```
/*****************************************************************************************************/
/*                                                                                                   */
/*                                                                                                   */
/*    MprgSgd7.c : Σ -7S/7W  電  流  制  御  プログラム                                              */
/*                                                                                                   */
/*                                                                                                   */
/*****************************************************************************************************/
/*                                                                                                   */
/*    - Main Functions and Interrupts -                                                              */
/*      main( )   : 初  期  化 処理 & Round L oop                                                    */
/*      MpIntHost( ) : ホ  ス  ト 処理 (CPU側 S c a n A 完  了  を  起 動タイ  ミ  ン グとして Roundからコール)      */
/*      MpIntAD( ) : 電  流  ル  ー  プ 演  算  処  理  ( 電  流  検  出 完了割 り込みにより起 動)          */
/*      MpIntEnc( ) : Safety機  能  対  応  用  分 周  出  力  処  理  ( エ  ン  コ ーダ通信割り込 みにより起動)      */
/*                                                                                                   */
/*****************************************************************************************************/
/*                                                                                                   */
/*    ※  注  意  事 項※                                                                             */
/*                                                                                                   */
/*    JL-086用  μ  プ  ロ  グ  ラ  ム  は  C 言  語  ベ  ー  ス  で  記  述  し  、  専  用  コンパイラによりコンパイルを行うことで、     */
/*    機  械 語  ( 命  令 テ  ー  ブル)である「  M p r g J L0 86.c」を生成す る。                      */
/*    詳  細  は 、【900-*** -***】IP Des i g n e r イ  ン  ス  ト  ー  ル 、バ ー  ジ  ョンアップ手順.docを参照。      */
/*                                                                                                   */
/*    μ  プ  ロ  グ  ラ  ム  か  らア ク  セ  ス  す  る  H / W レ  ジ  ス  タ  仕  様  、 ア センブラ命令仕様、イ ント リンシック関数    */
/*    仕  様  に  つ  い  て  は  、 以下に 準ずる。                                                   */
/*       【 RB1400590 】  マ  イ  ク ロ  演  算  I P ソ  フ トウ ェアマニュアル                        */
/*       【 RB1400592 】Mercur y サ  ーボSoC(JL08 6 A )_サ  ー ボIP説明 書                           */
/*                                                                                                   */
/*****************************************************************************************************/
/*                                                                                                   */
/*    ☆  コ  ー  デ  ィ  ン グ ルール☆                                                               */
/*                                                                                                   */
/*    ①  レ  ジ  ス  タ  定 義  を  編  集  ( 追  加  、  削除等)する場合は 、  必  ず  「  M a keJL086Reg.xls」にて行う。      */
/*    ②  キ  ャ  ス  ト は  明  示  的  に  行  う  ( イ  ン  ト  リ  ン  シ  ッ  ク  関 数 の 場合、予期しない動作 にな る可能性大)。    */
/*    ③  シ  フ  ト  演  算  は  コ  メ  ン  ト  に  論  理  シ  フ ト か  算術  シ  フ  ト かを明記する。「>>」や 「 <<」を直接記述   */
/*      し た 場  合  、  コ ン パ イ ラ は 論  理  シ  フ  ト  と  し  て  コ  ン  パ イルする。算術シフトは イ ントリンシック   */
/*      関  数  を  使  用  す  る  必  要 が ある。※ 重要※                                           */
/*    ④  通  常  の  四  則  演  算  で  あ  っ  て  も  イ  ン  ト  リ ン シック関数を優 先して使用する。      */
```

```
/*    ⑤ コ メ ン ト は 半 角 英 数 、 全 角 漢 字 、 か な 、 カ タ カ ナ で記述する。全角英数、 半 角カタカナ禁止。
/*                                                  */
/************** Copyright (C) Yaskawa Electric Corporation ***************************************/
/*                                                  */
/*    Rev.1.00 : 2014.01.05  Y.Oka     ・  Σ -V-SD Rev.0. 0 A、 Σ-V Re v 3 . 1 5 を  べ ースに新規作 成 */
/*                        (Σ -7 はMpgDebug_024+++++++ + を  ベー ス)          */
/*                                                  */
/*                                                  */
/*                                                  */
/*************************************************************************************************/
#include "IxInst.h"
#include "MprgStruct.h"
#include "MpConstTbl.h"


/*************************************************************************************************/
/*                                                  */
/*    Version Infomation                                    */
/*                                                  */
/*************************************************************************************************/
#define MSW_VER     0x0001          /* ソ  フ  ト  バ  ー  ジョン設定              */
#define TST_VER     0x0000          /* テ  ス  ト  バ  ー  ジョン設定              */
#define YSP_VER     0x0000          /* Y仕  様  バ  ー  ジ ョン設定              */


/*************************************************************************************************/
/*                                                  */
/*    Multi Axis Select Switch for SGD7W                        */
/*                                                  */
/*************************************************************************************************/
#define xxx_MULTI_AXIS              /* 多  軸  処  理有効( S G D 7Wの場  合  「xxx_」を削 除 )     */
#ifdef  MULTI_AXIS
#define MAX_AXIS_NUM  2           /* 最  大  制  御  軸数定義                */
#endif


/*************************************************************************************************/
```

```
/*                                                  */
/*    H/W Access resister definitions                                  */
/*                                                  */
/*****************************************************************************************************/
/*---------------------------------------------------------------------------------------------------*/
/* Standerd definitions                                  */
/*---------------------------------------------------------------------------------------------------*/
int chess_storage(ISA0) ISA0;          /* 割 り 込 み 0 ジ ャ ン プ先アドレス          */
int chess_storage(ISA1) ISA1;          /* 割 り 込 み 1 ジ ャ ン プ先アドレス          */
int chess_storage(IL)   INTLVWR;         /* 割 込 み レ ベル設定                  */
int chess_storage(EIX)  EIX;           /* 割 り 込 み イ ネーブル                */
int chess_storage(DIX)  DIX;           /* 割 り 込 み デ ィ スエーブル             */

/*---------------------------------------------------------------------------------------------------*/
/* Extern definitions                                  */
/*---------------------------------------------------------------------------------------------------*/
extern int chess_storage(PFREG:0x6D0) OUTPT;  /* INT2 Output Port(共 通)             */
extern int chess_storage(PFREG:0x6D1) WDT1L;  /* WDT Trigger Port(共 通)             */
extern int chess_storage(PFREG:0x6D9) HSUR0;  /* ホ ス ト 指 令ポ ート0(共通)              */
extern int chess_storage(PFREG:0x6DA) HSUR1;  /* ホ ス ト 指 令ポ ート1(共通)              */
/*---------------------------------------------------------------------------------------------------*/
extern int chess_storage(PFREG:0x6D2) BBSET;  /* Soft BB and INT1L Enable Setting(Axis1)      */
extern int chess_storage(PFREG:0x6D3) CRST;   /* PWM Carrier Start & Clock Setting(Axis1)     */
extern int chess_storage(PFREG:0x6D8) SDMECLR;  /* Decimation filter alarm clear(Axis1)       */
extern int chess_storage(PFREG:0x6D9) ADSYNC; /* Sycn current AD cycle(Axis1)          */
extern int chess_storage(PFREG:0x7D2) BBSET_2;  /* Soft BB and INT1L Enable Setting(Axis2)      */
extern int chess_storage(PFREG:0x7D3) CRST_2; /* PWM Carrier Start & Clock Setting(Axis2)     */
extern int chess_storage(PFREG:0x7D8) SDMECLR_2;/* Decimation filter alarm clear(Axis2)       */
extern int chess_storage(PFREG:0x7D9) ADSYNC_2; /* Sycn current AD cycle(Axis2)          */
/*---------------------------------------------------------------------------------------------------*/
extern int chess_storage(PFREG:0x6D0) IuAD;   /* U相 電 流 フ ィ ードバ ックADC現在値(Axis 1)      */
extern int chess_storage(PFREG:0x6D1) IvAD;   /* V相 電 流 フ ィ ードバ ックADC現在値(Axis 1)      */
extern int chess_storage(PFREG:0x7D0) IuAD_2; /* U相 電 流 フ ィ ードバ ックADC現在値(Axis 2)      */
extern int chess_storage(PFREG:0x7D1) IvAD_2; /* V相 電 流 フ ィ ードバ ックADC現在値(Axis 2)      */
/*---------------------------------------------------------------------------------------------------*/
extern int chess_storage(PFREG:0x6DB) PWMOS;  /* PWM出 力 選択 (Axis1)                */
extern int chess_storage(PFREG:0x6DF) CRFRQ;  /* PWMキ ャ リ ア 周 波 数 1 6 ビ ットカウンタ設 定 ( Axis1) */
```

```
extern int chess_storage(PFREG:0x7DB) PWMOS_2;  /* PWM出　力　選択 (Axis2)                */
extern int chess_storage(PFREG:0x6DF) CRFRQ_2;  /* PWMキ　ャ　リ　ア 周 波　数　１６ビ　ットカウンタ設 定　( Axis2) */
extern int chess_storage(PFREG:0x6E7) PwmT0;  /* PWM三　角　波　比較値 0 (Axis1)           */
extern int chess_storage(PFREG:0x6E8) PwmT1;  /* PWM三　角　波　比較値 1 (Axis1)           */
extern int chess_storage(PFREG:0x6E9) PwmT2;  /* PWM三　角　波　比較値 2 (Axis1)           */
extern int chess_storage(PFREG:0x7E7) PwmT0_2;  /* PWM三　角　波　比較値 0 (Axis2)           */
extern int chess_storage(PFREG:0x7E8) PwmT1_2;  /* PWM三　角　波　比較値 1 (Axis2)           */
extern int chess_storage(PFREG:0x7E9) PwmT2_2;  /* PWM三　角　波　比較値 2 (Axis2)           */
/*-------------------------------------------------------------------------------------------*/
extern int chess_storage(PFREG:0x6DF) FLTSTAT;  /* 故　障　入力 (Axis1)                 */
extern int chess_storage(PFREG:0x6E1) FCCDAT; /* SVIP異　常　異　常状態 (Axis1)           */
extern int chess_storage(PFREG:0x7DF) FLTSTAT_2;/* 故　障　入力 (Axis2)                 */
extern int chess_storage(PFREG:0x7E1) FCCDAT_2; /* SVIP異　常　異　常状態 (Axis2)           */
/*-------------------------------------------------------------------------------------------*/
extern int chess_storage(PFREG:0x6F9) DIVSET; /* 分　周　機　能設定 (Axis1)                */
extern int chess_storage(PFREG:0x6FA) PCVS0;  /* PWM-パ　ル　ス　変　換 位置設定 (Axis1)        */
extern int chess_storage(PFREG:0x6FB) PCVS1;  /* PWM-パ　ル　ス　変　換 原　点補正1設定 (Axis1)        */
extern int chess_storage(PFREG:0x6BC) PCVS2;  /* PWM-パ　ル　ス　変　換 原　点補正2設定 (Axis1)        */

/*******************************************************************************************/
/*                                                 */
/*    static variable definitions for debug                    */
/*                                                 */
/*******************************************************************************************/
static  INITWK    IniWk;
static  COMWORKS  ComWk;

/*******************************************************************************************/
/*                                                 */
/*    ProtoType Definitions                            */
/*                                                 */
/*******************************************************************************************/
/*-------------------------------------------------------------------------------------------*/
/* Standerd Functions and Interrupts                        */
/*-------------------------------------------------------------------------------------------*/
void  MpIntHost( void );                  /* ホ　ス　ト　割　込　み　プ　ロトタイプ宣言　*/
void  MpIntAD( void ) property(isr);              /* 電　流　制　御　割　込　み　プ ロトタイプ宣言　*/
```

```
void  MpIntEnc( void );                    /* エ ン コ ー ダ 割 込 み プロトタイプ宣言 */
void  MpDataClear( MICRO_AXIS_HANDLE *AxisRsc );       /* マ イ ク ロ 用 デ ータクリア     */

/*-----------------------------------------------------------------------------------------*/
/* Inline Functions                                                                        */
/*-----------------------------------------------------------------------------------------*/
inline  USHORT  MpSQRT( ULONG );                  /* 平 方 根 演 算 処理(整数)          */
inline  SHORT MpOVMMODK( LONG, SHORT, SHORT, CSHORT* );   /* オ ー バ モ ジ ュ レ ーション処理    */
inline  void  InitSbb( SHORT );                  /* soft BB & INT1L設 定 初 期 化処理 */
inline  void  InitPWM( MICRO_AXIS_HANDLE* );          /* PWM設 定 初 期 化処理        */
inline  void  StartPWM( MICRO_AXIS_HANDLE*, SHORT );      /* PWM出 力 開 始処理          */
inline  void  SetPWM( MICRO_AXIS_HANDLE* );          /* PWM三 角 波 比 較 値設定処理      */
inline  void  ChangeCarrierFreq( MICRO_AXIS_HANDLE*, SHORT ); /* キ ャ リ ア 周 波 数 設定処理(各軸)    */
inline  void  ChangeCarrierFreqAll( MICRO_AXIS_HANDLE* );   /* キ ャ リ ア 周 波 数 設定処理(全軸)    */
inline  void  SdmErrClrRequest( USHORT );           /* Decimation Filter Error Clear  */
inline  void  CurAdSyncRequest( void );           /* 電 流A D サ イ ク ル同期要求     */
inline  void  ADConvDataLoad( MICRO_AXIS_HANDLE* );     /* 電 流 検 出 値 取得処理        */
inline  void  GetSvipErrorSts( MICRO_AXIS_HANDLE* );      /* SVIP異 常 状 態 取得処理       */

/*-----------------------------------------------------------------------------------------*/
/* 演 算 ラ イ ブラリ                                                                      */
/*-----------------------------------------------------------------------------------------*/
/* MlibMulgain32 */
inline  void  IxMulgain32( LONG *x, LONG u, LONG k, DWREG *wk );
/* MlibMulgainNolim */
inline  void  IxMulgainNolim( LONG *x, LONG u, LONG k, DWREG *wk );
/* 1次 ロ ー パ ス フィルタ */
inline  void  IxLpfilter1( LONG *x, LONG u, LONG k, DWREG *wk );
/* 2次 ノ ッ チ フィルタ */
inline  void  IxNxfilter2( LONG *x, LONG u, LONG k[5], LONG z[4], DWREG wk[4] );
/* 積 分 演算 */
inline  void  IxIntegral( LONG *x, LONG u, LONG k, LONG iu[2], DWREG *wk );
/* 二 乗 和 演算 */
inline  void  IxSquareSum( DLREG *x, LONG a, LONG b, DWREG *wk );/* <S18E> */


/*****************************************************************************************/
```

```
/*                                                                      */
/*    初 期 化 処 理                                                    */
/*                                                                      */
/************************************************************************************************/
#ifndef IPD_SIM         /* IPDesigner用 シ ミ ュ レ ー ションスイッチ                    */
void  main( void )      /* JL-086に 搭 載 す る プ ロ グ ラ ム を 作 成 する 場 合 はこちら で定義する   */
#else
void  MpStart( void )     /* コ ン パ イ ラ の み で シ ミ ュ レ ー シ ョ ン を行 な う 場合はこ ちらで定義する  */
#endif
{
  USHORT        ax_noR;
  MICRO_AXIS_HANDLE *AxisRscR;
  LONG         *BlkTrAdr;

  SHORT BbSetW;                     /* Soft BB and INT1L Enable Status    */
  SHORT DivSetW;                    /* 分 周 機 能設定                    */
  SHORT PoSet1W;                    /* パ ル ス 変 換 原点補正1           */
  SHORT PoSet2W;                    /* パ ル ス 変 換 原点補正2           */
  USHORT  uswk;                     /* ワ ー ク レ ジスタ                 */

/*----------------------------------------------------------------------------------------------*/
/*    バ ー ジ ョ ン設定                                                */
/*----------------------------------------------------------------------------------------------*/
  VerInfo.MswVer = MSW_VER;               /* ソ フ ト バ ー ジョン設定        */
  VerInfo.TstVer = TST_VER;               /* テ ス ト バ ー ジョン設定        */
  VerInfo.YspVer = YSP_VER;               /* Y仕 様 バ ー ジョン設定          */

/*----------------------------------------------------------------------------------------------*/
/*    Initialize Const Variables                                        */
/*----------------------------------------------------------------------------------------------*/
  True = 0x00000001;                    /* True = 1                     */
  False = 0x00000000;                   /* False = 0                    */

/*----------------------------------------------------------------------------------------------*/
/*    Get Axis Number from CPU                                          */
/*----------------------------------------------------------------------------------------------*/
  AxisNum = AxisHdl[0].AxisInfo.AxisNum;
```

```
/*-------------------------------------------------------------------------------------------------*/
/*    Set H/W Register Address Pointer                                          */
/*-------------------------------------------------------------------------------------------------*/
#ifdef  MULTI_AXIS                          /* 多  軸  処  理有効              */
  for( ax_noR = 0; (SHORT)ax_noR < AxisNum; ax_noR++ )
#else
  ax_noR = 0;
#endif
   {
     AxisRscR = &(AxisHdl[ax_noR]);
     if( ax_noR == 0 )
     {
       AxisRscR->SvIpRegR = (SVIP_READ_REG*)(0x600);
       AxisRscR->SvIpRegW = (SVIP_WRITE_REG*)(0x600);
     }
     else if( ax_noR == 1 )
     {
       AxisRscR->SvIpRegR = (SVIP_READ_REG*)(0x700);
       AxisRscR->SvIpRegW = (SVIP_WRITE_REG*)(0x700);
     }
   }

/*-------------------------------------------------------------------------------------------------*/
/*    Set Interrupts and Divide Pulse Output Setting                            */
/*-------------------------------------------------------------------------------------------------*/
  if( AxisHdl[0].EncIfV.BitIprm & UPGDIVOUT )
  { /* μ  プ  ロ  グ  ラ  ム  に  よ る 分  周  出 力有効(0軸目 の み 処理) */
    /* level(AD=3, INT1=4, HOST=0) */
    INTLVWR = 0x0043;
    ISA0 = (int)MpIntAD;
    ISA1 = (int)MpIntEnc;
    BbSetW = 0x2004;
    InitSbb( BbSetW );

/*-------------------------------------------------------------------------------------------------*/
/*    分  周  出  力  関  連  レ ジスタ設 定                                    */
```

```
/*-------------------------------------------------------------------------------------------------*/
    PCVS0 = AxisHdl[0].EncIfV.DivPls.s[0];  /* パ ル ス 変 換位置 (bit15-0)              */

    PoSet1W = AxisHdl[0].DivPlsV.PoSet1In;  /* MpUPDATE_DIVPOS()で 比 較 処 理 が あ る た め 残 しておく */
    PoSet2W = AxisHdl[0].DivPlsV.PoSet2In;  /* MpUPDATE_DIVPOS()で 比 較 処 理 が あ る た め 残 しておく */
    PCVS1 = PoSet1W;            /* パ ル ス 変 換 原点補正1 (bit15-0)          */
    PCVS2 = PoSet2W;            /* パ ル ス 変 換 原点補正2 (bit15-0)          */

    DivSetW = AxisHdl[0].DivPlsV.DivSetIn;  /* MpUPDATE_DIVPOS()で 比 較 処 理 が あ る た め 残 しておく */
    DIVSET = DivSetW;           /* 分 周 機 能設定                             */
  }
  else
  { /* μ プ ロ グ ラ ム に よ る分周出力無効 */
    /* level(AD=3, INT1=0, HOST=0) */
    INTLVWR = 0x0003;
    ISA0 = (int)MpIntAD;
    BbSetW = 0x0004;
    InitSbb( BbSetW );
  }

/*-------------------------------------------------------------------------------------------------*/
/*    Initilize PWM                                          */
/*-------------------------------------------------------------------------------------------------*/
  InitPWM( &AxisHdl[0] );

/*-------------------------------------------------------------------------------------------------*/
/*    Initialize variables                                    */
/*-------------------------------------------------------------------------------------------------*/
#ifdef  MULTI_AXIS                          /* 多 軸 処 理有効       */
  for( ax_noR = 0; (SHORT)ax_noR < AxisNum; ax_noR++ )
#else
  ax_noR = 0;
#endif
  {
    AxisRscR = &AxisHdl[ax_noR];

/*-------------------------------------------------------------------------------------------------*/
```

```
/*    Initialize Sin and Cos Table                                    */
/*--------------------------------------------------------------------------------------------------*/
    AxisRscR->SinTbl.Sin1.l = 0x0000;   /* sin( θ )      sin(0)     =  0.000 →  0000h  */
    AxisRscR->SinTbl.Cos1.l = 0x4000;   /* cos( θ )      cos(0)     =  1.000 →  4000h  */
    AxisRscR->SinTbl.Sin2.l = 0xC893;   /* sin( θ +2 π/3) sin(2π /3) = -0.866 →  C893h  */
    AxisRscR->SinTbl.Cos2.l = 0xE000;   /* cos( θ +2 π/3) cos(2π /3) = -0.500 →  E000h  */
    AxisRscR->SinTbl.Sin3.l = 0x376D;   /* sin( θ -2 π/3) sin(-2π /3)  =  0.866 →  376Dh  */
    AxisRscR->SinTbl.Cos3.l = 0xE000;   /* cos( θ -2 π/3) cos(-2π /3)  = -0.500 →  E000h  */

/*--------------------------------------------------------------------------------------------------*/
/*    Clear Register                                                */
/*--------------------------------------------------------------------------------------------------*/
    MpDataClear( AxisRscR );

/*--------------------------------------------------------------------------------------------------*/
/*    input CtrlStsIn, DLIM = QLIM = 0, output CtrlStsOut                    */
/*--------------------------------------------------------------------------------------------------*/
    AxisRscR->StsFlg.CtrlStsRW = AxisRscR->MicroIf.CtrlStsIn;
    AxisRscR->StsFlg.CtrlStsRW = ( AxisRscR->StsFlg.CtrlStsRW & DLIMI );
    AxisRscR->MicroIf.CtrlStsOut = AxisRscR->StsFlg.CtrlStsRW;
  }

/*--------------------------------------------------------------------------------------------------*/
/*    Start PWM                                                 */
/*--------------------------------------------------------------------------------------------------*/
  BbSetW = BbSetW & 0xFFFB;              /* Release Soft BB            */
  StartPWM( &AxisHdl[0], BbSetW );

/*--------------------------------------------------------------------------------------------------*/
/*    Start Interrupts                                              */
/*--------------------------------------------------------------------------------------------------*/
