NEA630 – MICROPROCESSADORES

Projeto para P2

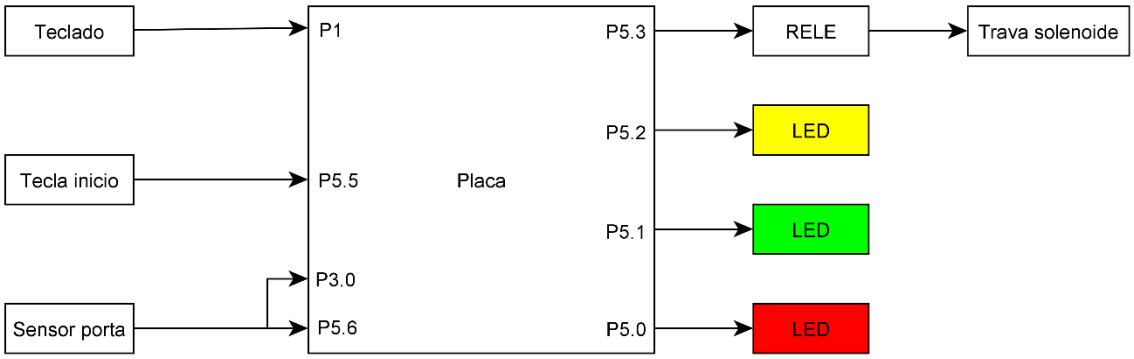
Gustavo Ryuji Sanomia 12.115.481-9

Projeto

A proposta é adaptar o projeto anterior de forma a ter as funcionalidades timer e interrupção. Assim:

- A função de interrupção substituirá o acionamento do alarme, simplificando o código, uma vez que não terá a função de *listening* do sensor nele.
- Originalmente, iria utilizar o alarme para redefinir a senha, assim a cada 24 horas a senha seria redefinida através de um algoritmo. Porém a função alarme só funciona com se o RTC estiver usando o clock XT1, ao invés disso, usarei o timer.

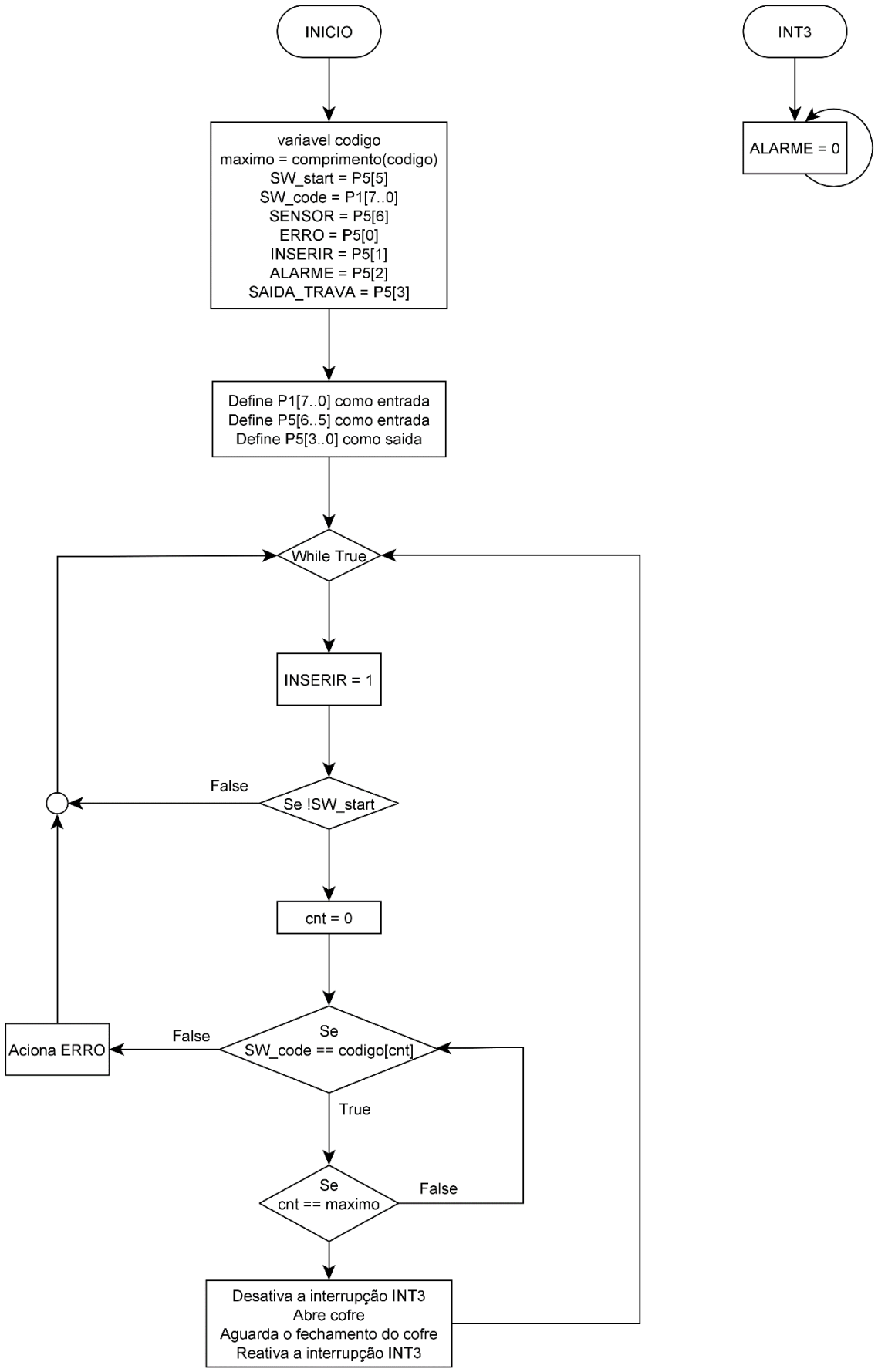
Diagrama simplificado do projeto:



