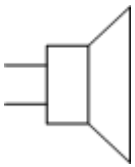
- Hardware block diagram
  - Use a rectangle for a hardware block

  - Use an arrow for a connection



2. Design block diagram

- Hardware block diagram
  - Use a symbol for a special block

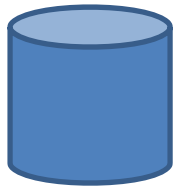

Speaker


Lamp

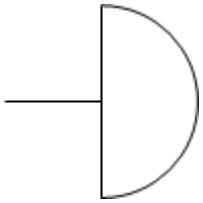

Energy

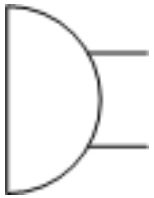

Network

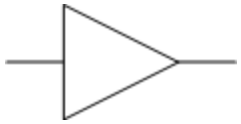

computer

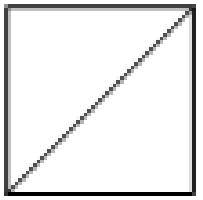

Database

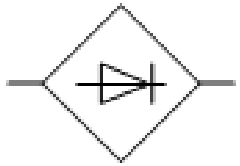

Antenna


Bell


Microphone


Amplifier


Converter


Rectifier

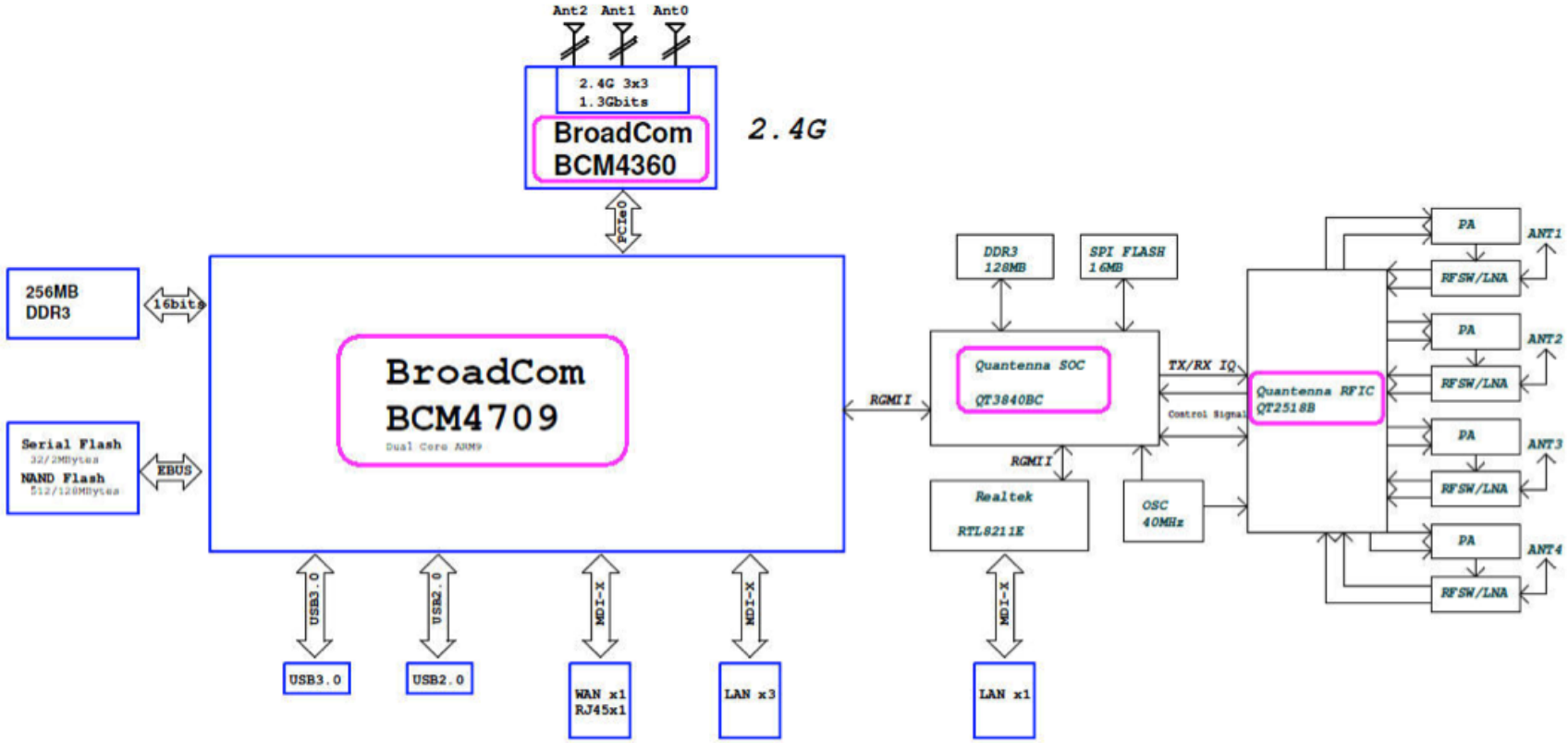
2. Design block diagram

- Block diagram features:
  - Block diagram name
  - Physical blocks
    - Block name
    - Have at least one connection
  - Connections
    - One direction / two directions/ multiple directions
    - Single / multiple connections
    - Data type of connections
  - Special block
    - Block name
    - Extra information

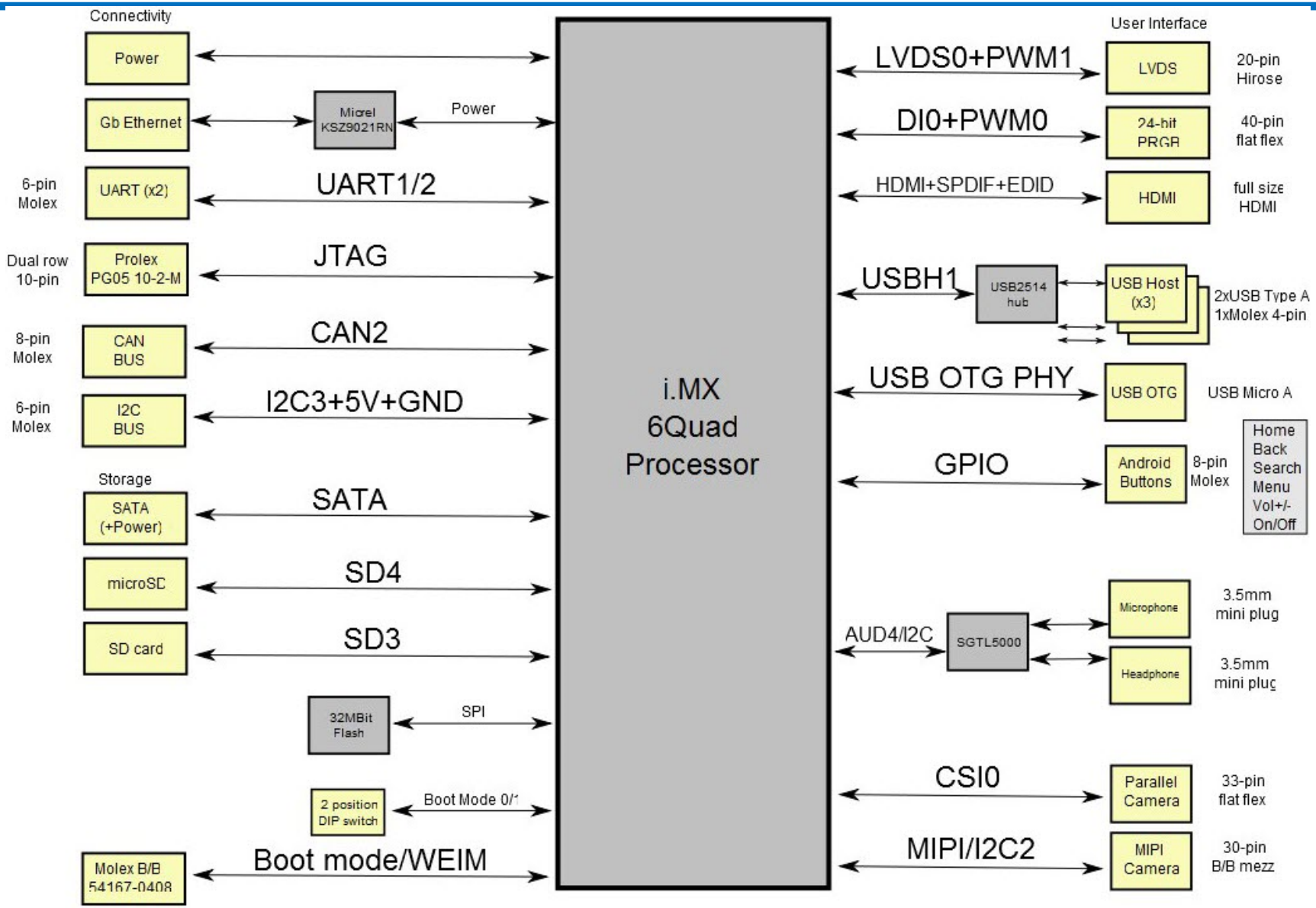
2. Design block diagram

- Recommendations
  1. CPU block is at the center
  2. Sensor/input blocks are at the left side
  3. Actuator blocks are at the right side
  4. User interface blocks are at the top
  5. Memory/ database/ blocks are at the bottom
  6. Use different colors for differently functional blocks
  7. Use symbols for special blocks
- Exceptions
  - Not enough space
  - Special systems such as SoC, NoC
  - Complex systems

