



# Computer Hardware Engineering (IS1200)

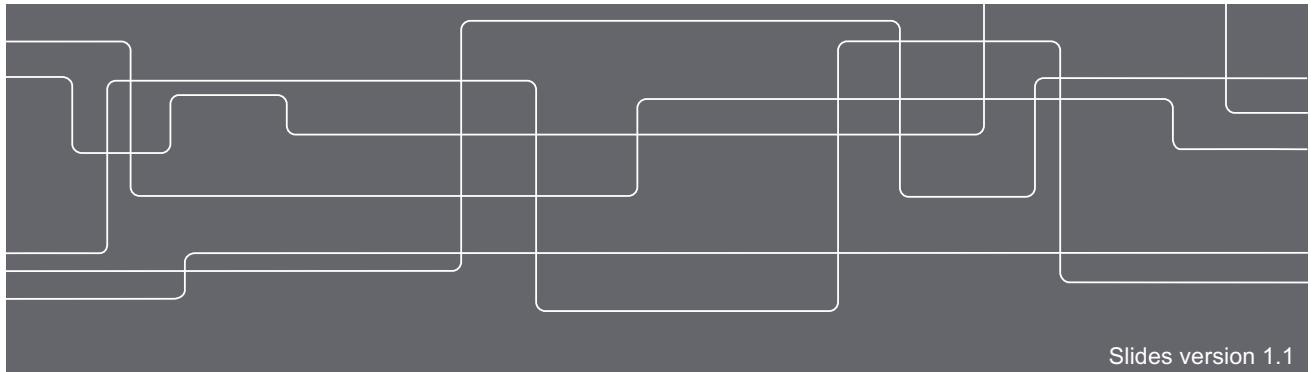
## Computer Organization and Components (IS1500)

Spring 2019

### Lecture 6: I/O Systems, part II

David Broman

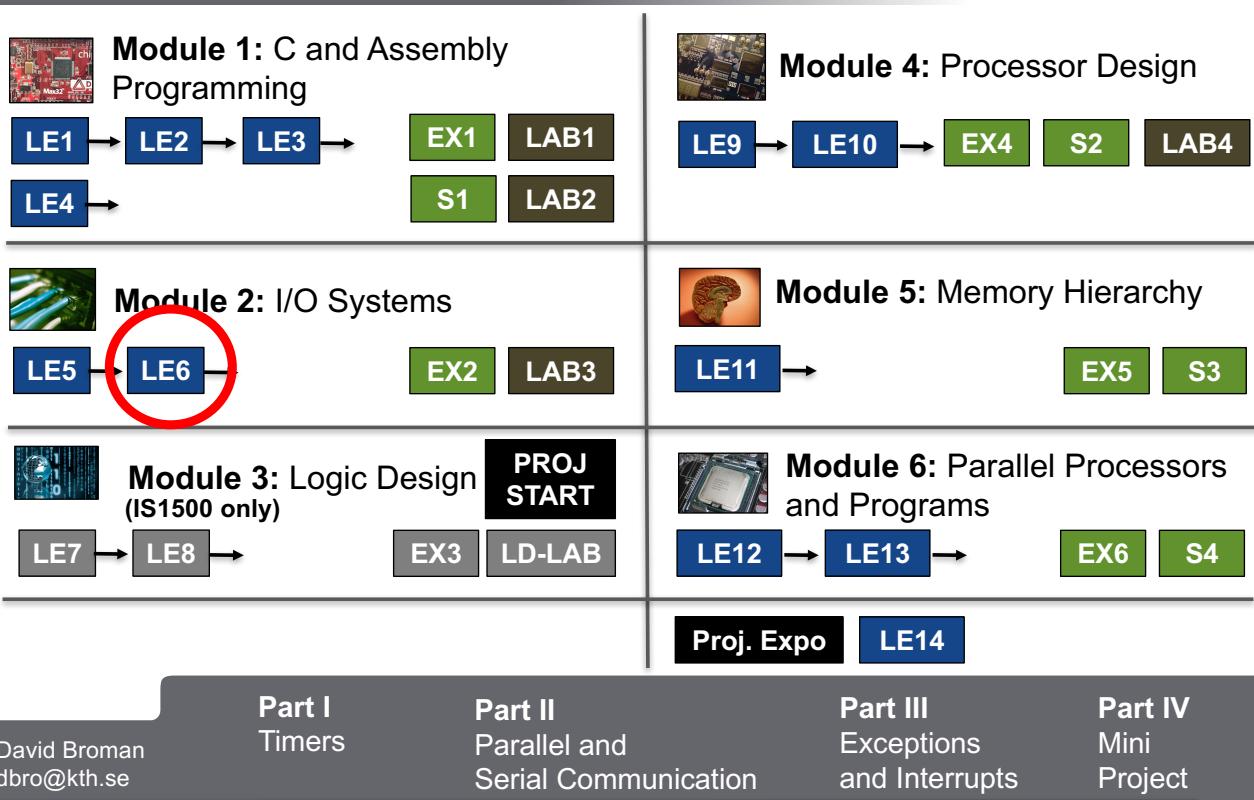
Associate Professor, KTH Royal Institute of Technology



