```
/*****************************************************************************************************/
/*                                                   */
/*                                                   */
/*    ScanI.c : Mercury電 流 制 御 プ ロ グラム                                  */
/*                                                   */
/*                                                   */
/*****************************************************************************************************/
/*                                                   */
/*                                                   */
/*                                                   */
/*                                                   */
/*                                                   */
/*                                                   */
/*                                                   */
/*                                                   */
/*                                                   */
/************** Copyright (C) Yaskawa Electric Corporation ***************************************/
/*                                                   */
/*    Rev.0.00 : 2012.08.06  Y.Tanaka    ・ JL-086 向 け 電 流 制 御 C 言 語 化 用 にバー ジョン取り直し    */
/*    Rev.0.01 : 2012.08.17  Y.Tanaka    ・ 構 造 体 、 ロ ー カル変数 見直し              */
/*    Rev.0.02 : 2012.08.20  Y.Tanaka    ・ 構 造 体 、 ロ ー カル変数 見直し              */
/*    Rev.0.03 : 2012.11.20  Y.Tanaka    ・ 多 軸 対 応 、 コ ンパイラ 確認用              */
/*                                                   */
/*    <1> 2013.05.07 T.Yamada   イ ン ト リ ン シ ッ ク関数に 変 更                  */
/*    <2> 2013.05.07 T.Yamada   記 述 見 直 し                                */
/*    <3> 2013.05.09 T.Yamada   MpSQRT修 正                                  */
/*    <4> 2013.05.13 T.Yamada   ア セ ン ブ ラ と の 違い修正                        */
/*****************************************************************************************************/
//#include "Basedef.h"
/*---------------------------------------------------------------------------------------------------*/
#include "IxInst.h"
#include "MprgStruct.h"
#include "MpConstTbl.h"          /* 定 数 テ ー ブ ル読み込み                */

#if defined(WIN32)
#include "IlibSvc.h"          /* VC版 の み で使用                    */
#include "MprgLmtChkVCMacro.h"      /* 加 減 算 リ ミ ッ ト 検 出用マクロ定義          */
```

```
#endif

//#define DEBUG_OUTPT   /* for debug Romsimの  実  行  箇  所 確認用      */


/*****************************************************************************************************/
/*    Definitions                                                */
/*****************************************************************************************************/
#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                */

#define MULTI_AXIS              /* 多  軸  処  理有効                    */
#define USE_CMOVE //<2> t-yamada



/* 周  辺  レ  ジ  ス  タ  定  義  (暫定処理？)     */
#ifdef  PREG_DEF
#include "equ.h"
/* read reg */
int chess_storage(PFREG:0x6BD) FCCST;
int chess_storage(PFREG:0x6D0) IuAD;
int chess_storage(PFREG:0x6D1) IvAD;
int chess_storage(PFREG:0x6D9) HSUR0;
int chess_storage(PFREG:0x6DA) HSUR1;
int chess_storage(PFREG:0x6DD) CTSTR;
int chess_storage(PFREG:0x6DF) FLTSTAT;
/* write reg  */
int chess_storage(PFREG:0x6D0) OUTPT;
int chess_storage(PFREG:0x6D1) WDT1L;
int chess_storage(PFREG:0x6D2) BBSET;
int chess_storage(PFREG:0x6D3) CRST;
int chess_storage(PFREG:0x6D8) SDMECLR;
int chess_storage(PFREG:0x6D9) ADSYNC;
int chess_storage(PFREG:0x6DB) PWMOS;
int chess_storage(PFREG:0x6DC) CRSET1;
```

```
int chess_storage(PFREG:0x6DD) CTSTW;
int chess_storage(PFREG:0x6DF) CRFRQ;
int chess_storage(PFREG:0x6F9) DIVSET;
int chess_storage(PFREG:0x6FA) PCVS0;
int chess_storage(PFREG:0x6FB) PCVS1;
int chess_storage(PFREG:0x6FC) PCVS2;
int chess_storage(PFREG:0x6E7) PwmT0;
int chess_storage(PFREG:0x6E8) PwmT1;
int chess_storage(PFREG:0x6E9) PwmT2;
#endif    //#ifdef  PREG_DEF
/* read reg */
extern int chess_storage(PFREG:0x6D9) HSUR0;  //<2> //軸  共 通,tanaka21
extern int chess_storage(PFREG:0x6DA) HSUR1;  //<2> //軸  共 通,tanaka21
extern int chess_storage(PFREG:0x6D0) IuAD;   //<2>
extern int chess_storage(PFREG:0x6D1) IvAD;   //<2>
extern int chess_storage(PFREG:0x6DD) CTSTR;
extern int chess_storage(PFREG:0x7D0) IuAD_2; //<2>
extern int chess_storage(PFREG:0x7D1) IvAD_2; //<2>
/* write reg  */
extern int chess_storage(PFREG:0x6D0) OUTPT;  //<2> //軸  共 通,tanaka21
extern int chess_storage(PFREG:0x6D1) WDT1L;  //<2> //軸  共 通,tanaka21
extern int chess_storage(PFREG:0x6DD) CTSTW;
extern int chess_storage(PFREG:0x6E7) PwmT0;  //<2>
extern int chess_storage(PFREG:0x6E8) PwmT1;  //<2>
extern int chess_storage(PFREG:0x6E9) PwmT2;  //<2>
extern int chess_storage(PFREG:0x7E7) PwmT0_2;  //<2>
extern int chess_storage(PFREG:0x7E8) PwmT1_2;  //<2>
extern int chess_storage(PFREG:0x7E9) PwmT2_2;  //<2>


INITWK  IniWk;/* for dubug */



/*****************************************************************************************************/
/*    ProtoType                                                */
/*****************************************************************************************************/
```

```
void  MpDataClear( MICRO_AXIS_HANDLE *AxisRsc );      /* マ イ ク ロ 用 デ ータクリア        */
void  MpIntHost( void );
void  MpIntAD( void ) property(isr);
void  MpIntEnc( void );
inline USHORT MpSQRT( ULONG src );    /* 2013.05.06 tanaka21 コ ー ド 整理<020>     */
inline void MpOVMMODK( INTADP *IntAdP, INTADV *IntAdV, CSHORT*  pCtbl );    /* 2013.05.06 tanaka21 コ ー ド 整理<020>     */
inline void ADConvDataLoad( MICRO_AXIS_HANDLE *AxisRsc ); //<2>
inline void SetPWM( MICRO_AXIS_HANDLE *AxisRsc ); //<2> /* <S015> */

#if defined(WIN32)    /* VC用 ダ ミ ー レ ジスタ定義   */
SVIP_READ_REG SvIpReadReg;
SVIP_WRITE_REG  SvIpWriteReg;
#endif

/* 機 能 レ ジ ス タ / 周辺レジ ス タ ( 0 x 5 F 0 以 降 )を使用 す る た め に 定 義 が必要 --> コンパイラ変更により不要、
#define FREG_DEF    /* 機 能 レ ジ ス タ定義有効   */
//#define PREG_DEF    /* 周 辺 レ ジ ス タ定義有効     */
/* 機 能 レ ジ ス タ 定 義 (暫定処理？)     */
#ifdef  FREG_DEF
int chess_storage(ISA0) ISA0;
int chess_storage(ISA1) ISA1;
int chess_storage(IL) INTLVWR;
int chess_storage(EIX) EIX;
int chess_storage(DIX) DIX;
#endif    //#ifdef  FREG_DEF

/*****************************************************************************************************/
/*                                                   */
/*    初 期 化 処 理                                        */
/*                                                   */
/*****************************************************************************************************/
#ifdef ASIP_CC
#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 //#ifndef IPD_SIM       /* IPDesigner用 シ ミ ュ レ ー シ ョ ンスイ ッチ    */
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  //#ifndef IPD_SIM       /* IPDesigner用 シ ミ ュ レ ー シ ョ ンスイ ッチ    */
```

```
#elif defined(WIN32)                    /* VC用                          */
void  MpStart( void )
#endif
{
  USHORT        ax_noR;
  MICRO_AXIS_HANDLE *AxisRscR;

  SHORT DivSetW;    /* 2013.05.06 tanaka21 コ  ー  ド 整理<020>     */
  SHORT PoSet1W;    /* 2013.05.06 tanaka21 コ  ー  ド 整理<020>     */
  SHORT PoSet2W;    /* 2013.05.06 tanaka21 コ  ー  ド 整理<020>     */
  USHORT  uswk;     /* 2013.05.06 tanaka21 コ  ー  ド 整理<020>     */

/*-----------------------------------------------------------------------------------------------*/
/*    interupt set                                          */
/*-----------------------------------------------------------------------------------------------*/
/*    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                */

/*-----------------------------------------------------------------------------------------------*/
/*    Get Axis Num 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   //#ifdef  MULTI_AXIS
  ax_noR = 0;
#endif    //#ifdef  MULTI_AXIS
  {
    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 Interrupt Level                                          */
/*-------------------------------------------------------------------------------------------------*/
  /* level(AD=3, INT1=4, HOST=0) */
  /* ★ H/W ア ク セ ス が 共 通 の も の を ま と め た い ！！0軸目っ て書くのが格好 悪い★    */
#ifdef  FREG_DEF
  INTLVWR = 0x0004;
#else //#ifdef  FREG_DEF
  AxisHdl[0].SvIpRegW->INTLVWR = 0x0004;
#endif  //#ifdef  FREG_DEF

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

    AxisRscR->StsFlg.BbSetW = 0x2004;           /* INT1=Encoder0, BB        */
    AxisRscR->SvIpRegW->BBSET = AxisRscR->StsFlg.BbSetW;  /* INT1=Encoder0, BB        */

    ISA0 = (int)MpIntAD;
```

```
//    ISA1 = (int)MpIntEnc;
/*-------------------------------------------------------------------------------------------------*/
    AxisRscR->SvIpRegW->PCVS0 = AxisRscR->EncIfV.DivPls.s[0]; /* パ ル ス 変換位置      (bit15-0)     */
/*-------------------------------------------------------------------------------------------------*/
    PoSet1W = AxisRscR->DivPlsV.PoSet1In; /* MpUPDATE_DIVPOS()で 比 較 処 理 が あ る た め 残 しておく   */
    PoSet2W = AxisRscR->DivPlsV.PoSet2In; /* MpUPDATE_DIVPOS()で 比 較 処 理 が あ る た め 残 しておく   */
    AxisRscR->SvIpRegW->PCVS1 = PoSet1W;  /* パ ル ス 変 換 原点補正1 (bit15-0)            */
    AxisRscR->SvIpRegW->PCVS2 = PoSet2W;  /* パ ル ス 変 換 原点補正2 (bit15-0)            */
/*-------------------------------------------------------------------------------------------------*/
    DivSetW = AxisRscR->DivPlsV.DivSetIn; /* MpUPDATE_DIVPOS()で 比 較 処 理 が あ る た め 残 しておく   */
    AxisRscR->SvIpRegW->DIVSET = DivSetW; /* 分 周 機 能設定                          */

/*-------------------------------------------------------------------------------------------------*/
/* 2013.05.06 tanaka21 コ ー ド 整 理 ( マクロ 化 ) <022>      */
    ZERO = 0x0000;  //<2>
    ONE  = 0x0001;  //<2>
//    /* 2012.12.21 Y.Oka 現 状 初 期 化必要 */
/*-------------------------------------------------------------------------------------------------*/

    AxisRscR->SinTbl.SinT = 0x0000; /* SinTbl.SinT= sin(θ )     sin(0)=    0.000 →  0000h    */
    AxisRscR->SinTbl.CosT = 0x4000; /* SinTbl.CosT= cos(θ )     cos(0)=    1.000 →  4000h    */
    AxisRscR->SinTbl.SinT2 = 0x376D;  /* SinTbl.SinT2=sin(θ +2 π/3)  sin(2π /3)=   0.866 →  376Dh      */
    AxisRscR->SinTbl.CosT2 = 0xE000;  /* SinTbl.CosT2=cos(θ +2 π/3)  cos(2π /3)=  -0.500 →  E000h      */
    AxisRscR->SinTbl.SinT3 = 0xC893;  /* SinTbl.SinT3=sin(θ -2 π/3)  sin(-2π /3)=-0.866  → C893h     */
    AxisRscR->SinTbl.CosT3 = 0xE000;  /* SinTbl.CosT3=cos(θ -2 π/3)  cos(-2π /3)=-0.500  → E000h     */
/*-------------------------------------------------------------------------------------------------*/
/*    PWM set                                          */
/*-------------------------------------------------------------------------------------------------*/
    AxisRscR->SvIpRegW->PWMOS = 0x0A0;              /* 2level,triangle,servo(bit7: no-Saw mode for JL-056)    */
    AxisRscR->IntAdV.CrFreqW = AxisRscR->IntAdP.CrFreq;     /* Carrier set(IntAdP.CrFreq must be set before starts)   */
    AxisRscR->SvIpRegW->CRSET1 = 0x10;              /* CLA=Both(unavailable on JL-056)              */