Block Diagram

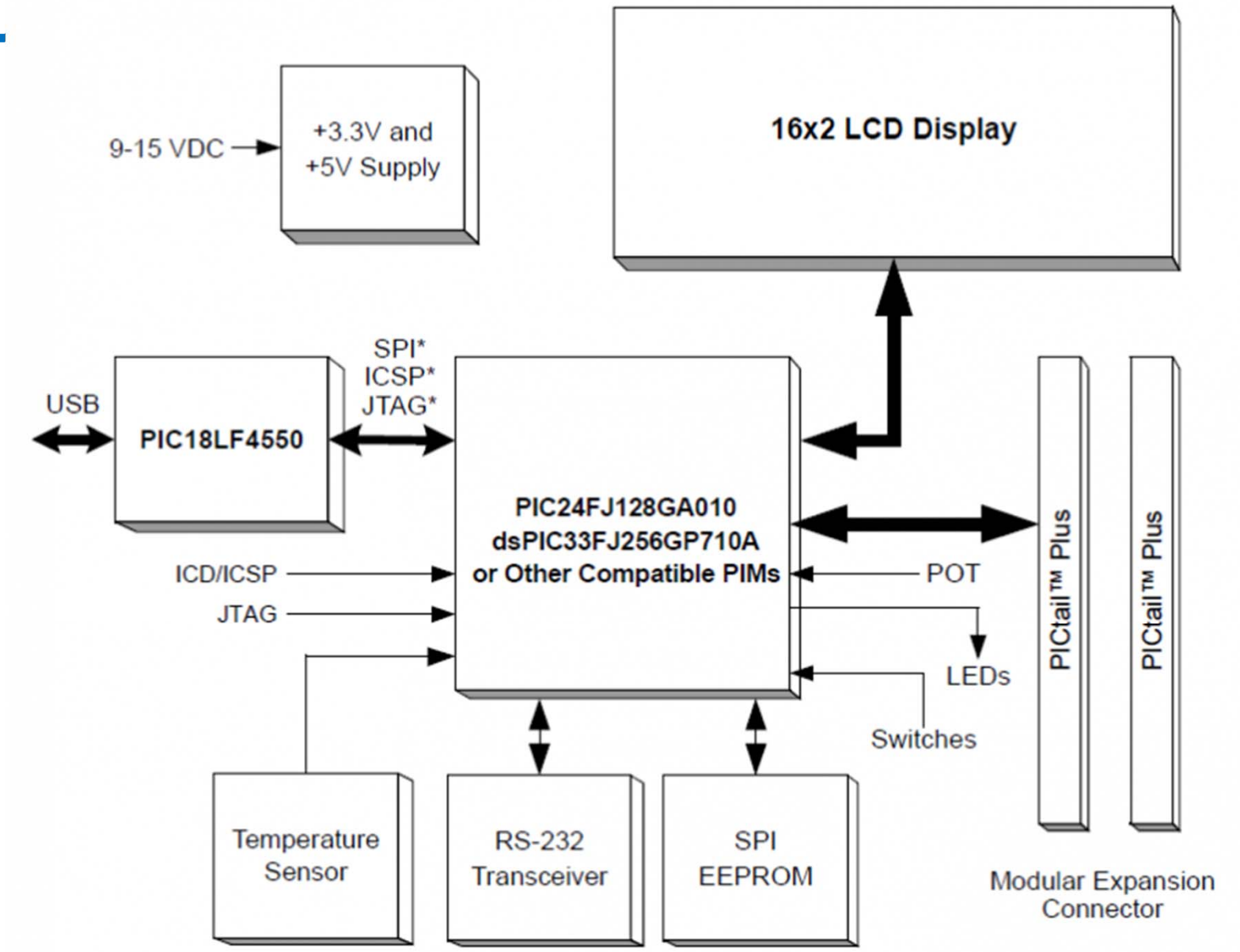


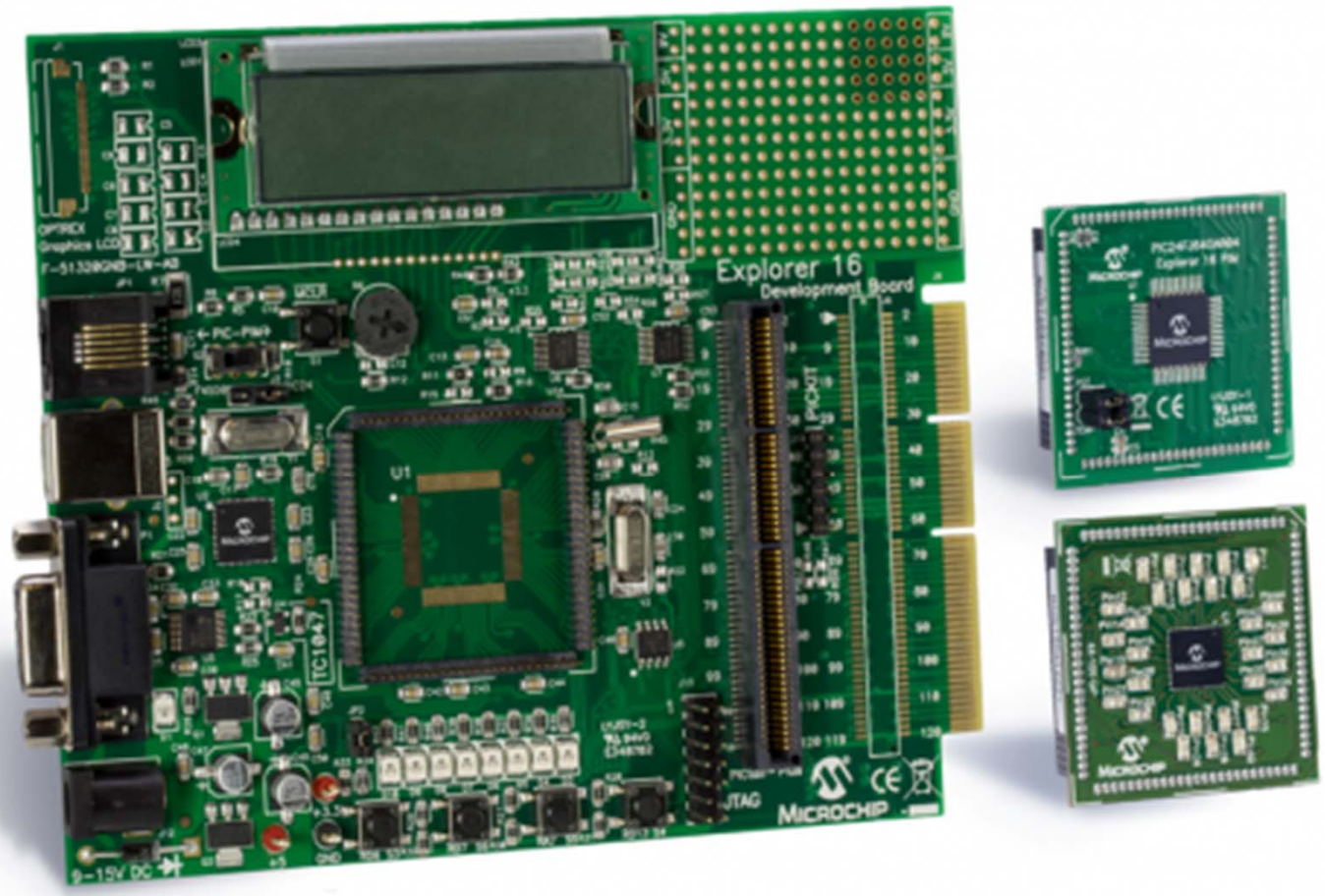
Sabre lite board



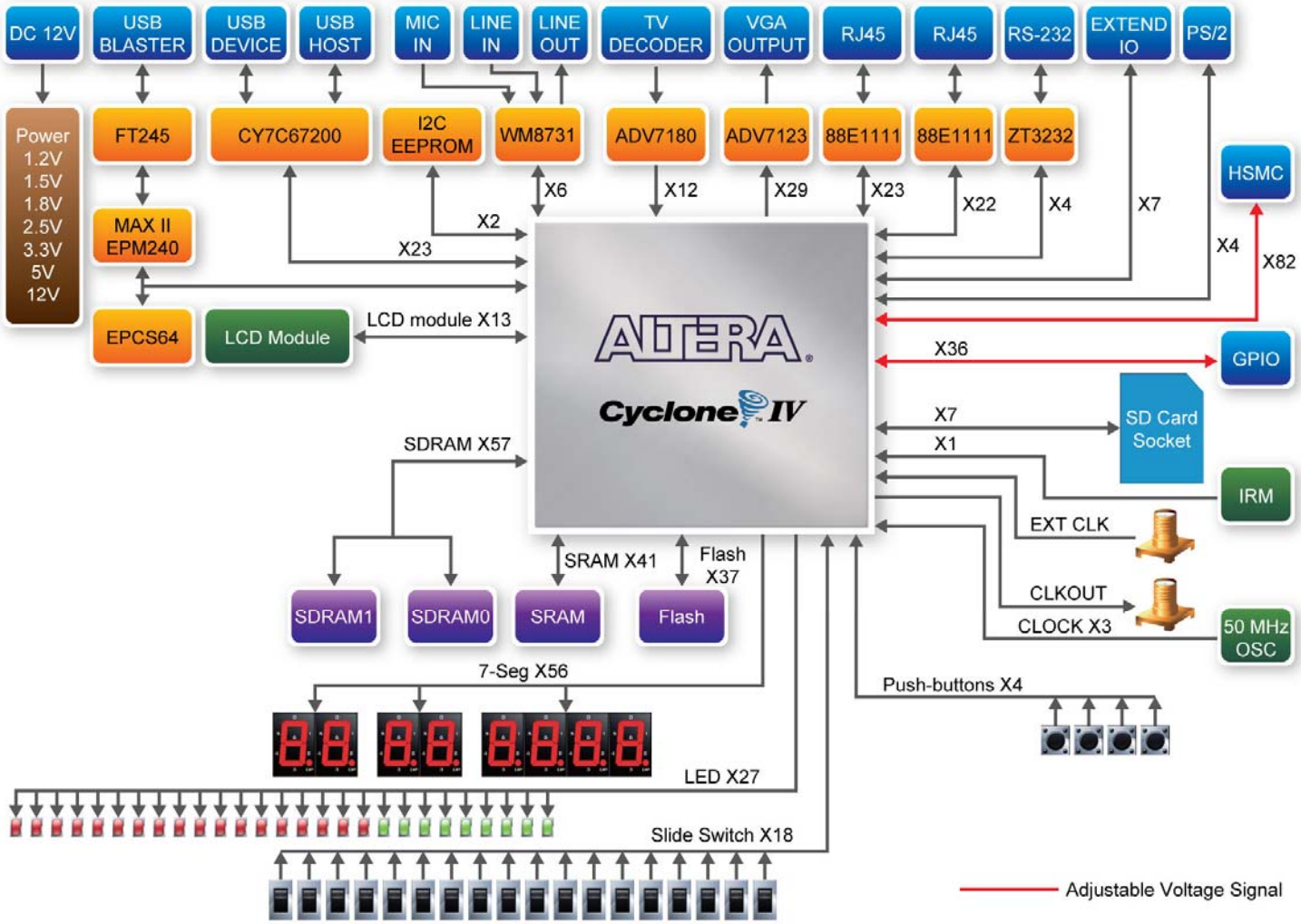
Sabre lite board

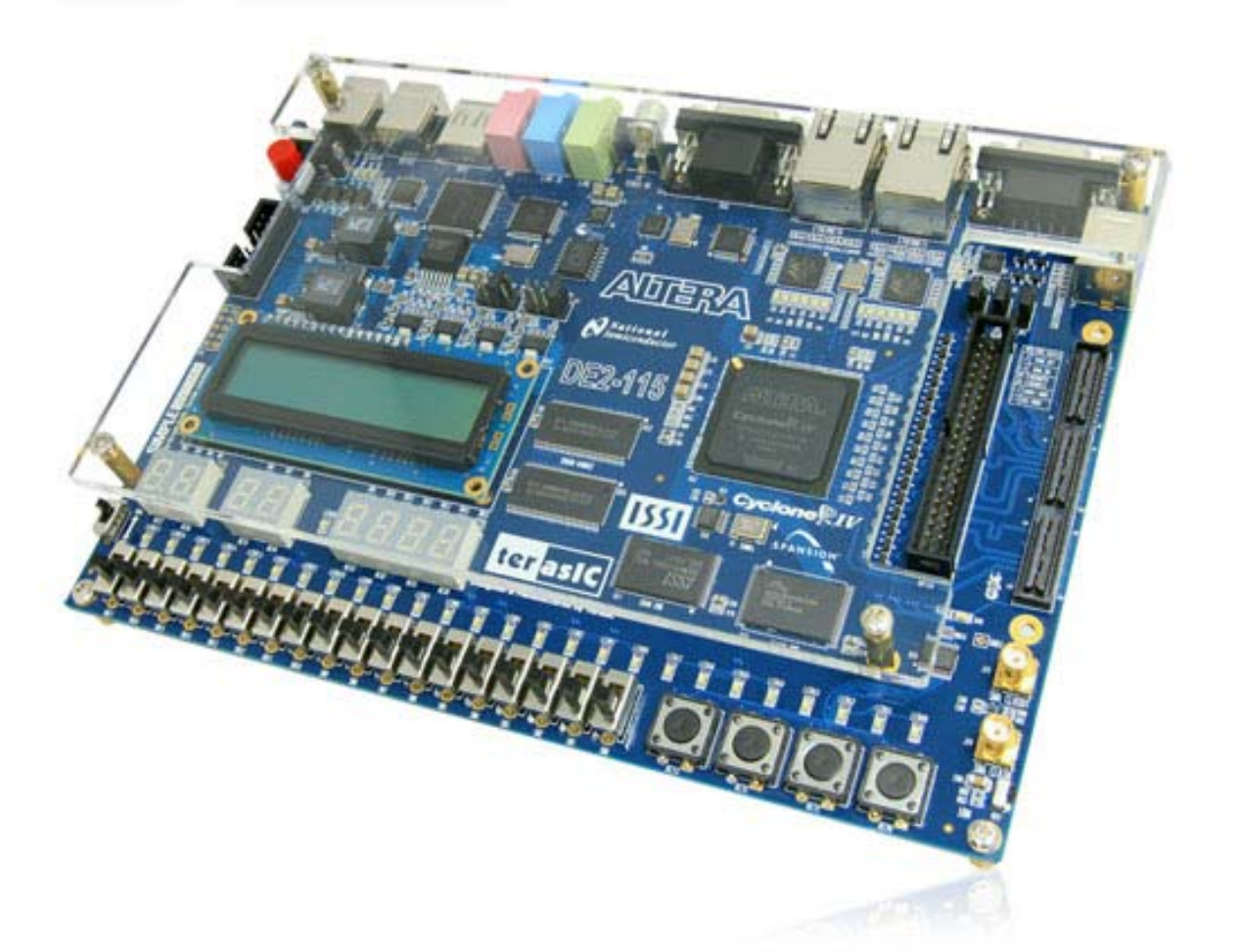
HIGH-LEVEL BLOCK DIAGRAM OF THE EXPLORER 16 DEVELOPMENT BOARD

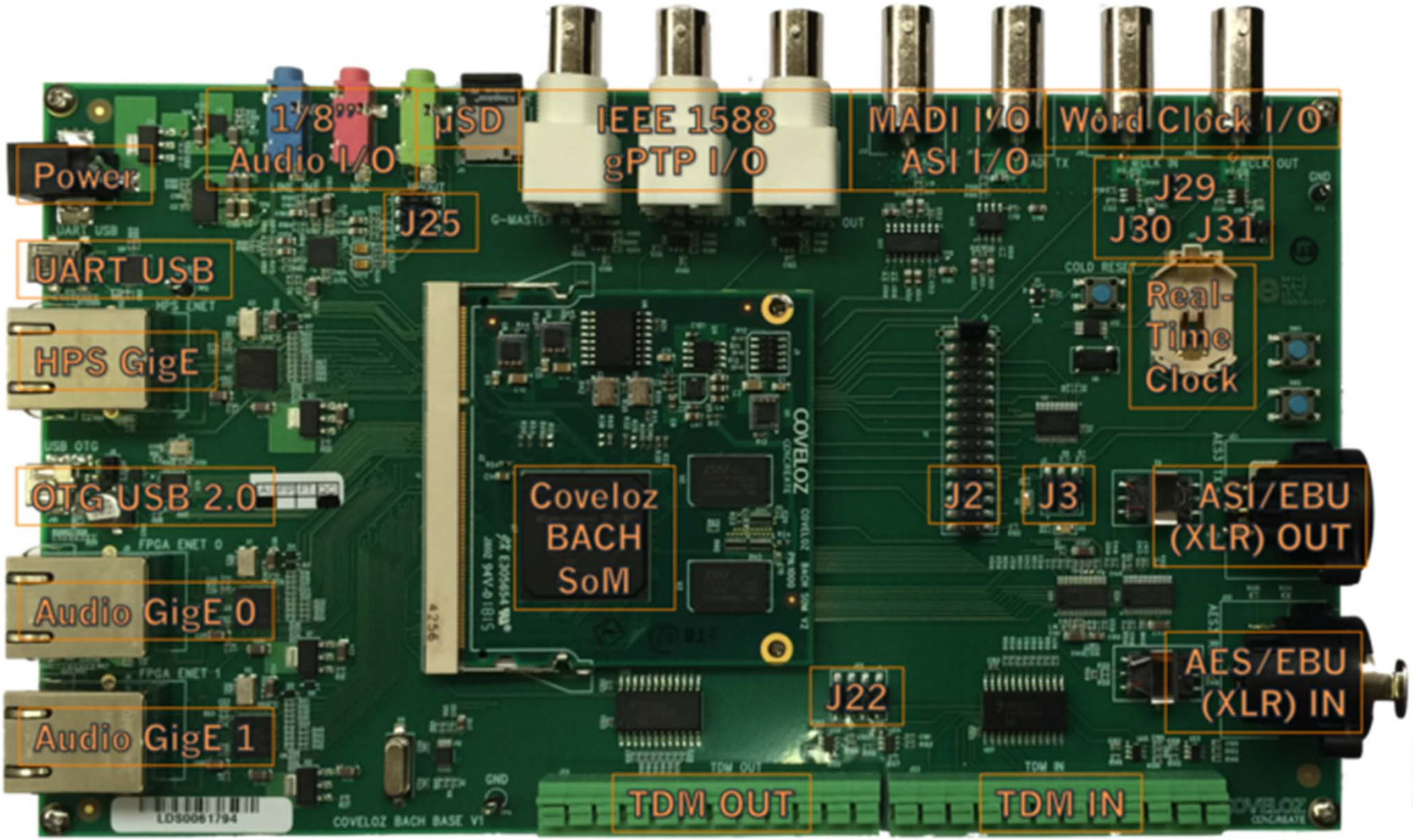



Explorer 16
Development Board
MICROCHIP

DE2 board

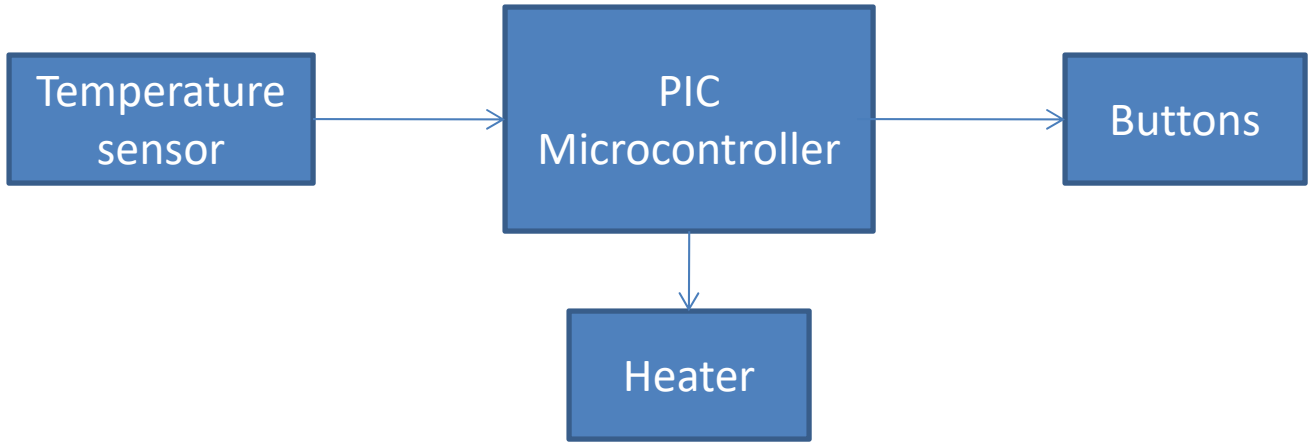

DC 12V
USB BLASTER
USB DEVICE
USB HOST
MIC IN
LINE IN
LINE OUT
TV DECODER
VGA OUTPUT
RJ45
RJ45
RS-232
EXTEND IO
PS/2
Power
1.2V
1.5V
1.8V
2.5V
3.3V
5V
12V
FT245
CY7C67200
I2C EEPROM
WM8731
ADV7180
ADV7123
88E1111
88E1111
ZT3232
HSMC
MAX II EPM240
X6
X12
X29
X23
X22
X4
X7
X2
X23
X4
X82
EPCS64
LCD Module
LCD module X13
ALTERA
Cyclone IV
X36
GPIO
SD Card Socket
X7
X1
SDRAM X57
IRM
EXT CLK
SRAM X41
Flash X37
CLKOUT
CLOCK X3
50 MHz OSC
SDRAM1
SDRAM0
SRAM
Flash
7-Seg X56
Push-buttons X4
LED X27
Slide Switch X18
Adjustable Voltage Signal


ALTERA
DE2-115
ISSI
terasIC


1/8" Audio I/O
µSD
IEEE 1588 gPTP I/O
MADI I/O ASI I/O
Word Clock I/O
Power
J25
J29 J30 J31
UART USB
Real-Time Clock
HPS GigE
OTG USB 2.0
Coveloz BACH SoM
J2
J3
ASI/EBU (XLR) OUT
Audio GigE 0
AES/EBU (XLR) IN
Audio GigE 1
J22
TDM OUT
TDM IN

