Anexo C

Código mais relevante

O código considerado mais relevante está indicado neste Anexo. Todo o código está bem documentado, mas a sua total compreensão exige um conhecimento mínimo da arquitectura do μC SAB80C167 e do seu *assembler*.

Incide principalmente no comando do inversor.

1 PWM.h
Várias definições e prototipagem das funções que controlam o inversor

```
/*****************************************************************************/
//       PWM.h
/*****************************************************************************/

#if !defined(PWM_c)
  #define EXTERNx2 extern
#else
  #define EXTERNx2 /* */
#endif

#pragma NOINIT

// numero de indices de modulacao disponiveis
// 'G_im_pwm' = {0 .. MAX_im}
#define MAX_im 973 // im= 0..973

// Macros com valores limite do periodo da rede admitido
  // com dt = 400ns
  // 20ms  50Hz
      #define N_PERIODO_CERTO  50000
  // 21ms  47.6Hz  -2.4
      #define N_PERIODO_MAXIMO 52500
  // 19ms  52.6Hz  +2.6
      #define N_PERIODO_MINIMO 47500

// Variaveis para controlar o indice de modulacao e a fase
  // Valor apontador para indice de molulacao (0 .. MAX_im)
  // 0 -> im mais baixo   MAX_im -> im mais alto
      EXTERNx2 unsigned int idata G_pwm_im;
  // Para controlar a fase do pwm
      EXTERNx2 unsigned int idata G_pwm_fase;
      EXTERNx2 unsigned int idata G_pwm_fase_calibra;
      EXTERNx2 unsigned int idata G_pwm_fase_atraso;

// Variaveis globais respeitantes a sincronizaçao com a rede
  // para indicar o período da rede
      EXTERNx2 unsigned int idata G_periodo_max;
      EXTERNx2 unsigned int idata G_periodo_min;
      EXTERNx2 unsigned int idata G_periodo;
      EXTERNx2 bit idata G_flag_sincr;

  // variaveis para indicar o numero de vezes que nao houve
  // detecao do zero da rede e erro no controlo de fase do PWM
      EXTERNx2 unsigned int idata G_erro_zero_rede;
      EXTERNx2 bit idata G_flag_erro_zero_rede;
      EXTERNx2 unsigned int idata G_erro_pwm_fase;
      EXTERNx2 bit idata G_flag_erro_pwm_fase;
      EXTERNx2 unsigned int idata G_erro_pwm_sync_bruta;
      EXTERNx2 bit idata G_flag_erro_pwm_sync_bruta;
      EXTERNx2 unsigned int idata G_contador1;
      EXTERNx2 unsigned int idata G_vetor_erro_sync[4];
      EXTERNx2 unsigned int idata* idata G_ptr_erro_sync;
      EXTERNx2 bit idata G_flag_erro_sync_grave;
      EXTERNx2 bit idata G_flag_erro_sync_normal;

// Apontadores para se aceder os valor de R1 e R2 de sync_regs
// e de R5 e R6 de pwm_regs a partir do C
  EXTERNx2 int idata *idata ptr_sync1_regs_R1;
  EXTERNx2 int idata *idata ptr_sync1_regs_R2;
  EXTERNx2 int idata *idata ptr_pwm_regs_R5;
  EXTERNx2 int idata *idata ptr_pwm_regs_R6;
/*****************************************************************************/
```

```
// Inicializa o comando do inversor
//    Inicializa o timer T6 para controlo da duração dos estados
//    Inicializa as portas das gates e do reset
//    Executa reset ao inversor (válido apenas depois de EINIT())
//    Inicializa o sistema de sincronização com a rede
//        Utiliza o timer T1, e o pino P2_0 (CC0) para a interrupcao de zero
//        Utiliza o timer T3 para controlo de fase
//    Inicializa variaveis de erro e comando
//    Inicializa a fase e o indice de modulacao
EXTERNx2 void inicializa_inversor(void);

// Inicializa a protecao dc
//    Inicializa as portas da gates e do reset do igbt
//    Executa reset ao igbt
EXTERNx2 void inicializa_protecao_dc(void);

// Funcao para arrancar com o pwm sincrono com a rede
EXTERNx2 void arranca_sincronizacao_e_pwm(void);

// Funcao para desligar o pwm e a sincronizacao com a rede
// Os igbts do inversor sao desligados
EXTERNx2 void desliga_sincronizacao_e_pwm_gates_off(void);
#define DESLIGA_SINC_PWM_GATES_OFF CC0IE=0; _nop_(); _nop_(); T1R=0; T3R=0; T6R=0;
PORTA_GATES = 0xFF;
#define DESLIGA_SINC_PWM_GATES_ON  CC0IE=0; _nop_(); _nop_(); T1R=0; T3R=0; T6R=0;
PORTA_GATES = 0x00;

// Funcao para activar o igbt de protecao
// !! Desliga a interrupcao de sobretensao !!
EXTERNx2 void activa_igbt_protecao(void);

// Funcao para activar o igbt de protecao
// !! Liga a interrupcao de sobretensao !!