    AxisRscR->SvIpRegW->CRFRQ = AxisRscR->IntAdV.CrFreqW;   /* Carrier 10.667kHz                    */
    uswk = (AxisRscR->IntAdV.CrFreqW >> 1); /* TMP0 <-- IntAdV.CrFreqW /2(50p duty)          */
    AxisRscR->PwmV.PwmCntT2 = uswk;
    AxisRscR->PwmV.PwmCntT1 = uswk;
    AxisRscR->PwmV.PwmCntT0 = uswk;
```

```
//<2>   AxisRscR->SvIpRegW->PwmT2 = uswk;     /* T2(W) = (duty:50p)                  */
//<2>   AxisRscR->SvIpRegW->PwmT1 = uswk;     /* T1(V) = (duty:50p)                  */
//<2>   AxisRscR->SvIpRegW->PwmT0 = uswk;     /* T0(U) = (duty:50p)                  */
//    SetPWM(uswk, uswk, uswk);
/*-------------------------------------------------------------------------------------------------*/
/*    Clear Register                                          */
/*-------------------------------------------------------------------------------------------------*/
    MpDataClear( AxisRscR );
/*-------------------------------------------------------------------------------------------------*/
/*    input CPORT, DLIM = QLIM = 0, output CPORT                          */
/*-------------------------------------------------------------------------------------------------*/
//<2>#ifdef PREG_DEF
//    AxisRscR->StsFlg.CtrlStsRW = CTSTR; /* StsFlg.CtrlStsRW <- Control register              */
    AxisRscR->StsFlg.CtrlStsRW = AxisRscR->CtrlStsIn; /* StsFlg.CtrlStsRW <- Control register *//* <Y.Oka01> */
    AxisRscR->StsFlg.CtrlStsRW = ( AxisRscR->StsFlg.CtrlStsRW & DLIMI );  /* StsFlg.CtrlStsRW <-- StsFlg.CtrlStsRW & DLIMI
    (imm_16)           */
//    CTSTW = AxisRscR->StsFlg.CtrlStsRW; /* Status Set                          */
    AxisRscR->CtrlStsOut = AxisRscR->StsFlg.CtrlStsRW;  /* Status Set             *//* <Y.Oka01> */
/*-------------------------------------------------------------------------------------------------*/
/*    START : INTERRUPT, PWM                                  */
/*-------------------------------------------------------------------------------------------------*/
    EIX = 0x0;             /* Interuput start                      */

//    AxisRscR->SvIpRegW->CRST = 0x1;             /* Carrier(PWM) start                  */
    AxisRscR->SvIpRegW->CRST = 0x3;            /* Carrier(PWM) start JL-086の  設  定  に対応 2013.07.17     */
    AxisRscR->StsFlg.BbSetW = ( AxisRscR->StsFlg.BbSetW & 0xFFFB ); /* Reset soft_BB                      */
    AxisRscR->SvIpRegW->BBSET = AxisRscR->StsFlg.BbSetW;    /*                                 */
    AxisRscR->SvIpRegW->BBSET = AxisRscR->StsFlg.BbSetW;    /*    <S015>                       */
    AxisRscR->SvIpRegW->BBSET = AxisRscR->StsFlg.BbSetW;    /*    <S015>                       */
    AxisRscR->SvIpRegW->BBSET = AxisRscR->StsFlg.BbSetW;    /*    <S015>                       */
    AxisRscR->SvIpRegW->BBSET = AxisRscR->StsFlg.BbSetW;    /*    <S015>                       */
  }

  /* Output PWM Data */
  SetPWM( &AxisHdl[0] );
```

```
/*****************************************************************************************************/
/*                                                   */
/*    ROUND Procedure                                      */
/*                                                   */
/*****************************************************************************************************/
#if !defined(WIN32)
#ifndef IPD_SIM       /* IPDesigner用  シ  ミ  ュ  レ  ー  シ ョンスイッチ    */
  while (1)
#endif  //#ifndef IPD_SIM       /* IPDesigner用  シ  ミ  ュ  レ  ー  シ ョ ンスイ ッチ    */
#endif
  {
#ifdef  MULTI_AXIS              /* 多  軸  処  理有効                    */
    for( ax_noR = 0; (SHORT)ax_noR < AxisNum; ax_noR++ )
#else   //#ifdef  MULTI_AXIS
    ax_noR = 0;
#endif    //#ifdef  MULTI_AXIS
    {
      AxisRscR = &AxisHdl[ax_noR];
/*-------------------------------------------------------------------------------------------------*/
/*    A/D error check and clear                                */
/*-------------------------------------------------------------------------------------------------*/
      AxisRscR->StsFlg.FccStsMon = AxisRscR->SvIpRegR->FCCST;
      AxisRscR->StsFlg.FltStsW = AxisRscR->SvIpRegR->FLTSTAT & 0x7FFF;
    }

/*-------------------------------------------------------------------------------------------------*/
/*    Host port check for host INT                             */
/*      現  在 、WREG1 00～WREG 1 0 4 ま  で  は  未  使 用の ため、削除。                  */
/*-------------------------------------------------------------------------------------------------*/
    /* ★ H/W ア  ク  セ  ス  が  共  通  の  も の  を  ま  と  め  た  い ！！0軸目っ て書くのが格好悪い★    */
//<2>#ifdef PREG_DEF
    if ( HSUR0 != 0x0 )
    {
      MpIntHost( );       /*                          */
    }

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

```
/*    Host port check for host INT2                                    */
/*-------------------------------------------------------------------------------------------------*/
    /* ★ H/W ア ク セ ス が 共 通 の も の を ま と め た い ！！0軸目っ て書くのが格好悪い★    */
//<2>#ifdef PREG_DEF
    if ( HSUR1 != 0x0 )
    {
      DIX = 0x0;    /* disable interupt   <V112>                */

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

        AxisRscR->PhaseV.PhaseH = AxisRscR->AdinV.PhaseHIn;       /*                                  */
        AxisRscR->PhaseV.PhaseIp = AxisRscR->PhaseV.PhaseIpIn;      /* 位 相 補 間量     <V112>              */
        AxisRscR->PhaseV.PhaseIpF = AxisRscR->PhaseV.PhaseIpFIn;    /* 位 相 補 間 フラグ   <V112>           */
        AxisRscR->PhaseV.PhaseIpFIn = 1;                /* 位 相 補 間 フ ラグセット <V112>                */
        AxisRscR->WeakFV.WfKpV.s[0] = AxisRscR->WeakFV.WfKpVLIn;    /* 電 圧F B 比 例 ゲ イン(下位16bit) <V214>          */
        AxisRscR->WeakFV.WfKpV.s[1] = AxisRscR->WeakFV.WfKpVHIn;    /* 電 圧F B 比 例 ゲ イン(上位16bit) <V214>        */
        AxisRscR->WeakFV.WfKiV.s[0] = AxisRscR->WeakFV.WfKiVLIn;    /* 電 圧F B 積 分 ゲ イン(下位16bit) <V214>          */
        AxisRscR->WeakFV.WfKiV.s[1] = AxisRscR->WeakFV.WfKiVHIn;    /* 電 圧F B 積 分 ゲ イン(上位16bit) <V214>        */
        AxisRscR->WeakFV.WfV1Max = AxisRscR->WeakFV.WfV1MaxIn;      /* 電 圧 指 令 制限値       <V214>          */
        AxisRscR->WeakFV.WfIdRefLim = AxisRscR->WeakFV.WfIdRefLimIn;  /* d軸 電 流 指 令 リミット     <V214>     */
      }

      EIX = 0x0;    /* enable interupt    <V112>                */
    }
  }
  return;
}


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

```
/*    HOST Interupt Procedure                                    */
/*                                                               */
/*****************************************************************************************************/
void  MpIntHost( void )
{
#ifdef  WIN32
  DWREG lmtBuf;     /* 加 減 演 算 用 リ ミ ッ ト判断用バッファ       */
  UCHAR lmtBufsign[2];  /* リ ミ ッ ト バ ッ フ ァ入力値符号   0:前 項 、 1 :後項   */
  UCHAR lmtBufSw;     /* リ ミ ッ ト バ ッ フ ァ 入力値スイッチ  0:前 項 、 1 :後項   */
#endif
  USHORT        ax_noH;
  USHORT        ActiveAxis;
  INT64       dlwk;
  MICRO_AXIS_HANDLE *AxisRscH;

  SHORT swk0;     /* 2013.05.06 tanaka21 コ ー ド 整理<020>     */
  SHORT swk1;     /* 2013.05.06 tanaka21 コ ー ド 整理<020>     */
  LONG  lwk1;     /* 2013.05.06 tanaka21 コ ー ド 整理<020>     */
  LONG  lwk2;     /* 2013.05.06 tanaka21 コ ー ド 整理<020>     */
  LONG  lwk3;     /* 2013.05.06 tanaka21 コ ー ド 整理<020>     */


  IniWk.IN_WK0++;   /* for debug counter tanaka21 */

  /* ★ H/W ア ク セ ス が 共 通 の も の を ま と め た い ！！0軸目っ て書く のが格好悪い★    */
  WDT1L = 0x1;    /* Watch dog set                    */
//  OUTPT = 0x1;    /* 1.13                          */


#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];
```

```
    AxisRscH->IntAdV.IqMon = AxisRscH->IntAdV.IqRef;  /* for CPU monitor                      */
/*-------------------------------------------------------------------------------------------*/
/*    キ  ャ  リ  ア  周  波  数 切り替え 処理                          <V057> <V075>    */
/*-------------------------------------------------------------------------------------------*/
    if ( AxisRscH->IntAdP.CrFreq != AxisRscH->IntAdV.CrFreqW )
    {
      AxisRscH->IntAdV.CrFreqW = AxisRscH->IntAdP.CrFreq;   /* Carrier Buffer Change              */
      AxisRscH->SvIpRegW->CRFRQ = AxisRscH->IntAdV.CrFreqW; /* Carrier Freq. Change               */
    }
  }

/*-------------------------------------------------------------------------------------------*/
/*    input from host                                    */
/*-------------------------------------------------------------------------------------------*/
  /* Check Current Ajust Request */
  ActiveAxis = 0;
#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];
//<2>#ifdef PREG_DEF
//    if ( ( CTSTR & RLOCK ) == 0 )
    if ( ( AxisRscH->CtrlStsIn & RLOCK ) == 0 )
    {
      ActiveAxis |= 0x01 << ax_noH; /* ビ  ッ  ト 登録 */
    }
  }


  if( ActiveAxis != 0 )
  { /* 電  流  検  出  調  整要求あり */
    /* ★ H/W ア  ク  セ  ス  が  共  通  の  も の  を  ま  と  め  た  い ！！0軸目っ て書くのが格好悪い★    */
    DIX = 0x0;    /* disable interupt   <V112>             */
```

```
#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( 0 != (ActiveAxis & (0x01 << ax_noH)) )
      {
        AxisRscH->IntAdV.IuOffset = AxisRscH->AdinV.IuOffsetIn; /* IntAdV.IuOffset <-- AdinV.IuOffsetIn                    */
        AxisRscH->IntAdV.IvOffset = AxisRscH->AdinV.IvOffsetIn; /* IntAdV.IvOffset <-- AdinV.IvOffsetIn                    */
        AxisRscH->IntAdP.Kcu = AxisRscH->AdinV.KcuIn;     /* IntAdP.Kcu <-- AdinV.KcuIn                    */
        AxisRscH->IntAdP.Kcv = AxisRscH->AdinV.KcvIn;     /* IntAdP.Kcv <-- AdinV.KcvIn                    */
      }
    }

    /* ★ H/W ア  ク  セ  ス  が  共  通  の  も の  を  ま  と  め  た  い ！！0軸目っ て書くのが格好悪い★    */
    EIX = 0x0;    /* enable interupt    <V112>              */
  }
/*-------------------------------------------------------------------------------------------------------*/
/*    interupt  enable                                   */
/*-------------------------------------------------------------------------------------------------------*/
  /* ★ H/W ア  ク  セ  ス  が  共  通  の  も の  を  ま  と  め  た  い ！！0軸目っ て書くのが格好悪い★    */
  DIX = 0x0;    /* disable interupt   <V112>              */

#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];
    AxisRscH->PhaseV.PhaseH = AxisRscH->AdinV.PhaseHIn;       /*                                  */
    AxisRscH->PhaseV.PhaseIp = AxisRscH->PhaseV.PhaseIpIn;      /* 位  相  補 間量      <V112>              */
    AxisRscH->PhaseV.PhaseIpF = AxisRscH->PhaseV.PhaseIpFIn;    /* 位  相  補  間 フラグ    <V112>           */
    AxisRscH->PhaseV.PhaseIpFIn = 1;                /* 位  相  補  間  フ  ラグセット <V112>              */
```

```
    AxisRscH->WeakFV.Vel = AxisRscH->AdinV.VelIn;         /*                              */
    AxisRscH->IntAdV.TLimP = AxisRscH->AdinV.TLimPIn;       /*                              */
    AxisRscH->IntAdV.TLimM = AxisRscH->AdinV.TLimMIn;       /*                              */
    AxisRscH->IntAdP.Kvv = AxisRscH->IntAdP.KvvIn;          /* for AVR                          */
    AxisRscH->VcmpV.VdRef = AxisRscH->AdinV.VdRefIn;        /*                              */
    AxisRscH->VcmpV.VqRef = AxisRscH->AdinV.VqRefIn;        /*                              */
