CSD table of content and site map

PDF text Org

An introduction to digital systems using VHDL and to microcontroller applications using C

CSD content discussed in lectures and labs and applied for designing projects. Similar materials and slides can be found in book chapters and webs all over the Internet. NOTE: be aware that materials that are not from this CSD course will contain different symbols, naming conventions and writing styles, and thus, we will rename and adapt them in order to be used in our CSD subject as study materials.

1.4. Analysis concept map: methods for simple combinational circuits *Circuit_C*

1.4.1. Method I: Proteus simulation (virtual laboratory) for truth tables and deducing logic circuits [Lab1_1]

1.4.1.1. SPICE algorithms

1.4.1.1.1. Proteus ISIS from Labcenter Electronics (EETAC licence available) (intended for all CSD projects)

1.4.1.1.2. Optional. Multisim (from National Instruments, UPC license available)

1.4.1.2. Optional. Digital simulators

1.4.1.2.1. Hades: interactive simulation framework

1.4.2.1.2. Deeds: Digital electronics education and design suite

1.4.1.3. Optional. Printed circuit board (PCB) design

1.4.1.3.1. Ultiboard (from National Instruments, UPC licence available)

1.4.1.3.2. Eagle, KiCAD, Fritzing

1.4.2. Method II: WolframAlpha engine for calculating truth tables and deducing logic circuits [Lab1_1]

1.4.3. Method III: Pen and paper analysis using Boolean algebra [L1.3]

1.4.3.1. Properties of boolean functions

1.4.3.2. Duality principle, De Morgan's laws

1.4.3.3. Product of sums (PoS)

1.4.3.4. Sum of products (SoP)

1.4.3.5. Any type of equation or non-standard forms

1.4.4. Method IV: Analysis using synthesis and simulation EDA tools [Lab1_2]

1.5. Design flow for inventing combinational circuits using VHDL [L1.4]

1.5.1. Specifications, design concept map

1.5.1.1. Symbol or entity

1.5.1.2. Truth table

1.5.1.3. Example timing diagram

1.5.2. Planning circuits

1.5.2.1. **Plan A**: structural/equations single-file [Lab2] *Circuit_C*

1.5.2.1.1. Canonical equations: sum of minterms or product of maxterms

1.5.2.1.2. Minimised equations: SoP or PoS

- Espresso heuristic algorithm: minilog.exe

- Truth table translated to Minilog text input format (.tbl)

- Minilog minimisation results: SoP or PoS logic equations and equation converter

- Don't-care inputs ("x" or "-")

- Incomplete functions: don't cares outputs [L2.5]

- Logic Friday and Karnaugh maps (not covered)

1.5.2.1.3. Only-NOR gates [L1.5]

1.5.2.1.4. Only-NAND gates [L1.5]

1.5.2.1.5. Any kind of nonstandard equation

1.5.2.2. **Plan B**: behavioural/algorithm single-file description [Lab2]

1.5.2.2.1. Direct translation of the truth table

1.5.2.2.2. Interpreting the truth table as a flowchart

1.5.2.2.3. Solving combinational circuits using C language and microcontrollers (covered in Chapter 3)

1.5.2.3. Plan C1: hierarchical designs (used only in the FSM structure in Chapter 2)

1.5.2.4. **Plan C2**: hierarchical design multiple-file using components and signals [Lab3]