  EIX = 0x0;                          /* Interuput start            */



/****************************************************************************************************/
/*                                                 */
```

```
/*    ROUND Procedure                                          */
/*                                                           */
/*************************************************************************************************/
#ifndef IPD_SIM                    /* IPDesigner用  シ  ミ  ュ  レ  ー  シ ョンスイッチ      */
  while (1)
#endif
  {

/*-----------------------------------------------------------------------------------------------*/
/*    A/D error check and clear                                 */
/*-----------------------------------------------------------------------------------------------*/
    GetSvipErrorSts( &AxisHdl[0] );

/*-----------------------------------------------------------------------------------------------*/
/*    Host port check for host INT                               */
/*-----------------------------------------------------------------------------------------------*/
    if ( HSUR0 != 0x0 )
    {
      MpIntHost( );                /* ホ  ス  ト  割  込  み処理実行         */
    }

/*-----------------------------------------------------------------------------------------------*/
/*    Host port check for host INT2                              */
/*-----------------------------------------------------------------------------------------------*/
    if ( HSUR1 != 0x0 )
    {
      DIX = 0x0;                   /* disable interupt          */

#ifdef  MULTI_AXIS                     /* 多  軸  処  理有効            */
      for( ax_noR = 0; (SHORT)ax_noR < AxisNum; ax_noR++ )
#else
      ax_noR = 0;
#endif
      {
        AxisRscR = &AxisHdl[ax_noR];

        /* 位  相 &  モ  ー タ速度 */
```

```
        AxisRscR->PhaseV.Phase = AxisRscR->MicroIf.PhaseIn;
        AxisRscR->PhaseV.PhaseBuf = AxisRscR->MicroIf.PhaseIn;
        AxisRscR->Vcmp.MotSpd = AxisRscR->MicroIf.MotSpdIn;
        /* ト  ル  ク  制限値[2^24/MaxTrq] */
        AxisRscR->Trqctrl.TrqLimPlus = AxisRscR->MicroIf.TrqLimPlusIn;
        AxisRscR->Trqctrl.TrqLimMinus = AxisRscR->MicroIf.TrqLimMinusIn;
        /* 外  乱  ト ルク */
        AxisRscR->Trqctrl.TrqDistAftFil = AxisRscR->MicroIf.TrqDistAftFilIn;
        /* AVRゲ  イ ン */
        AxisRscR->Curctrl.AVRGain = AxisRscR->MicroIf.AVRGainIn;
      }

      EIX = 0x0;                    /* enable interupt           */
    }

/*-------------------------------------------------------------------------------------------------*/
/*    CPU_Roundか  ら  の  書  き  込  み  デ  ータ 受け取 り処理                          */
/*-------------------------------------------------------------------------------------------------*/
#ifdef  MULTI_AXIS                      /* 多  軸  処  理有効            */
    for( ax_noR = 0; (SHORT)ax_noR < AxisNum; ax_noR++ )
#else
    ax_noR = 0;
#endif
    {
      AxisRscR = &AxisHdl[ax_noR];

      /* 常  時  変  更可能 */
      AxisRscR->CurDet.IuOffset = AxisRscR->MicroIf.IuOffsetIn;
      AxisRscR->CurDet.IvOffset = AxisRscR->MicroIf.IvOffsetIn;
      AxisRscR->CurDet.IuGain = AxisRscR->MicroIf.IuGainIn;
      AxisRscR->CurDet.IvGain = AxisRscR->MicroIf.IvGainIn;

      /* オ  ン  ラ  イ  ン  デ  ータ受け取り */
      if( AxisRscR->BlockTr.TxNumRToAsic != (LONG)ZeroR )
      { /* 0で  な  い  場  合  、  デ ータ取り込みOK */
        BlkTrAdr = AxisRscR->BlockTr.TxDstRToAsic;
        block_cp( (LONG*)BlkTrAdr,                  /* デ  ー  タ  転 送命令    */
```

```
            (LONG*)&AxisRscR->BlockTr.TxDataRToAsic0,
            (unsigned int)AxisRscR->BlockTr.TxNumRToAsic );
      AxisRscR->BlockTr.TxNumRToAsic = (LONG)ZeroR;       /* 転　送　デ　ー　タ 数クリア　*/
    }

    /* BB中　の　み　変 更可能 */
    if( AxisRscR->StsFlg.FltStsW & 0x0400 )
    { /* BB中　の　場合 */
      /* d軸 q 軸　比　例 ゲ イ　ン　, 積分ゲイン */
      AxisRscR->Curctrl.GainKpd = AxisRscR->MicroIf.GainKpdIn;
      AxisRscR->Curctrl.GainKpq = AxisRscR->MicroIf.GainKpqIn;
      AxisRscR->Curctrl.GainKid = AxisRscR->MicroIf.GainKidIn;
      AxisRscR->Curctrl.GainKiq = AxisRscR->MicroIf.GainKiqIn;

      /* 変　調　率　リ ミット　*/
      AxisRscR->Vltctrl.Vmax = AxisRscR->MicroIf.VmaxIn;

      /* 弱　め　界磁 */
      AxisRscR->WeakFV.WfKpv = AxisRscR->MicroIf.WfKpvIn;
      AxisRscR->WeakFV.WfKiv = AxisRscR->MicroIf.WfKivIn;
      AxisRscR->WeakFV.WfV1max = AxisRscR->MicroIf.WfV1maxIn;
      AxisRscR->WeakFV.WfLpfGain = AxisRscR->MicroIf.WfLpfGainIn;
      AxisRscR->WeakFV.WfILimLpfGain = AxisRscR->MicroIf.WfILimLpfGainIn;
    }
  }
 }


 return; /* Unreachable */
}


/*****************************************************************************************************/
/*                                                   */
/*    HOST Interupt Procedure                                  */
/*                                                   */
/*****************************************************************************************************/
```

```
void  MpIntHost( void )
{
  USHORT        ax_noH;
  INT64       dlwk;         /* ワ  ー  ク  レ ジスタ(64)                */
  MICRO_AXIS_HANDLE *AxisRscH;

  SHORT swk0;               /* ホ  ス  ト  割  り  込  み  ワークレジスタ0 SHORT        */
  SHORT swk1;               /* ホ  ス  ト  割  り  込  み  ワークレジスタ2 SHORT        */
  LONG  lwk1;               /* ホ  ス  ト  割  り  込  み  ワークレジスタ1 LONG       */
  LONG  lwk2;               /* ホ  ス  ト  割  り  込  み  ワークレジスタ2 LONG       */
  LONG  lwk3;               /* ホ  ス  ト  割  り  込  み  ワークレジスタ3 LONG       */
  DWREG Ixlwk[4];             /* ホ  ス  ト  割  り  込  み  演  算  ラ イブラリ用レジス タ       */

  IniWk.IN_WK0++;   /* for debug counter */

  WDT1L = 0x1;              /* Watch dog set                */
//  OUTPT = 0x1;              /* デ  バ  ッ グ用                  */

/*-------------------------------------------------------------------------------------------------*/
/*    キ  ャ  リ  ア  周  波  数 切り替え 処理                                  */
/*-------------------------------------------------------------------------------------------------*/
  ChangeCarrierFreqAll( &AxisHdl[0] );

/*-------------------------------------------------------------------------------------------------*/
/*    input from host                                   */
/*-------------------------------------------------------------------------------------------------*/
  DIX = 0x0;                /* disable interupt                */

#ifdef  MULTI_AXIS              /* 多  軸  処  理有効                  */
  for( ax_noH = 0; (SHORT)ax_noH < AxisNum; ax_noH++ )
#else
  ax_noH = 0;
#endif
  {
    AxisRscH = &AxisHdl[ax_noH];

    /* 位  相 &  モ  ー タ速度 */
```

```
    AxisRscH->PhaseV.Phase = AxisRscH->MicroIf.PhaseIn;
    AxisRscH->PhaseV.PhaseBuf = AxisRscH->MicroIf.PhaseIn;
    AxisRscH->Vcmp.MotSpd = AxisRscH->MicroIf.MotSpdIn;
    /* ト  ル  ク  制限値[2^24/MaxTrq] */
    AxisRscH->Trqctrl.TrqLimPlus = AxisRscH->MicroIf.TrqLimPlusIn;
    AxisRscH->Trqctrl.TrqLimMinus = AxisRscH->MicroIf.TrqLimMinusIn;
    /* 外  乱  ト ル ク  , トル  ク  F F ,  リ ップル補償トルク[2^24/Ma xTrq] */
    AxisRscH->Trqctrl.TrqDistAftFil = AxisRscH->MicroIf.TrqDistAftFilIn;
    AxisRscH->Trqctrl.TrqFF = AxisRscH->MicroIf.TrqFFIn;
    AxisRscH->Trqctrl.RippleComp = AxisRscH->MicroIf.RippleCompIn;
    /* AVRゲ  イ ン */
    AxisRscH->Curctrl.AVRGain = AxisRscH->MicroIf.AVRGainIn;
    /* d軸 , q 軸  電  圧 指  令  (通常不使用) */
//    AxisRscH->VcmpV.VdRef = AxisRscH->AdinV.VdRefIn; /* 削  除 */
//    AxisRscH->VcmpV.VqRef = AxisRscH->AdinV.VqRefIn; /* 削  除 */
  }

  EIX = 0x0;                    /* enable interupt                  */


/*-----------------------------------------------------------------------------------------------*/
/*    Carrier Freq Change check : if( status & BB ) Carrier Freq. change              */
/*-----------------------------------------------------------------------------------------------*/
#ifdef  MULTI_AXIS                  /* 多  軸  処  理有効                    */
  for( ax_noH = 0; (SHORT)ax_noH < AxisNum; ax_noH++ )
#else
  ax_noH = 0;
#endif
  {
    AxisRscH = &AxisHdl[ax_noH];
    if ( AxisRscH->MicroIf.FccRst != 0)
    {
      SdmErrClrRequest( ax_noH );
      AxisRscH->StsFlg.ADRst = AxisRscH->MicroIf.FccRst;
      AxisRscH->MicroIf.FccRst = 0;
      IniWk.IN_WK0H++;    /* for debug counter */
    }
```

```
  }

/*-----------------------------------------------------------------------------------------------------*/
/*    Sync current A/D Cycle                                    */
/*-----------------------------------------------------------------------------------------------------*/
  CurAdSyncRequest( );

/*-----------------------------------------------------------------------------------------------------*/
/*    data clear while BB                                   */
/*-----------------------------------------------------------------------------------------------------*/
#ifdef  MULTI_AXIS                /* 多  軸  処  理有効                    */
  for( ax_noH = 0; (SHORT)ax_noH < AxisNum; ax_noH++ )
#else   /* ↑ #ifdef MULTI_AXIS */
  ax_noH = 0;
#endif    /* ↑ #ifdef MULTI_AXIS */
  {
    AxisRscH = &AxisHdl[ax_noH];
    if( AxisRscH->MicroIf.CtrlStsIn & BB )
    { /* BB中  の  場合 */
      DIX = 0x0;              /* disable interupt                */

      /* 制  御  用  変  数初期化 */
      MpDataClear( AxisRscH );
      /* キ  ャ  リ  ア  周 波数変更 */
      ChangeCarrierFreq( AxisRscH, ax_noH );

      EIX = 0x0;              /* enable interupt                 */
    }
    else
    { /* BE中  の  場合 */
/*-----------------------------------------------------------------------------------------------------*/
/*    ト  ル ク F F 入  力 足 し込み                                  */
/*-----------------------------------------------------------------------------------------------------*/
      AxisRscH->Trqctrl.TrqBfrComp = AxisRscH->MicroIf.TrqrefIn + AxisRscH->Trqctrl.TrqFF;

/*-----------------------------------------------------------------------------------------------------*/
/*    リ  ッ  プ  ル  補  正  ト ルク足し 込み                            */
```

```
/*-----------------------------------------------------------------------------------------------------*/
      AxisRscH->Trqctrl.TrqAftComp = AxisRscH->Trqctrl.TrqBfrComp + AxisRscH->Trqctrl.RippleComp;

/*-----------------------------------------------------------------------------------------------------*/
/*    notch filter 1st                                          */
/*-----------------------------------------------------------------------------------------------------*/
      if( (AxisRscH->MicroIf.FilterSwitch & NOTCHFIL1) == 0 )
      { /* フ  ィ  ル  タ  処理なし */
        /* フ  ィ  ル  タ 素通り */
        AxisRscH->NotchFil.Notch1Out = AxisRscH->Trqctrl.TrqAftComp;
        /* デ  ー  タ  クリア */
        AxisRscH->NotchFil.Notch1Value0 = ZeroR;
        AxisRscH->NotchFil.Notch1Value1 = ZeroR;
        AxisRscH->NotchFil.Notch1Value2 = ZeroR;
        AxisRscH->NotchFil.Notch1Value3 = ZeroR;
      }
      else
      {
        IxNxfilter2( &AxisRscH->NotchFil.Notch1Out,
               AxisRscH->Trqctrl.TrqAftComp,
               &AxisRscH->NotchFil.Notch1Gain0,
               &AxisRscH->NotchFil.Notch1Value0,
               Ixlwk );
      }

/*-----------------------------------------------------------------------------------------------------*/
/*    ノ  ッ  チ  フ ィ  ルタ 2段目                                    */
/*-----------------------------------------------------------------------------------------------------*/
      if( (AxisRscH->MicroIf.FilterSwitch & NOTCHFIL2) == 0 )
      { /* フ  ィ  ル  タ 素通り */
        AxisRscH->NotchFil.Notch2Out = AxisRscH->NotchFil.Notch1Out;
        /* デ  ー  タ  クリア */
        AxisRscH->NotchFil.Notch2Value0 = ZeroR;
        AxisRscH->NotchFil.Notch2Value1 = ZeroR;
        AxisRscH->NotchFil.Notch2Value2 = ZeroR;
        AxisRscH->NotchFil.Notch2Value3 = ZeroR;
      }
```

```
      else
      {
        IxNxfilter2( &AxisRscH->NotchFil.Notch2Out,
               AxisRscH->NotchFil.Notch1Out,
               &AxisRscH->NotchFil.Notch2Gain0,
               &AxisRscH->NotchFil.Notch2Value0,
               Ixlwk );
      }

/*-------------------------------------------------------------------------------------------------*/
/*    ノ　ッ　チ　フ ィ　ルタ 3段目                                          */
/*-------------------------------------------------------------------------------------------------*/
      if( (AxisRscH->MicroIf.FilterSwitch & NOTCHFIL3) == 0 )
      { /* フ　ィ　ル　タ 素通り */
        AxisRscH->NotchFil.Notch3Out = AxisRscH->NotchFil.Notch2Out;
        /* デ　ー　タ　クリア */
        AxisRscH->NotchFil.Notch3Value0 = ZeroR;
        AxisRscH->NotchFil.Notch3Value1 = ZeroR;
        AxisRscH->NotchFil.Notch3Value2 = ZeroR;
        AxisRscH->NotchFil.Notch3Value3 = ZeroR;
      }
      else
      {
        IxNxfilter2( &AxisRscH->NotchFil.Notch3Out,
               AxisRscH->NotchFil.Notch2Out,
               &AxisRscH->NotchFil.Notch3Gain0,
               &AxisRscH->NotchFil.Notch3Value0,
               Ixlwk );
      }

/*-------------------------------------------------------------------------------------------------*/
/*    ノ　ッ　チ　フ ィ　ルタ 4段目                                          */
/*-------------------------------------------------------------------------------------------------*/
      if( (AxisRscH->MicroIf.FilterSwitch & NOTCHFIL4) == 0 )
      { /* フ　ィ　ル　タ 素通り */
        AxisRscH->NotchFil.Notch4Out = AxisRscH->NotchFil.Notch3Out;
        /* デ　ー　タ　クリア */
```

```
        AxisRscH->NotchFil.Notch4Value0 = ZeroR;
        AxisRscH->NotchFil.Notch4Value1 = ZeroR;
        AxisRscH->NotchFil.Notch4Value2 = ZeroR;
        AxisRscH->NotchFil.Notch4Value3 = ZeroR;
      }
      else
      {
        IxNxfilter2( &AxisRscH->NotchFil.Notch4Out,
               AxisRscH->NotchFil.Notch3Out,
               &AxisRscH->NotchFil.Notch4Gain0,
               &AxisRscH->NotchFil.Notch4Value0,
               Ixlwk );
      }

/*-------------------------------------------------------------------------------------------------*/
/*    ノ ッ チ フ ィ ルタ 5段目                                                    */
/*-------------------------------------------------------------------------------------------------*/
      if( (AxisRscH->MicroIf.FilterSwitch & NOTCHFIL5) == 0 )
      { /* フ ィ ル タ 素通り */
        AxisRscH->NotchFil.Notch5Out = AxisRscH->NotchFil.Notch4Out;
        /* デ ー タ クリア */
        AxisRscH->NotchFil.Notch5Value0 = ZeroR;
        AxisRscH->NotchFil.Notch5Value1 = ZeroR;
        AxisRscH->NotchFil.Notch5Value2 = ZeroR;
        AxisRscH->NotchFil.Notch5Value3 = ZeroR;
      }
      else
      {
        IxNxfilter2( &AxisRscH->NotchFil.Notch5Out,
               AxisRscH->NotchFil.Notch4Out,
               &AxisRscH->NotchFil.Notch5Gain0,
               &AxisRscH->NotchFil.Notch5Value0,
               Ixlwk );
      }

/*-------------------------------------------------------------------------------------------------*/
/*    ノ ッ チ フ ィ ル タ 6 段 目 （モ ータ共振 抑制用 ）                                  */
```

```
/*-------------------------------------------------------------------------------------------------------*/
      if( (AxisRscH->MicroIf.FilterSwitch & NOTCHFIL6) == 0 )
      { /* フ  ィ  ル  タ 素通り */
        AxisRscH->NotchFil.Notch6Out = AxisRscH->NotchFil.Notch5Out;
        /* デ  ー  タ  クリア */
        AxisRscH->NotchFil.Notch6Value0 = ZeroR;
        AxisRscH->NotchFil.Notch6Value1 = ZeroR;
        AxisRscH->NotchFil.Notch6Value2 = ZeroR;
        AxisRscH->NotchFil.Notch6Value3 = ZeroR;
      }
      else
      {
        IxNxfilter2( &AxisRscH->NotchFil.Notch6Out,
               AxisRscH->NotchFil.Notch5Out,
               &AxisRscH->NotchFil.Notch6Gain0,
               &AxisRscH->NotchFil.Notch6Value0,
               Ixlwk );
      }
      AxisRscH->Trqctrl.TrqAftFil = AxisRscH->NotchFil.Notch6Out;

/*-------------------------------------------------------------------------------------------------------*/
/*    外  乱  ト  ル ク加算                                                */
/*-------------------------------------------------------------------------------------------------------*/
      AxisRscH->Trqctrl.TrqAddDist = AxisRscH->Trqctrl.TrqAftFil + AxisRscH->Trqctrl.TrqDistAftFil;
    }

/*-------------------------------------------------------------------------------------------------------*/
/*    制  御  ス  テ  ー タ  ス  処  理  ( ト  ル ク制限 中フラ グクリア)                          */
/*-------------------------------------------------------------------------------------------------------*/
    DIX = 0x0;                        /* disable interupt          */
    AxisRscH->StsFlg.CtrlStsRW = AxisRscH->MicroIf.CtrlStsIn;
    AxisRscH->StsFlg.CtrlStsRW = ( AxisRscH->StsFlg.CtrlStsRW & TLIMI );
    EIX = 0x0;                        /* enable interupt           */


/*******************************************************************************************************/
/*    用  途  選択                                          */
```

```
/*****************************************************************************************************************/
    if( AxisRscH->MotInfo.MotUse == SERVO  )
    {
/*****************************************************************************************************************/
/*    用　途　：　サーボ                                                                                         */
/*****************************************************************************************************************/
/*---------------------------------------------------------------------------------------------------------------*/
/*    トルクリミット                                                                                             */
/*---------------------------------------------------------------------------------------------------------------*/
      if( AxisRscH->Trqctrl.TrqAddDist >= ZeroR )
      { /* 正　側　トルクリミット */
        AxisRscH->Trqctrl.TrqAftLim = limit( AxisRscH->Trqctrl.TrqAddDist,
                         AxisRscH->Trqctrl.TrqLimPlus );
        if( AxisRscH->Trqctrl.TrqAftLim == AxisRscH->Trqctrl.TrqLimPlus )
        {
          AxisRscH->StsFlg.CtrlStsRW |= TLIM;         /* トルク制限中フラグセット */
        }
      }
      else
      { /* 負　側　トルクリミット */
        AxisRscH->Trqctrl.TrqAftLim = limit( AxisRscH->Trqctrl.TrqAddDist,
                         AxisRscH->Trqctrl.TrqLimMinus );
        if( (AxisRscH->Trqctrl.TrqAftLim + AxisRscH->Trqctrl.TrqLimMinus) == 0 )
        {
          AxisRscH->StsFlg.CtrlStsRW |= TLIM;         /* トルク制限中フラグセット */
        }
      }
    }
    else
    {
/*****************************************************************************************************************/
/*    用　途　：主軸                                                                                             */
/*****************************************************************************************************************/
      // 未　対応
    }

/*****************************************************************************************************************/
```

```
/*    モ  ー  タ  種 類選択                                                    */
/*************************************************************************************************************/
    if( AxisRscH->MotInfo.MotTypeSMIM == IM )
    {
/*************************************************************************************************************/
/*    イ  ン  ダ  ク  シ  ョ  ン モータ用 処理                                  */
/*************************************************************************************************************/
      // 未  対 応
    }
    else
    {
/*************************************************************************************************************/
/*    同  期  機  用 処理                                                      */
/*************************************************************************************************************/
/*-------------------------------------------------------------------------------------------------------------*/
/*    CPU_ScanCか  ら  の  書  き  込  み  デ  ータ 受け取 り処理                */
/*-------------------------------------------------------------------------------------------------------------*/
//IPMMの  場 合
//IdrefLim(LONG), IqrefLim(LONG), CtrlMode(SHORT), MaxCurLimInput(LONG)
      AxisRscH->Systems.DebugCtr1 = AxisRscH->BlockTr.TxNumCToAsic;
      AxisRscH->Systems.DebugCtr2 = (SHORT)ZeroR;
      if( AxisRscH->BlockTr.TxNumCToAsic != (SHORT)ZeroR )
      { /* CPU側  が  デ  ー  タ  書  き  込  み  完  了  の  た  め、マ イ ク ロレジスタ に渡す */
        DIX = 0x0;                      /* disable interupt      */

        AxisRscH->MicroIf.IdrefLimIn = AxisRscH->BlockTr.TxDataCToAsic0.l;
        AxisRscH->MicroIf.IqrefLimIn = AxisRscH->BlockTr.TxDataCToAsic1.l;
        AxisRscH->MicroIf.WfCtrlModeIn = AxisRscH->BlockTr.TxDataCToAsic2.s[0];
        AxisRscH->MicroIf.MaxCurLimIn = AxisRscH->BlockTr.TxDataCToAsic3.l;
        AxisRscH->BlockTr.TxNumCToAsic = (SHORT)ZeroR;

        AxisRscH->Systems.DebugCtr1++;

        EIX = 0x0;                      /* enable interupt       */
      }

/*-------------------------------------------------------------------------------------------------------------*/
```

```
/*    d,q軸　電　流　指　令　リ　ミ　ッ　ト　ロ ーパス フィル タ処理                              */
/*    弱　め　界　磁　制　御　電　流　リ　ミ　ッ　トローパ スフィ ルタ処理                          */
/*-------------------------------------------------------------------------------------------------*/
      if( (AxisRscH->MicroIf.CtrlSwitch & V_FB) != ZeroR )
      {
        /* WfIdrefLimLpf = フ　ィ　ル　タ出力 */
        IxLpfilter1( &AxisRscH->WeakFV.WfIdrefLimLpf,
               AxisRscH->MicroIf.IdrefLimIn,
