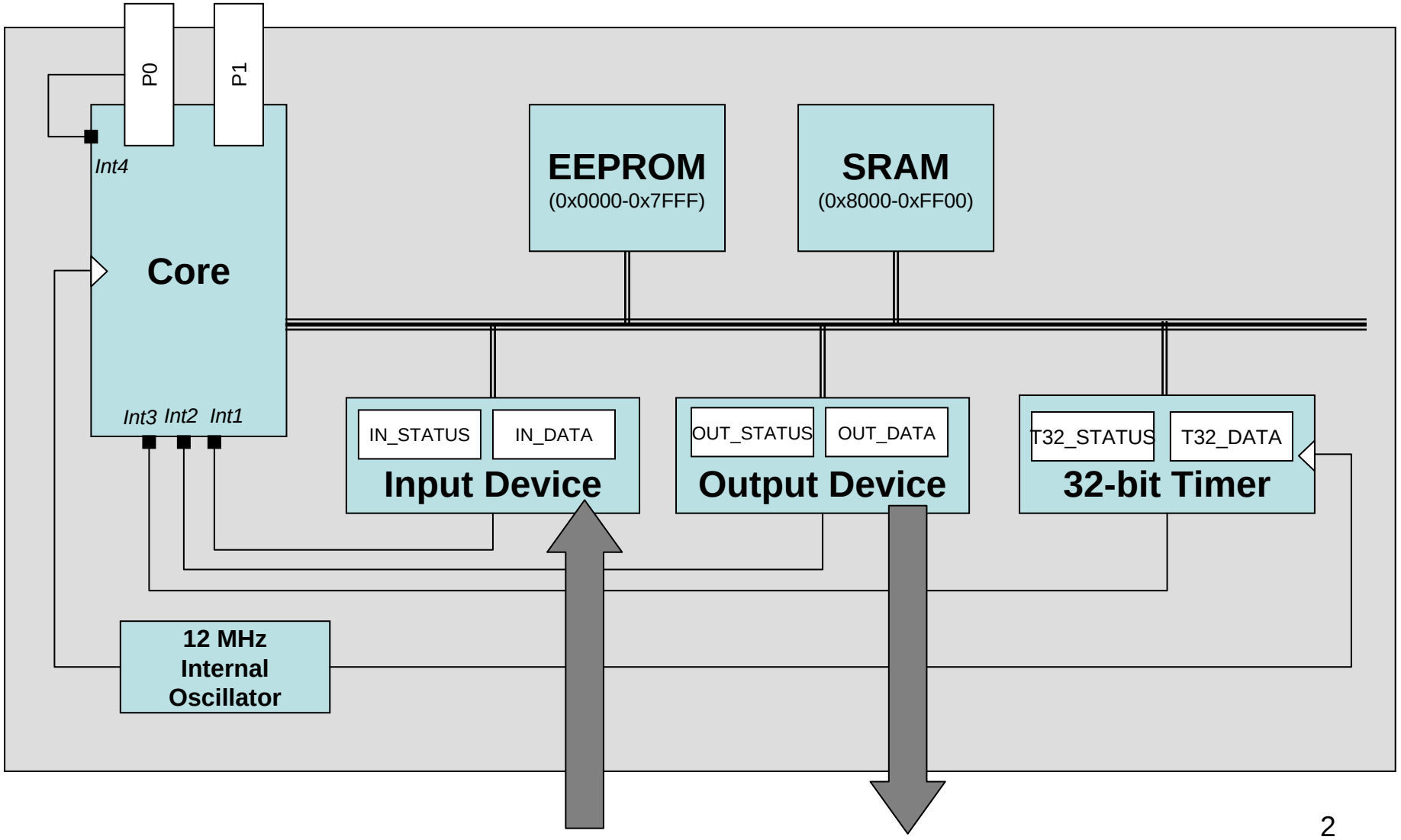
Didactical Architecture v1.1

Embedded Systems

Didactical Architecture



Didactical Architecture

- 12 MHz Internal Oscillator
- 8-bit Core
- 16-bit Address Bus, 8-bit Data Bus
- 32 KB EEPROM, 32 KB SRAM
- 2 8-bit I/O Ports
  - P0, P1
    - P0 generates an interrupt (if enabled) when written from the external world