    AxisRscH->IntAdV.IqDist = AxisRscH->IntAdV.IqDistIn;      /* <V224>                         */
    AxisRscH->WeakFV.WfKpV.s[0] = AxisRscH->WeakFV.WfKpVLIn;    /* 電  圧F B 比  例 ゲ  イン(下位16bit) <V214>          */
    AxisRscH->WeakFV.WfKpV.s[1] = AxisRscH->WeakFV.WfKpVHIn;    /* 電  圧F B 比  例 ゲ  イン(上位16bit) <V214>        */
    AxisRscH->WeakFV.WfKiV.s[0] = AxisRscH->WeakFV.WfKiVLIn;    /* 電  圧F B 積  分 ゲ  イン(下位16bit) <V214>          */
    AxisRscH->WeakFV.WfKiV.s[1] = AxisRscH->WeakFV.WfKiVHIn;    /* 電  圧F B 積  分 ゲ  イン(上位16bit) <V214>        */
    AxisRscH->WeakFV.WfV1Max = AxisRscH->WeakFV.WfV1MaxIn;      /* 電  圧  指  令 制限値       <V214>          */
    AxisRscH->WeakFV.WfIdRefLim = AxisRscH->WeakFV.WfIdRefLimIn;  /* d軸  電  流  指  令 リミット     <V214>      */
  }

  /* ★ H/W ア  ク  セ  ス  が  共  通  の  も  の  を  ま  と  め  た  い ！！0軸目っ て書くのが格好 悪い★    */
  EIX = 0x0;    /* enable interupt    <V112>              */

/*-----------------------------------------------------------------------------------------------------*/
/*    Carrier Freq Change check : if( status & BB ) Carrier Freq. change              */
/*-----------------------------------------------------------------------------------------------------*/
  /* Check Current Ajust Request */
  ActiveAxis = 0;
#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->IntAdP.FccRst != 0)
    {
      ActiveAxis |= 0x01 << ax_noH; /* ビ  ッ  ト 登録 */
      IniWk.IN_WK0H++;    /* for debug counter tanaka21 */
    }
  }
```

```
  if( ActiveAxis != 0 )
  { /* 電　流　検　出　調　整要求あり */
#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( 0 != (ActiveAxis & (0x01 << ax_noH)) )
      {
        /* 不　具 合No. 15は0 7 6 A の　不　具　合　の　ため対策は省略可能<00 2>(tanaka21)*/
        AxisRscH->SvIpRegW->SDMECLR = ( AxisRscH->SvIpRegR->FCCST | 8 );
        AxisRscH->AdStop.ADRst = AxisRscH->IntAdP.FccRst;
        AxisRscH->IntAdP.FccRst = 0;
      }
    }
//    AxisRscH->SvIpRegW->ADSYNC = 1;
  }

/*-----------------------------------------------------------------------------------------------*/
  /* Check BB Status */
  ActiveAxis = 0;
#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];
//<2>#ifdef PREG_DEF
//    if ( CTSTR & BB )/* <Y.Oka01> */
    if ( AxisRscH->CtrlStsIn & BB )
    {
      ActiveAxis |= 0x01 << ax_noH; /* ビ　ッ　ト 登録 */
    }
```

```
  }

  if( ActiveAxis != 0 )
  { /* BB状 態 の 軸 が ある場合 */
    /* ★ H/W ア ク セ ス が 共 通 の も の を ま と め た い ！！0軸目っ て書くのが格好悪い★    */
    DIX = 0x0;    /* disable interupt   <V112>                */

#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];
/*-------------------------------------------------------------------------------------------------*/
/*    data clear while BB                                  */
/*-------------------------------------------------------------------------------------------------*/
      if( 0 != (ActiveAxis & (0x01 << ax_noH)) )
      { /* BB中 の 軸 の場合 */
        MpDataClear( AxisRscH );

        if( AxisRscH->IntAdP.CrFreq == AxisRscH->IntAdV.CrFreqW )
        {
          AxisRscH->IntAdV.CrFreqW = AxisRscH->IntAdP.CrFreq;   /* Carrier Buffer Change                */
          AxisRscH->SvIpRegW->CRFRQ = AxisRscH->IntAdV.CrFreqW; /* Carrier Freq. Change                 */
        }
      }
    }

    /* ★ H/W ア ク セ ス が 共 通 の も の を ま と め た い ！！0軸目っ て書くのが格好悪い★    */
    EIX = 0x0;    /* enable interupt    <V112>                */
  }


#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
  {
    if( 0 == (ActiveAxis & (0x01 << ax_noH)) )
    { /* BB中  で  は  な  い 軸の場合 */
      AxisRscH = &AxisHdl[ax_noH];

/*****************************************************************************************************/
/*    notch filter 1st (before 2nd filter)                                                           */
/*****************************************************************************************************/
/*    input    : AdinV.IqIn  (max:15000)                                                             */
/*    output     : IntAdV.IqOut1L (max:15000,limit:32768)                                            */
/*    parameter : IntAdP.Kf11,IntAdP.Kf12,IntAdP.Kf13,IntAdP.Kf14 (KFx= Kfx * 8192)                  */
/*    buffer     : IntAdV.IqIn1PL,IntAdV.IqIn1PPL,IntAdV.IqOut1PL,IntAdV.IqOut1PPL                   */
/*****************************************************************************************************/
      if( AxisRscH->IntAdP.CtrlSw & F1DSABL )     /* Notch filter1 Disable    */
      {
        AxisRscH->IntAdV.IqOut1L.s[0] = AxisRscH->AdinV.IqIn; /* フ  ィ  ル  タ  処理なし       */
      }
      else
      {
/*-----------------------------------------------------------------------------------------------*/
/*    lwk1 = IntAdP.Kf12 * AdinV.IqIn + IntAdP.Kf11 * IntAdV.IqIn1PL + IntAdP.Kf14 * IntAdV.IqIn1PPL */
/*-----------------------------------------------------------------------------------------------*/
        lwk1 = mul(AxisRscH->IntAdP.Kf12, AxisRscH->AdinV.IqIn);
        lwk1 = mac((LONG)AxisRscH->IntAdP.Kf11, AxisRscH->IntAdV.IqIn1PL.l, lwk1);
        lwk1 = mac_limitf((LONG)AxisRscH->IntAdP.Kf14, AxisRscH->IntAdV.IqIn1PPL.l, lwk1);      /* 符  号 付32b i t 制 限処理
        */

/*-----------------------------------------------------------------------------------------------*/
/*    lwk1  = lwk1 - (IntAdP.Kf11 * IntAdV.IqOut1PL + IntAdP.Kf13 * IntAdV.IqOut1PPL)             */
/*-----------------------------------------------------------------------------------------------*/
        lwk2 = mulshr_limitf((LONG)AxisRscH->IntAdP.Kf11, AxisRscH->IntAdV.IqOut1PL.l, 13);
        lwk3 = mulshr_limitf((LONG)AxisRscH->IntAdP.Kf13, AxisRscH->IntAdV.IqOut1PPL.l, 13);    /* AxisRscH->IntAdP.Kf13 *
        AxisRscH->IntAdV.IqOut1PPL.l    */
        lwk1 = lwk1 - lwk2 - lwk3;
/*-----------------------------------------------------------------------------------------------*/
```

```
/*    IntAdV.IqIn1PPL = IntAdV.IqIn1PL, IntAdV.IqIn1PL = AdinV.IqIn, IntAdV.IqOut1PPL = IntAdV.IqOut1PL, IntAdV.IqOut1PL =
lwk1          */
/*-----------------------------------------------------------------------------------------------------*/
        AxisRscH->IntAdV.IqIn1PPL.l = AxisRscH->IntAdV.IqIn1PL.l; /* <V388> 追  加                        */
        AxisRscH->IntAdV.IqIn1PL.l = (LONG)AxisRscH->AdinV.IqIn;  /* <V388> 追  加                          */
        AxisRscH->IntAdV.IqOut1PPL.l = AxisRscH->IntAdV.IqOut1PL.l; /* <V388> 追  加                      */
        AxisRscH->IntAdV.IqOut1PL.l = lwk1;   /* <V388> 追  加                        */
        AxisRscH->IntAdV.IqOut1BufL.l = lwk1;   /*        <V502> 追  加               */

        AxisRscH->IntAdV.IqOut1L.s[0] = asr_limitf( AxisRscH->IntAdV.IqOut1BufL.l, 13 );  /*        <V502> 追  加       */

      }

/*****************************************************************************************************/
/*    notch filter                                     */
/*****************************************************************************************************/
/*    input    : IntAdV.IqOut1L (max:15000)                                  */
/*    output    : IntAdV.IqOut3L (max:15000,limit:32768)                          */
/*    parameter : IntAdP.Kf31,IntAdP.Kf32,IntAdP.Kf33,IntAdP.Kf34 (KF3x= Kf3x * 8192)                  */
/*    buffer    : IQI3P,IQI3PP,IQO3P,IQO3PP                                  */
/*****************************************************************************************************/
      if( AxisRscH->IntAdP.CtrlSw & F3DSABL )
      {
        AxisRscH->IntAdV.IqOut3L.s[0] = AxisRscH->IntAdV.IqOut1L.s[0];  /* フ  ィ  ル  タ  処理なし     */
      }
      else
      {
/*-----------------------------------------------------------------------------------------------------*/
/*    HTMP0 = IntAdP.Kf32 * IntAdV.IqOut1L + IntAdP.Kf31 * IQI3P + IntAdP.Kf34 * IQI3PP                    */
/*-----------------------------------------------------------------------------------------------------*/
        lwk1 = mul(AxisRscH->IntAdP.Kf32, AxisRscH->IntAdV.IqOut1L.s[0]);
        lwk1 = mac((LONG)AxisRscH->IntAdP.Kf31, AxisRscH->IntAdV.IqIn3PL.l, lwk1);
        lwk1 = mac_limitf((LONG)AxisRscH->IntAdP.Kf34, AxisRscH->IntAdV.IqIn3PPL.l, lwk1);

/*-----------------------------------------------------------------------------------------------------*/
/*    HTMP0  = HTMP0 - (IntAdP.Kf31 * IQO3P + IntAdP.Kf33 * IQO3PP)                    */
/*-----------------------------------------------------------------------------------------------------*/
```

```
        lwk2 = mulshr_limitf((LONG)AxisRscH->IntAdP.Kf31, AxisRscH->IntAdV.IqOut3PL.l, 13);

        lwk3 = mulshr_limitf((LONG)AxisRscH->IntAdP.Kf33, AxisRscH->IntAdV.IqOut3PPL.l, 13);

        lwk1 = lwk1 - lwk2 - lwk3;
/*-------------------------------------------------------------------------------------------------*/
/*    IQI3PP = IQI3P, IQI3P = IQO1, IQO3PP = IQO3P, IQO3P = HTMP0                                  */
/*-------------------------------------------------------------------------------------------------*/
        AxisRscH->IntAdV.IqIn3PPL.l = AxisRscH->IntAdV.IqIn3PL.l;     /* 前 々 回 値保存                    */
        AxisRscH->IntAdV.IqIn3PL.l = (LONG)AxisRscH->IntAdV.IqOut1L.s[0]; /* 前 回 値 保存                  */
        AxisRscH->IntAdV.IqOut3PPL.l = AxisRscH->IntAdV.IqOut3PL.l;     /* 前 々 回 値保存                */
        AxisRscH->IntAdV.IqOut3PL.l = lwk1;       /* 前 回 値 保存                                    */
        AxisRscH->IntAdV.IqOut3BufL.l = lwk1;       /* 整 数 化 前 出 力 今回値保存                   */

        AxisRscH->IntAdV.IqOut3L.s[0] = asr_limitf(lwk1, 13);
      }

/*************************************************************************************************/
/*    Low Pass Filter                                                                            */
/*************************************************************************************************/
/*    IntAdP.TLpf    : Time-constant                                                             */
/*    IntAdV.IqOut1Lpf : Output(32 bit) .. IQO1F: High 16 bit                                    */
/*    IntAdV.IqOut3   : INPUT                                                                    */
/*************************************************************************************************/
      if( AxisRscH->IntAdP.CtrlSw & LPFDSABL )
      {
        AxisRscH->IntAdV.IqOut1Lpf.s[1] = AxisRscH->IntAdV.IqOut3L.s[0];  /* フ ィ ル タ 処理なし  */
      }
      else
      {
        AxisRscH->IntAdV.IqOut3 = AxisRscH->IntAdV.IqOut3L.s[0];  /* フ ィ ル タ 処理なし  */

        swk1 = sub_limitf(AxisRscH->IntAdV.IqOut3, AxisRscH->IntAdV.IqOut1Lpf.s[1]);

        lwk2 = mul(AxisRscH->IntAdP.TLpf, swk1) << 2;

        AxisRscH->IntAdV.IqOut1Lpf.l = add_limitf(lwk2, AxisRscH->IntAdV.IqOut1Lpf.l);  /* HTMP0 <-- limit(HTMP0, 2^15 - 1)
```

```
        */
      }

/*****************************************************************************************************/
/*    notch filter (before data input)                                                               */
