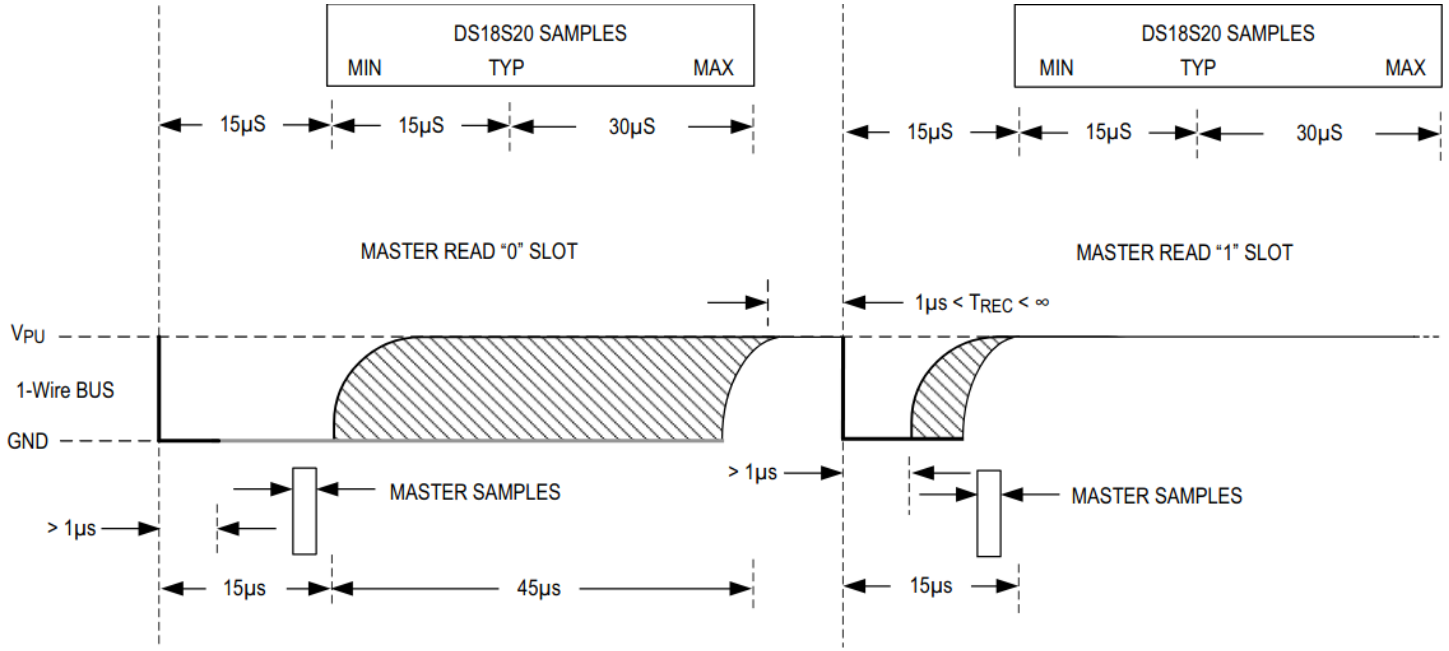
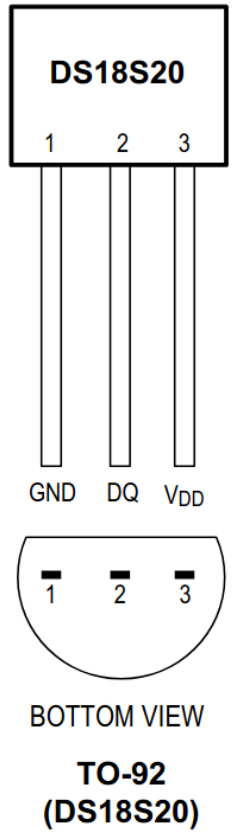
1-Wire Digital thermometer implementation



