ECU EFI PCM Powertrain Control Module

2013.1

Gerado por Doxygen 1.8.3.1

Segunda, 1 de Julho de 2013 20:18:54

Sumário

Capítulo 1

FATEC Santo Andre

Dr. Newton da Costa Brandão

Curso de Graduacao em Tecnologia em Eletrônica Automotiva

Disciplina : Unidades de Gerenciamento de Motores - 4o semestre

Professor : Edson Caoru Kitani

Software de Gerenciamento de Injeção Eletrônica Monoponto

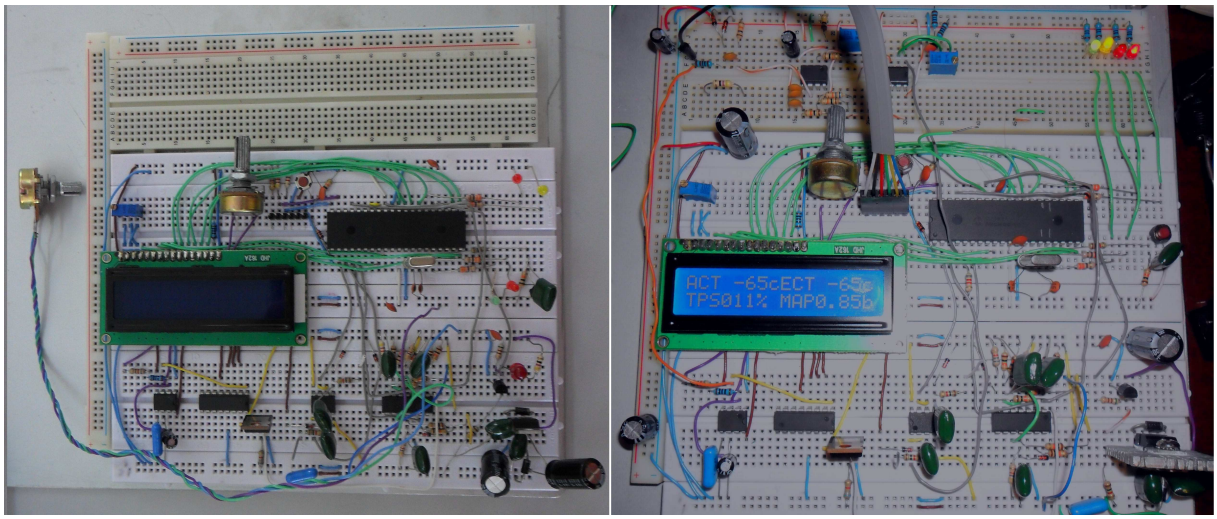


Figura 1.1: Montagem inicial conforme projeto anterior e montagem final com o condicionamento do MAP, chaveamento dos LEDs ao terra e dissipador no regulador de tensão.

1.1 Integrantes do Grupo

1.1.1 1o semestre de 2010

Autores

Albino Sergio de Souza R.A. :
Bruno Martin de Alcantara Dias R.A. :
Marcelo Silva de Santana R.A. :

1.1.2 2o semestre de 2010

Autores

Daniel Alvarez Businaro R.A. : 0910041
Douglas Barbosa de Oliveira R.A. : 0910112
Fabricio Sforcin R.A. : 0910341
Renan do Nascimento Felix R.A. : 0910090
Wellington Alves dos Santos R.A. : 0910880

1.1.3 1o semestre de 2011

Autores

Andre R.A. :
Bruno Rodrigues R.A. : 0910945
Rodrigo da Silva Cassimiro R.A. : 0920071
Ronaldo Siqueira da Gama R.A. : 0920037
Sergio Mitsue Kague R.A. : 0920215

1.1.4 2o semestre de 2011

Autores

Bruno Silva R.A. : 1012010
Filipi Assis Lima R.A. : 0920630
Henrique Mendes R.A. : 1012038
Lenilson Santana Lima R.A. : 1012028
Marcos Felipe R.A. : 1013003

1.1.5 1o semestre de 2012

Autores

Anderson Cazzolato R.A. : 1023005
Carlos Augusto Mortagua R.A. : 1013034
Eduardo Aquino R.A. : 1023026
Fabio Martinez de Moraes R.A. : 1023014
Roberto Juncker R.A. : 1112034

1.1.6 2o semestre de 2012

Autores

Bruno Zani Sampaio R.A. : 1112003
Douglas Matias Martins R.A. : 1112006
Renato C. de Souza R.A. : 1012020
Vitor Silva Santos R.A. : 1022028

Observação

2o semestre de 2012
A temperatura do ar de admissao não influencia o cálculo do tempo de injeção.
TPS esta sem função derivativa (bomba de aceleração).
SPOUT nao funciona corretamente.
Definir uma matriz para o 'Tempo_Injeta_Mais'.
Projetar circuito para monitoração da tensão da bateria.
Conectar o sinal do sensor MAP e anexar funcionalidade ao programa.
Corrigir a matriz referente aos valores de temperatura.
Aperfeiçoar a matriz de valores do TPS.

1.1.7 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

Aviso

1o semestre de 2013
Segundo nota do prof. Edson :
"Os softwares de controle do bico injetor encontram-se no sitio da disciplina. Eles devem ser analisados e implementados novamente."
Portanto, as fontes do semestre anterior foram usadas como ponto de partida.
A qualquer momento as fontes dos semestres anteriores estão a disposição com o prof. Edson.
Tendo em vista uma melhor compreensão e facilidade em trabalho em grupo, foi feita a separação em arquivos por função ou sinal.
Foi configurado as entradas analógicas na sequência: AN0 (TPS), AN1(ACT), AN2(MAP) e AN3(ECT).
A entrada RB1 que antes era usada em paralelo com RB0 para detetar as transições do PIP agora passa a comandar o display.

Problema 1o semestre de 2013

Erros encontrados durante a apresentação final serão passados para o próximo semestre pelo prof. Edson.

Futuras Atividades 1o semestre de 2013

- Passagem de porções do programa para assembler.
- Mudança do acesso das matrizes de indice (com um programa feito pelo CCS) para ponteiro.
- O condicionamento dos sinais analógicos estão limitados à excursão do LM358 alimentado a 5Vcc.

  Na prática a excursão de saída nos circuitos condicionadores analógicos ficou entre 0,6V e 3,5V. Ver figura ilustrativa (Figura_1_2).

  O adequado é substituí-los por Rail-to-Rail Operational Amplifiers.

  Exemplo: MCP6001, OP295, LTC6081, LT1677 entre outros.

  Uma outra possibilidade é alimentar os operacionais com 7 volts.
- Quanto ao condicionamento dos sinais digitais, o circuito com o comparador LM339 deve ser modificado para Schmitt trigger.

  Os divisores de tensão devem ser revistos pois se a tensão da bateria for abaixo de 10V durante a partida os sinais digitais (exemplo o sensor Hall) se perdem.

- Configurar AN4 para a leitura da tensão da bateria.
- Acrescentar circuto de condicionamento para AN3 e AN4
- Esta á a primeira vez que foi usado Doxygen para documentação. Melhores práticas para indexação, nomes de referências e diagramação são esperados.

Lista de Figuras

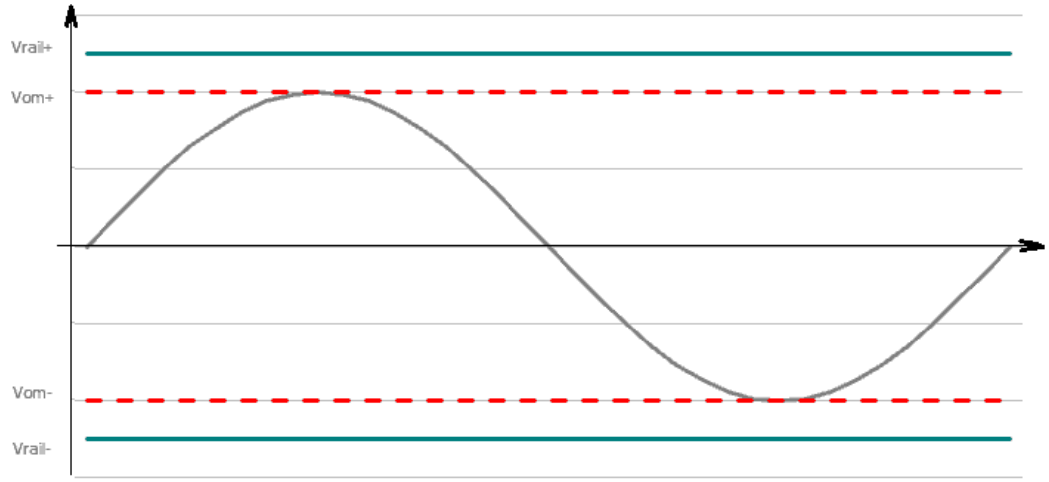


Figura 1.2: Excursão do sinal de saída de um amplificador operacional comum

Capítulo 2

Lista de tabelas, figuras e arquivos

2.1 Tabelas

Lista de Tabelas

2.2 Relacao de esquemas, imagens, gráficos, figuras

2.3 Como o programa foi compilado

```
::
:: \file build.bat
:: \brief compilar as fontes usando o CCS
::

del *.err
del *.hex

set CCS="C:\Program Files\PICC\CCSC.exe"

::  Valid options:
::  +FB Select PCB (12 bit)
::  +FM Select PCM (14 bit)
::  +FH Select PCH (PIC18XXX)
::  +Yx Optimization level x (0-9)
::  +ES Standard error file
::  -T  Do not generate a tree file
::  +T  Create call tree (.TRE)
::  -A  Do not create stats file (.STA)
::  +A  Create stats file (.STA)
```

```
::  -EW Suppress warnings (use with +EA)
::  +EW Show warning messages
::  -E  Only show first error
::  +EA Show all error messages and all warnings
::  +EX Error/warning message format uses GCC's "brief format" (compatible with GCC editor environments)
::  +DF Enables the output of an OFF debug file.
::  +PE Keep status window up only if there are errors

set D1= +FH +PE +A

set LIB="C:\Program Files\PICC\Devices;C:\Program Files\PICC\Drivers"

%CCS% %D1% I=%LIB% +EXPORT _PIP.c
%CCS% %D1% I=%LIB% +EXPORT _ANA.c
%CCS% %D1% I=%LIB% +EXPORT _BaseTempo.c
%CCS% %D1% I=%LIB% +EXPORT _BotaoDisplay.c
%CCS% %D1% I=%LIB% +EXPORT _lcd.c
%CCS% %D1% I=%LIB% +EXPORT _pinagem_ecu.c
%CCS% %D1% I=%LIB% +EXPORT _funcoes_gerais.c
%CCS% %D1% I=%LIB% +EXPORT _estrategia.c

%CCS% %D1% I=%LIB% +EXPORT main.c

%CCS% %D1%
       LINK="ger201301.hex=_PIP.o,_ANA.o,_BaseTempo.o,_BotaoDisplay.o,_estrategia.o,_funcoes_gerais.o,_pinagem_ecu.o,_lcd.o,ma

del c:\c.log
del *.lst
del *.TRE
del *.COF
del *.esym
del *.osym
del *.o
del *.cod
del *.sym
del *.pjt
```

2.4 Estatísticas de utilização de memória

```
ROM used:   14786 (45%)
            14786 (45%) including unused fragments

            5 Average locations per line
            24 Average locations per statement

RAM used:   857 (42%) at main() level
            898 (44%) worst case

Stack used: 7/31 worst case (4 in main + 3 for interrupts)

Lines Stmts  %   Files
----- ----- ---  -----
    0     0   0  ger201301.c
   92    50   8  _PIP.c
   52     0   0  _fuses.h
  464     0   0  ..\..\..\..\Program Files\PICC\Devices\18F4550.h
  269     0   0  regs_18F4550.h
  142     0   0  _pinagem_ecu.h
  195     0   0  _MemoryMap.h
    8     0   0  _ANA.h
  122     0   0  _matriz.h
   22     0   0  _LCD.h
   10     0   0  _BaseTempo.h
  118    37   6  _ANA.c
  101    42   7  _BaseTempo.c
   19     1   0  _BotaoDisplay.c
    5     0   0  _BotaoDisplay.h
  157    91  15  _estrategia.c
   10     0   0  _estrategia.h
   72    29   5  _funcoes_gerais.c
   21     4   1  _pinagem_ecu.c
  133    45   7  _lcd.c
  423   125  20  ..\..\..\..\Program Files\PICC\Drivers\string.h
   32     0   0  ..\..\..\..\Program Files\PICC\Drivers\stddef.h
   28     0   0  ..\..\..\..\Program Files\PICC\Drivers\ctype.h
  548   190  31  main.c
    5     0   0  _PIP.h
    9     0   0  _funcoes_gerais.h
----- -----
 3057   614 Total

Page ROM  %  RAM   Vol Diff  Functions:
```

```
---- --- --- ---   --- ----  ----------
0     42   0   1             @delay_ms1
0     38   0   1             @delay_us1
0    374   1   2  1737  8.8  Trata_int_RB0
0     34   0   5             @MUL1616
0     70   0   5             @DIV1616
0     20   0   0    47  1.5  AD_LeituraFinalizada
0     36   0   1    59  1.2  AD_Start
0    244   1   1  1421  8.3  AD_Next_Port_Choice
0     96   0   2   447  5.2  Trata_Timer0
0    140   0   2   644  8.1  Trata_Timer1
0     18   0   0    76  2.0  Trata_Timer2
0      8   0   0    25  1.9  Trata_int_RB1
0    118   0   0  1375  4.9  Calcular_Estrategia
0    172   1   4   348  6.1  Calcular_Avanco_IGN
0    144   0   2  1324  4.4  Calcular_Injecao
0     72   0   3   258  4.6  Calcular_Aceleracao
0     44   0   3             @DIV88
0    354   1  11   781  7.0  LongFix_to_ASCII
0     18   0   0   117  2.2  init_Pinagens
0     60   0   5   129  4.6  strlen
0      8   0   0    57  1.9  LCD_tris
0     12   0   3    76  2.4  LCD_escreve_word
0     54   0   1   231  2.3  LCD_pulso_comando
0    194   1   2   508  1.4  LCD_inicializacao
0    106   0   5   289  6.8  LCD_mensagem
0  10584  36  20  2456 10.1  MAIN
0    184   1   0  1535  2.9  InitHW
0      6   0   0    16  1.0  InitSystem
0     14   0   0    32  1.0  Reset_Prog
0    848   3  21  3079  8.6  Atualiza_Display
0     34   0   0             @goto10198
0     32   0   1             @SPRINTF
0    190   1   9             @PRINTF_LU_314
0    188   1   6             @PRINTF_D0_314
 Inline        0             @cinit1

Program metrics:
    Functions               25
    Statements              614
    Comments                257
    Volume (V)              27368
    Difficulty (D)          111.2
    Effort to implement (E) 3043779
    Time to implement (T)   46 hours, 58 minutes
    Est Delivered Bugs (B)  7
    Cyclomatic Complexity   80
    Maintainability (MI)    34

 Segment     Used  Free
-----------  ----  ----
00000-00002     4  0
00004-00006     0  4
00008-000E8   226  0
000EA-07FFE  14556  17978
```

Capítulo 3

Lista de Futuras Atividades

Arquivo _matriz.h

Eficiência Volumétrica = f(RPM, TPS)

Injetor = f(Vbat)

Na partida: Injetor = f(ECT)

Global Calcular_Injecao (void)

Acerto do tempo de injeção conforme tensão da bateria

Acerto do tempo de injeção conforme carga no motor

Acerto do tempo de injeção conforme estratégia / regime no motor, onde a soma dos tempos não pode passar de 255

page FATEC Santo Andre

1o semestre de 2013

- Passagem de porções do programa para assembler.
- Mudança do acesso das matrizes de indice (com um programa feito pelo CCS) para ponteiro.
- O condicionamento dos sinais analógicos estão limitados à excursão do LM358 alimentado a 5Vcc.
  Na prática a excursão de saída nos circuitos condicionadores analógicos ficou entre 0,6V e 3,5V. Ver figura ilustrativa (Figura_1_2).
  O adequado é substituí-los por Rail-to-Rail Operational Amplifiers.
  Exemplo: MCP6001, OP295, LTC6081, LT1677 entre outros.
  Uma outra possibilidade é alimentar os operacionais com 7 volts.
- Quanto ao condicionamento dos sinais digitais, o circuito com o comparador LM339 deve ser modificado para Schmitt trigger.
  Os divisores de tensão devem ser revistos pois se a tensão da bateria for abaixo de 10V durante a partida os sinais digitais (exemplo o sensor Hall) se perdem.
- Configurar AN4 para a leitura da tensão da bateria.
- Acrescentar circuto de condicionamento para AN3 e AN4
- Esta á a primeira vez que foi usado Doxygen para documentação. Melhores práticas para indexação, nomes de referências e diagramação são esperados.

Arquivo regs_18F4550.h

Acrescentar registros do PIC conforme a necessidade

Capítulo 4

Lista de Problemas

page FATEC Santo Andre

1o semestre de 2013

Erros encontrados durante a apresentação final serão passados para o próximo semestre pelo prof. Edson.

Capítulo 5

Índice dos Módulos

5.1 Módulos

Esta é a lista com todos os módulos:

Capítulo 6

Índice das Estruturas de Dados

6.1 Estruturas de Dados

Aqui estão as estruturas de dados, uniões e suas respectivas descrições:

Capítulo 7

Índice dos Arquivos

7.1 Lista de Arquivos

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

Capítulo 8

Módulos

8.1 Funcao dos pinos do PIC

MainStart

- #define LED_Start LATC1

  Indicador de inicio de programa.

Ignicao

- #define SPout LATB4

  saida para o modulo de ignicao
- #define LED_Ign LATC0

  Indicador Ignicao.
- #define Ign RA4

  Linha 15 - K15.

Injecao

- #define LED_Injetor LATC6

  Indicador Injecao.
- #define Injetor_out LATB3

  saida para valvula injetora
- #define LED_Bomba LATC7

  Indicador Bomba.
- #define Bomba LATb2

  saida para bomba

LCD_D

LCD Barramento de dados

- #define **LCD_PORTA_DADOS** output_D
- #define **LCD_PIN_0** pin_d0
- #define **LCD_PIN_1** pin_d1
- #define **LCD_PIN_2** pin_d2
- #define **LCD_PIN_3** pin_d3

- #define **LCD_PIN_4** pin_d4
- #define **LCD_PIN_5** pin_d5
- #define **LCD_PIN_6** pin_d6
- #define **LCD_PIN_7** pin_d7

LCD_C

LCD Linhas de comando

- #define **LCD_RS** pin_E0
- #define **LCD_EN** pin_E1
- #define **LCD_Botao** RB1

TPS

- #define TPSpin pin_a0

  pinagem usada no TPS

MAP

- #define MAPpin pin_a2

  pinagem usada no MAP

Portas

- void init_Pinagens (void)

  Definicao das portas analógicas.