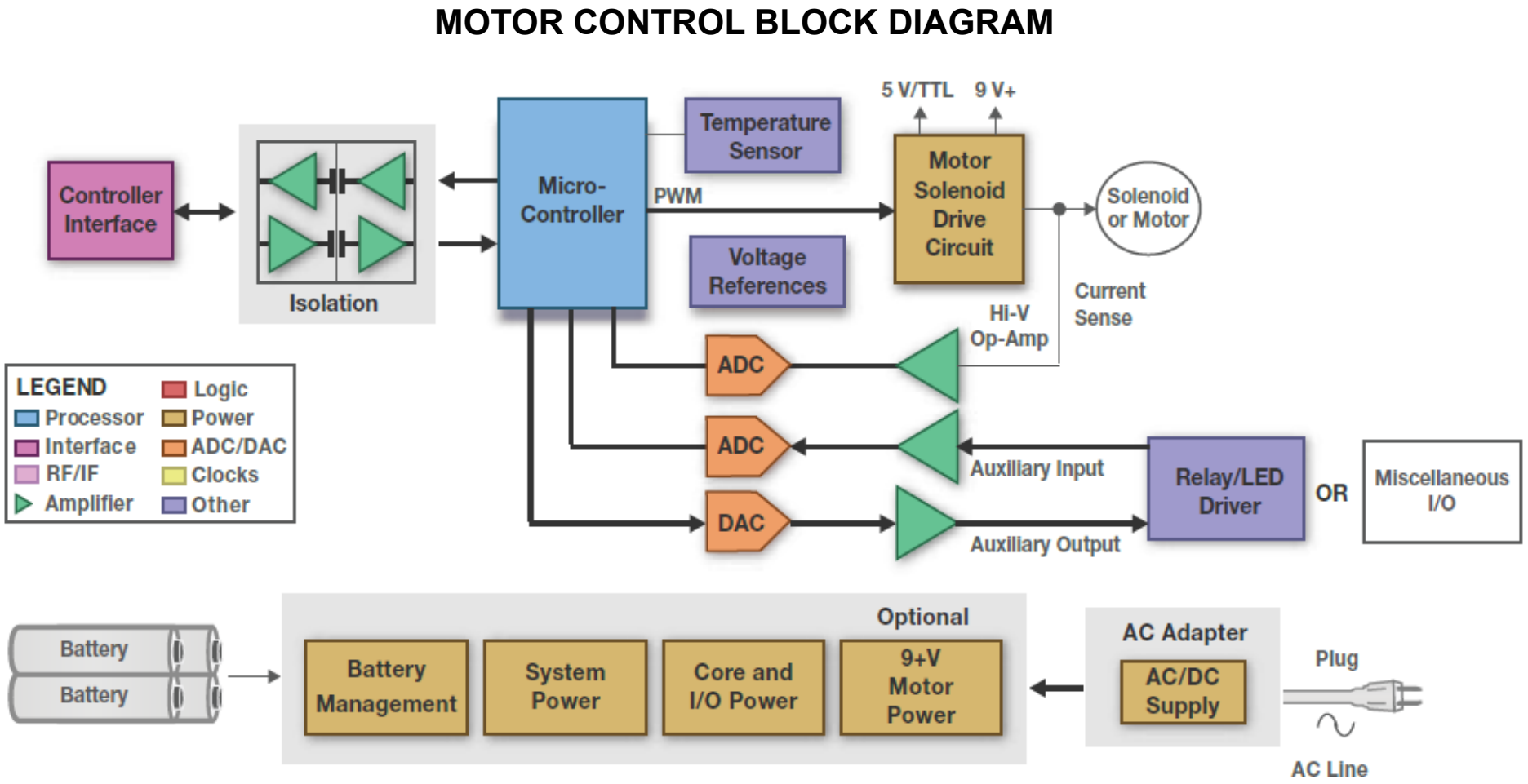
Hardware block diagram – Example 1

- Poor example



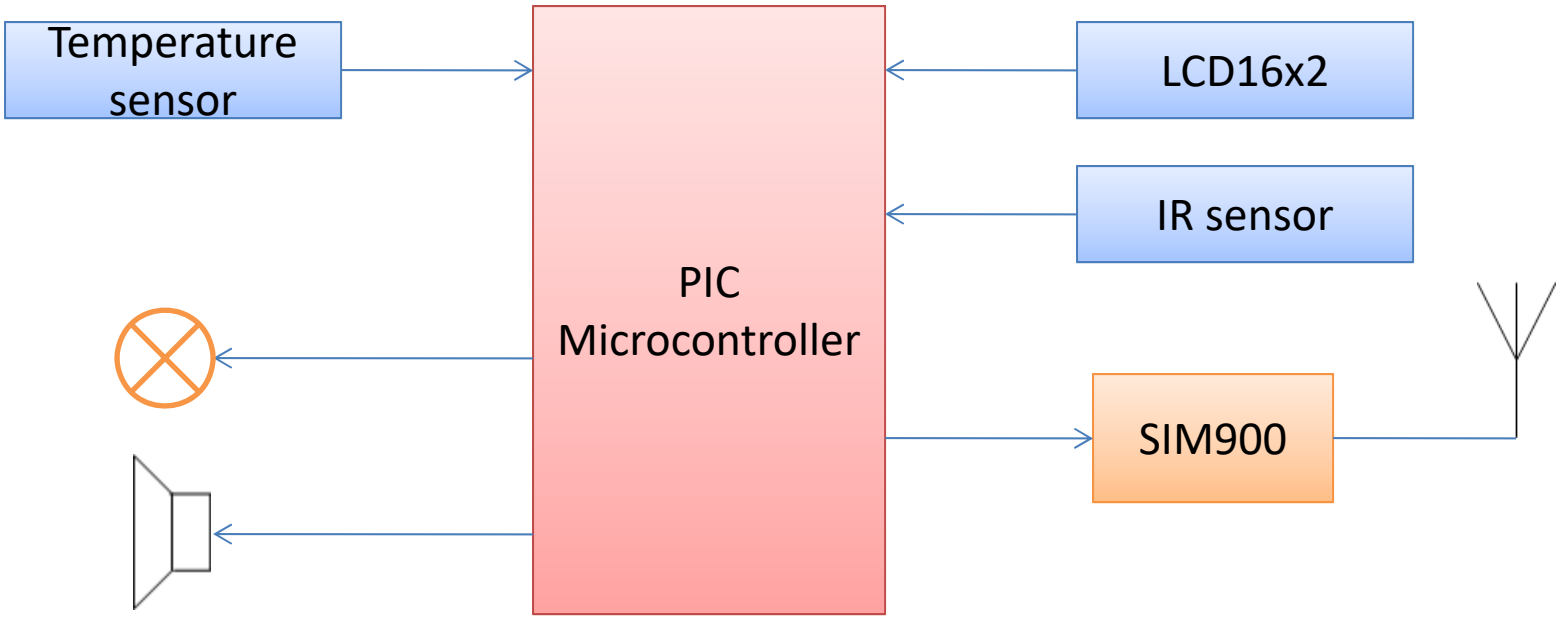
  - No block diagram name
  - Wrong direction of connection
  - Problem of single / multiple connections
  - No data type of connections

Hardware block diagram – Example 2



Hardware block diagram – Example

Temperature monitor board

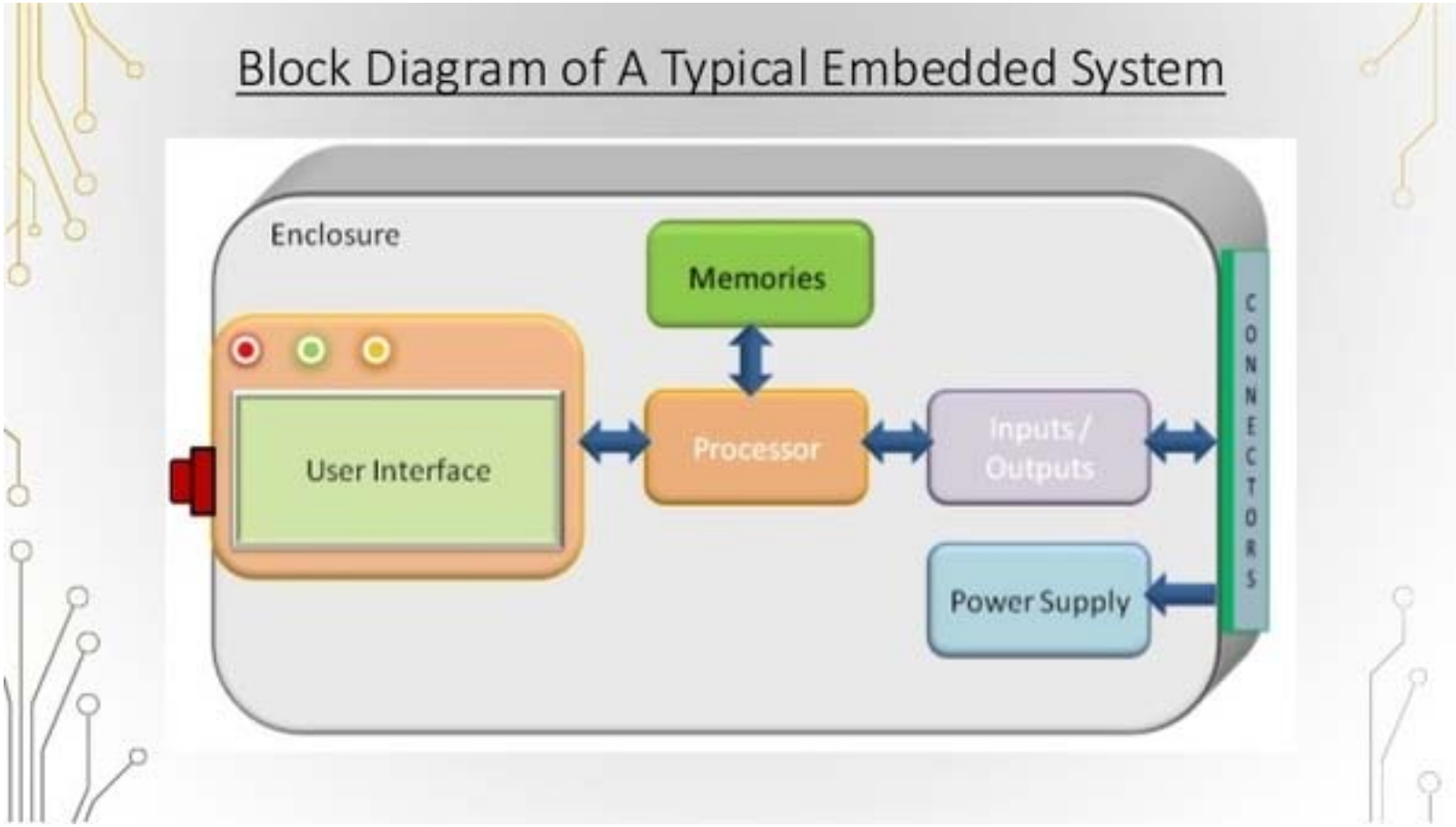


=> Make this block diagram better!