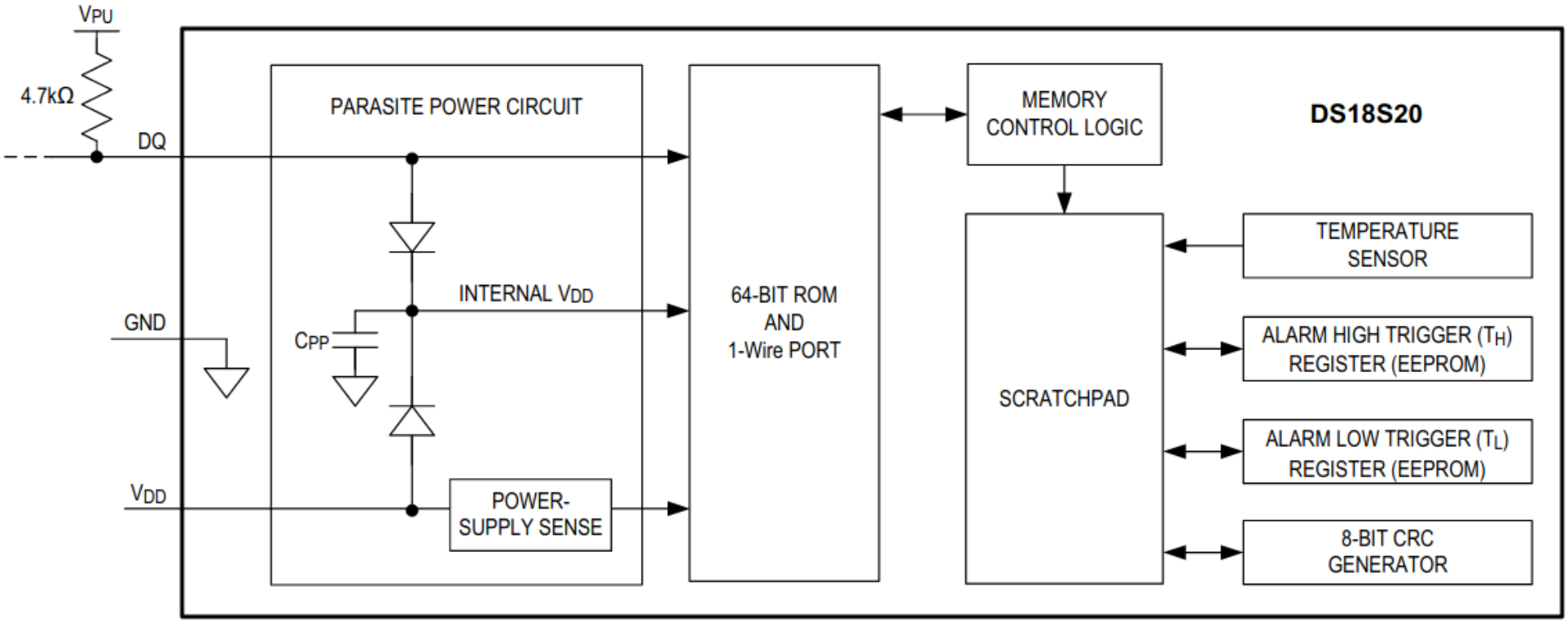
DS18S20 Digital thermometer

- **The DS18S20 is a digital themometer from MAXIM**
- It measures from -55°C to 85°C
- It comes in a 3 PIN TO-92 package
- **It transfers the measured temperature using the 1-Wire protocol**
- The data is coded over 16 bits among which the 8 MSB code the sign.
- It is able to store alarm for low and high temperatures (not used here)
- It can be driven in a parasitic power mode (not used here)
- In the component you have
  - 1 ROM
  - 1 RAM called the scratchpad



TO-92 (DS18S20)

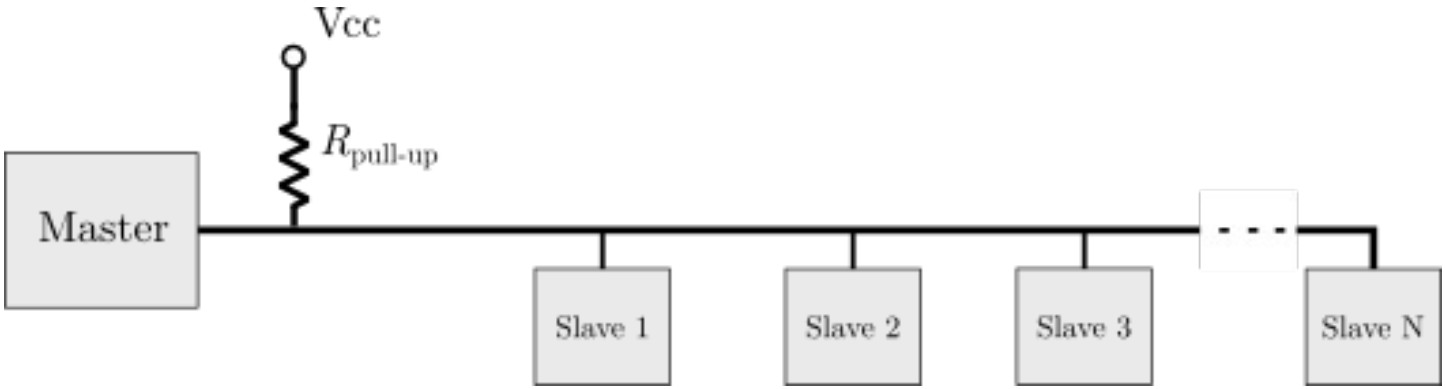
DS18S20 Digital thermometer



- **The ROM stores the identification code of the component**
- The scratchpad stores the informations acquired by the thermometer in a 8 Bytes array: temperature, alarms, CRC...