8.1.1 Descrição Detalhada

Capítulo 9

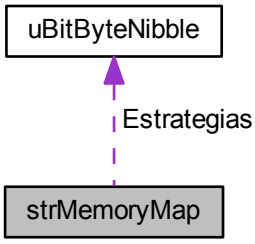
Estruturas

9.1 Referência da Estrutura strMemoryMap

Estrutura do Mapa de Memoria.

`#include <_MemoryMap.h>`

Diagrama de colaboração para strMemoryMap:



Campos de Dados

TPS

Throttle Position Sensor

- int **TPS**
- int **TPS_RAW**
- unsigned int **TPS_pointer**
- int **TPS_Port**
- int **TPS_Anterior**
- int **TPS_Aceleracao**
- unsigned int **TPS_CicloCalc**

RPM

Rotational Speed Sensor

- unsigned long int **RPM**
- unsigned long int **RPM_RAW**

- unsigned int **RPM_pointer**

MAP

Manifold Absolute Pressure Sensor

- long int **MAP**
- int **MAP_RAW**
- unsigned int **MAP_pointer**
- int **MAP_Port**

ACT

Air Charger Temperature Sensor

- int **ACT**
- int **ACT_RAW**
- unsigned int **ACT_pointer**
- int **ACT_Port**

ECT

Engine Coolant Temperature Sensor

- int **ECT**
- int **ECT_RAW**
- unsigned int **ECT_pointer**
- int **ECT_Port**

PIP

Profile Ignition Pickup Sensor

- unsigned long int **PIP_T_TOT**
- unsigned long int **PIP_TH**
- unsigned long int **PIP_TL**
- unsigned long int **IGN_AVANCO**
- unsigned long int **IGN_T1_set**
- unsigned int1 **PIP_SimNao**
- unsigned int1 **PIP_SimNaoComandar**

INJ

Engine Tempo Injetor

- unsigned long int **INJ_T2_set**
- unsigned long int **INJ**

LCD

LCD update

- int **LCD_SHOW**
- int **LCD_Pagina**

ANA

Analogics update

- unsigned int **ANA_UpdateCicle**

Lista

Lista de todas as estrategias de funcionamento

- union uBitByteNibble **Estrategias**
- int **Estrategia_ID**
- unsigned int **Estrategias_CicloCalc**

9.1.1 Descrição Detalhada

Estrutura do Mapa de Memoria.

Definição na linha 72 do arquivo _MemoryMap.h.

A documentação para esta estrutura foi gerada a partir do seguinte arquivo:

- _MemoryMap.h

9.2 Referência da União uBitByteNibble

Acesso a Byte/Nibble/Bits de uma posição de memória.

`#include <_MemoryMap.h>`

Campos de Dados

- unsigned int uByte
- struct {
  unsigned int **LSB**: 4
  unsigned int **MSB**: 4
  } Nibble

- struct {
  unsigned int **b0**: 1
  unsigned int **b1**: 1
  unsigned int **b2**: 1
  unsigned int **b3**: 1
  unsigned int **b4**: 1
  unsigned int **b5**: 1
  unsigned int **b6**: 1
  unsigned int **b7**: 1
  } bits

9.2.1 Descrição Detalhada

Acesso a Byte/Nibble/Bits de uma posição de memória.

Sintaxe de uso da UNION

- definicao de variaveis

  1. union uBitByteNibble IOA, IOB;

- Leitura e Escrita:

  1. IOA.uByte = PORTA;
  2. x = IOA.bits.b0;
  3. PORTA = IOA.uByte;

Definição na linha 19 do arquivo _MemoryMap.h.

9.2.2 Campos

9.2.2.1 struct { ... } bits

agrupamento de 8 bits

Referenciado por main().

9.2.2.2 struct { ... } Nibble

agrupamento de 4 bits

9.2.2.3 unsigned int uByte

agrupamento da estrutura Byte/Nibble/Bits

Definição na linha 20 do arquivo _MemoryMap.h.

Referenciado por main().

A documentação para esta união foi gerada a partir do seguinte arquivo:

- _MemoryMap.h

Capítulo 10

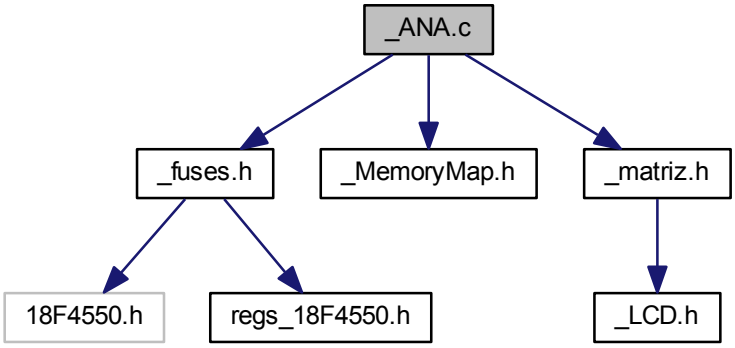
Arquivos

10.1 Referência do Arquivo _ANA.c

varredura e conversão das entradas analógicas

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

Gráfico de dependência de inclusões para _ANA.c:



Funções

- void AD_LeituraFinalizada (void)

  Fim da conversao analogica.
- void AD_Start (char canal)

  Inicio da conversao analogica.

10.1.1 Conversor FxV para o sensor MAP

Lista de Figuras

R6 ajusta a excursao do sinal e R11 define o ganho

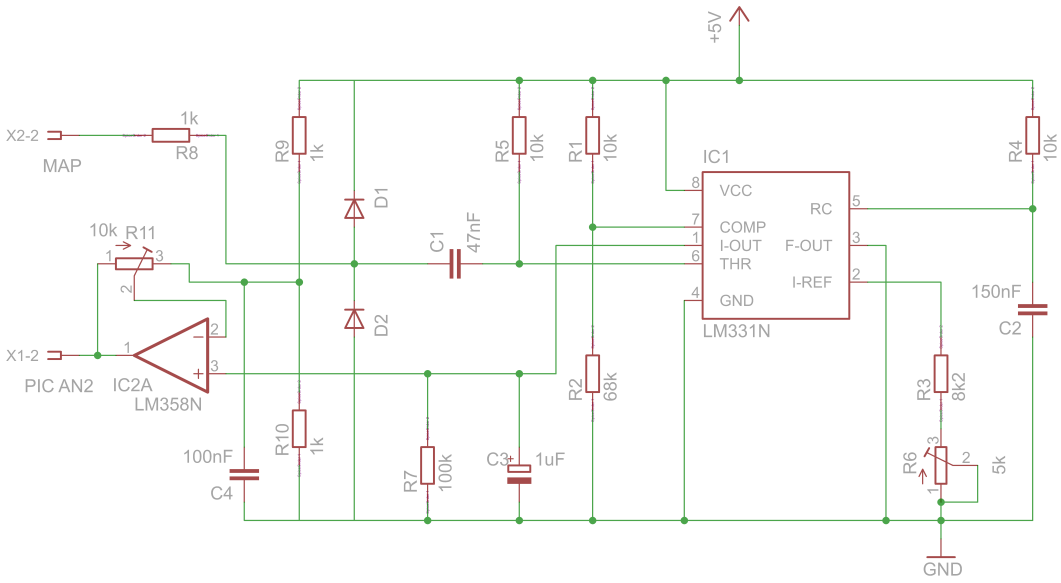


Figura 10.1: Conversor frequência tensao para o sinal MAP

- void AD_Next_Port_Choice (void)

  Seleciona proxima entrada analogica.

Variáveis

- int1 AD_END_Sample = 1

  Fim da conversao analógica, comeca indicando que finalizou!
- unsigned char AD_Valor

  Valor da conversão da ultima porta selecionada.

10.1.2 Descrição Detalhada

varredura e conversão das entradas analógicas

Definição no arquivo _ANA.c.

10.1.3 Funções

10.1.3.1 AD_Next_Port_Choice (void)

Seleciona proxima entrada analogica.

Esta rotina é executada toda vez que ocorre o PIP assim, as leituras ficam sincronizadas com o movimento do motor para obter um valor de leitura mais estavel sem efeito pulsativo.

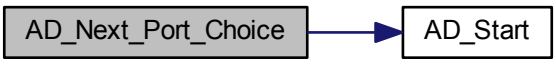
Definição na linha 63 do arquivo _ANA.c.

Referenciado por Trata_int_RB0().

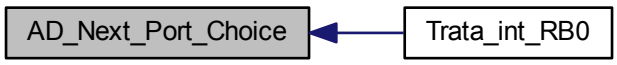
```
00063                                {
00064     static char AD_RelativePort;
00065
00066     if (!AD_END_Sample)
00067         return;
00068
00069     AD_END_Sample = 0;
```

```
00070
00071     switch (AD_RelativePort){
00072     case 0:
00073         //MemoryMap.XXX_RAW = AD_Valor;
00074         AD_RelativePort++;
00075         AD_Start (MemoryMap.TPS_Port); // Start : proxima porta
00076         break;
00077     case 1:
00078         MemoryMap.TPS_RAW = AD_Valor; // RAW = Valor : transfere leitura para a ultima
      porta direcionada
00079         MemoryMap.TPS_pointer = (MemoryMap.TPS_RAW >> 4) & 0x0F;
00080         MemoryMap.TPS = M_TPS_ANA[MemoryMap.TPS_pointer];
00081         AD_RelativePort++;
00082         AD_Start (MemoryMap.ACT_Port); // Start : proxima porta
00083         break;
00084     case 2:
00085         MemoryMap.ACT_RAW = AD_Valor;
00086         MemoryMap.ACT_pointer = (MemoryMap.ACT_RAW >> 4) & 0x0F;
00087         MemoryMap.ACT = MemoryMap.ACT_RAW; // M_ACT[MemoryMap.ACT_pointer];
00088         AD_RelativePort++;
00089         AD_Start (MemoryMap.MAP_Port);
00090         break;
00091     case 3:
00092         /*
00093          *  160 Hz = x.xxV = 0.00 bar \n
00094          *  135 Hz = 3.70V = 0.79 bar \n
00095          *  122 Hz = 3.52V = 0.54 bar \n
00096          *  77 Hz  = 0.75V = 1.00 bar \n
00097          *  bar = 1.1 - AD * 0.46/143
00098          */
00099         MemoryMap.MAP_RAW = AD_Valor;
00100         MemoryMap.MAP_pointer = (MemoryMap.MAP_RAW >> 4) & 0x0F;
00101         MemoryMap.MAP = M_MAP_ANA[MemoryMap.MAP_pointer];
00102         AD_RelativePort++;
00103         AD_Start (MemoryMap.ECT_Port);
00104         break;
00105     case 4:
00106         MemoryMap.ECT_RAW = AD_Valor;
00107         MemoryMap.ECT_pointer = (MemoryMap.ECT_RAW >> 4) & 0x0F;
00108         MemoryMap.ECT = M_ECT_ANA[MemoryMap.ECT_pointer];
00109         AD_RelativePort = 0;
00110         // AD_RelativePort++;
00111         // AD_Start (next port);
00112     default:
00113         AD_RelativePort = 0;
00114         AD_END_Sample = 1;
00115         break;
00116     }
00117 }
```

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



10.1.3.2 AD_Start (char *canal*)

Inicio da conversao analogica.

Parâmetros

in	*canal*	informar qual porta analogica sera iniciada

Definição na linha 32 do arquivo _ANA.c.

Referenciado por AD_Next_Port_Choice().

```
00032                           {
00033     set_adc_channel(canal);
00034     /*
00035     * Foi considerado o tempo de aquisição
00036     * TACQ = TAMP + TC + TCOFF
00037     * TACQ = 2.4us
00038     */
00039     delay_us(3);
00040     read_adc(ADC_START_ONLY);
00041 }
```

Este é o diagrama das funções que utilizam esta função:

10.2 _ANA.c

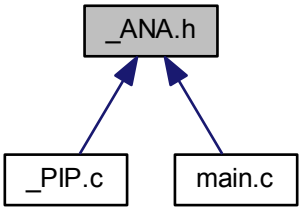
```
00001
00006 #include "_fuses.h"
00007 #include "_MemoryMap.h"
00008 #include "_matriz.h"
00009
00010 /*
00011  *  Variables
00012  */
00013 int1            AD_END_Sample = 1;
00014 unsigned char   AD_Valor;
00020 #INT_AD
00021 void AD_LeituraFinalizada(void) {
00022     AD_Valor = read_adc(ADC_READ_ONLY);
00023     AD_END_Sample = 1;
00024 }
00025
00032 void AD_Start(char canal){
00033     set_adc_channel(canal);
00034     /*
00035     * Foi considerado o tempo de aquisição
00036     * TACQ = TAMP + TC + TCOFF
00037     * TACQ = 2.4us
00038     */
00039     delay_us(3);
00040     read_adc(ADC_START_ONLY);
00041 }
00042
00063 void AD_Next_Port_Choice (void){
00064     static char AD_RelativePort;
00065
00066     if (!AD_END_Sample)
00067         return;
00068
00069     AD_END_Sample = 0;
00070
00071     switch (AD_RelativePort){
```

```
00072     case 0:
00073         //MemoryMap.XXX_RAW = AD_Valor;
00074         AD_RelativePort++;
00075         AD_Start (MemoryMap.TPS_Port); // Start : proxima porta
00076         break;
00077     case 1:
00078         MemoryMap.TPS_RAW = AD_Valor; // RAW = Valor : transfere leitura para a ultima
      porta direcionada
00079         MemoryMap.TPS_pointer = (MemoryMap.TPS_RAW >> 4) & 0x0F;
00080         MemoryMap.TPS = M_TPS_ANA[MemoryMap.TPS_pointer];
00081         AD_RelativePort++;
00082         AD_Start (MemoryMap.ACT_Port); // Start : proxima porta
00083         break;
00084     case 2:
00085         MemoryMap.ACT_RAW = AD_Valor;
00086         MemoryMap.ACT_pointer = (MemoryMap.ACT_RAW >> 4) & 0x0F;
00087         MemoryMap.ACT = MemoryMap.ACT_RAW; // M_ACT[MemoryMap.ACT_pointer];
00088         AD_RelativePort++;
00089         AD_Start (MemoryMap.MAP_Port);
00090         break;
00091     case 3:
00092         /*
00093          *  160 Hz = x.xxV = 0.00 bar \n
00094          *  135 Hz = 3.70V = 0.79 bar \n
00095          *  122 Hz = 3.52V = 0.54 bar \n
00096          *  77 Hz  = 0.75V = 1.00 bar \n
00097          *  bar = 1.1 - AD * 0.46/143
00098          */
00099         MemoryMap.MAP_RAW = AD_Valor;
00100         MemoryMap.MAP_pointer = (MemoryMap.MAP_RAW >> 4) & 0x0F;
00101         MemoryMap.MAP = M_MAP_ANA[MemoryMap.MAP_pointer];
00102         AD_RelativePort++;
00103         AD_Start (MemoryMap.ECT_Port);
00104         break;
00105     case 4:
00106         MemoryMap.ECT_RAW = AD_Valor;
00107         MemoryMap.ECT_pointer = (MemoryMap.ECT_RAW >> 4) & 0x0F;
00108         MemoryMap.ECT = M_ECT_ANA[MemoryMap.ECT_pointer];
00109         AD_RelativePort = 0;
00110         // AD_RelativePort++;
00111         // AD_Start (next port);
00112     default:
00113         AD_RelativePort = 0;
00114         AD_END_Sample = 1;
00115         break;
00116     }
00117 }
```

10.3 Referência do Arquivo _ANA.h

protótipos para _ANA.c

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



Funções

- void AD_Next_Port_Choice (void)

Seleciona proxima entrada analogica.

10.3.1 Descrição Detalhada

protótipos para _ANA.c

Definição no arquivo _ANA.h.

10.3.2 Funções

10.3.2.1 void AD_Next_Port_Choice (void)

Seleciona proxima entrada analogica.

Esta rotina é executada toda vez que ocorre o PIP assim, as leituras ficam sincronizadas com o movimento do motor para obter um valor de leitura mais estavel sem efeito pulsativo.

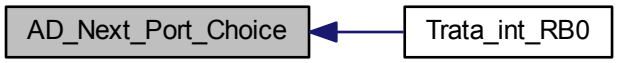
Definição na linha 63 do arquivo _ANA.c.

Referenciado por Trata_int_RB0().

```
00063                                {
00064     static char AD_RelativePort;
00065
00066     if (!AD_END_Sample)
00067         return;
00068
00069     AD_END_Sample = 0;
00070
00071     switch (AD_RelativePort){
00072     case 0:
00073         //MemoryMap.XXX_RAW = AD_Valor;
00074         AD_RelativePort++;
00075         AD_Start (MemoryMap.TPS_Port); // Start : proxima porta
00076         break;
00077     case 1:
00078         MemoryMap.TPS_RAW = AD_Valor; // RAW = Valor : transfere leitura para a ultima
      porta direcionada
00079         MemoryMap.TPS_pointer = (MemoryMap.TPS_RAW >> 4) & 0x0F;
00080         MemoryMap.TPS = M_TPS_ANA[MemoryMap.TPS_pointer];
00081         AD_RelativePort++;
00082         AD_Start (MemoryMap.ACT_Port); // Start : proxima porta
00083         break;
00084     case 2:
00085         MemoryMap.ACT_RAW = AD_Valor;
00086         MemoryMap.ACT_pointer = (MemoryMap.ACT_RAW >> 4) & 0x0F;
00087         MemoryMap.ACT = MemoryMap.ACT_RAW; // M_ACT[MemoryMap.ACT_pointer];
00088         AD_RelativePort++;
00089         AD_Start (MemoryMap.MAP_Port);
00090         break;
00091     case 3:
00092         /*
00093          *  160 Hz = x.xxV = 0.00 bar \n
00094          *  135 Hz = 3.70V = 0.79 bar \n
00095          *  122 Hz = 3.52V = 0.54 bar \n
00096          *  77 Hz  = 0.75V = 1.00 bar \n
00097          *  bar = 1.1 - AD * 0.46/143
00098          */
00099         MemoryMap.MAP_RAW = AD_Valor;
00100         MemoryMap.MAP_pointer = (MemoryMap.MAP_RAW >> 4) & 0x0F;
00101         MemoryMap.MAP = M_MAP_ANA[MemoryMap.MAP_pointer];
00102         AD_RelativePort++;
00103         AD_Start (MemoryMap.ECT_Port);
00104         break;
00105     case 4:
00106         MemoryMap.ECT_RAW = AD_Valor;
00107         MemoryMap.ECT_pointer = (MemoryMap.ECT_RAW >> 4) & 0x0F;
00108         MemoryMap.ECT = M_ECT_ANA[MemoryMap.ECT_pointer];
00109         AD_RelativePort = 0;
00110         // AD_RelativePort++;
00111         // AD_Start (next port);
00112     default:
00113         AD_RelativePort = 0;
00114         AD_END_Sample = 1;
00115         break;
00116     }
00117 }
```

Este é o diagrama das funções utilizadas por esta função:

Este é o diagrama das funções que utilizam esta função:



10.4 _ANA.h

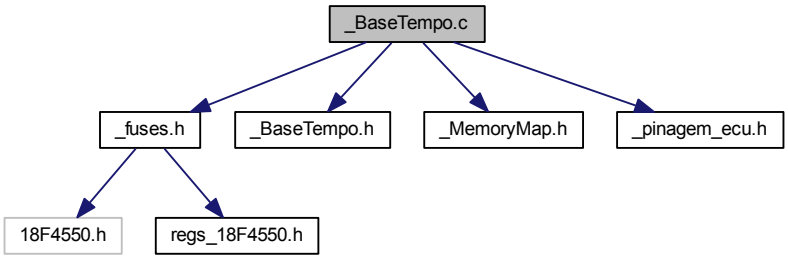
```
00001
00006 //void AD_TRIS(void) ;
00007 void AD_Next_Port_Choice (void);
```

10.5 Referência do Arquivo _BaseTempo.c

Trata a interrupção dos timers.

```
#include "_fuses.h"
#include "_BaseTempo.h"
#include "_MemoryMap.h"
#include "_pinagem_ecu.h"
```

Gráfico de dependência de inclusões para _BaseTempo.c:



Funções

- void Trata_Timer0 (void)

  Timer 0 = base de tempo fixo em 250us.
- void Trata_Timer1 (void)

  Timer 1 Tempo de avanco de ignicao SPOUT, base de tempo fixo em 250us.
- void Trata_Timer2 (void)

  Timer 2 Tempo de injecao.