               AxisRscH->WeakFV.WfILimLpfGain,
               Ixlwk );

        /* WfIqrefLimLpf = フ　ィ　ル　タ出力 */
//        IxLpfilter1( &AxisRscH->WeakFV.WfIqrefLimLpf,
//               AxisRscH->MicroIf.IqrefLimIn,
//               AxisRscH->WeakFV.WfILimLpfGain,
//               Ixlwk );

        /* MaxCurLimOutput = フ　ィ　ル　タ出力 */
        IxLpfilter1( &AxisRscH->WeakFV.MaxCurLimLpf,
               AxisRscH->MicroIf.MaxCurLimIn,
               AxisRscH->WeakFV.WfILimLpfGain,
               Ixlwk );

        DIX = 0x0;                      /* disable interupt       *//* <S18E> */
        AxisRscH->WeakFV.WfIdrefLim = AxisRscH->WeakFV.WfIdrefLimLpf;
//        AxisRscH->WeakFV.WfIqrefLim, AxisRscH->WeakFV.WfIqrefLimLpf );
        AxisRscH->MotInfo.MaxCurLim = AxisRscH->WeakFV.MaxCurLimLpf;
        EIX = 0x0;                      /* enable interupt        *//* <S18E> */
      }

/*-------------------------------------------------------------------------------------------------*/
/*    d軸　電　流　指 令計 算                                                                      */
/*-------------------------------------------------------------------------------------------------*/
      lwk1 = limit( AxisRscH->Trqctrl.TrqAftLim, (LONG)1 );          /* wkIntHost9 = sign(TrqAftLim) */
      lwk1 = AxisRscH->Trqctrl.TrqAftLim * lwk1;                   /* wkIntHost9 = |Trqref|  */

      lwk3 = lwk1 - AxisRscH->IntAdV.MaxTrq3;
```

```
if( lwk1 >= AxisRscH->IntAdV.MaxTrq3 )
{ /* |Trqref| >= MaxTrq[3] の  時 */
  IxMulgain32( &lwk2, lwk3, AxisRscH->IntAdV.IdE, Ixlwk );        /* wkIntHost0 = IdE*(|Trqref|-MaxTrq[3])  */
  lwk2 = add_limitf( lwk2, AxisRscH->IntAdV.IdrefOpt3 );          /* wkIntHost0 = IdE*(|Trqref|-MaxTrq[3]) + IdrefOpt[3]
  */
  lwk3 = limitz( lwk2, (LONG)No24bitM );                  /* wkIntHost1 = limit( wkIntHost0, -2^24～ 0 )  */
}
else
{
  lwk3 = lwk1 - AxisRscH->IntAdV.MaxTrq2;
  if( lwk1 >= AxisRscH->IntAdV.MaxTrq2 )
  { /* |Trqref| >= MaxTrq[2] の  時 */
    IxMulgain32( &lwk2, lwk3, AxisRscH->IntAdV.IdD, Ixlwk );      /* wkIntHost0 = IdD*(|Trqref|-MaxTrq[2])  */
    lwk2 = add_limitf( lwk2, AxisRscH->IntAdV.IdrefOpt2 );        /* wkIntHost0 = IdD*(|Trqref|-MaxTrq[2]) + IdrefOpt[2]
    */
    lwk3 = limitz( lwk2, (LONG)No24bitM );                /* wkIntHost1 = limit( wkIntHost0, -2^24～ 0 )  */
  }
  else
  {
    lwk3 = lwk1 - AxisRscH->IntAdV.MaxTrq1;
    if( lwk1 >= AxisRscH->IntAdV.MaxTrq1 )
    { /* |Trqref| >= MaxTrq[1] の  時 */
      IxMulgain32( &lwk2, lwk3, AxisRscH->IntAdV.IdC, Ixlwk );    /* wkIntHost0 = IdC*(|Trqref|-MaxTrq[1])  */
      lwk2 = add_limitf( lwk2, AxisRscH->IntAdV.IdrefOpt1 );      /* wkIntHost0 = IdC*(|Trqref|-MaxTrq[1]) + IdrefOpt[1]
      */
      lwk3 = limitz( lwk2, (LONG)No24bitM );              /* wkIntHost1 = limit( wkIntHost0, -2^24～ 0 )  */
    }
    else
    {
      lwk3 = lwk1 - AxisRscH->IntAdV.MaxTrq0;
      if( lwk1 >= AxisRscH->IntAdV.MaxTrq0 )
      { /* |Trqref| >= MaxTrq[0] の  時 */
        IxMulgain32( &lwk2, lwk3, AxisRscH->IntAdV.IdB, Ixlwk );  /* wkIntHost0 = IdB*(|Trqref|-MaxTrq[0])  */
        lwk2 = add_limitf( lwk2, AxisRscH->IntAdV.IdrefOpt0 );    /* wkIntHost0 = IdB*(|Trqref|-MaxTrq[0]) + IdrefOpt[0]
        */
        lwk3 = limitz( lwk2, (LONG)No24bitM );            /* wkIntHost1 = limit( wkIntHost0, -2^24～ 0 )  */
      }
```

```
            else
            { /* |Trqref| < MaxTrq[0] の  時 */
              IxMulgain32( &lwk2, lwk1, AxisRscH->IntAdV.IdA, Ixlwk );  /* wkIntHost0 = IdA*|Trqref|  */
              lwk3 = limitz( lwk2, (LONG)No24bitM );            /* wkIntHost1 = limit( wkIntHost0, -2^24～ 0 )  */
            }
          }
        }
      }
/*-------------------------------------------------------------------------------------------------*/
/*    磁  極  検  出 中  の  d 軸電流指令                                          */
/*-------------------------------------------------------------------------------------------------*/
      if( AxisRscH->IntAdV.PhaseReady == False )
      {
        /* モ  ー  タ  位  相  検 出  未  完 中 ( 磁  極  検  出  中  )はd 軸 電 流指令をゼ ロにする。 */
        lwk3 = (LONG)ZeroR;
      }
    }

/*-------------------------------------------------------------------------------------------------*/
/*    誘  起  電  圧 計算                                          */
/*-------------------------------------------------------------------------------------------------*/
    IxMulgain32( &lwk1, AxisRscH->Vcmp.MotSpd, AxisRscH->Vcmp.Phi, Ixlwk );     /* wkIntHost5 = ω   φ0        */
  }

/*-------------------------------------------------------------------------------------------------*/
/*    デ  ー  タをINT _ A Dに転送                                          */
/*-------------------------------------------------------------------------------------------------*/
  DIX = 0x0;                        /* disable interupt          */

#ifdef  MULTI_AXIS                      /* 多  軸  処  理有効            */
  for( ax_noH = 0; (SHORT)ax_noH < AxisNum; ax_noH++ )
#else
  ax_noH = 0;
#endif
  {
    AxisRscH = &AxisHdl[ax_noH];
```

```
    AxisRscH->Trqctrl.Trqref = AxisRscH->Trqctrl.TrqAftLim;   /* ト ル ク 指 令を代入       */
    AxisRscH->Curctrl.IdrefBeforeWF = lwk3;           /* Id*を 代 入          */
    AxisRscH->Vcmp.OmegaPhi = lwk1;               /* ω φ を 代入          */

    if( (AxisRscH->MicroIf.CtrlSwitch & V_FB) == 0 )
    { /* 弱 め 界 磁無効   = > 位 相 補 償 の 場 合(Σ-Ⅲ 以 前 のモータ ) */
      AxisRscH->WeakFV.WfIdref = AxisRscH->MicroIf.IdrefIn; /* WeakFV.WfIdref(reference)  */
    }
  }

  EIX = 0x0;                        /* enable interupt           */

/*****************************************************************************************************/
/*    Data Output for CPU                                    */
/*****************************************************************************************************/
#ifdef  MULTI_AXIS                      /* 多 軸 処 理有効            */
  for( ax_noH = 0; (SHORT)ax_noH < AxisNum; ax_noH++ )
#else
  ax_noH = 0;
#endif
  {
    AxisRscH = &AxisHdl[ax_noH];

    AxisRscH->MicroIf.TrqMon = AxisRscH->Trqctrl.Trqref;
    AxisRscH->MicroIf.TrqMonAftFil = AxisRscH->Trqctrl.TrqAftFil;
    AxisRscH->MicroIf.TrqMonBfrComp = AxisRscH->Trqctrl.TrqBfrComp;
    AxisRscH->MicroIf.IdrefMon = AxisRscH->Curctrl.Idref;
    AxisRscH->MicroIf.IqrefMon = AxisRscH->Curctrl.Iqref;
    AxisRscH->MicroIf.IdDetectMon = AxisRscH->Curctrl.IdDetect;
    AxisRscH->MicroIf.IqDetectMon = AxisRscH->Curctrl.IqDetect;
    AxisRscH->MicroIf.VdrefMon = AxisRscH->Curctrl.Vdref;
    AxisRscH->MicroIf.VqrefMon = AxisRscH->Curctrl.Vqref;

    AxisRscH->Systems.IntHostCtr++;
  }

  return;
```

```
}


/*****************************************************************************************************/
/*                                                   */
/*    AD Interupt Procedure                                  */
/*                                                   */
/*****************************************************************************************************/
void  MpIntAD( void ) property(isr)
{
  USHORT        ax_noI;
  INT64       dlwk;
  MICRO_AXIS_HANDLE *AxisRscI;

  /* Local Variables */
  CSHORT* pCtbl;              /* テ ー ブ ル ポ イ ン タ 用ワークレジスタ        */
  SHORT swk0;               /* 16bitワ ー ク レ ジスタ0              */
  SHORT swk1;               /* 16bitワ ー ク レ ジスタ1              */
  SHORT swk2;               /* 16bitワ ー ク レ ジスタ2              */
  SHORT swk3;               /* 16bitワ ー ク レ ジスタ3              */
  SHORT swk4;               /* 16bitワ ー ク レ ジスタ4              */
  SHORT swk5;               /* 16bitワ ー ク レ ジスタ5              */
  SHORT swk6;               /* 16bitワ ー ク レ ジスタ6              */
  SHORT swk7;               /* 16bitワ ー ク レ ジスタ7              */
  SHORT swk8;               /* 16bitワ ー ク レ ジスタ8              */
  SHORT swk9;               /* 16bitワ ー ク レ ジスタ9              */
  SHORT swk10;                /* 16bitワ ー ク レ ジスタ10             */
  SHORT swk11;                /* 16bitワ ー ク レ ジスタ11             */
  DWREG lwk0;               /* 32bitワ ー ク レ ジスタ0              */
  DWREG lwk1;               /* 32bitワ ー ク レ ジスタ1              */
  DWREG lwk2;               /* 32bitワ ー ク レ ジスタ2              */
  DWREG lwk3;               /* 32bitワ ー ク レ ジスタ2              */
  DWREG lwk4;               /* 32bitワ ー ク レ ジスタ4              */
  DWREG lwk5;               /* 32bitワ ー ク レ ジスタ2              */
  DWREG lwk6;               /* 32bitワ ー ク レ ジスタ6              */
  DWREG lwk7;               /* 32bitワ ー ク レ ジスタ7              */
```

```
  DWREG lwk8;                /* 32bitワ ー ク レ ジスタ8                */
  DWREG lwk9;                /* 32bitワ ー ク レ ジスタ9                */
  DWREG Ixlwk[4];              /* 32bit演 算 ラ イ ブ ラ リ用レジスタ          */
  DLREG dlwk0;                 /* 64bitワ ー ク レ ジスタ0                */
  OUTPT = 0x1;
  WDT1L = 0x0;                 /* Watch dog reset                  */
  IniWk.IN_WK1++;                /* for debug counter                */

/*-------------------------------------------------------------------------------------------------*/
/*    自 己 割 り 込 み禁 止設定                                          */
/*-------------------------------------------------------------------------------------------------*/
  INTLVWR &= 0x00F0;


/*************************************************************************************************/
/*    Current Control Input Procedure                                  */
/*************************************************************************************************/
#ifdef  MULTI_AXIS               /* 多 軸 処 理有効                    */
  for( ax_noI = 0; (SHORT)ax_noI < AxisNum; ax_noI++ )
#else
  ax_noI = 0;
#endif
  {
    AxisRscI = &AxisHdl[ax_noI];
/*-------------------------------------------------------------------------------------------------*/
/*    位 相 補 間 処理                                                */
/*-------------------------------------------------------------------------------------------------*/
    if( AxisRscI->PhaseV.PhaseBuf == AxisRscI->PhaseV.PhaseBufLast )
    { /* 位 相 補 間する */
      AxisRscI->PhaseV.Phase += AxisRscI->PhaseV.PhaseItplt;      /* 位 相 補 間 量 を足し込む */
    }
    else
    { /* 位 相 補 間 しない */
      IxMulgainNolim( &lwk0.l, AxisRscI->Vcmp.MotSpd, AxisRscI->PhaseV.CnvSpdToPhase, Ixlwk );
      AxisRscI->PhaseV.PhaseItplt = lwk0.s[0];            /* PhaseItplt = 位 相 補 間量 */
      AxisRscI->PhaseV.PhaseBufLast = AxisRscI->PhaseV.PhaseBuf;
    }
```

```
/*-------------------------------------------------------------------------------------------------*/
/*    sin, cos計  算  処 理                                          */
/*-------------------------------------------------------------------------------------------------*/
/*    sin1 = sinθ , sin2 = sin( θ - 2π/3), sin3 = si n(θ +2π/3 )                  */
/*    cos1 = cosθ , cos2 = cos( θ - 2π/3), cos3 = co s(θ +2π/3 )                  */
/*-------------------------------------------------------------------------------------------------*/
    lwk0.s[0] = AxisRscI->PhaseV.Phase;           /* lwk0.s[0] = Phase        */
    lwk0.s[0] += 32;
    lwk1.s[0] = (SHORT)PI23;
    lwk2.s[0] = lwk1.s[0] + lwk0.s[0];            /* lwk2.s[0] = Phase + 2π /3    */
    lwk3.s[0] = lwk0.s[0] - lwk1.s[0];            /* lwk3.s[0] = Phase - 2π /3    */

    lwk4.s[0] = lwk0.us[0] >> 6;                  /* lwk4.s[0] = lwk0.s[0] >> 6(論  理) */
    IxTblSin16( AxisRscI->SinTbl.Sin1.s[0], lwk4.s[0] );  /* sin1.sr = SinTbl[ lwk4.s[0] ]  */
    lwk0.s[0] += (SHORT)PI2;                     /* lwk0.s[0] = lwk0.s[0] + π /2   */
    lwk4.s[0] = lwk0.us[0] >> 6;                  /* lwk4.s[0] = lwk0.s[0] >> 6(論  理) */
    IxTblSin16( AxisRscI->SinTbl.Cos1.s[0], lwk4.s[0] );  /* cos1.sr = SinTbl[ lwk4.s[0] ]  */

    lwk4.s[0] = lwk3.us[0] >> 6;                  /* lwk4.s[0] = lwk3.s[0] >> 6論  理)  */
    IxTblSin16( AxisRscI->SinTbl.Sin2.s[0], lwk4.s[0] );  /* sin2.sr = SinTbl[ lwk4.s[0] ]  */
    lwk3.s[0] += (SHORT)PI2;                     /* lwk3.s[0] = lwk3.s[0] + π /2 */
    lwk4.s[0] = lwk3.us[0] >> 6;                  /* lwk4.s[0] = lwk3.s[0] >> 6論  理)  */
    IxTblSin16( AxisRscI->SinTbl.Cos2.s[0], lwk4.s[0] );  /* cos2.sr = SinTbl[ lwk4.s[0] ]  */

    lwk4.s[0] = lwk2.us[0] >> 6;                  /* lwk4.s[0] = lwk2.s[0] >> 6論  理)  */
    IxTblSin16( AxisRscI->SinTbl.Sin3.s[0], lwk4.s[0] );  /* sin3.sr = SinTbl[ lwk4.s[0] ]  */
    lwk2.s[0] += (SHORT)PI2;                     /* lwk2.s[0] = lwk2.s[0] + π /2 */
    lwk4.s[0] = lwk2.s[0] >> 6;                  /* lwk4.s[0] = lwk2.s[0] >> 6論  理)  */
    IxTblSin16( AxisRscI->SinTbl.Cos3.s[0], lwk4.s[0] );  /* cos3.sr = SinTbl[ lwk4.s[0] ]  */

    AxisRscI->SinTbl.Sin1.l = AxisRscI->SinTbl.Sin1.s[0]; /* 16bit→ 32bit( 符  号付)       */
    AxisRscI->SinTbl.Sin2.l = AxisRscI->SinTbl.Sin2.s[0]; /* 16bit→ 32bit( 符  号付)       */
    AxisRscI->SinTbl.Sin3.l = AxisRscI->SinTbl.Sin3.s[0]; /* 16bit→ 32bit( 符  号付)       */
    AxisRscI->SinTbl.Cos1.l = AxisRscI->SinTbl.Cos1.s[0]; /* 16bit→ 32bit( 符  号付)       */
    AxisRscI->SinTbl.Cos2.l = AxisRscI->SinTbl.Cos2.s[0]; /* 16bit→ 32bit( 符  号付)       */
    AxisRscI->SinTbl.Cos3.l = AxisRscI->SinTbl.Cos3.s[0]; /* 16bit→ 32bit( 符  号付)       */
```

```
  }

/*-------------------------------------------------------------------------------------------------*/
/*    A/D convert data loading                                     */
/*-------------------------------------------------------------------------------------------------*/
  ADConvDataLoad( &AxisHdl[0] );

#ifdef  MULTI_AXIS                /* 多  軸  処  理有効                    */
  for( ax_noI = 0; (SHORT)ax_noI < AxisNum; ax_noI++ )
#else
  ax_noI = 0;
#endif
   {
    AxisRscI = &AxisHdl[ax_noI];
/*-------------------------------------------------------------------------------------------------*/
/*    dq-trans(UVW to DQ)                                    */
/*-------------------------------------------------------------------------------------------------*/
/*    IdDetect = ((cos - cos3) * IuDetect / 2^14 + (cos2 - cos3) * IvDetect / 2^14) * 2 / 3   */
/*    IqDetect = ((sin3 - sin) * IuDetect / 2^14 + (sin3 - sin2) * IvDetect / 2^14) * 2 / 3   */
/*    IdDetect = (cos - cos3) * IuDetTmp / 2^14 + (cos2 - cos3) * IvDetTmp / 2^14         */
/*    IqDetect = (sin3 - sin) * IuDetTmp / 2^14 + (sin3 - sin2) * IvDetTmp / 2^14         */
/*-------------------------------------------------------------------------------------------------*/
    /* lwk1.l = (cos - cos3) * IuDetTmp / 2^14 (算  術) * /
    lwk1.l = mulshr( (AxisRscI->SinTbl.Cos1.l - AxisRscI->SinTbl.Cos3.l),
             AxisRscI->CurDet.IuDetTmp,
             14 );
    /* lwk2.l = (cos2 - cos3) * IvDetTmp / 2^14 (算  術) */
    lwk2.l = mulshr( (AxisRscI->SinTbl.Cos2.l - AxisRscI->SinTbl.Cos3.l),
             AxisRscI->CurDet.IvDetTmp,
             14 );
    /* IdDetect = (cos - cos3) * IuDetTmp / 2^14 + (cos2 - cos3) * IvDetTmp / 2^14 (算  術) */
    AxisRscI->Curctrl.IdDetect = lwk1.l + lwk2.l;
/*-------------------------------------------------------------------------------------------------*/
    /* lwk1.l = (sin3 - sin) * IuDetTmp / 2^14 (算  術) * /
    lwk1.l = mulshr( (AxisRscI->SinTbl.Sin3.l - AxisRscI->SinTbl.Sin1.l),
             AxisRscI->CurDet.IuDetTmp,
             14 );
```

```
    /* lwk2.l = (sin3 - sin2) * IvDetTmp / 2^14 (算　術) */
    lwk2.l = mulshr( (AxisRscI->SinTbl.Sin3.l - AxisRscI->SinTbl.Sin2.l),
             AxisRscI->CurDet.IvDetTmp,
             14 );
    /* IqDetect = (sin3 - sin) * IuDetTmp / 2^14 + (sin3 - sin2) * IvDetTmp / 2^14 (算　術) */
    AxisRscI->Curctrl.IqDetect = lwk1.l + lwk2.l;
  }


/*****************************************************************************************************/
/*    Current Control Main Procedure                                 */
/*****************************************************************************************************/
#ifdef  MULTI_AXIS                /* 多　軸　処　理有効                    */
  for( ax_noI = 0; (SHORT)ax_noI < AxisNum; ax_noI++ )
#else
  ax_noI = 0;
#endif
  {
    AxisRscI = &AxisHdl[ax_noI];
/*---------------------------------------------------------------------------------------------------*/
/*    Current Observer                                    */
/*---------------------------------------------------------------------------------------------------*/
    /* 電　流　オ　ブ　ザ　ー バ 未　対　応　　⇒　　電　流　位 相補償 オ ブ ザーバに 変更予定

/*---------------------------------------------------------------------------------------------------*/
/*    Base Block Check                                    */
/*---------------------------------------------------------------------------------------------------*/
    if( AxisRscI->StsFlg.ADRst != False )
    {
      AxisRscI->StsFlg.ADRst = 0;
      AxisRscI->PwmV.PwmCntT2 = (USHORT)AxisRscI->IntAdV.CarrierFreq >> 1;
      AxisRscI->PwmV.PwmCntT1 = AxisRscI->PwmV.PwmCntT2;
      AxisRscI->PwmV.PwmCntT0 = AxisRscI->PwmV.PwmCntT2;
/*---------------------------------------------------------------------------------------------------*/
    }
    else if( (AxisRscI->StsFlg.CtrlStsRW & BB) != False )
    {
```

```
      AxisRscI->PwmV.PwmCntT2 = (USHORT)AxisRscI->IntAdV.CarrierFreq >> 1;
      AxisRscI->PwmV.PwmCntT1 = AxisRscI->PwmV.PwmCntT2;
      AxisRscI->PwmV.PwmCntT0 = AxisRscI->PwmV.PwmCntT2;
/*-----------------------------------------------------------------------------------------------*/
    }
    else
    {
      if( AxisRscI->MotInfo.MotTypeSMIM == IM )
      {
/*************************************************************************************************/
/*    イ ン ダ ク シ ョ ン モータ用 処理                                          */
/*************************************************************************************************/
        ; // 未 対 応
      }
      else
      {
/*************************************************************************************************/
/*                                                  */
/*    同 期 機 用 処理                                        */
/*                                                  */
/*************************************************************************************************/
/*-----------------------------------------------------------------------------------------------*/
/*    磁 束 φ 飽 和計算    KtPercent = (φ Mg/ φMg0) * 2^14                    */
/*-----------------------------------------------------------------------------------------------*/
        lwk0.l = limit( AxisRscI->Curctrl.Iqref, (LONG)OneR );    /* lwk0 = sign(Iqref) */
        lwk0.l = lwk0.l * AxisRscI->Curctrl.Iqref;          /* lwk0 = |Iqref|   */

        lwk1.l = lwk0.l - AxisRscI->IntAdV.IKt2;
        if( lwk1.l >= 0 )
        { /* |Iqref| >= IKt2 の 場 合 */
          /* KtPercent = limit( Kt2+(Kt3-Kt2) / (Imax-IKt2) * (Iq-IKt2), 0～ 2^14 )     */
          IxMulgain32( &lwk2.l, lwk1.l, AxisRscI->IntAdV.KtB, Ixlwk );
          lwk3.l = add_limitf( AxisRscI->IntAdV.Kt2, lwk2.l );
          AxisRscI->IntAdV.KtPercent = limitz( lwk3.l, (LONG)No14bit );
        }
        else
        {
```

```
          lwk1.l = lwk0.l - AxisRscI->IntAdV.IKt;
          if( lwk1.l >= 0 )
          { /* |Iqref| >= IKt の  場 合 */
            /* KtPercent = limit( 100%+(Kt2-100%) / (IKt2-IKt) * (Iq-IKt), 0～ 2^14 )   */
            IxMulgain32( &lwk2.l, lwk1.l, AxisRscI->IntAdV.KtA, Ixlwk );
            lwk3.l = add_limitf( (LONG)No14bit, lwk2.l );
            AxisRscI->IntAdV.KtPercent = limitz( lwk3.l, (LONG)No14bit );
          }
          else
          { /* |Iqref| < IKt の  場 合 */
            AxisRscI->IntAdV.KtPercent = (LONG)No14bit;     /* KtPercent = 100%   */
          }
        }
/*-------------------------------------------------------------------------------------------------*/
/*    Lq飽  和  計 算     LqPercent = (Lq/Lq0) * 2^14                          */
/*-------------------------------------------------------------------------------------------------*/
        lwk1.l = lwk0.l - AxisRscI->IntAdV.ILq3;
        if( lwk1.l >= 0 )
        { /* |Iqref| >= ILq3 の  場 合 */
          /* LqPercent = limit( Lq3+(Lq4-Lq3) / (Imax-ILq3) * (Iq-ILq3), 0～ 2^14 )     */
          IxMulgain32( &lwk2.l, lwk1.l, AxisRscI->IntAdV.LqC, Ixlwk );
          lwk3.l = add_limitf( AxisRscI->IntAdV.Lq3, lwk2.l );
          AxisRscI->IntAdV.LqPercent = limitz( lwk3.l, (LONG)No14bit );
        }
        else
        {
          lwk1.l = lwk0.l - AxisRscI->IntAdV.ILq2;
          if( lwk1.l >= 0 )
          { /* |Iqref| >= ILq2 の  場 合 */
            /* LqPercent = limit( Lq2+(Lq3-Lq2) / (ILq3-ILq2) * (Iq-ILq2), 0～ 2^14 )   */
            IxMulgain32( &lwk2.l, lwk1.l, AxisRscI->IntAdV.Lq2, Ixlwk );
            lwk3.l = add_limitf( AxisRscI->IntAdV.Lq3, lwk2.l );
            AxisRscI->IntAdV.LqPercent = limitz( lwk3.l, (LONG)No14bit );
          }
          else
          { /* |Iqref| < ILq2 の  場 合 */
            /* LqPercent = limit( Lq+(Lq2-Lq) / ILq 2* Iq, 0～ 2^14 )         */
```

```
            IxMulgain32( &lwk2.l, lwk0.l, AxisRscI->IntAdV.LqA, Ixlwk );
            lwk3.l = add_limitf( (LONG)No14bit, lwk2.l );
            AxisRscI->IntAdV.LqPercent = limitz( lwk3.l, (LONG)No14bit );
          }
        }
/*-------------------------------------------------------------------------------------------------*/
/*    Ld飽　和　計 算     LdPercent = (Ld/Ld0) * 2^14                                              */
/*-------------------------------------------------------------------------------------------------*/
        lwk0.l = limit( AxisRscI->Curctrl.Idref, (LONG)OneR );    /* lwk0 = sign(Idref) */
        lwk0.l = lwk0.l * AxisRscI->Curctrl.Idref;          /* lwk0 = |Idref|   */