1-Wire : physical arrangement

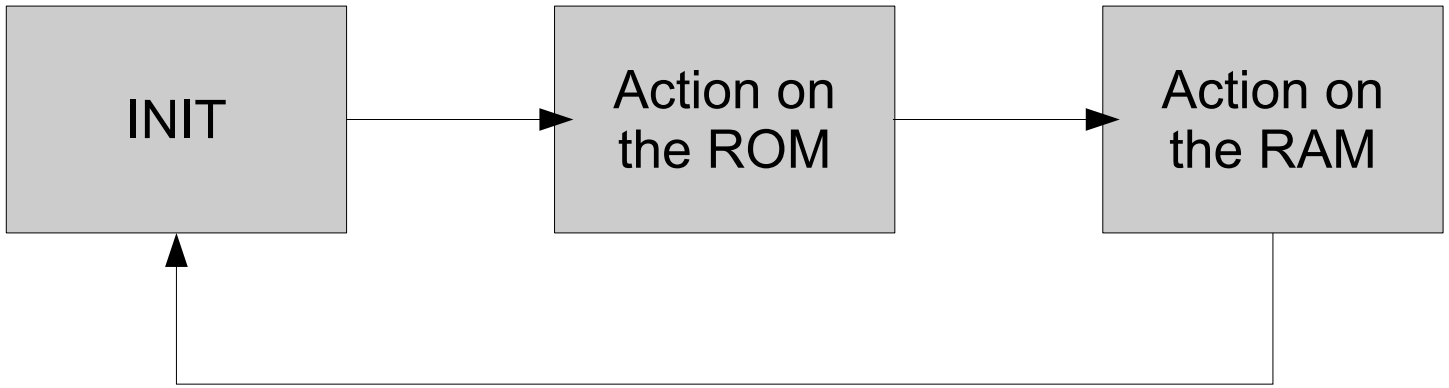
- **The 1-Wire is a communication bus protocol which can be run with several components connected to 1 single wire**



- In the 1-Wire protocol, **the line is by default set to '1'** (Pull-up resistor)
- **Only the master can drive the line to '0'** at any time
- **The slaves can only "hold" the line to '0'** for a given time **or release it.**

1-Wire : Transaction Sequence

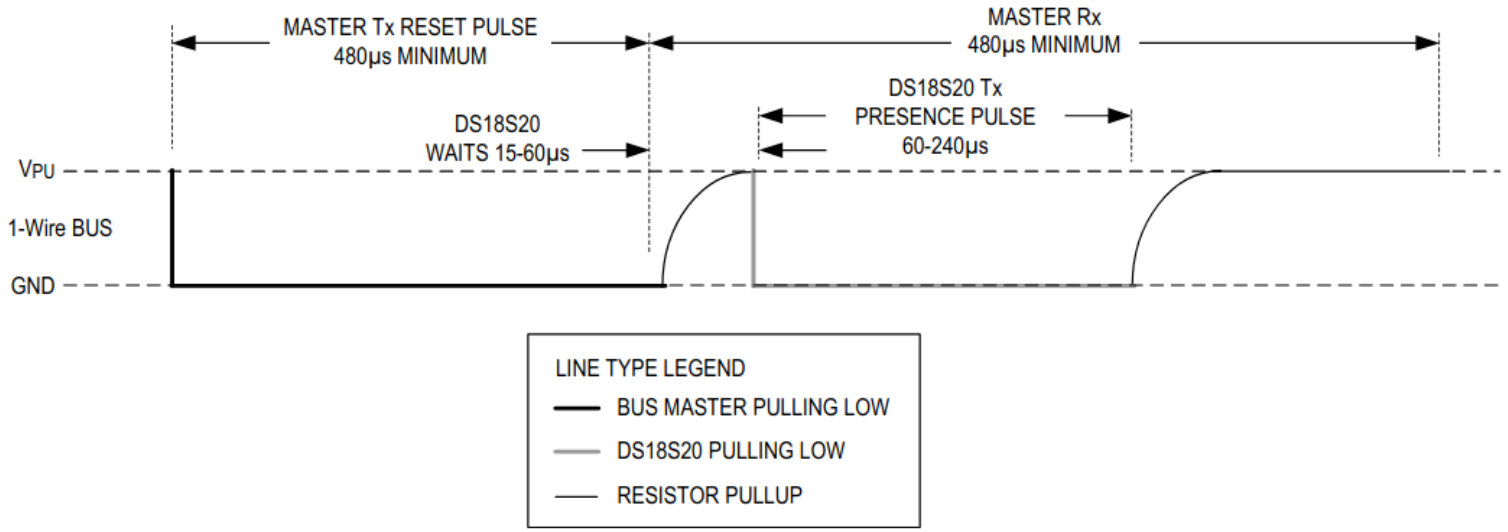
- The 1-Wire transaction sequence is based on 3 repetitive phases



- The complete protocol sequence must repeat the transaction a minimum of 2 times

Initialization

- **The initialization serves**
  - the master to ensure there are devices on the line
  - the slaves to be set in an initial mode
- **It is the simpliest sequence**
  - **The master pull the line to '0'** for 480 µs minimum then release it
  - **The slave(s)** have the next 480 µs to pull **back the line to '0'** for minimum 60 µs