Didactical Architecture

- 4 Fixed Interrupt Pins
  - Management
    - void enable_allint(), void disable_allint()
    - void enable_int1(), void disable_int1(), etc.
      - Default: disabled
  - ISR
    - void isr_int1(), void isr_int2(), void isr_int3(), void isr_int4()
- Memory Mapped I/O
  - 8-bit devices registers are mapped to 0xFF00-0xFFFF
    - Input Device, Output Device, 32-bit Timer

Didactical Architecture

- Input Device
  - Registers
    - IN_STATUS (status/commands)
    - IN_DATA (data)

      - IN_STATUS is normally 0x00 and it is set to 0x01 by the device itself when an external data is ready in IN_DATA
      - Who reads the data has to write a 0x00 in IN_STATUS to signal that the data has been read
        - » If a new data arrives while there is a read still to be done the new data is lost
      - The device generates an interrupt (if enabled) when a data is ready

Didactical Architecture

- Output Device
  - Registers
    - OUT_STATUS (status/commands)
    - OUT_DATA (data)
      - OUT_STATUS is set to 0x01 to ask to the device to write externally the data in OUT_DATA
        - » OUT_STATUS is set to 0x00 by the device to signal that the write has been done
      - To write a new data it is needed to wait for the completion of the previous write
        - » Changing OUT_DATA too early could lead to unpredictable results
      - The output device generates an interrupt (if enabled) when a write has been completed

Didactical Architecture

- Timer32 Device
  - Registers
    - T32_STATUS (status/commands)
    - T32_DATA (data)
      - T32_STATUS default value is 0xF0
        - » i.e. Timer NOT running
      - T32_STATUS shall be set to 0x00 to stop the timer
        - » T32_STATUS is set to 0xF0 by the device to signal that the operation has been done

Didactical Architecture

- Timer32 Device
  - Registers
    - T32_STATUS (status/commands)
    - T32_DATA (data)

      - T32_STATUS shall be set to 0x01 to set the pre-scaler (internal register T32_PRE) to the value indicated in T32_DATA
        - » Available values are: 1 (default), 2, 4, 8, 16, 32
        - » Not available values are considered as 1
        - » T32_STATUS is set to 0xF0 by the device to signal that the operation has been done
        - » Changing pre-scaler when the timer is running could lead to unpredictable results
        - » Changing T32_DATA too early could lead to unpredictable results

Didactical Architecture

- Timer32 Device
  - Registers
    - T32_STATUS (status/commands)
    - T32_DATA (data)
      - T32_STATUS shall be set to 0x02 to copy the value present in T32_DATA in the first byte (i.e., MSB) of an internal T32_COMPARE register
        - » 0x03 for the second one, 0x04 for the third one, 0x05 for the fourth one (i.e. the LSB)
        - » Changing T32_COMPARE when the timer is running could lead to unpredictable results
        - » T32_STATUS is set to 0xF0 by the device to signal that the operation has been done
        - » Changing T32_DATA too early could lead to unpredictable results

T32_COMPARE

1st BYTE (MSB)	2nd BYTE	3rd BYTE	4th BYTE (LSB)

Didactical Architecture

- Timer32 Device
  - Registers
    - T32_STATUS (status/commands)
    - T32_DATA (data)

      - T32_STATUS shall be set to 0x06 to ask to the device to generate a periodic interrupt when the internal counter reaches the value in the internal T32_COMPARE register (then the internal counter is reset to 0x00000000 immediately)
        - » T32_STATUS is set to 0xFF by the device to signal that the operation has been started (i.e. the timer is running)

Didactical Architecture

- Timer32 Device
  - Registers
    - T32_STATUS (status/commands)
    - T32_DATA (data)

      - T32_STATUS is set to 0x07 to ask to the device to generate an one-shot interrupt when the internal counter reaches the value in the internal T32_COMPARE register (then the internal counter is reset to 0x00000000 immediately)
        - » T32_STATUS is set to 0xFF by the device to signal that the operation has been started (i.e. the timer is running)
        - » T32_STATUS is set to 0xF0 by the device to signal that the operation has been concluded (i.e. the timer is not running)

