**ES-ESP5200 Modern Embedded Systems programming**

**Coursework 2017 / 2018**

**Designing, Developing and Testing an Embedded Level Controller**

Name: **Atle Husmo Undrum**

Partner name: **Torstein Gaarder**

The embedded application we have implemented is a cascade based level regulation of a fluid in a tank.

Part A

**1. Introduction**

We are developing a level controller based on a cascade regulation of flow into the tank. This tank has a manual control of the flow out of the tank. The cascade regulation is based on two sensors, one for the primary process variable and a second one for the secondary process variable. And a PID regulation is based on discrete time steps calculations. The regulation system will also have an alarm function for high and low level in the tank based on two ON and OFF switches. This is the asynchronous aspect with highest priority in the system. The general time discrete PID regulation is given by this formula:

*🡪*

Where is the controller output, is the Proportional element to the output controller, is the Integral element to the output controller and is the Derivative element to the output controller. is the proportional Gain factor. is the size if a time step, is the time step, and is the previous time step. The variable is the regulation error(offset) between Process value and Setpoint value. And is the derivate time, and is the integral time. is the process Setpoint and is the measured and filtered process value. There is also a LCD display for indicating system values as setpoint, current value both level and flow and output value to the pump. These are indicated in percent of the measured values. Figure 1 shows a normal PID regulator, while Figure 2 shows the specific PID implementation we are using.

|  |  |
| --- | --- |
| Figure 1 General PID | Figure 2 Out PID implementation |

2. Design and Development

The system developed are a regulation system with cascade and PID regulation. The hardware consists of a Pump (P-101), two analogue transmitters (LT-101 and FT-101), two level switches (LAH-101 and LAL-101), a pushbutton for resetting the level alarms (LAR-101), two LEDs for indicating level alarms (LAIH-101 and LAIL-101), a tank (T-101), also a LCD display for showing process values (I-101) and a manual valve for manipulating flow out of the tank (V-201). And to control all this an Atmel ATmega 2560 chip placed on an Arduino Mega Board, this chip also includes the flow controller (FC-101) but is called Level controller (LC-101) this is due to that it is the level in the tank that the system is supposed to control.

The general equation for discrete PID regulation is given in the introduction. But in the cascade regulation the flow control gets its setpoint from the level controller and adjust accordingly. This gives an effect that can be seen in the display there the value of the flow transmitter and the value of the output to the pump should be close in value to each other. Figure 3 shows the P&ID for the system.



Figure 3 P&ID

The software program for the Atmel ATmega 2560 consists of several parts. The code is divided into files (libraries) from 3rd party supplier, license file and the main code. The 3rd party supplier files are:

* I2cmaster.h
* Twimaster.c
* Pcf8574.h
* Pcf8574.c
* Lcdpcf8574.h
* Lcdpcf8574.c

In these files the setup of the TWI (I2C) bus is configured. It is necessary to set the address of the slave (0x27) in the pcf8574.h file and the values of the LCD parallel ports in the lcdpcf8574.h file.

The main.c file is also divided into several parts. These are:

* Includes and definitions
* Functions
  + InitialiseGeneral
  + InitialiseTimer1
  + InitialiseADC
  + InitialiseTimer3\_FastPWM\_Single
  + Initialise\_HW\_Interrupts
  + UpdatePID
* Structs
  + PID FlowController
  + PID LevelController
* Variables
* Interrupt Service Routines (ISR)
  + ISR(ADC\_vect)
  + ISR(TIMER1\_COMPA\_vect)
  + ISR(INT4\_vect)
  + ISR(INT5\_vect)
* Main
  + Main(void)

Figure 4 shows the structure of the program with IO and data flow.



Figure Program structure

##### Function explanation

As seen over the program has several functions and parts, here a short explanation of each of them follows:

* The main is responsible for writing to the LCD display, trigger the Alarm LED when LAH or LAL has been triggered and to reset the alarms. It’s also responsible for initializing the initializers. And to scale the variables for the LCD output.
* The InitialiseGeneral is responsible for initialising //stuff which doesn't go in other initialisations. // Like port declaration and enable interrupts.
* The InitialiseTimer1is responsible for initialising an interrupt on a one second interval //Copied from TimerDemo3.
* The InitialiseADC is responsible for initialising the ADC inputs.
* The InitialiseTimer3\_FastPWM\_Single is responsible for initialising the PWM for the output to the pump (P-101)
* The Initialise\_HW\_Interrupts is responsible for initialising the hardware interrupts.
* The ISR(ADC\_vect) is the interrupt handler for the ADC channels.
* The ISR(TIMER1\_COMPA\_vect) is the interrupt handler for the timer 1 for the PID regulator, it’s also responsible for scaling and updating the register (OCR3A) for the PWM.
* The ISR(INT4\_vect) is the interrupt handler for the LAH.
* The ISR(INT5\_vect) is the interrupt handler for the LAL.
* The UpdatePID is the PID controller and it is running inside the ISR(TIMER1\_COMPA\_vect)
* And the struct PID is for storing PID values

Figure 5 shows the flow of most of the program functions.