Action on the ROM

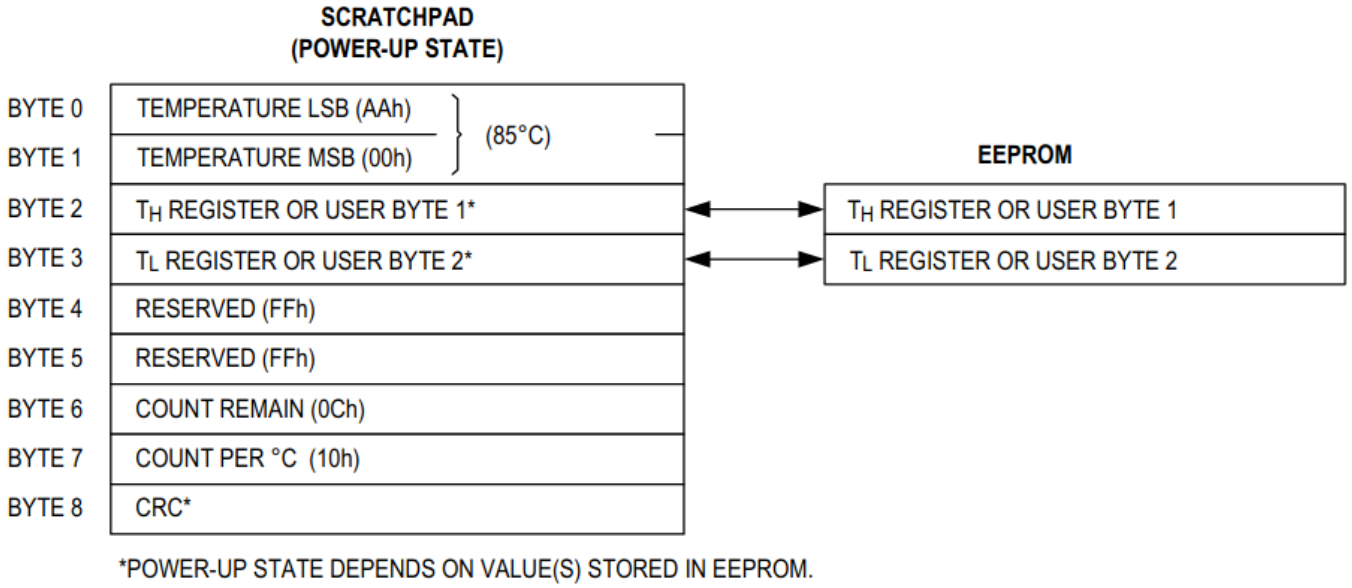
- **The action on the ROM consist for the master to send (or Write) a word of 1 Byte** just after the INIT sequence.
- **If required, it is followed by the reading of the data from the slave** (not applied here)
- For simplicity reasons we will use only the **Skip ROM** command for which the word is **"CCh"**
- Others command would have been : search ROM, Read ROM, Match ROM, and Alarm Search

Action on the RAM (Scratchpad)

- **The action on the RAM consist for the master to send (or Write) a word of 1 Byte** just after the ROM sequence.
- **It is followed by the reading of the data from the slave**
- **In the first cycle of transaction**, the master shall ask **the thermometer to acquire the temperature value**. This command is called **convertT and the associated word is "44h"**
- After this order has been emitted, the slave will do the acquisition which will take up to 800 ms. **During this time the master interrogates the slave. The slave response is '0' until the conversion has completed.**
- **In the second cycle of transaction,** the master shall ask **the slave to send you the acquired data** of the scratchpad. This command is called Read Scratchpad **and the associated word is "BEh"**
- After this order has been emitted, **the master interrogates the slave which return the 64 bits of the scratchpad starting with the LSB.**