EXTERNx2 void desliga_igbt_protecao(void);

// funcao para mudar a fase do pwm
// fase = -360 .. 360
EXTERNx2 void muda_fase_pwm(int fase);

// Funcao para analisar a gravidade dos erros de sincronismo com a rede (STATIC)
EXTERNx2 void trata_erro_sync(void);
/*****************************************************************************/

// Funcoes em assembler para o pwm e sincronismo

  // Rotina para iniciar os registos da rotina que implementa o PWM
  // Esta rotina deve ser executada sempre que o pwm arranca
      extern void far reset_pwm(void);

  // Rotina para mudar o indice de modulacao
  // NOTA: Antes de executar esta rotina e' necessario inicializar "G_im_pwm"
      extern void far muda_im_pwm(void);

  // Rotina para obter a correcao necessaria o periodo do PWM em funçao do
  // periodo da rede
  // E' necessario inicializar G_periodo
      extern void far calcula_correcao_periodo(void);

  // Rotina para corrigir o valor de controlo de fase em funcao do periodo
  // da rede
  // Atribui o valor corrigido ao timer T3 que controla a fase
      extern void far calcula_tempo_fase_atribui_T3(void);
/*****************************************************************************/

// Rotina que implementa a actualizacao dos registos que efectuam
// a correcao do periodo da onda gerada por PWM
// Coloca em 'R6 de pwm_regs' o 'R1 de sync1_regs'
// Coloca em 'R5 de pwm_regs' o dobro de 'R1 de sync1_regs'
      extern void far actualiza_registo_correcao_periodo_2(void);
```

2 PWM.c
Funções escritas em C para comando do inversor

```
//*****************************************************************************
//       PWM.C
//*****************************************************************************

#include <reg167.h>       /* special function register 80C166  */
#include <intrins.h>
#include <globais.h>
#include <gates_v1.h>
#include <misce_v1.h>

#define PWM_c
#include <pwm.h>

/*****************************************************************************/

// Inicializa o comando do inversor
//    Inicializa o timer T6 para controlo da duração dos estados
//    Inicializa as portas das gates e do reset
//    Executa reset ao inversor (válido apenas depois de EINIT())
//    Inicializa o sistema de sincronização com a rede
//        Utiliza o timer T1, e o pino P2_0 (CC0) para a interrupcao de zero
//        Utiliza o timer T3 para controlo de fase
//    Inicializa variaveis de erro e comando
//    Inicializa a fase e o indice de modulacao
void inicializa_inversor(void) {

  // timer GPT2_T6                                          Timer  200ns
      //          15 14 13 12 11  10    9     8     7    6   5 4,3  2 1 0
      // T6CON = T6SR x  x  x, x T6OTL T6OE T6UDE, T6UD T6R   T6M    T6I
      //           1  0  0  0, 0   1    0     0  ,  1    0    00,0   000
      T6CON = 0x8480;

      //          7    6  5 4 3 2  1 0
      // T6IC = T6IR T6IE   ILVL   GLVL
      //          0    1    1101    11    ILVL=13 (mais alto sem PEC)
      T6IC = 0x77;

  // prepara as portas de comando das gates
      PORTA_GATES= 0xFF; // desliga gates !logica negada!
      OD_PORTA_GATES= 0x0;
      D_PORTA_GATES= 0xFF;
      _nop_();_nop_(); PORTA_GATES= 0xFF; // desliga gates !logica negada!

  // porta de reset do inversor
      RST_IGBT_INV=1;  // !! logica negada !!
      D_RST_IGBT_INV = 1;
      OD_RST_IGBT_INV = 0;
      _nop_();_nop_(); RST_IGBT_INV=1;  // !! logica negada !!

  // executa o reset do inversor
      // NOTA: o inversor so e activado no fim da instrucao EINIT que
      //       liberta a linha /RSTOUT
      _nop_();
      RST_IGBT_INV=0;    // rst_activado !! logica negada !!
      espera_10ys(1000); // espera 10ms
      RST_IGBT_INV=1;    // rst_desactivado  !! logica negada !!

  // Levanta o sinal de fim de inicialização (/RSTOUT)
      _einit_();

  // inicializa o sistema de sincronizaçao com a rede

  // Timer T1
      //          15  14 13 12   11 10 9 8  7  6  5 4   3  2 1 0
```

```
      // T01CON =  - T1R  -  -, T1M   T1L , - T0R - -, T0M  T0L
      //           0   0  0  0,   0  0 0 0, x  x  x x,  x  x x x
      _bfld_(T01CON, 0xFF00, 0x0000);

      // valor inicial
      T1 = 0x0000;

      //          7   6   5 4 3 2  1 0
      // T1IC = T1IR T1IE   ILVL   GLVL
      //          0   1    10,10   01     nivel=10 prioridade de grupo=1
      T1IC = 0x69;

  // Timer T3
      //        15 14 13 12 11  10     9    8     7   6  5 4 3  2 1 0
      // T3CON = -  -  -  -  - T3OTL T3OE T3UDE T3UD T3R  T3M    T3I
      //         x  x  x  x  x   x     0    0  ,  1   0   00,0   000
      T3CON = 0x0080;

      //          7   6   5 4 3 2  1 0
      // T3IC = T3IR T3IE   ILVL   GLVL
      //          0   1    10,10   10     nivel=10 prioridade de grupo=2
      T3IC = 0x6A;


  // Interrupcao do pino P2.0  (CaptureCompare_0)
      //        15  14 13 12  11  10 9 8   7   6 5 4   3   2 1 0
      // CCM0= ACC3  CCMOD3  ACC2 CCMOD2 ACC1 CCMOD1 ACC0 CCMOD0
      //         X   X  X  X   X   X X X   X   X X X    0  0 0 1
      // Capture on positive  transitions at pin P2_0
      _bfld_(CCM0, 0x0000F, 0x0001);  // posi. transitions

      //          7     6    5 4 3 2  1 0
      // CC0IC= CC0IR CC0IE    ILVL   GLVL
      //          0     0      10,10   00     nivel=10 prioridade de grupo=0
      CC0IC = 0x28;

      // Pino P2_0 configurado como entrada
      DP2_0 = 0;

  // Inicia variaveis de sinalização de erro
      G_erro_zero_rede = 0;
      G_erro_pwm_fase = 0;
      G_erro_pwm_sync_bruta = 0;
      G_flag_erro_zero_rede = 0;
      G_flag_erro_pwm_fase = 0;
      G_flag_erro_pwm_sync_bruta = 0;

      G_contador1=0;
      G_vetor_erro_sync[0]=0; G_vetor_erro_sync[1]=0;
      G_vetor_erro_sync[2]=0; G_vetor_erro_sync[3]=0;
      G_ptr_erro_sync= G_vetor_erro_sync;
      G_flag_erro_sync_grave=0;
      G_flag_erro_sync_normal=0;

  // Inicializacao das variaveis de controlo/calibracao da fase
      G_pwm_fase_calibra = 47700;   // valor que deve ser retirado da calibracao
      G_pwm_fase_atraso = 2300;     // = T - G_pwm_fase_atraso

  // Reset das variaveis de controlo
      G_periodo_max = 0x0000;
      G_periodo_min = 0xFFFF;
      G_flag_sincr = 0;
}
/*****************************************************************************/


// Inicializa a protecao dc
//    Inicializa as portas da gates e do reset do igbt
//    Executa reset ao igbt
void inicializa_protecao_dc(void) {
  // porta de reset do igbt de protecao
```

```
  RST_IGBT_PROT=1;  // !! logica negada !!
