Testador de Bobinas até 2A

2013.1

Gerado por Doxygen 1.8.3.1

Terça, 25 de Junho de 2013 09:26:23

Sumário

Capítulo 1

FATEC Santo Andre

Curso de Graduacao em Tecnologia em Eletrônica Automotiva

Disciplina : Carga e Partida - 4o semestre

Professor : Edson Caoru Kitani

Testador de Bobinas até 2A

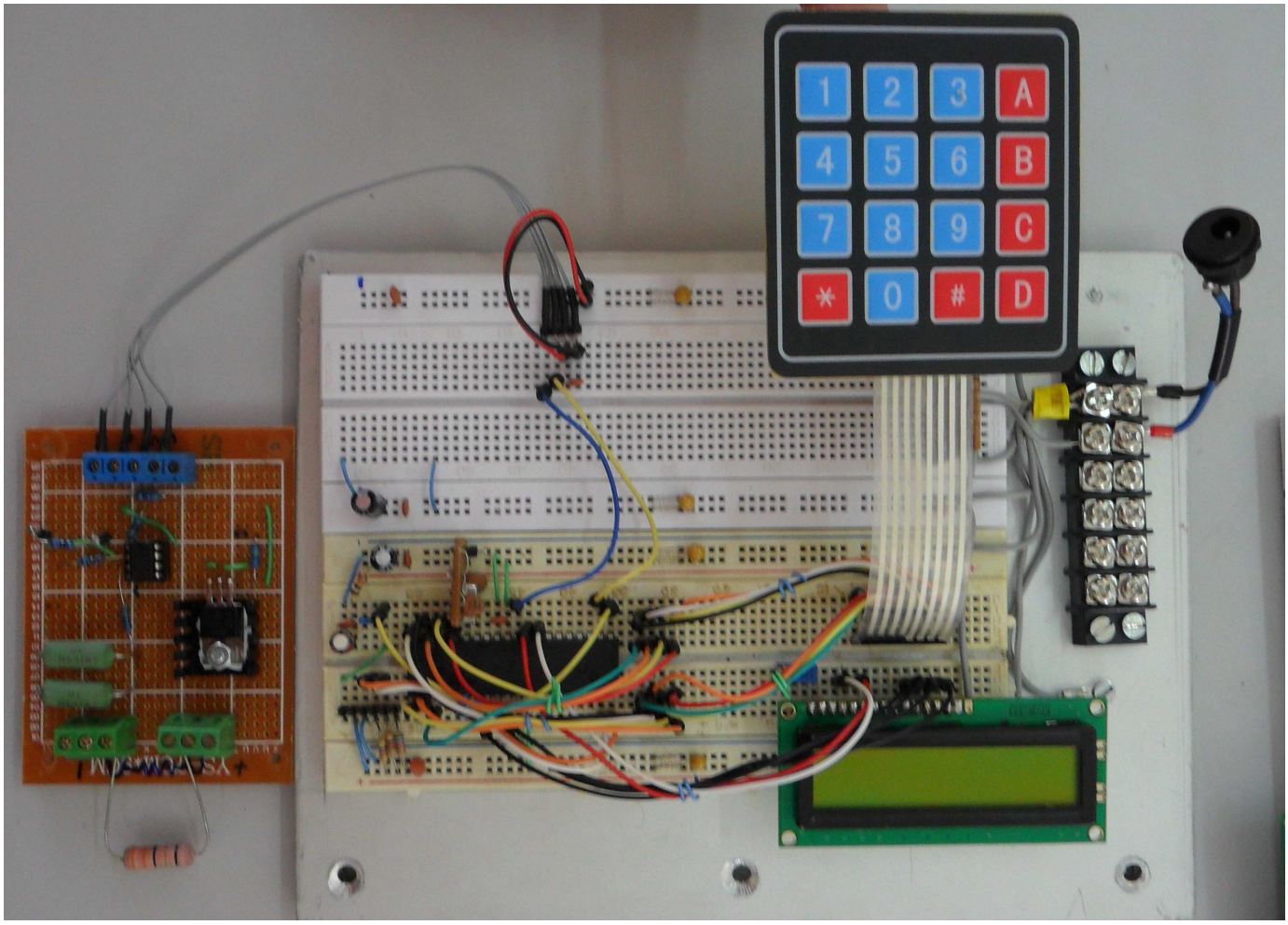


Figura 1.1: Montagem Final

1.1 Integrantes do Grupo

1.1.1 1o semestre de 2013

Autores

Alex Guedes R.A. : 1222009
Pedro Augusto Domingos R.A. : 1123009
Ricardo de Freitas R.A. : 1023007
Rogerio Zavan R.A. : 1123018
Vinicius Garcia Duarte R.A. : 1122018

Capítulo 2

Lista de Figuras

Relacao de Esquemas, Imagens, Gráficos, Figuras

Capítulo 3

Estatísticas de utilização de memória

```
ROM used:   7940 (24%)
            7940 (24%) including unused fragments

            2 Average locations per line
            9 Average locations per statement

RAM used:   667 (43%) at main() level
            685 (45%) worst case

Stack used: 7/31 worst case (5 in main + 2 for interrupts)
```

Lines	Stmts	%	Files
0	0	0	cp.c
9	0	0	_matriz.c
32	0	0	_fuses.h
467	0	0	..\..\..\..\Program Files\PICC\Devices\18F4520.h
224	0	0	regs_P18F4520.h
36	0	0	_matriz.h
46	5	1	_analogico.c
9	0	0	_analogico.h
110	34	4	_delay.c
38	0	0	_teclado.h
19	0	0	_delay.h
72	29	3	_funcoes_gerais.c
162	81	9	_teclado.c
235	105	12	_lcd.c
26	0	0	_lcd.h
423	125	14	..\..\..\..\Program Files\PICC\Drivers\string.h
32	0	0	..\..\..\..\Program Files\PICC\Drivers\stddef.h
28	0	0	..\..\..\..\Program Files\PICC\Drivers\ctype.h
362	172	19	main.c
37	0	0	..\..\..\..\Program Files\PICC\Drivers\stdio.h
1351	359	39	..\..\..\..\Program Files\PICC\Drivers\stdlib.h
11	0	0	_funcoes_gerais.h
3729	910	Total	

Page	ROM	%	RAM	Vol	Diff	Functions:
0	42	0	1			@delay_ms1
0	34	0	1			@delay_us1
0	14	0	0	30	1.3	AD_LeituraFinalizada
0	12	0	0	17	1.3	AD_Leitura
0	6	0	0	31	1.7	AD_TRIS
0	26	0	1	42	1.1	AD_Start
0	6	0	0	31	1.7	PULSO_tris
0	44	0	2	61	2.3	Trata_Timer0
0	164	1	4	687	6.6	Trata_Timer1
0	70	0	5			@DIV1616
0	12	0	1	35	1.7	ASCII_TO_INT
0	296	2	11	781	7.0	LongFix_to_ASCII
0	12	0	0	59	2.3	KB_tris
0	8	0	0	31	1.7	KB_Reset
0	378	2	4	2820	9.3	KB_trata_teclado
0	44	0	0			@goto10096
0	44	0	0			@goto10127
0	26	0	0	74	2.6	KB_Tecla_Nova
0	12	0	0	57	1.9	LCD_tris
0	38	0	3	163	4.2	LCD_escreve_word
0	114	1	2	1071	4.4	LCD_escreve_4bits
0	58	0	1	231	2.3	LCD_pulso_comando

```
0     50   0   2    88  2.1  LCD_posicao_inicio
0    282   2   2  1019  2.4  LCD_inicializacao
0    108   1   5   289  6.8  LCD_mensagem
0     60   0   5   129  4.6  strlen
0     34   0   5               @MUL1616
0   5240  33  27  6628 10.4  MAIN
0     32   0   1               @SPRINTF
0    188   1   9               @PRINTF_LU_530
0     22   0   0               @const704
0     10   0   0               @const706
0     40   0   3               @PSTRINGCN_530
0     10   0   0               @const712
0    154   1   0  1189  2.4  InitHW
0      6   0   0    16  1.0  InitSystem
0     50   0   6   158  5.1  ValidarEntrada
 Inline        0               @cinit1

Program metrics:
    Functions               25
    Statements              910
    Comments                216
    Volume (V)              23298
    Difficulty (D)          93.8
    Effort to implement (E) 2184197
    Time to implement (T)   33 hours, 42 minutes
    Est Delivered Bugs (B)  6
    Cyclomatic Complexity   83
    Maintainability (MI)    27

 Segment     Used  Free
-----------  ----  ----
00000-00002     4  0
00004-00006     0  4
00008-000C4   190  0
000C6-07FFE  7746  24824
```

Capítulo 4

Índice dos Arquivos

4.1 Lista de Arquivos

Esta é a lista de todos os arquivos documentados e suas respectivas descrições:

Capítulo 5

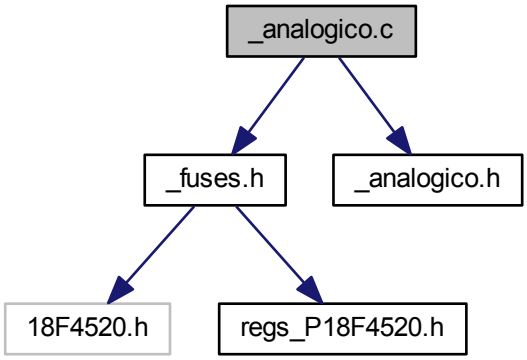
Arquivos

5.1 Referência do Arquivo _analogico.c

Funcoes de leitura dos sinais analogicos.

```
#include "_fuses.h"
#include "_analogico.h"
```

Gráfico de dependência de inclusões para _analogico.c:



Funções

- void AD_LeituraFinalizada (void)

  Fim da conversao analogica.
- unsigned char AD_Leitura (void)

  Medida bruta do AD.
- void AD_TRIS (void)

  Direcao da porta analogica como entrada.
- void AD_Start (char canal)

  Inicio da conversao analogica.

Variáveis

- unsigned char AD_Valor

5.1.1 Descrição Detalhada

Funcoes de leitura dos sinais analogicos.

Definição no arquivo _analogico.c.

5.1.2 Funções

5.1.2.1 unsigned char AD_Leitura (void)

Medida bruta do AD.

Retorna

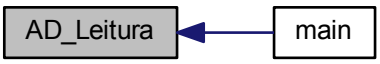
Último valor analógico lido

Definição na linha 24 do arquivo _analogico.c.

Referenciado por main().

```
00024                                {
00025     return AD_Valor;
00026 }
```

Este é o diagrama das funções que utilizam esta função:



5.1.2.2 AD_Start (char *canal*)

Inicio da conversao analogica.

Parâmetros

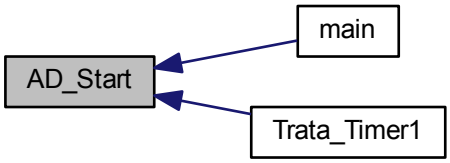
in	*canal*	informar qual porta analogica sera iniciada

Definição na linha 42 do arquivo _analogico.c.

Referenciado por main() e Trata_Timer1().

```
00042                           {
00043     set_adc_channel(canal);
00044     read_adc(ADC_START_ONLY);
00045 }
```

Este é o diagrama das funções que utilizam esta função:



5.1.3 Variáveis

5.1.3.1 unsigned char AD_Valor

Último valor analógico lido

Definição na linha 9 do arquivo _analogico.c.

Referenciado por AD_Leitura() e AD_LeituraFinalizada().

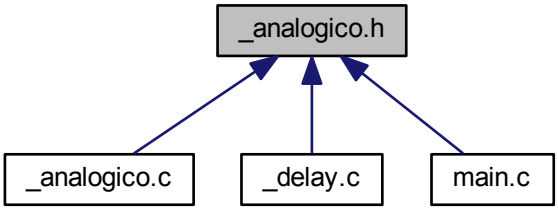
5.2 _analogico.c

```
00001
00006 #include    "_fuses.h"
00007 #include    "_analogico.h"
00008
00009 unsigned char AD_Valor;
00015 #INT_AD
00016 void AD_LeituraFinalizada(void) {
00017     AD_Valor = read_adc(ADC_READ_ONLY);
00018 }
00019
00024 unsigned char AD_Leitura(void){
00025     return AD_Valor;
00026 }
00027
00032 void AD_TRIS(void) {
00033
00034     TRISA = TRISA | 0b00000001;
00035 }
00036
00042 void AD_Start(char canal){
00043     set_adc_channel(canal);
00044     read_adc(ADC_START_ONLY);
00045 }
```

5.3 Referência do Arquivo _analogico.h

protótipos para _analogico.c

Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com este arquivo:



Funções

- void AD_TRIS (void)

  Direcao da porta analogica como entrada.

- void AD_Start (char)

  Inicio da conversao analogica.

- unsigned char AD_Leitura (void)

  Medida bruta do AD.

5.3.1 Descrição Detalhada

protótipos para _analogico.c

Definição no arquivo _analogico.h.

5.3.2 Funções

5.3.2.1 unsigned char AD_Leitura (void)

Medida bruta do AD.

Retorna

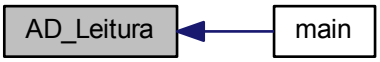
Último valor analógico lido

Definição na linha 24 do arquivo _analogico.c.

Referenciado por main().

```
00024                               {
00025     return AD_Valor;
00026 }
```

Este é o diagrama das funções que utilizam esta função:



5.3.2.2 void AD_Start (char *canal*)

Inicio da conversao analogica.

Parâmetros

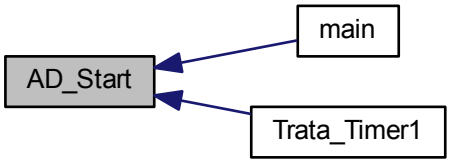
in	*canal*	informar qual porta analogica sera iniciada

Definição na linha 42 do arquivo _analogico.c.

Referenciado por main() e Trata_Timer1().

```
00042                           {
00043     set_adc_channel(canal);
00044     read_adc(ADC_START_ONLY);
00045 }
```

Este é o diagrama das funções que utilizam esta função:



5.4 _analogico.h

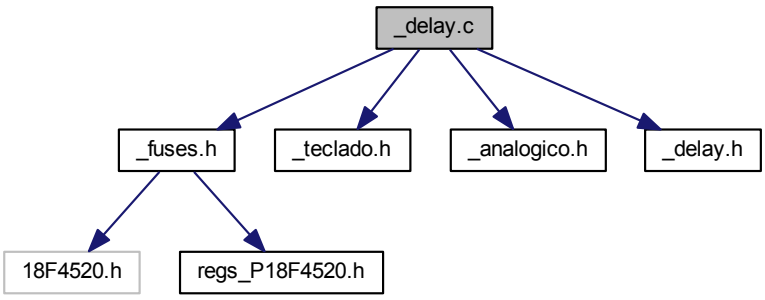
```
00001
00006 void AD_TRIS(void);
00007 void AD_Start(char);
00008 unsigned char AD_Leitura(void);
```

5.5 Referência do Arquivo _delay.c

Controle de tempo, temporizacao, atraso.

```
#include "_fuses.h"
#include "_teclado.h"
#include "_analogico.h"
#include "_delay.h"
```

Gráfico de dependência de inclusões para _delay.c:



Funções

- void PULSO_tris (void)

  define se porta é entrada ou saída
- void Trata_Timer0 (void)

  Interrupcao do timer 0. A cada 1ms.
- void Trata_Timer1 (void)

  Interrupcao do timer 1 a cada 200ns.
- void delay (void)

  Gera delay de 4 nops.
- void Atraso_Milisegundos (unsigned int valor)

  Gera atraso em milisegundos.