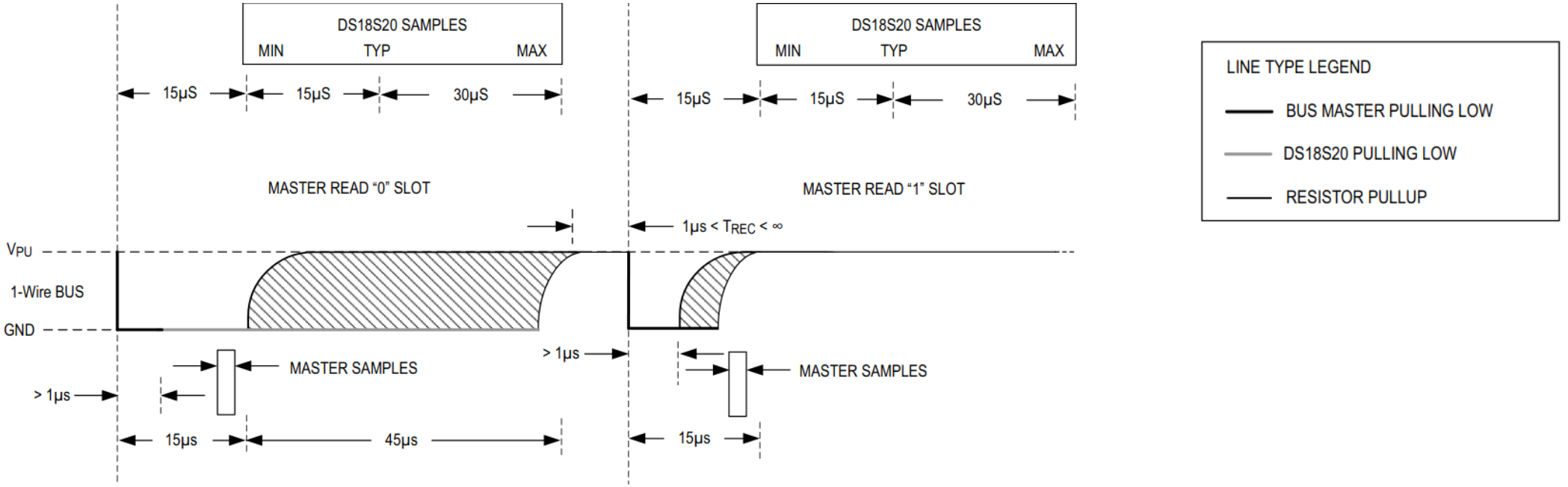
Reading the scratchpad

- During the reading sequence, **the master shall interrogate the slave for each bit of the 64.**
- **The Slave always communicate the LSB of the lowest Byte first**



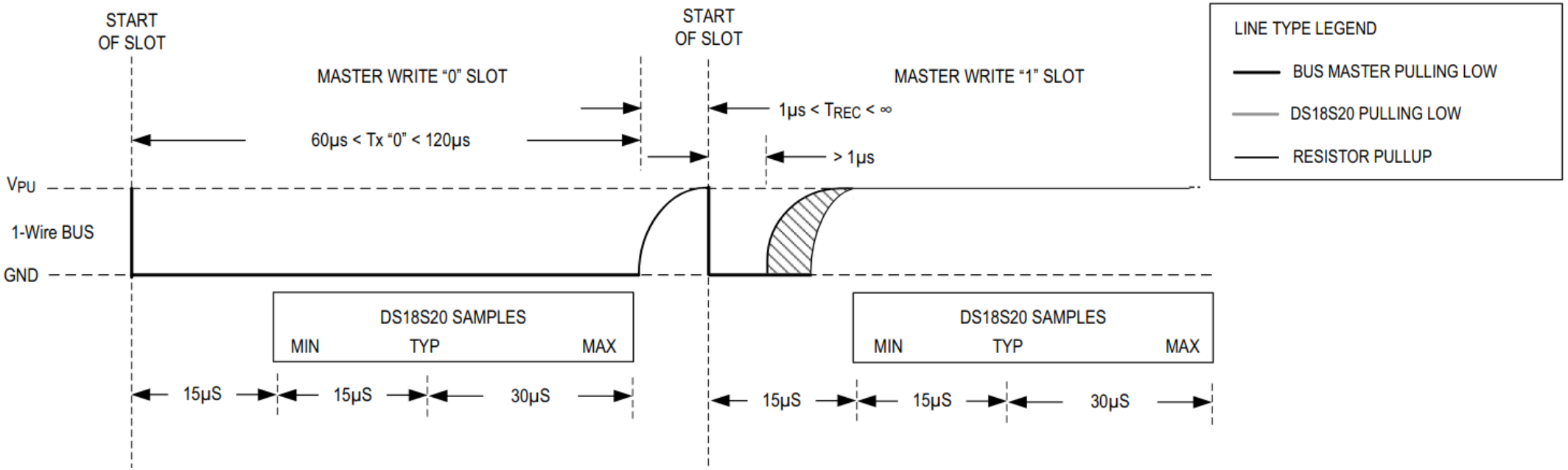
The reading sequence can be interrupted anytime by a new INIT sequence

Reading 1 bit



- **The master pull down the line for a minimum of 1 µs then release it**
- If the bit value is **'0', the slave keep the line down for a minimum of 15 µs**
- If the bit value is **'1', the slave "immediatly" release the line**
- To properly interpret the bit value, the master must interrogate the line status by the end of the 15 µs delay.