  D_RST_IGBT_PROT = 1;
  OD_RST_IGBT_PROT = 0;
  _nop_();_nop_(); RST_IGBT_PROT=1;  // !! logica negada !!

  // prepara a porta de actuacao do igbt de protecao
  ACT_IGBT_PROT = 1;  // desactiva_igbt !! logica negada !!
  D_ACT_IGBT_PROT = 1;
  OD_ACT_IGBT_PROT = 0;
  _nop_();_nop_(); ACT_IGBT_PROT = 1;  // desactiva_igbt !! logica negada !!

  _nop_();

  // executa o reset do igbt de protecao
  RST_IGBT_PROT=0;   // rst_activado !! logica negada !!
  espera_10ys(1000); // espera 10ms
  RST_IGBT_PROT=1;   // rst_desactivado  !! logica negada !!
}
/*****************************************************************************/

// Funcao que detecta a passagem por zero da tensao da rede por interrupcao
// A prioridade deve ser igual a  ** "passagem_zero_rede_falhada" **
void passagem_zero_rede(void) interrupt interrupt_id = CC0INT using sync2_regs{
  T1R = 0;  // para o timer que mede o periodo da rede
  G_periodo = T1;
  T1 = 0;   // reload e arranque do timer
  T1R = 1;  //

  // Quando G_contador atingir Overflow passa a zero
  G_contador1++;

  if ( (G_periodo>N_PERIODO_MINIMO) && (G_periodo<N_PERIODO_MAXIMO) ) {
      // Verifica se ja houve interrupcao de T3
      // O intT3 para o timer: T3R=0
      // Se houve uma detecao de zero e o timer T3 ainda nao
      // gerou a int. anterior, houve PROBLEMAS
          if (T3R) {
              G_erro_pwm_fase++;
              G_flag_erro_pwm_fase=1;

              *G_ptr_erro_sync = G_contador1;
              if ( G_ptr_erro_sync++ >= (G_vetor_erro_sync + 3) )
                  G_ptr_erro_sync = G_vetor_erro_sync;
          }
          else {
              // calcula o valor de correcao do periodo de pwm
                  calcula_correcao_periodo();
              // Coloca no timer T3 o valor certo para a fase pretendida
              // Quando o timer T3 acabar de contar o tempo indicado coloca
              // o PWM na origem.
                  calcula_tempo_fase_atribui_T3();
                  // T3= G_pwm_fase;
                  T3R=1;
          }
  }
  else {
      G_erro_zero_rede++;
      G_flag_erro_zero_rede=1;

      *G_ptr_erro_sync = G_contador1;
      if ( G_ptr_erro_sync++ >= (G_vetor_erro_sync + 3) )
          G_ptr_erro_sync = G_vetor_erro_sync;
  }

  if (G_flag_sincr) {
      if( G_periodo > G_periodo_max )
          G_periodo_max = G_periodo;
      if( G_periodo < G_periodo_min )
          G_periodo_min = G_periodo;
  }
```

```
  G_flag_sincr=1;
}
/*****************************************************************************/

// Funcao para detectar um Overflow do timer T1 que ocorre se a passagem por
// zero da rede nao for detectada (ao fim de 26ms)
// A prioridade deve ser igual a  ** "passagem_zero_rede" **
void passagem_zero_rede_falhada(void) interrupt interrupt_id = T1INT {
  G_erro_zero_rede++;
  G_flag_erro_zero_rede=1;
  G_flag_sincr = 0;

  // Quando G_contador atingir Overflow passa a zero
  G_contador1++;

  *G_ptr_erro_sync = G_contador1;
  if ( G_ptr_erro_sync++ >= (G_vetor_erro_sync + 3) )
      G_ptr_erro_sync = G_vetor_erro_sync;

}
/*****************************************************************************/


// Funcao para arrancar com o pwm sincrono com a rede
void arranca_sincronizacao_e_pwm(void) {
  // Inicializa a fase e o indice de modulacao
      G_pwm_im = MAX_im;
      muda_im_pwm();
      reset_pwm();

      G_pwm_fase = G_pwm_fase_calibra;  // valor carregado para T3 (fase=0)

  // inicia correcao do periodo
      *ptr_sync1_regs_R1 = 0;
      *ptr_sync1_regs_R2 = 0;
      actualiza_registo_correcao_periodo_2();

  // arranca a sincronizacao permitindo a interrupcao
      _atomic_(0);
          CC0IR = 0;  // elimina eventuais interrupções pendentes
          _nop_();
          T1R = 1;    // liga o timer da sincronizacao
          CC0IE = 1;
      _endatomic_();
  // O timer T6 do pwm e' arrancado pela sincronizacao
}
/*****************************************************************************/

// Funcao para desligar o pwm e a sincronizacao com a rede
// Os igbts do inversor sao desligados
// NOTA: tambem existem macros para fazer isto
void desliga_sincronizacao_e_pwm_gates_off(void) {
  CC0IE = 0;  // desliga as interrupcoes da sincronizacao
  _nop_(); _nop_();
  T1R = 0;    // desliga o timer do periodo da rede
  T3R = 0;    // desliga o timer da fase
  T6R = 0;    // desliga o timer do PWM
  PORTA_GATES = 0xFF; // desliga gates !logica negada!
}
/*****************************************************************************/

// Funcao para activar o igbt de protecao
// !! Desliga a interrupcao de sobretensao !!
void activa_igbt_protecao(void) {
  _atomic_(0);
      RST_IGBT_PROT = 1; // desliga o reset do igbt !! logica negada !!
      ACT_IGBT_PROT = 0; // activa_igbt !! logica negada !!
  _endatomic_();
}

// Funcao para activar o igbt de protecao
```

```
// !! Liga a interrupcao de sobretensao !!
void desliga_igbt_protecao(void) {
  _atomic_(0);
      RST_IGBT_PROT = 0; // liga o reset do igbts !! logica negada !!