2

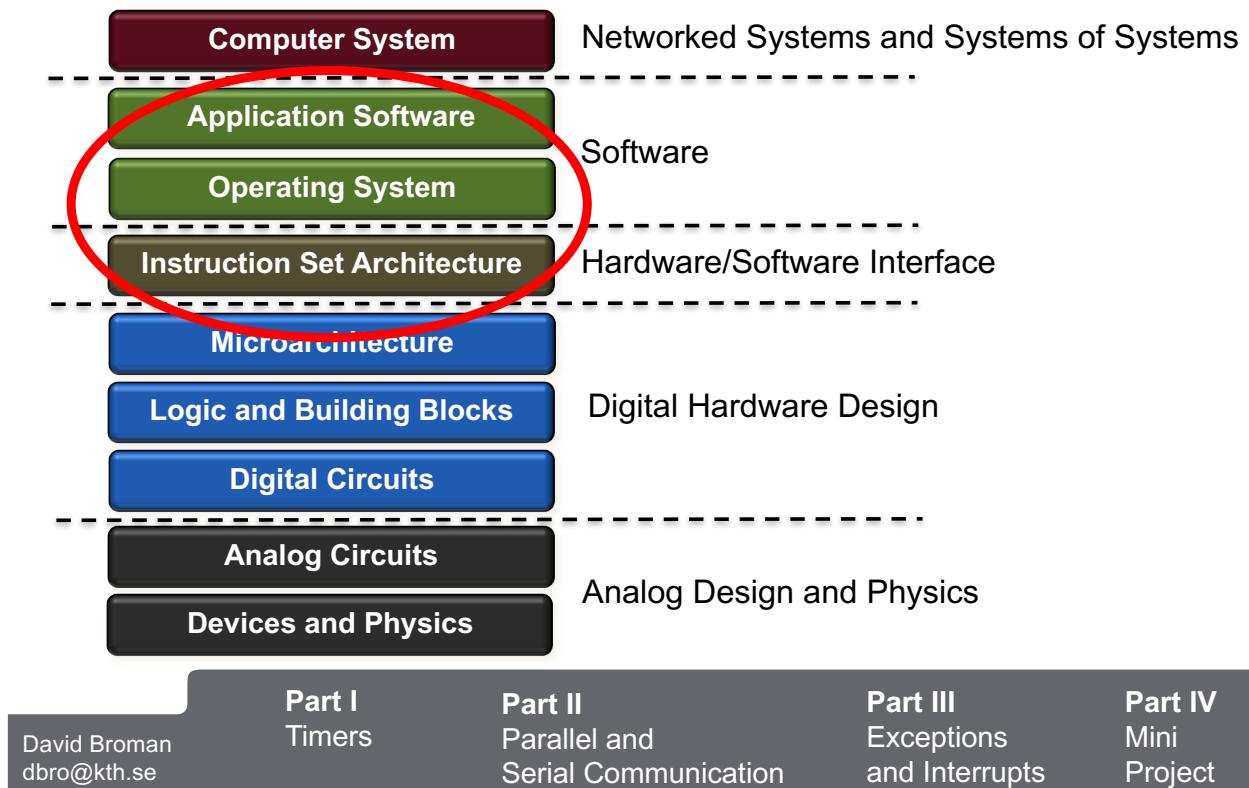


## Course Structure

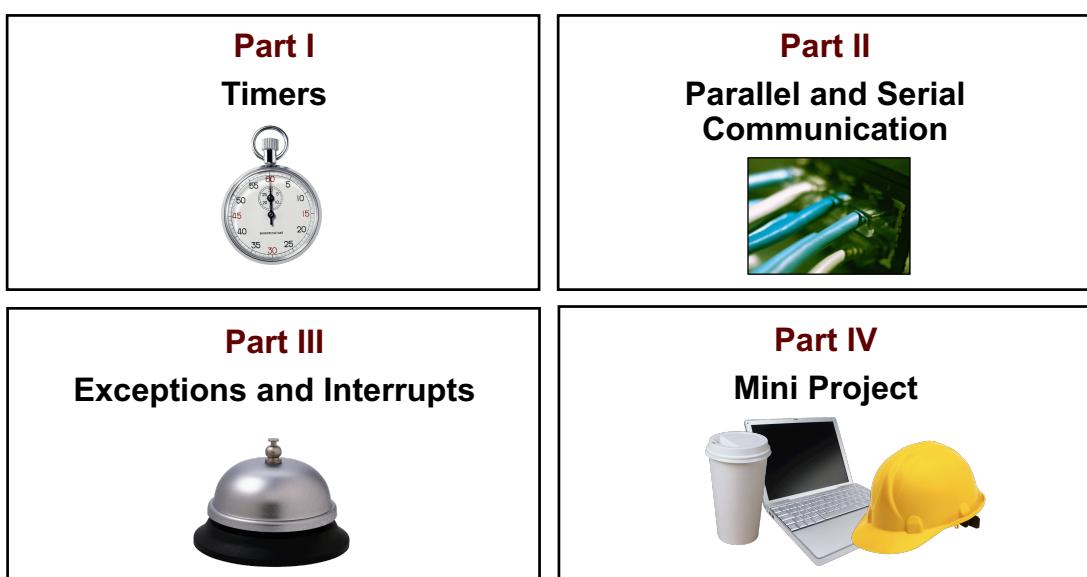




# Abstractions in Computer Systems



# Agenda



David Broman dbro@kth.se	<b>Part I</b> Timers	<b>Part II</b> Parallel and Serial Communication	<b>Part III</b> Exceptions and Interrupts	<b>Part IV</b> Mini Project
-----------------------------	-------------------------	--	---	-----------------------------------

## Part I

# Timers



Acknowledgement: The structure and several of the good examples are derived from the book "Digital Design and Computer Architecture" (2013) by D. M. Harris and S. L. Harris.

David Broman  
dbro@kth.se

Part I  
Timers

**Part II**  
Parallel and  
Serial Communication

**Part III**  
Exceptions  
and Interrupts

**Part IV**  
Mini  
Project



# Timers

**Timers** are I/O devices that are used to measure elapsed time.  
Timers can be configured in different ways, but have basically the following components.

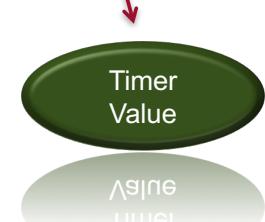
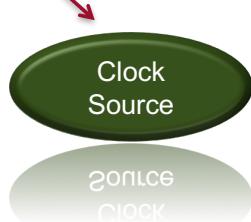


A timer is driven by a **clock source**. Example, 50Mhz external clock.