5.5.1 Descrição Detalhada

Controle de tempo, temporizacao, atraso.

Definição no arquivo _delay.c.

5.5.2 Funções

5.5.2.1 Atraso_Milisegundos (unsigned int *valor*)

Gera atraso em milisegundos.

Parâmetros

in	*valor*	tempo desejado em ms

Definição na linha 93 do arquivo _delay.c.

```
00093                                           {
00094     unsigned int i;
```

```
00095     unsigned char j;
00096
00097     for (i = 0; i < valor; i++) {
00098
00099         for (j = 0; j < 200; j++) {
00100 #asm
00101             NOP
00102             NOP
00103             NOP
00104             NOP
00105             NOP
00106 #endasm;
00107         }
00108     }
00109 }
```

5.6 _delay.c

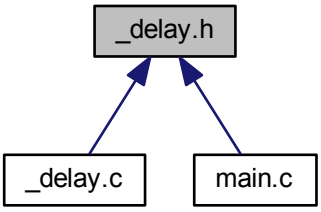
```
00001
00006 // Bibliotecas
00007 #include    "_fuses.h"
00008 #include    "_teclado.h"
00009 #include    "_analogico.h"
00010 #include    "_delay.h"
00011
00016  void PULSO_tris(void) {
00017     TRISE = TRISE & 0b11111011; // forca saidas
00018 }
00019
00025 #int_timer0
00026
00027 void Trata_Timer0(void) {
00028     set_timer0(TMR_Prescaler_1ms + get_timer0());
00029     if (KB_TempoVarredura) KB_TempoVarredura--;
00030 }
00031
00037 #int_timer1
00038
00039 void Trata_Timer1(void) {
00040     static int1 NivelAtual = 0;
00041     static long int TL_ciclo;
00042     long int calculo;
00043     if(NivelAtual){
00044         //fim do tempo em alto, imediatamente
00045         //prepara o proximo nivel: BAIXO
00046         setup_timer_1(T1_INTERNAL | T1_DIV_BY_8);
00047         TL_ciclo = TL; //transfere TL para TL_Ciclo TH em ms
00048         set_timer1(65036 + get_timer1()); //ajustar para 1ms o T1
00049         AD_Start(0); // Ler a corrente antes de zerar a saida
00050         PULSO_PIN = 0;
00051         NivelAtual=0;
00052     } else {
00053         //aguardar fim do TL para armar o sinal em alto
00054         if (TL_ciclo){
00055             TL_ciclo --;
00056             set_timer1(65036 + get_timer1()); // ajustar para 1ms o T1
00057         } else {
00058             //prepara o proximo nivel: ALTO
00059             setup_timer_1(T1_INTERNAL | T1_DIV_BY_1);
00060             // 65136 = 115us > 65196 = 100us(-15us)
00061             // 45536 = 5016ms
00062             //TH em us
00063             if (TH > 15)
00064                 calculo = (TH - 15);
00065             calculo *= 4;
00066             calculo = 0x00 - calculo;
00067             calculo += get_timer1();
00068             set_timer1(calculo);
00069             PULSO_PIN = 1;
00070             NivelAtual=1;
00071         }
00072     }
00073 }
00074
00079 void delay(void) {
00080 #asm
00081     NOP
00082     NOP
00083     NOP
00084     NOP
00085 #endasm
00086 }
00087
```

```
00093 void Atraso_Milisegundos(unsigned int valor) {
00094     unsigned int i;
00095     unsigned char j;
00096 
00097     for (i = 0; i < valor; i++) {
00098 
00099         for (j = 0; j < 200; j++) {
00100 #asm
00101             NOP
00102             NOP
00103             NOP
00104             NOP
00105             NOP
00106 #endasm;
00107         }
00108     }
00109 }
```

5.7 Referência do Arquivo _delay.h

protótipos para _delay.c

Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com este arquivo:



Definições e Macros

- #define TMR_Prescaler_1ms 131
- #define PULSO_PIN LATE2

Funções

- void Atraso_Milisegundos (unsigned int)

  Gera atraso em milisegundos.
- void delay (void)

  Gera delay de 4 nops.
- void PULSO_tris (void)

  define se porta é entrada ou saída

Variáveis

- unsigned long int TH = 9999
- unsigned long int TL = 9999

5.7.1 Descrição Detalhada

protótipos para _delay.c

Definição no arquivo _delay.h.

5.7.2 Definições e macros

5.7.2.1 #define PULSO_PIN LATE2

Sinal de comando para o chaveamento de corrente pelo FET

Definição na linha 15 do arquivo _delay.h.

Referenciado por Trata_Timer1().

5.7.2.2 #define TMR_Prescaler_1ms 131

Prescaler para Timer 0

Definição na linha 12 do arquivo _delay.h.

Referenciado por InitHW() e Trata_Timer0().

5.7.3 Funções

5.7.3.1 void Atraso_Milisegundos (unsigned int *valor*)

Gera atraso em milisegundos.

Parâmetros

in	*valor*	tempo desejado em ms

Definição na linha 93 do arquivo _delay.c.

```
00093                                                   {
00094     unsigned int i;
00095     unsigned char j;
00096
00097     for (i = 0; i < valor; i++) {
00098
00099         for (j = 0; j < 200; j++) {
00100 #asm
00101             NOP
00102             NOP
00103             NOP
00104             NOP
00105             NOP
00106 #endasm;
00107         }
00108     }
00109 }
```

5.7.4 Variáveis

5.7.4.1 unsigned long int TH = 9999

Tempo em Alto

Definição na linha 17 do arquivo _delay.h.

Referenciado por main() e Trata_Timer1().

5.7.4.2 unsigned long int TL = 9999

Tempo em Baixo

Definição na linha 18 do arquivo _delay.h.

Referenciado por main() e Trata_Timer1().

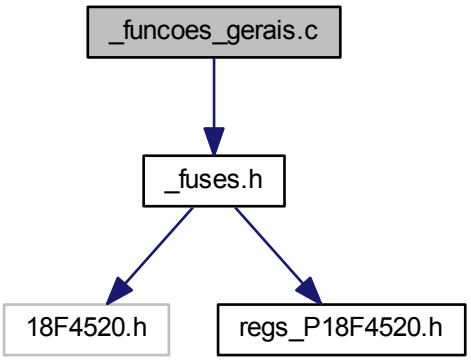
5.8 _delay.h

```
00001
00007 void Atraso_Milisegundos(unsigned int);
00008 void delay(void);
00009 void PULSO_tris(void);
00010
00012 #define TMR_Prescaler_1ms 131
00013
00015 #define PULSO_PIN LATE2
00016
00017 unsigned long int TH = 9999;
00018 unsigned long int TL = 9999;
```

5.9 Referência do Arquivo _funcoes_gerais.c

comandos de manipulacao de textos e numeros

`#include "_fuses.h"`
Gráfico de dependência de inclusões para _funcoes_gerais.c:



Funções

- char * Int_to_ASCII (unsigned int val)

  Converte um inteiro em ASCII.
- int ASCII_TO_INT (char val)

  só para 0~9
- void LongFix_to_ASCII (unsigned long int val, int pos, char *str)

  Converte um inteiro com ponto decimal em ASCII. 999 > 99.9 ou 9.99 ou .999.

5.9.1 Descrição Detalhada

comandos de manipulacao de textos e numeros

Definição no arquivo _funcoes_gerais.c.

5.9.2 Funções

5.9.2.1 int ASCII_TO_INT (char *val*)

só para 0∼9

Parâmetros

in	*val*	valor a ser convertido.

Retorna

valor convertido.

Definição na linha 31 do arquivo _funcoes_gerais.c.

Referenciado por main().

```
00031                           {
00032     return (val-48);
00033 }
```

Este é o diagrama das funções que utilizam esta função:

5.9.2.2 char Int_to_ASCII (unsigned int *val*)

Converte um inteiro em ASCII.

Parâmetros

in	*val*	valor a ser convertido.

Retorna

valor convertido.

Definição na linha 14 do arquivo _funcoes_gerais.c.

```
00014                                   {
00015     char str[6];
00016     str[0] = (val / 10000) + 0x30;
00017     str[1] = ((val % 10000) / 1000) + 0x30;
```

```
00018     str[2] = ((val % 1000) / 100) + 0x30;
00019     str[3] = ((val % 100) / 10) + 0x30;
00020     str[4] = (val % 10) + 0x30;
00021     str[5] = 0;
00022     return str;
00023 }
```

5.9.2.3 LongFix_to_ASCII (unsigned long int *val*, int *pos*, char * *str*)

Converte um inteiro com ponto decimal em ASCII. 999 > 99.9 ou 9.99 ou .999.

Parâmetros

in	*val*	valor a ser convertido.
in	*pos*	ponto decimal.
in,out	*str*	transfere dado

Definição na linha 43 do arquivo _funcoes_gerais.c.

Referenciado por main().

```
00043                                                                   {
00044     int i,p;
00045
00046     p = 3 - pos;
00047     i=0;
00048
00049     if (p==i){
00050         str[i] = '.';
00051         i++;
00052         }
00053     str[i] = ((val % 1000) / 100) + 0x30;
00054     i++;
00055
00056     if (p==i){
00057         str[i] = '.';
00058         i++;
00059         }
00060     str[i] = ((val % 100) / 10) + 0x30;
00061     i++;
00062
00063     if (p==i){
00064         str[i] = '.';
00065         i++;
00066         }
00067     str[i] = (val % 10) + 0x30;
00068     i++;
00069
00070     str[i] = 0;
00071 }
```

Este é o diagrama das funções que utilizam esta função:

5.10 _funcoes_gerais.c

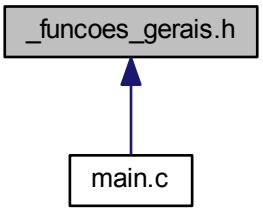
```
00001
00006 #include    "_fuses.h"
```

```
00007
00014 char* Int_to_ASCII(unsigned int val) {
00015     char str[6];
00016     str[0] = (val / 10000) + 0x30;
00017     str[1] = ((val % 10000) / 1000) + 0x30;
00018     str[2] = ((val % 1000) / 100) + 0x30;
00019     str[3] = ((val % 100) / 10) + 0x30;
00020     str[4] = (val % 10) + 0x30;
00021     str[5] = 0;
00022     return str;
00023 }
00024
00031 int ASCII_TO_INT(char val) {
00032     return (val-48);
00033 }
00034
00043 void LongFix_to_ASCII(unsigned long int val, int pos, char *str ) {
00044     int i,p;
00045
00046     p = 3 - pos;
00047     i=0;
00048
00049     if (p==i){
00050         str[i] = '.';
00051         i++;
00052         }
00053     str[i] = ((val % 1000) / 100) + 0x30;
00054     i++;
00055
00056     if (p==i){
00057         str[i] = '.';
00058         i++;
00059         }
00060     str[i] = ((val % 100) / 10) + 0x30;
00061     i++;
00062
00063     if (p==i){
00064         str[i] = '.';
00065         i++;
00066         }
00067     str[i] = (val % 10) + 0x30;
00068     i++;
00069
00070     str[i] = 0;
00071 }
```

5.11 Referência do Arquivo _funcoes_gerais.h

protótipos para _funcoes_gerais.c

Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com este arquivo:



Funções

- char * Int_to_ASCII (unsigned int)

  Converte um inteiro em ASCII.

- void LongFix_to_ASCII (unsigned long int, int, char *)

  Converte um inteiro com ponto decimal em ASCII. 999 > 99.9 ou 9.99 ou .999.
- int ASCII_TO_INT (char)

  só para 0~9

5.11.1 Descrição Detalhada

protótipos para _funcoes_gerais.c

Definição no arquivo _funcoes_gerais.h.

5.11.2 Funções

5.11.2.1 int ASCII_TO_INT (char *val*)

só para 0~9

Parâmetros

in	*val*	valor a ser convertido.

Retorna

valor convertido.

Definição na linha 31 do arquivo _funcoes_gerais.c.

Referenciado por main().

```
00031                           {
00032     return (val-48);
00033 }
```

Este é o diagrama das funções que utilizam esta função:

5.11.2.2 char* Int_to_ASCII (unsigned int *val*)

Converte um inteiro em ASCII.

Parâmetros

in	*val*	valor a ser convertido.

Retorna

valor convertido.

Definição na linha 14 do arquivo _funcoes_gerais.c.

```
00014                                       {
00015     char str[6];
00016     str[0] = (val / 10000) + 0x30;
00017     str[1] = ((val % 10000) / 1000) + 0x30;
00018     str[2] = ((val % 1000) / 100) + 0x30;
00019     str[3] = ((val % 100) / 10) + 0x30;
00020     str[4] = (val % 10) + 0x30;
00021     str[5] = 0;
00022     return str;
00023 }
```

5.11.2.3 void LongFix_to_ASCII (unsigned long int *val*, int *pos*, char * *str*)

Converte um inteiro com ponto decimal em ASCII. 999 > 99.9 ou 9.99 ou .999.

Parâmetros

in	*val*	valor a ser convertido.
in	*pos*	ponto decimal.
in,out	*str*	transfere dado

Definição na linha 43 do arquivo _funcoes_gerais.c.

Referenciado por main().

```
00043                                                                   {
00044     int i,p;
00045
00046     p = 3 - pos;
00047     i=0;
00048
00049     if (p==i){
00050         str[i] = '.';
00051         i++;
00052         }
00053     str[i] = ((val % 1000) / 100) + 0x30;
00054     i++;
00055
00056     if (p==i){
00057         str[i] = '.';
00058         i++;
00059         }
00060     str[i] = ((val % 100) / 10) + 0x30;
00061     i++;
00062
00063     if (p==i){
00064         str[i] = '.';
00065         i++;
00066         }
00067     str[i] = (val % 10) + 0x30;
00068     i++;
00069
00070     str[i] = 0;
00071 }
```

Este é o diagrama das funções que utilizam esta função:

5.12 _funcoes_gerais.h

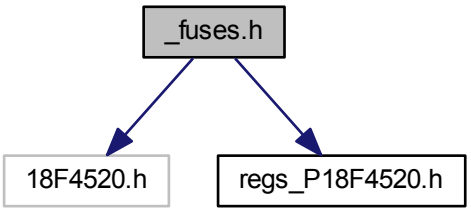
```
00001
00006 char *Int_to_ASCII(unsigned int);
00007
00008 void LongFix_to_ASCII(unsigned long int, int, char *);
00009
00010 int ASCII_TO_INT(char);
```

5.13 Referência do Arquivo _fuses.h

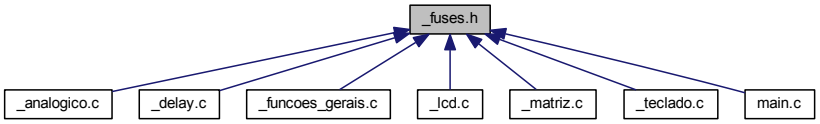
programa o PIC conforme o modelo

```
#include <18F4520.h>
#include "regs_P18F4520.h"
```

Gráfico de dependência de inclusões para _fuses.h:



Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com este arquivo:



5.13.1 Descrição Detalhada

programa o PIC conforme o modelo

Definição no arquivo _fuses.h.

