```
/* File name:
             util.c
/* File description: This file has a couple of useful functions to */
       make programming more productive
          Remarks: The soft delays consider
          core clock @ 40MHz
           bus clock @ 20MHz
/* Author name: dloubach
                                            */
/* Creation date: 09jan2015
/* Revision date: 31jul2020
#include "util.h"
/* Method name: util_genDelay088us */
/* Method description: generates ~ 088 micro sec */
/* Input params: n/a
/* Output params: n/a
void util_genDelay088us(void)
  char i;
  for(i=0; i<120; i++)
  {
      _asm("NOP");
      _asm("NOP");
    __asm("NOP");
    __asm("NOP");
    __asm("NOP");
    __asm("NOP");
    __asm("NOP");
     _asm("NOP");
    __asm("NOP");
    __asm("NOP");
     _asm("NOP");
    __asm("NOP");
    __asm("NOP");
     _asm("NOP");
    __asm("NOP");
}
/* Method name: util_genDelay250us
/* Method description: generates ~ 250 micro sec */
/* Input params: n/a */
/* Output params: n/a
void util_genDelay250us(void)
  char i;
  for(i=0; i<120; i++)
      _asm("NOP");
      _asm("NOP");
    __asm("NOP");
    __asm("NOP");
    __asm("NOP");
     _asm("NOP");
     __asm("NOP");
    __asm("NOP");
```

```
_asm("NOP");
      _asm("NOP");
  util_genDelay088us();
  util_genDelay088us();
}
/* Method name: util_genDelay1ms */
/* Method description: generates ~ 1 mili sec
/* Input params: n/a */
/* Output params: n/a
void util_genDelay1ms(void)
  util_genDelay250us();
  util_genDelay250us();
  util_genDelay250us();
  util_genDelay250us();
}
/* Method name: util_genDelay10ms
/* Method description: generates ~ 10 mili sec
/* Input params: n/a */
/* Output params: n/a */
/* Output params: n/a
void util_genDelay10ms(void)
  util_genDelay1ms();
  util_genDelay1ms();
  util_genDelay1ms();
  util_genDelay1ms();
  util_genDelay1ms();
  util_genDelay1ms();
  util_genDelay1ms();
  util_genDelay1ms();
  util_genDelay1ms();
  util_genDelay1ms();
}
  /* Method name: util_genDelay100ms
/* Method description: generates ~ 100 mili sec */
/* Input params: n/a */
/* Output params: n/a */
/* Output params: n/a
void util_genDelay100ms(void)
  util_genDelay10ms();
  util_genDelay10ms();
  util_genDelay10ms();
  util_genDelay10ms();
  util_genDelay10ms();
  util_genDelay10ms();
  util_genDelay10ms();
  util_genDelay10ms();
  util_genDelay10ms();
  util_genDelay10ms();
}
```

```
/* Nome do metodo: extrai_digito
/* Descricao:
             Extrai digitos de um numero e armazena num vetor */
/* Parametros de entrada: numero -> numero que os digitos serao extraidos */
     digitos -> vetor onde os digitos serao armazenados */
/* Parametros de saida: n/a
void extrai_digito(unsigned int numero, unsigned char* digitos){
  unsigned char i = 0;
  unsigned int x = 1;
  unsigned int sobra = 0;
  /*Ja inicializa cada digito com "0" (tabela ASCII)*/
  digitos[0] = 48;
  digitos[1] = 48;
  digitos[2] = 48;
  digitos[3] = 48;
  /*Checar quantos digitos tem no numero*/
  while(numero > x){
    x *= 10;
    i++;
  x /= 10;
  /*Divide por uma potencia de 10 para sobrar um digito so*/
  /*Extrai o digito e armazena no vetor +48 (ASCII)
  \mathbf{while}(i > 0)\{
    sobra = numero/x;
    digitos[4-i] = (sobra + 48);
    numero -= (sobra*x);
    x /= 10;
    i--;
  return;
```