- Tells if time-out is reached.
- Stop/start counting
- Check if running.

Write registers,  
states the period

Read/write register.  
A “snap-shot” of the timer value.



David Broman  
dbro@kth.se

Part I  
Timers

**Part II**  
Parallel and  
Serial Communication

**Part III**  
Exceptions  
and Interrupts

**Part IV**  
Mini  
Project



## PIC32 Timers – Searching for information

On the ChipKIT Uno32 board, we have a **PIC32MX320F128H** processor.



How can we find this information?

See the **chipKIT Uno32 Board reference manual**. Here called [UNO32 ref].



This processor has five 16-bit timers.

How can we find this information?

See the **PIC32MX3XX/4XX Family Data Sheet**. Here called [PIC32MX ref].

The PIC32 processor has two types of timers:

- Type A timers (Timer 1):**

Can operate on an external clock

- Type B timers (Timers #2-5):**

Can be combined to form 32-bit timers.

Runs with the peripheral clock.

How can we find this information?

See the **PIC32 Family Reference Manual, Section 14. Timers**.

Here called [PIC32Family ref, Sec 14].

See the course web, page “Literature & Resources” for links.

David Broman  
dbro@kth.se

Part I  
Timers

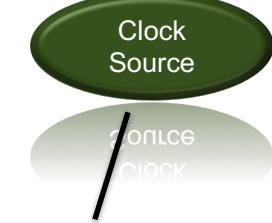
Part II  
Parallel and  
Serial Communication

Part III  
Exceptions  
and Interrupts

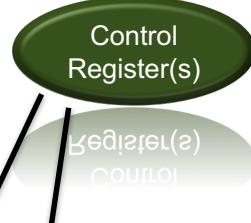
Part IV  
Mini  
Project



## PIC32 Timers

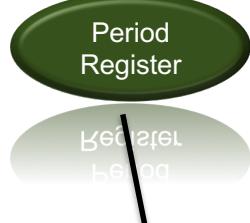


Timer 2 (that we will use) operates at 80 MHz.



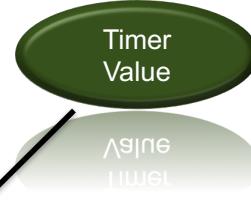
**TxCON** (x is the timer number)

- 16-bit control register.
- Configures the prescaling
- Start/stops the timer



**TMRx** (x is the timer number)

- Holds the current 16-bit timer value
- May be written to.



**PRx** (x is the timer number)

- The period register.
- TMRx counts upwards until reaching this value.

**IFS0**

- Interrupt register indicating when the counter has reached the period value.
- Note that the define in pic32mx.h is **IFS(0)** and not **IFS0**.

David Broman  
dbro@kth.se

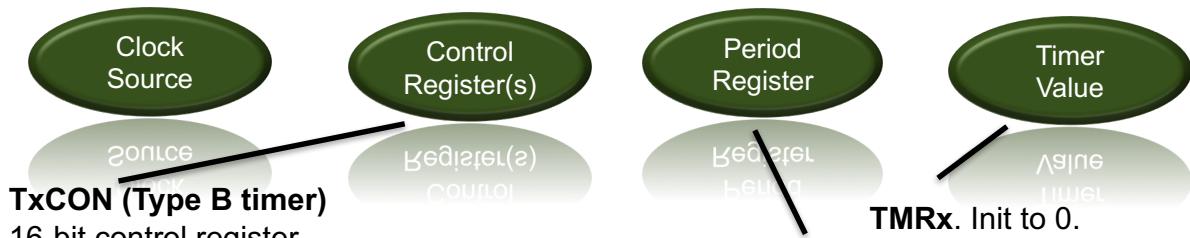
Part I  
Timers

Part II  
Parallel and  
Serial Communication

Part III  
Exceptions  
and Interrupts

Part IV  
Mini  
Project

# PIC32 Timers – Initialization (1/2)



## Exercise

What should PR2 (timer 2) be initiated to, if we want the period to be 1 second and we assume that the clock is 80 MHz?

**Solution:** Not possible for a 16-bit timer.  
 $80\ 000\ 000/256 = 312\ 500$ .  
Cannot fit in a 16-bit timer.

For details on the bit layout, see  
[PIC32Family ref, Sec 14, Page 14-9]

