Universidade de São Paulo-USP
Escola de Engenharia de São Carlos-EESC
SEL0629 - Aplicações de Microprocessadores I

Vinicius William da Silva - Número USP: 11233842
Prof. Marcelo Andrade da Costa Vieira

São Carlos, 2023

1 Introdução

Nesta prática utilizei o microcontrolador PIC18F45k22 por meio do Kit de Desenvolvimento EasyPIC v.7 da MikroEletronika. O objetivo da prática consiste em aprender a utilizar I/O, temporizador e interrupções do PIC18F45k22 ao desenvolver um sistema que, ao lê um teclado matricial e toca a frequência de uma nota musical da escala de Dó Maior no *buzzer*, de acordo com o botão especificado.

As teclas utilizadas no teclado matricial vão de 1 a 8 e geram uma saída na porta RE1 do PIC, saída essa que é uma onda quadrada de frequência correspondente à nota musical.

O teclado matricial é lido por uma varredura em um *loop* e é conectado na porta B, onde 4 pinos de I/O serão saída e os outros 4 entrada. A lógica de leitura é definir o estado lógico 1 quando nenhuma tecla é pressionada nos pinos de entrada e nível lógico 0 quando um botão corresponde for pressionado nos pinos de saída.

A interrupção utilizada para gerar cada onda será o temporizador TMR0.

2 Resultados Experimentais

2.1 Esquemático

O esquemático gerado no software **Proteus** está a seguir.

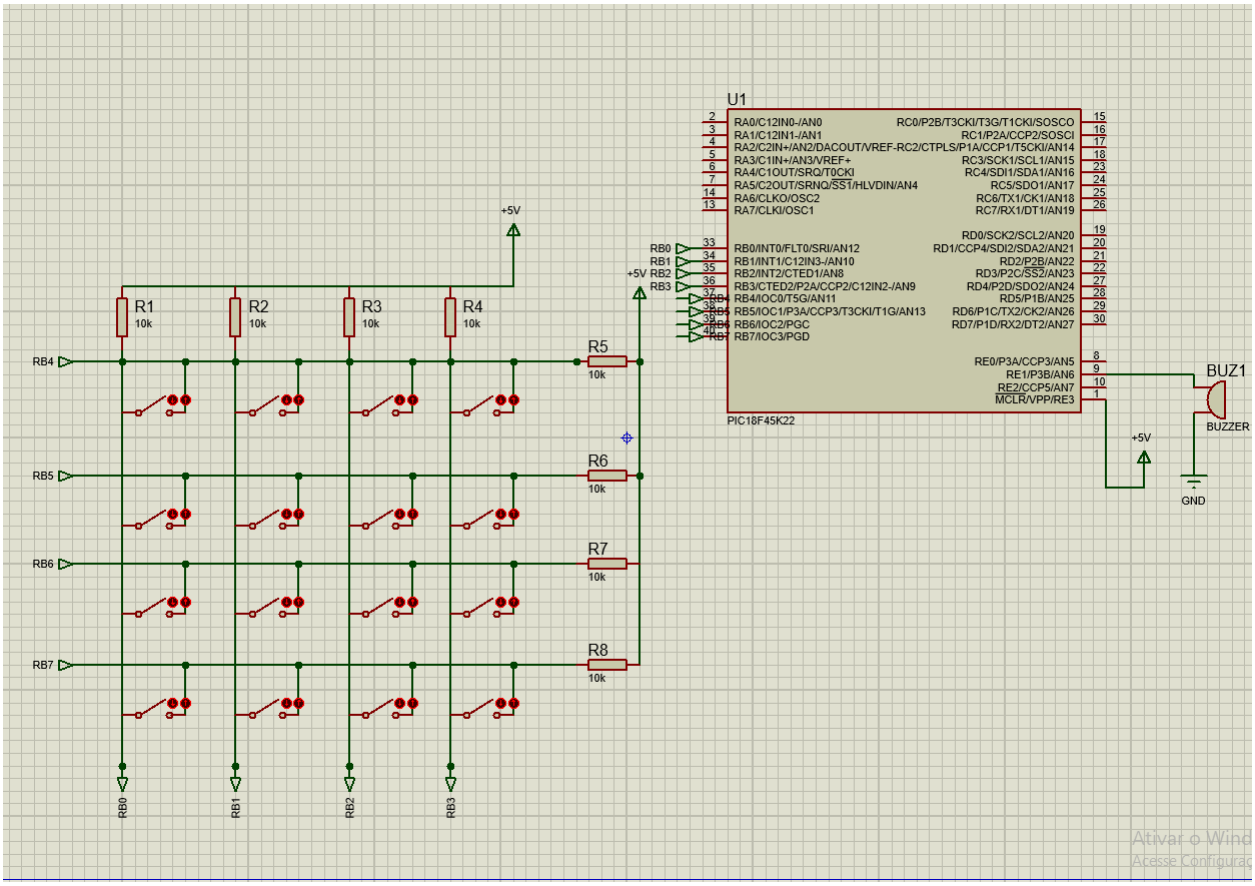

Figura 1: Esquemático do circuito.