5.14 _fuses.h

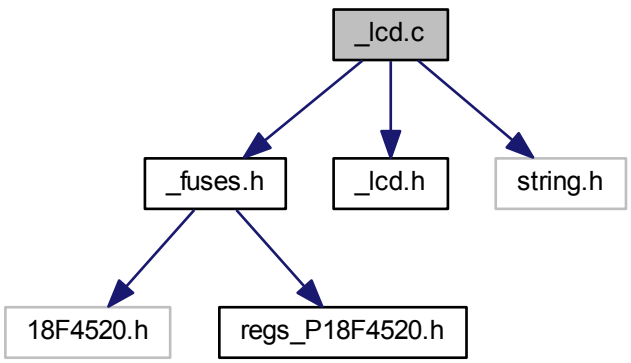
```
00001
00006 // Bibliotecas
00007 #include <18F4520.h>
00008 #include "regs_P18F4520.h"
00009
00010 // FUSES
00011 #FUSES NOIESO,HS,NOFCMEN
00012 #FUSES NOBROWNOUT,NOWRT
00013 #FUSES NOWDT,NOPROTECT
00014 #FUSES PBADEN,NOLPT1OSC,MCLR
00015 #FUSES NODEBUG,NOSTVREN,NOLVP,NOXINST
00016 #FUSES NOWRTC,NOWRTB,NOWRTD,NOPBADEN
00017 #FUSES NOCPD,NOCPB
00018 #FUSES NOPUT,NOEBTR,NOEBTRB
00019
00020 #DEVICE ADC=8 // conversor AD em 8 bits
00021 #use delay (clock = 16000000)
00022 #ZERO_RAM
00023
00024 // inicializacao dos port's
00025 #use fast_io(a)
00026 #use fast_io(b)
00027 #use fast_io(c)
00028 #use fast_io(d)
00029 #use fast_io(e)
00030
00031 #priority timer1,timer0,ad  // Estabelece a prioridade das interrupcoes
```

5.15 Referência do Arquivo _lcd.c

comandos especificos para o LCD

```
#include "_fuses.h"
#include "_lcd.h"
#include <string.h>
```

Gráfico de dependência de inclusões para _lcd.c:



Definições e Macros

5.15.1 JHD162A

Lista de Figuras

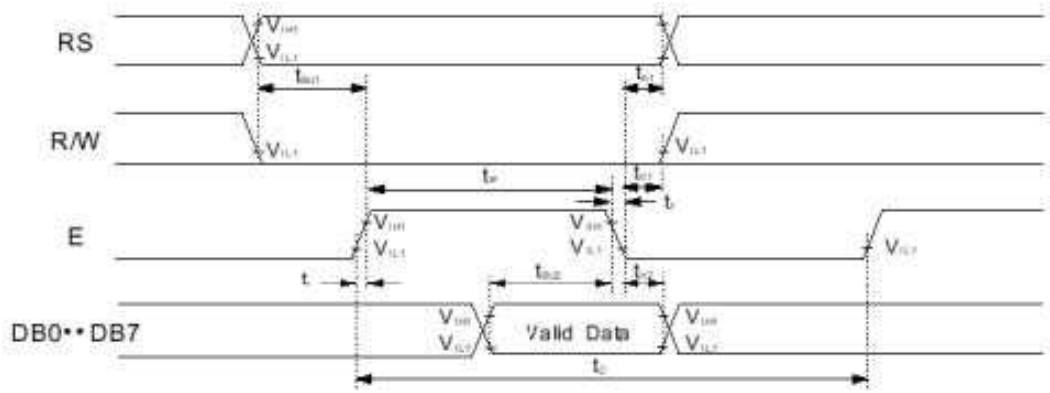


Figura 5.1: JHD162A - pulso de execucao de um comando: RS = 0 - pulso de escrita de letra: RS = 1

- #define lcd_PIN_4 pin_d4
- #define lcd_PIN_5 pin_d5
- #define lcd_PIN_6 pin_d6
- #define lcd_PIN_7 pin_d7
- #define lcd_rs pin_e0
- #define lcd_en pin_e1

- #define __dado 1
- #define __instrucao 0

Funções

- void LCD_tris (void)

  Configura Portas para entrada ou saida.
- void LCD_escreve_word (unsigned char dado, unsigned char tipo)

  Escreve uma letra ou comando no display.
- void LCD_escreve_4bits (unsigned char dado, char pos)

  Escreve no LCD usando ligação de 4 bits.
- void LCD_pulso_comando (unsigned char tipo)

  Aciona as linha RS e EN conforme o tipo do dado.
- void LCD_posicao_inicio (void)

  Movimenta o cursor para a posição inicial do LCD.
- void LCD_inicializacao ()

  Inicialização do display LCD.
- void LCD_mensagem (unsigned char posicao, unsigned char ∗frase)

  Escreve uma frase completa no display.

5.15.2 Descrição Detalhada

comandos especificos para o LCD

Autor

Ricardo de Freitas 1023007

Definição no arquivo _lcd.c.

5.15.3 Definições e macros

5.15.3.1 #define __dado 1

comandos para o LCD

Definição na linha 36 do arquivo _lcd.c.

Referenciado por LCD_mensagem() e LCD_pulso_comando().

5.15.3.2 #define __instrucao 0

comandos para o LCD

Definição na linha 37 do arquivo _lcd.c.

Referenciado por LCD_inicializacao(), LCD_mensagem(), LCD_posicao_inicio() e LCD_pulso_comando().

5.15.3.3 #define lcd_en pin_e1

Bits do LCD

Definição na linha 29 do arquivo _lcd.c.

Referenciado por LCD_inicializacao() e LCD_pulso_comando().

5.15.3.4 #define lcd_PIN_4 pin_d4

Bits do LCD

Definição na linha 23 do arquivo _lcd.c.

Referenciado por LCD_escreve_4bits() e LCD_inicializacao().

5.15.3.5 #define lcd_PIN_5 pin_d5

Bits do LCD

Definição na linha 24 do arquivo _lcd.c.

Referenciado por LCD_escreve_4bits() e LCD_inicializacao().

5.15.3.6 #define lcd_PIN_6 pin_d6

Bits do LCD

Definição na linha 25 do arquivo _lcd.c.

Referenciado por LCD_escreve_4bits() e LCD_inicializacao().

5.15.3.7 #define lcd_PIN_7 pin_d7

Bits do LCD

Definição na linha 26 do arquivo _lcd.c.

Referenciado por LCD_escreve_4bits() e LCD_inicializacao().

5.15.3.8 #define lcd_rs pin_e0

Bits do LCD

Definição na linha 28 do arquivo _lcd.c.

Referenciado por LCD_inicializacao() e LCD_pulso_comando().

5.15.4 Funções

5.15.4.1 void LCD_escreve_4bits (unsigned char *dado*, char *pos*)

Escreve no LCD usando ligação de 4 bits.

Parâmetros

in	*dado*	dado a ser mostrado no display
in	*pos*	Posição no display

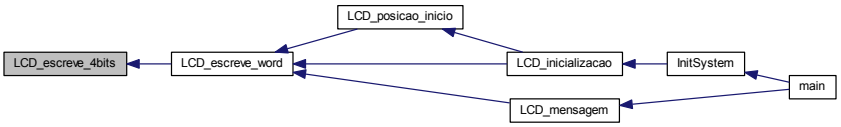
Definição na linha 78 do arquivo _lcd.c.

Referenciado por LCD_escreve_word().

```
00078                                                   {
00079     if (pos == 0) {
00080         if (0 != (dado & 0x80)) {
00081             output_high(lcd_PIN_7);
00082         } else {
00083             output_low(lcd_PIN_7);
00084         }
00085         if (0 != (dado & 0x40)) {
00086             output_high(lcd_PIN_6);
00087         } else {
00088             output_low(lcd_PIN_6);
00089         }
00090         if (0 != (dado & 0x20)) {
00091             output_high(lcd_PIN_5);
00092         } else {
00093             output_low(lcd_PIN_5);
00094         }
00095         if (0 != (dado & 0x10)) {
00096             output_high(lcd_PIN_4);
00097         } else {
00098             output_low(lcd_PIN_4);
00099         }
00100     } else {
00101         if (0 != (dado & 0x08)) {
00102             output_high(lcd_PIN_7);
00103         } else {
00104             output_low(lcd_PIN_7);
00105         }
00106         if (0 != (dado & 0x04)) {
00107             output_high(lcd_PIN_6);
00108         } else {
00109             output_low(lcd_PIN_6);
00110         }
00111         if (0 != (dado & 0x02)) {
00112             output_high(lcd_PIN_5);
00113         } else {
00114             output_low(lcd_PIN_5);
00115         }
00116         if (0 != (dado & 0x01)) {
00117             output_high(lcd_PIN_4);
00118         } else {
00119             output_low(lcd_PIN_4);
```

```
00120         }
00121     }
00122     delay_us(1);
00123 }
```

Este é o diagrama das funções que utilizam esta função:



5.15.4.2 void LCD_escreve_word (unsigned char *dado,* unsigned char *tipo*)

Escreve uma letra ou comando no display.

Parâmetros

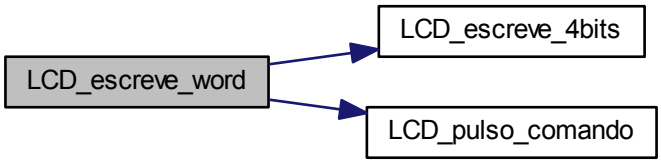
in	*dado*	Caractere a ser enviado ao display
in	*tipo*	Texto ou Comando ?

Definição na linha 65 do arquivo _lcd.c.

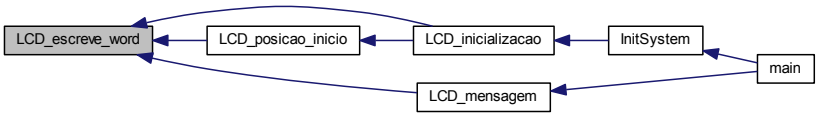
Referenciado por LCD_inicializacao(), LCD_mensagem() e LCD_posicao_inicio().

```
00065                                                       {
00066     char i;
00067     for (i = 0; i < 2; i++) {
00068         LCD_escreve_4bits(dado, i);
00069         LCD_pulso_comando(tipo);
00070     }
00071 }
```

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



5.15.4.3 void LCD_mensagem (unsigned char *posicao*, unsigned char * *frase*)

Escreve uma frase completa no display.

Parâmetros

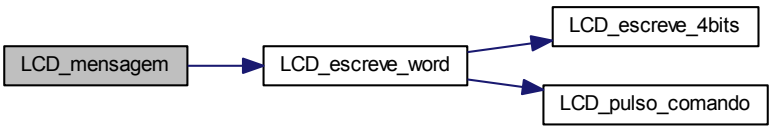
in	*posicao*	Posição a ser escrita no display
in	*frase*	Texto

Definição na linha 224 do arquivo _lcd.c.

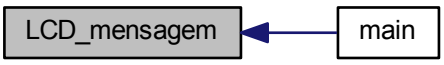
Referenciado por main().

```
00224                                                           {
00225     char i;
00226
00227     LCD_escreve_word(posicao, __instrucao); //comecar da linhaXcoluna
00228     delay_us(40);
00229
00230     for (i = 0; i < strlen(frase); i++) {
00231         if (0 == frase[i]) break;
00232         LCD_escreve_word(frase[i], __dado);
00233     }
00234 }
```

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



5.15.4.4 void LCD_pulso_comando (unsigned char *tipo*)

Aciona as linha RS e EN conforme o tipo do dado.

Parâmetros

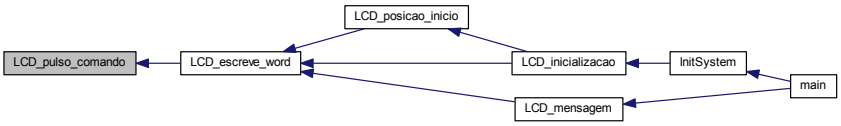
in	*tipo*	Dado ou instrução para o LCD?

Definição na linha 129 do arquivo _lcd.c.