O sensor agora está ligado na porta P3.0, definido para interrupção INT3.

As simulações e códigos foram feitos no software *"IAR EW for Renesas RL78 4.20.1"*, configurado para o hardware *"RL78 - R5F100LE"*.

Fluxograma


INICIO
variavel codigo
maximo = comprimento(codigo)
SW_start = P5[5]
SW_code = P1[7..0]
SENSOR = P5[6]
ERRO = P5[0]
INSERIR = P5[1]
ALARME = P5[2]
SAIDA_TRAVA = P5[3]
Define P1[7..0] como entrada
Define P5[6..5] como entrada
Define P5[3..0] como saida
While True
INSERIR = 1
False
Se !SW_start
cnt = 0
Aciona ERRO
False
Se
SW_code == codigo[cnt]
True
Se
cnt == maximo
False
Desativa a interrupção INT3
Abre cofre
Aguarda o fechamento do cofre
Reativa a interrupção INT3
INT3
ALARME = 0

Agora, ao abrir o cofre, é desativada a interrupção INT3, caso contrário o alarme iria acionar. Quando o alarme é acionado, assim como projeto anterior, ele permanece acionado até o desligamento ou reset da placa.

Código fonte

```
#include "ior5f100le.h"
#include "ior5f100le_ext.h"
#include "intrinsics.h"
#include "myRL78.h"

// Configura watchdog = desligado
#pragma location = "OPTBYTE"
__root __far const char opbyte0 = WDT_OFF;
// Configura detector de baixa tensão = desligado
#pragma location = "OPTBYTE"
__root __far const char opbyte1 = LVD_OFF;
// oscilador 32MHz flash high speed
#pragma location = "OPTBYTE"
__root __far const char opbyte2 = FLASH_HS | CLK_32MHZ;
// debug ativado, com apagamento em caso de falha de autenticação
#pragma location = "OPTBYTE"
__root __far const char opbyte3 = DEBUG_ON_ERASE;
/* Configura security ID */
#pragma location = "SECUID"
__root __far const char senha[10] = {0,0,0,0,0,0,0,0,0,0};

#define SW_start  PM5_bit.no5
#define SW_code  PM1
#define SENSOR  PM5_bit.no6

#define ERRO  PM5_bit.no0
#define INSERIR  PM5_bit.no1
#define ALARME  PM5_bit.no2
#define SAIDA_TRAVA  PM5_bit.no3

volatile unsigned long int temp;
volatile unsigned int cnt;

// Senha em hexadecimal
unsigned char codigo[2] = {
  0xF7,
  0xBF
};

// Numero de digitos da senha
unsigned int maximo = sizeof(codigo);

#pragma vector = INTP3_vect
__interrupt void aciona_alarme( void )
{
  ALARME = 0;
  while (1);
}

#pragma vector = INTIT_vect
__interrupt void muda_senha(void)
{
  codigo[0] = 0xA3;
  codigo[1] = 0x5E;
}

void aciona_erro( void )
{
  ERRO = 0;
```

```
  for (temp=5;temp;temp--); // Tempo de espera do erro
  ERRO = 1;
}

void aciona_trava( void )
{