Writing 1 bit



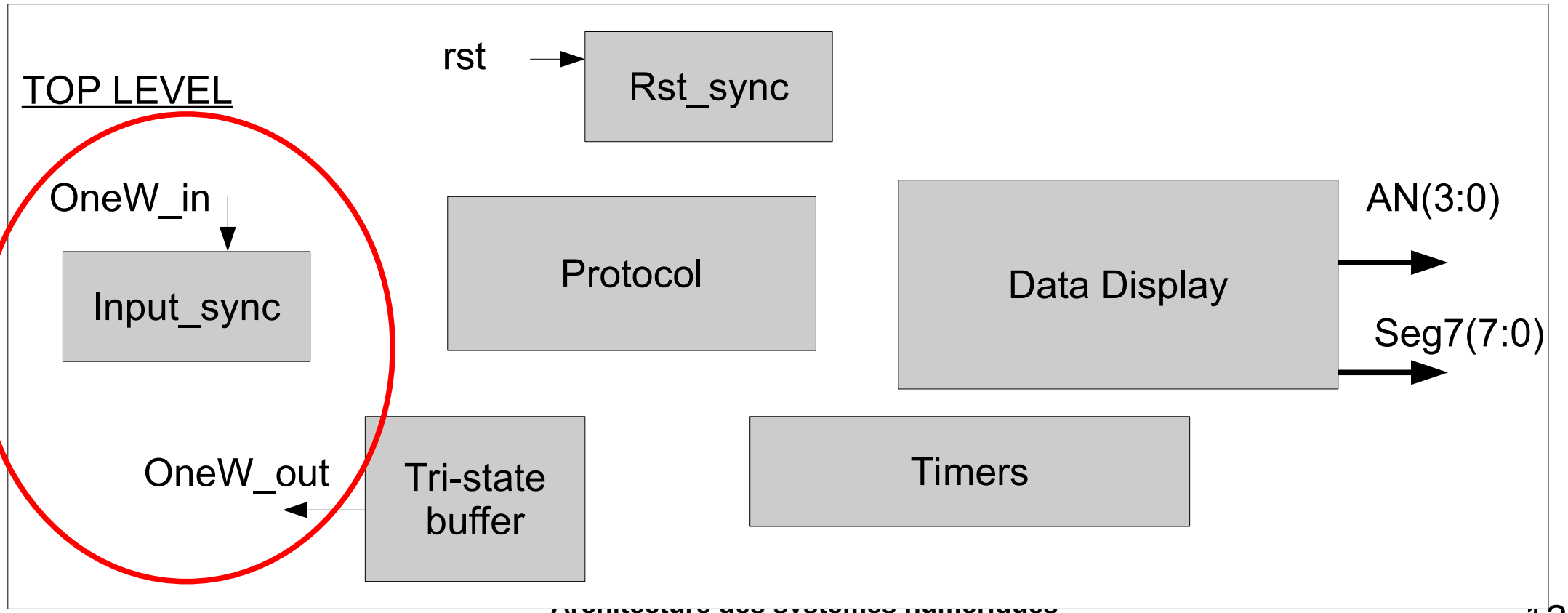
- **The master pull down the line for**
  - **a minimum of 1 µs** (and less than 15 µs) **to write '1'**
  - **a minimum of 60 µs** (and max 120 µs) **to write '0'**
  - **after writing a bit, the line is left released for minimum 1 µs**

Design requirements

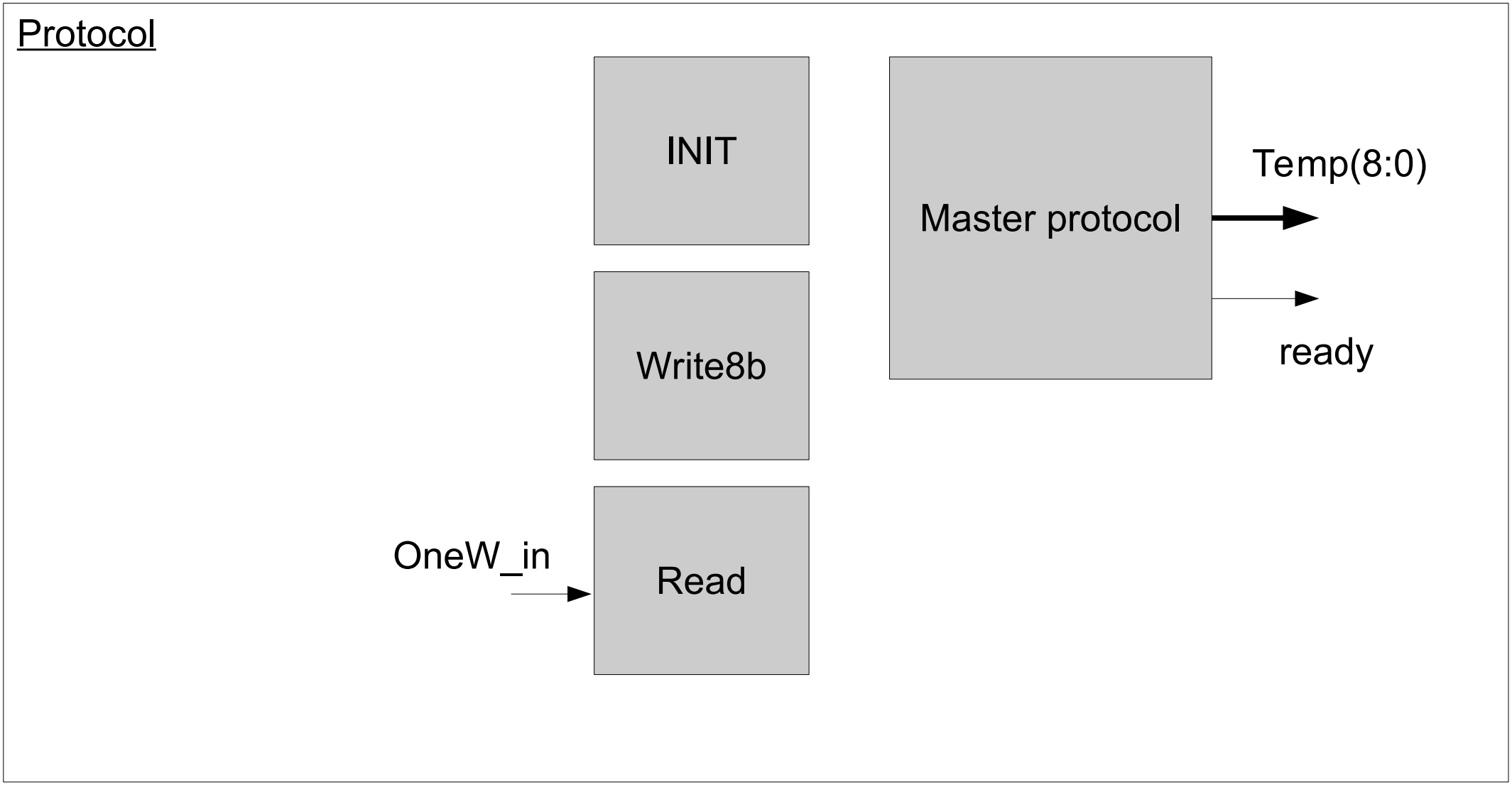
- **The system you will design must comply with the following requirements :**
  - Display of temperature in Celsius from 85 to -55 °C with a resolution of 0.5°C
  - Refreshing rate of 1 s
  - Full synchronous design
  - Most of the modules designed as FSM
  - At least one of each FSM type must be implemented (3 process Moore, 3 process Mealy, 1 process)
  - Each module shall be tested independently (simulation + implementation)

