Practical Exercise – Mikrocomputertechnik
Prof. Dr. habil Stefan J. Rupitsch
Laboratory for Electrical Instrumentation
and Embedded Systems

Exercise Sheet 8 - Project Flowchart

In the **Exercise Sheets 8 and 9** you can put your previous experience to use in a project. You can choose a project from multiple options, or create your own. Instead of a detailed task description, you will only receive a rough sketch of a task. Besides the basic functionality, you are completely free in the realization of the program. In **Exercise Sheet 8** you will create a graphical representation of your program as a preparation for the implementation. Then, in **Exercise Sheet 9** this program will be implemented accordingly, followed by a short documentation to facilitate the grading process.

Task 1

In this exercise there are **six** task options given, of which you have to **solve only one**. Alternatively, if you have your own idea for a small project, which can be realized with our development board and which seems to have a similar workload as the proposed tasks, please contact the organizers of the course.

Once you have chosen a project, **create a flowchart of your program idea**, i.e. a graphical representation of the algorithm/program you want to implement. Please refer to Fig. 1. Keep the flowchart on an abstract level, so do not include each small detail you would like to use within your code. Therefore, do not include routines such as increasing a counter variable and alike, unless this is crucial for the functionality of your program. Please also do not draw your flowchart by hand but rather use a computer software of your choice. (**9 pts.**)

Option 1 – Simple Oscilloscope and Signal Generator

Realize a simple MSP430-based oscilloscope and signal generator:

- To implement the oscilloscope, create a subroutine that periodically samples an analog input with the ADC of the MSP430, and stream the data to the serial interface. Then, use a PC-based tool such as *SerialPlot*[1] to visualize this data steam. Please use a timer module to define the time base for sampling, either by a timer interrupt to execute the next measurement or by directly feeding the ADC's trigger with the output signal of a timer (see *Repeat-Single-Channel Mode* of the MSP430). To test the oscilloscope, feel free to use buttons, the potentiometer's output `U_POT` or the voltage divider including the light-dependent resistor at the output `LDR`.
- To realize the signal generator, you need to create a digital-to-analog conversion by feeding a PWM signal to the low-pass filter consisting of the elements R_{24} and C_{13}, i.e. feeding the signal into the circuit pin `DAC_IN` and extracting it at `DAC_OUT`.
- Then, use the given DAC to create a sinusoidal, a trapezoidal and a rectangular signal. Use the buttons `PB1` and `PB2` to switch back and forth between the waveforms, `PB3` and `PB4` to decrease and increase the frequency and `PB5` and `PB6` to decrease and increase the amplitude. Keep in mind that due to the low cutoff frequency of the filter (approx. 16 Hz), only slow signals can be generated.

[1] `https://hackaday.io/project/5334-serialplot-realtime-plotting-software`

- In the practical implementation of Exercise Sheet 9, route your signal generator to the signal oscilloscope to verify the functionality of both features.

Option 2 – A Dynamic Light Show
This exercise is for **bachelor** students exclusivley! If you are a master student you will need to choose a different option!

Realize an MSP430 application with advanced LED control, which includes features such as:

- controlling the speed and direction of the LED running light including `D1` to `D4`.
- setting the brightness of the red, green and blue LED using pulse-width modulation.
- setting various LED effects such as constant illumination, blinking and breathing for the red, green and blue LED by appropriately controlling the pulse-width modulation.
- implement a mode in which the brightness is controlled by the ambient light level (using the LDR).

This involves a series of technical details to be implemented:

- Define the LED effects as arrays of integer values which define the time sequence of the brightness values. In an oversimplified example, this means that $\{255, 255, 255, 255\}$ represents a constant illumination, $\{255, 0, 255, 0\}$ a blinking illumination and $\{0, 127, 255, 127, 0\}$ a breathing illumination (please use longer and smoother sequences in your implementation). These values are periodically applied to the LEDs by pulse-width modulation.
- The brightness control should work on top of any LED effect, so scale the arrays' values accordingly.
- Use the potentiometer to set speed and direction of the running right, and the buttons to modify brightness and the type of the LED effects for each LED.
- Make sure that there are two brightness modes. In the first mode, the brightness can be adjusted by using the buttons. In the second mode, the brightness should be adjusted with help of the LDR.
- Bear in mind that the running light and the RGB LED modes should run simultaneously.
- The running light should never be blocked, even if the RGB LEDs' settings are changed.

Option 3 – A Command Line Tool for MSP430 Configuration
Create a microcontroller program that allows to control the MSP430s I/Os and features using serial commands, so that it can basically be configured by a command line tool. The tool should at least be able to

- enable and disable all LEDs individually.
- read the status of all buttons.
- get the analog value from the potentiometer, the LDR and the NTC.

- play a tone of a given frequency for a given duration.
- enable and disable the heater.