  PMK3 = 1; // desabilita a interrupção externa INTP3
  SAIDA_TRAVA = 0;
  while(SENSOR); // Espera a porta abrir
  SAIDA_TRAVA = 1;
  while(!SENSOR); // Espera a porta fechar
  PIF3 = 0; // apaga o flag de interrupção do INTP3 (ger. ao abrir a
porta)
  PMK3 = 0; // habilita novamente a interrupção externa INTP3
}

void main( void )
{
  // Set de portas
  PM1 = 0xFF; // Define a porta P1x como entrada (Ent. senha)
  PM5_bit.no5 = 1; // Define a porta P55 como entrada (Tecla inicio)
  PM5_bit.no6 = 1; // Define a porta P56 como entrada (Sensor porta)

  PM5_bit.no0 = 0; // Define a porta P50 como saida (LED de erro)
  PM5_bit.no1 = 0; // Define a porta P51 como saida (LED de inserção
de código)
  PM5_bit.no2 = 0; // Define a porta P52 como saida (Alarme)
  PM5_bit.no3 = 0; // Define a porta P53 como saida (Trava)

  //Interrupt INT3
  EGN0 = BIT3; // Borda de descida para INTP3 (P30)
  PIF3 = 0; // apaga o flag de interrupção do INTP3
  PMK3 = 0; // habilita a interrupção externa INTP3

  //Set do Timer
  CMC = 0; // desativa osciladores X1 e XT1
  OSMC = bWUTMMCK0; // configura o LOCO (15kHz)
  RTCEN = 1; // habilita o RTC e o IT
  ITMC = bRINTE | 10; // Configura intervalo de interrupções
  ITMK = 0; // habilita a interrupção do IT

  //Set de saidas
  ERRO = 1; //
  ALARME = 1; // Desativa o alarme
  SAIDA_TRAVA = 1; // Deixa a porta travada

  __enable_interrupt(); // Habilita interrupções globais

  while (1)
  {
    INSERIR = 1; // Apaga a luz apos a inserção do
codigo/destravamento

    // Tecla de abertura acionada
    if (!SW_start)
    {
      ITMK = 1; // Desabilita interrupção Timer
      cnt = 0;
      INSERIR = 0; // Acende LED para inserir o codigo
      for (temp=5;temp;temp--)
      {
```

```
        // Responde apenas se caso as chaves forem acionadas
        if (SW_code != 0xFF)
        {
          if (SW_code == codigo[cnt])
          {
            cnt++;
            // Se o numero de entradas for iqual a do codigo aciona
a trava
            if (cnt == maximo)
            {
              aciona_trava();
              ITMK = 0; // Reabilita interrupção Timer (Pos abertura
e fechamento)
              break;
            }
          }
          else
          {
            aciona_erro();
            ITMK = 0; // Reabilita interrupção Timer (Pos erro)
            break;
          }
        }
      }
      ITMK = 0; // Reabilita interrupção Timer (time out)
    }
  }
}
```