10.5.1 Descrição Detalhada

Trata a interrupção dos timers.

Definição no arquivo _BaseTempo.c.

10.6 _BaseTempo.c

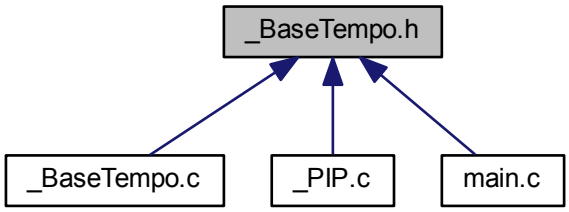
```
00001
00006 #include    "_fuses.h"
00007 #include    "_BaseTempo.h"
00008 #include    "_MemoryMap.h"
00009 #include    "_pinagem_ecu.h"
00010
00015 #int_timer0
00016 void Trata_Timer0(void) {
00017     static char cnt10ms;
00018     // Tempo Fixo
00019     set_timer0(TMR_0_Prescaler + get_timer0());
00020     /*
00021      *  timer 0 é usado para medir rotacao
00022      *  Limitado valor de 1 a 60 => 100 a 6000 RPMs
00023      *  [100RPM] 50 <= RPM_RAW <= 3000 [6000RPM]
00024      *  3001 p/ 100us
00025      *  751  p/ 400us
00026      *  O PIP_TH e PIP_TL são processados em _PIP.C
00027      */
00028     if (MemoryMap.RPM_RAW < 751)   MemoryMap.RPM_RAW++;
00029
00030     /* Base de 10ms */
00031     if ( cnt10ms)   cnt10ms--;
00032     if (!cnt10ms){
00033         cnt10ms = 25;
00034         // LCD
00035         if(MemoryMap.LCD_SHOW)     MemoryMap.LCD_SHOW--;
00036         // Sinais Analogicos
00037         if(MemoryMap.ANA_UpdateCicle)  MemoryMap.ANA_UpdateCicle--;
00038         // Estrategias
00039         if(MemoryMap.Estrategias_CicloCalc)    MemoryMap.Estrategias_CicloCalc--;
00040         // Aceleração
00041         if(MemoryMap.TPS_CicloCalc)    MemoryMap.TPS_CicloCalc--;
00042     }
00043 }
00044
00049 #int_timer1
00050 void Trata_Timer1(void) {
00051     static char PIP_PASSO;
00052     unsigned long int CALCULO;
00053
00054     switch(PIP_PASSO){
00055         case 0:
00056             /* Ligar a ignicao se a rotação > 200 RPM */
00057             //if ( MemoryMap.RPM_pointer > 1) {
00058                 SPout = 0;
00059                 LED_Ign = 0;
00060             //}
00061
00062             if ( MemoryMap.RPM_pointer > 4){
00063                 CALCULO = MemoryMap.PIP_TH;
00064             } else {
00065                 CALCULO = 200;
00066             }
00067             if(!CALCULO) CALCULO++;
00068
00069             CALCULO *= 150;
00070             CALCULO  = 0x00 - CALCULO;
```

```
00071             //CALCULO += get_timer1();
00072             set_timer1(CALCULO);
00073             PIP_PASSO = 1;
00074             break;
00075         case 1:
00076             /* desligar a ignicao e aguardar sincronismo do PIP */
00077             SPout = 1;
00078             LED_Ign = 1;
00079             set_timer1(1); /* MAX > 43ms */
00080             disable_interrupts(int_timer1);
00081             PIP_PASSO = 0;
00082             break;
00083         default:
00084             PIP_PASSO = 0;
00085             break;
00086         }
00087 }
00088
00093 #int_timer2
00094 void Trata_Timer2(void) {
00095     /* Desliga o injetor */
00096     Injetor_out = 0;
00097     LED_Injetor = 1;
00098     /* só liga no proximo PIP/Hall */
00099     setup_timer_2(T2_DISABLED, Padrao_Inj_OffSet, 15);
00100 }
```

10.7 Referência do Arquivo _BaseTempo.h

Tempos bases para _BaseTempo.c.

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



Definições e Macros

- #define TMR_0_Prescaler 63136

  64936=100us 64036=250us 63136=400us
- #define TMR_1_Prescaler 65535

  65386=100us 65161=250us 65535=rapido

10.7.1 Descrição Detalhada

Tempos bases para _BaseTempo.c.

Definição no arquivo _BaseTempo.h.

10.8 _BaseTempo.h

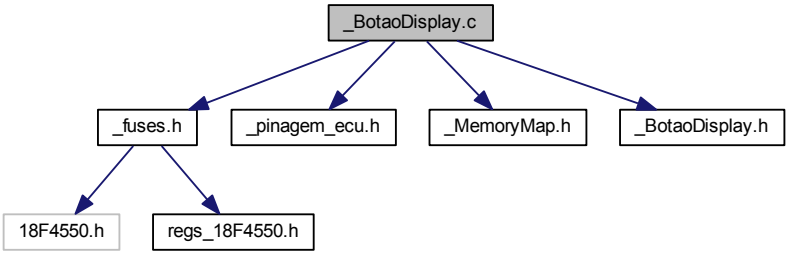
```
00001
00006 #define TMR_0_Prescaler 63136
00007 #define TMR_1_Prescaler 65535
```

10.9 Referência do Arquivo _BotaoDisplay.c

Botão usado para trocar tela no display LCD.

```
#include "_fuses.h"
#include "_pinagem_ecu.h"
#include "_MemoryMap.h"
#include "_BotaoDisplay.h"
```

Gráfico de dependência de inclusões para _BotaoDisplay.c:



Funções

- void Trata_int_RB1 (void)

  Funcao de tratamento da interrupcao externa em RB1, debouce por hardware com capacitor em paralelo ao botão.

10.9.1 Descrição Detalhada

Botão usado para trocar tela no display LCD.

Definição no arquivo _BotaoDisplay.c.

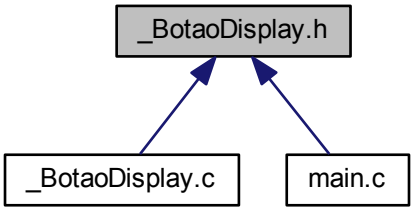
10.10 _BotaoDisplay.c

```
00001
00006 #include    "_fuses.h"
00007 #include    "_pinagem_ecu.h"
00008 #include    "_MemoryMap.h"
00009 #include    "_BotaoDisplay.h"
00015 #int_ext1
00016 void Trata_int_RB1(void){
00017     MemoryMap.LCD_Pagina++;
00018 }
```

10.11 Referência do Arquivo _BotaoDisplay.h

protótipos de _BotaoDisplay.c

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



10.11.1 Descrição Detalhada

protótipos de _BotaoDisplay.c

Definição no arquivo _BotaoDisplay.h.

10.12 _BotaoDisplay.h

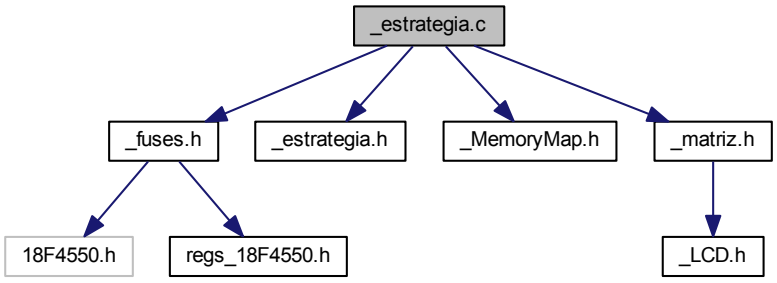
```
00001
```

10.13 Referência do Arquivo _estrategia.c

cálculos para controle do tempo de injeção e avanço de ignição

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

Gráfico de dependência de inclusões para _estrategia.c:



Funções

- void Calcular_Estrategia (void)

  Define qual o regime sendo processado.
- void Calcular_Avanco_IGN (void)

  Calcula o tempo (Timer 1) para o avanço de ignição.
- void Calcular_Injecao (void)

  Calcula o tempo (Timer 2) no eletro injetor onde + tempo se Vbat cair e + tempo se carga (TPS) aumentar Considerado 1=20us 100=2ms 255[max]=5.1ms.
- void Calcular_Aceleracao (void)

  Derivada do sinal do TPS como referência a um mapa.

10.13.1 Descrição Detalhada

cálculos para controle do tempo de injeção e avanço de ignição

Definição no arquivo _estrategia.c.

10.13.2 Funções

10.13.2.1 Calcular_Avanco_IGN (void)

Calcula o tempo (Timer 1) para o avanço de ignição.

Lista de Figuras

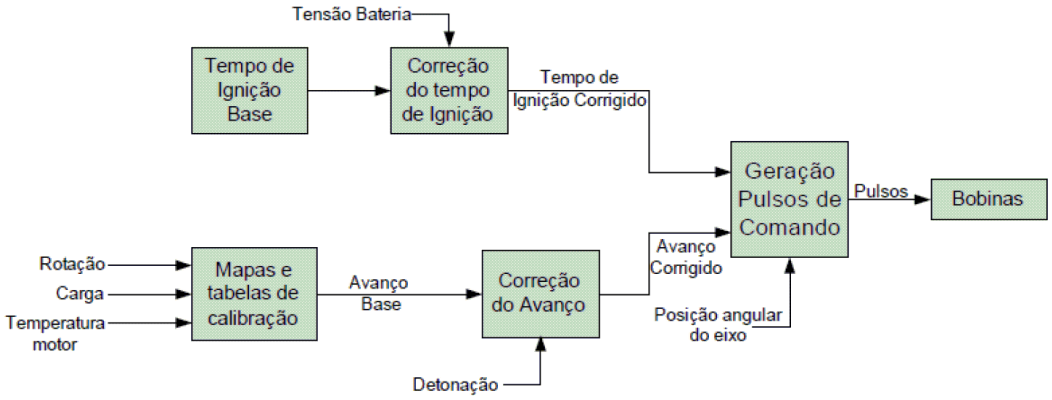


Figura 10.2: Fluxograma geral usado como base para comando do sinal de ignição

Definição na linha 69 do arquivo _estrategia.c.

Referenciado por main().

```
00069                                {
00070     unsigned long int CALCULO;
00071     // Avanco de Ignicao
00072     CALCULO = MemoryMap.RPM_pointer;
00073     CALCULO >>= 2;
00074     CALCULO &= 0x000F;
00075     MemoryMap.IGN_AVANCO = M_IgnAvanco_RPM16_TPS[CALCULO][
      MemoryMap.TPS_pointer];
00076     // Timer 1
00077     CALCULO  = MemoryMap.IGN_AVANCO;
00078     if(!CALCULO) CALCULO++;
00079     CALCULO *= 150;
00080     CALCULO  = 0x00 - CALCULO;
00081     MemoryMap.IGN_T1_set = CALCULO;
00082 }
```

Este é o diagrama das funções que utilizam esta função:

10.13.2.2 void Calcular_Injecao (void)

Calcula o tempo (Timer 2) no eletro injetor onde + tempo se Vbat cair e + tempo se carga (TPS) aumentar

Considerado 1=20us 100=2ms 255[max]=5.1ms.

Futuras Atividades Acerto do tempo de injeção conforme tensão da bateria

Futuras Atividades Acerto do tempo de injeção conforme carga no motor

Futuras Atividades Acerto do tempo de injeção conforme estratégia / regime no motor, onde a soma dos tempos não pode passar de 255

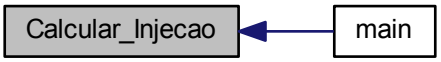
Definição na linha 88 do arquivo _estrategia.c.

Referenciado por main().

```
00088                           {
00089     unsigned long int CALCULO;
00090     CALCULO = Padrao_Inj_OffSet;
00092     // CALCULO += M_INJ_BAT
00094     // CALCULO += M_INJ_TPS
00095
00097     if (MOp_PartidaFrio){
00098         // Injetor = f(Vbat)
00099         // Injetor = f(ECT)
00100         CALCULO += 150;
00101         MemoryMap.PIP_SimNaoComandar=0;
00102     }
00103     if (MOp_PartidaQuente){
00104         CALCULO += 0;
00105         MemoryMap.PIP_SimNaoComandar=0;
00106     }
00107     if (MOp_Aquecimento){
00108         CALCULO += 0;
00109         MemoryMap.PIP_SimNaoComandar=0;
00110     }
00111     if (MOp_MarchaLenta){
00112         CALCULO += 0;
00113         MemoryMap.PIP_SimNaoComandar=0;
00114     }
00115     if (MOp_CargaParcial){
00116         CALCULO += 0;
00117         MemoryMap.PIP_SimNaoComandar=1;
00118     }
00119     if (MOp_PlenaCarga){
00120         CALCULO += 0;
00121         MemoryMap.PIP_SimNaoComandar=1;
00122     }
00123     if (MOp_AceleracaoRapida){
00124         CALCULO += 0;
00125         MemoryMap.PIP_SimNaoComandar=1;
00126     }
00127     if (MOp_Desaceleracao){
00128         CALCULO += 0;
00129         MemoryMap.PIP_SimNaoComandar=1;
00130     }
00131
```

```
00132     // Bomba de aceleração
00133     CALCULO += MemoryMap.TPS_Aceleracao;
00134     // Timer 2
00135     MemoryMap.INJ_T2_set = CALCULO;
00136     // Animação no display
00137     CALCULO <<= 1;
00138     CALCULO /= 10;
00139     MemoryMap.INJ = CALCULO;
00140 }
```

Este é o diagrama das funções que utilizam esta função:



10.14 _estrategia.c

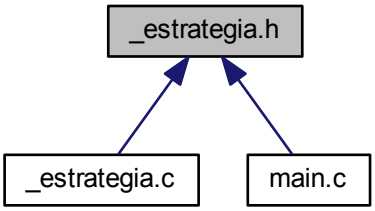
```
00001
00006 #include    "_fuses.h"
00007 #include    "_estrategia.h"
00008 #include    "_MemoryMap.h"
00009 #include    "_matriz.h"
00010
00014  void Calcular_Estrategia(void){
00015     MOp = 0;
00016     // escolha da estratégia
00017     // Partida
00018     if (MemoryMap.RPM < 700){
00019         if (MemoryMap.ECT < 50){
00020             MOp_PartidaFrio = 1;
00021             MemoryMap.Estrategia_ID=0;
00022         }else{
00023             MOp_PartidaQuente = 1;
00024             MemoryMap.Estrategia_ID=1;
00025         }
00026         return;
00027     }
00028     // Aquecendo / marcha lenta
00029     if (MemoryMap.RPM < 1300){
00030         if (MemoryMap.ECT < 50){
00031             MOp_Aquecimento = 1;
00032             MemoryMap.Estrategia_ID=2;
00033         } else {
00034             MOp_MarchaLenta = 1;
00035             MemoryMap.Estrategia_ID=3;
00036         }
00037         return;
00038     }
00039
00040     // Rodando
00041     if (MemoryMap.TPS < 50){
00042         if (!MemoryMap.TPS){
00043             MOp_Desaceleracao = 1;
00044             MemoryMap.Estrategia_ID=7;
00045         } else {
00046             MOp_CargaParcial = 1;
00047             MemoryMap.Estrategia_ID=4;
00048         }
00049     }else{
00050         if (!MemoryMap.TPS_Aceleracao){
00051             MOp_AceleracaoRapida = 1;
00052             MemoryMap.Estrategia_ID=6;
00053         } else {
00054         MOp_PlenaCarga = 1;
00055         MemoryMap.Estrategia_ID=5;
00056         }
00057     }
00058 }
00059
```

```
00069 void Calcular_Avanco_IGN(void) {
00070     unsigned long int CALCULO;
00071     // Avanco de Ignicao
00072     CALCULO = MemoryMap.RPM_pointer;
00073     CALCULO >>= 2;
00074     CALCULO &= 0x000F;
00075     MemoryMap.IGN_AVANCO = M_IgnAvanco_RPM16_TPS[CALCULO][
      MemoryMap.TPS_pointer];
00076     // Timer 1
00077     CALCULO  = MemoryMap.IGN_AVANCO;
00078     if(!CALCULO) CALCULO++;
00079     CALCULO *= 150;
00080     CALCULO  = 0x00 - CALCULO;
00081     MemoryMap.IGN_T1_set = CALCULO;
00082 }
00083
00088 void Calcular_Injecao(void){
00089     unsigned long int CALCULO;
00090     CALCULO = Padrao_Inj_OffSet;
00092     // CALCULO += M_INJ_BAT
00094     // CALCULO += M_INJ_TPS
00095
00097     if (MOp_PartidaFrio){
00098         // Injetor = f(Vbat)
00099         // Injetor = f(ECT)
00100         CALCULO += 150;
00101         MemoryMap.PIP_SimNaoComandar=0;
00102     }
00103     if (MOp_PartidaQuente){
00104         CALCULO += 0;
00105         MemoryMap.PIP_SimNaoComandar=0;
00106     }
00107     if (MOp_Aquecimento){
00108         CALCULO += 0;
00109         MemoryMap.PIP_SimNaoComandar=0;
00110     }
00111     if (MOp_MarchaLenta){
00112         CALCULO += 0;
00113         MemoryMap.PIP_SimNaoComandar=0;
00114     }
00115     if (MOp_CargaParcial){
00116         CALCULO += 0;
00117         MemoryMap.PIP_SimNaoComandar=1;
00118     }
00119     if (MOp_PlenaCarga){
00120         CALCULO += 0;
00121         MemoryMap.PIP_SimNaoComandar=1;
00122     }
00123     if (MOp_AceleracaoRapida){
00124         CALCULO += 0;
00125         MemoryMap.PIP_SimNaoComandar=1;
00126     }
00127     if (MOp_Desaceleracao){
00128         CALCULO += 0;
00129         MemoryMap.PIP_SimNaoComandar=1;
00130     }
00131
00132     // Bomba de aceleração
00133     CALCULO += MemoryMap.TPS_Aceleracao;
00134     // Timer 2
00135     MemoryMap.INJ_T2_set = CALCULO;
00136     // Animação no display
00137     CALCULO <<= 1;
00138     CALCULO /= 10;
00139     MemoryMap.INJ = CALCULO;
00140 }
00141
00145 void Calcular_Aceleracao(void){
00146     char tps_p;
00147     // 50 * 10ms = 0.5s
00148     // TPS 0 [0] a 100 [15]
00149     if (MemoryMap.TPS_pointer > MemoryMap.TPS_Anterior)
00150         tps_p = MemoryMap.TPS_pointer - MemoryMap.TPS_Anterior;
00151     else
00152         tps_p = 0;
00153
00154     MemoryMap.TPS_Aceleracao = M_TPS_ACELERACAO_TPS_ECT[tps_p][
      MemoryMap.ECT_pointer];
00155     MemoryMap.TPS_Anterior = MemoryMap.TPS_pointer;
00156 }
```

10.15 Referência do Arquivo _estrategia.h

protótipos para _estrategia.c

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



Funções

- void Calcular_Estrategia (void)

  Define qual o regime sendo processado.