        lwk1.l = lwk0.l - AxisRscI->IntAdV.ILd3;
        if( lwk1.l >= 0 )
        { /* |Idref| >= ILd3 の　場 合 */
          /* LdPercent = limit( Ld3+(Ld4-Ld3) / (Imax-ILd3) * (Id-ILd3), 0～ 2^14 )     */
          IxMulgain32( &lwk2.l, lwk1.l, AxisRscI->IntAdV.LdC, Ixlwk );
          lwk3.l = add_limitf( AxisRscI->IntAdV.Ld3, lwk2.l );
          AxisRscI->IntAdV.LdPercent = limitz( lwk3.l, (LONG)No14bit );
        }
        else
        {
          lwk1.l = lwk0.l - AxisRscI->IntAdV.ILd2;
          if( lwk1.l >= 0 )
          { /* |Idref| >= ILd2 の　場 合 */
            /* LdPercent = limit( Ld2+(Ld3-Ld2) / (ILd3-ILd2) * (Id-ILd2), 0～ 2^14 )   */
            IxMulgain32( &lwk2.l, lwk1.l, AxisRscI->IntAdV.LdB, Ixlwk );
            lwk3.l = add_limitf( AxisRscI->IntAdV.Ld2, lwk2.l );
            AxisRscI->IntAdV.LdPercent = limitz( lwk3.l, (LONG)No14bit );
          }
          else
          { /* |Idref| < ILd2 の　場 合 */
            /* LdPercent = limit( Ld+(Ld2-Ld) / ILd2 * Id, 0～ 2^14 )         */
            IxMulgain32( &lwk2.l, lwk0.l, AxisRscI->IntAdV.LdA, Ixlwk );
            lwk3.l = add_limitf( (LONG)No14bit, lwk2.l );
            AxisRscI->IntAdV.LdPercent = limitz( lwk3.l, (LONG)No14bit );
          }
        }
```

```
/*****************************************************************************************************************/
/*                                                                                                               */
/*    弱  め  界  磁 制御                                                                                        */
/*                                                                                                               */
/*****************************************************************************************************************//* <S18E>:修  正 */
        if( AxisRscI->MicroIf.CtrlSwitch & V_FB )
        { /* 弱  め  界  磁  有 効の場合 */
/*---------------------------------------------------------------------------------------------------------------*/
/*    q軸  電  圧  リ  ミ ッ ト計算                                                                              */
/*---------------------------------------------------------------------------------------------------------------*/
          dlwk0.dl = mul( AxisRscI->WeakFV.WfV1max, AxisRscI->WeakFV.WfV1max );
          lwk2.l = dlwk0.l[0];
          lwk3.l = dlwk0.l[1];                  /* lwk3,2 = WFV1max^2   */
          dlwk0.dl = mul( AxisRscI->Curctrl.VdBeforeLim,    /* dlwk0 = VdBeforeLim^2  */
                   AxisRscI->Curctrl.VdBeforeLim );
/*---------------------------------------------------------------------------------------------------------------*/
          lwk2.l = (ULONG)lwk2.l >> 1;            /* lwk2 = lwk2 / 2(論  理)  */
          lwk4.l = (ULONG)dlwk0.l[0] >> 1;          /* lwk4 = MacL / 2(論  理)  */
          lwk4.l = lwk2.l - lwk4.l;               /* 下  位32bi t 演算      */
          lwk6.l = (ULONG)lwk4.l >> 31;           /* ボ  ロ ー  (論理)        */
          lwk4.l = (ULONG)lwk4.l << 1;            /* lwk4 = lwk4 * 2(論  理)  */
          lwk5.l = lwk3.l - dlwk0.l[1];           /* 上  位32bi t 演算      */
          lwk5.l = lwk5.l - lwk6.l;               /* lwk5 = lwk5 - borrow   */
/*---------------------------------------------------------------------------------------------------------------*/
          if( lwk5.l < (LONG)ZeroR )
          { /* 負  の  場合 */
            lwk0.l = (LONG)ZeroR;               /* lwk0 = 0             */
          }
          else
          {
            if( lwk5.l == (LONG)ZeroR )
            { /* 上  位32bi tが  0 の場合 */
              lwk1.l = lwk4.l;
              lwk0.s[0] = MpSQRT( lwk1.l );       /* swk0 = √ (2^48 - Idref^2)    */
              lwk0.l = (ULONG)lwk0.s[0] & No0000ffff;
            }
```

```
            else
            {
              if( lwk5.l < (LONG)No8bit )
              { /* 上  位32bi tが0x000001 0 0 よ  り  小さい場合  * /
                lwk8.l = (ULONG)lwk4.l >> 8;       /* 論  理  シ フト           */
                lwk8.l = lwk8.l & No00ffffff;
                lwk9.l = (ULONG)lwk5.l << 24;      /* 論  理  シ フト           */
                lwk9.l = lwk9.l & Noff000000;
                lwk1.l = lwk8.l + lwk9.l;        /* lwk1 = (2^48 - Idref^2)/2^8    */
                lwk0.s[0] = MpSQRT( lwk1.l );      /* swk0 = √ (2^48 - Idref^2)/2^4  */
                lwk0.l = ((ULONG)lwk0.s[0] & No0000ffff) << 4;
                                  /* lwk0 = √ (2^48 - Idref^2) ( 論 理) */
              }
              else
              {
                if( lwk5.l < No16bit )
                { /* 上  位32bi tが0x000100 0 0 よ  り  小さい場合  * /
                  lwk8.l = (ULONG)lwk4.l >> 16;    /* 論  理  シ フト           */
                  lwk8.l = lwk8.l & No0000ffff;
                  lwk9.l = (ULONG)lwk5.l << 16;    /* 論  理  シ フト           */
                  lwk9.l = lwk9.l & Nofffff0000;
                  lwk1.l = lwk8.l + lwk9.l;      /* lwk1 = (2^48 - Idref^2)/2^16   */
                  lwk0.s[0] = MpSQRT( lwk1.l );    /* swk0 = √ (2^48 - Idref^2)/2^8  */
                  lwk0.l = ((ULONG)lwk0.s[0] & No0000ffff) << 8;
                                  /* lwk0 = √ (2^48 - Idref^2) ( 論 理) */
                }
                else
                { /* 上  位32bi tが0x000100 0 0 の場合 */
                  lwk0.l = (LONG)No24bit;        /* lwk0 = √ (2^48) = 2^24     */
                }
              }
            }
          }
          AxisRscI->WeakFV.WfVqLim = lwk0.l;
/*-------------------------------------------------------------------------------------------*/
/*    WFVqLim - |VqBeforeLim| の  計 算                                    */
/*-------------------------------------------------------------------------------------------*/
```

```
        lwk8.l = limit( AxisRscI->Curctrl.VqBeforeLim, (LONG)OneR );
        dlwk0.dl = mul ( AxisRscI->Curctrl.VqBeforeLim, lwk8.l ); /* MacL = |VqBeforeLim| */
        lwk0.l = AxisRscI->WeakFV.WfVqLim - dlwk0.l[0];       /* lwk0 = WFVqLim - |VqBeforeLim| */
        lwk0.l = limit( lwk0.l, (LONG)No23bit );          /* 除 算 の ため に± 2 3 b i t でリミットする * /

        lwk2.l = limit( lwk0.l, (LONG)OneR );
        lwk2.l = lwk0.l * lwk2.l;                  /* lwk2 = |lwk0|  */
/*-------------------------------------------------------------------------------------------*/
/*    比 例 項(wkInTtAd0 * (Kpv / ω ) ) の計 算                                    */
/*-------------------------------------------------------------------------------------------*/
        lwk9.l = limit( AxisRscI->Vcmp.MotSpd, (LONG)OneR );
        lwk9.l = AxisRscI->Vcmp.MotSpd * lwk9.l;          /* lwk9 = |MotSpd|  */
                                /* Ldの 変 化 分 を考慮 */
        lwk9.l = mulshr( lwk9.l, AxisRscI->IntAdV.LdPercent, 27 );  /* lwk9 = (|MotSpd|*LdPercent/16384)/2^13 (算 術) */
        lwk1.l = lwk0.l / lwk9.l;

        IxMulgain32( &lwk2.l, lwk1.l, AxisRscI->WeakFV.WfKpv, Ixlwk );
        lwk4.l = limit( lwk2.l, (LONG)No30bit );          /* ± 30bit で リ ミット */
/*-------------------------------------------------------------------------------------------*/
/*    積 分 項 計 算 ( ア ン チ ワ イン ドアッ プ付き )                                  */
/*-------------------------------------------------------------------------------------------*/
        lwk7.l = AxisRscI->WeakFV.Idref0 - AxisRscI->Curctrl.Idref; /* リ ミ ッ ト前  -  リミット後 */
        lwk8.l = lwk4.l - lwk7.l;                  /* 積 分 入 力 から引く  */
        lwk8.l = limit( lwk8.l, (LONG)No30bit );          /* ± 30bit で リ ミット */
        IxIntegral( &lwk5.l,                      /* lwk5 = 積 分 項  */
              lwk8.l,
              AxisRscI->WeakFV.WfKiv,
              AxisRscI->WeakFV.WfInteg.l,
              Ixlwk );

        if( lwk5.l >= (LONG)ZeroR )
        { /* 積 分 項 >= 0  の場合 */
          lwk5.l = (LONG)ZeroR;
          AxisRscI->WeakFV.WfInteg.l[0] = ZeroR;          /* 積 分 ク リア     */
          AxisRscI->WeakFV.WfInteg.l[1] = ZeroR;          /* 積 分 ク リア     */
        }
/*-------------------------------------------------------------------------------------------*/
```

```
  /*    比  例 項  +   積分 項                                          */
  /*-------------------------------------------------------------------------------------------------*/
          lwk6.l = lwk4.l + lwk5.l;
  /*-------------------------------------------------------------------------------------------------*/
  /*    ロ   ー   パ   ス   フィル タ                                    */
  /*-------------------------------------------------------------------------------------------------*/
          IxLpfilter1( &AxisRscI->WeakFV.WfIdref, lwk6.l, AxisRscI->WeakFV.WfLpfGain, Ixlwk );
  /*-------------------------------------------------------------------------------------------------*/
  /*    WFIdref > 0 な  ら  ば、WFIdref =   0,積分  = 0                          */
  /*      WFIdrefが  正  に  な  る  こ  と  は  無  い 。   正  にな った場合は0にす る。                */
  /*-------------------------------------------------------------------------------------------------*/
          if( AxisRscI->WeakFV.WfIdref >= (LONG)ZeroR )
          { /* WFIdref >= 0 の  場 合 */
            AxisRscI->WeakFV.WfIdref = (LONG)ZeroR;
            AxisRscI->WeakFV.WfInteg.l[0] = (LONG)ZeroR;
            AxisRscI->WeakFV.WfInteg.l[1] = (LONG)ZeroR;
          }
        }
        else
        { /* 弱  め  界  磁  制  御 無効の場合 */
          AxisRscI->WeakFV.WfIdref = (LONG)ZeroR;          /* 弱  め  界磁Idref=0    */
        }
  /*-------------------------------------------------------------------------------------------------*/
  /*    Idref = Idref1 + WFIdref (Idref = limit( lwk0, WFIdrefLim～ 0 ))              */
  /*-------------------------------------------------------------------------------------------------*/
        AxisRscI->WeakFV.Idref0 = AxisRscI->Curctrl.IdrefBeforeWF + AxisRscI->WeakFV.WfIdref;
        AxisRscI->Curctrl.Idref = limitz( AxisRscI->WeakFV.Idref0, AxisRscI->WeakFV.WfIdrefLim );
//        IxAndReg16( &FlagIdrefLim, IFLAG, FlagLimit );  /* リ  ミ  ッ  ト  フ  ラ グ  取 り出 し(不使 用)   */
  /*-------------------------------------------------------------------------------------------------*/
  /*    磁  束  φ  の  逆数計 算                                        */
  /*-------------------------------------------------------------------------------------------------*/
        /* lwk0 = (φ Mg /  φmax) * 2^14 */
        lwk0.l = mulshr( AxisRscI->IntAdV.KtPercent, AxisRscI->IntAdV.RatioPhiMg, 14 );
        /* lwk1 = (Ld/Ld0 * Ld0/Lq0) * 2^28 */
        lwk1.l = AxisRscI->IntAdV.LdPercent * AxisRscI->IntAdV.RatioLdLq;
        /* lwk2 = (Lq/Lq0) * 2^28 */
        lwk2.l = (ULONG)AxisRscI->IntAdV.LqPercent << 14;
```

```
/* lwk4 = ((Lq-Ld)/Lq0) * Idref */
lwk3.l = lwk2.l - lwk1.l;
lwk4.l = mulshr( lwk3.l, AxisRscI->Curctrl.Idref, 28 );
/* lwk5 = ((Lq-Ld)/Lq0)*Idref * (Lq0*MaxCur/2^24)/φ max*2^14 */
IxMulgain32( &lwk5.l, lwk4.l, AxisRscI->IntAdV.CnvRatioPhi, Ixlwk );
/* lwk6 = ((φ Mg/ φmax)-(((Lq-Ld)/Lq0)*Idref*(Lq0*MaxCur/2^24) /φmax))*2^1 4  */
lwk6.l = sub_limitf( lwk0.l, lwk5.l );
/* lwk6 = limit( lwk6, 0～ 32767 )  */
lwk6.l = limitz( lwk6.l, (LONG)No32767 );

if( lwk6.s[0] >= (SHORT)1024 )
{ /* lwk6.s[0] >= 1024 の  場 合 */
  lwk7.l = (LONG)No24bit / (LONG)lwk6.s[0];           /* lwk7.sr = 2^24 / lwk6.sr */
  lwk7.l = (ULONG)lwk7.s[0];                    /* 16bit→ 32bit( 符  号 なし) */
  /* PhiForTrq = 2^24/lwk6.sr * CnvToIqref = 2^10/φ  */
  IxMulgain32( &AxisRscI->IntAdV.PhiForTrq, lwk7.l, AxisRscI->IntAdV.CnvToIqref, Ixlwk );
}
else
{
  if( lwk6.s[0] >= (SHORT)32 )
  { /* lwk6.s[0] >= 32 の  場 合 */
    lwk7.l = (LONG)No19bit / (LONG)lwk6.s[0];         /* lwk7.sr = 2^19 / lwk6.sr */
    lwk7.l = (ULONG)lwk7.s[0];                  /* 16bit→ 32bit( 符  号 なし) */
    lwk7.l = (ULONG)lwk7.l << 5;                /* lwk7 = lwk7 << 5     */
    /* PhiForTrq = 2^24/lwk6.sr * CnvToIqref = 2^10/φ  */
    IxMulgain32( &AxisRscI->IntAdV.PhiForTrq, lwk7.l, AxisRscI->IntAdV.CnvToIqref, Ixlwk );
  }
  else
  {
    if( lwk6.s[0] >= (SHORT)OneR )
    { /* lwk6.s[0] >= 1 の  場 合 */
      lwk7.l = (LONG)No14bit / (LONG)lwk6.s[0];       /* lwk7.sr = 2^14 / lwk6.sr */
      lwk7.l = (ULONG)lwk7.s[0];                /* 16bit→ 32bit( 符  号 なし) */
      lwk7.l = (ULONG)lwk7.l << 10;             /* lwk7 = lwk7 << 10    */
      /* PhiForTrq = 2^24/lwk6.sr * CnvToIqref = 2^10/φ   */
      IxMulgain32( &AxisRscI->IntAdV.PhiForTrq, lwk7.l, AxisRscI->IntAdV.CnvToIqref, Ixlwk );
    }
```

```
            else
            { /* lwk6.s[0] = 0 の  場 合 */
              AxisRscI->IntAdV.PhiForTrq = (LONG)No24bit;
            }
          }
        }
/*-------------------------------------------------------------------------------------------------*/
/*    q軸  電  流  指 令計 算                                                    */
/*-------------------------------------------------------------------------------------------------*/
        /* MacH,L = Trqref /  φ   * 2^10 */
        lwk0.l = mulshr_limitf( AxisRscI->Trqctrl.Trqref, AxisRscI->IntAdV.PhiForTrq, 10 );
        AxisRscI->Curctrl.IqrefBeforeLim = lwk0.l;    /* IqrefBeforeLim = Trqref /  φ       */
/*-------------------------------------------------------------------------------------------------*/
/*    電  流  リ  ミ  ット処 理                                                  */
/*-------------------------------------------------------------------------------------------------*/
        lwk9.l = (LONG)No24bit;                  /* lwk9 = 2^24                */
        lwk9.l = limitz( lwk9.l, AxisRscI->MotInfo.MaxCur );  /* lwk9 = limit( lwk9, 0～ MaxCur )    */
        lwk9.l = limitz( lwk9.l, AxisRscI->MotInfo.MaxCurLim ); /* lwk9 = limit( lwk9, 0～ MaxCurLim )  */

/*-------------------------------------------------------------------------------------------------*/
/*    q軸  電  流  指  令  リ  ミット計算                                          */
/*-------------------------------------------------------------------------------------------------*/
        dlwk0.dl = mul( lwk9.l, lwk9.l );             /* MacH,L = lwk9^2       */
        lwk0.l = dlwk0.l[0];
        lwk1.l = dlwk0.l[1];                     /* lwk1,0 = MaxCurLim^2    */
        lwk0.l = (ULONG)lwk0.l >> 1;                 /* lwk0 = lwk0 / 2 (論  理) */
/*-------------------------------------------------------------------------------------------------*/
        dlwk0.dl = mul( AxisRscI->Curctrl.Idref, AxisRscI->Curctrl.Idref );/* MacH,L = Idref^2  */
        lwk2.l = (ULONG)dlwk0.l[0] >> 1;              /* lwk2 = MacL / 2 (論  理) */
        lwk2.l = lwk0.l - lwk2.l;                  /* 下  位32bi t 演算       */
        lwk4.l = (ULONG)lwk2.l >> 31;                /* ボ  ロ ー    (論理)       */
        lwk2.l = (ULONG)lwk2.l << 1;                 /* lwk2 = lwk2 * 2  (論  理)  */
        lwk3.l = lwk1.l - dlwk0.l[1];                /* 上  位32bi t 演算       */
        lwk3.l = lwk3.l - lwk4.l;                  /* lwk3 = lwk3 - borrow    */
        if( lwk3.l < (LONG)ZeroR )
        { /* 負  の  場合 */
          lwk0.l = (LONG)ZeroR;                  /* lwk0 = 0          */
```

```
}
else
{
  if( lwk3.l == (LONG)ZeroR )
  { /* 上  位32bi tが  0 の場合 */
    lwk0.s[0] = MpSQRT( lwk2.l );           /* lwk0 = √ (MaxCurLim^2 - Idref^2) */
    lwk0.l = (ULONG)lwk0.s[0] & No0000ffff;
  }
  else
  {
    if( lwk3.l < (LONG)No8bit )
    { /* 上  位32bi tが0x000001 0 0 よ  り  小さい場合  * /
      lwk8.l = (ULONG)lwk2.l >> 8;          /* 論  理  シ フト           */
      lwk8.l = lwk8.l & No00ffffff;
      lwk9.l = (ULONG)lwk3.l << 24;         /* 論  理  シ フト           */
      lwk9.l = lwk9.l & Noff000000;
      lwk1.l = lwk8.l + lwk9.l;          /* lwk1 = (MaxCurLim^2 - Idref^2)/2^8 */
      lwk0.s[0] = MpSQRT( lwk1.l );       /* lwk0 = √ (MaxCurLim^2 - Idref^2)/2^4 */
      lwk0.l = ((ULONG)lwk0.s[0] & No0000ffff) << 4;
                            /* lwk0 = √ (MaxCurLim^2 - Idref^2)   */
    }
    else
    {
      if( lwk3.l < (LONG)No16bit )
      { /* 上  位32bi tが0x000100 0 0 よ  り  小さい場合  * /
        lwk8.l = (ULONG)lwk2.l >> 16;       /* 論  理  シ フト           */
        lwk8.l = lwk8.l & No0000ffff;
        lwk9.l = (ULONG)lwk3.l << 16;       /* 論  理  シ フト           */
        lwk9.l = lwk9.l & Nofffff0000;
        lwk1.l = lwk8.l + lwk9.l;        /* lwk1 = (MaxCurLim^2 - Idref^2)/2^16  */
        lwk0.s[0] = MpSQRT( lwk1.l );     /* lwk0 = √ (MaxCurLim^2 - Idref^2)/2^8 */
        lwk0.l = ((ULONG)lwk0.s[0] & No0000ffff) << 8;
                            /* lwk0 = √ (MaxCurLim^2 - Idref^2)   */
      }
      else
      { /* 上  位32bi tが0x000100 0 0 の場合 */
        lwk0.l = (LONG)No24bit;             /* lwk0 = √ (2^48) = 2^24     */
```

```
            }
          }
        }
      }
  /*-----------------------------------------------------------------------------------------------*/
  /*    q軸  電  流  指  令 リ ミット                                          */
  /*-----------------------------------------------------------------------------------------------*/
      if( AxisRscI->Curctrl.IqrefBeforeLim >= ZeroR )
      { /* 正  側  リ  ミ  ット処理 */
        AxisRscI->Curctrl.Iqref = limit( AxisRscI->Curctrl.IqrefBeforeLim, lwk0.l );
        /* ト  ル  ク  制  限  中  フ ラグをセット */
        swk10 = AxisRscI->StsFlg.CtrlStsRW | TLIM;
        AxisRscI->StsFlg.CtrlStsRW = cmove( (AxisRscI->Curctrl.Iqref == lwk0.l),
                        swk10,
                        AxisRscI->StsFlg.CtrlStsRW );
      }
      else
      { /* 負  側  リ  ミ  ット処理 */
        AxisRscI->Curctrl.Iqref = limit( AxisRscI->Curctrl.IqrefBeforeLim, lwk0.l );
        /* ト  ル  ク  制  限  中  フ ラグをセット */
        swk10 = AxisRscI->StsFlg.CtrlStsRW | TLIM;
        AxisRscI->StsFlg.CtrlStsRW = cmove( (AxisRscI->Curctrl.Iqref == lwk0.l),
                        swk10,
                        AxisRscI->StsFlg.CtrlStsRW );
      }
    }

/*************************************************************************************************/
/*                                                */
/*    ACRd(d軸  電  流 制 御)                                  */
/*                                                */
/*************************************************************************************************/
/*-----------------------------------------------------------------------------------------------*/
/*    偏  差  計算                                          */
/*-----------------------------------------------------------------------------------------------*/
    lwk0.l = AxisRscI->Curctrl.Idref - AxisRscI->Curctrl.IdDetect;
/*-----------------------------------------------------------------------------------------------*/
```

```
/*    比  例  項 計 算                                                */
/*--------------------------------------------------------------------------------------------------*/
      IxMulgain32( &lwk1.l, lwk0.l, AxisRscI->Curctrl.GainKpd, Ixlwk ); /* lwk1 = GainKpd * lwk0      */
      lwk1.l = mulshr( lwk1.l, AxisRscI->IntAdV.LdPercent, 14 );      /* lwk1 = (GainKpd*Ld/Ld0) * lwk0 */
      lwk1.l = limit( lwk1.l, (LONG)No30bit );              /* ± 30bit で  リ  ミット       */
      lwk7.l = lwk1.l;
/*--------------------------------------------------------------------------------------------------*/
/*    積  分  項 計  算  ( ア  ン  チ  ワ イン ドアッ プ付き )                                    */
/*--------------------------------------------------------------------------------------------------*/
      if( AxisRscI->MicroIf.CtrlSwitch & ANTIWUP )
      { /* ア  ン  チ  ワ  イ  ン  ドアップ処理 */
        lwk2.l = AxisRscI->Curctrl.VdBeforeLim - AxisRscI->Curctrl.Vdref; /* lwk2 = 電  圧  リ  ミ ット分   */
        lwk7.l = lwk7.l - lwk2.l;                       /* lwk7 = 補  正  後  積 分入力   */
        lwk7.l = limit( lwk7.l, (LONG)No30bit );              /* ± 30bit で  リ  ミット     */
      }
      IxIntegral( &lwk3.l, lwk7.l, AxisRscI->Curctrl.GainKid, AxisRscI->Curctrl.IntegD.l, Ixlwk );
/*--------------------------------------------------------------------------------------------------*/
/*    比  例 項  +   積 分  項 計算                                    */
/*--------------------------------------------------------------------------------------------------*/
      lwk4.l = lwk1.l + lwk3.l;
/*--------------------------------------------------------------------------------------------------*/
/*    電  圧  フ  ィ ル  タ  ( ロ  ーパスフ ィルタ )                                  */
/*--------------------------------------------------------------------------------------------------*/
      IxLpfilter1( &AxisRscI->Vfil.VdLpfOut, lwk4.l, AxisRscI->Vfil.VoltFilGain, Ixlwk );

/****************************************************************************************************/
/*                                                 */
/*    ACRq(q軸  電  流 制 御)                                    */
/*                                                 */
/****************************************************************************************************/
/*--------------------------------------------------------------------------------------------------*/
/*    偏  差  計算                                          */
/*--------------------------------------------------------------------------------------------------*/
      lwk0.l = AxisRscI->Curctrl.Iqref - AxisRscI->Curctrl.IqDetect;
/*--------------------------------------------------------------------------------------------------*/
/*    比  例  項 計 算                                          */
/*--------------------------------------------------------------------------------------------------*/
```

```
      IxMulgain32( &lwk1.l, lwk0.l, AxisRscI->Curctrl.GainKpq, Ixlwk ); /* lwk1 = GainKpq * lwk0      */
      lwk1.l = mulshr( lwk1.l, AxisRscI->IntAdV.LqPercent, 14 );      /* lwk1 = (GainKpq*Lq/Lq0) * lwk0 */