Referenciado por LCD_escreve_word().

```
00129                                           {
00130     //1 - Dado, 0 - Instrução
00131     if (tipo == __dado)
00132         output_high(lcd_rs);
00133
00134     if (tipo == __instrucao)
00135         output_low(lcd_rs);
00136
00137     //Pulso em EN RE1
00138     output_low(lcd_en);
00139     delay_us(5);
00140     output_high(lcd_en);
00141     delay_us(10);
00142     output_low(lcd_en);
00143     delay_us(5);
00144 }
```

Este é o diagrama das funções que utilizam esta função:



5.16 _lcd.c

```
00001
00015 #include    "_fuses.h"
00016 #include    "_lcd.h"
00017 #include    <string.h>
00018
00023 #define lcd_PIN_4 pin_d4
00024 #define lcd_PIN_5 pin_d5
00025 #define lcd_PIN_6 pin_d6
00026 #define lcd_PIN_7 pin_d7
00027
00028 #define lcd_rs pin_e0
00029 #define lcd_en pin_e1
00030
00036 #define __dado 1
00037 #define __instrucao 0
00038
00044 void LCD_tris(void) {
00045     //Hex 0x1F
00046     // Binary 0b010010
00047     // TRISD4 = 0;
00048     // TRISD5 = 0;
00049     // TRISD6 = 0;
00050     // TRISD7 = 0;
00051
00052     // TRISE0 = 0;
00053     // TRISE1 = 0;
00054
00055     TRISD = TRISD & 0b00001111; // forca saidas
00056     TRISE = TRISE & 0b11111100; // forca saidas
00057 }
00058
```

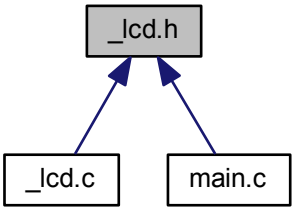
```
00065 void LCD_escreve_word(unsigned char dado, unsigned char tipo) {
00066     char i;
00067     for (i = 0; i < 2; i++) {
00068         LCD_escreve_4bits(dado, i);
00069         LCD_pulso_comando(tipo);
00070     }
00071 }
00072
00078 void LCD_escreve_4bits(unsigned char dado, char pos) {
00079     if (pos == 0) {
00080         if (0 != (dado & 0x80)) {
00081             output_high(lcd_PIN_7);
00082         } else {
00083             output_low(lcd_PIN_7);
00084         }
00085         if (0 != (dado & 0x40)) {
00086             output_high(lcd_PIN_6);
00087         } else {
00088             output_low(lcd_PIN_6);
00089         }
00090         if (0 != (dado & 0x20)) {
00091             output_high(lcd_PIN_5);
00092         } else {
00093             output_low(lcd_PIN_5);
00094         }
00095         if (0 != (dado & 0x10)) {
00096             output_high(lcd_PIN_4);
00097         } else {
00098             output_low(lcd_PIN_4);
00099         }
00100     } else {
00101         if (0 != (dado & 0x08)) {
00102             output_high(lcd_PIN_7);
00103         } else {
00104             output_low(lcd_PIN_7);
00105         }
00106         if (0 != (dado & 0x04)) {
00107             output_high(lcd_PIN_6);
00108         } else {
00109             output_low(lcd_PIN_6);
00110         }
00111         if (0 != (dado & 0x02)) {
00112             output_high(lcd_PIN_5);
00113         } else {
00114             output_low(lcd_PIN_5);
00115         }
00116         if (0 != (dado & 0x01)) {
00117             output_high(lcd_PIN_4);
00118         } else {
00119             output_low(lcd_PIN_4);
00120         }
00121     }
00122     delay_us(1);
00123 }
00124
00129 void LCD_pulso_comando(unsigned char tipo) {
00130     //1 - Dado, 0 - Instrução
00131     if (tipo == __dado)
00132         output_high(lcd_rs);
00133
00134     if (tipo == __instrucao)
00135         output_low(lcd_rs);
00136
00137     //Pulso em EN RE1
00138     output_low(lcd_en);
00139     delay_us(5);
00140     output_high(lcd_en);
00141     delay_us(10);
00142     output_low(lcd_en);
00143     delay_us(5);
00144 }
00145
00149 void LCD_posicao_inicio(void) {
00150     unsigned char info;
00151     info = 0x01; //limpar o display e posicionar o cursor linha 1 coluna 1
00152     LCD_escreve_word(info, __instrucao);
00153     delay_us(1800);
00154 }
00155
00159 void LCD_inicializacao() {
00160
00161     //1-function set
00162     //2-display on off
00163     //3-entry mode set
00164     //4 display clear
00165     //comunicacao 8 vias, duas colunas, 7x5 = 0x38
00166     //comunicacao 4 vias, duas colunas, 7x5 = 0x28
```

```
00167
00168     unsigned char info;
00169
00170     delay_ms(500); //necessario esperar antes de qq coisa
00171
00172     LCD_tris();
00173
00174     output_low(lcd_rs);
00175     output_high(lcd_en);
00176     output_low(lcd_PIN_7);
00177     output_low(lcd_PIN_6);
00178     output_low(lcd_PIN_5);
00179     output_low(lcd_PIN_4);
00180
00181     info = 0x33;
00182     LCD_escreve_word(info, __instrucao);
00183     delay_ms(4);
00184
00185     info = 0x32;
00186     LCD_escreve_word(info, __instrucao);
00187     delay_ms(4);
00188
00189     info = 0x28; //comunicacao 4 vias, duas colunas, 7x5 = 0x28
00190     LCD_escreve_word(info, __instrucao);
00191     delay_ms(2);
00192
00193     info = 0x08; //desliga display
00194     LCD_escreve_word(info, __instrucao);
00195     delay_us(40);
00196
00197     info = 0x0f; //Cursor com Alternância
00198     LCD_escreve_word(info, __instrucao);
00199     delay_us(40);
00200
00201     LCD_posicao_inicio();
00202
00203     info = 0x28; //comunicacao 4 vias, duas colunas, 7x5 = 0x28
00204     LCD_escreve_word(info, __instrucao);
00205     delay_ms(2);
00206
00207     info = 0x0c; //ligar o display sem cursor
00208     LCD_escreve_word(info, __instrucao);
00209     delay_us(40);
00210
00211     info = 0x06; //deslocamento auto para esquerda
00212     LCD_escreve_word(info, __instrucao);
00213     delay_us(40);
00214
00215     delay_ms(500); //aguarda LCD estabilizar
00216     LCD_posicao_inicio();
00217 }
00218
00224 void LCD_mensagem(unsigned char posicao, unsigned char *frase) {
00225     char i;
00226
00227     LCD_escreve_word(posicao, __instrucao); //comecar da linhaXcoluna
00228     delay_us(40);
00229
00230     for (i = 0; i < strlen(frase); i++) {
00231         if (0 == frase[i]) break;
00232         LCD_escreve_word(frase[i], __dado);
00233     }
00234 }
```

5.17 Referência do Arquivo _lcd.h

protótipos para _lcd.c

Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com este arquivo:



Definições e Macros

- #define LCD_colunas 16
- #define LCD_linhas 2
- #define LCD_L1 0x80
- #define LCD_L2 0xC0

Funções

- void LCD_tris (void)
  Configura Portas para entrada ou saida.
- void LCD_escreve_word (unsigned char dado, unsigned char tipo)
  Escreve uma letra ou comando no display.
- void LCD_escreve_4bits (unsigned char dado, char pos)
  Escreve no LCD usando ligação de 4 bits.
- void LCD_pulso_comando (unsigned char tipo)
  Aciona as linha RS e EN conforme o tipo do dado.
- void LCD_posicao_inicio (void)
  Movimenta o cursor para a posição inicial do LCD.
- void LCD_inicializacao ()
  Inicialização do display LCD.
- void LCD_mensagem (unsigned char, unsigned char ∗)
  Escreve uma frase completa no display.

5.17.1 Descrição Detalhada

protótipos para _lcd.c

Definição no arquivo _lcd.h.

5.17.2 Definições e macros

5.17.2.1 #define LCD_colunas 16

LCD 16x2

Definição na linha 10 do arquivo _lcd.h.

Referenciado por main().

5.17.2.2 #define LCD_L1 0x80

LCD 16x2

Definição na linha 13 do arquivo _lcd.h.

Referenciado por main().

5.17.2.3 #define LCD_L2 0xC0

LCD 16x2

Definição na linha 14 do arquivo _lcd.h.

Referenciado por main().

5.17.2.4 #define LCD_linhas 2

LCD 16x2

Definição na linha 11 do arquivo _lcd.h.

5.17.3 Funções

5.17.3.1 void LCD_escreve_4bits (unsigned char *dado*, char *pos*)

Escreve no LCD usando ligação de 4 bits.

Parâmetros

in	*dado*	dado a ser mostrado no display
in	*pos*	Posição no display

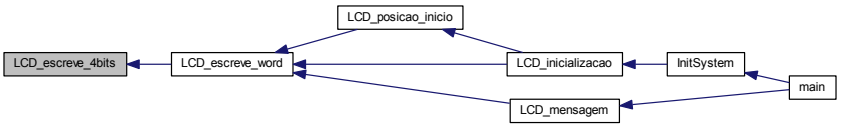
Definição na linha 78 do arquivo _lcd.c.

Referenciado por LCD_escreve_word().

```
00078                                                   {
00079     if (pos == 0) {
00080         if (0 != (dado & 0x80)) {
00081             output_high(lcd_PIN_7);
00082         } else {
00083             output_low(lcd_PIN_7);
00084         }
00085         if (0 != (dado & 0x40)) {
00086             output_high(lcd_PIN_6);
00087         } else {
00088             output_low(lcd_PIN_6);
00089         }
00090         if (0 != (dado & 0x20)) {
00091             output_high(lcd_PIN_5);
00092         } else {
00093             output_low(lcd_PIN_5);
00094         }
00095         if (0 != (dado & 0x10)) {
00096             output_high(lcd_PIN_4);
00097         } else {
00098             output_low(lcd_PIN_4);
00099         }
00100     } else {
00101         if (0 != (dado & 0x08)) {
00102             output_high(lcd_PIN_7);
00103         } else {
00104             output_low(lcd_PIN_7);
00105         }
00106         if (0 != (dado & 0x04)) {
00107             output_high(lcd_PIN_6);
00108         } else {
00109             output_low(lcd_PIN_6);
00110         }
```

```
00111         if (0 != (dado & 0x02)) {
00112             output_high(lcd_PIN_5);
00113         } else {
00114             output_low(lcd_PIN_5);
00115         }
00116         if (0 != (dado & 0x01)) {
00117             output_high(lcd_PIN_4);
00118         } else {
00119             output_low(lcd_PIN_4);
00120         }
00121     }
00122     delay_us(1);
00123 }
```

Este é o diagrama das funções que utilizam esta função:



5.17.3.2 void LCD_escreve_word (unsigned char *dado*, unsigned char *tipo*)

Escreve uma letra ou comando no display.

Parâmetros

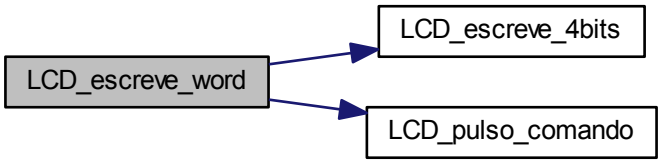
in	*dado*	Caractere a ser enviado ao display
in	*tipo*	Texto ou Comando ?

Definição na linha 65 do arquivo _lcd.c.

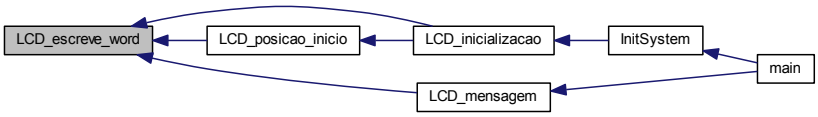
Referenciado por LCD_inicializacao(), LCD_mensagem() e LCD_posicao_inicio().

```
00065                                                       {
00066     char i;
00067     for (i = 0; i < 2; i++) {
00068         LCD_escreve_4bits(dado, i);
00069         LCD_pulso_comando(tipo);
00070     }
00071 }
```

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



5.17.3.3 void LCD_mensagem (unsigned char *posicao*, unsigned char * *frase*)

Escreve uma frase completa no display.

Parâmetros

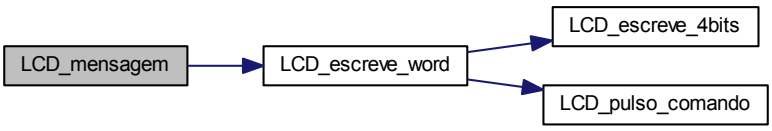
in	*posicao*	Posição a ser escrita no display
in	*frase*	Texto

Definição na linha 224 do arquivo _lcd.c.

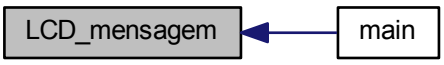
Referenciado por main().

```
00224                                                           {
00225     char i;
00226
00227     LCD_escreve_word(posicao, __instrucao); //comecar da linhaXcoluna
00228     delay_us(40);
00229
00230     for (i = 0; i < strlen(frase); i++) {
00231         if (0 == frase[i]) break;
00232         LCD_escreve_word(frase[i], __dado);
00233     }
00234 }
```

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



5.17.3.4 void LCD_pulso_comando (unsigned char *tipo*)

Aciona as linha RS e EN conforme o tipo do dado.

Parâmetros

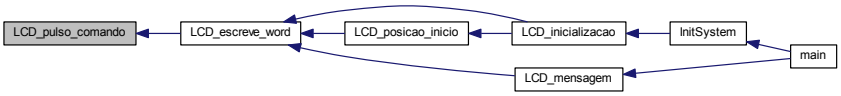
in	*tipo*	Dado ou instrução para o LCD?

Definição na linha 129 do arquivo _lcd.c.

Referenciado por LCD_escreve_word().

```
00129                                           {
00130     //1 - Dado, 0 - Instrução
00131     if (tipo == __dado)
00132         output_high(lcd_rs);
00133
00134     if (tipo == __instrucao)
00135         output_low(lcd_rs);
00136
00137     //Pulso em EN RE1
00138     output_low(lcd_en);
00139     delay_us(5);
00140     output_high(lcd_en);
00141     delay_us(10);
00142     output_low(lcd_en);
00143     delay_us(5);
00144 }
```

Este é o diagrama das funções que utilizam esta função:



5.18 _lcd.h

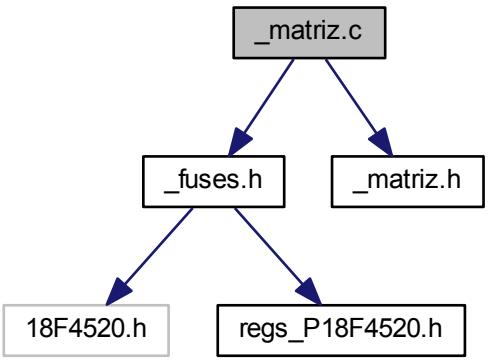
```
00001
00010 #define LCD_colunas 16
00011 #define LCD_linhas 2
00012
00013 #define LCD_L1  0x80
00014 #define LCD_L2  0xC0
00015
00017 // ##### prototipos #####
00018 void LCD_tris(void);
00019 void LCD_escreve_word(unsigned char dado, unsigned char tipo);
00020 void LCD_escreve_4bits(unsigned char dado, char pos);
00021 void LCD_pulso_comando(unsigned char tipo);
00022 void LCD_posicao_inicio(void);
00023 void LCD_inicializacao();
00024 void LCD_mensagem(unsigned char, unsigned char*);
00025 // ##### fim prototipos #####
```

5.19 Referência do Arquivo _matriz.c

Apenas referencia as bibliotecas.

```
#include "_fuses.h"
#include "_matriz.h"
```

Gráfico de dependência de inclusões para _matriz.c:



5.19.1 Descrição Detalhada

Apenas referencia as bibliotecas.

Definição no arquivo _matriz.c.

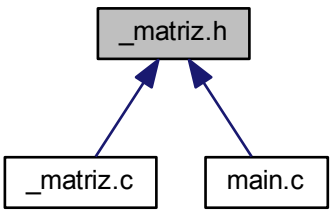
5.20 _matriz.c

```
00001
00006 #include    "_fuses.h"
00007 #include    "_matriz.h"
```

5.21 Referência do Arquivo _matriz.h

protótipos para _matriz.c

Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com este arquivo:



Variáveis

- unsigned long int Matriz [256]

5.21.1 Descrição Detalhada

protótipos para _matriz.c

Definição no arquivo _matriz.h.