This involves a series of functions that run on the MSP430:

- Create functions to realize the given functionality (such as enabling and disabling certain devices) on the microcontroller from your previous exercises.
- Set up a parser function that receives and interprets your serial commands and subsequently executes the required functions.
- When sending a command with the option `-v` (e.g. `enable D1 -v`), a verbose mode shall be activated, which means that the microcontroller gives you a brief feedback on the operation over the serial interface (e.g. `LED D1 enabled.`).
- Return an error message for commands that cannot be successfully decoded due to wrong syntax or options.

Option 4 – Tetris on a Serial Console
Program an executable Tetris game on the microcontroller. For this purpose, realize a graphical output of your game by the serial interface to show it on a terminal program. We recommend to use Putty[2].

- To draw content within Putty, familiarize yourself with the VT100 control codes. For example, you can use the command `serialWrite(0x0C);` to clear the the console on the terminal and to move the cursor to the top left corner. You can also redraw only certain areas of the console. If desired, it is also possible to use different colors.
- Use the buttons on the circuit board for the control.

If you have access to two development boards (e.g. when sharing a flat with one of your fellow students), there are two additional options. If you decide on choosing option 5 or option 6, make sure to have access to two development boards on time. There is only a limited amount of additional microcontroller boards avialable at the library.
Although option 5 and option 6 involve a second development board you might have borrowed from one of your fellow students, you will still have to make sure to develop your own code. Handing in an identical or too similar flowchart/source-code is considered as plagiarism which either results in failing the course work (B.Sc.) or failing the exam (M.Sc.).

Option 5 – Wireless Data Communication Between Two Dev Boards
Realize a wireless data transmission between two boards.

- Send data by the serial interface to board 1, which transmits the data packet to board 2 wirelessly. Board 2 will then return the data by the serial interface. The transfer should be bidirectional.
- You can choose any physical medium for the communication (i.e. optical, acoustic, electromagnetic, ...), just don't connect the boards by wires.

[2] `http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html`

- It may be beneficial to implement some form of error detection under some circumstances (e.g. parity bits or CRC).
- You may implement some form of a flow control, so that the buffer of the serial interface does not overflow (e.g. enable an LED indicating that the system is not able to receive any new commands as long as the data is not transmitted successfully).

Option 6 – Mastermind on Two Dev Boards

Program a modification of the game Mastermind on two development boards. Please make sure you reach out to the library on time since there are only limited boards available. If you decide on borrowing the board from one of your fellow students instead, please be informed that a collaboratoin between the two of you is still prohibited. This implies that your source codes are not identical or too similar. Otherwise, this project will be graded as plagiarism.

- Physically connect two boards by multiple wires for data communication. It is also necessary to connect the **ground pins** of the boards, otherwise the applied potentials are not the same.
- The abstract concept of the game is the following: Both controllers should agree on a common secret code (randomly chosen) and give a visual or audible feedback that the boards are ready to play. Both players have to guess this code one after the other. After each code entry, the system will display how well the entry and the secret code match together.
- You are completely free in designing and modifying the game principle. For example, the code could be the pressing sequence of four buttons (e.g. `PB1` to `PB4`). So, the sequence `PB3` - `PB1` - `PB2` - `PB4` has to pressed in order to win the game. Once you enter any sequence not being the correct one, the LEDs Di (i going from 1 to 4) might indicate that PBi was pressed at the correct position as a feedback. Alternatively, every player might have to enter a code of several 4-bit values. Remember, it's up to you. You can also use other peripherals (for example `PB5` and `PB6`, or else `U_POT`) as input and output.
- 'The feedback for each incorrect entry can be realized as a short visual or audible signal.
- If entered correctly, both boards are intended to provide a visual and audible feedback (depending on whether the corresponding player has won or lost).
- After a few seconds a new game should start.

Task 2

Create a file `feedback.txt` with a brief feedback statement, which contains specific problems and issues you experienced while solving the exercise, additional requests, positive remarks and alike. Import this text file `feedback.txt` in your **C**ode **C**omposer **S**tudio (CCS) project, so that you can upload it together with your software deliverable. (**1 pt.**)

Basics - Flowchart

Flowcharts represent the program flow in an abstract manner. It will be your task to make sure your flowchart is not too abstract but not too detailed either.
Please make sure your flowchart contains only following symbols:

- Terminals: Either *Start* or *Stop*
- Process: Arithmetic operation or data manipulation
- Decision: Conditional operation
- Subroutine: Function which is defined in an additional flowchart

Keep in mind that the flowline (arrows) indicates the logical order in which the program is executed. Therefore, it is absolutely necessary to draw arrowheads.

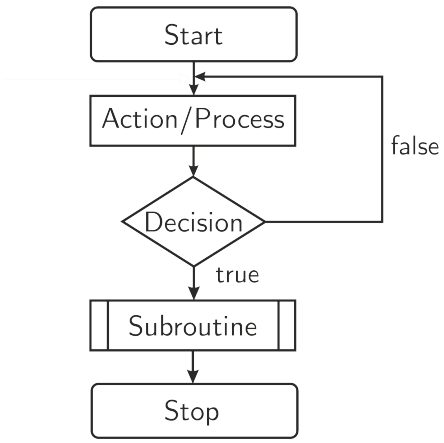

Figure 1: Flowchart Symbols.