COSTA RICA INSTITUTE OF TECHNOLOGY

- **Computer Engineering**
- **Programming Languages, Compilers and Interpreters**
- **Proyecto #1 - Temporalizador Led**
- **Project by: Roberto Bonilla and Pablo Rodríguez**

Introduction

The tecnology has grown at big steps, mainly in the middle of computer science and electonical engineering. Therefore is useful to develop creative elements and ingeneous that allow us to walk together with this creations and improve our creativity.

Problem Description

The project consists in a LED temporizer that has 4 modes. It blinks the lights making the patterns. It is implemented in a microcontroller Attiny85 using AVR ASM.

Development Environment

- Atmel Studio 7
- AVRDude
- AVRA
- GitHub

Alternative Solutions

We tought using the Atmega328p but we didnt use it because it was a simple and ligthweight problem. And a atmega is more powerful and unnecessary. So we choose the attiny85, it also is cheaper and simpler. With one port you can do everything.

Program design

The algorithm for the leds is based on a auxiliar PORTB register so we change the bit of that. Using CPI to compare when the led has compared the limit of the bit we restart the counter. Alternating between left shift and right shift. When changing the mode we change the limit. While doing the delay the button state is read and if the button is pressed the flag T is raised and the mode is changed. The delay algorithms just uses 3 registers to do arithmetic operations

decreasing by one. the Time is variable depending of the mode.

Hardware description

Materials: - ATTiny85 - 6 green LED - 6 red LED - 42 yellow LED - 1 kiloohm resistor - 330 ohm resistor - 1 Button - 5 volt voltage source

Three green LED, three red LED and three yellow LED are connected in parallel to create the two flags. Twenty one yellow LED are connected in parallel to create the C letter. Twenty three yellow LED are connected in parallel to create the E letter. These four arrays are connected in parallel between the microcontroller and the 330 ohm resistor that goes to the ground. The button is connected to the 10 kiloohm resistor that comes from the voltage source, the microcontroller is connected on the same node to allow lectures. The button is also connected to ground. The voltage source and the ground are also connected to the microcontroller.

Project final status

The project works at its 100 percent.

Issues Found

- Issue: The program doesnt do anything.
- Solution: Check for the jumps and infinite loops
- Issue: The input is not being read.
- Solution: Be sure to be using PINB and not PORTB
- Issue: series connected LED's instensity is almost null.
- Solution: connect them in parallel.
- Issue: a LED is turned off.
- Solution: check it, if is damaged change it, if isn't find the lost connection to weld it again.

Conclusions

- Its a realiabe solution for simple projects that requires few I/O pins to use an attiny.
- Its a very powerful tool
- LEDs have to be connected un parallel to generate the same voltage, because in series the voltage decreases.
- Microcontrollers are a good tool for simple and small circuits.
- Assembly is hard and unproductive but more powerful.
- Assembly language is literal and does exactly what you tell.

Suggestions and recommendations

- Reading the documentation helps a lot, and avoids errors.
- Make a lot of tags, it helps the organization of the code.
- Research about what exactly each instruction does, and experiment.
- Use the simulator when writing assembler to a microcontroller.
- If you are going to weld use gloves and glasses.
- Be sure you are welding the definitive circuit.
- Be sure you are welding undamaged components.
- Buy a few more components than you need because it's easy to burn them.

Student's Activity Log

Chart 1. Roberto Bonilla's timesheet.

Activity	Time(h)
Planning	2
Research	1
Buying components	1
Welding components	5
Unwelding components	1
Welding components again	5
Tests	0.08
Replacing leds	0.3
Tests	0.25
Writing documentation	2

Chart 2. Pablo Rodriguez's timesheet.

Activity	Time(h)	Description
Planning	2h	Meeting with Roberto
Researching	1h	Researching on the Internet and with the tutors
Writing code	5h	Making the assembler code
Writing code	4h	Debugging the button
Writing documentation	2h	Internal Documentation and External

References

- Atmel Corporation. (2014). Atmel AVR 8-bit Instruction Set. San Jose, California: Atmel Corporation.
- "Atmel 8-bit AVR Microcontroller with 2/4/8K Bytes In-System Programmable Flash." N.p., n.d. Web. 17 Oct. 2015.
- "Calculating Execution Time for Code with Single Loop." Calculating Execution Time for Code with Single Loop. N.p., n.d. Web. 17 Oct. 2015.
- "Where Is Include File?" AVR Freaks. N.p., n.d. Web. 17 Oct. 2015.
- "How to Make a Delay in Assembly for Avr Microcontrollers?" - Stack Overflow. N.p., n.d. Web. 17 Oct. 2015.
- "How to Create Delays in AVR Assembly Language." - Arduino Stack Exchange. N.p., n.d. Web. 17 Oct. 2015.