5.21.2 Variáveis

5.21.2.1 unsigned long int Matriz[256]

Valor Inicial:

```
=
{
    0,0,1,2,3,3,4,5,6,7,
    7,8,9,10,10,11,12,13,14,14,
    15,16,17,18,18,19,20,21,21,22,
    23,24,25,25,26,27,28,29,29,30,
    31,32,32,33,34,35,36,36,37,38,
    39,40,40,41,42,43,43,44,45,46,
    47,47,48,49,50,50,51,52,53,54,
    54,55,56,57,58,58,59,60,61,61,
    62,63,64,65,65,66,67,68,69,69,
    70,71,72,72,73,74,75,76,76,77,
    78,79,80,80,81,82,83,83,84,85,
    86,87,87,88,89,90,90,91,92,93,
    94,94,95,96,97,98,98,99,100,101,
    101,102,103,104,105,105,106,107,108,109,
    109,110,111,112,112,113,114,115,116,116,
    117,118,119,120,120,121,122,123,123,124,
    125,126,127,127,128,129,130,130,131,132,
    133,134,134,135,136,137,138,138,139,140,
    141,141,142,143,144,145,145,146,147,148,
    149,149,150,151,152,152,153,154,155,156,
    156,157,158,159,160,160,161,162,163,163,
    164,165,166,167,167,168,169,170,170,171,
    172,173,174,174,175,176,177,178,178,179,
    180,181,181,182,183,184,185,185,186,187,
    188,189,189,190,191,192,192,193,194,195,
    196,196,197,198,199,200
}
```

Matriz que converte o sinal do conversor AD em valor de corrente onde 200 = 2,00A

Definição na linha 7 do arquivo _matriz.h.

Referenciado por main().

5.22 _matriz.h

```
00001
00007 unsigned long int Matriz [256] =
00008 {
00009     0,0,1,2,3,3,4,5,6,7,
00010     7,8,9,10,10,11,12,13,14,14,
00011     15,16,17,18,18,19,20,21,21,22,
00012     23,24,25,25,26,27,28,29,29,30,
00013     31,32,32,33,34,35,36,36,37,38,
00014     39,40,40,41,42,43,43,44,45,46,
00015     47,47,48,49,50,50,51,52,53,54,
00016     54,55,56,57,58,58,59,60,61,61,
00017     62,63,64,65,65,66,67,68,69,69,
00018     70,71,72,72,73,74,75,76,76,77,
00019     78,79,80,80,81,82,83,83,84,85,
00020     86,87,87,88,89,90,90,91,92,93,
00021     94,94,95,96,97,98,98,99,100,101,
00022     101,102,103,104,105,105,106,107,108,109,
```

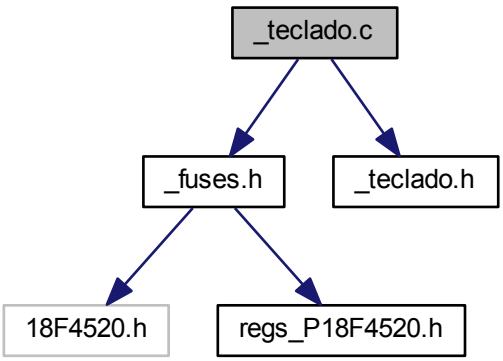
```
00023     109,110,111,112,112,113,114,115,116,116,
00024     117,118,119,120,120,121,122,123,123,124,
00025     125,126,127,127,128,129,130,130,131,132,
00026     133,134,134,135,136,137,138,138,139,140,
00027     141,141,142,143,144,145,145,146,147,148,
00028     149,149,150,151,152,152,153,154,155,156,
00029     156,157,158,159,160,160,161,162,163,163,
00030     164,165,166,167,167,168,169,170,170,171,
00031     172,173,174,174,175,176,177,178,178,179,
00032     180,181,181,182,183,184,185,185,186,187,
00033     188,189,189,190,191,192,192,193,194,195,
00034     196,196,197,198,199,200
00035 };
```

5.23 Referência do Arquivo _teclado.c

Comandos de leitura de teclado 16 teclas.

```
#include "_fuses.h"
#include "_teclado.h"
```

Gráfico de dependência de inclusões para _teclado.c:



5.23.1 Teclado

Lista de Figuras

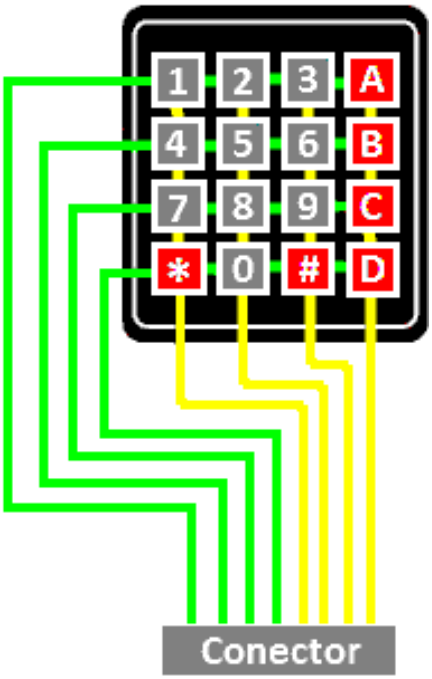


Figura 5.2: Teclado

- #define KB_IN PORTC
- #define KB_OUT LATD
- #define KB_Coluna_Nao 0x0F
- #define KB_Coluna_0 0b00001110
- #define KB_Coluna_1 0b00001101
- #define KB_Coluna_2 0b00001011
- #define KB_Coluna_3 0b00000111
- #define KB_linha_Nao 0x0F
- #define KB_linha_0 0b00001110
- #define KB_linha_1 0b00001101
- #define KB_linha_2 0b00001011
- #define KB_linha_3 0b00000111
- unsigned char KB_UltimaLida = '\0'
- short KB_Dado_Novo = FALSE
- void KB_tris (void)

  Configura Portas para entrada ou saida.
- void KB_Reset (void)

  Sinal inicial para varrer as teclas.
- void KB_trata_teclado (void)

  executa a varredura do teclado matricial
- char KB_Tecla_Nova (void)

  retorna a última tecla acionada uma vez

5.23.2 Descrição Detalhada

Comandos de leitura de teclado 16 teclas.

Definição no arquivo _teclado.c.

5.23.3 Definições e macros

5.23.3.1 #define KB_Coluna_0 0b00001110

Última tecla em ASCII, tipo One Shot

Definição na linha 25 do arquivo _teclado.c.

Referenciado por KB_trata_teclado().

5.23.3.2 #define KB_Coluna_1 0b00001101

Última tecla em ASCII, tipo One Shot

Definição na linha 26 do arquivo _teclado.c.

Referenciado por KB_trata_teclado().

5.23.3.3 #define KB_Coluna_2 0b00001011

Última tecla em ASCII, tipo One Shot

Definição na linha 27 do arquivo _teclado.c.

Referenciado por KB_trata_teclado().

5.23.3.4 #define KB_Coluna_3 0b00000111

Última tecla em ASCII, tipo One Shot

Definição na linha 28 do arquivo _teclado.c.

Referenciado por KB_trata_teclado().

5.23.3.5 #define KB_Coluna_Nao 0x0F

Última tecla em ASCII, tipo One Shot

Definição na linha 24 do arquivo _teclado.c.

Referenciado por KB_Reset() e KB_trata_teclado().

5.23.3.6 #define KB_IN PORTC

Última tecla em ASCII, tipo One Shot

Definição na linha 21 do arquivo _teclado.c.

Referenciado por KB_trata_teclado().

5.23.3.7 #define KB_linha_0 0b00001110

Última tecla em ASCII, tipo One Shot

Definição na linha 31 do arquivo _teclado.c.

Referenciado por KB_trata_teclado().

5.23.3.8 #define KB_linha_1 0b00001101

Última tecla em ASCII, tipo One Shot

Definição na linha 32 do arquivo _teclado.c.

Referenciado por KB_trata_teclado().

5.23.3.9 #define KB_linha_2 0b00001011

Última tecla em ASCII, tipo One Shot

Definição na linha 33 do arquivo _teclado.c.

Referenciado por KB_trata_teclado().

5.23.3.10 #define KB_linha_3 0b00000111

Última tecla em ASCII, tipo One Shot

Definição na linha 34 do arquivo _teclado.c.

Referenciado por KB_trata_teclado().

5.23.3.11 #define KB_linha_Nao 0x0F

Última tecla em ASCII, tipo One Shot

Definição na linha 30 do arquivo _teclado.c.

Referenciado por KB_trata_teclado().

5.23.3.12 #define KB_OUT LATD

Última tecla em ASCII, tipo One Shot

Definição na linha 22 do arquivo _teclado.c.

Referenciado por KB_Reset() e KB_trata_teclado().

5.23.4 Funções

5.23.4.1 char KB_Tecla_Nova (void)

retorna a última tecla acionada uma vez

Retorna

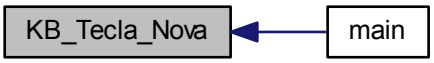
Código ASCII

Definição na linha 154 do arquivo _teclado.c.

Referenciado por main().

```
00154                           {
00155     if (KB_Dado_Novo) {
00156         KB_Dado_Novo = FALSE;
00157         return KB_UltimaLida;
00158     } else {
00159         return KB_Tecla_Nenhuma;
00160     }
00161 }
```

Este é o diagrama das funções que utilizam esta função:



5.23.5 Variáveis

5.23.5.1 short KB_Dado_Novo = FALSE

Sinal tipo One Shot

Definição na linha 38 do arquivo _teclado.c.

Referenciado por KB_Tecla_Nova() e KB_trata_teclado().

5.23.5.2 unsigned char KB_UltimaLida = '\0'

Última tecla em ASCII, tipo One Shot

Definição na linha 37 do arquivo _teclado.c.

Referenciado por KB_Tecla_Nova() e KB_trata_teclado().

5.24 _teclado.c

```
00001
00016 // Bibliotecas
00017 #include    "_fuses.h"
00018 #include    "_teclado.h"
00019
00020 // Definicoes
00021 #define KB_IN PORTC
00022 #define KB_OUT LATD
00023
00024 #define KB_Coluna_Nao 0x0F
00025 #define KB_Coluna_0 0b00001110
00026 #define KB_Coluna_1 0b00001101
00027 #define KB_Coluna_2 0b00001011
00028 #define KB_Coluna_3 0b00000111
00029
00030 #define KB_linha_Nao    0x0F
00031 #define KB_linha_0  0b00001110
00032 #define KB_linha_1  0b00001101
00033 #define KB_linha_2  0b00001011
00034 #define KB_linha_3  0b00000111
00035
00036 // Variaveis
00037 unsigned char KB_UltimaLida = '\0';
00038 short KB_Dado_Novo = FALSE;
00044 void KB_tris(void) {
00045     // Hex 0x1F
00046     // Binary 0b010010
00047
00048     TRISC = TRISC | 0b00011110; // forca entradas / RC0 é usada pelo TIMER_1
00049     TRISD = TRISD & 0b11110000; // forca saidas
00050 }
00051
00055 void KB_Reset(void) {
00056     KB_OUT = KB_OUT | KB_Coluna_Nao;
00057 }
00058
00062 void KB_trata_teclado(void) {
```

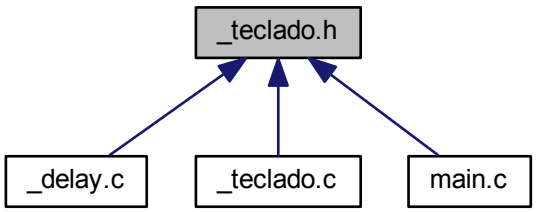
```
00063     unsigned char coluna_mascarado;
00064     unsigned char linha_mascarado;
00065     unsigned char tecla_apertada = KB_Tecla_Nenhuma;
00066     static char KB_PassoDedo = 0;
00067     static char bounce; // leitura de 4 colunas vazias
00068 
00069     // 1o sempre acertar o momento da proxima varredura
00070     KB_TempoVarredura = 50;
00071 
00072     // remocao de sinais inconvenientes atraves de mascara e rotacao
00073     coluna_mascarado = KB_OUT & 0x0F;
00074     linha_mascarado = KB_IN;
00075     linha_mascarado = linha_mascarado >> 1;
00076     linha_mascarado = linha_mascarado & 0x0F;
00077 
00078     switch (linha_mascarado) {
00079         case KB_linha_0:
00080             if (KB_Coluna_0 == coluna_mascarado) tecla_apertada =
      KB_Tecla_A;
00081             if (KB_Coluna_1 == coluna_mascarado) tecla_apertada =
      KB_Tecla_B;
00082             if (KB_Coluna_2 == coluna_mascarado) tecla_apertada =
      KB_Tecla_C;
00083             if (KB_Coluna_3 == coluna_mascarado) tecla_apertada =
      KB_Tecla_D;
00084             break;
00085         case KB_linha_1:
00086             if (KB_Coluna_0 == coluna_mascarado) tecla_apertada =
      KB_Tecla_3;
00087             if (KB_Coluna_1 == coluna_mascarado) tecla_apertada =
      KB_Tecla_6;
00088             if (KB_Coluna_2 == coluna_mascarado) tecla_apertada =
      KB_Tecla_9;
00089             if (KB_Coluna_3 == coluna_mascarado) tecla_apertada =
      KB_Tecla_Velha;
00090             break;
00091         case KB_linha_2:
00092             if (KB_Coluna_0 == coluna_mascarado) tecla_apertada =
      KB_Tecla_2;
00093             if (KB_Coluna_1 == coluna_mascarado) tecla_apertada =
      KB_Tecla_5;
00094             if (KB_Coluna_2 == coluna_mascarado) tecla_apertada =
      KB_Tecla_8;
00095             if (KB_Coluna_3 == coluna_mascarado) tecla_apertada =
      KB_Tecla_0;
00096             break;
00097         case KB_linha_3:
00098             if (KB_Coluna_0 == coluna_mascarado) tecla_apertada =
      KB_Tecla_1;
00099             if (KB_Coluna_1 == coluna_mascarado) tecla_apertada =
      KB_Tecla_4;
00100             if (KB_Coluna_2 == coluna_mascarado) tecla_apertada =
      KB_Tecla_7;
00101             if (KB_Coluna_3 == coluna_mascarado) tecla_apertada =
      KB_Tecla_Estrela;
00102             break;
00103         case KB_linha_Nao:
00104             tecla_apertada = KB_Tecla_Nenhuma;
00105             break;
00106     }
00107 
00108     if (tecla_apertada == KB_Tecla_Nenhuma) {
00109         if (bounce<255) bounce++;
00110     } else {
00111         bounce = 0;
00112     }
00113 
00114     // Houve tecla valida?
00115     switch (KB_PassoDedo){
00116     case 0:
00117         if (tecla_apertada != KB_Tecla_Nenhuma){
00118             KB_UltimaLida = tecla_apertada;
00119             KB_Dado_Novo = TRUE;
00120             KB_PassoDedo = 1;
00121         }
00122         break;
00123     case 1:
00124         if (bounce > 5){
00125             KB_PassoDedo = 0;
00126         }
00127         break;
00128     }
00129 
00130     //proxima coluna
00131     switch (coluna_mascarado) {
00132         case KB_Coluna_0:
00133             KB_OUT = (KB_OUT | 0x0F) & (0xF0 | KB_Coluna_1);
```

```
00134             break;
00135         case KB_Coluna_1:
00136             KB_OUT = (KB_OUT | 0x0F) & (0xF0 | KB_Coluna_2);
00137             break;
00138         case KB_Coluna_2:
00139             KB_OUT = (KB_OUT | 0x0F) & (0xF0 | KB_Coluna_3);
00140             break;
00141         case KB_Coluna_3:
00142             KB_OUT = (KB_OUT | 0x0F) & (0xF0 | KB_Coluna_0);
00143             break;
00144         case KB_Coluna_Nao:
00145             KB_OUT = (KB_OUT | 0x0F) & (0xF0 | KB_Coluna_0);
00146             break;
00147     }
00148 }
00149
00154 char KB_Tecla_Nova(void) {
00155     if (KB_Dado_Novo) {
00156         KB_Dado_Novo = FALSE;
00157         return KB_UltimaLida;
00158     } else {
00159         return KB_Tecla_Nenhuma;
00160     }
00161 }
```