      ACT_IGBT_PROT = 1; // desliga_igbt !! logica negada !!
  _endatomic_();
  _nop_(); _nop_(); _nop_(); _nop_();
  RST_IGBT_PROT = 1; // desliga o reset do igbt !! logica negada !!
}
/*****************************************************************************/

// funcao para mudar a fase do pwm
// fase = -360 .. 360
void muda_fase_pwm(int fase) {

  if (fase < 0)
      fase = fase + 360;

  fase = fase * 139;

  if( fase > G_pwm_fase_atraso)
      G_pwm_fase = fase - G_pwm_fase_atraso;
  else
      G_pwm_fase = fase + G_pwm_fase_calibra;
}
/*****************************************************************************/

// Funcao para analisar a gravidade dos erros de sincronismo com a rede
#pragma STATIC
void trata_erro_sync(void) {
  unsigned int Nmin, Nmax;

  Nmin = G_vetor_erro_sync[0];
  Nmax = G_vetor_erro_sync[0];

  if( G_vetor_erro_sync[1] < Nmin )
      Nmin = G_vetor_erro_sync[1];
  if( G_vetor_erro_sync[1] > Nmax )
      Nmax = G_vetor_erro_sync[1];

  if( G_vetor_erro_sync[2] < Nmin )
      Nmin = G_vetor_erro_sync[2];
  if( G_vetor_erro_sync[2] > Nmax )
      Nmax = G_vetor_erro_sync[2];

  if( G_vetor_erro_sync[3] < Nmin )
      Nmin = G_vetor_erro_sync[3];
  if( G_vetor_erro_sync[3] > Nmax )
      Nmax = G_vetor_erro_sync[3];

  if( Nmin != 0 ) {
      if ( (Nmax-Nmin) < 4 ) // 4 erros consecutivos
          G_flag_erro_sync_grave = 1;
      else if ( (Nmax-Nmin) < 100 ) // 4 erros em 2s
          G_flag_erro_sync_normal = 1;
  }
}
#pragma REENTRANT
/*****************************************************************************/
```

3 PWM_V8.A66
Procedimentos escritos em *assembly* para comando do inversor

```
;*****************************************************************
;*   PWM_V8.A66
;*****************************************************************

$SEGMENTED CASE MOD167
$MODINF (51)

  NAME PWM_V8_0

      CAPREL  DEFR  0FE4AH
      T6CON   DEFR  0FF48H
      T6R     BIT   T6CON.6
      T6      DEFR  0FE48H

      T3CON   DEFR  0FF42H
      T3R     BIT   T3CON.6
      T3      DEFR  0FE42H

      P7      DEFR  0FFD0H
      P7_7    BIT   P7.7
      P7_6    BIT   P7.6


  ?ID0?PWM_V8_0  SECTION  DATA WORD 'IDATA0'
      EXTRN G_pwm_im : WORD
      EXTRN G_periodo : WORD
      EXTRN G_pwm_fase : WORD
      EXTRN G_erro_pwm_sync_bruta : WORD

      EXTRN ptr_sync1_regs_R2 : WORD
      EXTRN ptr_sync1_regs_R1 : WORD
      EXTRN ptr_pwm_regs_R6 : WORD
      EXTRN ptr_pwm_regs_R5 : WORD
  ?ID0?PWM_V8_0  ENDS

  ?PWM_TAB?PWM_V8_0 SECTION  DATA WORD
      EXTRN  G_tab_pwm_gates : WORD
      EXTRN  G_tab_ptr_pwm_im_pag : WORD
      EXTRN  G_tab_ptr_pwm_im_pof : WORD
      EXTRN  G_tab_pwm_1000 : WORD
  ?PWM_TAB?PWM_V8_0 ENDS

  ?BI0?PWM_V8_0  SECTION  BIT BIT 'BIT0'
      EXTRN  G_flag_erro_pwm_sync_bruta : BIT
  ?BI0?PWM_V8_0  ENDS


  SDATA         DGROUP  ?ID0?PWM_V8_0,SYSTEM

  ASSUME  DPP3 : SDATA

                REGDEF   R0 - R15
       pwm_regs REGBANK  R0 - R15
     sync1_regs REGBANK  R0 - R15

;-------------------------------------------------------------------------------

      ;*********************************************************
      ;*                                                       *
      ;*  Atribuicao dos registos 'pwm_regs'                   *
      ;*                                                       *
      ;*  R0 -> apontador para a tabela de valores de 'tempos' *
      ;*  R1 -> apontador para a tabela de valores de 'gates'  *
```

```
      ;*  R2 -> guarda o valor de 'gates' entre 2 chamadas       *
      ;*        (O valor de 'tempos' e' guardado directamente    *
      ;*         em CAPREL)                                      *
      ;*  R3 -> auxiliar de 'gera_pwm' (guarda os 'tempos')      *
      ;*                                                         *
      ;*  R5 -> valor para corrigir o periodo da onda gerada     *
      ;*        quando o valor anterior na tabela de 'tempos'    *
      ;*        e' um zero. E' o dobro de R6                     *
      ;*  R6 -> valor para corrigir o periodo da onda gerada     *
      ;*        ( e' somado ao tempo do estado (CAPREL & R3))    *
      ;*                                                         *
      ;*  R7, R8 , R9 -> registos auxiliares para 'muda_im_pwm'  *
      ;*                                                         *
      ;*  R10 -> guarda o inicio (offset) da tabela de 'tempos'  *
      ;*  R11 -> guarda o inicio (offset) da tabela de 'gates'   *
      ;*                                                         *
      ;*  R12, R13, R14, R15 -> auxiliares de coloca_PWM_origem  *
      ;*                                                         *
      ;*  DPP0 -> indica a pagina das tabelas actuais            *
      ;*                                                         *
      ;*********************************************************

      ;*********************************************************
      ;*                                                         *
      ;*  Atribuicao dos registos 'sync1_regs'                   *
      ;*                                                         *
      ;*  R1 -> valor para corrigir o periodo da onda gerada     *
      ;*        e' funcao do 'periodo da rede'                   *
      ;*                                                         *
      ;*  R2 -> valor para corrigir o periodo da onda gerada     *
      ;*        e' funcao do atraso/adianto do PWM face a' rede  *
      ;*        serve para ajustar R1                            *
      ;*                                                         *
      ;*  NOTA: o valor total de correcao e' dado por R1+R2      *
      ;*                                                         *
      ;*                                                         *
      ;* R13 -> auxiliar de 'acerta_fase_pwm'                    *
      ;* R15 -> auxiliar de 'inicia_sync1_regs'                  *
      ;*********************************************************

  ?PR?PWM_V8_0  SECTION  CODE WORD 'FCODE'

;*****************************************************************************
;**    Rotina que implementa o PWM (por interrupcao)                        **
;**    A interrupcao e' gerada pelo timer T6                                **
;** NOTA: Antes de permitir esta rotina e' necessario executar 'reset_pwm'  **
;**                                                                         **
;** Tempos de execucao:                                                     **
;**                    normal:  7.10ys com 1.95ys de sobreposicao           **
;**  zero ou reload de tempos:  7.85ys com 2.70ys de sobreposicao           **
;**  reload de tempos e gates:  8.60ys com 2.70ys de sobreposicao           **
;**                                                                         **
;**  As tabelas de 'tempos' nao podem ter tempo zero no inicio da tabela    **
;**                                                                         **
;**  Esta rotina assume que DPP0 indica a pagina onde estao as tabelas      **
;*****************************************************************************
  gera_pwm  PROC  INTERRUPT T6INT = 38 USING pwm_regs
  GLOBAL  gera_pwm

; preparacao da interrupcao
  SCXT CP,#pwm_regs
  NOP

; corpo da interrupcao

; Actualiza as gates nas portas do yC ([estado i] OR [estado i-1] SOBREPOSICAO)
; O reload do timer 6 e' efectuado automaticamente atravez de CAPREL.