/*****************************************************************************************************/
/*    input    : IQO1F (max:15000)                                                                   */
/*    output   : IntAdV.IqOut2L (max:15000,limit:32768)                                              */
/*    parameter : IntAdP.Kf21,IntAdP.Kf22,IntAdP.Kf23,IntAdP.Kf24 (KF2x= Kf2x * 8192)                */
/*    buffer   : IQI2P,IQI2PP,IQO2P,IQO2PP                                                           */
/*****************************************************************************************************/
      if( AxisRscH->IntAdP.CtrlSw & F2DSABL )
      {
        AxisRscH->IntAdV.IqOut2L.s[0] = AxisRscH->IntAdV.IqOut1Lpf.s[1];  /* <V388> 追  加                */
      }
      else
      {
/*-----------------------------------------------------------------------------------------------*/
/*    HTMPO = IntAdP.Kf22 * IQO1F + IntAdP.Kf21 * IQI2P + IntAdP.Kf24 * IQI2PP                    */
/*-----------------------------------------------------------------------------------------------*/
        lwk1 = mul(AxisRscH->IntAdP.Kf22, AxisRscH->IntAdV.IqOut1Lpf.s[1]);
        lwk1 = mac((LONG)AxisRscH->IntAdP.Kf21, AxisRscH->IntAdV.IqOut2PL.l, lwk1);
        lwk1 = mac_limitf((LONG)AxisRscH->IntAdP.Kf24, AxisRscH->IntAdV.IqIn2PPL.l, lwk1);

/*-----------------------------------------------------------------------------------------------*/
/*    HTMPO  = HTMPO - (IntAdP.Kf21 * IQOP + IntAdP.Kf23 * IQOPH)                                */
/*-----------------------------------------------------------------------------------------------*/
        lwk2 = mulshr_limitf((LONG)AxisRscH->IntAdP.Kf21, AxisRscH->IntAdV.IqOut2PL.l, 13);

        lwk3 = mulshr_limitf((LONG)AxisRscH->IntAdP.Kf23, AxisRscH->IntAdV.IqOut2PPL.l, 13);

        lwk1 = lwk1 - lwk2 - lwk3;

/*-----------------------------------------------------------------------------------------------*/
/*    IQI2PP = IQI2P, IQI2P = IQO1F, IQO2PP = IQO2P, IQO2P = HTMPO                                */
/*-----------------------------------------------------------------------------------------------*/
        AxisRscH->IntAdV.IqIn2PPL.l = AxisRscH->IntAdV.IqIn2PL.l;     /* 前  々  回  値保存          */
```

```
        AxisRscH->IntAdV.IqIn2PL.l = (LONG)AxisRscH->IntAdV.IqOut1Lpf.s[1]; /* 前　回　値 保存                    */
        AxisRscH->IntAdV.IqOut2PPL.l = AxisRscH->IntAdV.IqOut2PL.l;     /* 前　々　回　値保存                  */
        AxisRscH->IntAdV.IqOut2PL.l = lwk1;       /* 前　回　値 保存                          */
        AxisRscH->IntAdV.IqOut2BufL.l = lwk1;       /* 整　数　化　前　出　力 今回値保存              */

        AxisRscH->IntAdV.IqOut2L.s[0] = asr_limitf(lwk1, 13);

      }
    }

/*-------------------------------------------------------------------------------------------------*/
/*    omega calculation                                    */
/*-------------------------------------------------------------------------------------------------*/
    swk0 = mulshr(AxisRscH->IntAdP.Ld, AxisRscH->WeakFV.Vel, 15);
    lwk1 = mul(swk0, AxisRscH->IntAdV.KEangle);
    swk0 = asr_limitf( lwk1, 0 );

    swk1 = mulshr(AxisRscH->IntAdP.Lq, AxisRscH->WeakFV.Vel, 15);
    lwk1 = mul(swk1, AxisRscH->IntAdV.KEangle);
    swk1 = asr_limitf( lwk1, 0 );
  }


/*-------------------------------------------------------------------------------------------------*/
/*    data transmit(2)                                    */
/*-------------------------------------------------------------------------------------------------*/
  /* ★ H/W ア　ク　セ　ス　が　共　通　の　も の　を　ま　と　め　た　い ！！0軸目っ て書くのが格好悪い★    */
  DIX = 0x0;    /* disable interupt   <V112>              */

#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];
```

```
    AxisRscH->VcmpV.MagC = (SHORT)mulshr(AxisRscH->IntAdP.Mag, AxisRscH->WeakFV.Vel, 15);   /* VcmpV.MagC <-- ACC >> 15
    */
    AxisRscH->VcmpV.LdC = swk0; /* VcmpV.LdC                                */
    AxisRscH->VcmpV.LqC = swk1; /* VcmpV.LqC                                */

/*-------------------------------------------------------------------------------------------------*/
    AxisRscH->WeakFV.IqOut = AxisRscH->IntAdV.IqOut2L.s[0]; /* <V388> 追  加                        */
    if( (AxisRscH->IntAdP.CtrlSw & V_FB) == 0 )
    {
      AxisRscH->WeakFV.IdOut = AxisRscH->AdinV.IdIn;    /* WeakFV.IdOut(reference)                  */
    }

/* 分  周  パ  ルス は  H/W化予定 */
/*-------------------------------------------------------------------------------------------------*/
/*    分  周  パ  ル  ス 更新 処理                                  <V720>  */
/*-------------------------------------------------------------------------------------------------*/
//    swk1 = EncIfV.BitIprm;  /* DivWk0 <-- EncIfV.BitIprm                */

//    if( AxisRscH->EncIfV.BitIprm & UPGDIVOUT )
//    {
//      MpUPDATE_DIVPOS( );     /*  --> 分  周  パ  ル ス更新 ,etc                */
//    }
/*-------------------------------------------------------------------------------------------------*/
//<2>#ifdef PREG_DEF
//    AxisRscH->StsFlg.CtrlStsRW = CTSTR; /* StsFlg.CtrlStsRW <- Control register              */
    AxisRscH->StsFlg.CtrlStsRW = AxisRscH->CtrlStsIn; /* StsFlg.CtrlStsRW <- Control register *//* <Y.Oka01> */
    AxisRscH->StsFlg.CtrlStsRW = ( AxisRscH->StsFlg.CtrlStsRW & DLIMI );  /* StsFlg.CtrlStsRW <-- StsFlg.CtrlStsRW & DLIMI
    (imm_16)            *///110525tanaka21,こ  の  ビ  ッ  ト  演  算  は必要な の  か ？
    AxisRscH->StsFlg.CtrlStsRW = ( AxisRscH->StsFlg.CtrlStsRW & TLIMI );  /* StsFlg.CtrlStsRW <-- StsFlg.CtrlStsRW & TLIMI
    (imm_16)            */
/*-------------------------------------------------------------------------------------------------*/
  }

  /* ★ H/W ア  ク  セ  ス  が  共  通  の  も の  を  ま  と  め  た  い ！！0軸目っ て書くのが格好悪い★    */
  EIX = 0x0;    /* enable interupt    <V112>                */

  return;
```

```
}



/*****************************************************************************************************/
/*                                                   */
/*    AD Interupt Procedure                                  */
/*                                                   */
/*    マ イ ク ロ 分 周 機 能 にてエンコーダ 割 込 ( @ I NT_ENC)追加 の た め 割込レベル(INTLVWR)マスク処理変更   */
/*****************************************************************************************************/
void  MpIntAD( void ) property(isr)
{
#ifdef  WIN32
  DWREG lmtBuf;     /* 加 減 演 算 用 リ ミ ッ ト判断用バッファ       */
  UCHAR lmtBufsign[2];  /* リ ミ ッ ト バ ッ フ ァ入力値符号    0:前 項 、 1 :後項   */
  UCHAR lmtBufSw;     /* リ ミ ッ ト バ ッ フ ァ 入力値スイッチ  0:前 項 、 1 :後項   */
#endif
  USHORT        ax_noI;
  INT64       dlwk;
  MICRO_AXIS_HANDLE *AxisRscI;


  SHORT swk0;       /* 16bitワ ー ク レ ジスタ0  2013.05.06 tanaka21 コ ー ド 整理<021>      */
  SHORT swk1;       /* 16bitワ ー ク レ ジスタ1  2013.05.06 tanaka21 コ ー ド 整理<021>      */
  SHORT swk2;       /* 16bitワ ー ク レ ジスタ2  2013.05.06 tanaka21 コ ー ド 整理<021>      */
  SHORT swk3;       /* 16bitワ ー ク レ ジスタ3  2013.05.06 tanaka21 コ ー ド 整理<021>      */
  SHORT swk4;       /* 16bitワ ー ク レ ジスタ4  2013.05.06 tanaka21 コ ー ド 整理<021>      */
  SHORT swk5;       /* 16bitワ ー ク レ ジスタ5  2013.05.06 tanaka21 コ ー ド 整理<021>      */
  SHORT swk6;       /* 16bitワ ー ク レ ジスタ6  2013.05.06 tanaka21 コ ー ド 整理<021>      */
  SHORT swk7;       /* 16bitワ ー ク レ ジスタ7  2013.05.06 tanaka21 コ ー ド 整理<021>      */
  SHORT swk8;       /* 16bitワ ー ク レ ジスタ8  2013.05.06 tanaka21 コ ー ド 整理<021>      */
  CSHORT* pCtbl;        /* テ ー ブ ル ポ イ ン タ 用ワークレジスタ 2013.05.06 tanaka21 コ ー ド 整理<021>    */
  LONG  lwk0;       /* 32bitワ ー ク レ ジスタ0  2013.05.06 tanaka21 コ ー ド 整理<021>      */
  LONG  lwk1;       /* 32bitワ ー ク レ ジスタ1  2013.05.06 tanaka21 コ ー ド 整理<021>      */
  LONG  lwk2;       /* 32bitワ ー ク レ ジスタ2  2013.05.06 tanaka21 コ ー ド 整理<021>      */
  LONG  lwk4;       /* 32bitワ ー ク レ ジスタ4  2013.05.06 tanaka21 コ ー ド 整理<021>      */
```

```
  LONG  lwk6;       /* 32bitワ ー ク レ ジスタ6  2013.05.06 tanaka21 コ ー ド 整理<021>      */
//  LONG  lwk8;       /* 32bitワ ー ク レ ジスタ 8 2013.05.06 tanaka21 コ ー ド 整理<021>      *//* コ メ ン ト アウト (1
*/
  SHORT swk10;  //<2>
  SHORT swk11;  //<2>

  SHORT PwmCnt;

  IniWk.IN_WK1++;   /* for debug counter tanaka21 */
  /* ★ H/W ア ク セ ス が 共 通 の も の を ま と め た い ！！0軸目っ て書くのが格好悪い★    */
  /* level(AD=0, INT1=0/4 HOST=0) */
  INTLVWR &= 0x00F0;

//<2>#ifdef PREG_DEF
  OUTPT = 0x1;
  WDT1L = 0x0;    /* Watch dog reset */


/*-----------------------------------------------------------------------------------------------------*/
/*    A/D convert data loading                                          */
/*-----------------------------------------------------------------------------------------------------*/
/*    IntAdV.IuInData = IntAdP.Kcu * ( IUS + IntAdV.IuOffset ) / 2^8                                   */
/*    IntAdV.IvInData = IntAdP.Kcv * ( IVS + IntAdV.IvOffset ) / 2^8                                   */
/*-----------------------------------------------------------------------------------------------------*/
  ADConvDataLoad( &AxisHdl[0] );

  {/* Axis0 start */
  /* Execute Current Loop Main Operation */
    AxisRscI = &AxisHdl[0];
//=================================================================
// 位 相 補 間処理     <V112>
//=================================================================
    swk10 = AxisRscI->PhaseV.PhaseH + AxisRscI->PhaseV.PhaseIp;
    AxisRscI->PhaseV.PhaseIpF = cmove((AxisRscI->PhaseV.PhaseIpF != 1), ONE, AxisRscI->PhaseV.PhaseIpF);
    AxisRscI->PhaseV.PhaseH   = cmove((AxisRscI->PhaseV.PhaseIpF != 1), AxisRscI->PhaseV.PhaseH, swk10);
//=================================================================
// PHASE_UPDATE処 理   <V112>
```

```
//=================================================================
/*-----------------------------------------------------------------------------------------------*/
/*    theta calculation                                          */
/*-----------------------------------------------------------------------------------------------*/
    swk0 = AxisRscI->PhaseV.PhaseH;
    swk0 = swk0 + 32;           /* TMP3 <-- PhaseV.PhaseH + 2^5 */
    swk1 = PI23;
    swk2 = swk1 + swk0; /* TMP4 <-- PhaseV.PhaseH + 2PI/3 */
    swk3 = swk0 - swk1; /* TMP5 <-- PhaseV.PhaseH - 2PI/3 */
/*-----------------------------------------------------------------------------------------------*/
/*    table read and get iu,iv by Id,Iq reference                          */
/*-----------------------------------------------------------------------------------------------*/
    swk1 = swk0 >> 6;       /* TMP1 <-- TMP3 >> 6 */
    IxTblSin16( AxisRscI->SinTbl.SinT, swk1 );    /* SinTbl.SinT <-- stable[ TMP1 ] *//* tanaka21,要　コ　メ　ン　ト解除 */
    swk0 = swk0 + PI2;        /* TMP3 <-- TMP3 + PI/2 */
    swk1 = swk0 >> 6;       /* TMP1 <-- TMP3 >> 6 */
    IxTblSin16( AxisRscI->SinTbl.CosT, swk1 );    /* SinTbl.CosT <-- stable[ TMP1 ] *//* tanaka21,要　コ　メ　ン　ト解除 */

    swk1 = swk3 >> 6;       /* TMP1 <-- TMP5 >> 6 */
    IxTblSin16( AxisRscI->SinTbl.SinT3, swk1 );   /* SinTbl.SinT3 <-- stable[ TMP1 ] *//* tanaka21,要　コ　メ　ン　ト解除　*/
    swk3 = swk3 + PI2;        /* TMP5 <-- TMP5 + PI/2 */
    swk1 = swk3 >> 6;       /* TMP1 <-- TMP5 >> 6 */