5.25 Referência do Arquivo _teclado.h

protótipos para _teclado.c

Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com este arquivo:



Definições e Macros

- #define KB_Tecla_Nenhuma 0
- #define KB_Tecla_0 48
- #define KB_Tecla_1 49
- #define KB_Tecla_2 50
- #define KB_Tecla_3 51
- #define KB_Tecla_4 52
- #define KB_Tecla_5 53
- #define KB_Tecla_6 54
- #define KB_Tecla_7 55
- #define KB_Tecla_8 56
- #define KB_Tecla_9 57
- #define KB_Tecla_A 65
- #define KB_Tecla_B 66
- #define KB_Tecla_C 67
- #define KB_Tecla_D 68

- #define KB_Tecla_Estrela 42
- #define KB_Tecla_Velha 35

Funções

- void KB_tris (void)
  Configura Portas para entrada ou saida.
- void KB_Reset (void)
  Sinal inicial para varrer as teclas.
- void KB_trata_teclado (void)
  executa a varredura do teclado matricial
- char KB_Tecla_Nova (void)
  retorna a última tecla acionada uma vez

Variáveis

- unsigned long int KB_TempoVarredura = 500

5.25.1 Descrição Detalhada

protótipos para _teclado.c

Definição no arquivo _teclado.h.

5.25.2 Definições e macros

5.25.2.1 #define KB_Tecla_0 48

Códigos ASCII

Definição na linha 19 do arquivo _teclado.h.

Referenciado por KB_trata_teclado().

5.25.2.2 #define KB_Tecla_1 49

Códigos ASCII

Definição na linha 20 do arquivo _teclado.h.

Referenciado por KB_trata_teclado().

5.25.2.3 #define KB_Tecla_2 50

Códigos ASCII

Definição na linha 21 do arquivo _teclado.h.

Referenciado por KB_trata_teclado().

5.25.2.4 #define KB_Tecla_3 51

Códigos ASCII

Definição na linha 22 do arquivo _teclado.h.

Referenciado por KB_trata_teclado().

5.25.2.5 #define KB_Tecla_4 52

Códigos ASCII

Definição na linha 23 do arquivo _teclado.h.

Referenciado por KB_trata_teclado().

5.25.2.6 #define KB_Tecla_5 53

Códigos ASCII

Definição na linha 24 do arquivo _teclado.h.

Referenciado por KB_trata_teclado().

5.25.2.7 #define KB_Tecla_6 54

Códigos ASCII

Definição na linha 25 do arquivo _teclado.h.

Referenciado por KB_trata_teclado().

5.25.2.8 #define KB_Tecla_7 55

Códigos ASCII

Definição na linha 26 do arquivo _teclado.h.

Referenciado por KB_trata_teclado().

5.25.2.9 #define KB_Tecla_8 56

Códigos ASCII

Definição na linha 27 do arquivo _teclado.h.

Referenciado por KB_trata_teclado().

5.25.2.10 #define KB_Tecla_9 57

Códigos ASCII

Definição na linha 28 do arquivo _teclado.h.

Referenciado por KB_trata_teclado().

5.25.2.11 #define KB_Tecla_A 65

Códigos ASCII

Definição na linha 30 do arquivo _teclado.h.

Referenciado por KB_trata_teclado().

5.25.2.12 #define KB_Tecla_B 66

Códigos ASCII

Definição na linha 31 do arquivo _teclado.h.

Referenciado por KB_trata_teclado().

5.25.2.13 #define KB_Tecla_C 67

Códigos ASCII

Definição na linha 32 do arquivo _teclado.h.

Referenciado por KB_trata_teclado().

5.25.2.14 #define KB_Tecla_D 68

Códigos ASCII

Definição na linha 33 do arquivo _teclado.h.

Referenciado por KB_trata_teclado().

5.25.2.15 #define KB_Tecla_Estrela 42

Códigos ASCII

Definição na linha 35 do arquivo _teclado.h.

Referenciado por KB_trata_teclado().

5.25.2.16 #define KB_Tecla_Nenhuma 0

Códigos ASCII

Definição na linha 18 do arquivo _teclado.h.

Referenciado por KB_Tecla_Nova(), KB_trata_teclado() e main().

5.25.2.17 #define KB_Tecla_Velha 35

Códigos ASCII

Definição na linha 36 do arquivo _teclado.h.

Referenciado por KB_trata_teclado().

5.25.3 Funções

5.25.3.1 char KB_Tecla_Nova (void)

retorna a última tecla acionada uma vez

Retorna

Código ASCII

Definição na linha 154 do arquivo _teclado.c.

Referenciado por main().

```
00154                           {
00155     if (KB_Dado_Novo) {
00156         KB_Dado_Novo = FALSE;
00157         return KB_UltimaLida;
00158     } else {
```

```
00159         return KB_Tecla_Nenhuma;
00160     }
00161 }
```

Este é o diagrama das funções que utilizam esta função:

5.25.4 Variáveis

5.25.4.1 unsigned long int KB_TempoVarredura = 500

VALOR GRANDE apenas para que demore a primeira varredura do teclado

Definição na linha 6 do arquivo _teclado.h.

Referenciado por KB_trata_teclado(), main() e Trata_Timer0().

5.26 _teclado.h

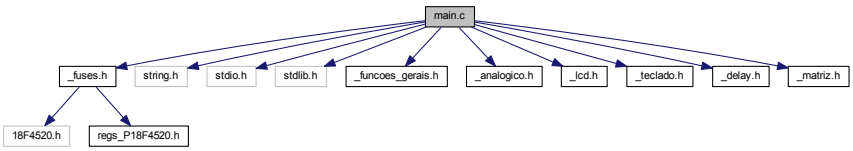
```
00001
00006 unsigned long int KB_TempoVarredura = 500;
00008 // prototipos
00009 void KB_tris(void);
00010 void KB_Reset(void);
00011 void KB_trata_teclado(void);
00012 char KB_Tecla_Nova(void);
00013
00018 #define KB_Tecla_Nenhuma 0
00019 #define KB_Tecla_0 48
00020 #define KB_Tecla_1 49
00021 #define KB_Tecla_2 50
00022 #define KB_Tecla_3 51
00023 #define KB_Tecla_4 52
00024 #define KB_Tecla_5 53
00025 #define KB_Tecla_6 54
00026 #define KB_Tecla_7 55
00027 #define KB_Tecla_8 56
00028 #define KB_Tecla_9 57
00029
00030 #define KB_Tecla_A 65
00031 #define KB_Tecla_B 66
00032 #define KB_Tecla_C 67
00033 #define KB_Tecla_D 68
00034
00035 #define KB_Tecla_Estrela 42
00036 #define KB_Tecla_Velha 35
00037
```

5.27 Referência do Arquivo main.c

Este arquivo contem a lista de desenvolvedores, rotinas de configuração e a máquina de estado principal.

```
#include "_fuses.h"
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include "_funcoes_gerais.h"
#include "_analogico.h"
#include "_lcd.h"
#include "_teclado.h"
#include "_delay.h"
#include "_matriz.h"
```

Gráfico de dependência de inclusões para main.c:



Definições e Macros

- #define POS_TH 2
- #define POS_TL 2
- #define POS_AD 11

Funções

5.27.1 Display

Lista de Figuras

Figura 5.3: Display

- void InitHW (void)

  Funcao de inicializacao com as configuracoes do uControlador.
- void InitSystem (void)

  Inicializa circuitos externos ao microcontrolador.

- char ValidarEntrada (unsigned long int A, unsigned long int MAX, unsigned long int MIN)

  Verifica se um valor está dentro de limites.
- void main (void)

  Funcao principal do programa com aplicacao das funcoes auxiliares e de tratamento gerando as interrupcoes para o gerenciamento em si
  .

Variáveis

- unsigned long int TH_Lim_SUP = 500
- unsigned long int TH_Lim_INF = 30
- unsigned long int TL_Lim_SUP = 1000
- unsigned long int TL_Lim_INF = 100
- unsigned char LCD_Mascara_L1 [1+LCD_colunas] = "TH ##.us I=#.##A"
- unsigned char LCD_Mascara_L2 [1+LCD_colunas] = "TL ##.ms "
- unsigned char LCD_BemVindo_L1 [1+LCD_colunas] = " FATEC 1o 2013 "
- unsigned char LCD_BemVindo_L2 [1+LCD_colunas] = " Carga e Partida"

5.27.2 Descrição Detalhada

Este arquivo contem a lista de desenvolvedores, rotinas de configuração e a máquina de estado principal.

Data

1o semestre 2013

Definição no arquivo main.c.

5.27.3 Definições e macros

5.27.3.1 #define POS_AD 11

Define posição no display

Definição na linha 64 do arquivo main.c.

Referenciado por main().

5.27.3.2 #define POS_TH 2

Define posição no display

Definição na linha 62 do arquivo main.c.

Referenciado por main().

5.27.3.3 #define POS_TL 2

Define posição no display

Definição na linha 63 do arquivo main.c.

Referenciado por main().

5.27.4 Funções

5.27.4.1 main (void)

Funcao principal do programa com aplicacao das funcoes auxiliares e de tratamento gerando as interrupcoes para o gerenciamento em si

.

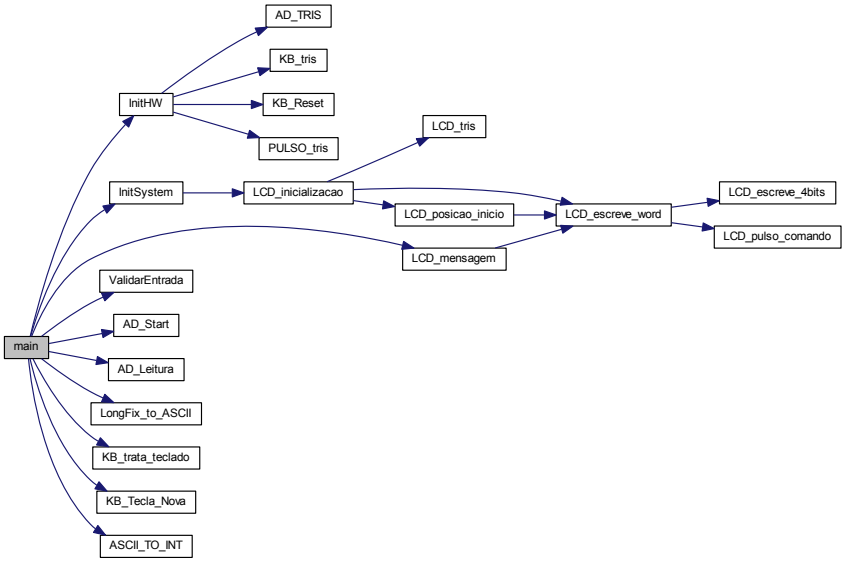
Definição na linha 98 do arquivo main.c.

```
00098                {
00099     static unsigned long int Tempo_montar;
00100     static unsigned char AD_ValorZ1;
00101     unsigned char AD_ValorAtual;
00102     unsigned char tecla = 0;
00103 
00104     unsigned char msg[1 + LCD_colunas];
00105     char dummy;
00106     unsigned long int dummyUL;
00107 
00108     unsigned char POS_TH_L = LCD_L1;
00109     unsigned char POS_TL_L = LCD_L2;
00110 
00111     char PassoTeclado = 0;
00112 
00113     InitHW();
00114     InitSystem();
00115 
00116     //Boas Vindas:
00117     strcpy(msg, LCD_BemVindo_L1);
00118     LCD_mensagem(LCD_L1, msg);
00119     strcpy(msg, LCD_BemVindo_L2);
00120     LCD_mensagem(LCD_L2, msg);
00121     delay_ms(3000);
00122     // Mascara:
00123     strcpy(msg, LCD_Mascara_L1);
00124     LCD_mensagem(LCD_L1, msg);
00125     strcpy(msg, LCD_Mascara_L2);
00126     LCD_mensagem(LCD_L2, msg);
00127 
00128     // Limites para TH e TL
00129     if (!ValidarEntrada(TH, TH_Lim_SUP, TH_Lim_INF))
00130         TH = TH_Lim_INF;
00131     if (!ValidarEntrada(TL, TL_Lim_SUP, TL_Lim_INF))
00132         TL = TL_Lim_INF;
00133 
00134     // Atualizar TH e TL
00135     dummy = sprintf(msg, "%04lu\0", TH);
00136     LCD_mensagem(POS_TH + POS_TH_L, msg);
00137 
00138     dummy = sprintf(msg, "%04lu\0", TL);
00139     LCD_mensagem(POS_TL + POS_TL_L, msg);
00140     delay_ms(100);
00141 
00142     //Partir a analógica
00143     AD_ValorAtual = 100;
00144     AD_ValorZ1 = 1;
00145     enable_interrupts(GLOBAL);
00146     AD_Start(0);
00147 
00148     while (TRUE) {
00149         // Pulso
00150         AD_ValorAtual = AD_Leitura();
00151         if (AD_ValorZ1 != AD_ValorAtual){
00152             AD_ValorZ1 = AD_ValorAtual;
00153             dummyUL = Matriz[AD_ValorAtual];
00154             LongFix_to_ASCII(dummyUL, 2, msg);
00155             LCD_mensagem(POS_AD + LCD_L1, msg);
00156         }
00157 
00158         // Tratamento de teclado
00159         // Atualizacao de TH TL
00160         if (!KB_TempoVarredura) KB_trata_teclado();
00161         tecla = KB_Tecla_Nova();
00162         if (tecla != KB_Tecla_Nenhuma) {
00163             //para acompanhamento
00164             dummy = sprintf(msg, "%c\0", tecla);
00165             LCD_mensagem(15 + LCD_L2, msg);
00166 
00167             switch (PassoTeclado){
00168             case 0:
00169                 if ('A' == tecla){
00170                     PassoTeclado = 1;
```

```
00171                   msg = "____";
00172                   LCD_mensagem(POS_TH + POS_TH_L, msg);
00173               }
00174               if ('B' == tecla) {
00175                   PassoTeclado = 101;
00176                   msg = "____";
00177                   LCD_mensagem(POS_TL + POS_TL_L, msg);
00178               }
00179               break;
00180
00181           // TEMPO EM ALTO
00182           case 1:
00183               if ((tecla >= '0') && (tecla <= '9')){
00184                   PassoTeclado = 2;
00185                   dummyUL = ASCII_TO_INT (tecla);
00186                   Tempo_montar =  dummyUL;
00187                   dummy = sprintf(msg, "___%01lu\0", Tempo_montar);
00188                   LCD_mensagem(POS_TH + POS_TH_L, msg);
00189               }
00190               break;
00191           case 2:
00192               if ((tecla >= '0') && (tecla <= '9')){
00193                   PassoTeclado = 3;
00194                   dummyUL = ASCII_TO_INT (tecla);
00195                   Tempo_montar = 10ul * Tempo_montar + dummyUL;
00196                   dummy = sprintf(msg, "__%02lu\0", Tempo_montar);
00197                   LCD_mensagem(POS_TH + POS_TH_L, msg);
00198               }
00199               break;
00200           case 3:
00201               if ((tecla >= '0') && (tecla <= '9')){
00202                   PassoTeclado = 4;
00203                   dummyUL = ASCII_TO_INT (tecla);
00204                   Tempo_montar = 10ul * Tempo_montar + dummyUL;
00205                   dummy = sprintf(msg, "_%03lu\0", Tempo_montar);
00206                   LCD_mensagem(POS_TH + POS_TH_L, msg);
00207               }
00208               break;
00209           case 4:
00210               if ((tecla >= '0') && (tecla <= '9')){
00211                   PassoTeclado = 0; //retornando por ser a UNIDADE
00212                   dummyUL = ASCII_TO_INT (tecla);
00213                   Tempo_montar = 10ul * Tempo_montar + dummyUL;
00214                   // Atualizar
00215                   if (ValidarEntrada(Tempo_montar, TH_Lim_SUP,
    TH_Lim_INF))
00216                       TH = Tempo_montar;
00217                   dummy = sprintf(msg, "%04lu\0", TH);
00218                   LCD_mensagem(POS_TH + POS_TH_L, msg);
00219                   PassoTeclado = 0;
00220               }
00221               break;
00222
00223           // TEMPO EM Baixo
00224           case 101:
00225               if ((tecla >= '0') && (tecla <= '9')){
00226                   PassoTeclado = 102;
00227                   dummyUL = ASCII_TO_INT (tecla);
00228                   Tempo_montar =  dummyUL;
00229                   dummy = sprintf(msg, "___%01lu\0", Tempo_montar);
00230                   LCD_mensagem(POS_TL + POS_TL_L, msg);
00231               }
00232               break;
00233           case 102:
00234               if ((tecla >= '0') && (tecla <= '9')){
00235                   PassoTeclado = 103;
00236                   dummyUL = ASCII_TO_INT (tecla);
00237                   Tempo_montar = 10ul * Tempo_montar + dummyUL;
00238                   dummy = sprintf(msg, "__%02lu\0", Tempo_montar);
00239                   LCD_mensagem(POS_TL + POS_TL_L, msg);
00240               }
00241               break;
00242           case 103:
00243               if ((tecla >= '0') && (tecla <= '9')){
00244                   PassoTeclado = 104;
00245                   dummyUL = ASCII_TO_INT (tecla);
00246                   Tempo_montar = 10ul * Tempo_montar + dummyUL;
00247                   dummy = sprintf(msg, "_%03lu\0", Tempo_montar);
00248                   LCD_mensagem(POS_TL + POS_TL_L, msg);
00249               }
00250               break;
00251           case 104:
00252               if ((tecla >= '0') && (tecla <= '9')){
00253                   PassoTeclado = 0; //retornando por ser a UNIDADE
00254                   dummyUL = ASCII_TO_INT (tecla);
00255                   Tempo_montar = 10ul * Tempo_montar + dummyUL;
00256                   // Atualizar
```

```
00257                     if (ValidarEntrada(Tempo_montar, TL_Lim_SUP,
      TL_Lim_INF))
00258                         TL = Tempo_montar;
00259                     dummy = sprintf(msg, "%04lu\0", TL);
00260                     LCD_mensagem(POS_TL + POS_TL_L, msg);
00261                     PassoTeclado = 0;
00262                 }
00263                 break;
00264             }
00265         }
00266     }
00267 }
```