David Broman  
dbro@kth.se

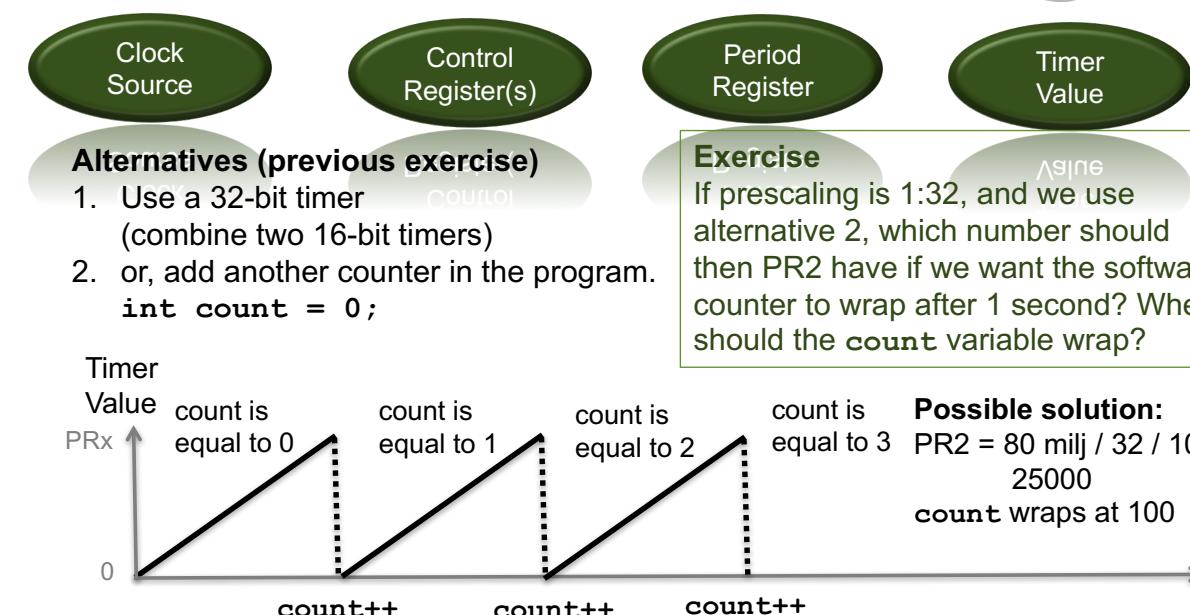
**Part I**  
Timers

**Part II**  
Parallel and  
Serial Communication

**Part III**  
Exceptions  
and Interrupts

**Part IV**  
Mini  
Project

# PIC32 Timers – Initialization (2/2)



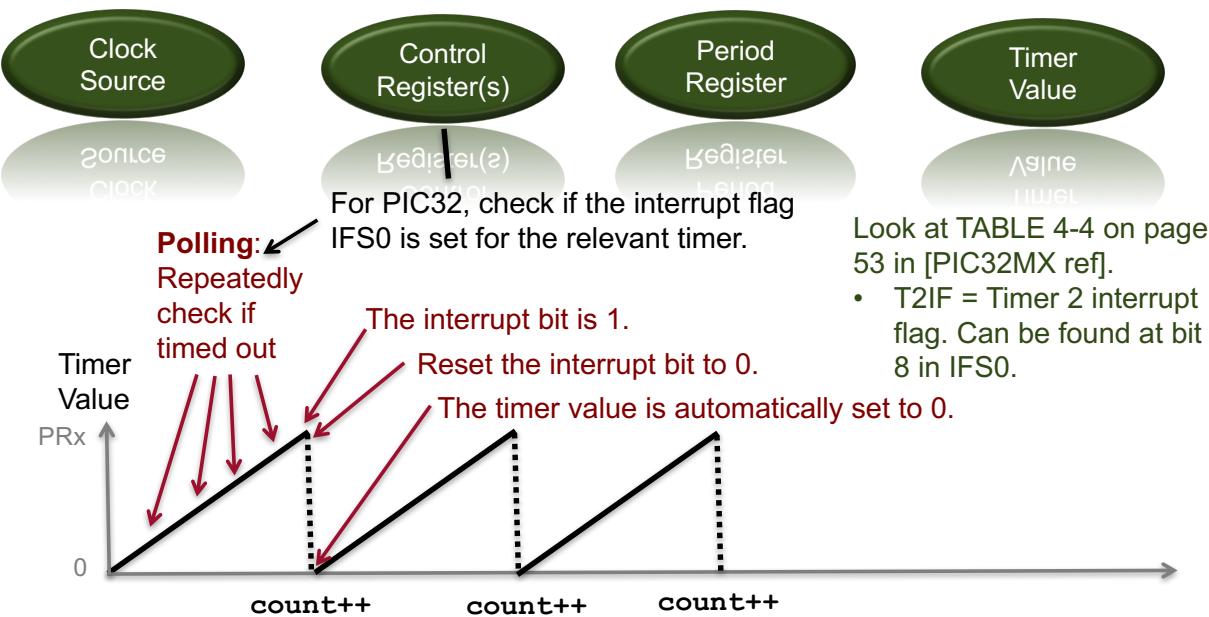
David Broman  
dbro@kth.se

**Part I**  
Timers

**Part II**  
Parallel and  
Serial Communication

**Part III**  
Exceptions  
and Interrupts

**Part IV**  
Mini  
Project



David Broman  
dbro@kth.se

Part I  
Timers

Part II  
Parallel and  
Serial Communication

Part III  
Exceptions  
and Interrupts

Part IV  
Mini  
Project



## Part II

# Parallel and Serial Communication



David Broman  
dbro@kth.se

Part I  
Timers

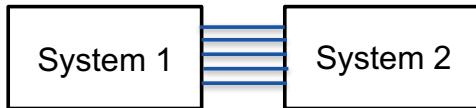
Part II  
Parallel and  
Serial Communication

Part III  
Exceptions  
and Interrupts

Part IV  
Mini  
Project

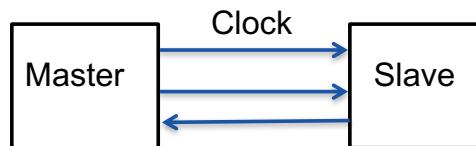


## Parallel vs. Serial Communication



### Parallel Communication

Several wires that transmits data in parallel.  
Problem: Requires many cables



### Synchronous Serial Communication

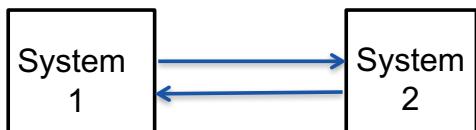
Examples:

#### **Serial Peripheral Interface (SPI)**

- Full duplex
- 4 wires, fast transfers, (10-100 MHz)

#### **I<sup>2</sup>C (Inter-integrated circuit)**

- Two lines: Serial Data Line (SDL) and Serial Clock Line (SCL)
- 7 or 10 bits addresses. (Fast mode 400kHz)



### Asynchronous Serial Communication

Example: **UART (pronounced “you-art”)**

David Broman  
dbro@kth.se

Part I  
Timers



Part II  
Parallel and  
Serial Communication

Part III  
Exceptions  
and Interrupts

Part IV  
Mini  
Project



## PC I/O Systems

- Demand for very high **performance**.
- Must be easy to **add devices**.