Didactical Architecture

Timer 32 Overview

Command (T32_STATUS)	T32_DATA	T32_COMPARE	Prerequisite	Action	Post	Comments
0x00	D.C (Don't Care!)	D.C	None	Stop the timer	T32_STATUS=0xF0	0xF0 is the default value for T32_STATUS
0x01	*n*	D.C.	Timer not running	Set the prescaler to *n* (T32_PRE=*n*)	T32_STATUS=0xF0	If (*n* != from 1,2,4,8,16,32) *n*=1 Changing T32_DATA too early could lead to unpredictable results
0x02	*n*	D.C.	Timer not running	Set the 1st byte of T32_compare to *n*	T32_STATUS=0xF0	Changing T32_DATA too early could lead to unpredictable results
0x03	*n*	D.C.	Timer not running	Set the 2nd byte of T32_compare to *n*	T32_STATUS=0xF0	Changing T32_DATA too early could lead to unpredictable results
0x04	*n*	D.C.	Timer not running	Set the 3rd byte of T32_compare to *n*	T32_STATUS=0xF0	Changing T32_DATA too early could lead to unpredictable results
0x05	*n*	D.C.	Timer not running	Set the 4th byte of T32_compare to *n*	T32_STATUS=0xF0	Changing T32_DATA too early could lead to unpredictable results
0x06	D.C.	*n*	None	Generate a periodic interrupt when the internal counter reaches the value of T32_COMPARE	T32_STATUS=0xFF (Timer running)	After interrupt internal counter is reset to 0x00000000
0x07	D.C.	*n*	None	Generate an one-shot interrupt when the internal counter reaches the value of T32_COMPARE	T32_STATUS=0xFF After the interrupt: T32_STATUS=0xF0	After interrupt internal counter is reset to 0x00000000

Didactical Architecture

- In *didarch.h*

```
#define TRUE  1
#define FALSE 0

typedef int8 … (don’t care!)
typedef uint8 … (don’t care!)
typedef int16 … (don’t care!)
typedef uint16 … (don’t care!)
typedef float16 … (don’t care!)

typedef PORT … (don’t care!)
```

Didactical Architecture

- In *didarch.h*

```
/* Input device data register */
#define IN_DATA    … (don’t care!)
/* Input device status register */
#define IN_STATUS  … (don’t care!)

/* Output device data register */
#define OUT_DATA   … (don’t care!)
/* Output device status register */
#define OUT_STATUS … (don’t care!)

/* Timer32 data register */
#define T32_DATA   … (don’t care!)
/* Output device status register */
#define T32_STATUS … (don’t care!)
```

Didactical Architecture

- In *didarch.h*
  - I/O Primitives
    - Memory-mapped I/O

```
char peek(int8 volatile *location)
{
     return *location;
}

void poke(int8 *location, int8 newval)
{
     (*location) = newval;
}
```

Didactical Architecture

- In *didarch.h*
  - I/O Primitives
    - Port-based I/O

```
char ppeek(int8 volatile port)
{
     return port;
}

void ppoke(PORT port, int8 newval)
{
     port = newval;
}
```

Port names can be used also as int8 variables

Exercises

- #1
  - Develop a C program that writes on P1 the number of seconds elapsed from the start of the system
    - Make hypothesis on overflow management
- #2
  - Develop a C program that receives 10 values ($\in$N) from the Input Device and then sends them on P1 one at a second in crescent order
    - Try solutions with and without Input Devices interrupt
- #3
  - Develop a C program that receives 10 values ($\in$N) from the Input Device and sends their average to the Output Device
    - Try solutions with and without interrupts

Exercises

- #4
  - Develop a C program that receives a sequence of *n* ($\in$N) values ($\in$N) terminated by 0xFF from the Input Device and send their average to the Output Device
    - Make hypothesis on *n* and on overflow management
    - Try solutions with and without interrupts
- #5
  - Develop a C program that receives 10 values ($\in$N) from the Input Device, send them out on P1 with a frequency of 2 Hz in descent order, and send their average to the Output Device. Meanwhile, the program writes on P0 the number of seconds elapsed from the start of the system
    - Make hypothesis on overflow management
    - Try solutions with and without I/O Devices interrupts
- …

Improvements

- Add peripherals
- Add timers
- Add WDT and reset
- Introduce buttons and led
- Add peripherals described in VHDL
- Bit banging
- Multi-core
- …