Este é o diagrama das funções utilizadas por esta função:



5.27.4.2 char ValidarEntrada (unsigned long int *A*, unsigned long int *MAX*, unsigned long int *MIN*)

Verifica se um valor está dentro de limites.

Parâmetros

in	*A*	Valor a ser comparado
in	*MAX*	Limite superior
in	*MIN*	Limite inferior

Retorna

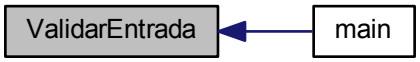
TRUE / FALSE

Definição na linha 357 do arquivo main.c.

Referenciado por main().

```
00357                                                                   {
00358     if (A > MAX) return false;
00359     if (A < MIN) return false;
00360     return true;
00361 }
```

Este é o diagrama das funções que utilizam esta função:



5.27.5 Variáveis

5.27.5.1 unsigned char LCD_BemVindo_L1[1+LCD_colunas] = " FATEC 1o 2013 "

Mensagem no Display

Definição na linha 81 do arquivo main.c.

Referenciado por main().

5.27.5.2 unsigned char LCD_BemVindo_L2[1+LCD_colunas] = " Carga e Partida"

Mensagem no Display

Definição na linha 82 do arquivo main.c.

Referenciado por main().

5.27.5.3 unsigned char LCD_Mascara_L1[1+LCD_colunas] = "TH ##.us I=#.##A"

Mascara no Display

Definição na linha 79 do arquivo main.c.

Referenciado por main().

5.27.5.4 unsigned char LCD_Mascara_L2[1+LCD_colunas] = "TL ##.ms "

Mascara no Display

Definição na linha 80 do arquivo main.c.

Referenciado por main().

5.27.5.5 unsigned long int TH_Lim_INF = 30

Limite inferior em us

Definição na linha 69 do arquivo main.c.

Referenciado por main().

5.27.5.6 unsigned long int TH_Lim_SUP = 500

Limite superior em us

Definição na linha 68 do arquivo main.c.

Referenciado por main().

5.27.5.7 unsigned long int TL_Lim_INF = 100

Limite inferior em ms

Definição na linha 71 do arquivo main.c.

Referenciado por main().

5.27.5.8 unsigned long int TL_Lim_SUP = 1000

Limite superior em ms

Definição na linha 70 do arquivo main.c.

Referenciado por main().

5.28 main.c

```
00001
00046 // Bibliotecas
00047 #include    "_fuses.h"
00048 #include    <string.h>
00049 #include    <stdio.h>
00050 #include    <stdlib.h>
00051 #include    "_funcoes_gerais.h"
00052 #include    "_analogico.h"
00053 #include    "_lcd.h"
00054 #include    "_teclado.h"
00055 #include    "_delay.h"
00056 #include    "_matriz.h"
00057
00062 #define POS_TH 2
00063 #define POS_TL 2
00064 #define POS_AD 11
00065
00067 //  Constantes
00068 unsigned long int TH_Lim_SUP = 500;
00069 unsigned long int TH_Lim_INF = 30;
00070 unsigned long int TL_Lim_SUP = 1000;
00071 unsigned long int TL_Lim_INF = 100;
00073 // prototipos
00074 void InitHW(void);
00075 void InitSystem(void);
00076 char ValidarEntrada( unsigned long int, unsigned long int, unsigned long int);
00077
00078 // Variaveis
00079 unsigned char LCD_Mascara_L1 [1 + LCD_colunas] = "TH ##.us I=#.##A";
00080 unsigned char LCD_Mascara_L2 [1 + LCD_colunas] = "TL ##.ms        ";
00081 unsigned char LCD_BemVindo_L1[1 + LCD_colunas] = " FATEC 1o 2013  ";
00082 unsigned char LCD_BemVindo_L2[1 + LCD_colunas] = " Carga e Partida";
00098 void main(void) {
00099     static unsigned long int Tempo_montar;
00100     static unsigned char AD_ValorZ1;
00101     unsigned char AD_ValorAtual;
00102     unsigned char tecla = 0;
00103
00104     unsigned char msg[1 + LCD_colunas];
00105     char dummy;
00106     unsigned long int dummyUL;
00107
00108     unsigned char POS_TH_L = LCD_L1;
00109     unsigned char POS_TL_L = LCD_L2;
00110
00111     char PassoTeclado = 0;
00112
00113     InitHW();
00114     InitSystem();
00115
00116     //Boas Vindas:
00117     strcpy(msg, LCD_BemVindo_L1);
00118     LCD_mensagem(LCD_L1, msg);
00119     strcpy(msg, LCD_BemVindo_L2);
00120     LCD_mensagem(LCD_L2, msg);
00121     delay_ms(3000);
```

```
00122     // Mascara:
00123     strcpy(msg, LCD_Mascara_L1);
00124     LCD_mensagem(LCD_L1, msg);
00125     strcpy(msg, LCD_Mascara_L2);
00126     LCD_mensagem(LCD_L2, msg);
00127
00128     // Limites para TH e TL
00129     if (!ValidarEntrada(TH, TH_Lim_SUP, TH_Lim_INF))
00130         TH = TH_Lim_INF;
00131     if (!ValidarEntrada(TL, TL_Lim_SUP, TL_Lim_INF))
00132         TL = TL_Lim_INF;
00133
00134     // Atualizar TH e TL
00135     dummy = sprintf(msg, "%04lu\0", TH);
00136     LCD_mensagem(POS_TH + POS_TH_L, msg);
00137
00138     dummy = sprintf(msg, "%04lu\0", TL);
00139     LCD_mensagem(POS_TL + POS_TL_L, msg);
00140     delay_ms(100);
00141
00142     //Partir a analógica
00143     AD_ValorAtual = 100;
00144     AD_ValorZ1 = 1;
00145     enable_interrupts(GLOBAL);
00146     AD_Start(0);
00147
00148     while (TRUE) {
00149         // Pulso
00150         AD_ValorAtual = AD_Leitura();
00151         if (AD_ValorZ1 != AD_ValorAtual){
00152             AD_ValorZ1 = AD_ValorAtual;
00153             dummyUL = Matriz[AD_ValorAtual];
00154             LongFix_to_ASCII(dummyUL, 2, msg);
00155             LCD_mensagem(POS_AD + LCD_L1, msg);
00156         }
00157
00158         // Tratamento de teclado
00159         // Atualizacao de TH TL
00160         if (!KB_TempoVarredura) KB_trata_teclado();
00161         tecla = KB_Tecla_Nova();
00162         if (tecla != KB_Tecla_Nenhuma) {
00163             //para acompanhamento
00164             dummy = sprintf(msg, "%c\0", tecla);
00165             LCD_mensagem(15 + LCD_L2, msg);
00166
00167             switch (PassoTeclado){
00168             case 0:
00169                 if ('A' == tecla){
00170                     PassoTeclado = 1;
00171                     msg = "____";
00172                     LCD_mensagem(POS_TH + POS_TH_L, msg);
00173                 }
00174                 if ('B' == tecla) {
00175                     PassoTeclado = 101;
00176                     msg = "____";
00177                     LCD_mensagem(POS_TL + POS_TL_L, msg);
00178                 }
00179                 break;
00180
00181             // TEMPO EM ALTO
00182             case 1:
00183                 if ((tecla >= '0') && (tecla <= '9')){
00184                     PassoTeclado = 2;
00185                     dummyUL = ASCII_TO_INT (tecla);
00186                     Tempo_montar =  dummyUL;
00187                     dummy = sprintf(msg, "___%01lu\0", Tempo_montar);
00188                     LCD_mensagem(POS_TH + POS_TH_L, msg);
00189                 }
00190                 break;
00191             case 2:
00192                 if ((tecla >= '0') && (tecla <= '9')){
00193                     PassoTeclado = 3;
00194                     dummyUL = ASCII_TO_INT (tecla);
00195                     Tempo_montar = 10ul * Tempo_montar + dummyUL;
00196                     dummy = sprintf(msg, "__%02lu\0", Tempo_montar);
00197                     LCD_mensagem(POS_TH + POS_TH_L, msg);
00198                 }
00199                 break;
00200             case 3:
00201                 if ((tecla >= '0') && (tecla <= '9')){
00202                     PassoTeclado = 4;
00203                     dummyUL = ASCII_TO_INT (tecla);
00204                     Tempo_montar = 10ul * Tempo_montar + dummyUL;
00205                     dummy = sprintf(msg, "_%03lu\0", Tempo_montar);
00206                     LCD_mensagem(POS_TH + POS_TH_L, msg);
00207                 }
00208                 break;
```