Code

.include "tn85def.inc"

.def BUTTON = R25 ; Last State of BUTTON

.def SPEED = R24 ;1-4

.def LED = R23 ;1-4 number of led

.def TEMP = r16

.cseg

;Initialize Inputs and Outputs

```
LDI TEMP,(1<<DDB4)|(1<<DDB3)|(1<<DDB2)|(1<<DDB1);PB0 as Input

LDI BUTTON, 1

LDI SPEED, 1

OUT DDRB,TEMP
```

;The algorithm to move the leds is based on a register LED

;representing a PORTB auxiliar and we can go changing the register

;and doing an OR with the fixed leds we get the result. Using CPI and the

;limit bit we restart the cycle

Start1:

```
CLT;Using flag T to specify change mode

LDI SPEED, 4
```

Restart1:

```
LDI LED, 1
```

Mode1:

```
LSL LED; change led

CPI LED,(1<<5) ; Subtract (1<<5) from LED

BREQ restart1; Branch if we got to the edge

LDI TEMP, (1<<PB0) ; fixed leds, none except for the pull up

OR TEMP, LED ;

OUT    PORTB, TEMP

RCALL  delay

BRTS start2; Changes mode depending of the flag T

RJMP mode1 ;Loop
```

; The structure is based on an init for each mode, The start for the cycle and

; the change of the led

Start2:

```
CLT

LDI SPEED, 3
```

Restart2:

```
LDI LED, (1<<4)
```

Mode2:

```
LSR LED

CPI LED, 1

BREQ restart2

LDI    TEMP,(1<<PB0)|(1<<PB4)

OR TEMP, LED

OUT    PORTB, TEMP

RCALL  delay

BRTS start3

RJMP mode2;Loop
```

Start3:

```
CLT

LDI SPEED, 2
```

Restart3:

```
LDI LED, 1
```

Mode3:

```
LSL LED

CPI LED,(1<<3) ; Subtract 3 from LED

BREQ restart3

LDI    TEMP,(1<<PB0)|(1<<PB4)|(1<<PB3)

OR TEMP, LED

OUT    PORTB, TEMP

RCALL  delay

BRTS start4

RJMP mode3;Loop
```

Start4:

```
LDI SPEED, 1

CLT

LDI LED, (1<<5)
```

Mode4:

```
LDI    TEMP,(1<<PB0)|(1<<PB4)|(1<<PB3)|(1<<PB2)|(1<<PB1)

OUT    PORTB, TEMP

RCALL  delay

BRTS start1

LDI    TEMP,(1<<PB0)|(1<<PB4)|(1<<PB3)|(1<<PB2)

OUT    PORTB, TEMP

RCALL  delay

BRTS start1

RJMP mode4;Loop
```

;Delay subroutine

:Algorithm: Using three registers for a delay, reducing by one this values

; And executing the read button to dont waste time while delaying

Delay:

```
MOV r20, SPEED      ; One clock cycle;

LSL r20

LSL r20
```

Delayloop:

```
LDI r19, 25

Delay1:

LDI r18, 10

Delay2:

LDI r17, 75
```

```
Delay3:

DEC r17

NOP

BRNE delay3


RCALL readbutton

BRTS return


DEC r18

NOP

BRNE delay2


RCALL readbutton

BRTS return


DEC r19

NOP

BRNE delay1


RCALL readbutton

BRTS return


DEC     r20           ; One clock Cycle
```

```
BRNE    delayloop          ; Two clock cycles
```

Return:

```
RET
```

ReadButton:

```
SBIC pinb, 0; reads the button

RJMP highpin

CPI BUTTON , 0; look for a change in state for a pressed button

LDI BUTTON, 0

BRNE changestate

RET
```

HighPIN:; Button not pressed

```
LDI BUTTON, 1

RET
```

ChangeState:

```
SET ; Use flag t for button

RET
```