    IxTblSin16( AxisRscI->SinTbl.CosT3, swk1 );   /* SinTbl.CosT3 <-- stable[ TMP1 ] *//* tanaka21,要　コ　メ　ン　ト解除　*/

    swk1 = swk2 >> 6;       /* TMP1 <-- TMP4 >> 6 */
    IxTblSin16( AxisRscI->SinTbl.SinT2, swk1 );   /* SinTbl.SinT2 <-- stable[ TMP1 ] *//* tanaka21,要　コ　メ　ン　ト解除　*/
    swk2 = swk2 + PI2;        /* TMP4 <-- TMP4 + PI/2 */
    swk1 = swk2 >> 6;       /* TMP1 <-- TMP4 >> 6 */
    IxTblSin16( AxisRscI->SinTbl.CosT2, swk1 );   /* SinTbl.CosT2 <-- stable[ TMP1 ] *//* tanaka21,要　コ　メ　ン　ト解除　*/

/*-----------------------------------------------------------------------------------------------*/
/*    dq-trans(UVW to DQ)                                        */
/*-----------------------------------------------------------------------------------------------*/
/*    ID = IntAdP.Kc * ( (SinTbl.CosT-SinTbl.CosT2)*IntAdV.IuInData/2^14 + (SinTbl.CosT3-SinTbl.CosT2)*IntAdV.IvInData/2^14 )
/2^9             */
/*    IQ = IntAdP.Kc * ( (SinTbl.SinT2-SinTbl.SinT)*IntAdV.IuInData/2^14 + (SinTbl.SinT2-SinTbl.SinT3)*IntAdV.IvInData/2^14 )
/2^9             */
```

```
/*-----------------------------------------------------------------------------------------------*/
    /* TMP1 <-- cos(th) - cos(th-2pi/3) */
    swk1 = AxisRscI->SinTbl.CosT - AxisRscI->SinTbl.CosT2;
    /* ACC <-- TMP1 * iu */
    swk2 = mulshr(swk1, AxisRscI->IntAdV.IuInData, 14 );
    /* TMP1 <-- cos(th-2pi/3)-cos(th+2pi/3) */
    swk1 = AxisRscI->SinTbl.CosT3 - AxisRscI->SinTbl.CosT2;
    /* ACC <-- TMP1 * iv */
    swk1 = mulshr(swk1, AxisRscI->IntAdV.IvInData, 14 );
    /* TMP2 <-- TMP2 + TMP1 */
    swk2 = swk1 + swk2;
    /* ACC <-- IntAdP.Kc * TMP2 */
    AxisRscI->IntAdV.IdInData = mulshr(AxisRscI->IntAdP.Kc, swk2, 9 );
/*-----------------------------------------------------------------------------------------------*/
    swk1 = AxisRscI->SinTbl.SinT2 - AxisRscI->SinTbl.SinT;                      /* TMP1 <-- sin(th+2pi/3) - sin(th)
    */
    swk2 = mulshr(swk1, AxisRscI->IntAdV.IuInData, 14 );  /* ACC <-- TMP1 * iu                    */
    swk1 = AxisRscI->SinTbl.SinT2 - AxisRscI->SinTbl.SinT3;                     /* TMP1 <-- sin(th+2pi/3)-sin(th-2pi/3)
    */
    swk1 = mulshr(swk1, AxisRscI->IntAdV.IvInData, 14 );  /* ACC <-- TMP1 * iv                    */
    swk2 = swk1 + swk2;                         /* TMP2 <-- TMP2 + TMP1            */
    AxisRscI->IntAdV.IqInData = mulshr(AxisRscI->IntAdP.Kc, swk2, 9 );    /* ACC <-- IntAdP.Kc * TMP2                    */

/*-----------------------------------------------------------------------------------------------*/
/*    Current Observer  <V038>                                    */
/*-----------------------------------------------------------------------------------------------*/
//===========================================
//  電  流  オ  ブ  ザ  ー バ スイッチ
//===========================================
    if( AxisRscI->IntAdP.CtrlSw & OBSSEL )
    {
//===========================================
//  ダ  ン  ピ  ン  グ  ゲ イ ンの設定  <V076>
//===========================================
//<2>     AxisRscI->DobsV.DmpGain = 2;
//===========================================
//  q軸  電  流  の  飽  和 チェック  <V076>
```

```
//=========================================
      if( AxisRscI->IntAdV.IqInData >= 0 )
      { /* 0以　上　の　とき */
        /* TMP3 = IntAdV.IqInData */
        swk2 = AxisRscI->IntAdV.IqInData;
      }
      else          /* 負　の　とき       */
      {
        swk2 = ~AxisRscI->IntAdV.IqInData;  /* TMP3 = ~IntAdV.IqInData;
        *///110530tanaka21作　業　メ　モ　、　－１掛　け　る　の　と　ど　っ　ちが速い？
        swk2 = swk2 + 1;  /* TMP3 = TMP3 + 1                        */
      }
      if( swk2 <= 14250 )
      {
        swk3 = ZERO;    /* TMP4 = 0 ( OverFlowCheck = OK )              */
      }
      else
      {
        swk3 = ONE;   /* TMP4 = 1 ( OverFlowCheck = NG )              */
      }
//=====================================
//  d軸　オ　ブ　ザ　ーバ 部
//=====================================
      swk0 = mulshr(AxisRscI->DobsP.TsPerL, AxisRscI->VcmpV.VdOut, 15 );    /* TMP0 <-- ACC >> 15   ( TMP0 = Ts/L * Vd_out >>
      15 )  */
      swk2 = AxisRscI->IntAdV.IdInData; /* TMP3 <-- IntAdV.IdInData         <V076>            */
      swk2 = limit(swk2, 15000);
      swk1 = swk2 - AxisRscI->DobsV.IdObsOut; /*                 <V076>      */
      swk1 = mulshr(AxisRscI->DobsP.Gobs, swk1, 16 ); /* ACC  <-- TMP2*DobsP.Gobs   ( TMP2 = g * ( Id - Id_obs ) )  */
      swk0 = swk1 + swk0; /* TMP0 <-- TMP0 + TMP2   ( TMP0 = ( g*(Id-Id_obs)>>16 ) + (Ts/L*Vd_out>>15) )  */
      swk1 = mulshr(AxisRscI->DobsP.RLTs, AxisRscI->DobsV.IdObsOut, 12 ); /* TMP2 <-- DobsV.IqObsOut      ( TMP2 = Id_obs )
      */
      AxisRscI->DobsV.IdObsOut = add_limitf(swk1, swk0);  /* DobsV.IdObsOut <-- limit( DobsV.IdObsOut, 2^15-1 )          */
//=====================================
//  d軸　フ　ィ　ルタ部
//=====================================
//-------------------------------------
```

```
//  error obs
//-------------------------------------
      swk0 = AxisRscI->IntAdV.IdInData - AxisRscI->DobsV.IdObsOut;  /*                              */
//-------------------------------------
//  low pass filter
//-------------------------------------
      swk0 = sub_limitf(swk0, AxisRscI->DobsV.LpfIld.s[1]);
      lwk2 = mul(AxisRscI->DobsP.FilObsGain, swk0 ) << 2; /*                                */
      AxisRscI->DobsV.LpfIld.l = add_limitf(lwk2, AxisRscI->DobsV.LpfIld.l);
//-------------------------------------
//  high pass filter
//-------------------------------------
      swk0 = sub_limitf(AxisRscI->DobsV.LpfIld.s[1], AxisRscI->DobsV.HpfIld.s[1]);
      lwk2 = mul(AxisRscI->DobsP.FilObsGain, swk0 ) << 2; /*                                */

      AxisRscI->DobsV.HpfIld.l = add_limitf(lwk2, AxisRscI->DobsV.HpfIld.l);  /*                                */
      AxisRscI->DobsV.IdObsFreq = AxisRscI->DobsV.LpfIld.s[1] - AxisRscI->DobsV.HpfIld.s[1];  /*                      */
//-------------------------------------
//  IntAdV.IdInData = IntAdV.IdInData - DobsV.IdObsFreq
//-------------------------------------
      AxisRscI->DobsV.IdObsFreq = AxisRscI->DobsV.IdObsFreq * 2;    /* ACC  <-- DobsV.IdObsFreq * DobsV.DmpGain
      */
      AxisRscI->DobsV.IdObsFreq = cmove((swk3 != 0), ZERO, AxisRscI->DobsV.IdObsFreq);
      AxisRscI->IntAdV.IdInData = AxisRscI->IntAdV.IdInData - AxisRscI->DobsV.IdObsFreq;  /*                              */
//=====================================
//  q軸  オ  ブ  ザ ーバ 部
//=====================================
      swk0 = mulshr(AxisRscI->DobsP.TsPerL, AxisRscI->VcmpV.VqOut, 15 );  /* ACC  <-- TMP0*Ts/L    ( TMP0 = Ts/L * Vq_out)
      */
      swk2 = AxisRscI->IntAdV.IqInData;                                  /* TMP3 <-- IntAdV.IqInData       <V076>
      */
      swk2 = limit(swk2, 15000);                        /* TMP3 <-- Limit(15000)  <V076>           */
      swk1 = swk2 - AxisRscI->DobsV.IqObsOut;                          /*              <V076>        */
      swk1 = mulshr(AxisRscI->DobsP.Gobs, swk1, 16 );   /* TMP2 <-- ACC >> 16    ( TMP2 = g * ( Iq - Iq_obs ) >> 16 )  */
      swk0 = swk1 + swk0;                      /* TMP0 <-- TMP0 + TMP2    ( TMP0 = ( g*(Iq-Iq_obs)>>16 ) + (Ts/L*Vq_out>>15) )
      */
      swk1 = mulshr(AxisRscI->DobsP.RLTs, AxisRscI->DobsV.IqObsOut, 12 );   /* TMP2 <-- ACC >> 12    ( TMP2 = (1-R*Ts/L)*Iq_obs
```

```
      >> 12 ) */
      AxisRscI->DobsV.IqObsOut = add_limitf(swk1, swk0);                          /* DobsV.IqObsOut <-- limit( DobsV.IqObsOut,
      2^15-1 )            */
//====================================
//  q軸  フ  ィ  ルタ部
//====================================
//------------------------------------
//  error obs
//------------------------------------
      swk0 = AxisRscI->IntAdV.IqInData - AxisRscI->DobsV.IqObsOut;  /*                          */
//------------------------------------
//  low pass filter
//------------------------------------
      swk0 = sub_limitf(swk0, AxisRscI->DobsV.LpfIlq.s[1]); /*                             */
      lwk2 = mul(AxisRscI->DobsP.FilObsGain, swk0) << 2;  /*                           */
      AxisRscI->DobsV.LpfIlq.l = add_limitf(lwk2, AxisRscI->DobsV.LpfIlq.l);  /*                              */
//------------------------------------
//  high pass filter
//------------------------------------
      swk0 = sub_limitf(AxisRscI->DobsV.LpfIlq.s[1], AxisRscI->DobsV.HpfIlq.s[1]);  /*                              */
      lwk2 = mul(AxisRscI->DobsP.FilObsGain, swk0) << 2;  /*                           */
      AxisRscI->DobsV.HpfIlq.l = add_limitf(lwk2, AxisRscI->DobsV.HpfIlq.l);  /*                              */
      AxisRscI->DobsV.IqObsFreq = AxisRscI->DobsV.LpfIlq.s[1] - AxisRscI->DobsV.HpfIlq.s[1];  /*                    */
//------------------------------------
//  IntAdV.IqInData = IntAdV.IqInData - DobsV.IqObsFreq
//------------------------------------
      AxisRscI->DobsV.IqObsFreq = AxisRscI->DobsV.IqObsFreq * 2;    /* ACC  <-- DobsV.IqObsFreq * DobsV.DmpGain
      */
      AxisRscI->DobsV.IqObsFreq = cmove((swk3 != 0), ZERO, AxisRscI->DobsV.IqObsFreq);
      AxisRscI->IntAdV.IqInData = AxisRscI->IntAdV.IqInData - AxisRscI->DobsV.IqObsFreq;  /*                          */
    }

/*-------------------------------------------------------------------------------------------------*///110526tanaka21,BBチ  ェ
ッ  ク  処  理  、  処  理  順  をいろいろ変更。
/*    Base Block Check                                  *///if-else if-elseの  形  で  書  き  換  え  。  正  し  く 動  作 するか要 確
/*-------------------------------------------------------------------------------------------------*/
    if( AxisRscI->AdStop.ADRst != 0 )
```

```
    {
      AxisRscI->AdStop.ADRst = 0;
      swk6 = AxisRscI->IntAdV.CrFreqW >> 1;
      AxisRscI->PwmV.PwmCntT2 = swk6;
      AxisRscI->PwmV.PwmCntT1 = swk6;
      AxisRscI->PwmV.PwmCntT0 = swk6;
/*-----------------------------------------------------------------------------------------------------*/
    }
    /* 2012.12.20 Y.Oka 誤　り　修正 */
    else if( (AxisRscI->StsFlg.CtrlStsRW & BB) != 0 )
    {
/*-----------------------------------------------------------------------------------------------------*/
      swk6 = AxisRscI->IntAdV.CrFreqW >> 1;
      AxisRscI->PwmV.PwmCntT2 = swk6;
      AxisRscI->PwmV.PwmCntT1 = swk6;
      AxisRscI->PwmV.PwmCntT0 = swk6;
/*-----------------------------------------------------------------------------------------------------*/
    }
    else
    {
/*****************************************************************************************************/
/*                                                   */
/*    弱　め　界 磁 用　Ｉｄ指 令計算処 理　　<V214>                          */
/*                                                   */
/*****************************************************************************************************/
/*-----------------------------------------------------------------------------------------------------*/
/*    弱　め　界　磁　方式選 択                                    */
/*-----------------------------------------------------------------------------------------------------*/
      if( AxisRscI->IntAdP.CtrlSw & V_FB )
      {
/*-----------------------------------------------------------------------------------------------------*/