Design architecture – Top Level

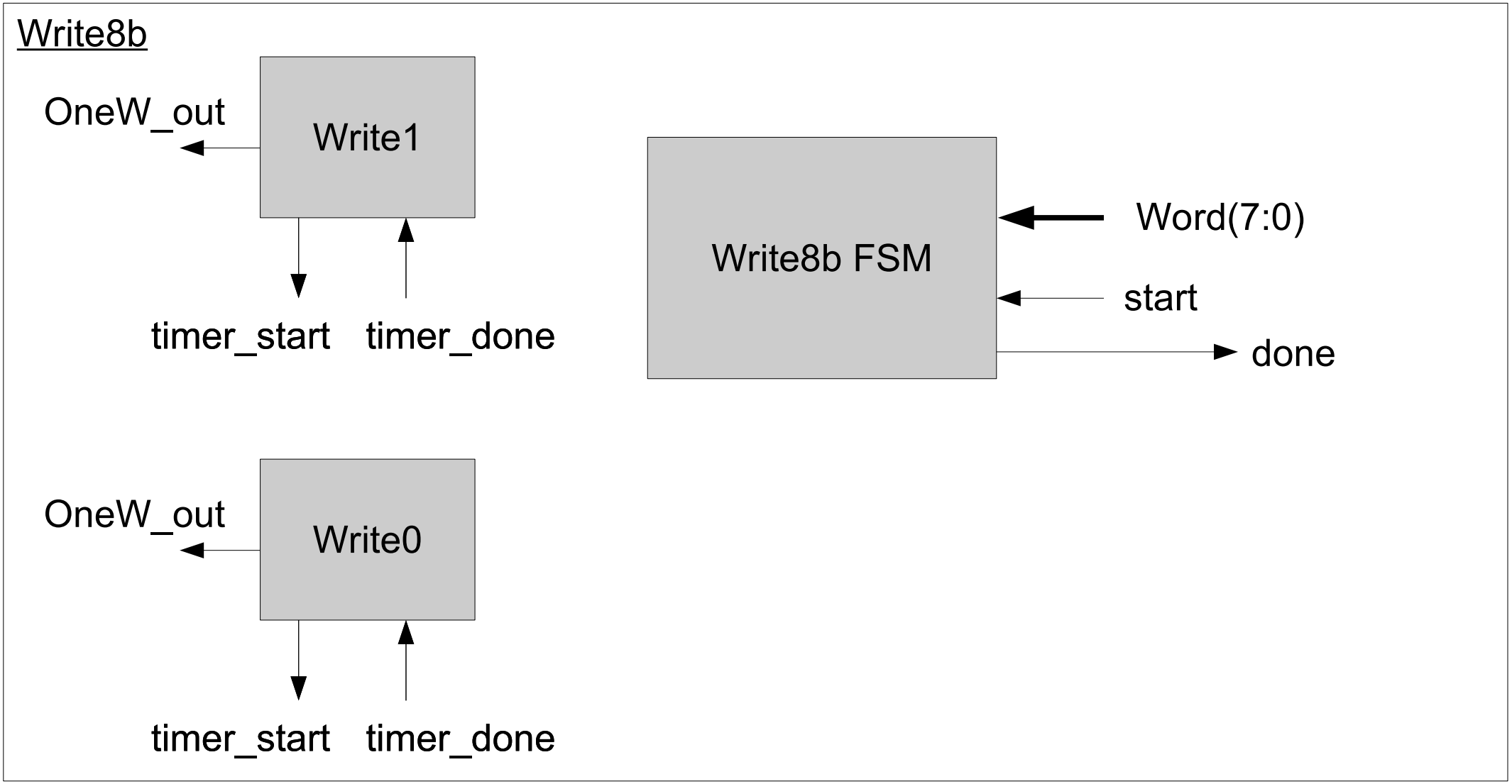
- **The global architecture below is imposed**
- **Connexion between blocs are not specified → It has to be prepared before next practical session**