1.5.2.4.1. Circuit expansion using components of the same kind

1.5.2.4.2. The method of decoders (MoD) [L3.3]

1.5.2.4.3. The method of multiplexers (MoM) [L3.3]

1.5.2.4.4. The method of ROM memory cells or RAM lookup tables (covered in Chapter 2)

1.5.3. Development [Lab2]

1.5.3.1. VHDL translation. Project location.

1.5.3.2. Project synthesis for a target PLD.

1.5.3.3. Target chip resource usage (logic elements)

1.5.3.4. RTL view schematic

1.5.3.5. Technology schematic

1.5.4. Test and verification (ideal/functional) [Lab2]

1.5.4.1. Test-bench fixture schematic and test-bench VHDL file

1.5.4.2. Stimulus process

1.5.4.3. Functional simulation and wave results discussion

1.5.5. Test and verification (technology) [L4.3] [Lab4]

1.5.5.1. Gate-level simulation: propagation delay measurement

1.5.5.2. Timing analyser spreadsheet: worst-case scenario, longest propagation delay

1.5.5.3. Calculating circuit's maximum speed

1.5.6. Project report and presentation [2] (intended for all CSD projects)

1.5.6.1. Handwritten sketches

1.5.6.2. English

1.5.6.3. Presentation slides

1.5.6.4. Correction grid and self-assessment

1.5.7. (optional) Prototyping and laboratory measurements

1.5.7.1. CPLD/FPGA laboratory training boards

1.5.7.1.1. Basic boards

1.5.7.1.2. Advanced boards

1.5.7.2. Laboratory instrumentation

1.5.7.2.1. Logic analyser

1.5.7.2.2. Pulse generator

1.6. Standard logic gates chips and circuits [L1.6]

1.6.1. Logic families

1.6.1.1. TTL, LS-TTL, HC, HCT, etc. (internal gate design not covered)

1.6.1.2. CMOS. How do NMOS/PMOS transistors work as ideal electronic switches?

1.6.2. Standard chip references for classic logic families

1.6.3. Electrical characteristics of logic gates

1.6.3.1. Voltage levels. What range is interpreted as '0' and as '1'?

1.6.3.2. Noise margin high (NMH), noise margin low (NML), logic family compatibility

1.6.3.3. Current and power consumption

1.6.3.4. Digital circuit propagation time, worst-case scenario: longest propagation delay [Lab4] [L4.3]

1.6.3.5. How fast is a digital circuit? [L4.3]

1.6.4. Input push-buttons and switches

1.6.4.1. Active-high button circuit

1.6.4.2. Active-low button circuit

1.6.5. Driving LED [L2.4]

1.6.5.1. LED biasing characteristics

1.6.5.2. Active-low and active-high outputs

1.6.5.3. Limiting resistor for worst-case scenario

1.6.6. Driving 7-segment displays [L2.4]

1.6.6.1. Common anode digits

1.6.6.2. Common cathode digits

1.6.6.3. Multiplexed operation

1.7. Standard combinational logic circuits [L2.1]

1.7.1. Concept map

1.7.1.1. Truth table, symbol, timing diagram

1.7.1.2. Chip expansion, enable input

1.7.1.3. Examples of commercial chips

1.7.1.4. VHDL design plan

1.7.2. Multiplexer or data selector

1.7.2.1. *MUX_2*, *MUX_4*, *MUX_8* [Lab2]