/*    差　分　電　圧 作成                                  */
/*      Vq*と　基　準 電 圧(√(IntAdP.Vmaxˆ2-V ｄ ˆ ２ ) ) を　比　較　し 、差 分電圧を作る。                */
/*-----------------------------------------------------------------------------------------------------*/
/*-----------------------------------------------------------------------------------------------------*/
//    Vqmax = √ ( VmaxXˆ2 - Vdˆ2 )                                  *
/*-----------------------------------------------------------------------------------------------------*/
```

```
        lwk2 = AxisRscI->WeakFV.WfV1Max * AxisRscI->WeakFV.WfV1Max; /* IntAdP.Vmax^2                                  */
        lwk4 = AxisRscI->WeakFV.WfVdRef * AxisRscI->WeakFV.WfVdRef; /* Vd^2                ; 削  除 <V309>   復  活<V531>  */
        lwk2 = sub_limitf(lwk2, lwk4);
        lwk2 = limitz( lwk2, LPX_REG32_MAX ); /* if (IntAdP.Vmax^2 - Vd^2)< 0, then (IntAdP.Vmax^2 - Vd^2) = 0  */
        swk0 = MpSQRT( lwk2 );          /* √ ( IntAdP.Vmax^2 - Vd^2 )                    */
        if( swk0 > 0x7FFF )
        {
          swk0 = 0x7FFF;  /*                                  */
        }
        AxisRscI->WeakFV.WfVqMax = swk0;  /* Vqmax = √ ( IntAdP.Vmax^2 - Vd^2 )             */
/*-------------------------------------------------------------------------------------------------*/
//    TMP0 = Vqmax - Vq                                  *
/*-------------------------------------------------------------------------------------------------*/
        swk1 = AxisRscI->WeakFV.WfVqRef;
        if( swk1 < 0 )
        {
          swk1 = (SHORT)ZEROR - swk1; /* TMP1 = |Vq|                */
        }
        swk0 = sub_limitf(AxisRscI->WeakFV.WfVqMax, swk1);
/*-------------------------------------------------------------------------------------------------*/
/*    比  例  項 計 算                                    */
/*-------------------------------------------------------------------------------------------------*/
        lwk1 = (LONG)swk0;  /* TMP1,0 = 符  号  拡張(TMP0)                  */
        swk2 = (SHORT)mulshr( lwk1 ,AxisRscI->WeakFV.WfKpV.l, 32 );
        if( swk2 > (SHORT)0x0080 )
        {
          swk2 = LPX_REG16_MAX; /* 正  の  最 大値                        */
        }
        else if( swk2 < (SHORT)0xFF80 )
        {
          swk2 = LPX_REG16_MIN; /* 負  の  最 大値                        */
        }
        else
        {
          lwk2 = mulshr16( lwk1, AxisRscI->WeakFV.WfKpV.l);
          swk2 = mulshr( lwk2, (LONG)256, 16 );
        }
```

```
/*-------------------------------------------------------------------------------------------------*/
/*    積　分　項 計 算                                                                             */
/*-------------------------------------------------------------------------------------------------*/
        lwk4 = lwk1 * AxisRscI->WeakFV.WfKiV.l; /* Δ Vq * Kiv                    */
        lwk6 = mulshr( lwk1, AxisRscI->WeakFV.WfKiV.l, 32 );  /* Δ Vq * Kiv                    */
        if( (SHORT)lwk6 > 0x08 )
        {
          lwk4 = LPX_REG32_MAX; /* 正　の　最 大値                    */
        }
        else if( (USHORT)lwk6 > 0xFFF8 )
        {
          lwk4 = LPX_REG32_MIN; /* 負　の　最 大値                    */
        }
        else
        {
          lwk4 = lwk4 >> 4; /*                              */
          lwk4 = lwk4 & 0x0fffffff; /*                                 */
          lwk6 = lwk6 << 28;  /*                              */
          lwk4 = lwk6 | lwk4; /* TMP5,4 = Δ Vq * Kiv (* 2^16)         */
        }
        AxisRscI->WeakFV.WfIntgl.l = add_limitf(lwk4, AxisRscI->WeakFV.WfIntgl.l);  /*                        */
        lwk6 = (ULONG)AxisRscI->WeakFV.WfIntegLim << 16;  /* TMP9,8 = WeakFV.WfIntegLim * 2^16           */
        AxisRscI->WeakFV.WfIntgl.l = limit( AxisRscI->WeakFV.WfIntgl.l, lwk6 ); /* WFINTEGH = Δ Vq * Kiv (* 2^16 / 2^16)  */
/*-------------------------------------------------------------------------------------------------*/
/*    比　例 項　+　 積分 項                                                                       */
/*-------------------------------------------------------------------------------------------------*/
        swk4 = add_limitf(AxisRscI->WeakFV.WfIntgl.s[1], swk2);
        swk4 = limit( swk4, AxisRscI->WeakFV.WfIdRefLim );  /* IdrefLimで　リ　ミ ッ ト           */
/*-------------------------------------------------------------------------------------------------*/
/*    Idref > 0 な　ら　ば、Idref =　 0,積分 =　0                                                 */
/*      Idref(d軸　電　流 指　令　) が　正　に　な　る　こ　と　は　無　い 。正になった 場合は０にする。          */
/*-------------------------------------------------------------------------------------------------*/
        AxisRscI->WeakFV.IdOut = swk4;
        swk10 = AxisRscI->WeakFV.IdOut;
        AxisRscI->WeakFV.IdOut = cmove((swk10 > 0), ZERO, AxisRscI->WeakFV.IdOut);
        AxisRscI->WeakFV.WfIntgl.l = cmove((swk10 > 0), (LONG)ZEROR, AxisRscI->WeakFV.WfIntgl.l);
    }
```

```
/*****************************************************************************************************/
/*                                                   */
/*    ACRd(d軸　電　流 制 御)                                  */
/*                                                   */
/*****************************************************************************************************/
/*---------------------------------------------------------------------------------------------------*/
/*    TMP1 = limit( WeakFV.IdOut - IntAdV.IdInData , 2^15 - 1)                                       */
/*---------------------------------------------------------------------------------------------------*/
      swk1 = sub_limitf(AxisRscI->WeakFV.IdOut, AxisRscI->IntAdV.IdInData); /* TMP1 <-- limit( TMP1 , 2^15 - 1 )        */
/*---------------------------------------------------------------------------------------------------*/
/*    TMP2 = limit( IntAdP.KdP * TMP1 / 2^9 , 2^15 - 1 )                                             */
/*---------------------------------------------------------------------------------------------------*/
      swk2 = mulshr_limitf(AxisRscI->IntAdP.KdP, swk1, 9);  /* ACC <-- IntAdP.KdP * TMP1                    */
/*---------------------------------------------------------------------------------------------------*/
/*    IdIntgl(32) = (IntAdP.KdI * TMP1)<<3 + IdIntgl(32)                                             */
/*    IDIH = limit( IDIH , IntAdP.VdLim )                                                            */
/*---------------------------------------------------------------------------------------------------*/
      lwk4 = ((ULONG)AxisRscI->IntAdP.VdLim) << 16; /*                                               */
      lwk6 = mul(AxisRscI->IntAdP.KdI, swk1 ) << 3; /*                                               */
      AxisRscI->AcrV.IdIntgl.l = add_limitf(lwk6, AxisRscI->AcrV.IdIntgl.l);  /* AcrV.IdIntgl <-- limit( AcrV.IdIntgl , 2^31 -
      1 )          */
      if( LPX_ABS(AxisRscI->AcrV.IdIntgl.l) > LPX_ABS(lwk4) )
      {
        AxisRscI->StsFlg.CtrlStsRW = AxisRscI->StsFlg.CtrlStsRW | DLIM;   /*                          */
        swk0 = AxisRscI->IntAdP.CtrlSw;
        AxisRscI->AcrV.IdIntgl.l = cmove(((AxisRscI->IntAdP.CtrlSw & ICLR) != 0), (LONG)ZEROR, AxisRscI->AcrV.IdIntgl.l);
      }
/*---------------------------------------------------------------------------------------------------*/
/*    VcmpV.VdOut = limit( TMP2 + IDIH +TMP3, 2^15 - 1 )                                             */
/*---------------------------------------------------------------------------------------------------*/
      swk1 = add_limitf(AxisRscI->AcrV.IdIntgl.s[1], swk2); /* TMP1 <-- limit( TMP1 , 2^15 - 1 )        */
/*---------------------------------------------------------------------------------------------------*/
/*    filter : AcrV.VdFil =  ( ( ( TMP1 - VDFH ) * IntAdP.Tfil ) << 2 ) + AcrV.VdFil                 */
/*---------------------------------------------------------------------------------------------------*/
      swk1 = sub_limitf(swk1, AxisRscI->AcrV.VdFil.s[1]); /* TMP1 <-- limit( TMP1 , 2^15 - 1 )        */
      lwk0 = mul(AxisRscI->IntAdP.Tfil, swk1) << 2; /*                                               */
```

```
      AxisRscI->AcrV.VdFil.l = add_limitf(AxisRscI->AcrV.VdFil.l, lwk0);  /*                                  */

/*****************************************************************************************************/
/*                                                  */
/*    ACRq(q軸  電  流 制 御)                                    */
/*                                                  */
/*****************************************************************************************************/
/*---------------------------------------------------------------------------------------------------*/
/*    Low Pass Filter                                       */
/*---------------------------------------------------------------------------------------------------*/
/*    IntAdP.TLpf2 : Time-constant                                    */
/*    IntAdV.IqOut2Lpf : Output(32 bit) .. IQOF: High 16 bit                          */
/*    WeakFV.IqOut   : Input                                  */
/*---------------------------------------------------------------------------------------------------*/
/*    IQOF(32) =  ( ( ( WeakFV.IqOut - IQOF(16) ) * IntAdP.TLpf2 ) << 2 ) + IntAdV.IqOut2Lpf(32)              */
/*---------------------------------------------------------------------------------------------------*/
      if( (AxisRscI->IntAdP.CtrlSw & LPFCDSABL) != 0 )
      {
        AxisRscI->IntAdV.IqOut2Lpf.s[1] = AxisRscI->WeakFV.IqOut; /* disable LPF                           */
      }
/*---------------------------------------------------------------------------------------------------*/
      else
      {
        swk0 = sub_limitf(AxisRscI->WeakFV.IqOut, AxisRscI->IntAdV.IqOut2Lpf.s[1]); /* TMP0 <-- limit( TMP0, 2^15 - 1 )
        */
        lwk2 = mul(AxisRscI->IntAdP.TLpf2, swk0 ) << 2;
        AxisRscI->IntAdV.IqOut2Lpf.l = add_limitf(AxisRscI->IntAdV.IqOut2Lpf.l, lwk2);
      }
/*---------------------------------------------------------------------------------------------------*/
      AxisRscI->IntAdV.IqMonFil = AxisRscI->IntAdV.IqOut2Lpf.s[1];  /* IntAdV.IqMonFil:フ ィ ル タ後 の q 軸 電流（モ ニ タ用）<V224>
      */
      AxisRscI->IntAdV.IqOfRef = add_limitf(AxisRscI->IntAdV.IqOut2Lpf.s[1], AxisRscI->IntAdV.IqDist);  /* IntAdV.IqOfRef <--
      limit( IntAdV.IqOfRef , 2^15 - 1 )  <V224>    */
/*---------------------------------------------------------------------------------------------------*/
/*    Torque Limit:     <V214>                                */
/*      電 圧 フ ィ ー ド バ ッ ク 弱 め 界 磁 制 御 で d 軸 電 流 指 令 が作られるので、q軸電流指令は以下の式で    */
/*      求 め た 値 と ト ル ク リ ミ ッ ト 設 定 値 の いずれか小さい方で リミットする。          */
```

```
/*        Iq*リ  ミ  ッ ト値  = √(Imax^2-Id* ^2)                                  */
/*-------------------------------------------------------------------------------------------------*/