- void Calcular_Avanco_IGN (void)

  Calcula o tempo (Timer 1) para o avanço de ignição.

- void Calcular_Injecao (void)

  Calcula o tempo (Timer 2) no eletro injetor onde + tempo se Vbat cair e + tempo se carga (TPS) aumentar Considerado 1=20us 100=2ms 255[max]=5.1ms.

- void Calcular_Aceleracao (void)

  Derivada do sinal do TPS como referência a um mapa.

10.15.1 Descrição Detalhada

protótipos para _estrategia.c

Definição no arquivo _estrategia.h.

10.15.2 Funções

10.15.2.1 void Calcular_Avanco_IGN (void)

Calcula o tempo (Timer 1) para o avanço de ignição.

Lista de Figuras

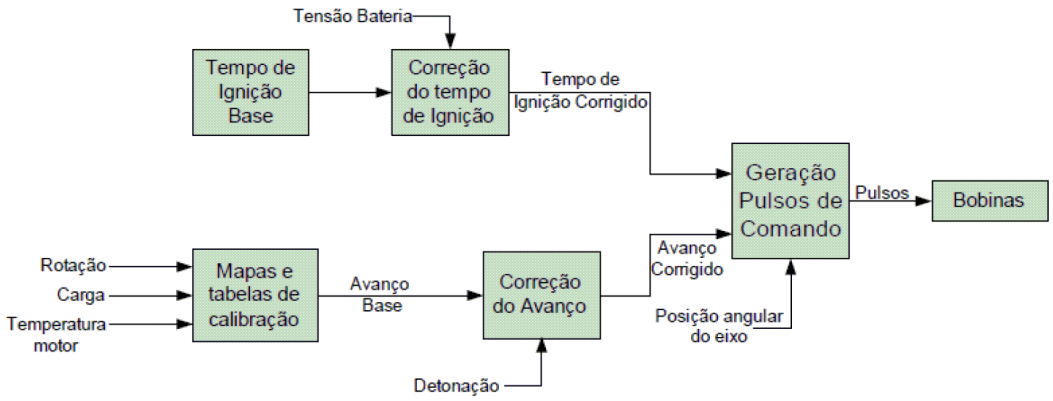


Figura 10.3: Fluxograma geral usado como base para comando do sinal de ignição

Definição na linha 69 do arquivo _estrategia.c.

Referenciado por main().

```
00069                                {
00070     unsigned long int CALCULO;
00071     // Avanco de Ignicao
00072     CALCULO = MemoryMap.RPM_pointer;
00073     CALCULO >>= 2;
00074     CALCULO &= 0x000F;
00075     MemoryMap.IGN_AVANCO = M_IgnAvanco_RPM16_TPS[CALCULO][
      MemoryMap.TPS_pointer];
00076     // Timer 1
00077     CALCULO  = MemoryMap.IGN_AVANCO;
00078     if(!CALCULO) CALCULO++;
00079     CALCULO *= 150;
00080     CALCULO  = 0x00 - CALCULO;
00081     MemoryMap.IGN_T1_set = CALCULO;
00082 }
```

Este é o diagrama das funções que utilizam esta função:

10.15.2.2 void Calcular_Injecao (void)

Calcula o tempo (Timer 2) no eletro injetor onde + tempo se Vbat cair e + tempo se carga (TPS) aumentar

Considerado 1=20us 100=2ms 255[max]=5.1ms.

Futuras Atividades Acerto do tempo de injeção conforme tensão da bateria

Futuras Atividades Acerto do tempo de injeção conforme carga no motor

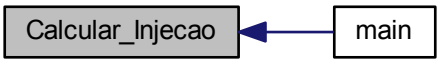
Futuras Atividades Acerto do tempo de injeção conforme estratégia / regime no motor, onde a soma dos tempos não pode passar de 255

Definição na linha 88 do arquivo _estrategia.c.

Referenciado por main().

```
00088                           {
00089     unsigned long int CALCULO;
00090     CALCULO = Padrao_Inj_OffSet;
00092     // CALCULO += M_INJ_BAT
00094     // CALCULO += M_INJ_TPS
00095
00097     if (MOp_PartidaFrio){
00098         // Injetor = f(Vbat)
00099         // Injetor = f(ECT)
00100         CALCULO += 150;
00101         MemoryMap.PIP_SimNaoComandar=0;
00102     }
00103     if (MOp_PartidaQuente){
00104         CALCULO += 0;
00105         MemoryMap.PIP_SimNaoComandar=0;
00106     }
00107     if (MOp_Aquecimento){
00108         CALCULO += 0;
00109         MemoryMap.PIP_SimNaoComandar=0;
00110     }
00111     if (MOp_MarchaLenta){
00112         CALCULO += 0;
00113         MemoryMap.PIP_SimNaoComandar=0;
00114     }
00115     if (MOp_CargaParcial){
00116         CALCULO += 0;
00117         MemoryMap.PIP_SimNaoComandar=1;
00118     }
00119     if (MOp_PlenaCarga){
00120         CALCULO += 0;
00121         MemoryMap.PIP_SimNaoComandar=1;
00122     }
00123     if (MOp_AceleracaoRapida){
00124         CALCULO += 0;
00125         MemoryMap.PIP_SimNaoComandar=1;
00126     }
00127     if (MOp_Desaceleracao){
00128         CALCULO += 0;
00129         MemoryMap.PIP_SimNaoComandar=1;
00130     }
00131
00132     // Bomba de aceleração
00133     CALCULO += MemoryMap.TPS_Aceleracao;
00134     // Timer 2
00135     MemoryMap.INJ_T2_set = CALCULO;
00136     // Animação no display
00137     CALCULO <<= 1;
00138     CALCULO /= 10;
00139     MemoryMap.INJ = CALCULO;
00140 }
```

Este é o diagrama das funções que utilizam esta função:



10.16 _estrategia.h

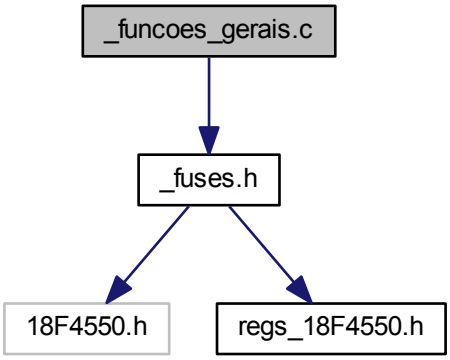
```
00001
00006 void Calcular_Estrategia(void);
00007 void Calcular_Avanco_IGN(void);
00008 void Calcular_Injecao(void);
00009 void Calcular_Aceleracao(void);
```

10.17 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.

10.17.1 Descrição Detalhada

comandos de manipulacao de textos e numeros

Definição no arquivo _funcoes_gerais.c.

10.17.2 Funções

10.17.2.1 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.

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

10.17.2.2 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 }
```

10.17.2.3 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

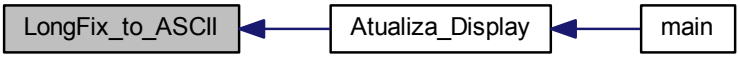
Definição na linha 43 do arquivo _funcoes_gerais.c.

Referenciado por Atualiza_Display().

```
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:



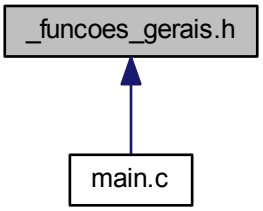
10.18 _funcoes_gerais.c

```
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 }
```

10.19 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

10.19.1 Descrição Detalhada

protótipos para _funcoes_gerais.c

Definição no arquivo _funcoes_gerais.h.

10.19.2 Funções

10.19.2.1 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.

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

10.19.2.2 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 }
```

10.19.2.3 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

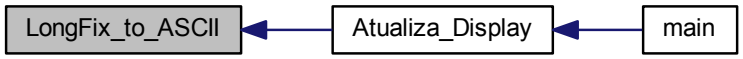
in	*val*	valor a ser convertido.
in	*pos*	ponto decimal.
in,out	*str*	transfere dado

Definição na linha 43 do arquivo _funcoes_gerais.c.

Referenciado por Atualiza_Display().

```
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:



10.20 _funcoes_gerais.h

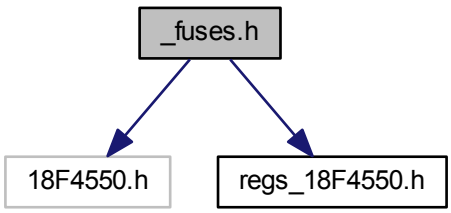
```
00001
00006 char *Int_to_ASCII(unsigned int);
00007 void LongFix_to_ASCII(unsigned long int, int, char *);
00008 int ASCII_TO_INT(char);
```

10.21 Referência do Arquivo _fuses.h

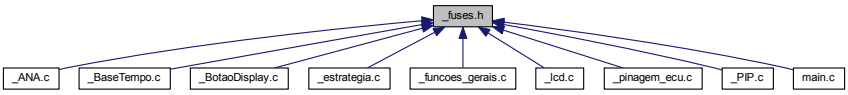
Configura o PIC conforme o modelo.

```
#include <18F4550.h>
#include "regs_18F4550.h"
```

Gráfico de dependência de inclusões para _fuses.h:



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



10.21.1 Descrição Detalhada

Configura o PIC conforme o modelo.

Definição no arquivo _fuses.h.

10.22 _fuses.h

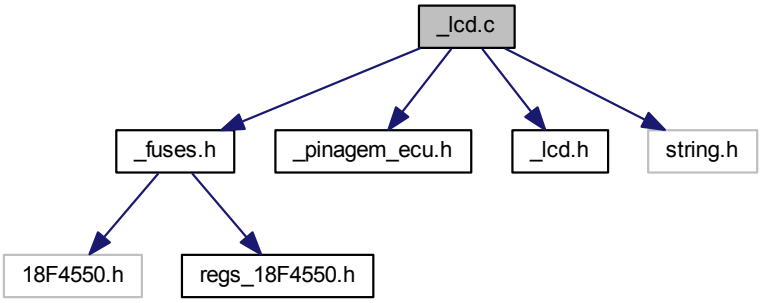
```
00001
00006 /* Bibliotecas */
00007 /* Microcontrolador utilizado */
00008 #include <18F4550.h>
00009 /* Registros 18F */
00010 #include "regs_18F4550.h"
00011
00012 /*
00013  * Definições
00014  * BANCOS DE RAM
00015  * COMUTAR ENTRE OS BANCOS DE MEMORIA.
00016  */
00017 #DEFINE BANK1   BSF STATUS,RP0  /* SELECIONA BANK1 DA MEMORIA RAM */
00018 #DEFINE BANK0   BCF STATUS,RP0  /* SELECIONA BANK0 DA MEMORIA RAM */
00019
00020 /*
00021  * FUSES
00022  * Configuracoes de gravacao, cristal de 20MHz para operar a 48MHz \n
00023  * http://www.ccsinfo.com/forum/viewtopic.php?t=47528 \n
00024  * http://www.ccsinfo.com/forum/viewtopic.php?t=37929&start=3 \n
00025  * HSPLL High-Speed Crystal/Resonator with PLL enabled  \n
00026  * prescaler 1,2,3,4,5,6,10,12  \n
00027  * postscaler 2,3,4,6   \n
00028  * Input Oscillator Frequency = 20 MHz
00029  * PLL Division = ÷5 (100)
00030  * Clock Mode = HSPLL, ECPLL, ECPIO
00031  * MCU Clock Division = ÷2 (00)
00032  * Microcontroller Clock Frequency = 48 MHz
00033  */
00034 #fuses HSPLL, NOWDT, NOPROTECT, NOLVP, NODEBUG, USBDIV, PLL5, CPUDIV1, VREGEN
00035 #fuses PUT, BROWNOUT
00036 /* Conversor AD em 10 ou 8 bits */
00037 #device  ADC = 8
00038 #device CONST = ROM
00039 #ZERO_RAM
00040 #use delay(clock=48000000)
00041 /* Estabelece a prioridade das interrupções */
00042 #priority ext,timer0,timer1,timer2,ad,ext1,ext2,eeprom
00043
00044 /*
00045  * inicialização dos PORTS
00046  */
00047 #use fast_io(a)
00048 #use fast_io(b)
00049 #use fast_io(c)
00050 #use fast_io(d)
00051 #use fast_io(e)
```

10.23 Referência do Arquivo _lcd.c

comandos especificos para o LCD

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

Gráfico de dependência de inclusões para _lcd.c:



10.23.1 JHD162A

Lista de Figuras

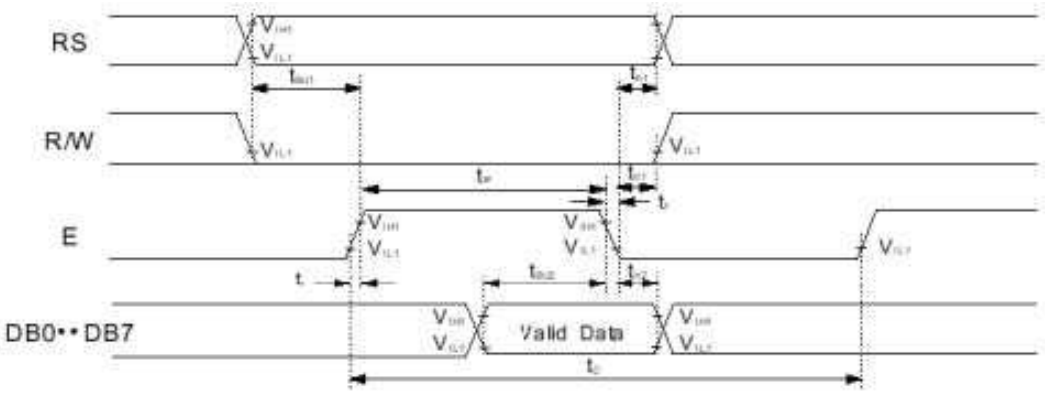


Figura 10.4: JHD162A - pulso de execução de um comando: RS = 0 - pulso de escrita de letra: RS = 1

- #define __dado 1

  Enviar dado.
- #define __instrucao 0

  Enviar comando.
- 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)

  Enviar dado.
- 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_tris (void)

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

Inicialização do display LCD.

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

  Escreve uma frase completa no display.

10.23.2 Descrição Detalhada

comandos especificos para o LCD

Autor

Ricardo de Freitas 1023007

Data

1o semestre de 2013

Definição no arquivo _lcd.c.

10.23.3 Funções

10.23.3.1 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 ?

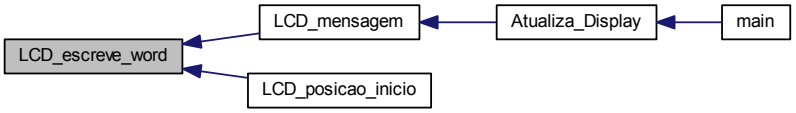
Definição na linha 48 do arquivo _lcd.c.

Referenciado por LCD_mensagem() e LCD_posicao_inicio().

```
00048                                                   {
00049     char i;
00050     output_d(dado);
00051     LCD_pulso_comando(tipo);
00052 }
```

Este é o diagrama das funções utilizadas por esta função:

Este é o diagrama das funções que utilizam esta função:



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

Escreve uma frase completa no display.

Parâmetros

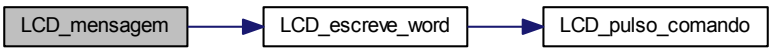
in	*posicao*	Posição a ser escrita no display
in	*frase*	Texto

Definição na linha 122 do arquivo _lcd.c.

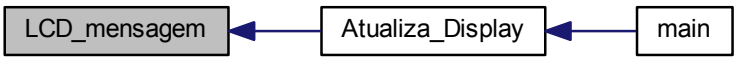
Referenciado por Atualiza_Display().

```
00122                                                           {
00123     char i;
00124
00125     LCD_escreve_word(posicao, __instrucao); //comecar da linhaXcoluna
00126     delay_us(40);
00127
00128     for (i = 0; i < strlen(frase); i++) {
00129         if (0 == frase[i]) break;
00130         LCD_escreve_word(frase[i], __dado);
00131     }
00132 }
```

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



10.23.3.3 void LCD_pulso_comando (unsigned char *tipo*)

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

Parâmetros

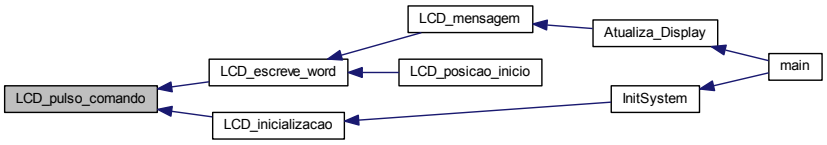
in	*tipo*	Dado ou instrução para o LCD?

Definição na linha 58 do arquivo _lcd.c.