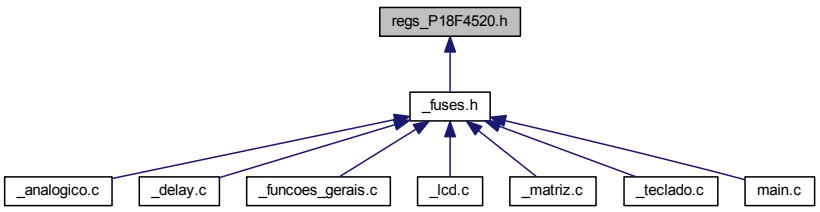
```
00209             case 4:
00210                 if ((tecla >= '0') && (tecla <= '9')){
00211                     PassoTeclado = 0; //retornando por ser a UNIDADE
00212                     dummyUL = ASCII_TO_INT (tecla);
00213                     Tempo_montar = 10ul * Tempo_montar + dummyUL;
00214                     // Atualizar
00215                     if (ValidarEntrada(Tempo_montar, TH_Lim_SUP,
      TH_Lim_INF))
00216                         TH = Tempo_montar;
00217                     dummy = sprintf(msg, "%04lu\0", TH);
00218                     LCD_mensagem(POS_TH + POS_TH_L, msg);
00219                     PassoTeclado = 0;
00220                 }
00221                 break;
00222
00223             // TEMPO EM Baixo
00224             case 101:
00225                 if ((tecla >= '0') && (tecla <= '9')){
00226                     PassoTeclado = 102;
00227                     dummyUL = ASCII_TO_INT (tecla);
00228                     Tempo_montar =  dummyUL;
00229                     dummy = sprintf(msg, "___%01lu\0", Tempo_montar);
00230                     LCD_mensagem(POS_TL + POS_TL_L, msg);
00231                 }
00232                 break;
00233             case 102:
00234                 if ((tecla >= '0') && (tecla <= '9')){
00235                     PassoTeclado = 103;
00236                     dummyUL = ASCII_TO_INT (tecla);
00237                     Tempo_montar = 10ul * Tempo_montar + dummyUL;
00238                     dummy = sprintf(msg, "__%02lu\0", Tempo_montar);
00239                     LCD_mensagem(POS_TL + POS_TL_L, msg);
00240                 }
00241                 break;
00242             case 103:
00243                 if ((tecla >= '0') && (tecla <= '9')){
00244                     PassoTeclado = 104;
00245                     dummyUL = ASCII_TO_INT (tecla);
00246                     Tempo_montar = 10ul * Tempo_montar + dummyUL;
00247                     dummy = sprintf(msg, "_%03lu\0", Tempo_montar);
00248                     LCD_mensagem(POS_TL + POS_TL_L, msg);
00249                 }
00250                 break;
00251             case 104:
00252                 if ((tecla >= '0') && (tecla <= '9')){
00253                     PassoTeclado = 0; //retornando por ser a UNIDADE
00254                     dummyUL = ASCII_TO_INT (tecla);
00255                     Tempo_montar = 10ul * Tempo_montar + dummyUL;
00256                     // Atualizar
00257                     if (ValidarEntrada(Tempo_montar, TL_Lim_SUP,
      TL_Lim_INF))
00258                         TL = Tempo_montar;
00259                     dummy = sprintf(msg, "%04lu\0", TL);
00260                     LCD_mensagem(POS_TL + POS_TL_L, msg);
00261                     PassoTeclado = 0;
00262                 }
00263                 break;
00264             }
00265         }
00266     }
00267 }
00268
00273 void InitHW(void) {
00274     // set all ports as input
00275     TRISA = 0XFF;
00276     TRISB = 0XFF;
00277     TRISC = 0XFF;
00278     TRISD = 0XFF;
00279     TRISE = 0XFF;
00280
00281     // desabilita interrupções
00282     disable_interrupts(GLOBAL);
00283     // desabilita interrupções individualmente
00284     disable_interrupts(int_timer0);
00285     disable_interrupts(int_timer1);
00286     disable_interrupts(int_timer2);
00287     disable_interrupts(int_timer3);
00288     disable_interrupts(int_ext);
00289     disable_interrupts(int_ext1);
00290     disable_interrupts(int_ext2);
00291     disable_interrupts(INT_AD);
00292     disable_interrupts(INT_CCP1);
00293     disable_interrupts(INT_CCP2);
00294     disable_interrupts(INT_COMP);
00295     // disable_interrupts(INT_PSP);
00296     disable_interrupts(INT_RB);
00297     disable_interrupts(INT_RDA);
```

```
00298     disable_interrupts(INT_RTCC);
00299     disable_interrupts(INT_SSP);
00300     disable_interrupts(INT_TBE);
00301     // desabilita temporizadores
00302     T1CON = 0x00;
00303     T2CON = 0x00;
00304     T3CON = 0x00;
00305     // AD Converter
00306     setup_adc(ADC_OFF);
00307     setup_adc_ports(NO_ANALOGS);
00308     setup_psp(PSP_DISABLED); // Parallel slave port
00309     setup_spi(FALSE); // SPI Disabled
00310
00311     setup_comparator(NC_NC_NC_NC); // Desliga comparadores internos
00312     setup_vref(FALSE); // Tensao de referencia  para o comparador
00313
00314     // Quero 8us que * 8 bits = 2048us
00315     // 16MHz/4 = 4MHz -> 1/4Mhz = 250ns
00316     // 8us / 250ns = 32
00317     setup_timer_0(RTCC_INTERNAL | RTCC_DIV_32 | RTCC_8_BIT);
00318     set_timer0(TMR_Prescaler_1ms);
00319     enable_interrupts(int_timer0);
00320
00321     // Perigo T1 toma controle da porta C /div 1 2 4 8
00322     // timer 1 : 16Mhz / 4 = 250ns
00323     // 250ns * 4 = 1us
00324     // a cada 10us
00325     setup_timer_1(T1_INTERNAL | T1_DIV_BY_8);
00326     set_timer1(65500);
00327     enable_interrupts(int_timer1);
00328
00329     //setup_timer_3(T3_DISABLED);
00330     // Perigo T3 toma controle da porta C | T3_DIV_BY_4
00331
00332     AD_TRIS();
00333     setup_adc(ADC_CLOCK_DIV_8);
00334     setup_adc_ports(AN0);
00335     enable_interrupts(int_AD);
00336
00337     KB_tris();
00338     KB_Reset();
00339     PULSO_tris();
00340 }
00341
00346 void InitSystem(void) {
00347     LCD_inicializacao(); // Funcao que inicializa o Display
00348 }
00349
00357 char ValidarEntrada(unsigned long int A, unsigned long int MAX, unsigned long int MIN){
00358     if (A > MAX) return false;
00359     if (A < MIN) return false;
00360     return true;
00361 }
```

5.29 Referência do Arquivo regs_P18F4520.h

definicoes dos registros, é necessario comentar / descomentar conforme a necessidade

Este grafo mostra quais arquivos estão direta ou indiretamente relacionados com este arquivo:



5.29.1 Descrição Detalhada

definicoes dos registros, é necessario comentar / descomentar conforme a necessidade

Data

1o semestre 2013

Definição no arquivo regs_P18F4520.h.

5.30 regs_P18F4520.h

```
00001
00011 #BYTE PORTA = 0x0F80
00012     #BIT RA0 = PORTA.0
00013     //  #BIT AN0 = PORTA.0
00014     #BIT RA1 = PORTA.1
00015     //  #BIT AN1 = PORTA.1
00016     #BIT RA2 = PORTA.2
00017     //  #BIT AN2 = PORTA.2
00018     //  #BIT VREFN = PORTA.2
00019     //  #BIT CVREF = PORTA.2
00020     #BIT RA3 = PORTA.3
00021     //  #BIT AN3 = PORTA.3
00022     //  #BIT VREFP = PORTA.3
00023     #BIT RA4 = PORTA.4
00024     //  #BIT T0CKI = PORTA.4
00025     //  #BIT C1OUT_PORTA = PORTA.4
00026     #BIT RA5 = PORTA.5
00027     //  #BIT AN4 = PORTA.5
00028     //  #BIT SS = PORTA.5
00029     //  #BIT NOT_SS = PORTA.5
00030     //  #BIT LVDIN = PORTA.5
00031     //  #BIT HLVDIN = PORTA.5
00032     //  #BIT C2OUT_PORTA = PORTA.5
00033     #BIT RA6 = PORTA.6
00034     //  #BIT OSC2 = PORTA.6
00035     //  #BIT CLKO = PORTA.6
00036     #BIT RA7 = PORTA.7
00037     //  #BIT OSC1 = PORTA.7
00038     //  #BIT CLKI = PORTA.7
00039
00040 #BYTE PORTB = 0x0F81
00041     #BIT RB0 = PORTB.0
00042     //  #BIT INT0 = PORTB.0
00043     //  #BIT AN12 = PORTB.0
00044     //  #BIT FLT0 = PORTB.0
00045     #BIT RB1 = PORTB.1
00046     //  #BIT INT1 = PORTB.1
00047     //  #BIT AN10 = PORTB.1
00048     #BIT RB2 = PORTB.2
00049     //  #BIT INT2 = PORTB.2
00050     //  #BIT AN8 = PORTB.2
00051     #BIT RB3 = PORTB.3
00052     //  #BIT CCP2_PORTB = PORTB.3
00053     //  #BIT AN9 = PORTB.3
00054     #BIT RB4 = PORTB.4
00055     //  #BIT KBI0 = PORTB.4
00056     //  #BIT AN11 = PORTB.4
00057     #BIT RB5 = PORTB.5
00058     //  #BIT KBI1 = PORTB.5
00059     //  #BIT PGM = PORTB.5
00060     #BIT RB6 = PORTB.6
00061     //  #BIT KBI2 = PORTB.6
00062     //  #BIT PGC = PORTB.6
00063     #BIT RB7 = PORTB.7
00064     //  #BIT KBI3 = PORTB.7
00065     //  #BIT PGD = PORTB.7
00066
00067 #BYTE PORTC = 0x0F82
00068     #BIT RC0 = PORTC.0
00069     //  #BIT T1OSO = PORTC.0
00070     //  #BIT T13CKI = PORTC.0
00071     #BIT RC1 = PORTC.1
00072     //  #BIT T1OSI = PORTC.1
00073     //  #BIT CCP2_PORTC = PORTC.1
00074     #BIT RC2 = PORTC.2
00075     //  #BIT CCP1 = PORTC.2
00076     #BIT RC3 = PORTC.3
00077     //  #BIT SCK = PORTC.3
```

```
00078     //  #BIT SCL = PORTC.3
00079     #BIT RC4 = PORTC.4
00080     //  #BIT SDI = PORTC.4
00081     //  #BIT SDA = PORTC.4
00082     #BIT RC5 = PORTC.5
00083     //  #BIT SDO = PORTC.5
00084     #BIT RC6 = PORTC.6
00085     //  #BIT TX = PORTC.6
00086     //  #BIT CK = PORTC.6
00087     #BIT RC7 = PORTC.7
00088     //  #BIT RX = PORTC.7
00089
00090 #BYTE PORTD = 0x0F83
00091     #BIT RD0 = PORTD.0
00092     //  #BIT PSP0 = PORTD.0
00093     #BIT RD1 = PORTD.1
00094     //  #BIT PSP1 = PORTD.1
00095     #BIT RD2 = PORTD.2
00096     //  #BIT PSP2 = PORTD.2
00097     #BIT RD3 = PORTD.3
00098     //  #BIT PSP3 = PORTD.3
00099     #BIT RD4 = PORTD.4
00100     //  #BIT PSP4 = PORTD.4
00101     #BIT RD5 = PORTD.5
00102     //  #BIT PSP5 = PORTD.5
00103     //  #BIT P1B = PORTD.5
00104     #BIT RD6 = PORTD.6
00105     //  #BIT PSP6 = PORTD.6
00106     //  #BIT P1C = PORTD.6
00107     #BIT RD7 = PORTD.7
00108     //  #BIT PSP7 = PORTD.7
00109     //  #BIT P1D = PORTD.7
00110
00111 #BYTE PORTE = 0x0F84
00112     #BIT RE0 = PORTE.0
00113     //  #BIT RD = PORTE.0
00114     //  #BIT NOT_RD = PORTE.0
00115     //  #BIT AN5 = PORTE.0
00116     #BIT RE1 = PORTE.1
00117     //  #BIT WR = PORTE.1
00118     //  #BIT NOT_WR = PORTE.1
00119     //  #BIT AN6 = PORTE.1
00120     #BIT RE2 = PORTE.2
00121     //  #BIT CS = PORTE.2
00122     //  #BIT NOT_CS = PORTE.2
00123     //  #BIT AN7 = PORTE.2
00124     #BIT RE3 = PORTE.3
00125     //  #BIT MCLR = PORTE.3
00126     //  #BIT NOT_MCLR = PORTE.3
00127     //  #BIT VPP = PORTE.3
00128
00129 #BYTE LATA = 0x0F89
00130     #BIT LATA0 = LATA.0
00131     #BIT LATA1 = LATA.1
00132     #BIT LATA2 = LATA.2
00133     #BIT LATA3 = LATA.3
00134     #BIT LATA4 = LATA.4
00135     #BIT LATA5 = LATA.5
00136     #BIT LATA6 = LATA.6
00137     #BIT LATA7 = LATA.7
00138
00139 #BYTE LATB = 0x0F8A
00140     #BIT LATB0 = LATB.0
00141     #BIT LATB1 = LATB.1
00142     #BIT LATB2 = LATB.2
00143     #BIT LATB3 = LATB.3
00144     #BIT LATB4 = LATB.4
00145     #BIT LATB5 = LATB.5
00146     #BIT LATB6 = LATB.6
00147     #BIT LATB7 = LATB.7
00148
00149 #BYTE LATC = 0x0F8B
00150     #BIT LATC0 = LATC.0
00151     #BIT LATC1 = LATC.1
00152     #BIT LATC2 = LATC.2
00153     #BIT LATC3 = LATC.3
00154     #BIT LATC4 = LATC.4
00155     #BIT LATC5 = LATC.5
00156     #BIT LATC6 = LATC.6
00157     #BIT LATC7 = LATC.7
00158
00159 #BYTE LATD = 0x0F8C
00160     #BIT LATD0 = LATD.0
00161     #BIT LATD1 = LATD.1
00162     #BIT LATD2 = LATD.2
00163     #BIT LATD3 = LATD.3
00164     #BIT LATD4 = LATD.4
```

```
00165     #BIT LATD5 = LATD.5
00166     #BIT LATD6 = LATD.6
00167     #BIT LATD7 = LATD.7
00168
00169 #BYTE LATE = 0x0F8D
00170     #BIT LATE0 = LATE.0
00171     #BIT LATE1 = LATE.1
00172     #BIT LATE2 = LATE.2
00173
00174 #BYTE TRISA = 0x0F92
00175     #BIT TRISA0 = TRISA.0
00176     #BIT TRISA1 = TRISA.1
00177     #BIT TRISA2 = TRISA.2
00178     #BIT TRISA3 = TRISA.3
00179     #BIT TRISA4 = TRISA.4
00180     #BIT TRISA5 = TRISA.5
00181     #BIT TRISA6 = TRISA.6
00182     #BIT TRISA7 = TRISA.7
00183
00184 #BYTE TRISB = 0x0F93
00185     #BIT TRISB0 = TRISB.0
00186     #BIT TRISB1 = TRISB.1
00187     #BIT TRISB2 = TRISB.2
00188     #BIT TRISB3 = TRISB.3
00189     #BIT TRISB4 = TRISB.4
00190     #BIT TRISB5 = TRISB.5
00191     #BIT TRISB6 = TRISB.6
00192     #BIT TRISB7 = TRISB.7
00193
00194 #BYTE TRISC = 0x0F94
00195     #BIT TRISC0 = TRISC.0
00196     #BIT TRISC1 = TRISC.1
00197     #BIT TRISC2 = TRISC.2
00198     #BIT TRISC3 = TRISC.3
00199     #BIT TRISC4 = TRISC.4
00200     #BIT TRISC5 = TRISC.5
00201     #BIT TRISC6 = TRISC.6
00202     #BIT TRISC7 = TRISC.7
00203
00204 #BYTE TRISD = 0x0F95
00205     #BIT TRISD0 = TRISD.0
00206     #BIT TRISD1 = TRISD.1
00207     #BIT TRISD2 = TRISD.2
00208     #BIT TRISD3 = TRISD.3
00209     #BIT TRISD4 = TRISD.4
00210     #BIT TRISD5 = TRISD.5
00211     #BIT TRISD6 = TRISD.6
00212     #BIT TRISD7 = TRISD.7
00213
00214 #BYTE TRISE = 0x0F96
00215     #BIT TRISE0 = TRISE.0
00216     #BIT TRISE1 = TRISE.1
00217     #BIT TRISE2 = TRISE.2
00218
00219 #BYTE T1CON = 0x0FCD
00220 #BYTE T2CON = 0x0FCA
00221 #BYTE T3CON = 0x0FB1
00222
```

Índice Remissivo