1.7.2.2. Design examples

1.7.2.3. MUX expansion circuits (plan C2)

1.7.2.4. Commercial chips

1.7.3. De-multiplexer or data distributor [L2.2]

1.7.3.1. *DeMUX_4*, *DeMUX_8*

1.7.3.2. Design examples

1.7.3.3. DeMUX expansion circuits (plan C2)

1.7.3.4. Commercial chips

1.7.4. Binary decoder [L2.3]

1.7.4.1. *Dec_3_8*, *Dec_4_16*

1.7.4.2. Design examples

1.7.4.3. Decoder expansion circuits (plan C2)

1.7.4.4. Commercial chips

1.7.5. Binary encoders (priority high) [L2.3]

1.7.5.1. *Enc_8_3*, *Enc_10_4*

1.7.5.2. Design examples

1.7.5.3. Encoder expansion circuits (plan C2)

1.7.5.4. Commercial chips

1.7.6. Hexadecimal to 7-segment decoder [L2.4]

1.7.6.1. Basic decoder

1.7.6.2. Control signals: blanking, ripple blanking, lamp test *Hex_7seg_decoder*

1.7.6.3. Commercial chips

1.7.6.4. Design examples

1.8. Binary codes and code converters [L3.1]

1.8.1. Radix-2 (base 2) and radix-16 (hexadecimal) number systems

1.8.2. Gray code

1.8.3. BCD (binary-coded decimal)

1.8.4. One-hot and one-cold

1.8.5. Johnson

1.8.6. Keyboard symbols: ASCII, etc.

1.8.7. Code converter circuits

1.8.7.1. Bin_BCD_converter_6bit

1.8.7.2. Gray_Bin_converter_4bit

1.9. Binary (radix-2) arithmetic circuits

1.9.1. Addition

1.9.1.1. 1-bit full adder block: *Adder_1bit* [L3.1]

1.9.1.2. *n*-bit adders [Lab 4] [L3.2]

1.9.1.2.1. Ripple-carry adder: *Adder_4bit*, Adder_8bit

1.9.1.2.2. Carry-lookahead adder: *Adder_4bit*

1.9.1.3. Commercial standard arithmetic chips in classic technologies

1.9.2. Comparator

1.9.2.1. Expandable 1-bit comparator (*Comp_1bit*) [L3.1]

1.9.2.2. *n*-bit comparator (*Comp_4bit*, *Comp_8bit*, etc.)

1.9.2.3. 24-bit tree comparator (*Comp_24bit*)

1.9.2.3. 24-bit tree comparator (*Comp_24bit*)

1.9.2.4. Commercial standard arithmetic chips in classic technologies

1.9.3. Multiplier

1.9.3.1. 1-bit multiplier cell, *Mult_1bit* [L3.1]