Figure 5 Program flow

##### I/O List



##### Control Registers

Atmega 2560 Control Registers and their values and expected control range













##### Timing requirements

The timing requirement for the system aren’t that complex, but they are important for proper operations of the system, namely cascade level control in a single tank. Figure 6 shows the sequence diagram for the system, while Figure 7 shows the timing diagram.

##### Prioritization

There will be four interrupts with priority in the system. Level alarm high, level alarm low, PID calculations, and analogue to digital conversion. They will be implemented to get priorities to match the order of this list, with level alarm high getting the highest priority. A PWM timer will also be used, but because it won’t use the CPU, its priority isn’t that important to consider.

##### Scheduling

There will be no scheduling beyond using interrupts. In this system, the use of interrupts should be more than enough to guarantee proper function.

##### Event periodicity

The PID calculations must be periodic to get a proper result and the PWM must be periodic to be able to properly control the servo. Making the PWM periodic won’t be a challenge as it is a hardware timer which automatically resets and little or no impact from the CPU. Making the PID periodic should also be strait forward, as the only other interrupt which triggers during normal operations has a lower priority, and the period is quite long at one second. If the time to calculate the results is less than one second, the PID should stay periodic, and PID calculations are not very resource intensive to do.

##### WCET

The worst-execution time for the PID controllers must be less than one second. This will be achieved by having everything it relies on controlled by a single timer interrupt. No other timers used will have higher priority. The only parts of the system with higher priority will be the alarm inputs, and when they are triggered, something is wrong with the levels.

The only other parts of the system which cares about execution time is the PWM, which doesn’t use CPU, and the ADC, which only uses a small amount of CPU when it has completed a conversion.

Updating the LCD is a slow process which will take a long time, but it’s not in any way critical for the operation of the system, so it’s not necessary to worry about its WCET.

##### Programmable timers.

Only two programable timers are used, timer 1 and timer 3.

* Timer 1 is set up with an output compare march interrupt. This interrupt causes the PID calculations to run and the output to be updated. The exact duration this timer should be tuned for the individual application, but it is important that it is periodic.
* Timer 3 is set up as fast PWM. As such it doesn’t need to generate an interrupt as the PWM automatically outputs to Port E bit 3 & 4.



Figure 6 Sequence diagram



Figure 7 Timing diagram

3. Testing

To test this system there has been developed a Test Plan with functional testing and non-functional testing. Every hardware component has been identified with a Test ID and a pass/fail grade and other comments for further work and improvements. Also, software component has been identified with a Test ID and a pass/fail grade and other comments for further work and improvements. The same goes for Non-functional tests.



4. Program code listings

The program code is included as a separate .pdf file.

Part B

5. Critical evaluation and conclusion

##### Evaluation

The system is not properly (finished due to some errors) developed because of several things. These are: power management, interrupt handlers and code maintainability. These problem areas of the system will be explained further. But the overall function cascade regulation function of the system works properly. There are some hardware faults that also should be corrected mainly the servo of this system. Further explanations about these problems and its solutions will be given in a later paragraph.

The development process was an iterative and cooperative one between Torstein and Me. We started the project with a short description including a general PID Equation and a P&ID diagram of the system to be developed. After that we decided the Inputs- and Outputs- ports to the ATmega 2560 chip

Then we chose a code base to start from. This includes an example project and library for the LCD display we had available and a code base for a PID regulation for C code and code provided from class for the PWM output and ADC input.

The coding process was done in an iterative way with rapid prototyping and testing on the actual board configuration. The main work done in coding was the code for the LCD display and the code for the PID regulation itself driven from the ISR(TIMER1\_COMPA\_vect) and the Struct for PID variables. There was also some work with the output from the PID regulation to the PWM output, here there were some scaling adjustments work to be done. My partner Torstein did most of the regulation part and I did most of the HMI on the LCD part. So, for the coding part it was divided roughly into 50/50 % between each other.

For the documentation parts Torstein did Timing Requirements including figures to the timing Requirements. While I did I/O list and control registers list and test plan. Both of us worked on the P&ID. For the other parts of the report Torstein did the drawings while I did the writing, this was also a roughly divided into 50/50 % between each other.

The testing of the system was done in an iterative manner as mentioned before. This worked well except the LAH-101 and LAL-101 was implemented to late in the process and was not finished before deadline. The parts of the system that works well is the ADC, the LCD, the PID-regulation and the PWM output.

##### Problems and Errors

There were some problems in this project. One of them was the servo where due to inaccuracy in the servo module itself the servo did not respond properly to our control signal. The other was the HW interrupts for the Level Alarms. Here a solution might be to change the port for these inputs. And an effect of this was that the LAIH-101, LAIL-101 and LAR-101 was not properly tested. Under the testing another problem occurred. The problem was that the LCD flickered while the servo was adjusting its position. Probable solution to this is to implement extra power into the system or to chose other parts for the system.

##### Conclusions

In conclusion the system can be considered a prototype not a finished project. This is due to the problems described in the above paragraph. For future work the Level Alarms, the servo unit, the power usage and development of code libraries should be prioritised. And to test it against an actual test environment.

##### Learning outcomes

The learning outcome of this project has been repetitioning of things and skills from earlier.