2.2 Códigos e Simulação

As variáveis globais necessárias para serem manipuladas no código são inicialmente declaradas, nesse caso são duas variáveis que armazenam os valores de 8 bits para compor o TMR0.

A função de interrupção baseia-se no *overflow* da *flag* "TMR0IF". Ao estourar, ele inverte a saída do *buzzer*, reseta o temporizador e limpa a *flag*.

```
char timerh, timerl;
void interrupts() iv 0x0008 ics ICS_AUTO {
    if(tmr0if_bit) {
```

```
        // Inverte a saida do buzzer (liga/desliga)
        late.f1 = !late.f1;


        // Reseta o temporizador
        tmr0l = timerl;
        tmr0h = timerh;
        tmr0if_bit = 0;
    }
}
```

Há uma função para escanear o teclado matricial. Ela percorre as linhas e colunas do teclado e, e verifica o estado de cada botão. Caso ache um botão pressionado, ela o retorna.

```
char scanKeypad() {
    char button = -1;  // Valor padro para indicar que nenhum
   boto foi pressionado
    char row, columns, col;

    // Varrer as linhas do teclado
    for (row = 0; row < 3; row++) {
        // Configurar a linha atual como sada e as outras como
   entrada
        LATB = ~(1 << (row + 4));

        // Ler o estado das colunas
        columns = PORTB & 0x0F;

        // Verificar se algum boto foi pressionado
        if (columns != 0x0F) {
            // Encontrar a coluna correspondente ao boto
   pressionado
```

```
            for ( col = 0; col < 3; col++) {
                if (!(columns & (1 << col))) {
                    button = (row * 4) + col;  // Calcula o
   nmero do boto pressionado
                    if(button >= 3 && button <= 7)
                              return button-1;
                    else if(button > 7 && button <= 10)
                                   return button-2;
                    break;
                }
            }
            break;  // Sai do loop se um boto foi pressionado
        }
    }

    // Configurar os pinos do teclado matricial como entrada
   novamente
    TRISB = 0x0F;

    return button;
}
```

Na função **main**, são declarados os valores dos para os bits de TMR em timerHigh e timerLow, valores calculados para darem a frequência de cada nota musical dada no enunciado da prática.

Os registradores "INTCON"e "T0CON"habilitam o TMR0 e as interrupções, bem como define o *prescaler* como 1:2. Os registradores "INTCON2"e "WPUB"desabilitam os resistores de *pull-up* internos. Por fim, são declaradas como digitais as portas B e E e definidos os bits menos significativos como entrada e os 4 mais significativos como saída no PORTB onde será conectado o teclado matricial.

```
    // Interrupcao (Apenas timer 0)
    // timer0 de 16 bits com prescaler 1:2.
    rcon.ipen = 0; // Sem prioridade
    intcon = 0b11100000;
    t0con = 0b00000000;
    intcon2 = 0b0000100;
    WPUB = 0;

    // Teclado matricial tem mais significativos como
    // saida e menos como entrada
    anselb = 0;
    trisb = 0b00001111;

    // Buzzer digital
    ansele = 0;
    trise = 0;
    late = 0;
```

Por fim, o *loop* checa qual o valor do botão pressionado (se houver) e registra na variável "buttonPressed". Caso exista um botão pressionado, os valores de "timerh"e "timerl"são definidos de acordo com o botão e o TMR0 é iniciado.

```
    while(1) {
        buttonPressed = scanKeypad();
        if(buttonPressed != -1) {
            timerh = timerHighVector;
            timerl = timerLowVector[buttonPressed];
            t0con.f7 = 1;
            i = 1;
        }
        else if(!(i++)){
```

```
            t0con.f7 = 0;
        }
    }
```

3 Resultados e Discussão

Para avaliar a precisão das frequências emitidas, utilizei um osciloscópio para medir a frequência na saída RE1.

O osciloscópio apontou os seguintes resultados:

Tabela 1: Resultados obtidos por osciliscópio.

Nota Musical	Frequência Esperada (Hz)	Frequência Medida (Hz)	Desvio (%)
Dó	2073	2070	0.14
Ré	2349	2352	0.12
Mi	2637	2638	0.03
Fá	2794	2801	0.25
Sol	3136	3134	0.06
Lá	3520	3508	0.34
Si	3951	3952	0.02
Dó	4146	4149	0.07

4 Conclusão

Ao comparar os resultados, observou-se uma ótima precisão, com a porcentagem de erro está abaixo de 1% . Logo, o *prescaler* utilizado e as interrupções de TMR0 mostraram-se bastante eficientes para gerar as ondas quadradas no *buzzer*.