/*    Id*に  よ るTorque Lim it値       ;                                        */
/*-------------------------------------------------------------------------------------------------*/
      lwk2 = 0x0d693a40;  /* 15000^2                    */
      swk0 = AxisRscI->IntAdP.CtrlSw;
      swk1 = V_FB | V_FB2;
      swk0 = swk0 & swk1; /* TMP0の bit11,bit13 以  外  を  マ スクする      */
      if( swk0 != V_FB )
      {
        lwk4 = mul(AxisRscI->WeakFV.IdOut, AxisRscI->WeakFV.IdOut);     /* Idref^2              ; 削  除<V309 > 復活<V531>  */
      }
      else
      {
        lwk4 = mul(AxisRscI->WeakFV.WfIdRefLim, AxisRscI->WeakFV.WfIdRefLim); /* IdrefLim^2           ; <V309>    */
      }
      lwk2 = lwk2 - lwk4; /* Imax^2 - Id^2                */
      swk0 = MpSQRT( lwk2 );          /*                              */
      swk1 = swk0;  /* TMP0 = √ ( Imax^2 - Id^2 )             */

/*-------------------------------------------------------------------------------------------------*/
/*    Torque Limit                                          */
/*-------------------------------------------------------------------------------------------------*/
      if( AxisRscI->IntAdV.IqOfRef >= 0 )
      {
        swk1 = limit( swk1, AxisRscI->IntAdV.TLimP ); /* 正  側  ト  ル  ク リミット              */
        AxisRscI->IntAdV.IqRef = limit( AxisRscI->IntAdV.IqOfRef, swk1 ); /* <V224> 外  乱  ト  ル  ク 加  算  後 のq軸電流指令       */
        swk10 = AxisRscI->StsFlg.CtrlStsRW | TLIM;    /* TLIM flag set                    */
        AxisRscI->StsFlg.CtrlStsRW = cmove((AxisRscI->IntAdV.IqRef == swk1), swk10, AxisRscI->StsFlg.CtrlStsRW);
      }
      else
      {
        swk1 = limit( swk1, AxisRscI->IntAdV.TLimM ); /* 負  側  ト  ル  ク リミット              */
        AxisRscI->IntAdV.IqRef = limit( AxisRscI->IntAdV.IqOfRef, swk1 ); /* <V224> 外  乱  ト  ル  ク 加  算  後 のq軸電流指令       */
        swk10 = AxisRscI->IntAdV.IqRef + swk1;
        swk11 = AxisRscI->StsFlg.CtrlStsRW | TLIM;    /* TLIM flag set                    */
        AxisRscI->StsFlg.CtrlStsRW = cmove((swk10 == 0), swk11, AxisRscI->StsFlg.CtrlStsRW);    /* TLIM flag set
```

```
      */
    }
/*-----------------------------------------------------------------------------------------------*/
/*    TMP1 = limit( IntAdV.IqRef - IntAdV.IqInData , 2^15 - 1 )                                  */
/*-----------------------------------------------------------------------------------------------*/
    swk1 = sub_limitf(AxisRscI->IntAdV.IqRef, AxisRscI->IntAdV.IqInData); /* TMP1 <-- limit( TMP1 , 2^15 - 1 )        */

/*-----------------------------------------------------------------------------------------------*/
/*    TMP2 = limit( IntAdP.KqP * TMP1 / 2^9 , 2^15 - 1 )                                         */
/*-----------------------------------------------------------------------------------------------*/
    swk2 = mulshr_limitf(AxisRscI->IntAdP.KqP, swk1, 9);  /* TMP2 <-- limit( TMP2 , 2^15 - 1 )          */
/*-----------------------------------------------------------------------------------------------*/
/*    AcrV.IqIntgl(32) = (IntAdP.KqI * TMP1)<<3 + AcrV.IqIntgl(32)                               */
/*    IQIH = limit( IQIH , IntAdP.VqLim )                                                        */
/*-----------------------------------------------------------------------------------------------*/
    if( ( (AxisRscI->IntAdP.CtrlSw & INT_ST) == 0) || ( (AxisRscI->StsFlg.IntglFlg & 1) == 0 ) )
    {
      lwk6 = mul(AxisRscI->IntAdP.KqI, swk1); /* ACC <-- IntAdP.KqI * TMP1                        */
      lwk4 = (ULONG)AxisRscI->IntAdP.VqLim; /*                                                   */
      lwk4 = lwk4 << 16;  /*                                                                    */
      lwk6 = lwk6 << 3; /*                                                                      */
      AxisRscI->AcrV.IqIntgl.l = add_limitf(lwk6, AxisRscI->AcrV.IqIntgl.l);  /* AcrV.IqIntgl <-- limit( AcrV.IqIntgl , 2^32
      - 1 )         */
      if( LPX_ABS(AxisRscI->AcrV.IqIntgl.l) > LPX_ABS(lwk4) )
      {
        AxisRscI->StsFlg.CtrlStsRW = AxisRscI->StsFlg.CtrlStsRW | QLIM;   /* IMM3 <-- STAT | QLIM (imm_16)              */
        swk10 = AxisRscI->IntAdP.CtrlSw & ICLR;
        AxisRscI->AcrV.IqIntgl.l = cmove((swk10 != 0), (LONG)ZEROR, AxisRscI->AcrV.IqIntgl.l);
      }
    }
/*-----------------------------------------------------------------------------------------------*/
/*    VcmpV.VqOut = limit( TMP2 + IQIH +TMP3 , 2^15 - 1 )                                        */
/*-----------------------------------------------------------------------------------------------*/
    swk1 = add_limitf(AxisRscI->AcrV.IqIntgl.s[1], swk2); /* TMP1 <-- limit( TMP1 , 2^15 - 1 )        */
/*-----------------------------------------------------------------------------------------------*/
/*    filter : AcrV.VqFil = ( ( ( TMP1 - VQFH ) * IntAdP.Tfil ) << 2 ) + AcrV.VqFil              */
/*-----------------------------------------------------------------------------------------------*/
```

```
      swk1 = sub_limitf(swk1, AxisRscI->AcrV.VqFil.s[1]); /* TMP1 <-- limit( TMP1 , 2^15 - 1 )        */
      lwk0 = mul(AxisRscI->IntAdP.Tfil, swk1 ) << 2;  /*                                  */
      AxisRscI->AcrV.VqFil.l = add_limitf(AxisRscI->AcrV.VqFil.l, lwk0);


/*****************************************************************************************************/
/*                                                  */
/*    Voltage Compensation(電  圧  補償 )                                  */
/*                                                  */
/*****************************************************************************************************/
      if( (AxisRscI->IntAdP.CtrlSw & ISEL) != 0 )
      {
        swk1 = AxisRscI->WeakFV.IdOut;  /* TMP1 <-- reference ID                  */
        swk2 = AxisRscI->IntAdV.IqRef;  /*                                  */
      }
      else
      {
        swk1 = AxisRscI->IntAdV.IdInData; /* TMP1 <-- feedback ID                   */
        swk2 = AxisRscI->IntAdV.IqInData; /* TMP2 <-- feedback IQ                   */
      }
/*---------------------------------------------------------------------------------------------------*/
/*    TMP4(VcmpV.VdComp) = IntAdP.MotResist*TMP1/2^15 - VcmpV.LqC * TMP2 / 2^15                      */
/*---------------------------------------------------------------------------------------------------*/
      swk4 = mulshr(AxisRscI->VcmpV.LqC, swk2, 15 );    /* VcmpV.VdComp <-- ACC >> 15                    */
      swk0 = mulshr(AxisRscI->IntAdP.MotResist, swk1, 15 );
      swk4 = swk0 - swk4;
/*---------------------------------------------------------------------------------------------------*/
/*    TMP5(VcmpV.VqComp) = VcmpV.LdC * TMP1 / 2^15 + VcmpV.MagC + IntAdP.MotResist*TMP2/2^15                    */
/*---------------------------------------------------------------------------------------------------*/
      swk3 = mulshr(AxisRscI->VcmpV.LdC, swk1, 15 );  /* TMP3 <-- ACC >> 15                    */
      swk0 = mulshr(AxisRscI->IntAdP.MotResist, swk2, 15 );
      swk3 = swk3 + AxisRscI->VcmpV.MagC;
      swk5 = swk3 + swk0; /* VcmpV.VqComp <-- VcmpV.MagC + TMP3 + TMP0        */
/*---------------------------------------------------------------------------------------------------*/
/*    if(IntAdP.CtrlSw & DIDTSET) VcmpV.VdComp = TMP4 + KDD * (IntAdV.IdDataP - IntAdV.IdInData),
IntAdV.IdDataP=IntAdV.IdInData                    */
/*              VcmpV.VqComp = TMP5 + KQD * (IntAdV.IqDataP - IntAdV.IqRef), IntAdV.IqDataP=IntAdV.IqRef                */
```

```
/*-----------------------------------------------------------------------------------------------------*/
      if( (AxisRscI->IntAdP.CtrlSw & DIDTSEL) == 0 )
      {
        AxisRscI->VcmpV.VdComp = swk4;  /*                                 */
        AxisRscI->VcmpV.VqComp = swk5;  /*                                 */
      }
/*-----------------------------------------------------------------------------------------------------*/
/*    filter : I*FL =  ( ( ( TMP1 - I*FH ) * IntAdP.Tfil ) << 2 ) + I*FL                   */
/*-----------------------------------------------------------------------------------------------------*/
      else
      {
        swk1 = AxisRscI->WeakFV.IdOut;  /*                                 */
        swk1 = sub_limitf(swk1, AxisRscI->IntAdV.IdLfil.s[1]);  /*                           */
        lwk0 = mul(AxisRscI->IntAdP.Tfil, swk1) << 2; /*                          */
        AxisRscI->IntAdV.IdLfil.l = add_limitf(AxisRscI->IntAdV.IdLfil.l, lwk0);  /*                                */
/*-----------------------------------------------------------------------------------------------------*/
        swk1 = AxisRscI->IntAdV.IqRef;  /*                                 */
        swk1 = sub_limitf(swk1, AxisRscI->IntAdV.IqLfil.s[1]);  /*                           */
        lwk0 = mul(AxisRscI->IntAdP.Tfil, swk1 ) << 2;  /*                          */
        AxisRscI->IntAdV.IqLfil.l = add_limitf(AxisRscI->IntAdV.IqLfil.l, lwk0);  /*                                */
/* ----------------------------------------------------------------------------------------------- */
        swk2 = AxisRscI->IntAdV.IdLfil.s[1] - AxisRscI->IntAdV.IdDataP; /*                        */
        AxisRscI->IntAdV.IdDataP = AxisRscI->IntAdV.IdLfil.s[1];  /*                        */
        swk2 = mulshr_limitf(AxisRscI->IntAdP.L_dIdt, swk2, 9); /* limit( VDL , 2^15 - 1 )            */
        AxisRscI->VcmpV.VdComp = add_limitf(swk2, swk4);  /* VcmpV.VdComp <-- limit( VcmpV.VdOut , 2^15 - 1 )             */
/*-----------------------------------------------------------------------------------------------------*/
        swk2 = AxisRscI->IntAdV.IqLfil.s[1] - AxisRscI->IntAdV.IqDataP; /*                        */
        AxisRscI->IntAdV.IqDataP = AxisRscI->IntAdV.IqLfil.s[1];
        swk2 = mulshr_limitf(AxisRscI->IntAdP.L_dIdt, swk2, 9); /* limit( VQL , 2^15 - 1 )            */
        AxisRscI->VcmpV.VqComp = add_limitf(swk2, swk5);  /* VcmpV.VqComp <-- limit( VcmpV.VqOut , 2^15 - 1 )             */
      }
/*-----------------------------------------------------------------------------------------------------*/
/*    TMP1 = limit( VDFH + VcmpV.VdComp , 2^15 - 1 )                          */
/*    TMP2 = limit( VQFH + VcmpV.VqComp , 2^15 - 1 )                          */
/*-----------------------------------------------------------------------------------------------------*/
      swk1 = add_limitf(AxisRscI->AcrV.VdFil.s[1], AxisRscI->VcmpV.VdComp); /* VcmpV.VdOut <-- limit( VcmpV.VdOut , 2^15 - 1 )
      */
```

```
      swk2 = add_limitf(AxisRscI->AcrV.VqFil.s[1], AxisRscI->VcmpV.VqComp); /* VcmpV.VqOut <-- limit( VcmpV.VqOut , 2^15 - 1 )
      */

/*-------------------------------------------------------------------------------------------------*/
/*    TMP1 = limit( VcmpV.VdRef + TMP1 , 2^15 - 1 )                                                */
/*    TMP2 = limit( VcmpV.VqRef + TMP2 , 2^15 - 1 )                                                */