1.9.3.2. *Mult_8bit*

1.9.4. Counting the number of ones in a vector input (for example counting cars in parking slots) [L3.2]

1.9.4.1. *Ones_counter_4bit*

1.9.4.2. *Ones_counter_8bit*

1.9.5. Parity generators and checkers (*Parity_gen_8bit*)

1.10. Integer arithmetic [L4.1]

1.10.1. Integer number two's complement (2C) representation

1.10.2. Adder-subtractor (*Int_Add_Subt_8bit*, etc.)

1.10.3. Multiplier (*Int_Mult_8bit*)

1.10.4. Comparator (*Int_Comp_8_bit*, etc.)

1.11. Arithmetic Logic Unit (*ALU_4bit*) [L4.2]

1.11.1. Conceptual architecture

1.11.2. Bitwise logic operations (AND, OR, etc.)

1.11.3. Commercial standard arithmetic chips in classic technologies

1.12. Hardware description languages

1.12.1. VHDL (intended for all Ch1 and Ch2 CSD projects)

1.12.2. Verilog (not covered)

1.13. Target chips: programmable logic devices (PLD) [L4.3]

1.13.1. sPLD (ispGAL22V10) and programmable arrays to implement logic functions

1.13.2. CPLD: complex programmable logic device

1.13.3. FPGA: field programmable gate array

1.14. Electronic design automation (EDA) tools for synthesis

1.14.1. Lattice Semiconductor ispLEVER Classic / Diamond

1.14.2. Xilinx Vivado / ISE

1.14.3. Intel Quartus Prime (intended for all Ch1 and Ch2 CSD projects)

1.15. EDA tools for testing and verification

1.15.1. ALDEC Active HDL Lattice Edition

1.15.2. Xilinx ISim

1.15.3. Mentor Graphics ModelSim Intel FPGA Starter Edition (intended for all Ch1 and Ch2 CSD projects)

1.16. Additional materials and projects

1.16.1. Questionnaires and controls Q

1.16.2. Problem collection (under revision) P

1.16.3. Preliminary laboratory assignments (PLA) and sub-projects PLA

Chapter II. **Sequential systems**

2.1. The concept of memory in digital circuits. Current state and next state [L5.1]

2.2. Latch RS (*RS_latch*). Asynchronous 1-bit memory cell [L5.1]

2.2.1. Function table, state diagram, timing diagram

2.2.2. How to design *RS_latch* using gates (structural, plan A)

2.2.3. *RS_latch* used as push-button debouncing circuit

2.3. Flip-flop. Synchronous 1-bit memory cell [L5.2]

2.3.1. The concept of CLK signal. Synchronicity

2.3.2. CLK circuits [L8.2]

2.3.2.1. RC circuit

2.3.2.2. Quartz crystal oscillator

2.3.2.3. Integrated circuit 555 oscillator

2.3.3. Timer circuits [L8.2]

2.3.3.1. RC timer and re-triggerable functionality

2.3.2.3. Integrated circuit 555 timer

2.3.4. The concept of clear direct (CD) or asynchronous reset [L5.2]

2.3.5. Power-ON reset and system initialisation[L5.2]

2.3.6. Standard flip-flops [L5.3]

2.3.6.1. RS flip-flop (*RS_FF*)

2.3.6.1.1. Deducing *RS_FF* from *RS_latch*, CLK's rising-edge detector

2.3.6.1.2. Function table, state diagram, timing diagram

2.3.6.1.3. Commercial chip

2.3.6.2. Data flip-flop (*D_FF*)

2.3.6.2.1. Deducing *D_FF* from *RS_FF*

2.3.6.2.2. Function table, state diagram, timing diagram

2.3.6.2.3. Commercial chip

2.3.6.3. JK flip-flop (*JK_FF*)

2.3.6.3.1. Function table, state diagram, timing diagram

2.3.6.3.2. Commercial chip

2.3.6.4. Toggle flip-flop (*T_FF*)

2.3.6.4.1. Function table, state diagram, timing diagram

2.3.6.4.2. Frequency divider by two

2.3.7. Analysis of asynchronous and synchronous circuits based on flip-flops and logic [Lab 5]

2.3.7.1. Method 1: handwritten pen-and-paper analysis and discussion. Has the circuit any application? How many states does the circuit have?

2.3.7.2. Method 2: using Proteus

2.3.7.3. Method 3: using VHDL synthesis and simulation tools (plan C2 circuit)

2.4. Massive digital memories [L5.4]

2.4.1. Symbol, address and data

2.4.2. General architecture

2.4.3. ROM: read-only memory

2.4.3.1.VHDL circuit (ROM_2^nxm, for example: *ROM_16x5*)

2.4.3.2.Method of ROM for implementing logic functions in VHDL, look-up tables (LUT)

2.4.4. (optional) RAM: random access memory

2.4.4.1. VHDL circuit (RAM_2^nxm, for example: *RAM_16x5*) and the idea of intellectual property (IP) specific circuits for a given target technology