Team work

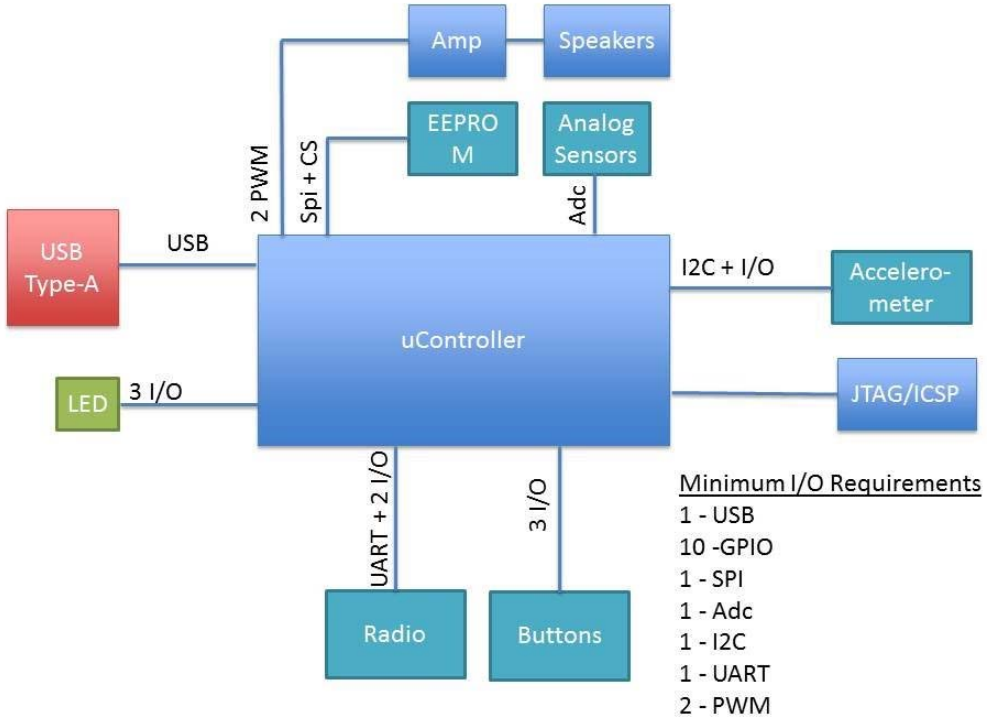
- Draw the hardware block diagram of your project

Embedded system hardware



Micro controller selection

- Step1. Make a list of required hardware interfaces



Micro controller selection

- Examine the software architecture
  - the algorithms require floating point mathematics?
  - Do we need special hardware like FPU, DMA
  - Are there any high frequency control loops or sensors?
  - how long and how often each task will need to run?
  - What interrupts will we need?
  - How many timer will we need?

Micro controller selection

- Select the architecture
  - Do we need to process 16/32 bit data often?
  - Can the application get by with 8/16 bit architectures?
  - Are there libraries that support the architecture we chose?

Micro controller selection

- Identify Memory Needs
  - What is the largest data structure?
  - How much is the size of the RTOS/libraries we will use?

Micro controller selection

- Start searching for microcontrollers
  - Supplier like Digikey, Arrow or other trusty website.
  - Chip manufacturer website (microchip, ST,TI, Atmel, etc)
  - https://en.wikipedia.org/wiki/List_of_common_microcontrollers

Micro controller selection

- Examine Costs and Power Constraints
  - If the device will be powered from a battery and mobile, low-power feature is absolutely necessary
  - Price is very important with large quantity project

Micro controller selection

- Check part availability
  - Are they kept in stock at multiple distributors or is there 6 – 12 week lead time?
  - What are your requirements for availability?
  - When will this part be obsolete? (life cycle)

Micro controller selection

- Step 8: Select a development kit
  - Is there any development kit available?
- Step 9: Investigate compilers and tools, resource
  - Can we have C compiler, IDE and programming tools?
  - Does this micro controller has good support (community, libraries, resource)

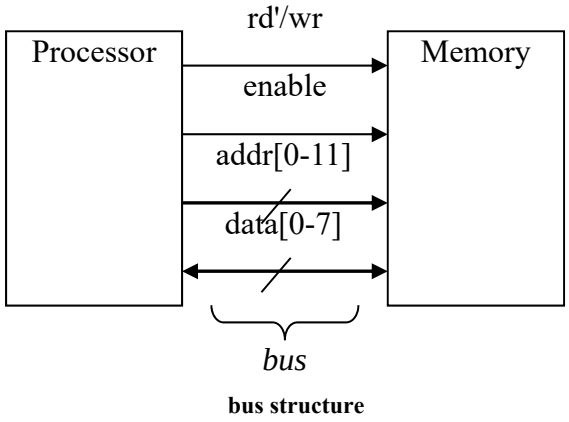
Memory

Type	*Volatile?*	*Writeable?*	*Erase Size*	*Max Erase Cycles*	*Cost (per Byte)*	*Speed*
SRAM	Yes	Yes	Byte	Unlimited	Expensive	Fast
DRAM	Yes	Yes	Byte	Unlimited	Moderate	Moderate
Masked ROM	No	No	n/a	n/a	Inexpensive	Fast
PROM	No	Once, with a device programmer	n/a	n/a	Moderate	Fast
EPROM	No	Yes, with a device programmer	Entire Chip	Limited (consult datasheet)	Moderate	Fast

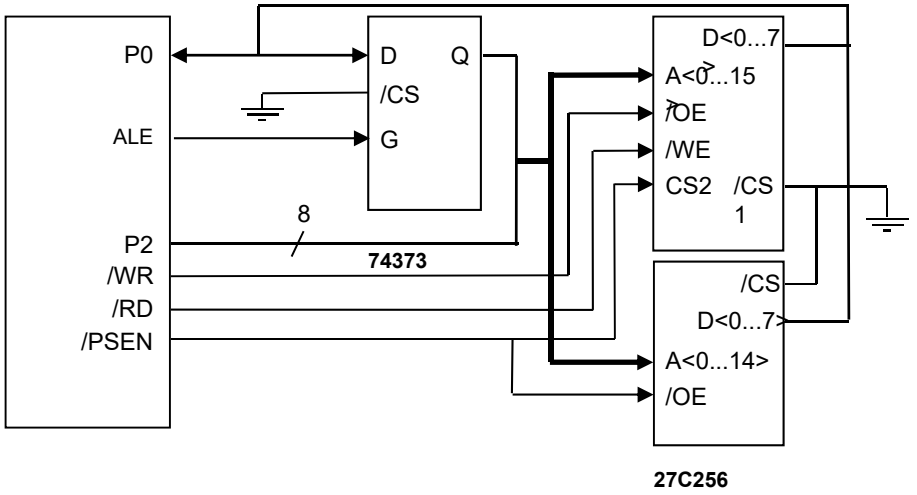
Memory

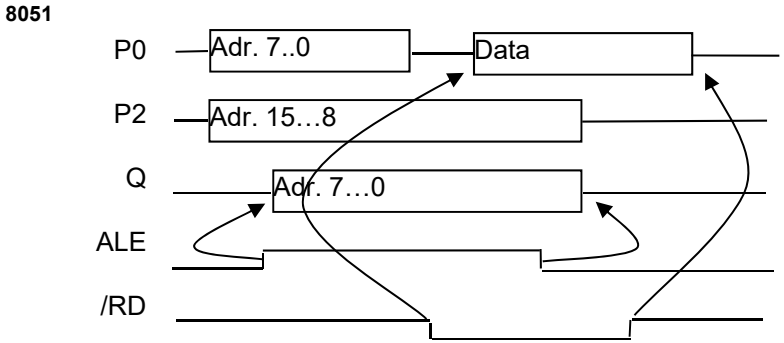
Type	*Volatile?*	*Writeable ?*	*Erase Size*	*Max Erase Cycles*	*Cost (per Byte)*	*Speed*
EEPROM	No	Yes	Byte	Limited (consult datasheet)	Expensive	Fast to read, slow to erase/write
Flash	No	Yes	Sector	Limited (consult datasheet)	Moderate	Fast to read, slow to erase/write
NVRAM	No	Yes	Byte	Unlimited	Expensive (SRAM + battery)	Fast