Referenciado por LCD_escreve_word() e LCD_inicializacao().

```
00058                                           {
00059     //1 - Dado, 0 - Instrução
00060     if (tipo == __dado)
00061         output_high(LCD_RS);
00062
00063     if (tipo == __instrucao)
00064         output_low(LCD_RS);
00065
00066     //Pulso em EN RE1
00067     output_low(LCD_EN);
00068     delay_us(10);
00069     output_high(LCD_EN);
00070     delay_us(30);
00071     output_low(LCD_EN);
00072     delay_us(10);
00073 }
```

Este é o diagrama das funções que utilizam esta função:



10.24 _lcd.c

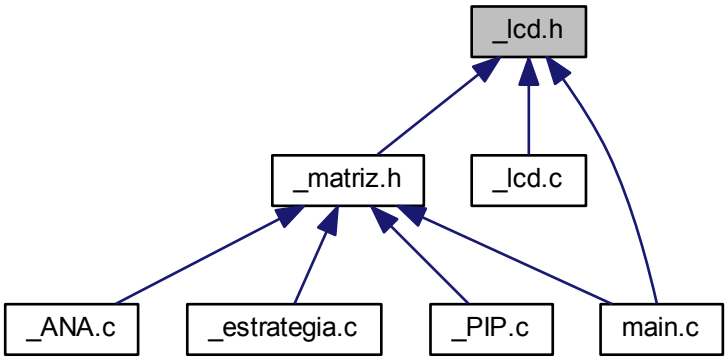
```
00001
00016 #include    "_fuses.h"
00017 #include    "_pinagem_ecu.h"
00018 #include    "_lcd.h"
00019 #include    <string.h>
00020
00021 // comandos para o LCD
00022 #define __dado 1
00023 #define __instrucao 0
00025 // ##### prototipos #####
00026 //void LCD_tris(void);
00027 void LCD_escreve_word(unsigned char dado, unsigned char tipo);
00028 void LCD_escreve_4bits(unsigned char dado, char pos);
00029 void LCD_pulso_comando(unsigned char tipo);
00030 void LCD_posicao_inicio(void);
00031 // ##### fim prototipos #####
00032
00037 void LCD_tris(void) {
00038     TRISD = TRISD & 0b00000000; // forca saidas
00039     TRISE = TRISE & 0b11111100; // forca saidas
00040 }
00041
00048 void LCD_escreve_word(unsigned char dado, unsigned char tipo) {
00049     char i;
00050     output_d(dado);
00051     LCD_pulso_comando(tipo);
00052 }
00053
00058 void LCD_pulso_comando(unsigned char tipo) {
```

```
00059     //1 - Dado, 0 - Instrução
00060     if (tipo == __dado)
00061         output_high(LCD_RS);
00062
00063     if (tipo == __instrucao)
00064         output_low(LCD_RS);
00065
00066     //Pulso em EN RE1
00067     output_low(LCD_EN);
00068     delay_us(10);
00069     output_high(LCD_EN);
00070     delay_us(30);
00071     output_low(LCD_EN);
00072     delay_us(10);
00073 }
00074
00078 void LCD_posicao_inicio(void) {
00079     unsigned char info;
00080     info = 0x01; //limpar o display e posicionar o cursor linha 1 coluna 1
00081     LCD_escreve_word(info, __instrucao);
00082     delay_us(1800);
00083 }
00084
00088 void LCD_inicializacao(void) {
00089
00090     unsigned char info;
00091
00092     delay_ms(500); //necessário esperar antes de qq coisa
00093
00094     LCD_tris();
00095
00096     delay_ms(15);
00097     output_d(0x30);
00098     LCD_pulso_comando(__instrucao);
00099     delay_ms(4);
00100     LCD_pulso_comando(__instrucao);
00101     delay_us(100);
00102     LCD_pulso_comando(__instrucao);
00103     delay_us(40);
00104     output_d(0x38);
00105     LCD_pulso_comando(__instrucao);
00106     delay_us(40);
00107     output_d(0x01);
00108     LCD_pulso_comando(__instrucao);
00109     delay_us(1800);
00110     output_d(0x0C);
00111     LCD_pulso_comando(__instrucao);
00112     delay_us(40);
00113     output_d(0x06);
00114     delay_us(40);
00115 }
00116
00122 void LCD_mensagem(unsigned char posicao, unsigned char *frase) {
00123     char i;
00124
00125     LCD_escreve_word(posicao, __instrucao); //comecar da linhaXcoluna
00126     delay_us(40);
00127
00128     for (i = 0; i < strlen(frase); i++) {
00129         if (0 == frase[i]) break;
00130         LCD_escreve_word(frase[i], __dado);
00131     }
00132 }
```

10.25 Referência do Arquivo _lcd.h

prototipos para _lcd.c

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



Definições e Macros

LCD162

LCD 16x2

- #define LCD_colunas 16

  Quantidade de colunas do display.
- #define LCD_linhas 2

  Quantidade de linhas do display.
- #define LCD_L1 0x80

  Endereco Base da primeira linha do display.
- #define LCD_L2 0xC0

  Endereco Base da segunda linha do display.
- #define LCD_DataWidth 8

  Tamanho do barramento de dados do display.

Funções

- void LCD_inicializacao (void)

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

  Escreve uma frase completa no display.
- void LCD_tris (void)

  Configura Portas para entrada ou saida.

10.25.1 Descrição Detalhada

prototipos para _lcd.c

Definição no arquivo _lcd.h.

10.25.2 Funções

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

Escreve uma frase completa no display.

Parâmetros

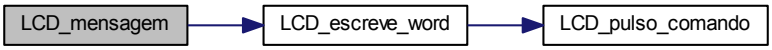
in	*posicao*	Posição a ser escrita no display
in	*frase*	Texto

Definição na linha 122 do arquivo _lcd.c.

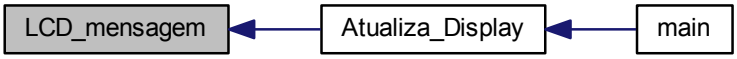
Referenciado por Atualiza_Display().

```
00122                                                                   {
00123     char i;
00124
00125     LCD_escreve_word(posicao, __instrucao); //comecar da linhaXcoluna
00126     delay_us(40);
00127
00128     for (i = 0; i < strlen(frase); i++) {
00129         if (0 == frase[i]) break;
00130         LCD_escreve_word(frase[i], __dado);
00131     }
00132 }
```

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



10.26 _lcd.h

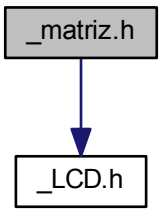
```
00001
00011 #define LCD_colunas 16
00012 #define LCD_linhas 2
00013 #define LCD_L1   0x80
00014 #define LCD_L2   0xC0
00015 #define LCD_DataWidth   8
00018 //prototipos das funcoes
00019 void LCD_inicializacao(void);
00020 void LCD_mensagem(unsigned char, unsigned char*);
00021 void LCD_tris(void);
```

10.27 Referência do Arquivo _matriz.h

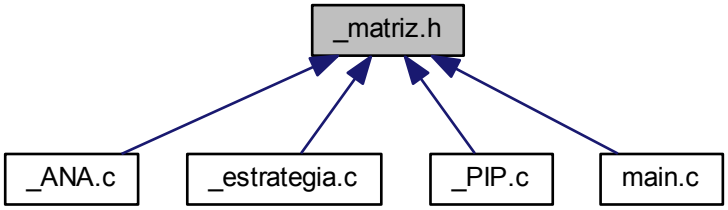
Mapas de comandos, conversoes AD e mensagens.

```
#include "_LCD.h"
```

Gráfico de dependência de inclusões para _matriz.h:



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



Variáveis

- byte M_MAP_ANA [16]
- byte M_TPS_ANA [16]
- byte M_ECT_ANA [16]
- byte M_TPS_ACELERACAO_TPS_ECT [16][16]
- byte M_IgnAvanco_RPM16_TPS [16][16]

LCD_M

Mascaras usadas no display

- #define **Pagina_BoasVindas** 0
- #define **Pagina_K15** 1
- #define **Pagina_Tela1** 2
- #define **Pagina_Tela2** 3
- char **LCD_BemVindo_L1** [1+LCD_colunas] = "FATEC 1o 2013 "

- char **LCD_BemVindo_L2** [1+LCD_colunas] = "Gerenciamento I "
- char **LCD_EsperaK15_L1** [1+LCD_colunas] = "Aguarda Ignicao "
- char **LCD_Mascara_L1** [1+LCD_colunas] = "____RPM "
- char **LCD_Mascara_L2** [1+LCD_colunas] = "I=__,_msA=__,_ms"
- char **LCD_Mascara_L3** [1+LCD_colunas] = "ACT___ cECT___ c"
- char **LCD_Mascara_L4** [1+LCD_colunas] = "TPS___% MAP___ b"

10.27.1 Descrição Detalhada

Mapas de comandos, conversoes AD e mensagens.

Data

1o semestre de 2013

Futuras Atividades Eficiência Volumétrica = f(RPM, TPS)
Injetor = f(Vbat)
Na partida: Injetor = f(ECT)

Pré-Condição

As matrizes são nomeadas M_(onde é usado)_(origem da informação).
Matrizes dependentes da rotação têm tamanho preferencial 60.
Contudo RPM16 indica um índice de 0 a 15 sendo obtido pela divisão por 4 do índice RPM.
Matrizes dependentes de sinais analógicos têm tamanho 16.

Definição no arquivo _matriz.h.

10.27.2 Variáveis

10.27.2.1 M_ECT_ANA[16]

Valor Inicial:

```
=
{
    -30,    -20,    -10,    0,      10,     20,     30,     40,
    50,     60,     70,     80,     90,     100,    110,    120
}
```

Temperatura do motor (água de arrefecimento) em graus Celsius

Definição na linha 65 do arquivo _matriz.h.

Referenciado por AD_Next_Port_Choice().

10.27.2.2 M_IgnAvanco_RPM16_TPS[16][16]

Valor Inicial:

```
=
{
    {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    {3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3},
    {10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10},
    {12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12},
    {14,14,14,14,14,14,14,14,14,14,14,14,14,14,14,14},
    {14,14,14,14,14,14,14,14,14,14,14,14,14,14,14,14},
    {15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15},
```

```
    {15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15},
    {16,16,16,16,16,16,16,16,16,16,16,16,16,16,16,16},
    {14,14,14,14,14,14,14,14,14,14,14,14,14,14,14,14},
    {13,13,13,13,13,13,13,13,13,13,13,13,13,13,13,13},
    {12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12},
    {11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11},
    {10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10},
    {9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9}
}
```

Mapa de avanco baseado na rotação e na carga:

- indice: 0 [100 RPM] a 15 [6000 RPM], neste caso, o indice RPM16 = [RPM_pointer]/4
- indice: 0 [1.05 bar] a 15 [0.28 bar]
- valor: 1 = 100us, 10 = 1ms

Definição na linha 104 do arquivo _matriz.h.

Referenciado por Calcular_Avanco_IGN().

10.27.2.3 M_MAP_ANA[16]

Valor Inicial:

```
=
{
    43,49,54,59,64,69,74,79,
    85,90,95,100,101,102,103,104
}
```

Matriz que relaciona Conversor Frequência Tensão para o sensor MAP (Conversor FxV para o sensor MAP) conforme :

- 160 Hz = x.xxV = 0.00 bar
- 135 Hz = 3.70V = 0.79 bar
- 122 Hz = 3.52V = 0.54 bar
- 77 Hz = 0.75V = 1.00 bar
- bar = 1.1 - AD $*$ 0.46/143

Definição na linha 45 do arquivo _matriz.h.

Referenciado por AD_Next_Port_Choice().

10.27.2.4 M_TPS_ACELERACAO_TPS_ECT[16][16]

Valor Inicial:

```
=
{
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {37, 31, 26, 21, 16, 12, 8, 7, 7, 6, 5, 4, 3, 3, 2, 2},
    {41, 35, 30, 25, 21, 17, 14, 11, 8, 7, 6, 5, 4, 3, 2, 2},
    {43, 37, 31, 26, 22, 18, 15, 12, 9, 8, 7, 6, 5, 4, 3, 2},
    {46, 40, 34, 28, 24, 20, 17, 13, 10, 9, 8, 7, 6, 5, 4, 3},
    {50, 44, 38, 32, 28, 24, 21, 17, 14, 12, 11, 10, 9, 8, 6, 5},
    {55, 52, 49, 46, 40, 34, 29, 25, 22, 20, 18, 17, 15, 14, 11, 8},
    {60, 57, 55, 53, 47, 42, 37, 33, 30, 28, 26, 25, 23, 22, 18, 14},
    {69, 68, 67, 67, 60, 54, 48, 46, 45, 42, 39, 37, 34, 32, 26, 21},
    {100, 99, 98, 98, 92, 86, 80, 70, 60, 58, 56, 54, 52, 51, 45, 39},
    {108, 107, 106, 106, 101, 96, 91, 83, 75, 73, 71, 70, 68, 67, 62, 57},
```

```
    {116, 116, 115, 115, 111, 107, 103, 96, 90, 88, 87, 86, 84, 84, 80, 76},
    {125, 124, 124, 124, 121, 118, 115, 110, 105, 104, 103, 102, 101, 100, 97, 94},
    {133, 133, 132, 132, 130, 128, 126, 123, 120, 119, 118, 118, 117, 117, 115, 113},
    {141, 141, 141, 141, 140, 139, 138, 136, 135, 134, 134, 134, 133, 133, 132, 131},
    {150, 150, 150, 150, 150, 150, 150, 150, 150, 150, 150, 150, 150, 150, 150, 150}
}
```

TPS_ACELERACAO depende da variação de TPS[16] e de ECT[16]

M_TPS_ACELERACAO_TPS_ECT = f(TPS,ECT)

onde 1=20us, 100=2ms, 255[max]=5.1ms

Definição na linha 77 do arquivo _matriz.h.

Referenciado por Calcular_Aceleracao().

10.27.2.5 M_TPS_ANA

Valor Inicial:

```
=
{
    0,  6,  13, 20, 26, 33, 40, 46,
    53, 60, 66, 73, 80, 86, 93, 100
}
```

Matriz de linearização do sinal TPS

Definição na linha 55 do arquivo _matriz.h.

Referenciado por AD_Next_Port_Choice().

10.28 _matriz.h

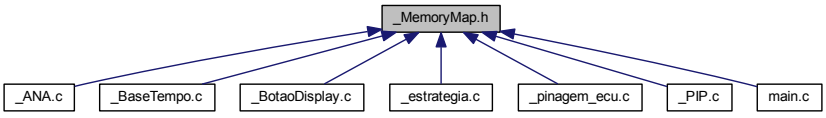
```
00001
00015 #include "_LCD.h"
00016
00022 #define Pagina_BoasVindas   0
00023  char LCD_BemVindo_L1 [1 + LCD_colunas]   = "FATEC 1o 2013   ";
00024  char LCD_BemVindo_L2 [1 + LCD_colunas]   = "Gerenciamento I ";
00025 #define Pagina_K15  1
00026  char LCD_EsperaK15_L1[1 + LCD_colunas]   = "Aguarda Ignicao ";
00027 #define Pagina_Tela1    2
00028  char LCD_Mascara_L1  [1 + LCD_colunas]   = "____RPM         ";
00029  char LCD_Mascara_L2  [1 + LCD_colunas]   = "I=__,_msA=__,_ms";
00030 #define Pagina_Tela2    3
00031  char LCD_Mascara_L3  [1 + LCD_colunas]   = "ACT___ cECT___ c";
00032  char LCD_Mascara_L4  [1 + LCD_colunas]   = "TPS___% MAP___ b";
00045 byte  M_MAP_ANA [16] =
00046 {
00047     43,49,54,59,64,69,74,79,
00048     85,90,95,100,101,102,103,104
00049 };
00050
00055 byte  M_TPS_ANA [16] =
00056 {
00057     0,  6,  13, 20, 26, 33, 40, 46,
00058     53, 60, 66, 73, 80, 86, 93, 100
00059 };
00060
00065 byte  M_ECT_ANA [16] =
00066 {
00067     -30,    -20,    -10,    0,      10,     20,     30,     40,
00068     50,     60,     70,     80,     90,     100,    110,    120
00069 };
00070
00077 byte  M_TPS_ACELERACAO_TPS_ECT [16][16] =
00078 {
00079     {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
00080     {37, 31, 26, 21, 16, 12, 8, 7, 7, 6, 5, 4, 3, 3, 2, 2},
00081     {41, 35, 30, 25, 21, 17, 14, 11, 8, 7, 6, 5, 4, 3, 2, 2},
00082     {43, 37, 31, 26, 22, 18, 15, 12, 9, 8, 7, 6, 5, 4, 3, 2},
00083     {46, 40, 34, 28, 24, 20, 17, 13, 10, 9, 8, 7, 6, 5, 4, 3},
00084     {50, 44, 38, 32, 28, 24, 21, 17, 14, 12, 11, 10, 9, 8, 6, 5},
```

```
00085     {55, 52, 49, 46, 40, 34, 29, 25, 22, 20, 18, 17, 15, 14, 11, 8},
00086     {60, 57, 55, 53, 47, 42, 37, 33, 30, 28, 26, 25, 23, 22, 18, 14},
00087     {69, 68, 67, 67, 60, 54, 48, 46, 45, 42, 39, 37, 34, 32, 26, 21},
00088     {100, 99, 98, 98, 92, 86, 80, 70, 60, 58, 56, 54, 52, 51, 45, 39},
00089     {108, 107, 106, 106, 101, 96, 91, 83, 75, 73, 71, 70, 68, 67, 62, 57},
00090     {116, 116, 115, 115, 111, 107, 103, 96, 90, 88, 87, 86, 84, 84, 80, 76},
00091     {125, 124, 124, 124, 121, 118, 115, 110, 105, 104, 103, 102, 101, 100, 97, 94},
00092     {133, 133, 132, 132, 130, 128, 126, 123, 120, 119, 118, 118, 117, 117, 115, 113},
00093     {141, 141, 141, 141, 140, 139, 138, 136, 135, 134, 134, 134, 133, 133, 132, 131},
00094     {150, 150, 150, 150, 150, 150, 150, 150, 150, 150, 150, 150, 150, 150, 150, 150}
00095 };
00096
00104 byte  M_IgnAvanco_RPM16_TPS[16][16] =
00105 {
00106     {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
00107     {3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3},
00108     {10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10},
00109     {12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12},
00110     {14,14,14,14,14,14,14,14,14,14,14,14,14,14,14,14},
00111     {14,14,14,14,14,14,14,14,14,14,14,14,14,14,14,14},
00112     {15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15},
00113     {15,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15},
00114     {16,16,16,16,16,16,16,16,16,16,16,16,16,16,16,16},
00115     {14,14,14,14,14,14,14,14,14,14,14,14,14,14,14,14},
00116     {13,13,13,13,13,13,13,13,13,13,13,13,13,13,13,13},
00117     {12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12},
00118     {11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11},
00119     {10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10},
00120     {9,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9}
00121 };
```

10.29 Referência do Arquivo _MemoryMap.h

Memoria usada para transferencia de informacao entre os modulos do programa.