Consequence: complex I/O protocols



#### **Peripheral Component Interconnect (PCI)**

- Used for adding expansion cards (from 1994)
- For example, Ethernet directly on the motherboard.

#### **Universal Serial Bus (USB)**

- Taking over. High speed serial communication.
- Versions 1.0 to 3.0. Up to 5Gb/s.
- Simple for users, complex hardware and software.

#### **Other Devices**

- Ethernet (Wired networking)
- Wi-Fi (Wireless communication)
- SATA (Serial interface to hard disks)

David Broman  
dbro@kth.se

Part I  
Timers



Part II  
Parallel and  
Serial Communication

Part III  
Exceptions  
and Interrupts

Part IV  
Mini  
Project

## Part III

# Exceptions and Interrupts



David Broman  
dbro@kth.se

**Part I**  
Timers

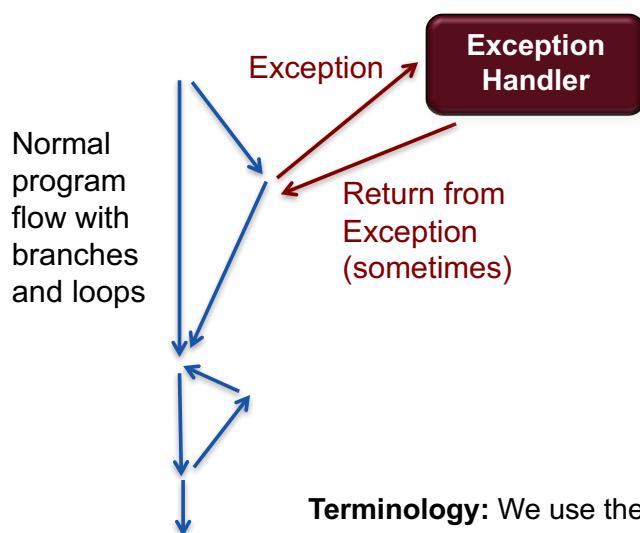
**Part II**  
Parallel and  
Serial Communication

**Part III**  
Exceptions  
and Interrupts

**Part IV**  
Mini  
Project



## Exceptions and Interrupts



### Causes of Exceptions

- **Error conditions.** For instance division by zero, unknown instruction etc.
- **Interrupts.** External interrupt, e.g., keyboard event.
- **Software Interrupts (also called traps).** For instance system calls (from user to supervisory mode).

**Terminology:** We use the MIPS terminology to differentiate between exceptions and interrupts. Some authors use the term interrupts for everything (e.g. used in x86 terminology).

David Broman  
dbro@kth.se

**Part I**  
Timers

**Part II**  
Parallel and  
Serial Communication

**Part III**  
Exceptions  
and Interrupts

**Part IV**  
Mini  
Project

# Exception Handling in MIPS (general description)

In MIPS, the exception handler is located at address 0x80000180.

## Exception handling step-by-step procedure

1. The processor stores the current PC in a register called **EPC**.
2. The processor stores the cause of the exception in register called **Cause**.
3. The processor jumps to address 0x80000180 where the exception handler is stored.
4. The exception handler saves registers on the stack.
5. The exception handler inspects the cause of the exception and handles the exception. EPC and Cause are not normal register. They are accessed using instruction **mfc0** (move from coprocessor 0).
6. The exception handler restores saved registers, copies the return program pointer EPC to \$k0, and jumps back using `jr $k0`. Alternatively, for MIPS32, use the **eret** instruction.

David Broman  
dbro@kth.se

**Part I**  
Timers

**Part II**  
Parallel and  
Serial Communication

**Part III**  
Exceptions  
and Interrupts

**Part IV**  
Mini  
Project

# Exception Vectors and the ARM ISA

In ARM, the cause of the exception is shown by jumping to different exception handlers, depending on the cause. This is called an **exception vector**.

The vector will normally contain jumps to where the actual handler is stored.

Exception	Vector address
Reset	0x0000 0000
Undefined instruction	0x0000 0004
Software Interrupt (SWI)	0x0000 0008
Prefetch abort	0x0000 000C
Data abort	0x0000 0010
IRQ (normal interrupt)	0x0000 0018
FIQ (fast interrupt)	0x0000 001C

PIC32 can be configured to use either single vector (with cause register) or multi-vector mode.

David Broman  
dbro@kth.se

**Part I**  
Timers

**Part II**  
Parallel and  
Serial Communication

**Part III**  
Exceptions  
and Interrupts

**Part IV**  
Mini  
Project



## Maskable Interrupts and Priorities

**External interrupts** may be enabled (allowed) or disabled. This is typically configured by using an interrupt flag status register.

Interrupts can be given different **priorities**. An interrupt with higher priority will **preempt** a lower priority interrupt.

For instance, a timer might have higher priority than the UART and will preempt the UART handling routine.



Non-maskable interrupts are interrupts that cannot be masked (disabled). Typically involved non-recoverable hardware errors.

David Broman  
dbro@kth.se

Part I  
Timers

Part II  
Parallel and  
Serial Communication

Part III  
Exceptions  
and Interrupts

Part IV  
Mini  
Project



## PIC32 Interrupts - Initialization

### IECx (Interrupt Enable Control)

- The initialization procedure should set the bit for the interrupt that should be enabled.