      AND P7,R2

; Obtem o valor de 'tempo' para o proximo estado e Incrementa o apontador
      MOV R3,[R0+]
      JMP cc_NZ,?PWM00A1
;     Verifica se o tempo e' nulo. Se for passa para o proximo valor
```

```
;//******* O valor de tempos e' nulo (caso B)*******\\
?PWM00B1:
;     passa para o proximo valor de 'tempos' e 'gates'
      ADD R1,#2    ; incrementa o apontador de 'gates' (word= 2 posicoes)
      MOV R3,[R0+]
      JMP cc_NN,?PWM00B2
;     Verifica se chegou ao fim da tabela.
;     O final e' sinalizado com um valor negativo
;     Se for verdade volta para o inicio da tabela
          MOV R0,R10      ; reload do apontador
          MOV R3,[R0+]
?PWM00B2:

; actualiza as gates nas portas do yC ([estado i])
      MOV P7,R2

; Obtem o valores de 'gates' para o proximo estado
; Incrementa o apontador e verifica se chegou ao fim da tabela
      MOV R2,[R1+]
      JMP cc_NZ,?PWM00B3
;     Se chegou ao fim da tabela,
;     Actualiza o apontador da tabela e obtem o valor correspondente
          MOV R1,R11
          MOV R2,[R1+]
?PWM00B3:

; Actualiza o valor de Reload do timer para o proximo estado
      ADD R3,R5  ; soma o valor que corrige o valor do periodo da onda gerada
      MOV CAPREL,R3  ; reload efectuado directamente para CAPREL de T6

; finalizacao da interrupcao no caso B
  POP CP
  RETI

;//******* O valor de tempos nao e' nulo (caso A)*******\\
?PWM00A1:
      JMP cc_NN,?PWM00A2
;     Verifica se chegou ao fim da tabela.
;     O final e' sinalizado com um valor negativo
;     Se for verdade volta para o inicio da tabela
          MOV R0,R10      ; reload do apontador
          MOV R3,[R0+]
?PWM00A2:

; actualiza as gates nas portas do yC ([estado i])
      MOV P7,R2

; Obtem o valores de 'gates' para o proximo estado
; Incrementa o apontador e verifica se chegou ao fim da tabela
      MOV R2,[R1+]
      JMP cc_NZ,?PWM00A3
;     Se chegou ao fim da tabela,
;     Actualiza o apontador da tabela e obtem o valor correspondente
          MOV R1,R11
          MOV R2,[R1+]
?PWM00A3:

; Actualiza o valor de Reload do timer para o proximo estado
      ADD R3,R6  ; soma o valor que corrige o valor do periodo da onda gerada
      MOV CAPREL,R3  ; reload efectuado directamente para CAPREL de T6

; finalizacao da interrupcao no caso A
  POP CP
  RETI
  gera_pwm ENDP


;*************************************************************************
;**    Rotina para iniciar os registos da rotina que implementa o PWM         **
;**    Esta rotina deve ser executada sempre que o pwm arranca                **
```

```
;*******************************************************************************
  reset_pwm  PROC  FAR USING pwm_regs
  PUBLIC reset_pwm

; preparacao da rotina
  SCXT CP,#pwm_regs
  NOP

; corpo da rotina

; Inicializa a posicao das tabelas na pagina que as contem
  MOV R10, #POF (G_tab_pwm_1000)   ; tabela de 'tempos'
  MOV R11, #POF (G_tab_pwm_gates)  ; tabela de 'gates'
  MOV DPP0, #PAG (G_tab_pwm_1000)  ; pagina de 'tempos' e 'gates'

; Obtem os valores de arranque do pwm

;     Inicia os apontadores
      MOV R0,R10
      MOV R1,R11

;     Obtem o valor de 'tempo' para o 1º estado e Incrementa o apontador
      EXTP #PAG (G_tab_pwm_1000),#1
      MOV R3,[R0+]
      MOV CAPREL,R3  ; reload efectuado directamente para CAPREL de T6

;     Obtem o valor de 'gates' para o 1º estado e Incrementa o apontador
      EXTP #PAG (G_tab_pwm_gates),#1
      MOV R2,[R1+]

; Inicia o valor de correcao do periodo da onda gerada
  MOV R6,ZEROS ; coloca o valor de correcao a zero
  AND R5,ZEROS ; coloca o valor de correcao a zero

; Inicializa o timer T6 para gerar interrupcao mal comece a correr
  MOV T6,#1

; Inicia os registos de 'sync1_regs'
  CALL inicia_sync1_regs

; finalizacao da rotina
  POP CP
  RETS
  reset_pwm  ENDP


;*******************************************************************************
;**   Rotina para mudar o indice de modulacao                                 **
;**                                                                           **
;**   Muda os registos R0 , R10 e DPP0                                        **
;**   Utiliza e destroi os registos R7 , R8 e R9                              **
;**                                                                           **
;**   Muda o reload da tabela de tempos para o novo im                        **
;**   Muda o apontador R1 de modo a manter o pwm na mesma posicao             **
;**   Muda a pagina das tabelas para a pagina onde se encontra o im actual    **
;**                                                                           **
;** NOTA: Antes de executar esta rotina e' necessario inicializar "G_im_pwm"  **
;*******************************************************************************
  muda_im_pwm  PROC  FAR USING pwm_regs
  PUBLIC muda_im_pwm

; preparacao da rotina
  SCXT CP,#pwm_regs
  NOP

  MOV R7, WORD G_pwm_im  ; coloca o unsigned int em R7
  SHL R7, #1             ; multiplica R7 por 2 (cada apontador ocupa 2 bytes)

; R8 <- aponta para o offset de pagina do inicio da proxima tabela actual
; R9 <- aponta para a pagina da proxima tabela actual
;    As tabelas G_tab_ptr_pwm_im_pof e G_tab_ptr_pwm_im_pag devem estar na mesma
```

```
;    pagina.