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



Estruturas de Dados

- union uBitByteNibble

  Acesso a Byte/Nibble/Bits de uma posição de memória.
- struct strMemoryMap

  Estrutura do Mapa de Memoria.

Variáveis

- struct strMemoryMap MemoryMap

  Variável com as informações comums.

Const

Constantes de funcionamento dos componentes

- int Padrao_Inj_OffSet = 160

Modos

- Modos de Operacao

1. Partida do motor para 300 RPM.
2. Aquecimento
3. Marcha lenta
4. Cargas parciais
5. Plena carga
6. Aceleração rápida
7. Desaceleração

Autor

Fabrício José Pacheco Pujatti

- #define **MOp_PartidaFrio** MemoryMap.Estrategias.bits.b0
- #define **MOp_PartidaQuente** MemoryMap.Estrategias.bits.b1
- #define **MOp_Aquecimento** MemoryMap.Estrategias.bits.b2
- #define **MOp_MarchaLenta** MemoryMap.Estrategias.bits.b3
- #define **MOp_CargaParcial** MemoryMap.Estrategias.bits.b4
- #define **MOp_PlenaCarga** MemoryMap.Estrategias.bits.b5
- #define **MOp_AceleracaoRapida** MemoryMap.Estrategias.bits.b6
- #define **MOp_Desaceleracao** MemoryMap.Estrategias.bits.b7
- #define **MOp** MemoryMap.Estrategias.uByte
- int **EstrategiaTexto** [8][9]

10.29.1 Descrição Detalhada

Memoria usada para transferencia de informacao entre os modulos do programa.

Data

1o semestre de 2013

Definição no arquivo _MemoryMap.h.

10.29.2 Variáveis

10.29.2.1 int EstrategiaTexto[8][9]

Valor Inicial:

```
={
    "Pt Frio ", "PtQuente", "Aquecimt", "Mr Lenta",
    "CParcial", "Plena Cg", "Acelera ", "Desacel "
}
```

Definição na linha 62 do arquivo _MemoryMap.h.

10.29.2.2 Padrao_Inj_OffSet = 160

1=20us 100=2ms 255[max]=5.1ms

Definição na linha 193 do arquivo _MemoryMap.h.

Referenciado por Calcular_Injecao(), InitHW() e Trata_Timer2().

10.30 _MemoryMap.h

```
00001
00019 union uBitByteNibble {
00020     unsigned int uByte;
00022     struct {
00023         unsigned int LSB : 4;
00024         unsigned int MSB : 4;
00025     } Nibble;
00027     struct {
00028         unsigned int b0 : 1;
00029         unsigned int b1 : 1;
00030         unsigned int b2 : 1;
00031         unsigned int b3 : 1;
00032         unsigned int b4 : 1;
00033         unsigned int b5 : 1;
00034         unsigned int b6 : 1;
00035         unsigned int b7 : 1;
00036     } bits;
00037 };
00038
00052 #define MOp_PartidaFrio             MemoryMap.Estrategias.bits.b0
00053 #define MOp_PartidaQuente           MemoryMap.Estrategias.bits.b1
00054 #define MOp_Aquecimento             MemoryMap.Estrategias.bits.b2
00055 #define MOp_MarchaLenta             MemoryMap.Estrategias.bits.b3
00056 #define MOp_CargaParcial            MemoryMap.Estrategias.bits.b4
00057 #define MOp_PlenaCarga              MemoryMap.Estrategias.bits.b5
00058 #define MOp_AceleracaoRapida        MemoryMap.Estrategias.bits.b6
00059 #define MOp_Desaceleracao           MemoryMap.Estrategias.bits.b7
00060 #define MOp                         MemoryMap.Estrategias.uByte
00061
00062  int EstrategiaTexto [8][9]={
00063     "Pt Frio ", "PtQuente", "Aquecimt", "Mr Lenta",
00064     "CParcial", "Plena Cg", "Acelera ", "Desacel "
00065 };
00072  struct strMemoryMap{
00073
00079                     int TPS;
00080                     int TPS_RAW;
00081         unsigned    int TPS_pointer;
00082                     int TPS_Port;
00083                     int TPS_Anterior;
00084                     int TPS_Aceleracao;
00085         unsigned    int TPS_CicloCalc;
00093         unsigned long int   RPM;
00094         unsigned long int   RPM_RAW;
00095         unsigned int        RPM_pointer;
00103         long int        MAP;
00104             int         MAP_RAW;
00105         unsigned int    MAP_pointer;
00106             int         MAP_Port;
00114                 int     ACT;
00115                 int     ACT_RAW;
00116         unsigned int    ACT_pointer;
00117                 int     ACT_Port;
00125                 int     ECT;
00126                 int     ECT_RAW;
00127         unsigned int    ECT_pointer;
00128                 int     ECT_Port;
00136         unsigned long int   PIP_T_TOT;
00137         unsigned long int   PIP_TH;
00138         unsigned long int   PIP_TL;
00139         unsigned long int   IGN_AVANCO;
00140         unsigned long int   IGN_T1_set;
00141         unsigned      int1  PIP_SimNao;
00142         unsigned      int1  PIP_SimNaoComandar;
00150         unsigned long int   INJ_T2_set;
00151         unsigned long int   INJ;
00159         int LCD_SHOW;
00160         int LCD_Pagina;
00168         unsigned int    ANA_UpdateCicle;
00176                     union   uBitByteNibble        Estrategias;
00177                     int     Estrategia_ID;
00178         unsigned    int     Estrategias_CicloCalc;
```

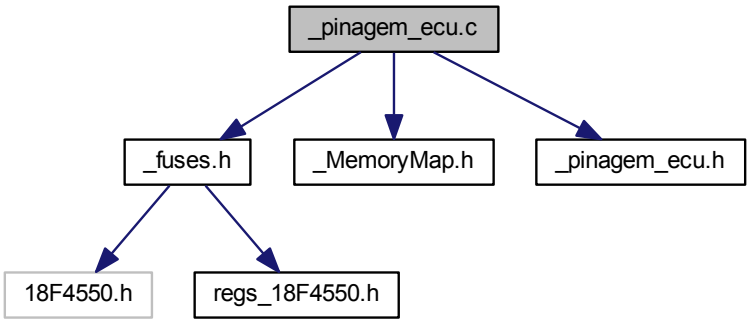
```
00180  } ;
00181
00182 struct strMemoryMap MemoryMap;
00193      int Padrao_Inj_OffSet = 160;
```

10.31 Referência do Arquivo _pinagem_ecu.c

Arquivo de Configuracao.

```
#include "_fuses.h"
#include "_MemoryMap.h"
#include "_pinagem_ecu.h"
```

Gráfico de dependência de inclusões para _pinagem_ecu.c:



Funções

- void init_Pinagens (void)

  Definicao das portas analógicas.

10.31.1 Descrição Detalhada

Arquivo de Configuracao.

Definição no arquivo _pinagem_ecu.c.

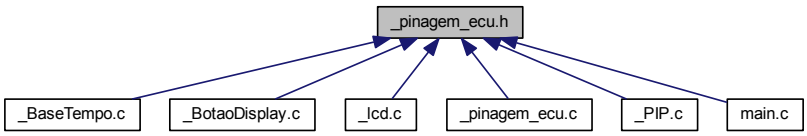
10.32 _pinagem_ecu.c

```
00001
00006 #include "_fuses.h"
00007 #include "_MemoryMap.h"
00008 #include "_pinagem_ecu.h"
00009
00014 void init_Pinagens(void){
00015     /* Definicao das portas analógicas */
00016     MemoryMap.TPS_Port = 0 ;
00017     MemoryMap.ACT_Port = 1 ;
00018     MemoryMap.MAP_Port = 2 ;
00019     MemoryMap.ECT_Port = 3 ;
00020 }
```

10.33 Referência do Arquivo _pinagem_ecu.h

Arquivo de Configuracao e prototipo de _pinagem_ecu.c.

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



Definições e Macros

MainStart

- #define LED_Start LATC1
  Indicador de inicio de programa.

Ignicao

- #define SPout LATB4
  saida para o modulo de ignicao
- #define LED_Ign LATC0
  Indicador Ignicao.
- #define Ign RA4
  Linha 15 - K15.

Injecao

- #define LED_Injetor LATC6
  Indicador Injecao.
- #define Injetor_out LATB3
  saida para valvula injetora
- #define LED_Bomba LATC7
  Indicador Bomba.
- #define Bomba LATb2
  saida para bomba

LCD_D

LCD Barramento de dados

- #define **LCD_PORTA_DADOS** output_D
- #define **LCD_PIN_0** pin_d0
- #define **LCD_PIN_1** pin_d1
- #define **LCD_PIN_2** pin_d2
- #define **LCD_PIN_3** pin_d3
- #define **LCD_PIN_4** pin_d4
- #define **LCD_PIN_5** pin_d5
- #define **LCD_PIN_6** pin_d6
- #define **LCD_PIN_7** pin_d7

LCD_C

LCD Linhas de comando

- #define **LCD_RS** pin_E0
- #define **LCD_EN** pin_E1
- #define **LCD_Botao** RB1

TPS

- #define TPSpin pin_a0

  pinagem usada no TPS

MAP

- #define MAPpin pin_a2

  pinagem usada no MAP

Funções

Portas

- void init_Pinagens (void)

  Definicao das portas analógicas.

10.33.1 Descrição Detalhada

Arquivo de Configuracao e prototipo de _pinagem_ecu.c.

??ables "Lista de Tabelas"

Definição no arquivo _pinagem_ecu.h.

10.34 _pinagem_ecu.h

```
00001
00069 #define LED_Start       LATC1
00076 #define SPout           LATB4
00077 #define LED_Ign         LATC0
00078 #define Ign             RA4
00085 #define LED_Injetor     LATC6
00086 #define Injetor_out     LATB3
00087 #define LED_Bomba       LATC7
00088 #define Bomba           LATb2
00096 #define LCD_PORTA_DADOS output_D
00097 #define LCD_PIN_0 pin_d0
00098 #define LCD_PIN_1 pin_d1
00099 #define LCD_PIN_2 pin_d2
00100 #define LCD_PIN_3 pin_d3
00101 #define LCD_PIN_4 pin_d4
00102 #define LCD_PIN_5 pin_d5
00103 #define LCD_PIN_6 pin_d6
00104 #define LCD_PIN_7 pin_d7
00105
00112 #define LCD_RS pin_E0
00113 #define LCD_EN pin_E1
00114 #define LCD_Botao RB1
00115
00121 #define TPSpin          pin_a0
00128 #define MAPpin          pin_a2
00135 void init_Pinagens(void);
00136
```

10.35 Referência do Arquivo _PIP.c

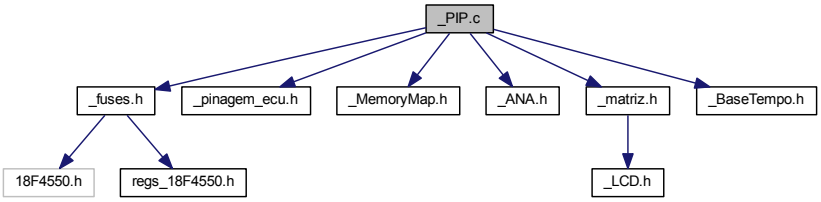
Tratamento do sinal do PIP.

Tabela 10.12: Pinagem do PIC 18F4550 e uso no projeto

pino	Funções	Utilizado	ligação	comentário
1	MCLR/VPP/RE3	MCLR		Circuito de reset
2	RA0/AN0	AN0		Sensor TPS
3	RA1/AN1	AN1		Sensot ACT
4	RA2/AN2/VREF-/CVREF	AN2		Sensor MAP após conversor FxV
5	RA3/AN3/VREF+	AN3		Sensor ECT
6	RA4/T0CKI/C1OUT/RCV	RA4		IGN – Linha 15
7	RA5/AN4/SS/HLVDIN/C2OUT	AN4	VBat	Tensão da Bateria
8	RE0/AN5/CK1SPP	RE0	LCD CMD	LCD RS (4)
9	RE1/AN6/CK2SPP	RE1	LCD CMD	LCD E (6)
10	RE2/AN7/OESPP			
11	VDD	VDD	+5V	Alimentação
12	VSS	VSS	0V	Alimentação
13	OSC1/CLKI	OSC1	XTAL	Cristal 20 Mhz
14	OSC2/CLKO/RA6	OSC2	XTAL	Cristal 20 Mhz
15	RC0/T1OSO/T13CKI	RC0	LED	Indicador Ignicao
16	RC1/T1OSI/CCP2/UOE	RC1	LED	Indicador de inicio de programa
17	RC2/CCP1/P1A			
18	VUSB			
19	RD0/SPP0	RD0	LCD Data	LCD D0 (7)
20	RD1/SPP1	RD1	LCD Data	LCD D1 (8)
21	RD2/SPP2	RD2	LCD Data	LCD D2 (9)
22	RD3/SPP3	RD3	LCD Data	LCD D3 (10)
23	RC4/D-/VM			
24	RC5/D+/VP			
25	RC6/TX/CK	RC6	LED	Indicador Injecao
26	RC7/RX/DT/SDO	RC7	LED	Indicador Bomba
27	RD4/SPP4	RD4	LCD Data	LCD D4 (11)
28	RD5/SPP5/P1B	RD5	LCD Data	LCD D5 (12)
29	RD6/SPP6/P1C	RD6	LCD Data	LCD D6 (13)
30	RD7/SPP7/P1D	RD7	LCD Data	LCD D7 (14)
31	VSS	VSS	0V	Alimentação
32	VDD	VDD	+5V	Alimentação
33	RB0/AN12/INT0/FLT0/SDI/SDA	INT0	HALL	Rotação Motor
34	RB1/AN10/INT1/SCK/SCL	INT1	LCD	Troca Tela no Display
35	RB2/AN8/INT2/VMO	RB2		saida para bomba
36	RB3/AN9/CCP2(1)/VPO	RB3		saida para valvula injetora
37	RB4/AN11/KBI0/CSSPP	RB4	Spout	saida para o modulo de ignicao
38	RB5/KBI1/PGM			
39	RB6/KBI2/PGC	PGC		Programação In-Circuit
40	RB7/KBI3/PGD	PGD		Programação In-Circuit

```
#include "_fuses.h"
#include "_pinagem_ecu.h"
#include "_MemoryMap.h"
#include "_ANA.h"
#include "_matriz.h"
#include "_BaseTempo.h"
```

Gráfico de dependência de inclusões para _PIP.c:



Funções

- void Trata_int_RB0 (void)

  Funcao de tratamento da interrupcao externa em RB0,
  sensivel a borda de descida ou subida do sinal do sensor Hall/PIP.

10.35.1 Descrição Detalhada

Tratamento do sinal do PIP.

Definição no arquivo _PIP.c.

10.36 _PIP.c

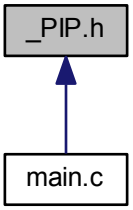
```
00001
00006 #include    "_fuses.h"
00007 #include    "_pinagem_ecu.h"
00008 #include    "_MemoryMap.h"
00009 #include    "_ANA.h"
00010 #include    "_matriz.h"
00011 #include    "_BaseTempo.h"
00012
00018 #int_ext
00019 void Trata_int_RB0(void){
00020     unsigned long int CALCULO;
00021
00022     if (INTEDG0){
00023         /* 1  = rising edge */
00024
00025         /* Medir Tempo do PIP */
00026         CALCULO = MemoryMap.RPM_RAW;
00027         CALCULO <<= 2;
00028         MemoryMap.PIP_TL = CALCULO;
00029         MemoryMap.PIP_TH = MemoryMap.PIP_T_TOT - MemoryMap.PIP_TL;
00030
00031         /* As leituras analógicas estao sincronizadas com o movimento do pistão */
00032         if(!MemoryMap.ANA_UpdateCicle){
00033             MemoryMap.ANA_UpdateCicle = 20;
00034             AD_Next_Port_Choice();
00035         }
00036
00037     } else {
00038
00039         /* Avanco de Ignicao */
00040         CALCULO = MemoryMap.PIP_TL;
00041         if (CALCULO > MemoryMap.IGN_AVANCO)
```

```
00042             CALCULO -= MemoryMap.IGN_AVANCO;
00043         if(!CALCULO) CALCULO++;
00044         CALCULO *= 150;
00045         CALCULO  = 0x00 - CALCULO;
00046         set_timer1(CALCULO);
00047         enable_interrupts(int_timer1);
00048 
00049         /* A injecao comeca com a descida do PIP */
00050         if(MemoryMap.PIP_SimNaoComandar){
00051             if(MemoryMap.PIP_SimNao){
00052                 Injetor_out = 1;
00053                 LED_Injetor = 0;
00054             }
00055             MemoryMap.PIP_SimNao = !MemoryMap.PIP_SimNao;
00056         } else{
00057             Injetor_out = 1;
00058             LED_Injetor = 0;
00059         }
00060 
00061         /* Sinal para o eletroinjetor */
00062         setup_timer_2(T2_DIV_BY_16, MemoryMap.INJ_T2_set, 15);
00063         enable_interrupts(int_timer2);
00064 
00065         /* 0 = falling edge
00066          * Limitado valor de 1 a 60 => 100 a 6000 RPMs
00067          */
00068         CALCULO = MemoryMap.RPM_RAW;
00069         MemoryMap.RPM_RAW = 0;
00070         CALCULO <<= 2;
00071         if(!CALCULO){
00072             //Não está girando, apenas a K15 está acionada
00073             MemoryMap.RPM = 0;
00074             MemoryMap.RPM_pointer=15;
00075             MemoryMap.PIP_T_TOT = 0;
00076         } else {
00077             if(CALCULO > 3000)  CALCULO = 3000;
00078             if(CALCULO < 50)    CALCULO = 50;
00079 
00080             /* Medir Tempo do PIP */
00081             MemoryMap.PIP_T_TOT = CALCULO;
00082             /* RPM */
00083             CALCULO = 3000/CALCULO;
00084             MemoryMap.RPM_pointer = CALCULO - 1;
00085             MemoryMap.RPM = CALCULO * 100;
00086         }
00087     }
00088 
00089     /* Troca int0 ext borda descida/subida */
00090     INTEDG0 = !INTEDG0;
00091 }
```

10.37 Referência do Arquivo _PIP.h

prototipos para _PIP.c

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



10.37.1 Descrição Detalhada

prototipos para _PIP.c

Definição no arquivo _PIP.h.