      lwk1.l = limit( lwk1.l, No30bit );                  /* ± 30bit で  リ  ミット       */
      lwk7.l = lwk1.l;
/*-----------------------------------------------------------------------------------------------*/
/*    積  分  項 計  算  ( ア  ン  チ  ワ イン ドアッ プ付き )                                    */
/*-----------------------------------------------------------------------------------------------*/
      if( AxisRscI->MicroIf.CtrlSwitch & ANTIWUP )
      {
        lwk2.l = AxisRscI->Curctrl.VqBeforeLim - AxisRscI->Curctrl.Vqref; /* lwk2 = 電  圧  リ  ミ ット分   */
        lwk7.l = lwk7.l - lwk2.l;                     /* lwk7 = 補  正  後  積 分入力   */
        lwk7.l = limit( lwk7.l, (LONG)No30bit );              /* ± 30bit で  リ  ミット     */
      }
      IxIntegral( &lwk3.l, lwk7.l, AxisRscI->Curctrl.GainKiq, AxisRscI->Curctrl.IntegQ.l, Ixlwk );
/*-----------------------------------------------------------------------------------------------*/
/*    比  例 項  +  積 分  項 計算                                    */
/*-----------------------------------------------------------------------------------------------*/
      lwk4.l = lwk1.l + lwk3.l;
/*-----------------------------------------------------------------------------------------------*/
/*    電  圧  フ  ィ ル  タ  ( ロ  ーパスフ ィルタ )                                  */
/*-----------------------------------------------------------------------------------------------*/
      IxLpfilter1( &AxisRscI->Vfil.VqLpfOut, lwk4.l, AxisRscI->Vfil.VoltFilGain, Ixlwk );

/*************************************************************************************************/
/*                                          */
/*    Voltage Compensation(電  圧  補償 )                          */
/*                                          */
/*************************************************************************************************/
      if( AxisRscI->MicroIf.CtrlSwitch & ISEL )
      {                         /* 電  流  指  令 を使用          */
        lwk1.l = AxisRscI->Curctrl.Idref;       /* lwk1 = d軸  電  流 指令        */
        lwk2.l = AxisRscI->Curctrl.Iqref;       /* lwk2 = q軸  電  流 指令        */
      }
      else
      {                         /* 電  流F B を 使用           */
        lwk1.l = AxisRscI->Curctrl.IdDetect;      /* lwk1 = d軸  電 流FB          */
        lwk2.l = AxisRscI->Curctrl.IqDetect;      /* lwk2 = q軸  電 流FB          */
```

```
    }
/*-------------------------------------------------------------------------------------------------*/
/*    RId - ω LqIq  の  計算                                                                       */
/*-------------------------------------------------------------------------------------------------*/
    IxMulgain32( &lwk3.l, lwk1.l, AxisRscI->Vcmp.Resist, Ixlwk ); /* lwk3 = RId       */

    lwk4.l = mulshr( AxisRscI->Vcmp.MotSpd, lwk2.l, 20 );     /* lwk4 = (ω  * Iq) >> 20 */
    IxMulgain32( &lwk4.l, lwk4.l,  AxisRscI->Vcmp.Lq, Ixlwk );    /* lwk4 = ω LqOIq     */
    lwk4.l = mulshr( lwk4.l,  AxisRscI->IntAdV.LqPercent, 14 );   /* lwk4 = ω LqOIq * Lq/Lq0  */

    lwk3.l = lwk3.l - lwk4.l;                   /* lwk3 = RId - ω LqIq    */
/*-------------------------------------------------------------------------------------------------*/
/*    RIq + ω LdId +  ω  φ    の 計算                                                              */
/*-------------------------------------------------------------------------------------------------*/
    IxMulgain32( &lwk4.l, lwk2.l, AxisRscI->Vcmp.Resist, Ixlwk ); /* lwk4 = RIq       */

    lwk5.l = mulshr( AxisRscI->Vcmp.MotSpd, lwk1.l, 20 );     /* lwk5 = (ω  * Id) >> 20 */
    IxMulgain32( &lwk5.l, lwk5.l, AxisRscI->Vcmp.Ld, Ixlwk );   /* lwk5 = ω LdOId     */
    lwk5.l = mulshr( lwk5.l, AxisRscI->IntAdV.LdPercent, 14 );    /* lwk5 = ω LdOId * Ld/Ld0  */

    lwk6.l = mulshr( AxisRscI->Vcmp.OmegaPhi, AxisRscI->IntAdV.KtPercent, 14 );
                                /* lwk6 = ω  φ0 *  φ /φ0     */

    lwk4.l = lwk4.l + lwk5.l;                   /* lwk4 = RIq + ω LdId        */
    lwk4.l = lwk4.l + lwk6.l;                   /* lwk4 = RIq + ω LdId +  ω  φ   */
/*-------------------------------------------------------------------------------------------------*/
/*    Ld*dId/dt, Lq*dIq/dt の  計 算 (Ldi/ d t補償)                                                */
/*-------------------------------------------------------------------------------------------------*/
/* 今  の  と  こ  ろ  必  要  な  い ので、作らない。  */
    AxisRscI->Vcmp.VoltageffD = lwk3.l;             /* VoltageffD = RId - ω LqIq    */
    AxisRscI->Vcmp.VoltageffQ = lwk4.l;             /* VoltageffQ = RIq + ω LdId +  ω  φ */
/*-------------------------------------------------------------------------------------------------*/
/*    電  圧  指  令 に 電  圧  F F補償を 加える                                                   */
/*-------------------------------------------------------------------------------------------------*/
    lwk5.l = AxisRscI->Vfil.VdLpfOut + AxisRscI->Vcmp.VoltageffD; /* lwk5 = VdLpfOut + VoltageffD   */
    lwk6.l = AxisRscI->Vfil.VqLpfOut + AxisRscI->Vcmp.VoltageffQ; /* lwk6 = VqLpfOut + VoltageffQ   */
/*-------------------------------------------------------------------------------------------------*/
```

```
/*    AVRゲ  イ  ン 補償                                          */
/*--------------------------------------------------------------------------------------------------*/
      lwk7.l = (ULONG)AxisRscI->Curctrl.AVRGain;          /* 16bit→ 32bit( 符  号 なし)       */
      lwk8.l = mulshr( lwk5.l, lwk7.l, 13 );            /* MacH,L = d軸  電  圧 指令 * AVRGain    */
                                /* lwk8 = MacH,L >> 13          */
      AxisRscI->Curctrl.Vdref = limit( lwk8.l, (LONG)No30bit ); /* ± 2^30 で  リ  ミット            */

      lwk9.l = mulshr( lwk6.l, lwk7.l, 13 );            /* MacH,L = q軸  電  圧 指令 * AVRGain    */
                                /* lwk9 = MacH,L >> 13          */
      AxisRscI->Curctrl.Vqref = limit( lwk9.l, (LONG)No30bit ); /* ± 2^30 で  リ  ミット            */

/****************************************************************************************************/
/*                                                  */
/*    電  圧  リ  ミ  ット処 理                                  */
/*                                                  */
/****************************************************************************************//* <S18E>:修  正 */
/*--------------------------------------------------------------------------------------------------*/
/*    変  調  率 計 算                                          */
/*--------------------------------------------------------------------------------------------------*/
      lwk0.l = ZeroR;
      IxSquareSum( (DLREG*)&dlwk0,
             AxisRscI->Curctrl.Vdref,
             AxisRscI->Curctrl.Vqref,
             (DWREG*)Ixlwk );

      if( dlwk0.l[1] >= (LONG)No24bit )
      { /* 上  位32bi tが0x010000 0 0 以  上の場合 */
        lwk0.s[0] = MpSQRT( dlwk0.l[1] );                 /* lwk0 = √ (Vdref^2+Vqref^2)/2^16  */
        AxisRscI->Vltctrl.V1 = ((ULONG)lwk0.s[0] & No0000ffff) << 16;   /* V1 = √ (Vdref^2+Vqref^2)     */
        lwk9.s[0] = 16;                             /* シ  フ  ト  数保存          */
      }
      else
      {
        if( dlwk0.l[1] >= (LONG)No16bit )
        { /* 上  位32bi tが0x000100 0 0 以  上の場合 */
          lwk8.l = (ULONG)dlwk0.l[0] >> 24;               /* 論  理  シ フト            */
          lwk8.l = lwk8.l & No000000ff;
```

```
    lwk9.l = (ULONG)dlwk0.l[1] << 8;                /* 論　理　シ フト           */
    lwk9.l = lwk9.l & Noffffff00;
    lwk1.l = lwk8.l + lwk9.l;                     /* lwk1 = (Vdref^2+Vqref^2)/2^24  */
    lwk0.s[0] = MpSQRT( lwk1.l );                   /* lwk0 = √ (Vdref^2+Vqref^2)/2^12  */
    AxisRscI->Vltctrl.V1 = ((ULONG)lwk0.s[0] & No0000ffff) << 12; /* V1 = √ (Vdref^2+Vqref^2)     */
    lwk9.s[0] = 12;                             /* シ　フ　ト　数保存          */
  }
  else
  {
    if( dlwk0.l[1] >= (LONG)No8bit )
    { /* 上　位32bi tが0x000001 0 0 以　上の場合 */
      lwk8.l = (ULONG)dlwk0.l[0] >> 16;             /* 論　理　シ フト           */
      lwk8.l = lwk8.l & No0000ffff;
      lwk9.l = (ULONG)dlwk0.l[1] << 16;             /* 論　理　シ フト           */
      lwk9.l = lwk9.l & Noffff0000;
      lwk1.l = lwk8.l + lwk9.l;                   /* lwk1 = (Vdref^2+Vqref^2)/2^16  */
      lwk0.s[0] = MpSQRT( lwk1.l );                 /* lwk0 = √ (Vdref^2+Vqref^2)/2^8 */
      AxisRscI->Vltctrl.V1 = ((ULONG)lwk0.s[0] & No0000ffff) << 8;/* V1 = √ (Vdref^2+Vqref^2)     */
      lwk9.s[0] = 8;                            /* シ　フ　ト　数保存          */
    }
    else
    {
      if( dlwk0.l[1] != (LONG)ZeroR )
      { /* 上　位32bi tが　0 で　ない場合 */
        lwk8.l = (ULONG)dlwk0.l[0] >> 8;            /* 論　理　シ フト           */
        lwk8.l = lwk8.l & No00ffffff;
        lwk9.l = (ULONG)dlwk0.l[1] << 24;           /* 論　理　シ フト           */
        lwk9.l = lwk9.l & Noff000000;
        lwk1.l = lwk8.l + lwk9.l;                 /* lwk1 = (Vdref^2+Vqref^2)/2^8   */
        lwk0.s[0] = MpSQRT( lwk1.l );               /* lwk0 = √ (Vdref^2+Vqref^2)/2^4 */
        AxisRscI->Vltctrl.V1 = ((ULONG)lwk0.s[0] & No0000ffff) << 4;
                                  /* V1 = √ (Vdref^2+Vqref^2)     */
        lwk9.s[0] = 4;                            /* シ　フ　ト　数保存          */
      }
      else
      { /* 上　位32bi tが　0 の場合 */
        lwk1.l = dlwk0.l[0];                    /* lwk1 = (Vdref^2+Vqref^2)     */
```

```
            lwk0.s[0] = MpSQRT( lwk1.l );              /* lwk0 = √ (Vdref^2+Vqref^2)   */
            AxisRscI->Vltctrl.V1 = ((ULONG)lwk0.s[0] & No0000ffff); /* V1 = √ (Vdref^2+Vqref^2)    */
            lwk9.s[0] = (SHORT)ZeroR;                 /* シ フ ト 数保存             */
          }
        }
      }
    }

    AxisRscI->Curctrl.VdBeforeLim = AxisRscI->Curctrl.Vdref;        /* 保 存                */
    AxisRscI->Curctrl.VqBeforeLim = AxisRscI->Curctrl.Vqref;        /* 保 存                */
    AxisRscI->Vltctrl.V1BeforeLim = AxisRscI->Vltctrl.V1;         /* 保 存                */

/*-------------------------------------------------------------------------------------------*/
/*    電 圧 リ ミ ット処 理                                                                  */
/*-------------------------------------------------------------------------------------------*/
    if( AxisRscI->Vltctrl.Vmax < AxisRscI->Vltctrl.V1 )
    { /* 変 調 率 が リ ミット以下 */
      lwk1.l = AxisRscI->Vltctrl.Vmax;
      if( lwk0.s[0] < (SHORT)ZeroR )
      { /* lwk0.s[0]が 負 の 場合 */
        /* lwk0.srが 32767 を 超 え て 負 の 値 にな っ ている ので、2で 割る */
        lwk1.l = (ULONG)lwk1.l >> 1;                  /* 論 理 シ フト */
        lwk0.l = ((ULONG)lwk0.s[0] & No0000ffff) >> 1;          /* 論 理 シ フト */
      }
      /* lwk2=Vmax/V1*2^(lwk9.sr), lwk0=4096～ 65535, lwk1=4194304 ～5340353 * /
      lwk2.l = lwk1.l / lwk0.l;
      AxisRscI->Vltctrl.V1 = AxisRscI->Vltctrl.Vmax;    /* V1 = Vmax          */
      AxisRscI->BlockTr.TxDataCToAsic3.l = lwk2.l;

      dlwk0.dl = 0;                        /* Buffer Clear                */
      dlwk0.dl = mul( AxisRscI->Curctrl.Vdref, lwk2.l );    /* MacH,L = Vdref * Vmax / V1 * 2^(lwk9.sr) */
      AxisRscI->Curctrl.Vdref = asr( dlwk0.dl, lwk9.s[0] ); /* Vdref = Vdref * Vmax / V1  */

      dlwk0.dl = 0;                        /* Buffer Clear                */
      dlwk0.dl = mul( AxisRscI->Curctrl.Vqref, lwk2.l );    /* MacH,L = Vdref * Vmax / V1 * 2^(lwk9.sr) */
      AxisRscI->Curctrl.Vqref = asr( dlwk0.dl, lwk9.s[0] ); /* Vdref = Vdref * Vmax / V1  */
    }
```

```
/*****************************************************************************************************/
/*                                                   */
/*    UVW transform : dq( 2phase ) to UVW( 3phase ) Transform                    */
/*                                                   */
/*****************************************************************************************************/
/*---------------------------------------------------------------------------------------------------*/
/*    Vuref = Vdref*cos θ   - Vqref*sin  θ                                    */
/*---------------------------------------------------------------------------------------------------*/
      lwk8.l = mulshr( AxisRscI->Curctrl.Vdref, AxisRscI->SinTbl.Cos1.l, 14 );  /* lwk8 = (Vderf * cos θ  ) >> 14     */
      lwk9.l = mulshr( AxisRscI->Curctrl.Vqref, AxisRscI->SinTbl.Sin1.l, 14 );  /* lwk9 = (Vqerf * sin θ  ) >> 14     */
      AxisRscI->Curctrl.Vuref = lwk8.l - lwk9.l;                  /* Vuref = Vdref*cos θ   - Vqref*sin  θ  */
/*---------------------------------------------------------------------------------------------------*/
/*    Vvref = Vdref*cos( θ  -2  π/3) - Vqref*sin ( θ  -2π/3)                    */
/*---------------------------------------------------------------------------------------------------*/
      lwk8.l = mulshr( AxisRscI->Curctrl.Vdref, AxisRscI->SinTbl.Cos2.l, 14 );  /* lwk8 = (Vderf *  θ  -2  π/3) >> 14
      */
      lwk9.l = mulshr( AxisRscI->Curctrl.Vqref, AxisRscI->SinTbl.Sin2.l, 14 );  /* lwk9 = (Vqerf *  θ  -2  π/3) >> 14
      */
      AxisRscI->Curctrl.Vvref = lwk8.l - lwk9.l;                  /* Vuref = Vdref*cos( θ  -2  π/3) - Vqref*sin ( θ  -2π/3 )  */
/*---------------------------------------------------------------------------------------------------*/
/*    Vwref = -Vuref - Vvref                                  */
/*---------------------------------------------------------------------------------------------------*/
      lwk5.l = (LONG)ZeroR - AxisRscI->Curctrl.Vuref;
      AxisRscI->Curctrl.Vwref = lwk5.l - AxisRscI->Curctrl.Vvref;         /* Vwref = -Vuref - Vvref       */

/*****************************************************************************************************/
/*                                                   */
/*    変  調  補  正  &  過変 調補正                                    */
/*                                                   */
/*****************************************************************************************************/
/*---------------------------------------------------------------------------------------------------*/
/*    Over modulation type select                                 */
/*---------------------------------------------------------------------------------------------------*/
      if( (AxisRscI->MicroIf.CtrlSwitch & OVMSEL2) == ZeroR )
      { /* タ  イ プ  2 有  効  で はない場合 */
        if( AxisRscI->Vltctrl.V1 >= (LONG)No22bit )
```

```
        { /* 変　調 率 >= 100 %　の場合　*/
          if( (AxisRscI->MicroIf.CtrlSwitch & OVMSEL1) != ZeroR )
          { /* タ　イ プ　1 有　効の場合　*/
/*-------------------------------------------------------------------------------------------*/
/*    Over modulation1                                        */
/*-------------------------------------------------------------------------------------------*/
            IxSetCtblAdr( pCtbl, &(OVMODTBLG[0][0]) );          /* gain type            */
            lwk9.l = mulshr( AxisRscI->Vltctrl.V1, (LONG)OneR, 9 );
                                    /* 正　規　化　を 変更( ± 2^22→±8192)   */
            AxisRscI->Vltctrl.ModulationComp = MpOVMMODK( AxisRscI->Vltctrl.V1,
                                lwk9.s[0],
                                AxisRscI->MicroIf.CtrlSwitch,
                                pCtbl );
            lwk0.l = (ULONG)AxisRscI->Vltctrl.ModulationComp;     /* 16bit→ 32bit( 符　号 なし)     */
            AxisRscI->Curctrl.Vuref = mulshr( AxisRscI->Curctrl.Vuref, lwk0.l, 13 );
            AxisRscI->Curctrl.Vvref = mulshr( AxisRscI->Curctrl.Vvref, lwk0.l, 13 );
            AxisRscI->Curctrl.Vwref = mulshr( AxisRscI->Curctrl.Vwref, lwk0.l, 13 );
/*-------------------------------------------------------------------------------------------*/
/*    lwk1 = |Vuref|,     lwk2 = |Vvref|,     lwk3 = |Vwref|                    */
/*    lwk4 = sign(Vuref), lwk5 = sign(Vvref), lwk6 = sign(Vwref)                  */
/*-------------------------------------------------------------------------------------------*/
            lwk4.l = limit( AxisRscI->Curctrl.Vuref, (LONG)OneR );
            lwk1.l = lwk4.l * AxisRscI->Curctrl.Vuref;

            lwk5.l = limit( AxisRscI->Curctrl.Vvref, (LONG)OneR );
            lwk2.l = lwk5.l * AxisRscI->Curctrl.Vvref;

            lwk6.l = limit( AxisRscI->Curctrl.Vwref, (LONG)OneR );
            lwk3.l = lwk6.l * AxisRscI->Curctrl.Vwref;

            if( lwk1.l >= lwk2.l )
            { /* |Vuref| >= |Vvref| の　場 合 */
              if( lwk1.l >= lwk3.l )
              { /* U相　が　最　も　大　き い時の処理 */
                lwk1.l = lwk1.l - (LONG)No22bit;      /* lwk1 = |Vuref|-2^22  */
                lwk1.l = limitz( lwk1.l, (LONG)No24bit ); /* zero limit     */
                lwk0.l = lwk4.l * lwk1.l;
```

```
            }
            else
            { /* W相　が　最　も　大　き い時の処理 */
              lwk3.l = lwk3.l - (LONG)No22bit;      /* lwk3 = |Vwref|-2^22  */
              lwk3.l = limitz( lwk3.l, (LONG)No24bit ); /* zero limit     */
              lwk0.l = lwk6.l * lwk3.l;
            }
          }
          else
          {
            if( lwk2.l >= lwk3.l )
            { /* V相　が　最　も　大　き い時の処理 */
              lwk2.l = lwk2.l - (LONG)No22bit;      /* lwk2 = |Vvref|-2^22  */
              lwk2.l = limitz( lwk2.l, (LONG)No24bit ); /* zero limit     */
              lwk0.l = lwk5.l * lwk2.l;
            }
            else
            { /* W相　が　最　も　大　き い時の処理 */
              lwk3.l = lwk3.l - (LONG)No22bit;      /* lwk3 = |Vwref|-2^22  */
              lwk3.l = limitz( lwk3.l, (LONG)No24bit ); /* zero limit   */
              lwk0.l = lwk6.l * lwk3.l;
            }
          }

          AxisRscI->Curctrl.Vuref = AxisRscI->Curctrl.Vuref - lwk0.l;
          AxisRscI->Curctrl.Vvref = AxisRscI->Curctrl.Vvref - lwk0.l;
          AxisRscI->Curctrl.Vwref = AxisRscI->Curctrl.Vwref - lwk0.l;
          AxisRscI->Vltctrl.Vcentral = lwk0.l;
        }
      }
    }
    else
    {
/*-------------------------------------------------------------------------------------------------*/
/*    Over modulation2                                          */
/*-------------------------------------------------------------------------------------------------*/
      IxSetCtblAdr( pCtbl, &(OVMODTBLO[0][0]) );          /* gain type            */
```

```
        lwk9.l = mulshr( AxisRscI->Vltctrl.V1, (LONG)OneR, 9 );
                                    /* 正  規  化  を 変更( ± 2^22→±8192)   */
        AxisRscI->Vltctrl.ModulationComp = MpOVMMODK( AxisRscI->Vltctrl.V1,
                                lwk9.s[0],
                                AxisRscI->MicroIf.CtrlSwitch,
                                pCtbl );
/*-------------------------------------------------------------------------------------------------*/
/*    MAX = lwk1, MIN = lwk2                                       */
/*    OFS = (lwk1+lwk2)/2                                        */
/*-------------------------------------------------------------------------------------------------*/
        if( AxisRscI->Curctrl.Vuref >= AxisRscI->Curctrl.Vvref )
        { /* Vuref >= Vref の  場 合 */
          lwk1.l = AxisRscI->Curctrl.Vuref;
          lwk2.l = AxisRscI->Curctrl.Vvref;
        }
        else
        { /* Vuref < Vref の  場 合 */
          lwk1.l = AxisRscI->Curctrl.Vvref;
          lwk2.l = AxisRscI->Curctrl.Vuref;
        }

        if( lwk1.l < AxisRscI->Curctrl.Vwref )
        { /* Vurefと Vvref の  大  きい方 < V w r ef の場 合  * /
          lwk1.l = AxisRscI->Curctrl.Vwref;
        }
        else
        {
          if( AxisRscI->Curctrl.Vwref < lwk2.l )
          { /* Vurefと Vvref の  小  さい方 < V w r ef の場 合  * /
            lwk2.l = AxisRscI->Curctrl.Vwref;
          }
        }

        lwk0.l = lwk2.l + lwk1.l;
        lwk0.l = mulshr( lwk0.l, (LONG)OneR, 1 );

        AxisRscI->Curctrl.Vuref = AxisRscI->Curctrl.Vuref - lwk0.l;
```

```
        AxisRscI->Curctrl.Vvref = AxisRscI->Curctrl.Vvref - lwk0.l;
        AxisRscI->Curctrl.Vwref = AxisRscI->Curctrl.Vwref - lwk0.l;
        AxisRscI->Vltctrl.Vcentral = lwk0.l;
  /*-------------------------------------------------------------------------------------------------*/
        lwk0.l = (ULONG)AxisRscI->Vltctrl.ModulationComp;     /* 16bit→ 32bit( 符  号 なし)   */
        lwk0.l = (ULONG)lwk0.l << 9;                  /* 論  理  シ フト          */

        lwk1.l = limit( AxisRscI->Curctrl.Vuref, (LONG)OneR );    /* lwk1 = -1 / 0 / 1      */
        lwk1.l = lwk1.l | (LONG)OneR;                 /* lwk1 = -1 / 1(0を 1 に  する)  */
        lwk1.l = lwk1.l * lwk0.l;
        AxisRscI->Curctrl.Vuref = lwk1.l + AxisRscI->Curctrl.Vuref;

        lwk1.l = limit( AxisRscI->Curctrl.Vvref, (LONG)OneR );    /* lwk1 = -1 / 0 / 1      */
        lwk1.l = lwk1.l | (LONG)OneR;                 /* lwk1 = -1 / 1(0を 1 に  する)  */
        lwk1.l = lwk1.l * lwk0.l;
        AxisRscI->Curctrl.Vvref = lwk1.l + AxisRscI->Curctrl.Vvref;

        lwk1.l = limit( AxisRscI->Curctrl.Vwref, (LONG)OneR );    /* lwk1 = -1 / 0 / 1      */
        lwk1.l = lwk1.l | (LONG)OneR;                 /* lwk1 = -1 / 1(0を 1 に  する)  */
        lwk1.l = lwk1.l * lwk0.l;
        AxisRscI->Curctrl.Vwref = lwk1.l + AxisRscI->Curctrl.Vwref;
      }

/*************************************************************************************************/
/*                                                 */
/*    On-Delay                                        */
/*                                                 */
/*************************************************************************************************/
/*-------------------------------------------------------------------------------------------------*/