2.4.4.2. Tri-state buffer in VHDL

2.5. Finite State Machine (FSM): Concept and design procedure [L6.1] [*Lab 6*]

2.5.1.Specifications

2.5.1.1. Function table, symbol, state diagram

2.5.1.2. CLK and CD circuits

2.5.1.3. Example of timing diagram

2.5.1.4. FSM systematic design procedure

2.5.2.Planning

2.5.2.1. Architecture: canonical, synchronous, plan C1: hierarchical, structural in a single VHDL file

2.5.2.2. State diagram, and FSM adaptation

2.5.2.3. State register: r-bit memory (*D_FF)*, state encoding (binary sequential, Gray, one-hot, etc.)

2.5.2.4. Output logic (CC2): truth table, behavioural interpretation: flowchart

2.5.2.5. Next state logic (CC1): truth table, behavioural interpretation: flowchart

2.5.3. Developing

2.5.3.1. VHDL translation, state enumeration, project location.

2.5.3.2. Synthesis project for a target chip. FSM encoding options

2.5.3.3. Target chip resource usage (*D_FF* registers)

2.5.3.4. RTL and technology view discussion

2.5.3.5. FSM state diagram discussion

2.5.4. Testing (functional)

2.5.4.1. Test-bench fixture schematic and test-bench VHDL file

2.5.4.2. CLK and other signals stimulus processes

2.5.4.3. Functional simulation and wave results discussion

2.5.4.3. Internal signals representation (current_state, next_state)

2.5.5. Testing (technology)

2.5.5.1. Gate-level (timing) simulation and results discussion

2.5.5.2. Propagation time CLK to output measurements (t_{CO})

2.5.5.3. Timing analyser spreadsheet and measurement of the maximum frequency of operation

2.5.6. (optional) Prototyping

2.6. Examples of FSM in VHDL: single-file (plan C1) projects [L6.2]

2.6.1. Designing flip-flops as FSM (two-state machines)

2.6.1.1. RS flip-flop (*RS_FF*)

2.6.1.2. D-type (data) flip-flop (*D_FF*)

2.6.1.3. JK flip-flop (*JK_FF*)

2.6.1.4. T-type (toggle) flip-flop (*T_FF*)

2.6.2. Classroom luminaries control [*Lab 6*]

2.6.3. Push-button de-bouncing filter

2.6.4. 16-key matrix keypad encoder [L6.2]

2.6.5. Traffic light controller

2.6.6. Stepper motor controller

2.7. Standard synchronous sequential systems as FSM [L7.1]

2.7.1. Concept map

2.7.2. Counters

2.7.2.1. Symbol, function table, modulo, timing diagram, state diagram, commercial chips

2.7.2.2. Control signals: count enable (CE), up and down (UD_L) or reversibility

2.7.2.3. Control signals: terminal count (TC) pulse

2.7.2.4. Output code:

2.7.2.4.1. Radix-2 (binary sequential)

2.7.2.4.2. BCD

2.7.2.4.3. One-hot or one-cold

2.7.2.4.4. Gray, Johnson, etc.

2.7.2.5. Design **plan X**: designing counters as FSM for small number of states and any output code

2.7.2.5.1. Example: *Counter_mod12*

2.7.2.5.2. Example: *Counter_BCD_1digit* [Lab 7]

2.7.2.5.3. Example: *Counter_Gray_3bit*

2.7.2.5.4. Example: *Counter_Johnson_5bit*

2.7.3. Radix-2 binary counters (Counter_modM), large number of states [L7.1]

2.7.3.1. Symbol, function table, modulo, timing diagram, state diagram, commercial chips

2.7.3.2. Additional control signal: parallel load (LD) or pre-setting output value

2.7.3.3. Design **plan Y**: designing counters using the VHDL arithmetic library and STD_LOGIC_VECTOR, single-file VHDL project

2.7.3.3.1. Example: versatile/universal *Counter_mod16*

2.7.3.3.2. Example: *Counter_mod12*

2.7.3.3.3. Example: *Counter_mod1M* (one million states)

2.7.3.3.4. Example: *Counter_BCD_1digit* [Lab 7]

2.7.4. Count truncation and count expansion [L7.3]

2.7.4.1. Concepts and chaining signals

2.7.4.2. Design **plan C2**: designing counters using hierarchical structures, standard components (Counter_mod16) and logic, VHDL multiple-file project

2.7.4.2.1. Example of count truncation: *Counter_mod12*

2.7.4.2.2. Example of count expansion: *Counter_BCD_2digit*

2.7.4.2.3. Example of count expansion: Minutes_counter (*Counter_BCD_mod60*)

2.7.4.2.4. Example of count expansion: Hour_counter (*Counter_BCD_mod24*)

2.7.5. *n*-bit data register (Data_reg_nbit) [L7.2]