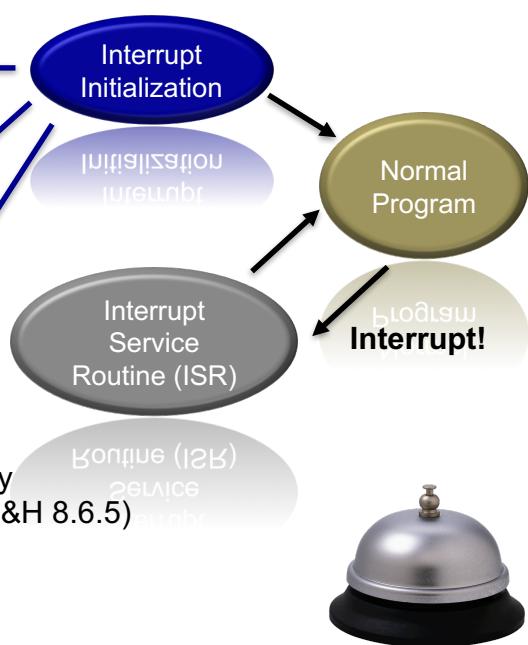
### IPCx (Interrupt Priority Control)

- Configure the priority (0-7, where 7 is highest) and the sub-priority (0-3 where 3 is highest).

### Enable Interrupts

You need to enable interrupts by executing instruction `ei` (See H&H 8.6.5)

The x means that there exist several registers, e.g. IPC0, IPC1, etc. Note that for pic32mx.h, the syntax is e.g., IPC(1).



David Broman  
dbro@kth.se

Part I  
Timers

Part II  
Parallel and  
Serial Communication

Part III  
Exceptions  
and Interrupts

Part IV  
Mini  
Project



## PIC32 Interrupts – Interrupt Execution

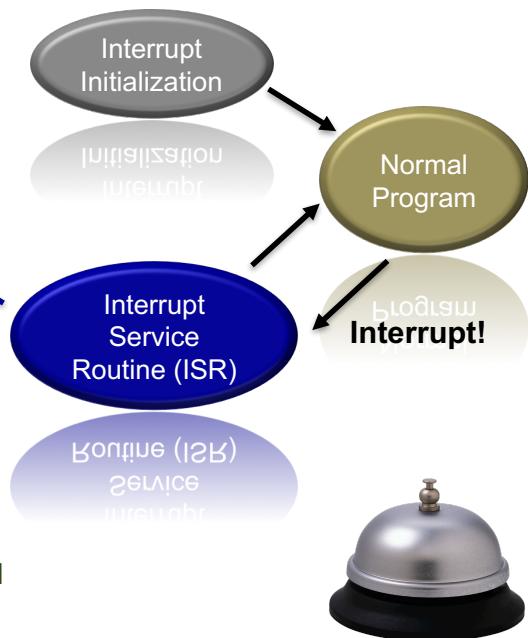
### IFSx (Interrupt Flag Status Register)

- The interrupt request sets the bit (e.g. for a timer at timeout).
- The ISR must clear the flag before returning from the ISR.

The relevant bits for IECx, IPCx, and IFSx can be found in TABLE 4-4 on page 53 in [PIC32MX ref] or on page 90 (table).

**Advice for lab 3.** Use the above and find:

- T2IF (for timer 2 interrupt flag), ii) T2IE (for timer 2 interrupt enable), iii) T2IP for priority, and iv) T2IS for sub priority. Set highest priorities.



David Broman  
dbro@kth.se

**Part I**  
Timers

**Part II**  
Parallel and  
Serial Communication

**Part III**  
Exceptions  
and Interrupts

**Part IV**  
Mini  
Project



## Part IV

### Mini Project



David Broman  
dbro@kth.se

**Part I**  
Timers

**Part II**  
Parallel and  
Serial Communication

**Part III**  
Exceptions  
and Interrupts

**Part IV**  
Mini  
Project



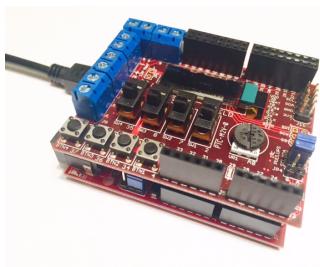
# Mini Project



## Groups

- Preferably, you work in the same group as you do with the labs (1-2 students).
- Student groups may communicate, e.g., connect hardware using UART.

## Equipment



- You must use the PIC32 platform supplied by the course.
- You may extend this platform in any way you want (Arduino shields, custom electronics, communication with computer...)
- You may use the MCB32 tool chain **or** the MPLAB X IDE, but **not** the MPIDE (a ported Arduino IDE)
- Use a version handling system, such as SVN or Git (recommended).

David Broman  
dbro@kth.se

### Part I Timers

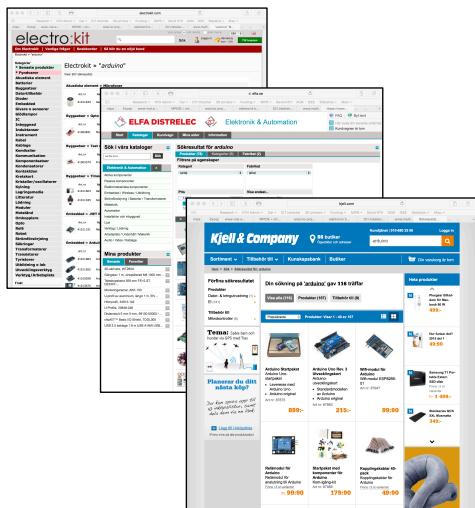
### Part II Parallel and Serial Communication

### Part III Exceptions and Interrupts

### Part IV Mini Project



# Mini Project



## Shields and Electronics

You may (optionally) extend your project with electronics. Check out the following for inspiration:  
<http://www.instructables.com/id/Arduino-Projects/>

KTH cannot supply more hardware than the boards. If you want to buy e.g. shields, check out:

- Kjell & Company: [www.kjell.se](http://www.kjell.se)
- ELFA: [www.elfa.se](http://www.elfa.se)
- Electrokit: [www.electrokit.se](http://www.electrokit.se)
- Or international delivery, e.g. [www.aliexpress.com](http://www.aliexpress.com) or [www.ebay.co.uk](http://www.ebay.co.uk)

Search for e.g., “arduino sensors” or “picKIT”

David Broman  
dbro@kth.se

### Part I Timers

### Part II Parallel and Serial Communication

### Part III Exceptions and Interrupts

### Part IV Mini Project



# Mini Project



## What kind of project?

It is very flexible and it is up to you what you want to do. Examples:

- Make a game (snake, Tetris, ping/pong, Othello etc.)
- Use serial communication or WIFI. Solve a problem.
- Use sensors such as temperature sensor (available on the I/O board), accelerometer etc.
- Do something with sound (add microphone or speaker). Play with A/D (built in) and alternative ways for analog output (see H&H section 8.6.6)
- ...