  EXTP #PAG (G_tab_ptr_pwm_im_pof),#2
  MOV R8,[R7+#POF(G_tab_ptr_pwm_im_pof)]
  MOV R9,[R7+#POF(G_tab_ptr_pwm_im_pag)]

; Actualiza os registos R1 , R11 e R4 de forma indivisivel
  ATOMIC #4
      SUB R0,R10  ; R0 <- deslocamento a partir do inicio das tabelas
      ADD R0,R8   ; R0 <- apontador para a nova tabela
      MOV R10,R8  ; actualiza o novo reload de 'tempos'
      MOV DPP0,R9 ; actualiza a pagina das novas tabelas

; finalizacao da rotina
  POP CP
  RETS
  muda_im_pwm ENDP


;*****************************************************************************
;** Coloca o PWM na origem para o sincronizar                                **
;**                                                                          **
;** Utiliza os registos: R12, R13, R14, R15                                  **
;** Utiliza sem modificar os registos: R10, R6                               **
;** Actualiza os registos: R0, R1, R2                                        **
;*****************************************************************************
  coloca_PWM_origem  PROC  FAR USING pwm_regs
  PUBLIC  coloca_PWM_origem

; preparacao da rotina
  SCXT CP,#pwm_regs
  NOP

; Obtem o 1º valor de 'gates' e coloca o apontador a apontar para o 2º valor
; R12 <- apontador
; R13 <- 1º valor de gates
  MOV R12, #POF (G_tab_pwm_gates)
  EXTP #PAG (G_tab_pwm_gates),#1
  MOV R13,[R12+]

; Obtem o 1º valor de 'tempos' e coloca o apontador a apontar para o 2º valor
; R14 <- apontador
; R15 <- 1º valor de tempos
  MOV R14, R10
  EXTP #PAG (G_tab_pwm_gates),#1
  MOV R15,[R14+]
  ADD R15,R6

; Actualiza os registos R0, R1, R2, CAPREL e coloca 1 em T6 de forma a
; provocar de imediato um interrupcao de PWM
; A execucao das 7 linhas seguintes e' efectuada de forma indivisivel
  ATOMIC #4
  MOV R1,R12      ; apontador de 'gates' fica a apontar para o 2º elemento
  MOV R2,R13      ; R2 <- 'gates' do 1º estado
  MOV R0,R14      ; apontador de 'tempos' fica a apontar para o 2º elemento
  ATOMIC #2
  MOV CAPREL,R15  ; CAPREL <- 'tempo' do 1º estado
  MOV T6,#1       ; para provocar uma interrupcao de PWM

; finalizacao da rotina
  POP CP
  RETS
  coloca_PWM_origem  ENDP



;*****************************************************************************
;** Rotina que implementa a actualizacao dos registos que efectuam           **
;** a correcao do periodo da onda gerada por PWM                             **
;**                                                                          **
;** Coloca em 'R6 de pwm_regs' o 'R1 de sync1_regs'                          **
;** Coloca em 'R5 de pwm_regs' o dobro de 'R1 de sync1_regs'                 **
```

```
;**                                                                            **
;** NOTA: utiliza os registos do procedimento que a chamou                     **
;**       destroi o registo R5                                                 **
;**                                                                            **
;**    ******** ESTA ROTINA JA NAO E UTILIZADA ********                        **
;**    a funcao e implementada directamente em 'acerta_fase_pwm'               **
;********************************************************************************
  actualiza_registo_correcao_periodo_2  PROC FAR
  PUBLIC actualiza_registo_correcao_periodo_2

; corpo da rotina
  MOV R5,sync1_regs+1*2
  MOV pwm_regs+6*2, R5
  ROL R5, #1 ; multiplica R5 por 2
  MOV pwm_regs+5*2, R5

; finalizacao da rotina
  RETS
  actualiza_registo_correcao_periodo_2  ENDP


;********************************************************************************
;** Rotina para obter a correcao necessaria o periodo do PWM em funçao do      **
;** periodo da rede                                                            **
;**                                                                            **
;** Actualiza o registo R1 de sync1_regs                                       **
;**                                                                            **
;** NOTA: utiliza os registos do procedimento que a chamou                     **
;**       destroi os registo R1 e R2                                           **
;**       destroi os registos de multiplicacao/divisao                         **
;**       E' necessario inicializar G_periodo                                  **
;**                                                                            **
;** Demora 4.85ys                                                              **
;********************************************************************************
  calcula_correcao_periodo  PROC  FAR
  PUBLIC  calcula_correcao_periodo

; R1_sync1_regs = floor(753/2^16*G_periodo) - 574
; R1_sync1_regs = floor(753/2^16*G_periodo) - 575 <- melhor

  MOV  R1, WORD G_periodo
  MOV  R2, #753
  MULU R1, R2
  MOV  R1, MDH
  SUB  R1, #575
  MOV  sync1_regs+1*2, R1 ; R1 de sync1_regs = R1

; finalizacao da rotina
  RETS
  calcula_correcao_periodo ENDP


;********************************************************************************
;** Funcao para detectar um Underflow do timer T3 que ocorre quando            **
;** o pwm deve ser sincronizado. Mede o tempo que decorre desde uma passagem   **
;** por zero da rede, ate o PWM passar por zero. Controla assim a fase do PWM  **
;**                                                                            **
;** Coloca o PWM no ponto certo em relacao a' passagem por zero da rede        **
;**                                                                            **
;** Tambem ajusta o periodo do PWM de forma a mante-lo sincrono sem mudancas   **