Memory mapped bus

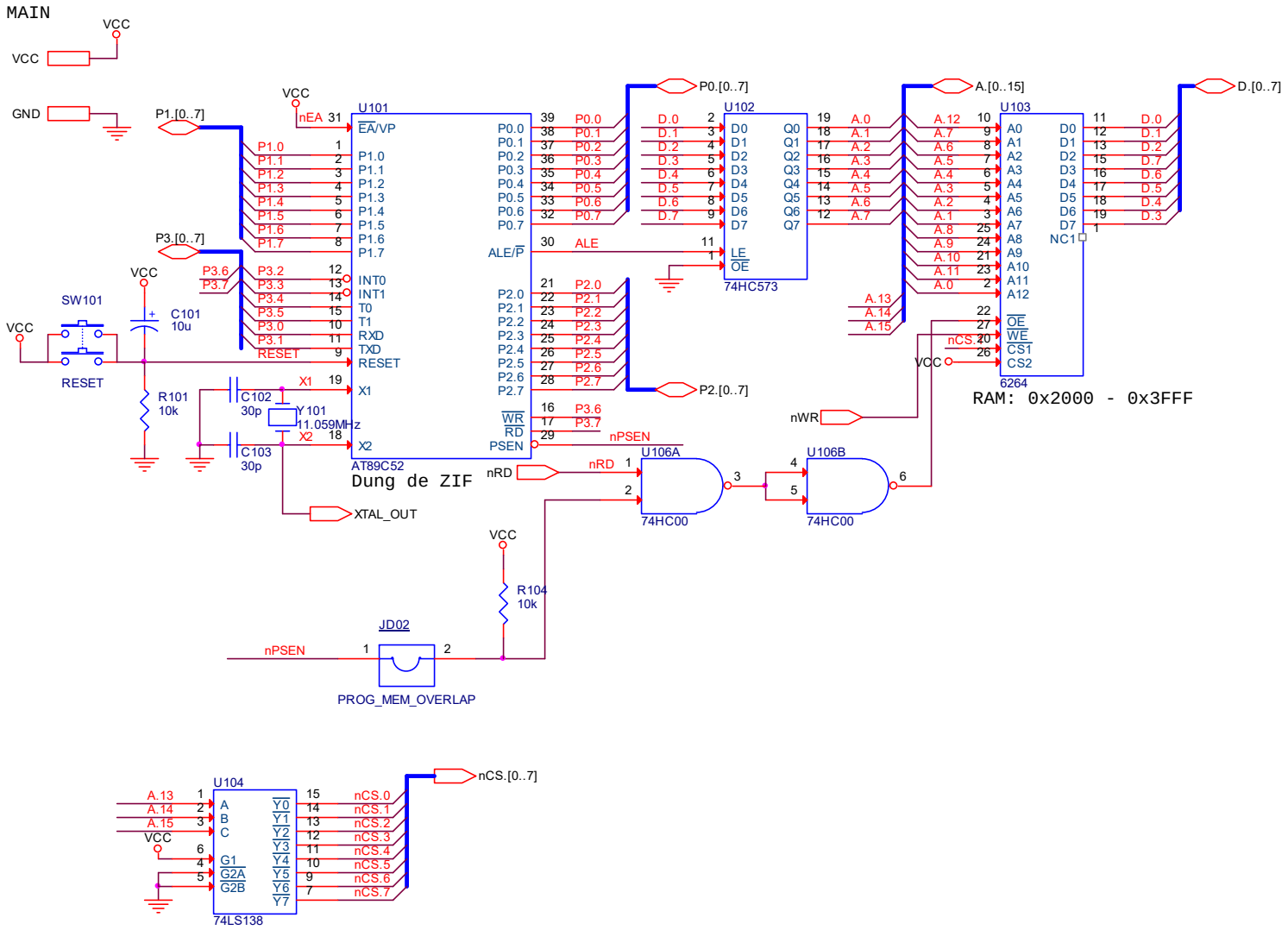


bus structure

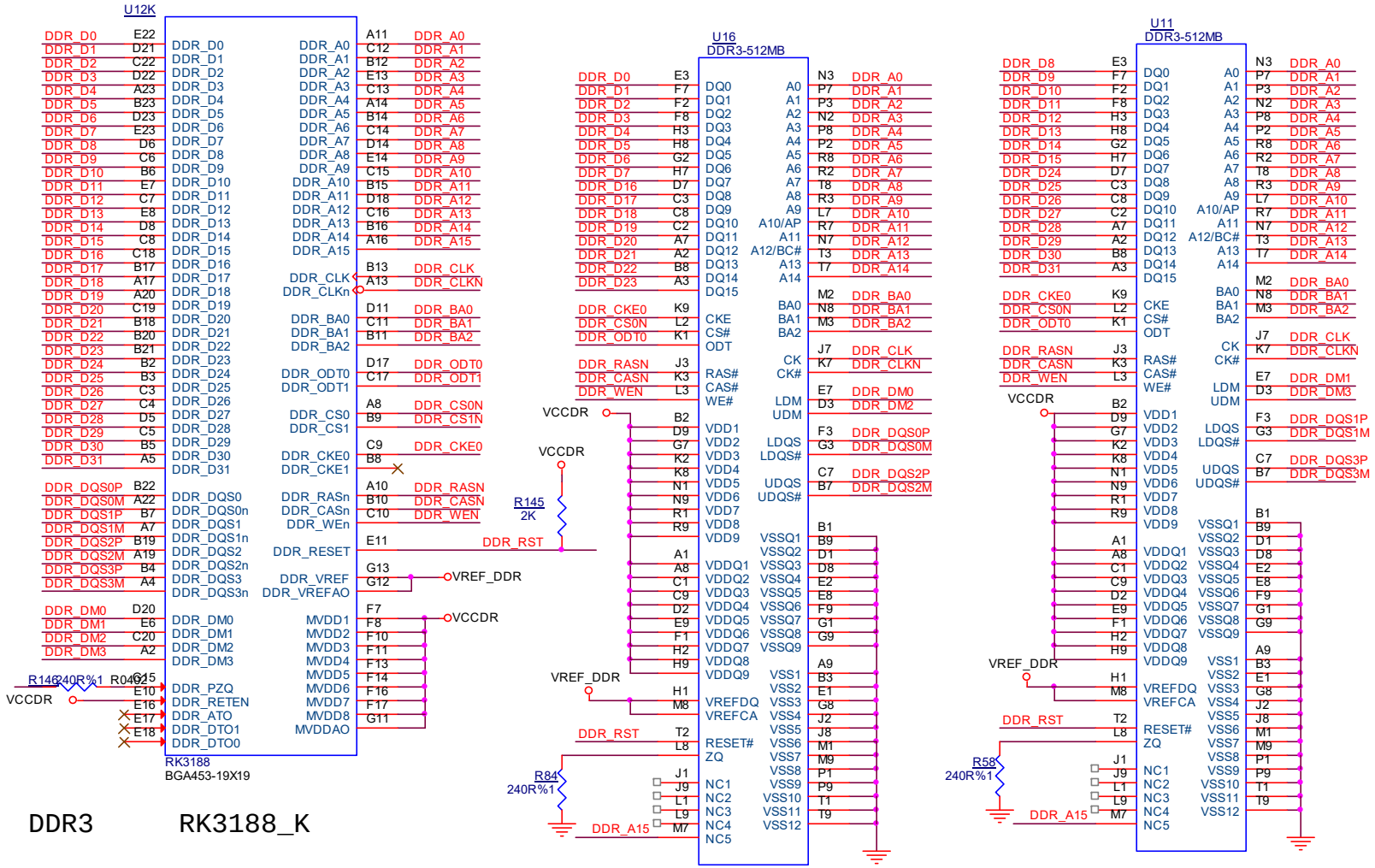




Memory mapped bus with 8051

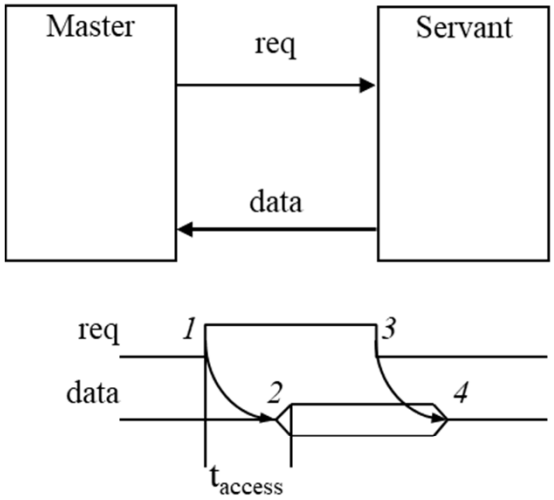


Memory bus



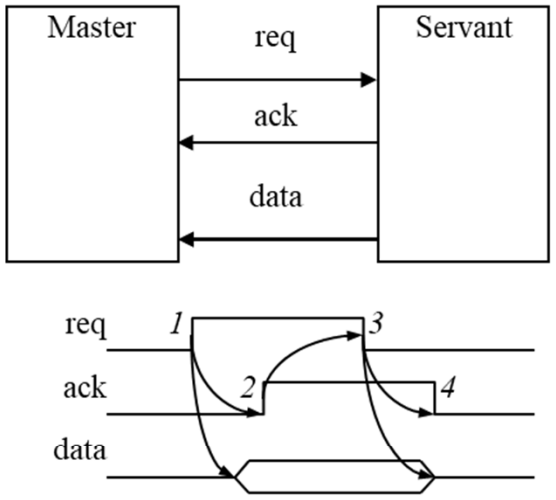
DDR3 RK3188_K

Basic protocol concepts



1. Master asserts *req* to receive data
2. Servant puts data on bus **within time t_{access}**
3. Master receives data and deasserts *req*
4. Servant ready for next request

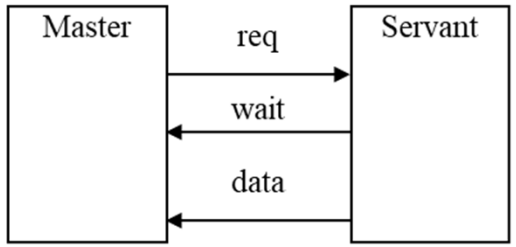
Strobe protocol

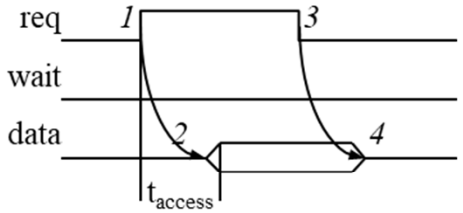


1. Master asserts *req* to receive data
2. Servant puts data on bus **and asserts *ack***
3. Master receives data and deasserts *req*
4. Servant ready for next request

Handshake protocol

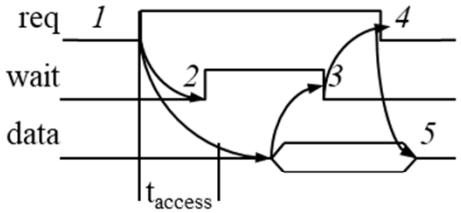
A strobe/handshake compromise





1. Master asserts *req* to receive data
2. Servant puts data on bus **within time t_{access}** (wait line is unused)
3. Master receives data and deasserts *req*
4. Servant ready for next request

Fast-response case



1. Master asserts *req* to receive data
2. Servant can't put data within t_{access}, **asserts *wait*** ack
3. Servant puts data on bus and **deasserts *wait***
4. Master receives data and deasserts *req*
5. Servant ready for next request

Slow-response case

Parallel communication

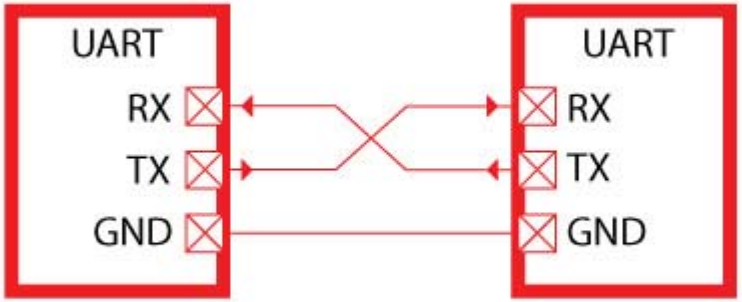
- Multiple data, control, and possibly power wires
  - One bit per wire
- High data throughput with short distances
- Typically used when connecting devices on same IC or same circuit board
  - Bus must be kept short
    - long parallel wires result in high capacitance values which requires more time to charge/discharge
    - Data misalignment between wires increases as length increases
- Higher cost, bulky

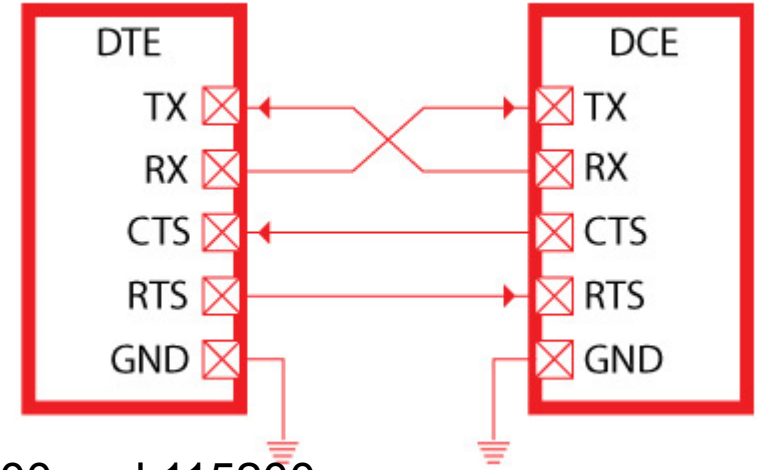
Question:
List some parallel communications and peripherals you know?