David Broman  
dbro@kth.se

**Part I**  
Timers

**Part II**  
Parallel and  
Serial Communication

**Part III**  
Exceptions  
and Interrupts

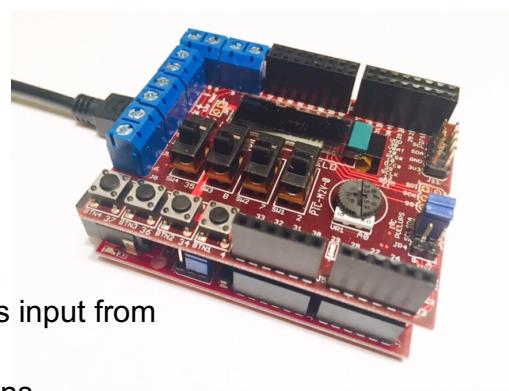
 **Part IV**  
Mini  
Project



# Mini Project – Requirements

## Basic Project

- Should use some other device/component besides the LEDs, switches, and push buttons on the board (e.g. display, temperature sensor etc.)
- Should be non-trivial and actual perform something. Should make use of timers and/or interrupts.



## Not qualified as a project

- A program with very simple logic that just reads input from the buttons and displays some text.
- An implementation with a lot of code duplications.

**In general, if you start to copy and duplicate your own code, then you are doing something that is not very good.**

David Broman  
dbro@kth.se

**Part I**  
Timers

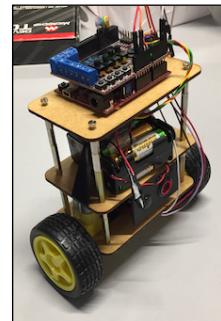
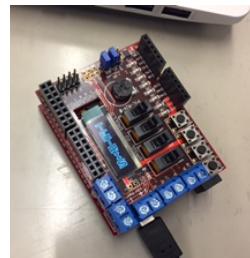
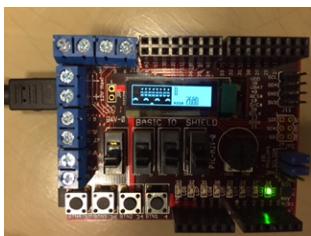
**Part II**  
Parallel and  
Serial Communication

**Part III**  
Exceptions  
and Interrupts

 **Part IV**  
Mini  
Project



## Mini Project – Requirements



### Advanced project

Requirements for getting a final grade A or B. The main categories:

1. Use external components in a complex manner by using specific protocols: SPI, I<sup>2</sup>C, or another non-trivial protocol (please check with David). UART is not counted as advanced.
2. Sophisticated graphics on the screen (see examples on the next pages).
3. Advanced interaction with sound (see examples on the next pages).

Besides the above, the program logic must be rather complex. It must clearly show that the project required some substantial development effort.

David Broman  
dbro@kth.se

**Part I**  
Timers

**Part II**  
Parallel and  
Serial Communication

**Part III**  
Exceptions  
and Interrupts

 **Part IV**  
Mini  
Project



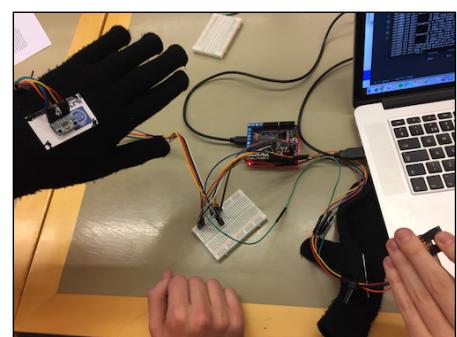
## Examples – Advanced projects

### Category 1 – Advanced protocols

Requirement: Make use of SPI, I<sup>2</sup>C or other advanced protocols.

*Example 1:* A program that is using external accelerometers, which communicate using I<sup>2</sup>C. The application shows a non trivial C program (e.g., a simple snake game).

*Example 2:* A program that is using an external display, which is communicating over SPI. The program itself is non trivial, with different modes, menus, user interactions etc.



David Broman  
dbro@kth.se

**Part I**  
Timers

**Part II**  
Parallel and  
Serial Communication

**Part III**  
Exceptions  
and Interrupts

 **Part IV**  
Mini  
Project



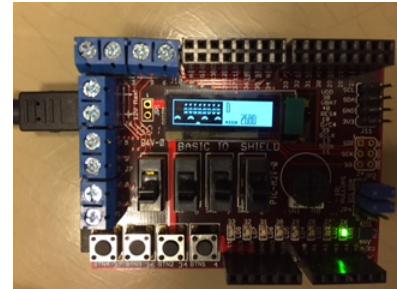
## Examples – Advanced projects

### Category 2 – Advanced graphics on the basic I/O shield's OLED display

Requirement: Pixel-by-pixel movements of objects that are larger than 2x2 pixels. The objects move smoothly 1 pixel at each frame update, both in X and Y directions.

*Example 1:* A pong game where the ball is 3x3 pixels, and moves according to the pixel-by-pixel requirement. The game supports several advanced features, such as 2 player mode, 1 player mode (simple AI for the computer player), different difficulty levels, high score, and more.