;** bruscas                                                                    **
;**                                                                            **
;** Utiliza o registo R1                                                       **
;** Utiliza e actualiza o registo R2                                           **
;** Destroi o registo R13                                                      **
;**                                                                            **
;** Actualiza o registo R6 de pwm_regs                                         **
;********************************************************************************

; macro para indicar o numero de desvios possiveis da correcao de periodo de PWM
; R6_pwm_regs = R1 + R2 onde R2 e' limitado a +-DCT
  DCT LIT '2'
```

```
  acerta_fase_pwm  PROC  INTERRUPT T3INT = 35 USING sync1_regs
  GLOBAL  acerta_fase_pwm

; preparacao da interrupcao
  SCXT CP,#sync1_regs
  NOP

; corpo da interrupcao

; Para o timer T3 para nao haver multiplas interrupcoes no caso
; de falhar a sincronizacao
  BCLR    T3R

; arranca o pwm no caso de ainda nao ter arrancado
  BSET    T6R

; Verifica a sincronizacao do PWM
  JB  P7_6, ?P7_6e1
  JB  P7_7, ?P7_7e1
      ;//**** 00 esta mal sincronizado ****\\
          CALL coloca_PWM_origem
          MOV R2, ZEROS
          ; actualiza_registos_correcao_periodo (R5 e R6 de pwm_regs)
          MOV R13, R1
          MOV pwm_regs+6*2, R13  ; R6(pwm_regs) = R1(sync1_regs)
          ROL R13, #1 ; multiplica R13 por 2
          MOV pwm_regs+5*2, R13  ; R6(pwm_regs) = R1(sync1_regs) * 2
          ; incrementa a variavel que indica o numero de sincronizacoes brutas
          SUB G_erro_pwm_sync_bruta,ONES
          BSET G_flag_erro_pwm_sync_bruta
      JMP cc_UC, ?FIM_AFP
?P7_7e1:
      ;//**** 10 esta adiantado ****\\
          CMP R2, #DCT
          JMP cc_SGE, ?FIM_AFP
          ; se R2 e' menor que 2 incrementa R2 em 1 e
          ; actualiza_registos_correcao_periodo (R6 de pwm_regs)
              SUB R2, ONES ; R2 = R2 + 1
              MOV R13, R1
              ADD R13, R2
              MOV pwm_regs+6*2, R13 ; R6(pwm_regs) = R1(sync1_regs)+R2(sync1_regs)
              ROL R13, #1 ; multiplica R13 por 2
              MOV pwm_regs+5*2, R13 ; R5(pwm_regs) =
2*(R1(sync1_regs)+R2(sync1_regs))

      JMP cc_UC, ?FIM_AFP
?P7_6e1:
  JB P7_7, ?FIM_AFP
      ;//**** 01 esta atrasado ****\\
          CMP R2, #(-DCT)
          JMP cc_SLE, ?FIM_AFP
          ; se R2 e' maior que 2 decrementa R2 em 1 e
          ; actualiza_registos_correcao_periodo (R6 de pwm_regs)
              ADD R2, ONES ; R2 = R2 - 1
              MOV R13, R1
              ADD R13, R2
              MOV pwm_regs+6*2, R13 ; R6(pwm_regs) = R1(sync1_regs)+R2(sync1_regs)
              ROL R13, #1 ; multiplica R13 por 2
              MOV pwm_regs+5*2, R13 ; R5(pwm_regs) =
2*(R1(sync1_regs)+R2(sync1_regs))
      JMP cc_UC, ?FIM_AFP
?FIM_AFP:

; finalizacao da interrupcao
  POP CP
  RETI
  acerta_fase_pwm  ENDP

;*************************************************************************
;** Funcao para inicializar os registos de sync1_regs                    **
```

```
;** Tambem inicia os apontadores ptr_sync1_regs_R1..R2  e pwn_regs_R6..R5     **
;*******************************************************************************
  inicia_sync1_regs  PROC  FAR USING sync1_regs
  PUBLIC  inicia_sync1_regs

; preparacao da interrupcao
  SCXT CP,#sync1_regs
  NOP

  MOV R15, #(sync1_regs+1*2)
  MOV WORD ptr_sync1_regs_R1, R15

  MOV R15, #(sync1_regs+2*2)
  MOV WORD ptr_sync1_regs_R2, R15

  MOV R15, #(pwm_regs+6*2)
  MOV WORD ptr_pwm_regs_R6, R15

  MOV R15, #(pwm_regs+5*2)
  MOV WORD ptr_pwm_regs_R5, R15

  MOV R1, ZEROS
  MOV R2, ZEROS

; finalizacao da interrupcao
  POP CP
  RETS
  inicia_sync1_regs  ENDP

;*******************************************************************************
;** Rotina para corrigir o valor de controlo de fase em funcao do periodo     **
;** da rede.                                                                  **
;** Atribui o valor corrigido ao timer T3 que controla a fase                 **
;**                                                                           **
;** Actualiza o registo especial T3                                           **
;**                                                                           **
;** NOTA: utiliza os registos do procedimento que a chamou                    **
;**       destroi os registos R4 e R5                                         **
;**       destroi os registos de multiplicacao/divisao                        **
;**       E' necessario inicializar G_periodo e G_pwm_fase                    **
;**                                                                           **
;** Demora 6.75ys a executar                                                  **
;*******************************************************************************
  calcula_tempo_fase_atribui_T3  PROC  FAR
  PUBLIC  calcula_tempo_fase_atribui_T3

  ; T3 = (( (G_pwm_fase*G_periodo/2^16) * 42951 )/2^16)*2
  ; T3 = G_pwm_fase * G_periodo/Periodo50Hz

  ; corpo da rotina
  MOV R4,WORD G_pwm_fase
  MOV R5,WORD G_periodo
  MULU R4,R5
  MOV R4,MDH
  MOV R5,#42951
  MULU R4,R5
  MOV R4,MDH
  ROL R4,#1
  MOV T3, R4

  ; finalizacao da rotina
  RETS
  calcula_tempo_fase_atribui_T3 ENDP

;*******************************************************************************

  ?PR?PWM_V8_0  ENDS

END
```

4 Exemplificação das tabelas para o PWM

Aqui apenas se exemplifica o modo como as tabelas estão dispostas. Definem-se tabelas para os tempos de três índices de modulação (0.072; 0.500 e 1.000) e para o estado das *gates* dos IGBTs.