/*    Iuref, Ivref の  計 算                                    */
/*-------------------------------------------------------------------------------------------------*/
/*-------------------------------------------------------------------------------------------------*/
/*    Iuref = Idref*cos θ  - Iqref*sin θ                                */
/*-------------------------------------------------------------------------------------------------*/
      lwk0.l = mulshr( AxisRscI->Curctrl.Idref, AxisRscI->SinTbl.Cos1.l, 14 );  /* lwk0 = (Idref * cos θ  ) >> 14
      */
      lwk1.l = mulshr( AxisRscI->Curctrl.Iqref, AxisRscI->SinTbl.Sin1.l, 14 );  /* lwk1 = (Iqref * sin θ  ) >> 14
```

```
      */
      AxisRscI->OnDelay.Iuref = lwk0.l - lwk1.l;                  /* Iuref = Idref*cosθ  - Iqref*sin θ          */
/*--------------------------------------------------------------------------------------------------------------*/
/*    Ivref = Idref*cos(θ -2 π/3) - Iqref*sin (θ -2π/3)                                                         */
/*--------------------------------------------------------------------------------------------------------------*/
      lwk0.l = mulshr( AxisRscI->Curctrl.Idref, AxisRscI->SinTbl.Cos2.l, 14 );  /* lwk0 = (Idref * cos(θ -2 π/3)) >> 1 4
      */
      lwk1.l = mulshr( AxisRscI->Curctrl.Iqref, AxisRscI->SinTbl.Sin2.l, 14 );  /* lwk1 = (Iqref * sin(θ -2 π/3)) >> 1 4
      */
      AxisRscI->OnDelay.Ivref = lwk0.l - lwk1.l;                  /* Ivref = Idref*cos(θ -2 π/3) - Iqref*sin (θ -2π/3 )  */
/*--------------------------------------------------------------------------------------------------------------*/
/*    if ( |IuDetect| <= OnDelayLevel ) lwk1 = Iuref                        */
/*    else                lwk1 = IuDetect                  */
/*    if ( |IvDetect| <= OnDelayLevel ) lwk2 = Ivref                        */
/*    else                lwk2 = IvDetect                  */
/*    if ( |IwDetect| <= OnDelayLevel ) lwk3 = Iwref                        */
/*    else                lwk3 = IwDetect                  */
/*--------------------------------------------------------------------------------------------------------------*/
      if( LPX_ABS(AxisRscI->CurDet.IuDetect) > LPX_ABS(AxisRscI->OnDelay.OnDelayLevel) )
      {
        lwk1.l = AxisRscI->CurDet.IuDetect;               /* FB電  流  を 使用      */
      }
      else
      {
        lwk1.l = AxisRscI->OnDelay.Iuref;               /* 電  流  指  令 を使用    */
      }

      if( LPX_ABS(AxisRscI->CurDet.IvDetect) > LPX_ABS(AxisRscI->OnDelay.OnDelayLevel ) )
      {
        lwk2.l = AxisRscI->CurDet.IvDetect;               /* FB電  流  を 使用      */
      }
      else
      {
        lwk2.l = AxisRscI->OnDelay.Ivref;               /* 電  流  指  令 を使用    */
      }

      lwk3.l = (LONG)ZeroR - AxisRscI->CurDet.IuDetect;
```

```
      lwk3.l = lwk3.l - AxisRscI->CurDet.IvDetect;        /* lwk3 = -IuDetect - IvDetect  */

      if( LPX_ABS(lwk3.l) <= LPX_ABS(AxisRscI->OnDelay.OnDelayLevel) )
      {
        lwk3.l = (LONG)ZeroR - AxisRscI->OnDelay.Iuref;
        lwk3.l = lwk3.l - AxisRscI->OnDelay.Ivref;        /* lwk3 = -Iuref - Ivref    */
      }
/*-------------------------------------------------------------------------------------------------*/
/*    if(SlopeOnDelay != 0) trapezoid type else rectangle type                   */
/*-------------------------------------------------------------------------------------------------*/
      if( AxisRscI->OnDelay.OnDelaySlope == (LONG)ZeroR )
      {
/*-------------------------------------------------------------------------------------------------*/
/*    矩 形 波 補 償                                                 */
/*-------------------------------------------------------------------------------------------------*/
        lwk6.l = limit( lwk1.l, (LONG)OneR );         /* lwk6 = -1/0/+1         */
        lwk1.s[0] = AxisRscI->OnDelay.OnDelayComp * lwk6.s[0];  /* lwk1.s = sign(Iu) * OnDelayComp  */

        lwk6.l = limit( lwk2.l, (LONG)OneR );         /* lwk6 = -1/0/+1         */
        lwk2.s[0] = AxisRscI->OnDelay.OnDelayComp * lwk6.s[0];  /* lwk2.s = sign(Iv) * OnDelayComp  */

        lwk6.l = limit( lwk3.l, (LONG)OneR );         /* lwk6 = -1/0/+1         */
        lwk3.s[0] = AxisRscI->OnDelay.OnDelayComp * lwk6.s[0];  /* lwk3.sr = sign(Iw) * OnDelayComp */
      }
      else
      {
/*-------------------------------------------------------------------------------------------------*/
/*    台 形 波 補 償                                                 */
/*-------------------------------------------------------------------------------------------------*/
        lwk0.l = mulshr( AxisRscI->OnDelay.OnDelaySlope, lwk1.l, 24 );    /* lwk0 = (Iu * SlopeOnDelay) >> 24     */
        lwk0.l = limit( lwk0.l, (LONG)No14bit );              /* ± 16384 で リ ミット           */
        lwk1.s[0] = mulshr( AxisRscI->OnDelay.OnDelayComp, lwk0.s[0], 14 ); /* lwk1.s = (lwk0.s * OnDelayComp) >> 14  */

        lwk0.l = mulshr( AxisRscI->OnDelay.OnDelaySlope, lwk2.l, 24 );    /* lwk0 = (Iv * SlopeOnDelay) >> 24     */
        lwk0.l = limit( lwk0.l, (LONG)No14bit );              /* ± 16384 で リ ミット           */
        lwk2.s[0] = mulshr( AxisRscI->OnDelay.OnDelayComp, lwk0.s[0], 14 ); /* lwk2.s = (lwk0.s * OnDelayComp) >> 14  */
```

```
        lwk0.l = mulshr( AxisRscI->OnDelay.OnDelaySlope, lwk3.l, 24 );    /* lwk0 = (Iw * SlopeOnDelay) >> 24     */
        lwk0.l = limit( lwk0.l, (LONG)No14bit );              /* ± 16384 で リミット           */
        lwk3.s[0] = mulshr( AxisRscI->OnDelay.OnDelayComp, lwk0.s[0], 14 ); /* lwk3.s = (lwk0.sr * OnDelayComp) >> 14 */
      }
    }

/*****************************************************************************************************/
/*                                                   */
/*    三 角 波 比 較値計 算                                  */
/*                                                   */
/*****************************************************************************************************//* <S18E>:修 正 */
/*-----------------------------------------------------------------------------------------------------*/
/*    -400000h..400000h →   0h..800000h  → 0h..CRFRQ                  */
/*-----------------------------------------------------------------------------------------------------*/
    AxisRscI->Curctrl.Vuref = limit( AxisRscI->Curctrl.Vuref, (LONG)No22bit );  /* ± 400000h で リ ミット */
    AxisRscI->Curctrl.Vvref = limit( AxisRscI->Curctrl.Vvref, (LONG)No22bit );  /* ± 400000h で リ ミット */
    AxisRscI->Curctrl.Vwref = limit( AxisRscI->Curctrl.Vwref, (LONG)No22bit );  /* ± 400000h で リ ミット */
/*-----------------------------------------------------------------------------------------------------*/
/*    本 当 は、(Vxref-4000 0 0 ) で 計 算するのを(4000 0 0 h - V xre f)としている。          */
/*    理 由 : 電 流 検 出 の向 き が 逆 の た め ( ア ン プ → モ ー タ に 流れたときにマイナスになる)、電流のみ符号を
/*       逆 転 さ せ れ ば よ か っ た が 、 出 力 電 圧 の 符 号 を 逆 転した。そのため、エンコーダの取り付 けも */
/*       180° ず ら し て 取 り付け ている 。                    */
/*-----------------------------------------------------------------------------------------------------*/
    lwk0.l = (ULONG)AxisRscI->IntAdV.CarrierFreq;     /* 16bit→ 32bit( 符 号 なし)     */

    lwk4.l = (LONG)No22bit - AxisRscI->Curctrl.Vuref;
    lwk7.l = mulshr( lwk4.l, lwk0.l, 23 );

    lwk5.l = (LONG)No22bit - AxisRscI->Curctrl.Vvref;
    lwk8.l = mulshr( lwk5.l, lwk0.l, 23 );

    lwk6.l = (LONG)No22bit - AxisRscI->Curctrl.Vwref;
    lwk9.l = mulshr( lwk6.l, lwk0.l, 23 );
/*-----------------------------------------------------------------------------------------------------*/
/*    Deat-time compensation (timer) : if(Vx == 0 || Vx == CarrierFreq) No compensation     */
/*-----------------------------------------------------------------------------------------------------*/
    if( ( lwk7.s[0] != (SHORT)ZeroR ) && (lwk7.s[0] != AxisRscI->IntAdV.CarrierFreq ) )
```

```
    {
      lwk7.s[0] = lwk7.s[0] - lwk1.s[0];                /* Vuref + オ ン デ ィ レ イ補正分 */
      lwk7.s[0] = limit( lwk7.s[0], AxisRscI->IntAdV.CarrierFreq ); /* 0～ CarrierFreq で リ ミット */
    }

    if( ( lwk8.s[0] != (SHORT)ZeroR ) && (lwk8.s[0] != AxisRscI->IntAdV.CarrierFreq ) )
    {
      lwk8.s[0] = lwk8.s[0] - lwk2.s[0];                /* Vvref + オ ン デ ィ レ イ補正分 */
      lwk8.s[0] = limit( lwk8.s[0], AxisRscI->IntAdV.CarrierFreq ); /* 0～ CarrierFreq で リ ミット */
    }

    if( ( lwk9.s[0] != (SHORT)ZeroR ) && (lwk9.s[0] != AxisRscI->IntAdV.CarrierFreq ) )
    {
      lwk9.s[0] = lwk9.s[0] - lwk3.s[0];                /* Vwref + オ ン デ ィ レ イ補正分 */
      lwk9.s[0] = limit( lwk9.s[0], AxisRscI->IntAdV.CarrierFreq ); /* 0～ CarrierFreq で リ ミット */
    }
    AxisRscI->PwmV.PwmCntT2 = lwk9.s[0];
    AxisRscI->PwmV.PwmCntT1 = lwk8.s[0];
    AxisRscI->PwmV.PwmCntT0 = lwk7.s[0];
  }


/*****************************************************************************************************/
/*    Current Control Output Procedure                                                               */
/*****************************************************************************************************/
#ifdef  MULTI_AXIS                /* 多 軸 処 理有効                    */
  for( ax_noI = 0; (SHORT)ax_noI < AxisNum; ax_noI++ )
#else
  ax_noI = 0;
#endif
  {
    AxisRscI = &AxisHdl[ax_noI];

/*---------------------------------------------------------------------------------------------------*/
/*    Output status                                                  */
/*---------------------------------------------------------------------------------------------------*/
    AxisRscI->MicroIf.CtrlStsOut = AxisRscI->StsFlg.CtrlStsRW;
```

```
/*-----------------------------------------------------------------------------------------------*/
/*    System Counter                                          */
/*-----------------------------------------------------------------------------------------------*/
    AxisRscI->Systems.IntHostCtr++;
  }
/*-----------------------------------------------------------------------------------------------*/
/*    PWM data set                                          */
/*-----------------------------------------------------------------------------------------------*/
  SetPWM( &AxisHdl[0] );

/*-----------------------------------------------------------------------------------------------*/
/*    自 己 割 り 込 み禁 止解除                                          */
/*-----------------------------------------------------------------------------------------------*/
  INTLVWR = 0x0003;

  OUTPT = 0x0;
  IniWk.IN_WK1H++;

  return;
}


/*************************************************************************************************/
/*                                                          */
/*    Encoder(SPG0) Interrupt Procedure   ; 通 常 ( 初 期 イ ン ク レ パルス出力 完了時 ):11clk      */
/*                                                          */
/*    [注 意 ] 優 先 順 位 が 最 高 位 の 割 込 処 理 な の で、できるだけ 短い処理にすること。          */
/*************************************************************************************************/
void  MpIntEnc( void )
{
#if 0   /* JL086で 実 行 す る た め コメントアウト    */
/*-----------------------------------------------------------------------------------------------*/
    if( EncIfV.IncPlsReq == 1 )
    {
      PCVS0 = EncIfV.DivPls.s[0];     /* パ ル ス 変 換 位置セット              */
    }
```

```
    else if( EncIfV.PAOSeqCmd != PAOPLSOUT )
    {
      PCVS0 = (SHORT)IHostWk.IncInitPls;  /* パ ル ス 変 換 位置セット                */
    }
/*-------------------------------------------------------------------------------------------------*/
    IEncWk.RxFlg0 = FCCST;    /* SDM status bit8 : IEncWk.RxFlg0(Serial-Enc0 receive flag)  */
/*-------------------------------------------------------------------------------------------------*/
/*    処 理 時 間 短 縮 の た め 、 使 用 し な い データ の読込みはしな い。                          */
/*-------------------------------------------------------------------------------------------------*/
    IEncWk.RxPos.s[0] = SRPG0RD5;     /* 今 回 値 読 込み:Position Low           */
    IEncWk.RxPos.s[1] = SRPG0RD6;     /* 今 回 値 読 込み:Position High          */
/*-------------------------------------------------------------------------------------------------*/
    IEncWk.EncWk0 = INT1SET;        /* INT1 Acknowledge                  */
/*-------------------------------------------------------------------------------------------------*/
#endif  //#if 0   /* JL086で 実 行 す る た め コメント アウト     */
    return;                 /* return                        */
}



#if 0   /* JL086で 実 行 す る た め コメントアウト    */
/*************************************************************************************************/
/*                                                   */
/*    分 周 パ ル ス 更新 処理           ; 最 大:???clk,   通常:???cl k          */
/*                                                   */
/*************************************************************************************************/
void  MpUPDATE_DIVPOS( void )
{
/*-------------------------------------------------------------------------------------------------*/
    IHostWk.Divuswk = INT1SET;        /* INT1 Acknowledge                  */
/*-------------------------------------------------------------------------------------------------*/
    IHostWk.LastRcvPosX = EncIfV.RcvPosX0.l;      /* 前 回 位 置 デ ータ更新          */
/*-------------------------------------------------------------------------------------------------*/
/*    シ リ ア ル エ ン コ ー ダ受信 チェック      ; IEncWk.RxFlg0の 値 は@INT_E N C 割 込 に て更新     */
/*-------------------------------------------------------------------------------------------------*/
//    Divuswk = IEncWk.RxFlg0;        /* SDMSTS bit8 : SPG0 Recieve Completed Check   */
    if( (IEncWk.RxFlg0 & 0x100 ) == 0 )
```

```
    {
      if( EncIfV.SPGFail >= IHostWk.EncMstErrCnt )
      {
        EncIfV.RcvPosX2.l = EncIfV.RcvPosX1.l;            /* 前 々 回 位 置データ                    */
        EncIfV.RcvPosX1.l = EncIfV.RcvPosX0.l;            /* 前 回 位 置 データ                      */
        EncIfV.RcvPosX0.l = EncIfV.RcvPosX0.l + EncIfV.RcvPosX1.l;  /* 補 間 演算                    */
        EncIfV.RcvPosX0.l = EncIfV.RcvPosX0.l - EncIfV.RcvPosX2.l;  /* EncIfV.RcvPosX0 += (EncIfV.RcvPosX1 - EncIfV.RcvPosX2)
        */
        IHostWk.EncMstErrCnt++;                   /* IHostWk.EncMstErrCnt++                  */
      }
    }
/*-------------------------------------------------------------------------------------------------*/
    else
    {
      IHostWk.RxPos0 = IEncWk.RxPos.l;  /* 今 回 値 更新 : IEncWk.R x P osの値は@ I N T _ E NC 割込にて更 新      */
/*-------------------------------------------------------------------------------------------------*/
/*    位 置 演算                                          */
/*    IHostWk.RcvPosX = MencP.MposSign * ((MencV.RxPosL[0].sl>>MencP.MposSftX)<<MencP.MposSftR);  */
/*                                                        */
/*    32bit上 位 詰 め デ ー タ の た め 、 論 理 シ フ ト にて計算(符号 ビットの影響なし)              */
/*-------------------------------------------------------------------------------------------------*/
      IHostWk.RcvPosX = ( IHostWk.RxPos0 >> EncIfV.MotPosSftX ) << EncIfV.MotPosSftR;     /* IHostWk.RcvPosX = (ULONG)DivWk0
      << EncIfV.MotPosSftR  */
/*-------------------------------------------------------------------------------------------------*/
/*    IHostWk.RcvPosX = IHostWk.RcvPosX * EncIfV.MotPosSign                    */
/*-------------------------------------------------------------------------------------------------*/
      if( EncIfV.MotPosSign != 1 )
      {
        IHostWk.RcvPosX = ~IHostWk.RcvPosX;
        IHostWk.RcvPosX = IHostWk.RcvPosX + ONER; /* IHostWk.RcvPosX = -IHostWk.RcvPosX */
      }
/*-------------------------------------------------------------------------------------------------*/
/*    加 速 度 演 算 チェ ック                                    */
/*-------------------------------------------------------------------------------------------------*/
      if( DivPlsV.AccCntClrReq != 0 )
      {
        IHostWk.Divuswk = ~EncIfV.BitData;      /* DivWk0=~EncIfV.BitData              */
```

```
        IHostWk.Divuswk = IHostWk.Divuswk | ACCCHKENA;      /* DivWk0.ACCCHKENA = TRUE        */
        EncIfV.BitData = ~IHostWk.Divuswk;      /* EncIfV.BitData=~DivWk0              */
        IHostWk.AccChkCnt = 0;             /* IHostWk.AccChkCnt = 0              */
        DivPlsV.AccCntClrReq = 0;          /* 加 速 度 チ ェ ッ ク 開 始 カ ウ ントクリア要求 リ セ ット    */
      }
//      Divuswk = EncIfV.BitData;
      if( ( EncIfV.BitData & ACCCHKENA ) == 0 )
      {
        IHostWk.MotAcc = ZEROR;            /* IHostWk.MotAcc = 0                 */
        IHostWk.AccChkCnt++;             /* IHostWk.AccChkCnt++                */
        if( IHostWk.AccChkCnt >= 4 )
        {
          EncIfV.BitData = EncIfV.BitData | ACCCHKENA;    /* EncIfV.BitData.ACCCHKENA = TRUE    */
        }
        EncIfV.RcvPosX0.l = IHostWk.RcvPosX;  /* EncIfV.RcvPosX0 = IHostWk.RcvPosX    */
        EncIfV.RcvPosX1.l = IHostWk.RcvPosX;  /* EncIfV.RcvPosX1 = IHostWk.RcvPosX    */
        EncIfV.RcvPosX2.l = IHostWk.RcvPosX;  /* EncIfV.RcvPosX2 = IHostWk.RcvPosX    */
      }
      else
      {
        IHostWk.DivWk0 = IHostWk.RcvPosX - EncIfV.RcvPosX0.l; /* DivWk0   = IHostWk.RcvPosX  - EncIfV.RcvPosX0  */
        IHostWk.DivWk1 = EncIfV.RcvPosX0.l - EncIfV.RcvPosX1.l; /* DivWk1   = EncIfV.RcvPosX0 - EncIfV.RcvPosX1   */
        IHostWk.MotAcc = IHostWk.DivWk0 - IHostWk.DivWk1;   /* IHostWk.MotAcc   = DivWk0 - DivWk1       */
        if( EncIfV.AccErrLv.l >= IHostWk.MotAcc )
        {
          if( ( EncIfV.AccErrLv.l + IHostWk.MotAcc ) < 0 )
          {
/*-------------------------------------------------------------------------------------------------*/
/*    DivWk0 = (IHostWk.RcvPosX - EncIfV.RcvPosX1) >> 1                    */
/*-------------------------------------------------------------------------------------------------*/
            IHostWk.DivWk0 = IHostWk.RcvPosX - EncIfV.RcvPosX1.l; /* DivWk0 =  IHostWk.RcvPosX  - EncIfV.RcvPosX1   */
            IHostWk.DivWk0 = IHostWk.DivWk0 & 0xfffffffe;     /* 算 術 右 シ フ ト の 四 捨 五入無効化の対策        */
            IHostWk.DivWk0 = IlibASR32(IHostWk.DivWk0 , 1);     /* DivWk0  = (IHostWk.RcvPosX  - EncIfV.RcvPosX1) >> 1  */
            IHostWk.DivWk1 = EncIfV.RcvPosX1.l - EncIfV.RcvPosX2.l; /* DivWk1  =  EncIfV.RcvPosX1 - EncIfV.RcvPosX2   */
            IHostWk.MotAcc = IHostWk.DivWk0 - IHostWk.DivWk1;   /* IHostWk.MotAcc  =  DivWk0 - DivWk1       */
          }
        }
```

```
        else
        {
/*-------------------------------------------------------------------------------------------------*/
/*    DivWk0 = (IHostWk.RcvPosX - EncIfV.RcvPosX1) >> 1                    */
/*-------------------------------------------------------------------------------------------------*/
          IHostWk.DivWk0 = IHostWk.RcvPosX - EncIfV.RcvPosX1.l;   /* DivWk0 =  IHostWk.RcvPosX  - EncIfV.RcvPosX1   */
          IHostWk.DivWk0 = IHostWk.DivWk0 & 0xfffffffe;       /* 算 術 右 シ フ ト の 四 捨 五入無効化の対策        */
          IHostWk.DivWk0 = IlibASR32(IHostWk.DivWk0 , 1);       /* DivWk0  = (IHostWk.RcvPosX  - EncIfV.RcvPosX1) >> 1  */
          IHostWk.DivWk1 = EncIfV.RcvPosX1.l - EncIfV.RcvPosX2.l;   /* DivWk1  =  EncIfV.RcvPosX1 - EncIfV.RcvPosX2   */
          IHostWk.MotAcc = IHostWk.DivWk0 - IHostWk.DivWk1;     /* IHostWk.MotAcc  =  DivWk0 - DivWk1       */
        }
      }
      if( EncIfV.AccErrLv.l >= IHostWk.MotAcc )
      {
/*-------------------------------------------------------------------------------------------------*/
/*    加 速 度 異 常時                                    */
/*-------------------------------------------------------------------------------------------------*/
        if( EncIfV.SPGFail < IHostWk.EncMstErrCnt )
        {
          EncIfV.RcvPosX2.l = EncIfV.RcvPosX1.l;  /* 前 々 回 位 置データ          */
          EncIfV.RcvPosX1.l = EncIfV.RcvPosX0.l;  /* 前 回 位 置 データ          */
          EncIfV.RcvPosX0.l = IHostWk.RcvPosX;  /* 加 速 度 異 常 時 は補間しない       */
          IHostWk.EncMstErrCnt++;         /* IHostWk.EncMstErrCnt++       */
        }
      }
      else if( ( EncIfV.AccErrLv.l + IHostWk.MotAcc ) < 0 )
      {
/*-------------------------------------------------------------------------------------------------*/
/*    加 速 度 正 常時                                    */
/*-------------------------------------------------------------------------------------------------*/
        IHostWk.EncMstErrCnt = 0;         /* IHostWk.EncMstErrCnt=0       */
        EncIfV.RcvPosX2.l = EncIfV.RcvPosX1.l;    /* 前 々 回 位 置データ          */
        EncIfV.RcvPosX1.l = EncIfV.RcvPosX0.l;    /* 前 回 位 置 データ          */
        EncIfV.RcvPosX0.l = IHostWk.RcvPosX;    /* 今 回 位 置 データ          */
      }
/*-------------------------------------------------------------------------------------------------*/
    }
```

```
/*-------------------------------------------------------------------------------------------------*/
/*    dMotPos = RMX_dPosOfXpos( MencV.MotPosX[0], LastMotPosX );                  */
/*-------------------------------------------------------------------------------------------------*/
/*    算 術 右 シ フ ト に て 切 り 捨 て ら れ る 下 位 ビ ットは0のため 、四捨五入の影響なし。          */
/*-------------------------------------------------------------------------------------------------*/
    IHostWk.DMotPos = EncIfV.RcvPosX0.l - IHostWk.LastRcvPosX;    /* IHostWk.DMotPos = EncIfV.RcvPosX0 - IHostWk.LastRcvPosX
    */
    IHostWk.DMotPos = IlibASR32(IHostWk.DMotPos , EncIfV.MotPosSftR);