2.7.5.1. Symbol, function table, parallel load, timing diagram, state diagram, commercial chips

2.7.5.2. Plan Y

2.7.5.3. Plan C2 using components (*Counter_mod16* or *Data_reg_4bit*)

2.7.6. *n*-bit shift register (Shift_reg_nbit) [L7.2]

2.7.6.1. Symbol, function table, parallel load, timing diagram, state diagram, commercial chips

2.7.6.2. Plan Y

2.7.6.3. Plan C2 using components (*Shift_reg_4bit*)

2.8. Dedicated processors or subsystems [L8.1]

2.8.1. Architecture of an advanced digital system

2.8.2. Datapath (operational) unit, data input/output, status signals and flags

2.8.3. Control unit (FSM). Control signals, external inputs and outputs

2.8.4. CLK generator circuit [L8.2]

2.8.4.1. Frequency divider

2.8.4.2. Pulsed to square waveform converter using *T_FF*

2.8.4.3. Adaptable generic structure

2.8.5. Examples and applications of dedicated processors

2.8.5.1. Serial multiplier

2.8.5.2. Programmable timer

2.8.5.3. Serial adder

2.8.5.4. Serial asynchronous receiver and transmitter subsystem (USART)

2.8.5.5. Pulse generator

2.8.5.6. MM:SS timer (P8) {*Timer_MMSS*}

2.9. Additional materials and projects

2.9.1. Questionnaires and controls Q

2.9.2. Problem collection (under revision) P

2.9.3. Preliminary laboratory assignments (PLA) and sub-projects PLA

Chapter III. **Microcontrollers**

3.1. Microcomputer architecture and basics [L9.1]

3.1.1. Microprocessor (µP)

3.1.2. Microcontroller (µC), RAM, ROM, I/O

3.1.3. Harvard and Von Neumann architectures

3.1.4. **PIC18F4520** chip architecture. 8-bit µC from Microchip. The architecture of the PIC18F family all projects

3.1.5. Microchip: PIC16F877A, ATmega8535, ATmega328P (the one used in Arduino boards)

3.1.6. Low-level assembly and high-level C languages

3.2. Other µC families and chips

3.2.1. Texas Instruments

3.2.2. Renesas

3.2.3. NXP

3.2.4. STMicroelectronics

3.2.5. Infineon (Cypress)

3.3. EDA tools for developing and testing

3.3.1. Microchip integrated development environment (IDE) MPLAB X (intended for all Ch3 CSD projects)

3.3.2. C compiler XC8 (intended for all Ch3 CSD projects)

3.3.3. Proteus VSM as the virtual laboratory simulation tool

3.4. Digital I/O. Combinational circuits in C

3.4.1. Specifications

3.4.1.1. Truth table, symbol, timing diagram

3.4.1.2. Hardware-software diagram and software organisation

3.4.1.3. Software organisation: setup, main loop

3.4.2. Planning [L9.2]

3.4.2.1. Hardware schematic

3.4.2.1.1. Connecting switches and buttons

3.4.2.1.2. Oscillator (OSC) and reset (MCLR_L) circuits

3.4.2.1.3. Interfacing LED

3.4.2.1.4. Tri-state logic gates and wire bi-directionality.

3.4.2.2. Software program

3.4.2.2.1. RAM variables

3.4.2.2.2. *init_system()*: I/O port pin configuration data direction register TRIS.

3.4.2.2.3. *read_inputs()* [L9.3]

3.4.2.2.4. *truth_table(),* behavioural description, algorithm

3.4.2.2.5. *write_outputs()* [L9.4]