Foram adicionados os set do timer e da interrupção INT3 bem como sua funções

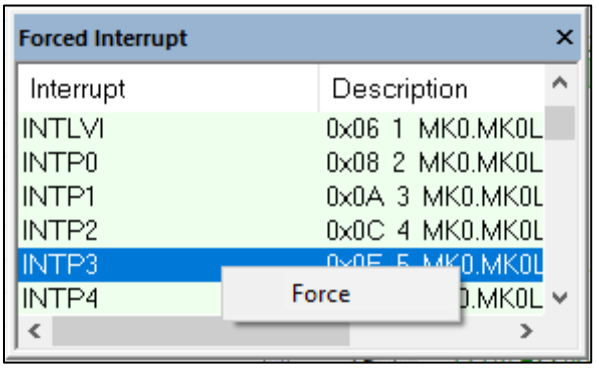
Nota: os valores nos temporizadores são valores teóricos para simulação, não são valores práticos.

Simulação

Nesse relatório focarei apenas nas modificações feitas.

O alarme:

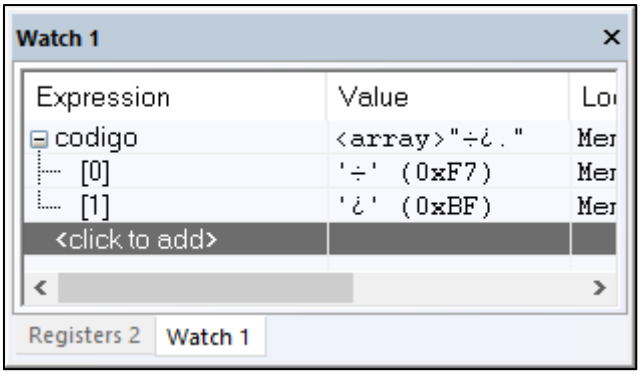
Ao causar uma interrupção, o alarme é acionado.




```
34  #pragma vector = INTP3_vect
35  __interrupt void aciona_alarme( void )
36  {
37    ALARME = 0;
38    while (1);
39  }
```

A geração do token:

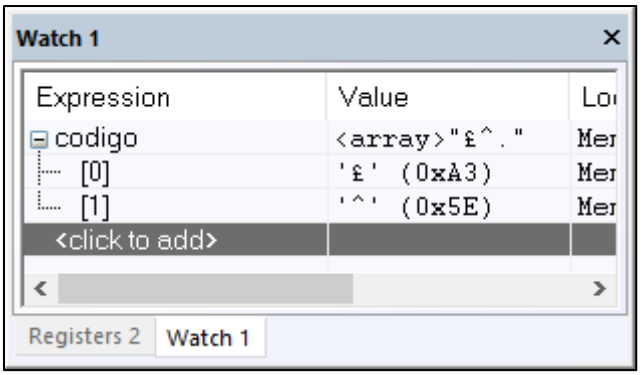
Código é a senha inicial.



Na interrupção INTIT do timer:

```
50  #pragma vector = INTIT_vect
51  __interrupt void muda_senha(void)
52  {
53    codigo[0] = 0xA3;
54    codigo[1] = 0x5E;
55  }
```

O valor do código é alterado.



A princípio utilizaria o RTC para:

- Acionar a interrupção do alarme as 00:00 horas, ou seja, a cada 24 horas.
- Obter os valores do dia, mês e ano para gerar um token único a cada dia.

Considerações

Como utilizei o timer usando LOCO para gerar o token, infelizmente o projeto se torna inviável, o timer tem um limite de contagem em torno de 4000. Todavia, utilizando o clock XT1, ao invés do timer funcionar a 15kHz (LOCO), o clock poderia ser definido para 0,02 Hz (com um contagem no caso de 1728 para 24hrs) ou como dito no relatório anterior, poderia ter sido feito de maneira mais simples e precisa utilizando o RTC e alarme.