10.38 _PIP.h

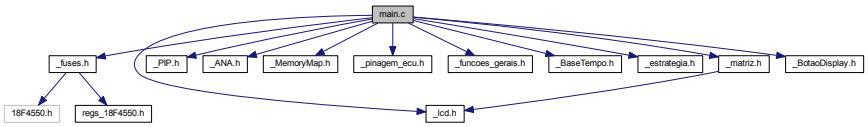
```
00001
```

10.39 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 "_lcd.h"
#include "_PIP.h"
#include "_ANA.h"
#include "_MemoryMap.h"
#include "_pinagem_ecu.h"
#include "_funcoes_gerais.h"
#include "_BaseTempo.h"
#include "_estrategia.h"
#include "_matriz.h"
#include "_BotaoDisplay.h"
```

Gráfico de dependência de inclusões para main.c:



Funções

- void InitHW (void)

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

  Inicializa circuitos externos ao microcontrolador.
- void Atualiza_Display (char)

  Mostra no display ou uma mensagem estática ou a rotação de duas ou mais.
- void Reset_Prog (void)

  Funcao de reset do programa.
- void main (void)

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

10.39.1 Descrição Detalhada

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

Definição no arquivo main.c.

10.39.2 Funções

10.39.2.1 void Atualiza_Display (char *pagina*)

Mostra no display ou uma mensagem estática ou a rotação de duas ou mais.

Parâmetros

in	*pagina*	referência do que é mostrado no display

Lista de Figuras

Figura 10.5: Aceleração a 2200 RPM e motor a 90C

Figura 10.6: ACT ECT TPS MAP

Lista de Figuras

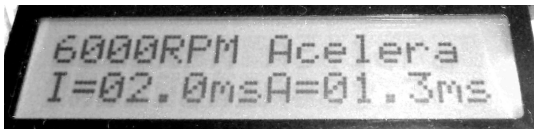


Figura 10.7: Avanço de ignição a 6000 RPM e motor a 90C

Figura 10.8: ACT ECT TPS MAP

Lista de Figuras

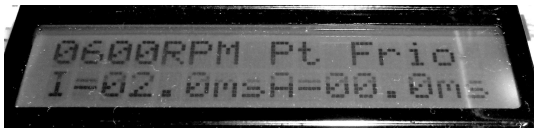


Figura 10.9: Partida a frio 10C e 1 bar

Figura 10.10: ACT ECT TPS MAP

Lista de Figuras

Figura 10.11: Partida a quente a 60C

Figura 10.12: ACT ECT TPS MAP

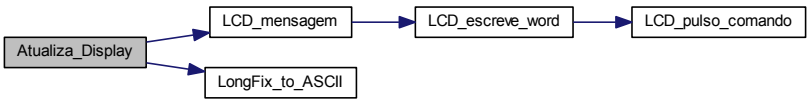
Definição na linha 447 do arquivo main.c.

Referenciado por main().

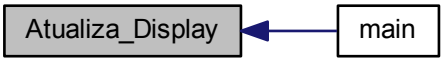
```
00447                                   {
00448     static char loop_LCD = 0;
00449     static char pagina_anterior = 0xFF;
00450     char dummy;
00451     char texto[1 + LCD_colunas] ; // Caracteres do display
00452 
00453     if(pagina_anterior != pagina){
00454         pagina_anterior = pagina;
00455 
00456         switch(pagina){
00457         case Pagina_BoasVindas:
00458             // Boas Vindas:
00459             strcpy(texto, LCD_BemVindo_L1);
00460             LCD_mensagem(LCD_L1, texto);
00461             strcpy(texto, LCD_BemVindo_L2);
00462             LCD_mensagem(LCD_L2, texto);
00463             break;
00464         case Pagina_K15:
00465             // Aguarda K15:
00466             strcpy(texto, LCD_EsperaK15_L1);
00467             LCD_mensagem(LCD_L1, texto);
00468             break;
00469         case Pagina_Tela1:
00470             // Mascara RPM, Ignição, Temperatura Motor, TPS:
00471             strcpy(texto, LCD_Mascara_L1);
00472             LCD_mensagem(LCD_L1, texto);
00473             strcpy(texto, LCD_Mascara_L2);
00474             LCD_mensagem(LCD_L2, texto);
00475             break;
00476         case Pagina_Tela2:
00477             // Valores analógicos ACT ECT TPS MAP:
00478             strcpy(texto, LCD_Mascara_L3);
00479             LCD_mensagem(LCD_L1, texto);
00480             strcpy(texto, LCD_Mascara_L4);
00481             LCD_mensagem(LCD_L2, texto);
00482             break;
00483         }
00484         // atualiza pagina apenas no próximo ciclo
00485         return;
00486     }
00487 
```

```
00488     if (pagina==Pagina_Tela1){
00489         switch (loop_LCD){
00490         case 0:
00491             dummy = sprintf(texto, "%04lu\0", MemoryMap.RPM);
00492             LCD_mensagem(LCD_L1 + 0, texto);
00493             loop_LCD = 1;
00494             break;
00495         case 1:
00496             strcpy(texto,EstrategiaTexto[MemoryMap.Estrategia_ID]);
00497             LCD_mensagem(LCD_L1 + 8, texto);
00498             loop_LCD = 2;
00499             break;
00500         case 2:
00501             LongFix_to_ASCII(MemoryMap.INJ, 1, texto);
00502             LCD_mensagem(LCD_L2 + 2, texto);
00503             loop_LCD = 3;
00504             break;
00505         case 3:
00506             LongFix_to_ASCII(MemoryMap.IGN_AVANCO, 1, texto);
00507             LCD_mensagem(LCD_L2 + 10, texto);
00508             loop_LCD = 0;
00509             break;
00510         default:
00511             loop_LCD = 0;
00512             break;
00513         }
00514     }
00515 
00516     if (pagina==Pagina_Tela2){
00517         switch (loop_LCD){
00518         case 0:
00519             // Air Charger Temperature Sensor
00520             dummy = sprintf(texto, "%03d", MemoryMap.ACT ); //MemoryMap.ACT
00521             LCD_mensagem(LCD_L1 + 4, texto);
00522             loop_LCD = 1;
00523             break;
00524         case 1:
00525             // Engine Coolant Temperature Sensor
00526             dummy = sprintf(texto, "%03d", MemoryMap.ECT);
00527             LCD_mensagem(LCD_L1 + 12, texto);
00528             loop_LCD = 2;
00529             break;
00530         case 2:
00531             // Throttle Position Sensor
00532             dummy = sprintf(texto, "%03d", MemoryMap.MAP_RAW);//MemoryMap.TPS
00533             LCD_mensagem(LCD_L2 + 3, texto);
00534             loop_LCD = 3;
00535             break;
00536         case 3:
00537             // Manifold Absolute Pressure Sensor
00538             LongFix_to_ASCII(MemoryMap.MAP, 2, texto);
00539             LCD_mensagem(LCD_L2 + 11, texto);
00540             loop_LCD = 0;
00541             break;
00542         default:
00543             loop_LCD = 0;
00544             break;
00545         }
00546     }
00547 }
```

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:



10.39.2.2 void InitHW (void)

Funcao de inicializacao com as configuracoes do uControlador.

set all ports as input

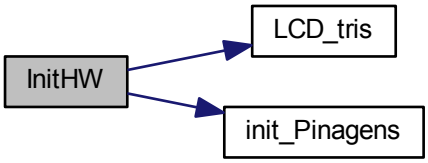
Definição na linha 307 do arquivo main.c.

Referenciado por main().

```
00307                   {
00309      TRISA = 0XFF;
00310      TRISB = 0XFF;
00311      TRISC = 0XFF;
00312      TRISD = 0XFF;
00313      TRISE = 0XFF;
00314
00315      /* desabilita interrupções */
00316      disable_interrupts(GLOBAL);
00317      /* desabilita interrupções individualmente */
00318      disable_interrupts(int_timer0);
00319      disable_interrupts(int_timer1);
00320      disable_interrupts(int_timer2);
00321      disable_interrupts(int_timer3);
00322      disable_interrupts(int_ext);
00323      disable_interrupts(int_ext1);
00324      disable_interrupts(int_ext2);
00325      disable_interrupts(INT_AD);
00326      disable_interrupts(INT_CCP1);
00327      disable_interrupts(INT_CCP2);
00328      disable_interrupts(INT_COMP);
00329      // disable_interrupts(INT_PSP);
00330      disable_interrupts(INT_RB);
00331      disable_interrupts(INT_RDA);
00332      disable_interrupts(INT_RTCC);
00333      disable_interrupts(INT_SSP);
00334      disable_interrupts(INT_TBE);
00335      /* desabilita temporizadores */
00336      T1CON = 0x00;
00337      T2CON = 0x00;
00338      T3CON = 0x00;
00339      /* AD Converter */
00340      setup_adc(ADC_OFF);
00341      setup_adc_ports(NO_ANALOGS);
00342      setup_psp(PSP_DISABLED); /* Parallel slave port */
00343      setup_spi(FALSE); /* SPI Disabled */
00344
00345      setup_comparator(NC_NC_NC_NC); /* Desliga comparadores internos */
00346      setup_vref(FALSE); /* Tensao de referencia  para o comparador */
00347
00348      /* agora set o que necessário */
00349      setup_adc_ports(AN0_TO_AN3); /* ConfiG analogicas: AN0-TPS>>AN1-TEMP */
00350      setup_adc(ADC_CLOCK_DIV_32); /* Configura divisao do clock ADC */
00351      enable_interrupts(INT_AD);
00352      setup_psp(PSP_DISABLED); /* PSP nao utilizado */
00353      setup_spi(FALSE); /* SPI nao utilizado */
00354
00355      /* Timer 0 = base de tempo fixo em 100us */
00356      setup_timer_0(RTCC_INTERNAL | RTCC_DIV_2);
00357      set_timer0(TMR_0_Prescaler);
00358      enable_interrupts(int_timer0);
00359
00360      /* Timer 1 Tempo de ignicao SPOUT em 100us */
00361      setup_timer_1(T1_INTERNAL | T1_DIV_BY_8);
```

```
00362     set_timer1(0); /* MAX > 43ms */
00363     enable_interrupts(int_timer1);
00364
00365     /* Timer 2 Tempo de injecao
00366      * (mode,period,postscale)
00367      * mode = T2_DISABLED, T2_DIV_BY_1, T2_DIV_BY_4, T2_DIV_BY_16
00368      * period = 0-255 that determines when the clock value is reset
00369      * postscale = 1-16 that determines how many timer overflows before
00370      * div_16:15 = 20us
00371      * 1=20us 5=100us 200=4ms 255=5.1ms
00372      */
00373     setup_timer_2(T2_DIV_BY_16, Padrao_Inj_OffSet, 15);
00374     //enable_interrupts(int_timer2); no PIP.c
00375
00376     set_tris_a(0b10111111);
00377     set_tris_b(0b00000011); // Configuracao de I/O do PORTB
00378     set_tris_c(0b00000100); // Configuracao de I/O do PORTC
00379     LCD_tris();
00380     //set_tris_d(0b00000000); // Configuracao de I/O do PORTD LCD
00381     //set_tris_e(0b1100); // Configuracao de I/O do PORTE LCD
00382
00383     output_a(0x00); // Atribui PORT A em nivel logico 0
00384     output_b(0x00); // Atribui PORT B em nivel logico 0
00385     output_c(0x00); // Atribui PORT C em nivel logico 0
00386     //output_d(0x00); // Atribui PORT D em nivel logico 0
00387     //output_e(0x00); // Atribui PORT E em nivel logico 0
00388
00389     //set_adc_channel(0); // Seta RA0 como entrada analogica
00390
00391     //ext_int_edge(0, H_TO_L); // Configura int0 ext c borda descida
00392     INTEDG0 = 0;
00393     //enable_interrupts(int_ext); // habilita interrupcoes Externas (apos K15)
00394
00395     ext_int_edge(1, H_TO_L); // Configura botão LCD int1 ext com borda descida
00396
00397     init_Pinagens();
00398 }
```

Este é o diagrama das funções utilizadas por esta função:



Este é o diagrama das funções que utilizam esta função:

10.40 main.c

```
00001
```

```
00152 /*
00153  * Bibliotecas
00154  */
00155 #include    "_fuses.h"
00156 #include    "_lcd.h"
00157 #include    "_PIP.h"
00158 #include    "_ANA.h"
00159 #include    "_MemoryMap.h"
00160 #include    "_pinagem_ecu.h"
00161 #include    "_funcoes_gerais.h"
00162 #include    "_BaseTempo.h"
00163 #include    "_estrategia.h"
00164 #include    "_matriz.h"
00165 #include    "_BotaoDisplay.h"
00166
00167 /*
00168  * prototipos
00169  */
00170 void InitHW(void);
00171 void InitSystem(void);
00172 void Atualiza_Display(char);
00173 void Reset_Prog(void);
00174
00181 void main(void) {
00182     static char RPMpointer_Anterior;
00183     //static char LCDpagina_loop = 0;
00184     static char LCDpagina_atualizacoes = 0;
00185     char texto[1 + LCD_colunas] ; // Caracteres do display
00186     union uBitByteNibble i; // usado em loops
00187     char dummy;
00188
00189     InitHW(); // Congiguracoes iniciais do Pic
00190     InitSystem(); // Configuracoes de dispositivos externos
00191
00192     /* limpa variaveis */
00193     MemoryMap.IGN_AVANCO = 0;
00194     MemoryMap.RPM_RAW = 0;
00195     MemoryMap.RPM_pointer = 15;
00196     MemoryMap.Estrategias = 0;
00197
00198     /* Boas Vindas: */
00199     Atualiza_Display(Pagina_BoasVindas);
00200     delay_ms(2000);
00201
00202     //Animacao dos LEDS
00203     LED_Start = 0;
00204     LED_Ign = 0;
00205     LED_Injetor = 0;
00206     LED_Bomba = 0;
00207
00208     for (i.uByte=1; i.uByte<31; i.uByte=i.uByte<<1){
00209         LED_Start = i.bits.B0;
00210         LED_Ign = i.bits.B1;
00211         LED_Bomba = i.bits.B2;
00212         LED_Injetor = i.bits.B3;
00213         delay_ms(100);
00214     }
00215
00216     // Pronto
00217     LED_Ign = 1;
00218     LED_Injetor = 1;
00219     LED_Bomba = 1;
00220
00221     /* Aguarda K15 */
00222     LED_Start = 1;
00223     Atualiza_Display(Pagina_K15);
00224
00225     while (!Ign){
00226         //Aguarda Sinal Linha 15
00227         LED_Start = !LED_Start;
00228         delay_ms(300);
00229     }
00230     LED_Start = 0;// sempre aceso
00231
00232     // acerto inicial das variáveis para 700RPM
00233     MemoryMap.RPM_pointer = 7;
00234     Calcular_Estrategia();
00235     Calcular_Avanco_IGN();
00236     Calcular_Injecao();
00237
00238     enable_interrupts(int_ext); // habilita interrupcao Externa (apos K15)
00239     enable_interrupts(int_ext1); // habilita botão LCD troca tela
00240
00241     // Ligar bomba
00242     LED_Bomba = 0;
00243     Bomba = 1;
00244
```