Note that simple games like Pong, Snake, Tetris etc. need to fulfil the pixel-by-pixel requirement above **and** support advanced features such as different player modes (1 and 2 player) and difficulty levels.



David Broman  
dbro@kth.se

**Part I**  
Timers

**Part II**  
Parallel and  
Serial Communication

**Part III**  
Exceptions  
and Interrupts

**Part IV**  
Mini  
Project



## Examples – Advanced projects

### Category 3 – Advanced interaction with sound

*Example 1:* A music player that reads sampled music files (e.g. wav files) from the flash and outputs the sound to the speaker. The user can select songs, move forward and backwards within the song, etc.

*Example 2:* Reading MIDI signals directly on the ChipKIT, and then generate sounds, including polyphony (e.g., at least 8 notes can be played simultaneously). The user can select different sounds, and instruments, adjust volume etc. The music is output to a real speaker.



David Broman  
dbro@kth.se

**Part I**  
Timers

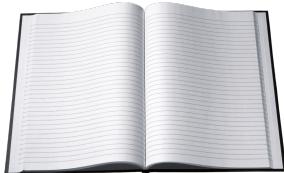
**Part II**  
Parallel and  
Serial Communication

**Part III**  
Exceptions  
and Interrupts

**Part IV**  
Mini  
Project



## **Extended Abstract (1 page) - Contents**



- **Title of the project**  
The title must clearly state what the project is about.
  - **Objective and Requirements**  
These paragraphs should answer the question: What should your project do and which are the main requirements?
  - **Solution**  
The solution paragraph should state: How did you design and implement your solution?
  - **Verification**  
The verification paragraph should answer: How did you test or verify that your solution is correct?
  - **Contributions**  
This section should explain who did what of the two project members.
  - **Reflections**  
The last paragraph should include a few lines of reflections about the project.

Part I	Part II	Part III	Part IV
Timers David Broman dbro@kth.se	Parallel and Serial Communication	Exceptions and Interrupts	Mini Project



## Extended Abstract - Submission

- **A DRAFT** of the extended abstract is submitted Canvas.
  - The draft should include 1. the title, 2. the objective and requirements, 3. the solution (preliminary idea of how it will be done), and 4. verification (preliminary idea of how the verification is done).
  - In the draft: **State if you aim for an advanced project!**

The screenshot shows a Moodle course page. At the top, there's a navigation bar with links like 'Doktor', 'Mentor', 'Courses', 'Programme', 'Groups', 'Services', and 'Logout'. Below the navigation is the course title 'KUNGLIGA TEKNIKA HÖGSKOLAN' in large blue letters. Underneath the title is a search bar with placeholder text 'Search for content...' and a 'Search' button. The main content area has a yellow header bar with the text 'KTH COURSE INFO / COMPUTER ORGANIZATION AND COMPONENTS / ONE PROJECT' and a 'View details' link. The main content starts with a section titled 'Mini Project'. It contains several sections: 'Description', 'Objectives', 'Assessment', 'Project tasks', 'Project phases', 'Prerequisites', 'Literature and Resources', 'Learning outcomes', 'Lectures', 'Assignments', 'Quizzes', 'Late', and 'Info for Teachers (NOT FOR STUDENTS)'. Below these are sections for 'Software for Lab', 'Links', 'File Project', 'Previous Course', 'Current Feedback', 'Older Feedback', 'MOTOR feedback', 'OLD 2014 Edition', 'Comments', and 'Feedback'. A sidebar on the left lists 'COMPUTER ORGANIZATION AND COMPONENTS' and 'My subjects'.

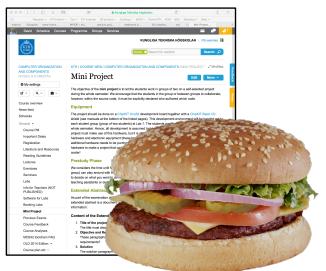
Part I	Part II	Part III	Part IV
Timers David Broman dbro@kth.se	Parallel and Serial Communication	Exceptions and Interrupts	Mini Project



## Project Expo



- **Everyone shows their projects!**
- **Nominations and Awards** for best projects.



### Getting Help

- Post questions on Canvas discussion forum!
- Come to the lunch office hours.

David Broman  
dbro@kth.se

**Part I**  
Timers

**Part II**  
Parallel and  
Serial Communication

**Part III**  
Exceptions  
and Interrupts

 **Part IV**  
Mini  
Project



**Yeah, you may leave this room  
soon...**

(please do not fumble with the bags)



David Broman  
dbro@kth.se

**Part I**  
Timers

**Part II**  
Parallel and  
Serial Communication

**Part III**  
Exceptions  
and Interrupts

 **Part IV**  
Mini  
Project



# Reading Guidelines



## Module 2 (I/O Systems)

H&H Chapters 8.5-8.7

For the labs, focus on 8.6.2-8.6.5  
(GPIO, Timers, and Interrupts)

Also, look carefully at the solutions of exercise 2.

### Reading Guidelines

See the course webpage  
for more information.

David Broman  
dbro@kth.se

**Part I**  
Timers

**Part II**  
Parallel and  
Serial Communication

**Part III**  
Exceptions  
and Interrupts

**Part IV**  
Mini  
Project



# Summary

### Some key take away points:

- **A Timer** is used for keeping track of time and timing.
- **Exceptions** cause the processor to switch context and jump to an exception handling routine.
- **Interrupts** are special kinds of exceptions that are typically triggered by an external hardware event.
- **Good luck with the project!**



Thanks for listening!

David Broman  
dbro@kth.se

**Part I**  
Timers

**Part II**  
Parallel and  
Serial Communication

**Part III**  
Exceptions  
and Interrupts

**Part IV**  
Mini  
Project