3.4.3. Development and testing [Lab9]

3.4.3.1. Proteus schematic capture: *pdsprj* file

3.4.3.2. MPLAB project for PIC18F4520: C source file

3.4.3.3. Project compilation and chip configuration files: *hex*, *cof*

3.4.3.4. Proteus simulation and testing. Step by step debugging, watch variables window

3.4.4. Example projects

3.4.4.1. Dual 4-channel multiplexer {*Dual_MUX_4*}

3.4.4.2. 1-digit BCD adder (P9) {*Adder_BCD_1digit*}

3.4.5. (optional) Prototyping and laboratory experimentation

3.4.5.1. Chip programmer and program download

3.4.5.2. In-circuit debugging

3.4.5.3. Measurements and characterisation

3.5. FSM implementation in C language

3.5.1. Specifications [L10.1]

3.5.1.1. FSM concept adaptation to µC

3.5.1.2. CLK interface using interrupts

3.5.2. Planning

3.5.2.1. Hardware-software diagram

3.5.2.2. Interrupts: interrupt service routine *ISR().* External event detection (CLK and push-button interface)

3.5.2.3. *output_logic()* and *write_outputs()*

3.5.2.4. *state_logic()* and *read_inputs()*

3.5.3. Development and testing

3.5.4. Examples

3.5.4.1. 4-bit asynchronous serial transmitter [L10.2](P10) {*Serial_transmitter*}

3.5.4.2. 12-states Johnson sequencer {*Johnson_sequencer_mod12*}

3.5.4.3. 1-digit BCD counter (plan X) [Lab10]{*Counter_BCD_1digit*}

3.5.4.5. Modul 1572 binary radix-2 counter (plan Y) [Lab10] {*Counter_mod1572*}

3.5.4.4. Designing the class 74HCT4017chip (plan X) {*Chip_74HCT405*}

3.5.4.6. Traffic light controller

3.5.4.7. Stepper motor driver

3.6. Peripherals: LCD [L11]

3.6.1. LCD technology and controllers

3.6.2. LCD C libraries

3.6.3. Examples

3.6.3.1- 1-digit BCD adder with LCD display {*Adder_BCD_1digit_LCD*}

3.6.3.2. Serial transmitter with LCD display (P11) {*Serial_transmitter_LCD*} (design phase#2)

3.7. Dedicated processors in C (datapath, control unit) [L12.1]

3.7.1. Hardware-software diagram

3.7.2. Examples: Timer [Lab11]

3.7.2.1. Fixed-time timer with external CLK time-base {*Timer*} (design phase#1)

3.7.2.2. Fixed-time timer with LCD {*Timer_LCD*} (design phase#2)

3.7.3. Examples: event counter

3.7.3.1. Tachometer, speed meter, odometer

3.8. Peripherals: Timer0 (TMR0) [L12.2] and Timer2 (TMR2) [L12.2]

3.8.1. TMR0 hardware circuit and configuration registers

3.8.2. TMR2 hardware circuit and configuration registers

3.8.3. Examples: Timer (continuation)

3.8.3.1. Fixed-time timer with LCD and time-base TMR0 {*Timer_LCD_TMR0*} (design phase#3) [Lab11]

3.8.3.2. Fixed-time timer with LCD and time-base TMR2 {*Timer_LCD_TMR2*} (design phase#4)

3.8.4. Other examples [L12.3]

3,8.4.1. 6-bit Johnson sequencer uisng TMR0 (P12) {*Johnson_sequencer_mod12_LCD_TMR*} (design phase#3)

3.8.4.2. Serial transmitter with LCD and TMR2

3.8.4.3. Duty-cycle modulator: LED dimmer

3.9. Optional extended content. Other peripherals

3.9.1. Timer1 (TMR1)

3.9.2. A/D, EEPROM, USART, I2C, PWM, etc.

3.10. Optional. Other microcontrollers and microcomputers

3.10.1. PIC16F877, ATmega8535

3.10.2. Arduino. ATmega328P chip

3.10.3. Raspberry Pi.

3.11. Project examples, final bachelor thesis, research papers, books, etc.

3.12. Additional materials and projects

3.11.1. Questionnaires and controls Q

3.11.2. Problem collection (under revision) P

3.11.3. Preliminary laboratory assignments (PLA) and sub-projects PLA