/*-------------------------------------------------------------------------------------------------*/
    if( EncIfV.IncPlsReq == 1 )
    {
      EncIfV.PlsOSetCmd = DivPlsV.PlsOSetCmdIn;   /* パ ル ス 出 力 回 路 初期化要求更新 from H os tC PU   */
      if( EncIfV.PlsOSetCmd == POSETCMD00 )
      {
        PCVS0 = 0x0000;
        DivPlsV.PlsOSetCmdIn = POSETNOCMD;      /* 初 期 化 要 求クリア          */
      }
      else if( EncIfV.PlsOSetCmd == POSETCMDFF )
      {
        PCVS0 = 0xFFFF;
        DivPlsV.PlsOSetCmdIn = POSETNOCMD;      /* 初 期 化 要 求クリア          */
      }
      else
      {
        IHostWk.IncInitPls = DivPlsV.IncInitPlsIn.l;
        EncIfV.DivPls.l = DivPlsV.IncInitPlsIn.l;
        EncIfV.DivPos.l = DivPlsV.IncInitPlsIn.l;       /* for Linear       */
        EncIfV.DivPlsRem.l = DivPlsV.IncInitRemIn.l;      /* for Linear       */
      }
    }
    else
    {
      if( IHostWk.PoSet1W != DivPlsV.PoSet1In )
      {
        IHostWk.PoSet1W = DivPlsV.PoSet1In;
        IHostWk.PoSet2W = DivPlsV.PoSet2In;
        PCVS1 = IHostWk.PoSet1W;    /* パ ル ス 変 換 原 点 補正1セット   ( H o s tCPUと 同 じ 状態に設 定)     */
```

```
        PCVS2 = IHostWk.PoSet2W;    /* パ ル ス 変 換 原 点 補正2セット                */
      }
    }
    if( IHostWk.DivSetW != DivPlsV.DivSetIn )
    {
      IHostWk.DivSetW = DivPlsV.DivSetIn;
      DivSet = IHostWk.DivSetW;     /* 分 周 機 能 セット (H o s t C P U と 同じ状態に 設 定 )    */
    }
    if( EncIfV.IncPlsReq != 1 )
    {
      if( EncIfV.AmpType != LINEAR )
      {
/*-------------------------------------------------------------------------------------------*/
//    分 周 パ ルス = (MencV.MotPosX[0] >> MencP.EncIfV.DivOutSft);                *
/*-------------------------------------------------------------------------------------------*/
//    算 術 右 シ フ ト に て 切 り 捨 て ら れ る 下 位 ビ ットを0にする (四捨五入無効化 対策)          *
/*-------------------------------------------------------------------------------------------*/
        IHostWk.DivWk1 = NONER << EncIfV.DivOutSft;          /* DivWk1=(FFFFFFFFh<<EncIfV.DivOutSft)                    */
        IHostWk.DivWk0 = EncIfV.RcvPosX0.l & IHostWk.DivWk1;      /* DivWk0=((EncIfV.RcvPosX0&(FFFFFFFFh<<EncIfV.DivOutSft))
        */
        EncIfV.DivPls.l = IlibASR32(IHostWk.DivWk0 , EncIfV.DivOutSft); /*
        EncIfV.DivPls=((EncIfV.RcvPosX0&(FFFFFFFFh<<EncIfV.DivOutSft))>>EncIfV.DivOutSft  */
      }
      else
      {
        DivPlsV.Argu0.l = IHostWk.DMotPos;      /* DivPlsV.Argu0 <-- IHostWk.DMotPos  */
        DivPlsV.Argu1.l = EncIfV.DivOutGain.l;    /* DivPlsV.Argu1 <-- EncIfV.DivOutGain  */
        DivPlsV.Iu0.l = EncIfV.DivPlsRem.l;     /* DivPlsV.Iu0 <-- EncIfV.DivPlsRem   */
        MpMlibPfbkxremNolim( );           /* DivPlsV.Ret0 = MLIBPFBKXREMNOLIM() */
        EncIfV.DivPos.l = EncIfV.DivPos.l + DivPlsV.Ret0.l;   /* EncIfV.DivPos = EncIfV.DivPos + DivPlsV.Ret0   */
        EncIfV.DivPlsRem.l = DivPlsV.Iu0.l;     /* EncIfV.DivPlsRem <-- DivPlsV.Iu0   */
        EncIfV.DivPls.l = EncIfV.DivPos.l;      /* EncIfV.DivPls = EncIfV.DivPos    */
      }
    }
    EncIfV.IncPlsReq = DivPlsV.IncPlsReqIn;       /* 初 期 イ ン ク レ パ ル ス出力要求更新 from  H os tCPU    */
    EncIfV.PAOSeqCmd = DivPlsV.PAOSeqCmdIn;
```

```
    return;                         /* return               */
}
#endif  //#if 0   /* JL086で  実  行  す  る  た  め  コメント アウト     */


/*****************************************************************************************************/
/*                                                   */
/*    DATA clear subroutin                                    */
/*                                                   */
/*****************************************************************************************************/
void  MpDataClear( MICRO_AXIS_HANDLE *AxisRsc )
{
/*---------------------------------------------------------------------------------------------------*/
/*    HOST int clear                                 */
/*---------------------------------------------------------------------------------------------------*/
  AxisRsc->Curctrl.Idref = ZeroR;
  AxisRsc->Curctrl.Iqref = ZeroR;
  AxisRsc->Curctrl.IntegD.l[0] = ZeroR;
  AxisRsc->Curctrl.IntegD.l[1] = ZeroR;
  AxisRsc->Curctrl.VdBeforeLim = ZeroR;
  AxisRsc->Curctrl.Vdref = ZeroR;
  AxisRsc->Curctrl.IntegQ.l[0] = ZeroR;
  AxisRsc->Curctrl.IntegQ.l[1] = ZeroR;
  AxisRsc->Curctrl.VqBeforeLim = ZeroR;
  AxisRsc->Curctrl.Vqref = ZeroR;
  AxisRsc->Curctrl.Vuref = ZeroR;
  AxisRsc->Curctrl.Vvref = ZeroR;
  AxisRsc->Curctrl.Vwref = ZeroR;
/*---------------------------------------------------------------------------------------------------*/
  AxisRsc->Vfil.VdLpfOut = ZeroR;
  AxisRsc->Vfil.VqLpfOut = ZeroR;
/*---------------------------------------------------------------------------------------------------*/
  AxisRsc->OnDelay.Iuref = ZeroR;
  AxisRsc->OnDelay.Ivref = ZeroR;
/*---------------------------------------------------------------------------------------------------*/
  AxisRsc->WeakFV.WfInteg.l[0] = ZeroR;
```

```
  AxisRsc->WeakFV.WfInteg.l[1] = ZeroR;
  AxisRsc->WeakFV.WfIdref = ZeroR;
  AxisRsc->WeakFV.Idref0 = ZeroR;
/*-----------------------------------------------------------------------------------------------------*/
  AxisRsc->Trqctrl.Trqref = ZeroR;
  AxisRsc->Trqctrl.TrqAftFil = ZeroR;
  AxisRsc->Trqctrl.TrqAddDist = ZeroR;
  AxisRsc->Trqctrl.TrqAftLim = ZeroR;
/*-----------------------------------------------------------------------------------------------------*/
  AxisRsc->MicroIf.TrqMon = ZeroR;
  AxisRsc->MicroIf.TrqMonAftFil = ZeroR;
  AxisRsc->MicroIf.IdrefMon = ZeroR;
  AxisRsc->MicroIf.IqrefMon = ZeroR;
  AxisRsc->MicroIf.IdDetectMon = ZeroR;
  AxisRsc->MicroIf.IqDetectMon = ZeroR;
  AxisRsc->MicroIf.VdrefMon = ZeroR;
  AxisRsc->MicroIf.VqrefMon = ZeroR;
/*-----------------------------------------------------------------------------------------------------*/
  AxisRsc->NotchFil.Notch1Value0 = ZeroR;
  AxisRsc->NotchFil.Notch1Value1 = ZeroR;
  AxisRsc->NotchFil.Notch1Value2 = ZeroR;
  AxisRsc->NotchFil.Notch1Value3 = ZeroR;
  AxisRsc->NotchFil.Notch1Out = ZeroR;
  AxisRsc->NotchFil.Notch2Value0 = ZeroR;
  AxisRsc->NotchFil.Notch2Value1 = ZeroR;
  AxisRsc->NotchFil.Notch2Value2 = ZeroR;
  AxisRsc->NotchFil.Notch2Value3 = ZeroR;
  AxisRsc->NotchFil.Notch2Out = ZeroR;
  AxisRsc->NotchFil.Notch3Value0 = ZeroR;
  AxisRsc->NotchFil.Notch3Value1 = ZeroR;
  AxisRsc->NotchFil.Notch3Value2 = ZeroR;
  AxisRsc->NotchFil.Notch3Value3 = ZeroR;
  AxisRsc->NotchFil.Notch3Out = ZeroR;
  AxisRsc->NotchFil.Notch4Value0 = ZeroR;
  AxisRsc->NotchFil.Notch4Value1 = ZeroR;
  AxisRsc->NotchFil.Notch4Value2 = ZeroR;
  AxisRsc->NotchFil.Notch4Value3 = ZeroR;
```

```
  AxisRsc->NotchFil.Notch4Out = ZeroR;
  AxisRsc->NotchFil.Notch5Value0 = ZeroR;
  AxisRsc->NotchFil.Notch5Value1 = ZeroR;
  AxisRsc->NotchFil.Notch5Value2 = ZeroR;
  AxisRsc->NotchFil.Notch5Value3 = ZeroR;
  AxisRsc->NotchFil.Notch5Out = ZeroR;
/*-----------------------------------------------------------------------------------------------*/
  return;
}



/*************************************************************************************************/
/*                                                  */
/*     SQRT(TMP2(32)) Sub-routin                                  */
/*                                                  */
/*************************************************************************************************/
/*    Input   ULONG src    : High(16), Low(16)                        */
/*    Output    uswk0      : SQRT(dat)                                */
/*************************************************************************************************/
inline  USHORT  MpSQRT( ULONG src )
{
  USHORT    uswk0;
  ULONG   ulwk0;
  ULONG   ulwk2;

  uswk0 = sqrt( src );                  /* 結 果 は 小 数 点 以 下は切り捨て      */
  ulwk2 = mul( (SHORT)uswk0, (SHORT)uswk0 );        /* 平 方 根 の 結 果を自乗          */
  ulwk2 = src - ulwk2;                  /* 入 力 と 自 乗 の 差 を 取る(切捨て誤差)    */
  ulwk0 = (ULONG)uswk0;
  if( uswk0 < 0xffff )
  { /* 最 大 値 を 超 えない場合 */
    if( ulwk0 < ulwk2 )
    { /* 切 捨 て 誤 差 が 平 方 根 の 結果より大きい場 合   */
      /* 補 正 処理 */
      uswk0 = uswk0 + 1;
    }
```

```
  }
  else
  {
    /* 最　大　値　を　超　え　る　場　合　は切捨ての補正なし　*/
  }

  return ( uswk0 );
}


/*****************************************************************************************************/
/*                                                   */
/*    Over modulation composation calculation                          */
/*                                                   */
/*****************************************************************************************************/
/*    INPUT:   IntAdP: table address, V1:modulation, NormV1:modulation(=V1/2^9)        */
/*    OUTPUT:  Kmod:   compensation gain/offset                        */
/*    work:    swk0,swk1,swk2,swk3,swk4                                */
/*****************************************************************************************************/
inline  SHORT MpOVMMODK( LONG V1, SHORT NormV1, SHORT CtrlSwitch, CSHORT* pCtbl )
{

  SHORT swk0;                   /* 16bitワ　ー　ク　レ ジスタ0          */
  SHORT swk1;                   /* 16bitワ　ー　ク　レ ジスタ1          */
  SHORT swk2;                   /* 16bitワ　ー　ク　レ ジスタ2          */
  SHORT swk3;                   /* 16bitワ　ー　ク　レ ジスタ3          */
  SHORT swk4;                   /* 16bitワ　ー　ク　レ ジスタ4          */

  if( V1 < (LONG)V115 )
  {
    IxLoadMpmem16( swk4, pCtbl, 0 );          /* IntAdP->Kmod = G[0];         */
  }
  else if( (CtrlSwitch & OVMMOD) == 0 )
  {
    IxLoadMpmem16( swk4, pCtbl, 0 );          /* IntAdP->Kmod = G[0];         */
  }
  else
```

```
{
  if( V1 < (LONG)V127 )
  {
    swk0 = NormV1;
    swk0 = swk0 - 9443;                /* -9439-5(margin)            */
    swk1 = swk0;
    swk0 = (USHORT)swk0 >> 5;            /* high(論  理  シ フト)          */
    swk1 = swk1 & 0x1F;                /* low                        */
    if( swk0 >= 32 )
    { /* 変  調 率1. 2 7 以 上場合 */
      pCtbl = pCtbl + 30;
      IxLoadMpmem16( swk4, pCtbl, 1 );
    }
    else
    { /* 変  調 率1. 2 7 以 下場合 */
      swk2 = swk0;
      if( ( swk2 & 1 ) == 0 )
      {
        /* テ  ー  ブ ル  2 点 取り出し */
        pCtbl = pCtbl + swk0;
        IxLoadMpmem16( swk2, pCtbl, 0 );
        IxLoadMpmem16( swk3, pCtbl, 1 );
      }
      else
      {
        /* テ  ー  ブ ル  2 点 取り出し */
        pCtbl = pCtbl + swk0;
        IxLoadMpmem16( swk2, pCtbl, 1 );
        pCtbl = pCtbl + 2;
        IxLoadMpmem16( swk3, pCtbl, 0 );
      }
      /* 補  間  処理 */
      swk0 = swk3 - swk2;
      swk0 = mulshr(swk0, swk1, 5);
      swk4 = swk0 + swk2;
    }
  }
```

```
    else
    { /* 変　調 率1. 2 7 以 上場合 */
      pCtbl = pCtbl + 30;
      IxLoadMpmem16( swk4, pCtbl, 1 );
    }
  }
  return  swk4;
}


#if 0
/*****************************************************************************************************/
/*                                                   */
/*    制　御　演　算　ラ イブ ラリ                                  */
/*                                                   */
/*****************************************************************************************************/
/*                                                   */
/*    余　り　付　き　位　置ＦＢ計算 : rv = (kx*u+pfbrem)>>sx   ; ??clk                  */
/*                                                   */
/*****************************************************************************************************/
//LONG  MpMlibPfbkxremNolim(
/*    LONG u,                 /* DivPlsV.Argu0    : 入　力              */
/*    LONG k,                 /* DivPlsV.Argu1    : ゲ　イ ン             */
/*    LONG *pfbrem )              /* DivPlsV.Iu0    : 余　り　へ　の　ポ インタ        */
/*-----------------------------------------------------------------------------------------------*/
/*                        /* DivPlsV.Ret0     : 戻　り 値           */
/*-----------------------------------------------------------------------------------------------*/
/*    LONG  kx                /* DivPlsV.Kx     : kx                */
/*    LONG  sx                /* DivPlsV.Sx     : sx                */
/*    LONG  rv                /* lswk10  : 演　算　結果               */
/*    LONG  pfbrem              /* lswk11  : 余　り                */
/*    LONG  wk1               /* lswk1    : 作　業 用               */
/*    LONG  wk2               /* lswk2   : 作　業 用                */
/*                        /* lswk3   : 乗　算　結　果 保　持用(下位32b it)       */
/*                        /* lswk4   : 乗　算　結　果 保　持用(上位32b it)       */
/*-----------------------------------------------------------------------------------------------*/
void  MpMlibPfbkxremNolim( void )
```

```
{
/*-----------------------------------------------------------------------------------------------*/
    DivPlsV.Kx.l = DivPlsV.Argu1.l << 8;    /* DivPlsV.Kx = k<<8              */
    DivPlsV.Sx.l = DivPlsV.Argu1.l >> 24;   /* DivPlsV.Sx = k>>24             */
/*-----------------------------------------------------------------------------------------------*/
    IPfbwk.lswk1 = 24;              /* lswk1 = 24                 */
    if( IPfbwk.lswk1 >= DivPlsV.Sx.l )
    {
/*-----------------------------------------------------------------------------------------------*/
      IPfbwk.dlwk.l[0] = DivPlsV.Argu0.l * DivPlsV.Kx.l;      /* provision        */
      IPfbwk.lswk1 = IPfbwk.lswk1 - DivPlsV.Sx.l;         /* lswk1 = 24 - sx      */
/*-----------------------------------------------------------------------------------------------*/
      IPfbwk.lswk2 = IPfbwk.dlwk.l[0] >> DivPlsV.Sx.s[0];     /* lswk2 = (xl>>sx)             */
      IPfbwk.lswk2 = IPfbwk.lswk2 >> 8;             /* lswk2 =((xl>>sx)>>8)           */
      IPfbwk.lswk10 = IPfbwk.dlwk.l[1] << IPfbwk.lswk1;     /* lswk10 = (xh<<(24-sx))         */
      IPfbwk.lswk10 = IPfbwk.lswk10 + IPfbwk.lswk2;       /* lswk10 =((xh<<(24-sx)) + ((xl>>sx)>>8))  */
/*-----------------------------------------------------------------------------------------------*/
      IPfbwk.lswk11 = IPfbwk.dlwk.l[0] << IPfbwk.lswk1;     /* lswk11 = (xl<<(24-sx))         */
      IPfbwk.lswk11 = IPfbwk.lswk11 >> 8;             /* lswk11 =((xl<<(24-sx))>>8)       */
      IPfbwk.lswk11 = IPfbwk.lswk11 + DivPlsV.Iu0.l;
    }
    else
    {
      IPfbwk.dlwk.l[0] = DivPlsV.Argu0.l * DivPlsV.Kx.l;      /* provision        */
      IPfbwk.lswk3 = IPfbwk.dlwk.l[0];              /* lswk3 = xl       */
      IPfbwk.lswk4 = IPfbwk.dlwk.l[1];              /* lswk4 = xh       */
      IPfbwk.lswk1 = DivPlsV.Sx.l - IPfbwk.lswk1;         /* lswk1 = sx - 24      */
/*-----------------------------------------------------------------------------------------------*/
/*      算 術 右 シ フ ト に て 切 り 捨 て ら れ る 下 位 ビ ットを0にする (四捨 五入無効化対策)          */
/*-----------------------------------------------------------------------------------------------*/
      IPfbwk.lswk2 = NONER << IPfbwk.lswk1;     /* lswk2 =(FFFFFFFFh<<(sx-24))      */
      IPfbwk.lswk2 = IPfbwk.lswk4 & IPfbwk.lswk2;   /* lswk2 =(xh & (FFFFFFFFh<<(sx-24))) */
/*-----------------------------------------------------------------------------------------------*/
      IPfbwk.lswk11 = IPfbwk.lswk3 >> IPfbwk.lswk1; /* lswk11 =   (xl>>(sx-24))           */
      IPfbwk.lswk11 = IPfbwk.lswk11 >> 7;       /* lswk11 =  ((xl>>(sx-24))>>7)         */
      IPfbwk.lswk11 = IPfbwk.lswk11 + ONER;     /* lswk11 = (((xl>>(sx-24))>>7)+1)        */
      IPfbwk.lswk11 = IPfbwk.lswk11 >> 1;       /* lswk11 =((((xl>>(sx-24))>>7)+1)>>1)      */
```

```
      IPfbwk.lswk11 = IPfbwk.lswk11 + DivPlsV.Iu0.l;  /* lswk11 = pfbrem + ((((x1>>(sx-24))>>7)+1)>>1)  */
/*--------------------------------------------------------------------------------------------------*/
      IPfbwk.lswk1 = 56;                /* lswk1 = 56                    */
      IPfbwk.lswk1 = IPfbwk.lswk1 - DivPlsV.Sx.l;   /* lswk1 = 56 - sx                  */
      IPfbwk.lswk2 = IPfbwk.lswk4 << IPfbwk.lswk1;  /* lswk2 = (xh<<(56-sx))              */
      IPfbwk.lswk2 = IPfbwk.lswk2 >> 8;       /* lswk2 =((xh<<(56-sx))>>8)            */
      IPfbwk.lswk11 = IPfbwk.lswk11 + IPfbwk.lswk2; /* lswk11= lswk11 + ((xh<<(56-sx))>>8)       */
    }
    IPfbwk.lswk2 = 0x00800000;              /* lswk2 = 0x00800000               */
#if 0
    if( IPfbwk.lswk11 >= IPfbwk.lswk2 )
    {
      IPfbwk.lswk11 = IPfbwk.lswk11 - ( IPfbwk.lswk2 << 1 );    /* lswk11 = pfbrem - 0x00800000 * 2   */
      IPfbwk.lswk10 = IPfbwk.lswk10 + ONER;           /* lswk10 = lswk10 + 1          */
    }
#endif
    DivPlsV.Iu0.l = IPfbwk.lswk11;            /* lswk11 --> pfbrem          */
    DivPlsV.Ret0.l = IPfbwk.lswk10;           /* lswk10 --> DivPlsV.Ret0        */
/*--------------------------------------------------------------------------------------------------*/
    return;
}
#endif


/****************************************************************************************************/
/*                                                  */
/*    IxMulgain32                                       */
/*                                                  */
/****************************************************************************************************/
inline  void  IxMulgain32( LONG *x, LONG u, LONG k, DWREG *wk )
{
    DLREG dlwk0;
/*--------------------------------------------------------------------------------------------------*/
/*    k --> kx, sx                                      */
/*--------------------------------------------------------------------------------------------------*/
    wk[0].l = (ULONG)k << 8;              /* kx = (k <<  8)           */
    wk[1].l = (ULONG)k >> 24;             /* sx = (k >> 24)           */
```

```
    wk[1].s[0] = wk[1].s[0] + 8;            /* sx = (k >> 24) + 8         */
/*-----------------------------------------------------------------------------------------------*/
/*    x = (((u * kx) >> (sx-1)) + 1) >> 1                                    */
/*    Limit by 0x7FFFFFFF                                  */
/*-----------------------------------------------------------------------------------------------*/
//    *x = mulshr_limitf( u, wk[0].l, wk[1].s[0] );   /* Mac = (u * kx)           */
    dlwk0.dl = mul( u, wk[0].l );
    *x = asr_limitf( dlwk0.l[1], dlwk0.l[0], wk[1].s[0] );
                            /* wk[0] = (((u*kx)>>(sx-1)) + 1) >> 1  */
                            /* x = Limit( wk[0], 0x7FFFFFFF )   */
/*-----------------------------------------------------------------------------------------------*/
    return;
}


/*************************************************************************************************/
/*                                                  */
/*    IxMulgainNolim                                        */
/*                                                  */
/*************************************************************************************************/
inline  void  IxMulgainNolim( LONG *x, LONG u, LONG k, DWREG *wk )
{
    DLREG dlwk0;
/*-----------------------------------------------------------------------------------------------*/
/*    k --> kx, sx                                        */
/*-----------------------------------------------------------------------------------------------*/
    wk[0].l = (ULONG)k << 8;              /* kx = (k <<  8)           */
    wk[1].l = (ULONG)k >> 24;             /* sx = (k >> 24)           */
    wk[1].s[0] = wk[1].s[0] + 8;            /* sx = (k >> 24) + 8         */
/*-----------------------------------------------------------------------------------------------*/
/*    x = (((u * kx) >> (sx-1)) + 1) >> 1                                    */
/*-----------------------------------------------------------------------------------------------*/
//    *x = mulshr( u, wk[0].l, wk[1].s[0] );        /* Mac = (u * kx)           */
//    *x = (u * wk[0].l) >> wk[1].s[0];         /* Mac = (u * kx)           */
    dlwk0.dl = mul( u, wk[0].l );
    *x = asr( dlwk0.dl, wk[1].s[0] );         /* x = (((u*kx)>>(sx-1)) + 1) >> 1    */
/*-----------------------------------------------------------------------------------------------*/
```

```
    return;
}


/*****************************************************************************************************/
/*                                                   */
/*    IxLpfilter1                                          */
/*                                                   */
/*****************************************************************************************************/
inline  void  IxLpfilter1( LONG *x, LONG u, LONG k, DWREG *wk )
{
/*---------------------------------------------------------------------------------------------------*/
/*    Check k                                          */
/*---------------------------------------------------------------------------------------------------*/
    if( k == (LONG)ZeroR )
    {                          /* if( k == 0 )                */
      *x = u;                      /* x = u                   */
    }
    else
    {
/*---------------------------------------------------------------------------------------------------*/
/*    wk[1].dr = ((((u - x) * k) >> 23) + 1) >> 1                          */
/*---------------------------------------------------------------------------------------------------*/
      wk[0].l = u - *x;              /* wk[0] = (u - x)              */
      wk[1].l = mulshr( wk[0].l, k, 24 );     /* wk[1] = ((((u-x)*k)>>23) + 1) >> 1   */
/*---------------------------------------------------------------------------------------------------*/
/*    Check Zero and Set +1/-1                                  */
/*---------------------------------------------------------------------------------------------------*/