As tabelas são colocadas na memória do C167 na página **PWM_P1**. Deve ser notado que no programa real, são ocupadas 4 páginas de memória e cada página tem no seu início a tabela **G_tab_pwm_gates**.

```
$SEGMENTED CASE MOD167
$MODINF (51)

  NAME PWMPT_T

  ?PWM_TAB?PWMPT_T SECTION  DATA PAGE 'PWM_P1'

;-----------------------------------------------------
;-----------------------------------------------------

                  ;g3L g2L g1L g3H g2H g1H
 G_tab_pwm_gates  DW 00000000$10$101101$B  ; T1  G2H_G2L
                  DW 00000000$10$101011$B  ; T1  G3H_G2L
                  DW 00000000$10$101101$B  ; T1  G2H_G2L
                  DW 00000000$00$101110$B  ; T1  G1H_G2L
                  DW 00000000$00$101101$B  ; T1  G2H_G2L
                  DW 00000000$00$101011$B  ; T1  G3H_G2L
                  DW 00000000$00$101101$B  ; T1  G2H_G2L
                  DW 00000000$00$101110$B  ; T1  G1H_G2L
                      .    .    .    .    .    .
                      .    .    .    .    .    .
                      .    .    .    .    .    .
                      .    .    .    .    .    .
                      .    .    .    .    .    .
                  DW 00000000$00$110011$B  ; T6  G3H_G1L
                  DW 00000000$00$011011$B  ; T6  G3H_G3L
                  DW 00000000$00$101011$B  ; T6  G3H_G2L
                  DW 00000000$00$011011$B  ; T6  G3H_G3L
                  DW 00000000$00$110011$B  ; T6  G3H_G1L
                  DW 00000000$01$011011$B  ; T6  G3H_G3L
                  DW 00000000$01$101011$B  ; T6  G3H_G2L
                  DW 00000000$01$011011$B  ; T6  G3H_G3L
                  DW 0 ; O zero sinaliza o fim da tabela
                  PUBLIC G_tab_pwm_gates

;-----------------------------------------------------
;-----------------------------------------------------

  G_tab_pwm_0027    DW  1115     ; D1
                    DW  0        ; D2
                    DW  1111     ; D3
                    DW  0        ; D4
                       .    .    .    .    .    .
                       .    .    .    .    .    .
                       .    .    .    .    .    .
                    DW  0        ; D6
                    DW  1109     ; D7
                    DW  1115     ; D1
                    DW  0FFFFH ; o nº negativo sinaliza o fim da tabela
                    PUBLIC  G_tab_pwm_0027
```

```
  G_tab_pwm_0500    DW  642       ; D1
                    DW  921       ; D2
                    DW  585       ; D3
                    DW  155       ; D4
                    DW  608       ; D5
                    DW  826       ; D6
                    DW  560       ; D7
                    DW  307       ; D8
                    DW  586       ; D9
                    DW  715       ; D10
                    DW  544       ; D11
                    DW  452       ; D12
                    DW  574       ; D13
                    DW  589       ; D14
                    DW  539       ; D15
                    DW  589       ; D14
                    DW  574       ; D13
                    DW  452       ; D12
                    DW  544       ; D11
                    DW  715       ; D10
                    DW  586       ; D9
                    DW  307       ; D8
                    DW  560       ; D7
                    DW  826       ; D6
                    DW  608       ; D5
                    DW  155       ; D4
                    DW  585       ; D3
                    DW  921       ; D2
                    DW  642       ; D1
                    DW  0FFFFH ; o nº negativo sinaliza o fim da tabela
                    PUBLIC  G_tab_pwm_0500

  G_tab_pwm_1000    DW  156       ; D1
                    DW  1859      ; D2
                    DW  82        ; D3
                       .    .    .    .    .    .
                       .    .    .    .    .    .
                       .    .    .    .    .    .
                    DW  1859      ; D2
                    DW  156       ; D1
                    DW  0FFFFH ; o nº negativo sinaliza o fim da tabela
                    PUBLIC  G_tab_pwm_1000

;------------------------------------------------
;------------------------------------------------

; NOTA: como o indicador de pagina (DPP) que vai ser utilizado nas
;       funcoes de pwm e' o DPP0, os ultimos 2 bits de cada apontador
;       de im devem ser 0.

  ; tabela com o offset dentro da página da tabela do índice de modulação _xxxx
  G_tab_ptr_pwm_im_pof  DW POF(G_tab_pwm_0027)      ; 0
                        DW POF(G_tab_pwm_0500)      ; 1
                        DW POF(G_tab_pwm_1000)      ; 2
                    PUBLIC  G_tab_ptr_pwm_im_pof

; tabela com a página da tabela do índice de modulação _xxxx
  G_tab_ptr_pwm_im_pag  DW PAG(G_tab_pwm_0027)       ; 0
                        DW PAG(G_tab_pwm_0500)       ; 1
                        DW PAG(G_tab_pwm_1000)       ; 2
                    PUBLIC  G_tab_ptr_pwm_im_pag

;------------------------------------------------
;------------------------------------------------

  ?PWM_TAB?PWMPT_T ENDS

END
```