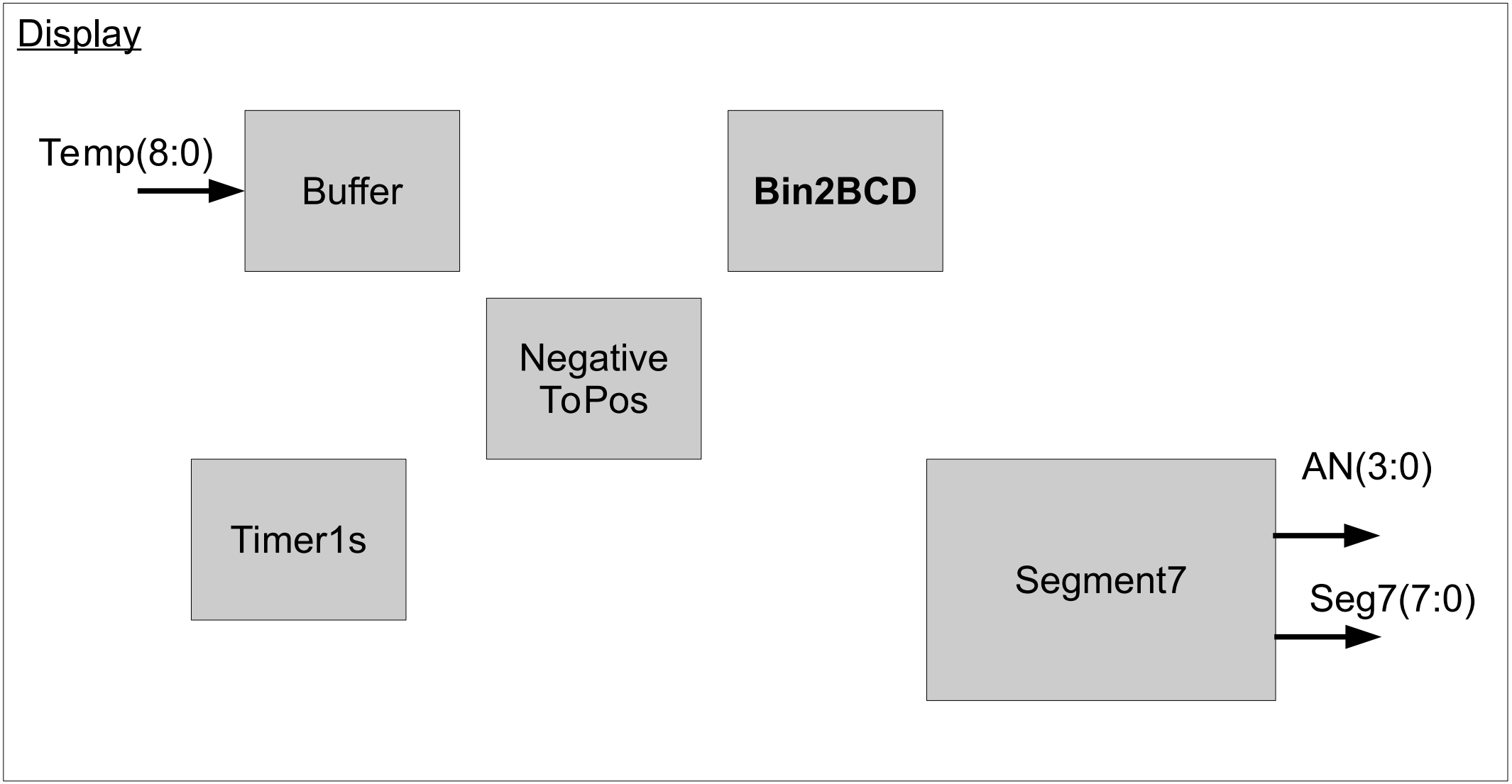
Design architecture – Protocol



Design architecture – Write8b



Design architecture – Display



Binary-to-BCD : the double dabble

- **The Double Dabble algorithm will be preferentially chosen**
- It is a recipe for which the initial binary number is shifted from LSB to MSB until the 4 bit BCD (nibble) most significant value reaches a value superior to 4.
  When the nibble is superior to 4, 3 is added to this nibble then a new shift towards MSB is done.
  The number of shifts in total shall not exceed the number of bits of the initial vector
- **This algorithm shall be implemented using a single process FSM.**

```
0000 0000 0000   11110011   Initialization
0000 0000 0001   11100110   Shift
0000 0000 0011   11001100   Shift
0000 0000 0111   10011000   Shift
0000 0000 1010   10011000   Add 3 to ONES, since it was 7
0000 0001 0101   00110000   Shift
0000 0001 1000   00110000   Add 3 to ONES, since it was 5
0000 0011 0000   01100000   Shift
0000 0110 0000   11000000   Shift
0000 1001 0000   11000000   Add 3 to TENS, since it was 6
0001 0010 0001   10000000   Shift
0010 0100 0011   00000000   Shift
   2    4    3
      BCD
```

- Example from https://en.wikipedia.org/wiki/Double_dabble