ECEN301 Embedded Systems Lab 3
PWM, LDRs, Interrupts & Timers Submission

Daniel Eisen 300447549

September 2, 2020

1 Objectives

The purpose of this lab was to introduce and familiarise the use of PWM generation, internal and external interrupts. The use of interrupts allows for events of interest to instantaniously be 'noticed' and trigger specific code in a section known as the 'Interrupt Service Routine.'

2 Methodology

2.1 Introduction

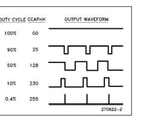
This lab consisted of 2 sections. One implemented simple PWM output with a varying duty cycle set by the IO box. The second utilised a comparator and light sensor to form a simple brightness level transducer

2.2 Part I: PWM Output

PWM generation on the 8051 utilises the PCA counters compare function. Therefore the PCA must be enabled (set CR bit), and the module in use (in this case 0) being set into PWM mode (PWN0, CCAPM0.1) with the compare function enabled (ECOM0, CCAPM0.6).

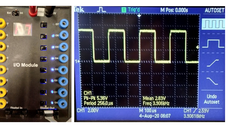
```
/*
    main.c

    Created : 8/4/2020 10:07:46 AM
    Author  : eisendani
*/
#include "AT89C51AC3.h"

void main(void)
{
    CR = 1;
    CCAPM0 |= 0b01000010;
    CCAPH = 128;
    while (1)
    {
        CCAP0H = P0;
    }
}
```

The duty cycle of the PWM output is set by loading an 8bit unsigned value input the CCAP0H registers, with 0 resulting in a 100% cycle and 255 being near zero.

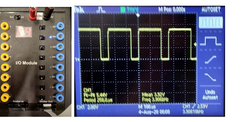


https://sites.google.com/site/controlandelectronics/pwm-tutorial-using-8051

To test the implementation, the IO module was connected to P0 to provide input to the duty cycle register and the output pin, CEX0, was probed with a scope.



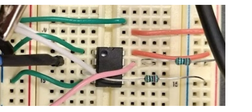
Seen above is a 128 input the 50% output.



Seen above is a 64 input the roughly 2 thirds duty cycle output.

2.3 Part II: LDRs and External Interrupts

To construct the brightness change transducer the main sensing unit used is a Light Dependent Resistor. Using this in a voltage divider, a light level dependant variable voltage can be extracted. To trigger an external interrupt, there must be either a falling or rising edge generated on the input pin to be detected. To translated the variable voltage into 2 distinct levels a comparator was used. Specifically the LM311 in its single supply operation. To put it simply there are 2 inputs, 1 is held constant and a varying signal is connected to the other. When the varying signal is large or smaller (in voltage) than the reference voltage the output will snap to either supply rail, in this case 5V or 0V. So when inputting the sensor voltage, then required edge can be generated when the light level sufficiently changes and an external interrupt can be triggered.



Above shows the comparator circuit as constructed. With measurement of the varying resistance of the LDR a 10K was chosen for the other side of the divider. Also due to the open-collector output of the LM311 the output is nominally pulled high through a 10K pullup.

```
/*
    main.c

    Created : 8/4/2020 11:34:30 AM
    Author  : eisendani
*/
#include "AT89C51AC3.h"
#include "ECEN301LibSDCC.h"
#include <stdint.h>
#include <stdio.h>

volatile unsigned int count = 0;

void ISR (void) __interrupt (0)
{
    count++;
}

void main(void)
{
    IEN0 =  0b10000001; //enable external interrupt 0
    IEN1 = 0; //disable SPI and ACD inter
    TCON =  0b00000001; //set to select falling edge active

    char str[16];

    initLCD();

    while (1)
    {
        clearLCD();
        sprintf(str, "%d", count);
        writeLineLCD(str);
        delay(10000);
    }
}
```

Now that the hardware of the transducer is constructed and functional. The uC must be setup to enable external interrupts on the selected pin (with others disabled) and the relevant edge type set, in this case falling.
A simple ISR just counts the number of interrupts generated, making the (wrong) assumption that accurately represents number of large light level changes and it then just displayed to the LCD panel as in previous labs.
Notes: The interrupt flag is hardware cleared so the program doe not have to handle this.
Due to output bouncing on the comparator output the count is in fact inaccurate, see Q.4 for more details.

3 Questions

1. ***What is meant by 'bit addressable'?***
   This is an attribute of specific registers on the 8051. It describes the registers as being able to be set either with a full word, ie the full value of the entire register:

   ```
   ADCON = 0b0000000
   ```

   Or each/some bits of the register can be set individually using its 'bit mnemonic':

   ```
   PSIDLE=1 /*equivalent*/ ADCON |= 0b0100000
   ```

2. ***Explain what the bits of the CCAPMn and CMOD registers do.***
   The CCAPMn are the special function registers that are associated with each of the modules in the PCA.

   Bit 0 : ECCF, this enables or disables the CCON CCFx flag to allow for an interrupt request be generated when in match or compare mode.

   Bit 1 : PWM, this bit enables the pulse width modulation mode (on CEX output pin).

   Bit 2 : TOG, when set the output pin CEX will toggle when a math

   Bit 3 : MAT, this sets when the PCA matches the capture/compare register. This triggers an interrupt when enabled.

   Bit 4 , 5 : CAPN,CAPP. These determine enable the edge type trigger of the capture mode. **P**ositive and/or **N**egative

   Bit 6 : ECOM, enables comparator function.

   Bit 7 : Reserved

   The CMOD register is the PCA counter mode SFR.

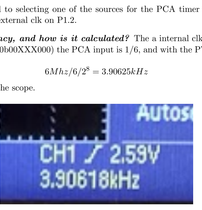
   Bit 0 enables an interrupt to be generated when the PCA overflows, ie CCON:CF is set, others are reserved.

   Bits 2-1 are dedicated to selecting one of the sources for the PCA timer input; clk/6, clk/2, timer0 overflow, and external clk on P1.2.

3. ***What is the frequency, and how is it calculated?*** The a internal clk frequency is 6Mhz, by default (CMOD = 0b00XXX000) the PCA input is 1/6, and with the PWM using the lower 8 of 16 bits:

   $$6Mhz/6/2^8 = 3.90625kHz$$

   This is confirmed on the scope.

   To change the frequency a different PCA input could be chosen, ie clk/2 or an external clock. But for greater control; using a auto-reloaded timer0 at varying values and using the overflow input can allow for varying the frequency in software.

4. ***Does switch bounce occur, and if so how do you deal with it? Does your software restrict the number of interrupts you can generate in real time (i.e. is the software working at a restrictively slow pace)?***
   Yes, bouncing occurred quite heavily, often counting upwards of 3 time per external trigger. The software itself does not handle, filter or attempt to mitigate the comparator bouncing in anyway and runs as fast as possible.

   To eliminate bouncing the LM311 can setup with hysteresis to allow for thresholding the output switching levels. This can be achieved with non-inverting feedback (R2) and input (R1) resistors to get:

   $$V_{TH} = ((R1 + R2) * V_{ref} - (R1 * V_{output-low}))/R2$$
   $$V_{TL} = ((R1 + R2) * V_{ref} - (R1 * V_{output-high}))/R2$$