      if( wk[1].l != (LONG)ZeroR )         /* if( wk[1] != 0 )             */
      {
        *x = *x + wk[1].l;             /* x = (((((u-x)*k)>>23) + 1) >> 1)     */
      }
/*---------------------------------------------------------------------------------------------------*/
      else if( wk[0].l != (LONG)ZeroR )
      {                          /* if( (u-x) != 0 )             */
        if( wk[0].l > (LONG)ZeroR )
        {                        /* if( (u-x) > 0 )              */
```

```
          wk[1].l = (LONG)OneR;        /* wk[1] = 1                */
          *x = *x + wk[1].l;           /* x = (((((u-x)*k)>>23) + 1) >> 1)     */
        }
        else
        {
          wk[1].l = (LONG)NoneR;         /* wk[1] = -1               */
          *x = *x + wk[1].l;           /* x = (((((u-x)*k)>>23) + 1) >> 1)     */
        }
      }
      else
      {                     /* if( (u-x) == 0 )             */
        ;                   /* 処  理  なし                 */
      }
    }
/*-----------------------------------------------------------------------------------------------*/
    return;
}


/*************************************************************************************************/
/*                                                  */
/*    IxNxfilter2 : 2次  ノ  ッ  チ  フィル タ                          */
/*                                                  */
/*************************************************************************************************/
inline  void  IxNxfilter2( LONG *x, LONG u, LONG k[5], LONG z[4], DWREG wk[4] )
{
#define NXF_PRM_BITS  24                    /* NxFilter2 Prameter Bits Number */
/*-----------------------------------------------------------------------------------------------*/
/*    wk[0].dr = k[4] * u                                   */
/*-----------------------------------------------------------------------------------------------*/
    wk[0].l = mulshr( k[4], u, (SHORT)NXF_PRM_BITS );   /* Mac    = k[4] * u         */
                                /* wk[0] = ((Mac>>23) + 1) >> 1   */
/*-----------------------------------------------------------------------------------------------*/
/*    wk[0].dr = wk[0].dr + (k[0] * z[0]) + (k[1] * z[1])                   */
/*-----------------------------------------------------------------------------------------------*/
    wk[1].l = mulshr( k[0], z[0], (SHORT)NXF_PRM_BITS );  /* Mac    = k[0] * z[0]        */
                                /* wk[1] = ((Mac>>23) + 1) >> 1   */
```

```
    wk[0].l = wk[0].l + wk[1].l;              /* wk[0] = wk[0] + wk[1]      */
/*-----------------------------------------------------------------------------------------------*/
    wk[1].l = mulshr( k[1], z[1], (SHORT)NXF_PRM_BITS );  /* Mac   = k[1] * z[1]        */
                                /* wk[1] = ((Mac>>23) + 1) >> 1   */
    wk[0].l = wk[0].l + wk[1].l;              /* wk[0] = wk[0] + wk[1]      */
/*-----------------------------------------------------------------------------------------------*/
/*    wk[0].dr = wk[0].dr - (k[2] * z[2]) - (k[3] * z[3])                    */
/*-----------------------------------------------------------------------------------------------*/
    wk[1].l = mulshr( k[2], z[2], (SHORT)NXF_PRM_BITS );  /* Mac   = k[2] * z[2]        */
                                /* wk[1] = ((Mac>>23) + 1) >> 1   */
    wk[0].l = wk[0].l - wk[1].l;              /* wk[0] = wk[0] - wk[1]      */
/*-----------------------------------------------------------------------------------------------*/
    wk[1].l = mulshr( k[3], z[3], (SHORT)NXF_PRM_BITS );  /* Mac   = k[3] * z[3]        */
                                /* wk[1] = ((Mac>>23) + 1) >> 1   */
    *x = wk[0].l - wk[1].l;                  /* x     = wk[0] - wk[1]      */
/*-----------------------------------------------------------------------------------------------*/
/*    Update z[i]                                          */
/*-----------------------------------------------------------------------------------------------*/
    z[1] = z[0];                        /* z[1] = z[0]            */
    z[0] = u;                          /* z[0] = u             */
    z[3] = z[2];                        /* z[3] = z[2]            */
    z[2] = *x;                          /* z[2] = x             */
/*-----------------------------------------------------------------------------------------------*/
    return;
}


/*************************************************************************************************/
/*                                                  */
/*    IxIntegral                                        */
/*                                                  */
/*************************************************************************************************/
inline  void  IxIntegral( LONG *x, LONG u, LONG k, LONG iu[2], DWREG *wk )
{
SHORT swk0;
LONG  carry;
DLREG dlwk0;
```

```
/*-------------------------------------------------------------------------------------------------*/
/*    k --> kx, sx                                                 */
/*-------------------------------------------------------------------------------------------------*/
    wk[0].l = (ULONG)k << 8;              /* kx = (k <<  8)           */
    wk[1].l = (ULONG)k >> 24;             /* sx = (k >> 24)           */

/*-------------------------------------------------------------------------------------------------*/
/*    入  力  演算                                                 */
/*-------------------------------------------------------------------------------------------------*/
    dlwk0.dl = mul( u, wk[0].l );           /* dlwk0.dl = u * wk[0]         */
    wk[2].l = asr( dlwk0.l[1], wk[1].s[0] );      /* wk[2] = dlwk0.l[1] >> wk[1] (算  術) */

/*-------------------------------------------------------------------------------------------------*/
/*    積  分  演算                                                 */
/*-------------------------------------------------------------------------------------------------*/
    if( wk[2].l > (LONG)ZeroR )
    {
      iu[1] = iu[1] + (LONG)No25bit;
    }
    else if( wk[2].l < (LONG)NoneR )
    {
      iu[1] = iu[1] - (LONG)No25bit;
    }
    else
    {
/*-------------------------------------------------------------------------------------------------*/
/*      carry = ( (ULONG)(iu[0]+(xx[0]<<(25-sx))) < (ULONG)iu[0] );              */
/*      iu[0] = iu[0] + (xx[0]<<(25-sx));                                */
/*      iu[1] = iu[1] + (xx[1]<<(25-sx)) + (((ULONG)xx[0]>>sx)>>7) + carry;          */
/*-------------------------------------------------------------------------------------------------*/
      swk0 = (SHORT)25 - wk[1].s[0];
      wk[3].l = dlwk0.l[0] << swk0;
      wk[3].l += iu[0];
      if( (ULONG)wk[3].l < (ULONG)iu[0] )
      {
        carry = 1;
```

```
      }
      else
      {
        carry = 0;
      }
/*-----------------------------------------------------------------------------------------------------*/
      iu[0] = wk[3].l;
      wk[2].l = dlwk0.l[1] << swk0;
      wk[3].l = (ULONG)dlwk0.l[0] >> wk[1].s[0];
      wk[3].l = asr( wk[3].l, 7 );
      iu[1] = wk[2].l + wk[3].l;
      iu[1] += carry;
    }
/*-----------------------------------------------------------------------------------------------------*/
/*    積　分　値　リ ミット                                                                            */
/*-----------------------------------------------------------------------------------------------------*/
    if( iu[1] >= (LONG)No25bit )
    {
      iu[0] = 0;
      iu[1] = No25bit;
    }
    else if( iu[1] < (LONG)No25bitM )
    {
      iu[0] = 0;
      iu[1] = No25bitM;
    }
/*-----------------------------------------------------------------------------------------------------*/
/*    戻　り　値 計　算　( 四 捨五入処理 )                                                            */
/*-----------------------------------------------------------------------------------------------------*/
    *x = asr( iu[1], (SHORT)OneR );           /* x = (iu[1] + 1) >> 1          */
}


/*******************************************************************************************************/
/*                                                    */
/*    IxSquareSum                                          */
/*                                                    */
```

```
/*****************************************************************************************************/
inline  void  IxSquareSum( DLREG *x, LONG a, LONG b, DWREG *wk )
{
  volatile DLREG  dlwk0, dlwk1;

  dlwk0.dl = mul( a, a );
  wk[0].l = dlwk0.l[0];
  wk[1].l = dlwk0.l[1];

  dlwk1.dl = mul( b, b );
  wk[2].l = dlwk1.l[0];
  wk[3].l = dlwk1.l[1];

  x->l[0] = wk[0].l + wk[2].l;
  x->l[1] = wk[1].l + wk[3].l;

  wk[3].l = (wk[0].l >> 31) & (wk[2].l >> 31);        /* キ  ャ  リー             */
  x->l[1] = x->l[1] + (wk[3].l & (LONG)OneR);
}


/*****************************************************************************************************/
/*                                                   */
/*    Initialize Soft BB & INT1L Enable                          */
/*                                                   */
/*****************************************************************************************************/
/*                                                   */
/*    備  考 :                                          */
/*    JL-086A不  具  合  回  避 の  た  め  、  5 回連 続書き 込み実施                    */
/*                                                   */
/*****************************************************************************************************/
inline  void  InitSbb( SHORT BbSet )            /* soft BB & INT1L設  定  初  期 化処理   */
{
  BBSET = BbSet;
  BBSET = BbSet;
  BBSET = BbSet;
  BBSET = BbSet;
```

```
  BBSET = BbSet;
#ifdef  MULTI_AXIS              /* 多　軸　処　理有効                      */
  BBSET_2 = BbSet;
  BBSET_2 = BbSet;
  BBSET_2 = BbSet;
  BBSET_2 = BbSet;
  BBSET_2 = BbSet;
#endif /* MULTI_AXIS */           /* 多　軸　処　理有効                      */
}

/*****************************************************************************************************/
/*                                                   */
/*    PWM初　期　化                                        */
/*                                                   */
/*****************************************************************************************************/
inline void InitPWM( MICRO_AXIS_HANDLE *AxisRsc )
{
  USHORT  uswk;

  /* PWM出　力　選　択設定 */
  PWMOS = 0x0A0;                /* PWM出　力　選択　　：キ　ャ　リ　ア　カ　ウ　ンタ比較 出力　　*/
                      /* 相　比　較 選択　　：比　較　値 から PU1 、　N U1信 号作成　　*/
                      /* 即　ロ　ー　ド モード　：マ　イ　ク　ロ　書　き　込　み時に 即ロード　　*/
                      /* 周　波　数　ロ　ード選択：キ　ャ　リ　ア　谷　で　周波数ロー ド　　*/
                      /* ノ　コ　ギ　リ 波選択　：三　角 波                */

  /* キ　ャ　リ　ア　周 波数　取得 & 設定 */
  AxisRsc->IntAdV.CarrierFreq = AxisRsc->MicroIf.CarrierFreqIn;
  CRFRQ = AxisRsc->IntAdV.CarrierFreq;        /* キ　ャ　リ　ア　周　波　数現在地設定          */

  /* キ　ャ　リ　ア　カ　ウ　ン タ比較値設定 */
  uswk = ((USHORT)AxisRsc->IntAdV.CarrierFreq >> 1);  /* IntAdV.CrFreqW /2(50p duty)        */
  AxisRsc->PwmV.PwmCntT2 = uswk;
  AxisRsc->PwmV.PwmCntT1 = uswk;
  AxisRsc->PwmV.PwmCntT0 = uswk;
  PwmT2 = AxisRsc->PwmV.PwmCntT2;
  PwmT1 = AxisRsc->PwmV.PwmCntT1;
```

```
  PwmT0 = AxisRsc->PwmV.PwmCntT0;

#ifdef  MULTI_AXIS              /* 多　軸　処　理有効                      */
  AxisRsc++;
  PWMOS_2 = 0x0A0;              /* PWM出　力　選択　　：キ　ャ　リ　ア　カ　ウ　ンタ比較 出力     */
                    /* 相　比　較 選択　　：比　較　値 から PU1 、　N U1信 号作成　 */
                    /* 即　ロ　ー　ド モード　：マ　イ　ク　ロ　書　き　込　み時に 即ロード     */
                    /* 周　波　数　ロ　ード選択：キ　ャ　リ　ア　谷　で　周波数ロー ド     */
                    /* ノ　コ　ギ　リ 波選択　：三　角 波              */

  /* キ　ャ　リ　ア　周 波数　取得 & 設定 */
  AxisRsc->IntAdV.CarrierFreq = AxisRsc->MicroIf.CarrierFreqIn;
  CRFRQ_2 = AxisRsc->IntAdV.CarrierFreq;          /* キ　ャ　リ　ア　周　波　数現在地設定         */

  /* キ　ャ　リ　ア　カ　ウ　ン タ比較値設定 */
  uswk = ((USHORT)AxisRsc->IntAdV.CarrierFreq >> 1);  /* IntAdV.CrFreqW /2(50p duty)        */
  AxisRsc->PwmV.PwmCntT2 = uswk;
  AxisRsc->PwmV.PwmCntT1 = uswk;
  AxisRsc->PwmV.PwmCntT0 = uswk;
  PwmT2_2 = AxisRsc->PwmV.PwmCntT2;
  PwmT1_2 = AxisRsc->PwmV.PwmCntT1;
  PwmT0_2 = AxisRsc->PwmV.PwmCntT0;
#endif /* MULTI_AXIS */          /* 多　軸　処　理有効                      */
}


/*****************************************************************************************************/
/*                                                   */
/*    PWM出　力　開始                                        */
/*                                                   */
/*****************************************************************************************************/
/*                                                   */
/*    備　考　:                                          */
/*    JL-086A不　具　合　回　避 の　た　め　、　5 回連 続書き 込み実施                    */
/*                                                   */
/*****************************************************************************************************/
inline void StartPWM( MICRO_AXIS_HANDLE *AxisRsc, SHORT BbSet )
```

```
{
  /* Carrier Counter Clock Set and PWM start  */
  CRST = AxisRsc->MicroIf.CarrierClk | 0x0001;

  /* Release Soft BB */
  BBSET = BbSet;
  BBSET = BbSet;
  BBSET = BbSet;
  BBSET = BbSet;
  BBSET = BbSet;

#ifdef  MULTI_AXIS
  AxisRsc++;

  /* Carrier Counter Clock Set and PWM start  */
  CRST_2 = AxisRsc->MicroIf.CarrierClk | 0x0001;

  /* Release Soft BB */
  BBSET_2 = BbSet;
  BBSET_2 = BbSet;
  BBSET_2 = BbSet;
  BBSET_2 = BbSet;
  BBSET_2 = BbSet;
#endif
}


/*****************************************************************************************************/
/*                                                   */
/*    PWM出  力                                              */
/*                                                   */
/*****************************************************************************************************/
inline void SetPWM( MICRO_AXIS_HANDLE *AxisRsc )
{
  PwmT2 = AxisRsc->PwmV.PwmCntT2;
  PwmT1 = AxisRsc->PwmV.PwmCntT1;
  PwmT0 = AxisRsc->PwmV.PwmCntT0;
```

```
#ifdef  MULTI_AXIS
  AxisRsc++;
  PwmT2_2 = AxisRsc->PwmV.PwmCntT2;
  PwmT1_2 = AxisRsc->PwmV.PwmCntT1;
  PwmT0_2 = AxisRsc->PwmV.PwmCntT0;
#endif
}


/*****************************************************************************************************/
/*                                                   */
/*    キ ャ リ ア 周 波 数変更(各軸 )                                  */
/*                                                   */
/*****************************************************************************************************/
inline  void  ChangeCarrierFreq( MICRO_AXIS_HANDLE *AxisRsc, SHORT axno )
{
  if( axno == 0 )
  {
    if( AxisRsc->IntAdV.CarrierFreq != AxisRsc->MicroIf.CarrierFreqIn )
    {
      AxisRsc->IntAdV.CarrierFreq = AxisRsc->MicroIf.CarrierFreqIn; /* Carrier Buffer Change  */
      CRFRQ = AxisRsc->IntAdV.CarrierFreq;              /* Carrier Freq. Change */
    }
  }
#ifdef  MULTI_AXIS
  else if( axno == 1 )
  {
    if ( AxisRsc->IntAdV.CarrierFreq != AxisRsc->MicroIf.CarrierFreqIn )
    {
      AxisRsc->IntAdV.CarrierFreq = AxisRsc->MicroIf.CarrierFreqIn; /* Carrier Buffer Change  */
      CRFRQ_2 = AxisRsc->IntAdV.CarrierFreq;              /* Carrier Freq. Change */
    }
  }
#endif
  else
  {
```

```
    ; //処　理　なし
  }
}


/*****************************************************************************************************/
/*                                                   */
/*    キ　ャ　リ　ア　周 波　数変更(全軸 )                                  */
/*                                                   */
/*****************************************************************************************************/
inline  void  ChangeCarrierFreqAll( MICRO_AXIS_HANDLE *AxisRsc )
{
  if( AxisRsc->IntAdV.CarrierFreq != AxisRsc->MicroIf.CarrierFreqIn )
  {
    AxisRsc->IntAdV.CarrierFreq = AxisRsc->MicroIf.CarrierFreqIn; /* Carrier Buffer Change  */
    CRFRQ = AxisRsc->IntAdV.CarrierFreq;              /* Carrier Freq. Change   */
  }
#ifdef  MULTI_AXIS
  AxisRsc++;
  if ( AxisRsc->IntAdV.CarrierFreq != AxisRsc->MicroIf.CarrierFreqIn )
  {
    AxisRsc->IntAdV.CarrierFreq = AxisRsc->MicroIf.CarrierFreqIn; /* Carrier Buffer Change  */
    CRFRQ_2 = AxisRsc->IntAdV.CarrierFreq;              /* Carrier Freq. Change   */
  }
#endif
}


/*****************************************************************************************************/
/*                                                   */
/*    デ　シ　メ　ー　シ　ョ　ン　フ　ィルタ エラーク リア                                */
/*                                                   */
/*****************************************************************************************************/
inline  void  SdmErrClrRequest( USHORT axno )
{
  if( axno == 0 )
  {
```

```
    SDMECLR = 0x000F;                   /* Decimation filter Error Clear  */
  }
#ifdef  MULTI_AXIS
  else if( axno == 1 )
  {
    SDMECLR_2 = 0x000F;                   /* Decimation filter Error Clear  */
  }
#endif
  else
  {
    ; //処  理  なし
  }
}


/*****************************************************************************************************/
/*                                                   */
/*    デ  シ  メ  ー  シ  ョ  ン  フ  ィ ルタ エラーク リア                              */
/*                                                   */
/*****************************************************************************************************/
inline  void  CurAdSyncRequest( void )          /* 電  流A D サ  イ  ク  ル同期要求       */
{
  ADSYNC = 0x0001;
#ifdef  MULTI_AXIS
  ADSYNC_2 = 0x0001;
#endif
}


/*****************************************************************************************************/
/*                                                   */
/*    A/D convert data loading                                  */
/*                                                   */
/*****************************************************************************************************/
/*    IuDetTmp = (IuGain * ( IuAD + IuOffset )) * KnorCurrent / 2^14                */
/*    IvDetTmp = (IvGain * ( IuAD + IvOffset )) * KnorCurrent / 2^14                */
/*    IuDetect = (IuGain * ( IuAD + IuOffset )) * KnorCurrent / 2^14 * 3 / 2            */
```

```
/*    IvDetect = (IvGain * ( IuAD + IvOffset )) * KnorCurrent / 2^14 * 3 / 2            */
/*****************************************************************************************/
inline void ADConvDataLoad( MICRO_AXIS_HANDLE *AxisRsc )
{
  SHORT swk;
  DWREG lwk0;

  /* swk = IuAD >> 2(算　術) */
  swk = mulshr( IuAD, (SHORT)OneR, 2 );
  /* lwk0.l = (swk + IuOffset) * IuGain */
  lwk0.l = mul( (swk + AxisRsc->CurDet.IuOffset), AxisRsc->CurDet.IuGain );
  /* lwk0.l = (swk + IuOffset) * IuGain * KnorCurrent / 2^14 (算　術) */
  AxisRsc->CurDet.IuDetTmp = mulshr( lwk0.l, AxisRsc->CurDet.KnorCurrent, 14 );
  /* IuDetect = (swk + IvOffset) * IvGain * KnorCurrent / 2^14 * 3 / 2 (算　術) * /
  AxisRsc->CurDet.IuDetect = mulshr( AxisRsc->CurDet.IuDetTmp, (LONG)No3, 1 );
  /*-------------------------------------------------------------------------*/
  /* swk = IvAD >> 2(算　術) */
  swk = mulshr( IvAD, (SHORT)OneR, 2 );
  /* lwk0.l = (swk + IvOffset) * IvGain */
  lwk0.l = mul( (swk + AxisRsc->CurDet.IvOffset), AxisRsc->CurDet.IvGain );
  /* lwk0.l = (swk + IvOffset) * IvGain * KnorCurrent / 2^14 (算　術　シ フト )   */
  AxisRsc->CurDet.IvDetTmp = mulshr( lwk0.l, AxisRsc->CurDet.KnorCurrent, 14 );
  /* IvDetect = (swk + IvOffset) * IvGain * KnorCurrent / 2^14 * 3 / 2 (算　術) * /
  AxisRsc->CurDet.IvDetect = mulshr( AxisRsc->CurDet.IvDetTmp, (LONG)No3, 1 );

#ifdef  MULTI_AXIS

  AxisRsc++;
  /* swk = IuAD >> 2(算　術) */
  swk = mulshr( IuAD_2, (SHORT)OneR, 2 );
  /* lwk0.l = (swk + IuOffset) * IuGain */
  lwk0.l = mul( (swk + AxisRsc->CurDet.IuOffset), AxisRsc->CurDet.IuGain );
  /* lwk0.l = (swk + IuOffset) * IuGain * KnorCurrent / 2^14 (算　術) */
  AxisRsc->CurDet.IuDetTmp = mulshr( lwk0.l, AxisRsc->CurDet.KnorCurrent, 14 );
  /* IuDetect = (swk + IvOffset) * IvGain * KnorCurrent / 2^14 * 3 / 2 (算　術) * /
  AxisRsc->CurDet.IuDetect = mulshr( AxisRsc->CurDet.IuDetTmp, (LONG)No3, 1 );
  /*-------------------------------------------------------------------------*/
```

```
  /* swk = IvAD >> 2(算　術) */
  swk = mulshr( IvAD_2, (SHORT)OneR, 2 );
  /* lwk0.l = (swk + IvOffset) * IvGain */
  lwk0.l = mul( (swk + AxisRsc->CurDet.IvOffset), AxisRsc->CurDet.IvGain );
  /* lwk0.l = (swk + IvOffset) * IvGain * KnorCurrent / 2^14 (算　術　シ フト )　*/
  AxisRsc->CurDet.IvDetTmp = mulshr( lwk0.l, AxisRsc->CurDet.KnorCurrent, 14 );
  /* IvDetect = (swk + IvOffset) * IvGain * KnorCurrent / 2^14 * 3 / 2 (算　術) * /
  AxisRsc->CurDet.IvDetect = mulshr( AxisRsc->CurDet.IvDetTmp, (LONG)No3, 1 );

#endif
}


/*****************************************************************************************************/
/*                                                   */
/*    SVIP異　常　状　態取 得                                    */
/*                                                   */
/*****************************************************************************************************/
inline  void  GetSvipErrorSts( MICRO_AXIS_HANDLE *AxisRsc )
{
  AxisRsc->StsFlg.FccStsMon = FCCDAT;
  AxisRsc->StsFlg.FltStsW = FLTSTAT & 0x7FFF;
#ifdef  MULTI_AXIS
  AxisRsc++;
  AxisRsc->StsFlg.FccStsMon = FCCDAT_2;
  AxisRsc->StsFlg.FltStsW = FLTSTAT_2 & 0x7FFF;
#endif
}


/************************************* end of file ***********************************************/
```