```
00245     //aceita interrupcoes
00246     enable_interrupts(GLOBAL);
00247
00248     while (Ign) {
00249         if(!MemoryMap.Estrategias_CicloCalc){
00250             MemoryMap.Estrategias_CicloCalc = 200;
00251             Calcular_Estrategia();
00252         }
00253
00254         // Bomba de aceleração
00255         if(!MemoryMap.TPS_CicloCalc){
00256             MemoryMap.TPS_CicloCalc = 50;
00257             Calcular_Aceleracao();
00258         }
00259
00260         if (!MemoryMap.LCD_SHOW){
00261             MemoryMap.LCD_SHOW = 30;
00262             /* varredura de telas */
00263             if (LCDpagina_atualizacoes) LCDpagina_atualizacoes--;
00264
00265             if (!LCDpagina_atualizacoes){
00266                 LCDpagina_atualizacoes = 35;
00267                 //substituido para escolha por botão no lugar de tempo
00268                 //LCDpagina_loop++;
00269             }
00270
00271             switch(MemoryMap.LCD_Pagina){
00272             case 1:
00273                 Atualiza_Display(Pagina_Tela1);
00274                 break;
00275             case 2:
00276                 Atualiza_Display(Pagina_Tela2);
00277                 break;
00278             default:
00279                 MemoryMap.LCD_Pagina = 1;
00280             }
00281         }
00282
00283         /* Calculos que dependem da variação da rotação */
00284         if ( RPMpointer_Anterior != MemoryMap.RPM_pointer){
00285             RPMpointer_Anterior = MemoryMap.RPM_pointer;
00286             Calcular_Avanco_IGN();
00287             Calcular_Injecao();
00288         }
00289
00290         /* trick */
00291         #ASM
00292             NOP
00293             NOP
00294             NOP
00295         #ENDASM
00296     }
00297
00298     /* Sinal K15 = desligado */
00299     Reset_Prog();
00300
00301 } /* Final da função main */
00302
00307 void InitHW(void) {
00309     TRISA = 0XFF;
00310     TRISB = 0XFF;
00311     TRISC = 0XFF;
00312     TRISD = 0XFF;
00313     TRISE = 0XFF;
00314
00315     /* desabilita interrupções */
00316     disable_interrupts(GLOBAL);
00317     /* desabilita interrupções individualmente */
00318     disable_interrupts(int_timer0);
00319     disable_interrupts(int_timer1);
00320     disable_interrupts(int_timer2);
00321     disable_interrupts(int_timer3);
00322     disable_interrupts(int_ext);
00323     disable_interrupts(int_ext1);
00324     disable_interrupts(int_ext2);
00325     disable_interrupts(INT_AD);
00326     disable_interrupts(INT_CCP1);
00327     disable_interrupts(INT_CCP2);
00328     disable_interrupts(INT_COMP);
00329     // disable_interrupts(INT_PSP);
00330     disable_interrupts(INT_RB);
00331     disable_interrupts(INT_RDA);
00332     disable_interrupts(INT_RTCC);
00333     disable_interrupts(INT_SSP);
00334     disable_interrupts(INT_TBE);
00335     /* desabilita temporizadores */
00336     T1CON = 0x00;
```

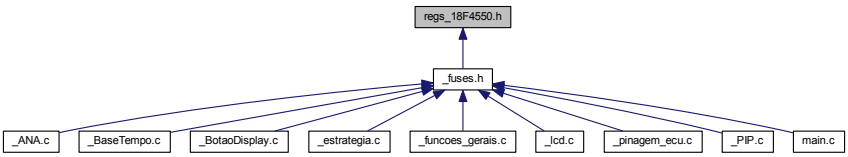
```
00337     T2CON = 0x00;
00338     T3CON = 0x00;
00339     /* AD Converter */
00340     setup_adc(ADC_OFF);
00341     setup_adc_ports(NO_ANALOGS);
00342     setup_psp(PSP_DISABLED); /* Parallel slave port */
00343     setup_spi(FALSE); /* SPI Disabled */
00344
00345     setup_comparator(NC_NC_NC_NC); /* Desliga comparadores internos */
00346     setup_vref(FALSE); /* Tensao de referencia  para o comparador */
00347
00348     /* agora set o que necessário */
00349     setup_adc_ports(AN0_TO_AN3); /* ConfiG analogicas: AN0-TPS>>AN1-TEMP */
00350     setup_adc(ADC_CLOCK_DIV_32); /* Configura divisao do clock ADC */
00351     enable_interrupts(INT_AD);
00352     setup_psp(PSP_DISABLED); /* PSP nao utilizado */
00353     setup_spi(FALSE); /* SPI nao utilizado */
00354
00355     /* Timer 0 = base de tempo fixo em 100us */
00356     setup_timer_0(RTCC_INTERNAL | RTCC_DIV_2);
00357     set_timer0(TMR_0_Prescaler);
00358     enable_interrupts(int_timer0);
00359
00360     /* Timer 1 Tempo de ignicao SPOUT em 100us */
00361     setup_timer_1(T1_INTERNAL | T1_DIV_BY_8);
00362     set_timer1(0); /* MAX > 43ms */
00363     enable_interrupts(int_timer1);
00364
00365     /* Timer 2 Tempo de injecao
00366      * (mode,period,postscale)
00367      * mode = T2_DISABLED, T2_DIV_BY_1, T2_DIV_BY_4, T2_DIV_BY_16
00368      * period = 0-255 that determines when the clock value is reset
00369      * postscale = 1-16 that determines how many timer overflows before
00370      * div_16:15 = 20us
00371      * 1=20us 5=100us 200=4ms 255=5.1ms
00372      */
00373     setup_timer_2(T2_DIV_BY_16, Padrao_Inj_OffSet, 15);
00374     //enable_interrupts(int_timer2); no PIP.c
00375
00376     set_tris_a(0b10111111);
00377     set_tris_b(0b00000011); // Configuracao de I/O do PORTB
00378     set_tris_c(0b00000100); // Configuracao de I/O do PORTC
00379     LCD_tris();
00380     //set_tris_d(0b00000000); // Configuracao de I/O do PORTD LCD
00381     //set_tris_e(0b1100); // Configuracao de I/O do PORTE LCD
00382
00383     output_a(0x00); // Atribui PORT A em nivel logico 0
00384     output_b(0x00); // Atribui PORT B em nivel logico 0
00385     output_c(0x00); // Atribui PORT C em nivel logico 0
00386     //output_d(0x00); // Atribui PORT D em nivel logico 0
00387     //output_e(0x00); // Atribui PORT E em nivel logico 0
00388
00389     //set_adc_channel(0); // Seta RA0 como entrada analogica
00390
00391     //ext_int_edge(0, H_TO_L); // Configura int0 ext c borda descida
00392     INTEDG0 = 0;
00393     //enable_interrupts(int_ext); // habilita interrupcoes Externas (apos K15)
00394
00395     ext_int_edge(1, H_TO_L); // Configura botão LCD int1 ext com borda descida
00396
00397     init_Pinagens();
00398 }
00399
00404 void InitSystem(void) {
00405     LCD_inicializacao(); // Funcao que inicializa o Display
00406 }
00407
00412 void Reset_Prog(void) {
00413     disable_interrupts(global); // Desabilita interrupcoes Globais
00414     Reset_CPU();
00415 }
00416
00447 void Atualiza_Display(char pagina) {
00448     static char loop_LCD = 0;
00449     static char pagina_anterior = 0xFF;
00450     char dummy;
00451     char texto[1 + LCD_colunas] ; // Caracteres do display
00452
00453     if(pagina_anterior != pagina){
00454         pagina_anterior = pagina;
00455
00456         switch(pagina){
00457         case Pagina_BoasVindas:
00458             // Boas Vindas:
00459             strcpy(texto, LCD_BemVindo_L1);
00460             LCD_mensagem(LCD_L1, texto);
00461             strcpy(texto, LCD_BemVindo_L2);
```

```
00462             LCD_mensagem(LCD_L2, texto);
00463             break;
00464         case Pagina_K15:
00465             // Aguarda K15:
00466             strcpy(texto, LCD_EsperaK15_L1);
00467             LCD_mensagem(LCD_L1, texto);
00468             break;
00469         case Pagina_Tela1:
00470             // Mascara RPM, Ignição, Temperatura Motor, TPS:
00471             strcpy(texto, LCD_Mascara_L1);
00472             LCD_mensagem(LCD_L1, texto);
00473             strcpy(texto, LCD_Mascara_L2);
00474             LCD_mensagem(LCD_L2, texto);
00475             break;
00476         case Pagina_Tela2:
00477             // Valores analógicos ACT ECT TPS MAP:
00478             strcpy(texto, LCD_Mascara_L3);
00479             LCD_mensagem(LCD_L1, texto);
00480             strcpy(texto, LCD_Mascara_L4);
00481             LCD_mensagem(LCD_L2, texto);
00482             break;
00483         }
00484         // atualiza pagina apenas no próximo ciclo
00485         return;
00486     }
00487
00488     if (pagina==Pagina_Tela1){
00489         switch (loop_LCD){
00490         case 0:
00491             dummy = sprintf(texto, "%04lu\0", MemoryMap.RPM);
00492             LCD_mensagem(LCD_L1 + 0, texto);
00493             loop_LCD = 1;
00494             break;
00495         case 1:
00496             strcpy(texto,EstrategiaTexto[MemoryMap.Estrategia_ID]);
00497             LCD_mensagem(LCD_L1 + 8, texto);
00498             loop_LCD = 2;
00499             break;
00500         case 2:
00501             LongFix_to_ASCII(MemoryMap.INJ, 1, texto);
00502             LCD_mensagem(LCD_L2 + 2, texto);
00503             loop_LCD = 3;
00504             break;
00505         case 3:
00506             LongFix_to_ASCII(MemoryMap.IGN_AVANCO, 1, texto);
00507             LCD_mensagem(LCD_L2 + 10, texto);
00508             loop_LCD = 0;
00509             break;
00510         default:
00511             loop_LCD = 0;
00512             break;
00513         }
00514     }
00515
00516     if (pagina==Pagina_Tela2){
00517         switch (loop_LCD){
00518         case 0:
00519             // Air Charger Temperature Sensor
00520             dummy = sprintf(texto, "%03d", MemoryMap.ACT ); //MemoryMap.ACT
00521             LCD_mensagem(LCD_L1 + 4, texto);
00522             loop_LCD = 1;
00523             break;
00524         case 1:
00525             // Engine Coolant Temperature Sensor
00526             dummy = sprintf(texto, "%03d", MemoryMap.ECT);
00527             LCD_mensagem(LCD_L1 + 12, texto);
00528             loop_LCD = 2;
00529             break;
00530         case 2:
00531             // Throttle Position Sensor
00532             dummy = sprintf(texto, "%03d", MemoryMap.MAP_RAW);//MemoryMap.TPS
00533             LCD_mensagem(LCD_L2 + 3, texto);
00534             loop_LCD = 3;
00535             break;
00536         case 3:
00537             // Manifold Absolute Pressure Sensor
00538             LongFix_to_ASCII(MemoryMap.MAP, 2, texto);
00539             LCD_mensagem(LCD_L2 + 11, texto);
00540             loop_LCD = 0;
00541             break;
00542         default:
00543             loop_LCD = 0;
00544             break;
00545         }
00546     }
00547 }
```

10.41 Referência do Arquivo regs_18F4550.h

Definições dos registros e bits dos registros.

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



10.41.1 Descrição Detalhada

Definições dos registros e bits dos registros.

Aviso

É necessário comentar / descomentar mantendo uma relação 1 para 1

Futuras Atividades Acrescentar registros do PIC conforme a necessidade

Data

1o semestre de 2013

Definição no arquivo regs_18F4550.h.

10.42 regs_18F4550.h

```
00001
00009 #BYTE PORTA = 0x0F80
00010     //#BIT RA0 = PORTA.0
00011     #BIT PAN0 = PORTA.0
00012
00013     //#BIT RA1 = PORTA.1
00014     #BIT PAN1 = PORTA.1
00015
00016     //#BIT RA2 = PORTA.2
00017     #BIT PAN2 = PORTA.2
00018     //#BIT VREFN = PORTA.2
00019     //#BIT CVREF = PORTA.2
00020
00021     #BIT PAN3 = PORTA.3
00022     //#BIT RA3 = PORTA.3
00023     //#BIT VREFP = PORTA.3
00024
00025     #BIT RA4 = PORTA.4
00026     //#BIT T0CKI = PORTA.4
00027     //#BIT C1OUT_PORTA = PORTA.4
00028
00029     #BIT RA5 = PORTA.5
00030     //#BIT AN4 = PORTA.5
00031     //#BIT SS = PORTA.5
00032     //#BIT NOT_SS = PORTA.5
00033     //#BIT LVDIN = PORTA.5
00034     //#BIT HLVDIN = PORTA.5
00035     //#BIT C2OUT_PORTA = PORTA.5
00036
00037     #BIT RA6 = PORTA.6
00038     //#BIT OSC2 = PORTA.6
00039     //#BIT CLKO = PORTA.6
00040
```

```
00041     #BIT RA7 = PORTA.7
00042     //#BIT OSC1 = PORTA.7
00043     //#BIT CLKI = PORTA.7
00044
00045 #BYTE PORTB = 0x0F81
00046     #BIT RB0 = PORTB.0
00047     //#BIT INT0 = PORTB.0
00048     //#BIT AN12 = PORTB.0
00049     //#BIT FLT0 = PORTB.0
00050
00051     #BIT RB1 = PORTB.1
00052     //#BIT INT1 = PORTB.1
00053     //#BIT AN10 = PORTB.1
00054
00055     #BIT RB2 = PORTB.2
00056     //#BIT INT2 = PORTB.2
00057     //#BIT AN8 = PORTB.2
00058
00059     #BIT RB3 = PORTB.3
00060     //#BIT CCP2_PORTB = PORTB.3
00061     //#BIT AN9 = PORTB.3
00062
00063     #BIT RB4 = PORTB.4
00064     //#BIT KBI0 = PORTB.4
00065     //#BIT AN11 = PORTB.4
00066
00067     #BIT RB5 = PORTB.5
00068     //#BIT KBI1 = PORTB.5
00069     //#BIT PGM = PORTB.5
00070
00071     #BIT RB6 = PORTB.6
00072     //#BIT KBI2 = PORTB.6
00073     //#BIT PGC = PORTB.6
00074
00075     #BIT RB7 = PORTB.7
00076     //#BIT KBI3 = PORTB.7
00077     //#BIT PGD = PORTB.7
00078
00079 #BYTE PORTC = 0x0F82
00080     #BIT RC0 = PORTC.0
00081     //#BIT T1OSO = PORTC.0
00082     //#BIT T13CKI = PORTC.0
00083
00084     #BIT RC1 = PORTC.1
00085     //#BIT T1OSI = PORTC.1
00086     //#BIT CCP2_PORTC = PORTC.1
00087
00088     #BIT RC2 = PORTC.2
00089     //#BIT CCP1 = PORTC.2
00090
00091     #BIT RC3 = PORTC.3
00092     //#BIT SCK = PORTC.3
00093     //#BIT SCL = PORTC.3
00094
00095     #BIT RC4 = PORTC.4
00096     //#BIT SDI = PORTC.4
00097     //#BIT SDA = PORTC.4
00098
00099     #BIT RC5 = PORTC.5
00100     //#BIT SDO = PORTC.5
00101
00102     #BIT RC6 = PORTC.6
00103     //#BIT TX = PORTC.6
00104     //#BIT CK = PORTC.6
00105
00106     #BIT RC7 = PORTC.7
00107     //#BIT RX = PORTC.7
00108
00109 #BYTE PORTD = 0x0F83
00110     #BIT RD0 = PORTD.0
00111     //#BIT PSP0 = PORTD.0
00112
00113     #BIT RD1 = PORTD.1
00114     //#BIT PSP1 = PORTD.1
00115
00116     #BIT RD2 = PORTD.2
00117     //#BIT PSP2 = PORTD.2
00118
00119     #BIT RD3 = PORTD.3
00120     //#BIT PSP3 = PORTD.3
00121
00122     #BIT RD4 = PORTD.4
00123     //#BIT PSP4 = PORTD.4
00124
00125     #BIT RD5 = PORTD.5
00126     //#BIT PSP5 = PORTD.5
00127     //#BIT P1B = PORTD.5
```

```
00128
00129     #BIT RD6 = PORTD.6
00130     //#BIT PSP6 = PORTD.6
00131     //#BIT P1C = PORTD.6
00132
00133     #BIT RD7 = PORTD.7
00134     //#BIT PSP7 = PORTD.7
00135     //#BIT P1D = PORTD.7
00136
00137 #BYTE PORTE = 0x0F84
00138     #BIT RE0 = PORTE.0
00139     //#BIT RD = PORTE.0
00140     //#BIT NOT_RD = PORTE.0
00141     //#BIT AN5 = PORTE.0
00142
00143     #BIT RE1 = PORTE.1
00144     //#BIT WR = PORTE.1
00145     //#BIT NOT_WR = PORTE.1
00146     //#BIT AN6 = PORTE.1
00147
00148     #BIT RE2 = PORTE.2
00149     //#BIT CS = PORTE.2
00150     //#BIT NOT_CS = PORTE.2
00151     //#BIT AN7 = PORTE.2
00152
00153     #BIT RE3 = PORTE.3
00154     //#BIT MCLR = PORTE.3
00155     //#BIT NOT_MCLR = PORTE.3
00156     //#BIT VPP = PORTE.3
00157
00158 #BYTE LATA = 0x0F89
00159     #BIT LATA0 = LATA.0
00160     #BIT LATA1 = LATA.1
00161     #BIT LATA2 = LATA.2
00162     #BIT LATA3 = LATA.3
00163     #BIT LATA4 = LATA.4
00164     #BIT LATA5 = LATA.5
00165     #BIT LATA6 = LATA.6
00166     #BIT LATA7 = LATA.7
00167
00168 #BYTE LATB = 0x0F8A
00169     #BIT LATB0 = LATB.0
00170     #BIT LATB1 = LATB.1
00171     #BIT LATB2 = LATB.2
00172     #BIT LATB3 = LATB.3
00173     #BIT LATB4 = LATB.4
00174     #BIT LATB5 = LATB.5
00175     #BIT LATB6 = LATB.6
00176     #BIT LATB7 = LATB.7
00177
00178 #BYTE LATC = 0x0F8B
00179     #BIT LATC0 = LATC.0
00180     #BIT LATC1 = LATC.1
00181     #BIT LATC2 = LATC.2
00182     #BIT LATC3 = LATC.3
00183     #BIT LATC4 = LATC.4
00184     #BIT LATC5 = LATC.5
00185     #BIT LATC6 = LATC.6
00186     #BIT LATC7 = LATC.7
00187
00188 #BYTE LATD = 0x0F8C
00189     #BIT LATD0 = LATD.0
00190     #BIT LATD1 = LATD.1
00191     #BIT LATD2 = LATD.2
00192     #BIT LATD3 = LATD.3
00193     #BIT LATD4 = LATD.4
00194     #BIT LATD5 = LATD.5
00195     #BIT LATD6 = LATD.6
00196     #BIT LATD7 = LATD.7
00197
00198 #BYTE LATE = 0x0F8D
00199     #BIT LATE0 = LATE.0
00200     #BIT LATE1 = LATE.1
00201     #BIT LATE2 = LATE.2
00202
00203 #BYTE TRISA = 0x0F92
00204     #BIT TRISA0 = TRISA.0
00205     #BIT TRISA1 = TRISA.1
00206     #BIT TRISA2 = TRISA.2
00207     #BIT TRISA3 = TRISA.3
00208     #BIT TRISA4 = TRISA.4
00209     #BIT TRISA5 = TRISA.5
00210     #BIT TRISA6 = TRISA.6
00211     #BIT TRISA7 = TRISA.7
00212
00213 #BYTE TRISB = 0x0F93
00214     #BIT TRISB0 = TRISB.0
```

```
00215     #BIT TRISB1 = TRISB.1
00216     #BIT TRISB2 = TRISB.2
00217     #BIT TRISB3 = TRISB.3
00218     #BIT TRISB4 = TRISB.4
00219     #BIT TRISB5 = TRISB.5
00220     #BIT TRISB6 = TRISB.6
00221     #BIT TRISB7 = TRISB.7
00222
00223 #BYTE TRISC = 0x0F94
00224     #BIT TRISC0 = TRISC.0
00225     #BIT TRISC1 = TRISC.1
00226     #BIT TRISC2 = TRISC.2
00227     #BIT TRISC3 = TRISC.3
00228     #BIT TRISC4 = TRISC.4
00229     #BIT TRISC5 = TRISC.5
00230     #BIT TRISC6 = TRISC.6
00231     #BIT TRISC7 = TRISC.7
00232
00233 #BYTE TRISD = 0x0F95
00234     #BIT TRISD0 = TRISD.0
00235     #BIT TRISD1 = TRISD.1
00236     #BIT TRISD2 = TRISD.2
00237     #BIT TRISD3 = TRISD.3
00238     #BIT TRISD4 = TRISD.4
00239     #BIT TRISD5 = TRISD.5
00240     #BIT TRISD6 = TRISD.6
00241     #BIT TRISD7 = TRISD.7
00242
00243 #BYTE TRISE = 0x0F96
00244     #BIT TRISE0 = TRISE.0
00245     #BIT TRISE1 = TRISE.1
00246     #BIT TRISE2 = TRISE.2
00247
00248 #BYTE INTCON = 0xFF2
00249     #BIT  RBIF   = INTCON.0
00250     #BIT  INT0IF = INTCON.1
00251     #BIT  TMR0IF = INTCON.2
00252     #BIT  RBIE   = INTCON.3
00253     #BIT  INT0IE = INTCON.4
00254     #BIT  TMR0IE = INTCON.5
00255     #BIT  PEIE   = INTCON.6
00256     #BIT  GIE    = INTCON.7
00257
00258 #BYTE INTCON2 = 0xFF1
00259     #BIT  INTEDG2 = INTCON2.4
00260     #BIT  INTEDG1 = INTCON2.5
00261     #BIT  INTEDG0 = INTCON2.6
00262
00263 #BYTE T1CON = 0x0FCD
00264 #BYTE T2CON = 0x0FCA
00265 #BYTE T3CON = 0x0FB1
00266
00267 //  Acrescentar registros do PIC conforme a necessidade ...
00268
```

Índice Remissivo