Serial communication

- Single data wire, possibly also control and power wires
- Words transmitted one bit at a time
- Higher data throughput with long distances
  - Less average capacitance, so more bits per unit of time
- Cheaper, less bulky
- More complex interfacing logic and communication protocol
  - Sender needs to decompose word into bits
  - Receiver needs to recompose bits into word
  - Control signals often sent on same wire as data increasing protocol complexity

UART connection



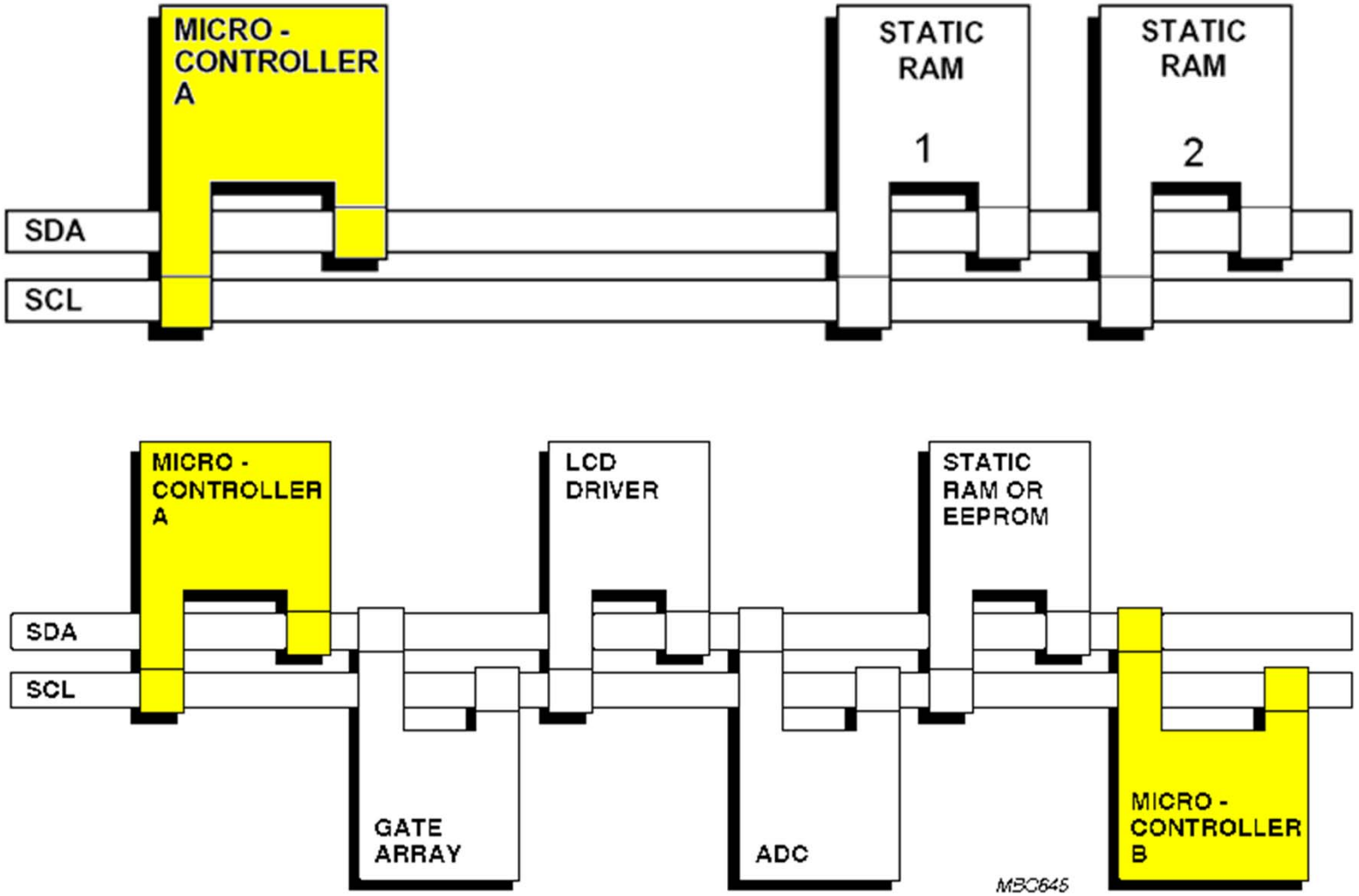


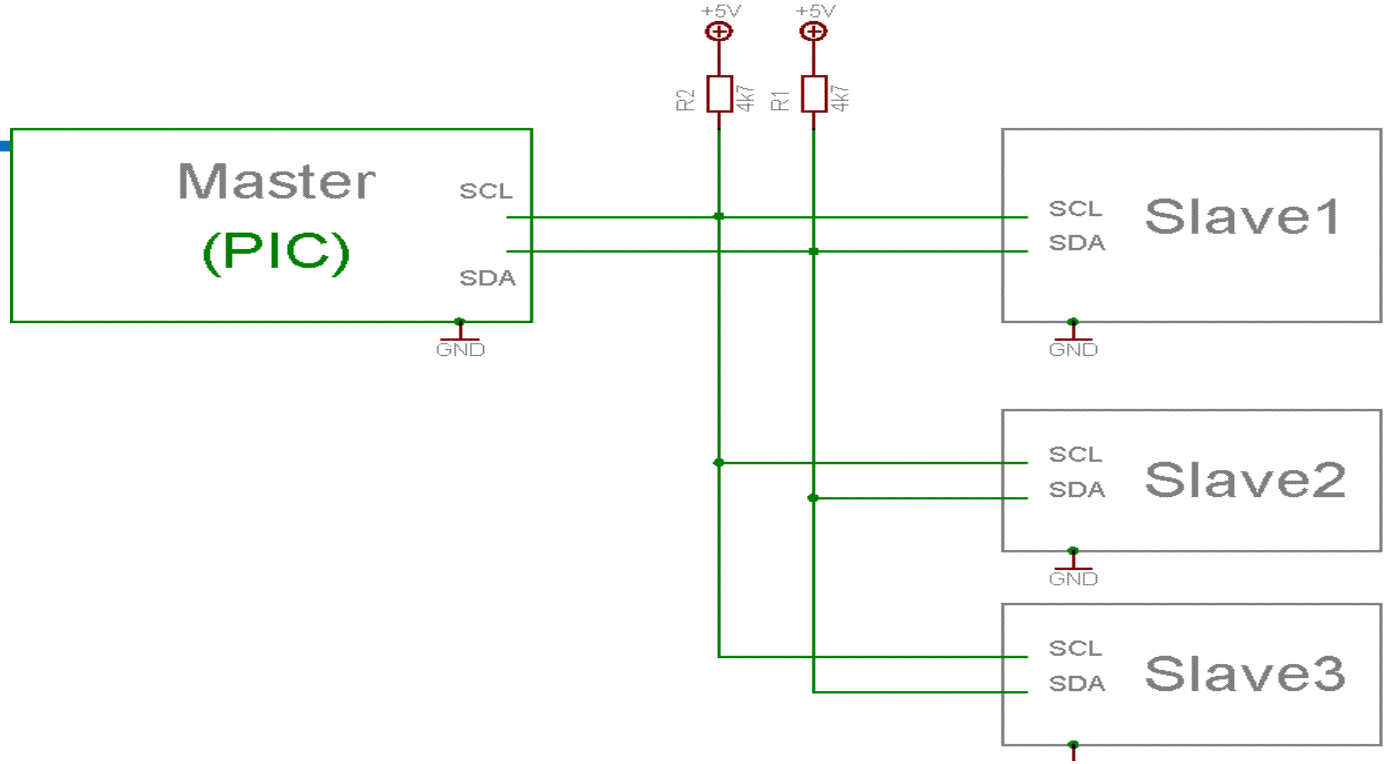
Standard baudrate:
1200, 2400, 4800, 9600, 19200, 38400, 57600 and 115200

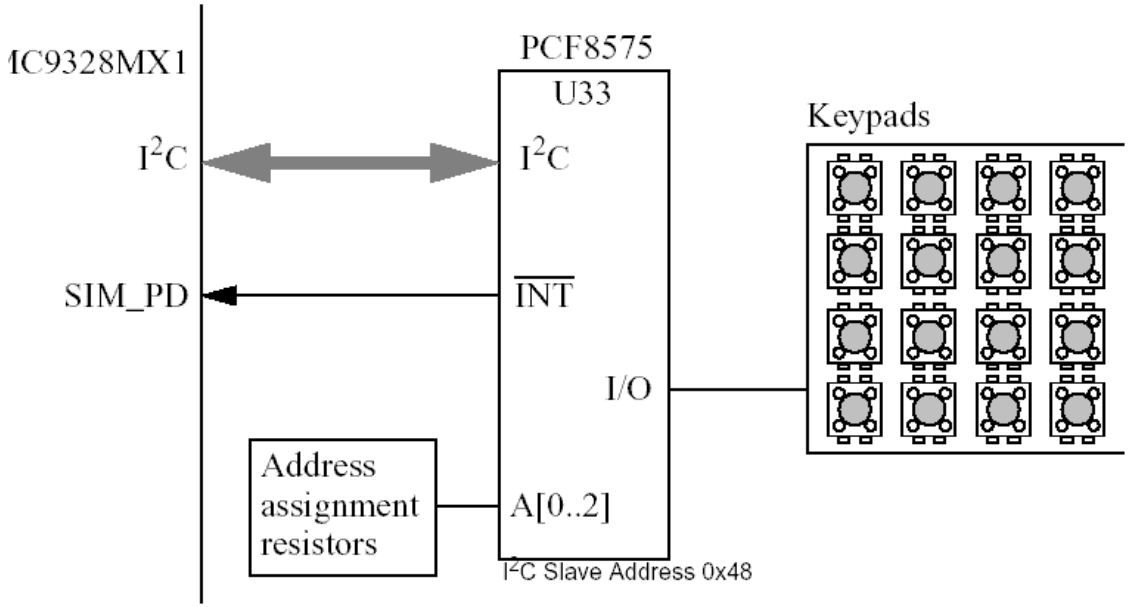
Find and list all the modules and sensor you know that use UART interface?

I2C bus

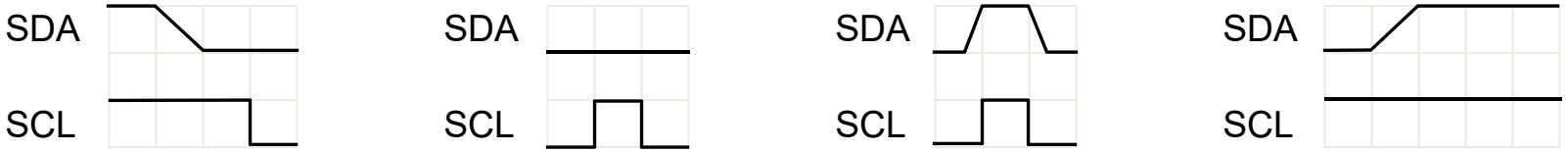
- I^2C (Inter-IC)
  - Two-wire serial bus protocol developed by Philips Semiconductors nearly 20 years ago
  - Enables peripheral ICs to communicate using simple communication hardware
  - Data transfer rates up to 100 kbits/s and 7-bit addressing possible in normal mode
  - 3.4 Mbits/s and 10-bit addressing in fast-mode
  - Common devices capable of interfacing to I^2C bus:
    - EPROMS, Flash, and some RAM memory, real-time clocks, watchdog timers, and microcontrollers

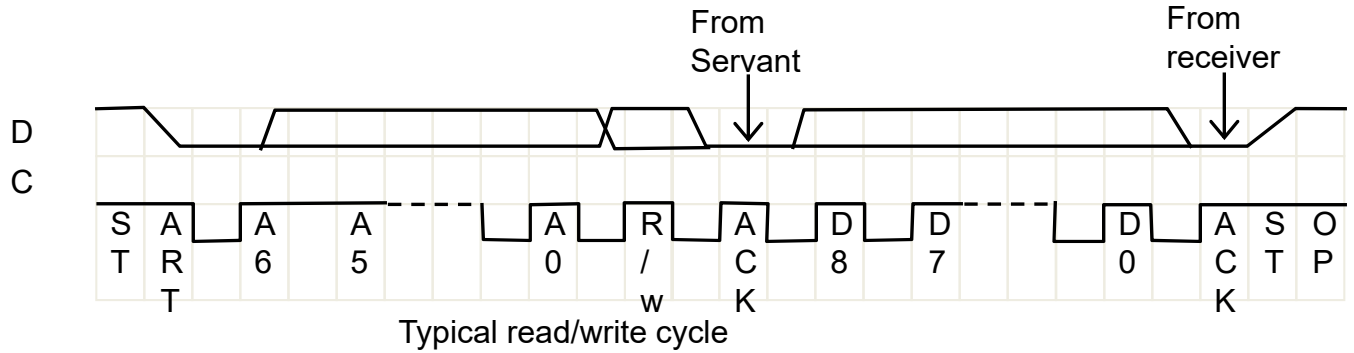

MICRO -
CONTROLLER
A
STATIC
RAM
1
STATIC
RAM
2
SDA
SCL
MICRO -
CONTROLLER
A
LCD
DRIVER
STATIC
RAM OR
EEPROM
SDA
SCL
GATE
ARRAY
ADC
MICRO -
CONTROLLER
B
MBC645


+5V
+5V
R2
4K7
R1
4K7
Master
(PIC)
SCL
SDA
GND
SCL
SDA
Slave1
GND
SCL
SDA
Slave2
GND
SCL
SDA
Slave3

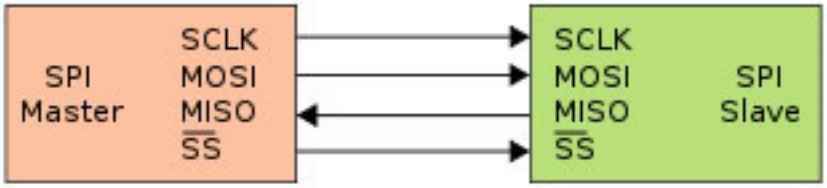

IC9328MX1
I2C
SIM_PD
PCF8575
U33
I2C
INT
I/O
A[0..2]
Address
assignment
resistors
I2C Slave Address 0x48
Keypads

I2C bus structure





SPI bus



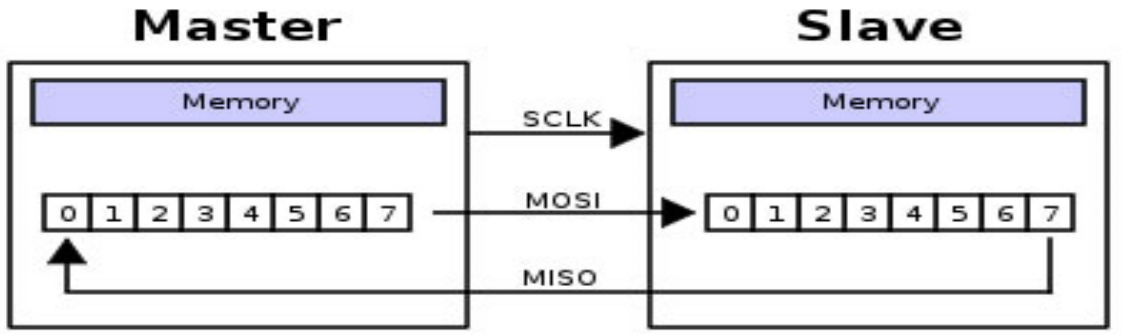
- A 4-wire communications bus
- Typically communicate across short distances
- Supports
  - Single master
  - Multiple slaves
- Synchronized
  - Communications are “clocked”

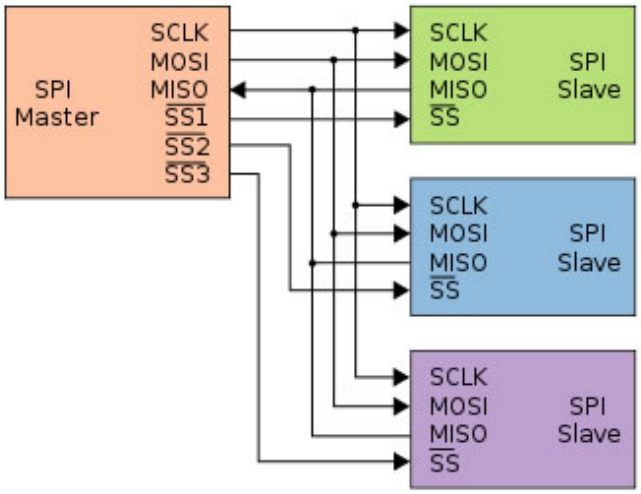
Bus wires

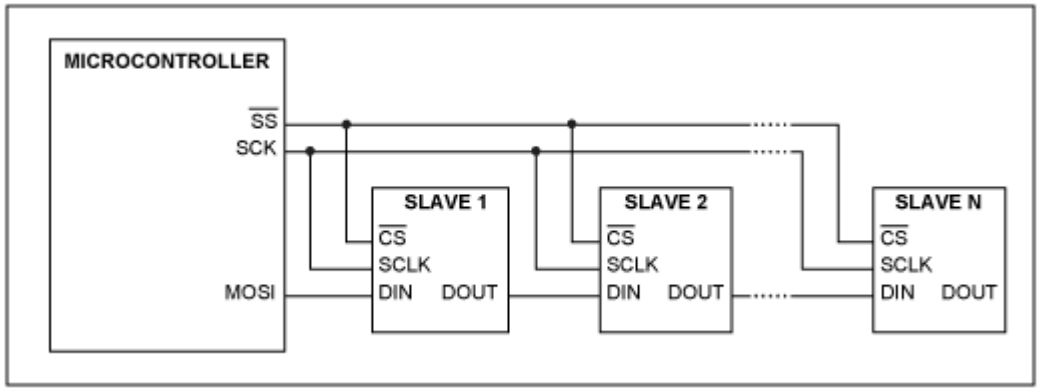
- Master-Out, Slave-In (MOSI)
- Master-In, Slave-Out (MISO)
- System Clock (SCLK)
- Slave Select/Chip Select (SS1#, …, SS#n or CS1, …, CSn)

- Always full-duplex
  - Communicates in both directions simultaneously
  - Transmitted (or received) data may not be meaningful
- Multiple Mbps transmission speeds
  - 0-50 MHz clock speeds not uncommon
- Transfer data in 4 to 16 bit characters
- Supports multiple slaves

Bus configuration



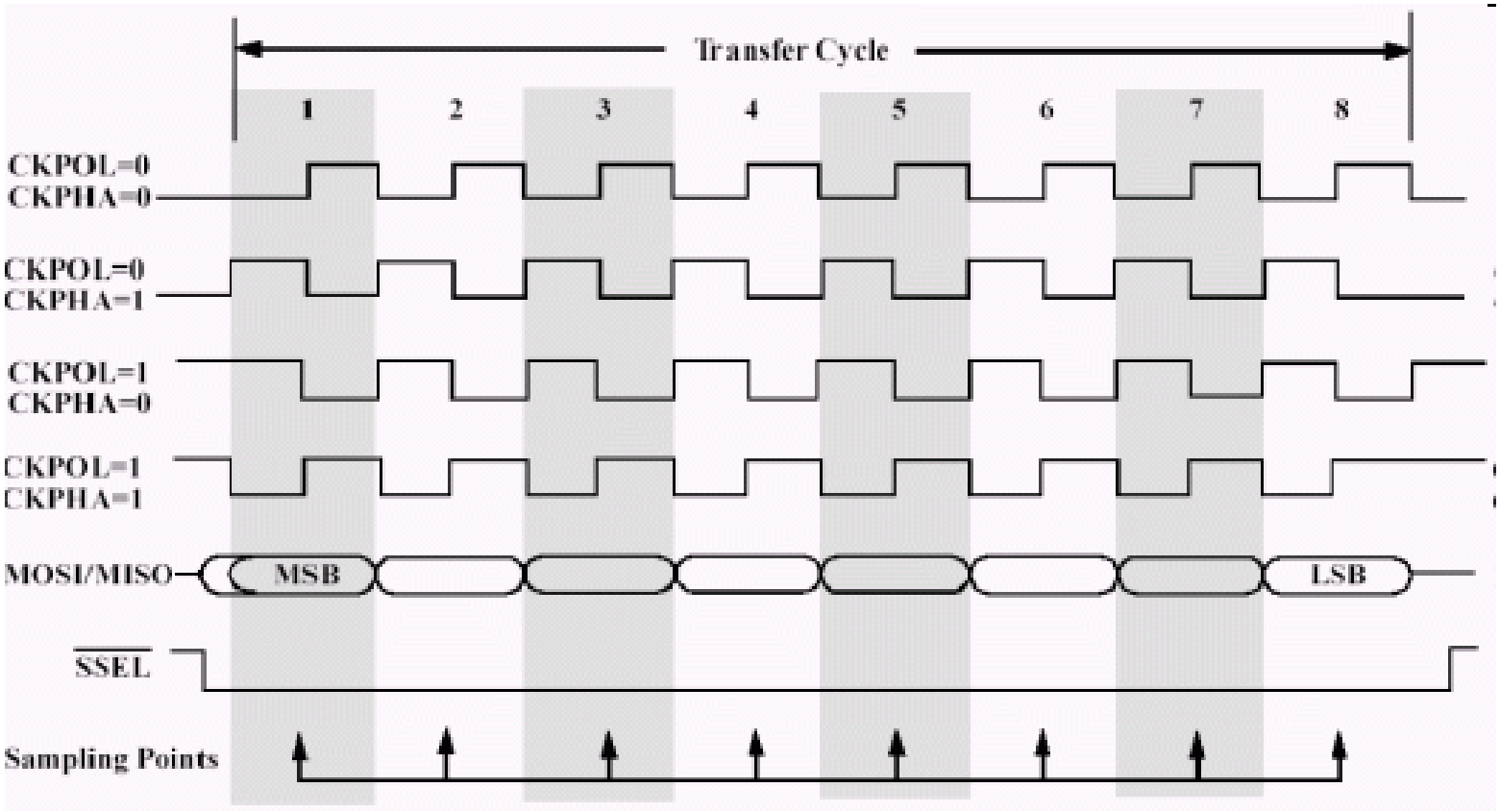




SPI clocking

- Four clocking “modes”
  - Two phases
  - Two polarities
- Master and *selected* slave must be in the same mode
- During transfers with slaves A and B, Master must
  - Configure clock to Slave A’s clock mode
  - Select Slave A
  - Do transfer
  - Deselect Slave A
  - Configure clock to Slave B’s clock mode
  - Select Slave B
  - Do transfer
  - Deselect Slave B
- Master reconfigures clock mode on-the-fly!

SPI clock modes



SPI pros and cons

- Pros
  - Fast for point-to-point connections
  - Easily allows streaming/constant data inflow
  - No addressing in protocol, so it's simple to implement
  - Broadly supported

- Cons
  - Slave select/chip select makes multiple slaves more complex
  - No acknowledgement (can't tell if clocking in garbage)
  - No inherent arbitration
  - No flow control (must know slave speed)

Other board bus

- One wire
- USB
- PCI
- LVDS