/*-------------------------------------------------------------------------------------------------*/
      swk1 = add_limitf(AxisRscI->VcmpV.VdRef, swk1); /* VcmpV.VdOut <-- limit( VcmpV.VdOut , 2^15 - 1 )            */
      swk2 = add_limitf(AxisRscI->VcmpV.VqRef, swk2); /* VcmpV.VqOut <-- limit( VcmpV.VqOut , 2^15 - 1 )            */

/*-------------------------------------------------------------------------------------------------*/
/*    VcmpV.VdOut = limit( IntAdP.Kvv * TMP1 / 2^13 , 2^15 - 1 )                                   */
/*    VcmpV.VqOut = limit( IntAdP.Kvv * TMP2 / 2^13 , 2^15 - 1 )                                   */
/*-------------------------------------------------------------------------------------------------*/
      AxisRscI->VcmpV.VdOut = mulshr_limitf(AxisRscI->IntAdP.Kvv, swk1, 13);  /* VcmpV.VdOut   <-- limit( TMP1 , 2^15 - 1 )
      */
      AxisRscI->VcmpV.VqOut = mulshr_limitf(AxisRscI->IntAdP.Kvv, swk2, 13);    /* VcmpV.VqOut   <-- limit( TMP2 , 2^15 - 1 )
      */
      AxisRscI->WeakFV.WfVdRef = AxisRscI->VcmpV.VdOut;   /* d軸  電  圧  指 令保存     <V531>             */
      AxisRscI->WeakFV.WfVqRef = AxisRscI->VcmpV.VqOut;   /* q軸  電  圧  指 令保存     <V531>             */

/*************************************************************************************************/
/*    電  圧  ベ  ク  ト  ル 補 正値計算     <V537> 新  弱  め  界  磁  制  御  以  外  は  こ  の処理 をジャンプする      */
/*************************************************************************************************/
      if( (AxisRscI->IntAdP.CtrlSw & V_FB2) != 0 )
      {
/*************************************************************************************************/
/*    Get modulation              <V531> 変  調  率  計  算を移動                 */
/*************************************************************************************************/
        lwk2 = mul(AxisRscI->VcmpV.VdOut, AxisRscI->VcmpV.VdOut);
        lwk2 = mac(AxisRscI->VcmpV.VqOut, AxisRscI->VcmpV.VqOut, lwk2);
        swk0 = MpSQRT( lwk2 );          /* TMP0 = √ (VcmpV.VdOut^2 + VcmpV.VqOut^2)                  */
        AxisRscI->IntAdV.V1 = swk0;   /* IntAdV.V1   = TMP0                                */
/*************************************************************************************************/
/*    飽  和  判断              <V531> IntAdV.V1 > 8192*127%(10403.8) -> 飽  和  状態         */
/*************************************************************************************************/
        AxisRscI->VcmpV.Vmax2 = 10403;  /* VcmpV.Vmax2 = 8192 * 1.27                    */
```

```
        AxisRscI->VcmpV.V12 = AxisRscI->IntAdV.V1;      /* VcmpV.V12 = √ (VcmpV.VdOut^2 + VcmpV.VqOut^2)                  */
        swk10 = AxisRscI->VcmpV.Vmax2 >> 1;   /* VcmpV.Vmax2 = 8192 * 1.27 / 2                  */
        swk11 = AxisRscI->IntAdV.V1 >> 1;   /* VcmpV.V12 = √ (VcmpV.VdOut^2 + VcmpV.VqOut^2) / 2              */
        AxisRscI->VcmpV.Vmax2 = cmove((AxisRscI->IntAdV.V1 < 0), swk10, AxisRscI->VcmpV.Vmax2);
        AxisRscI->VcmpV.V12   = cmove((AxisRscI->IntAdV.V1 < 0), swk11, AxisRscI->VcmpV.V12);
        if( AxisRscI->VcmpV.Vmax2 < AxisRscI->VcmpV.V12 )
        {
          AxisRscI->IntAdV.V1 = 10403;    /* IntAdV.V1 = IntAdP.Vmax( 8192 * 1.27 )              */
          AxisRscI->StsFlg.IntglFlg = AxisRscI->StsFlg.IntglFlg | 1;      /* 積 分 停 止 フ ラグセット                  */
//;*************************************************************************************************
//;*    電 圧 ベ ク ト ル 補 正値計算       <V531> VcmpV.VdOut',VcmpV.VqOut' = IntAdP.Vmax / IntAdV.V1 * VcmpV.VdOut, VcmpV.VqOut
<V537> 削 除     *
//;*************************************************************************************************
/*-------------------------------------------------------------------------------------------------*/
/*    電 圧 制 限 テ ー ブ ルアドレ ス取得                                    */
/*-------------------------------------------------------------------------------------------------*/
        lwk2 = mul(AxisRscI->VcmpV.V12, AxisRscI->VcmpV.V12); /* TMP3,2 = VcmpV.V12^2                  */
        lwk2 = lwk2 -  0x00400000;  /* TMP3,2 = IntAdV.V1^2 - 2^22                  */
        lwk2 = lwk2 >> 4; /* TMP3,2 = (VcmpV.V12^2 - 2^22) / 2^4              */
        swk0 = (USHORT)( lwk2 >> 16 );  /* TMP0 = (VcmpV.V12^2 - 2^22) / 2^4 / 2^16 = addr      */
        lwk2 = lwk2 & 0x0000ffff; /* TMP2 = { (VcmpV.V12^2 - 2^22) / 2^4 } & 0x0000ffff       */
/*-------------------------------------------------------------------------------------------------*/
/*    電 圧 制 限 ベ ク ト ル 直 線補 間用デー タ取得                          */
/*-------------------------------------------------------------------------------------------------*/
        lwk4 = 65536; /* TMP5,TMP4 = 65536                    */
        lwk6 = lwk4 - lwk2; /* TMP7,6 = 10000h - Table Index (Lo) -> (addr*2^16-low)  */
        IxTblVlmt16( swk8, swk0 );  /* TMP8 : テ ー ブ ル デ ー タ 読 み 出 し(読み出 し ア ドレスad dr)  *//* tanaka21,コ ン
        */
        lwk6 = (ULONG)swk8 * lwk6;  /* TMP6 = tblrv(addr)*(2^16-low)            */
        swk0 = swk0 + 1;    /* TMP0 = addr+1                  */
        IxTblVlmt16( swk8, swk0 );  /* TMP8 : テ ー ブ ル デ ー タ 読 み 出 し(読み出 し ア ドレスad dr+1)  *//*
        tanaka21,コ ン パ イ ラ 対応待ち    */
        lwk4 = (ULONG)swk8 * lwk2;  /* TMP4 = tblrv(addr+1)*low               */
        lwk0 = lwk6 + lwk4; /* TMP0 = tblrv(addr)*(2^16-low) + tblrv(addr+1)*low  */
/*-------------------------------------------------------------------------------------------------*/
/*    電 圧 電 圧 ベ ク ト ル補正値 計算                                      */
/*-------------------------------------------------------------------------------------------------*/
```

```
          swk8 = AxisRscI->VcmpV.Vmax2; /* TMP8 = VcmpV.Vmax2                          */
          lwk2 = mulshr((ULONG)swk8, lwk0, 28 );    /* TMP2 = MAC / 2^28                    */
          AxisRscI->VcmpV.VdOut = mulshr(swk2, AxisRscI->VcmpV.VdOut, 14 );     /* VcmpV.VdOut = IntAdP.Vmax / VcmpV.V12 *
          VcmpV.VdOut * 2^(13+13+16) / 2^(28+14)      */
          AxisRscI->VcmpV.VqOut = mulshr(swk2, AxisRscI->VcmpV.VqOut, 14 );     /* VcmpV.VqOut = IntAdP.Vmax / VcmpV.V12 *
          VcmpV.VqOut * 2^(13+13+16) / 2^(28+14)      */
        }
        else
        {
          AxisRscI->StsFlg.IntglFlg = AxisRscI->StsFlg.IntglFlg & 0xFFFE;   /* 積　分　停　止　フ　ラグクリア                    */
        }
      }
/*****************************************************************************************************/
/*                                                   */
/*    UVW transform : dq( 2phase ) to UVW( 3phase ) Transform                    */
/*                                                   */
/*****************************************************************************************************/
/*---------------------------------------------------------------------------------------------------*/
/*    VcmpV.VuOut = limit( SinTbl.CosT * VcmpV.VdOut / 2^14 - SinTbl.SinT * VcmpV.VqOut / 2^14 , 2^15 - 1 )               */
/*---------------------------------------------------------------------------------------------------*/
      swk4 = AxisRscI->IntAdP.Vmax; /*                                  */
      swk1 = mulshr(AxisRscI->SinTbl.CosT, AxisRscI->VcmpV.VdOut, 14 ); /* TMP1 <-- ACC >> 14                  */
      swk2 = mulshr(AxisRscI->SinTbl.SinT, AxisRscI->VcmpV.VqOut, 14 );   /* TMP2 <-- ACC >> 14                    */
      AxisRscI->VcmpV.VuOut = sub_limitf(swk1, swk2);     /* VcmpV.VuOut <-- limit( VcmpV.VuOut , 2^15 - 1 )               */
      AxisRscI->VcmpV.VuOut = IxLIMIT( AxisRscI->VcmpV.VuOut, swk4 ); /*                                    */
/*---------------------------------------------------------------------------------------------------*/
/*    VcmpV.VvOut = limit( SinTbl.CosT3 * VcmpV.VdOut / 2^14 - SinTbl.SinT3 * VcmpV.VqOut / 2^14 , 2^15 - 1 )               */
/*---------------------------------------------------------------------------------------------------*/
      swk1 = mulshr(AxisRscI->SinTbl.CosT3, AxisRscI->VcmpV.VdOut, 14 );  /* TMP1 <-- ACC >> 14                  */
      swk2 = mulshr(AxisRscI->SinTbl.SinT3, AxisRscI->VcmpV.VqOut, 14 );    /* TMP2 <-- ACC >> 14                    */
      AxisRscI->VcmpV.VvOut = sub_limitf(swk1, swk2);     /* VcmpV.VvOut <-- limit( VcmpV.VvOut , 2^15 - 1 )               */
      AxisRscI->VcmpV.VvOut = IxLIMIT( AxisRscI->VcmpV.VvOut, swk4 ); /*                                    */
/*---------------------------------------------------------------------------------------------------*/
/*    VcmpV.VwOut = limit( - VcmpV.VuOut - VcmpV.VvOut , 2^15 - 1 )                          */
/*---------------------------------------------------------------------------------------------------*/
      swk1 = (SHORT)ZEROR - AxisRscI->VcmpV.VuOut;  /* VcmpV.VwOut <-- - VcmpV.VuOut - VcmpV.VvOut                  */
      AxisRscI->VcmpV.VwOut = sub_limitf(swk1, AxisRscI->VcmpV.VvOut);      /* VcmpV.VwOut <-- limit( VcmpV.VwOut , 2^15 - 1 )
```

```
      */
      AxisRscI->VcmpV.VwOut = IxLIMIT( AxisRscI->VcmpV.VwOut, swk4 ); /*                                    */

#if 0 /* for debug 2013.07.17 tanaka21*/
/*****************************************************************************************************/
/*    新 弱 め 界 磁 制 御 判断処理    <V537> 新 弱 め 界 磁 の 場 合 変 調 率 計 算 ,   飽和判断処理を ジャンプする  */
/*****************************************************************************************************/
      if( (AxisRscI->IntAdP.CtrlSw & V_FB2) == 0 )
      {
/*****************************************************************************************************/
/*    Get modulation          <V531> 変 調 率 計 算 は  2 相  3 相変換前に  す る <V537> 復活      */
/*****************************************************************************************************/
        lwk2 = mul(AxisRscI->VcmpV.VdOut, AxisRscI->VcmpV.VdOut);
        lwk2 = mac(AxisRscI->VcmpV.VqOut, AxisRscI->VcmpV.VqOut, lwk2);
        swk0 = MpSQRT( lwk2 );
        if( (USHORT)swk0 > 0x7FFF )
        {
          swk0 = 0x7FFF;  /* √  の  計 算が3 2 7 6 7 を 超えた  ら  、32767にす る  。          ; <V350>  */
        }
        AxisRscI->IntAdV.V1 = swk0;
/*---------------------------------------------------------------------------------------------------*/
/*    飽  和  判断            <V531> <V537> 復  活                        */
/*---------------------------------------------------------------------------------------------------*/
        AxisRscI->StsFlg.IntglFlg = AxisRscI->StsFlg.IntglFlg & 0xFFFE;   /*                          */
        swk10 = AxisRscI->StsFlg.IntglFlg | 1;      /*                          */
        AxisRscI->StsFlg.IntglFlg = cmove((AxisRscI->IntAdV.V1 >= 9421), swk10, AxisRscI->StsFlg.IntglFlg);
      }
/*****************************************************************************************************/
/*    Over modulation type select                                  */
/*****************************************************************************************************/
      if( AxisRscI->IntAdP.Vmax >= 0x2000 )
      {
        if( (AxisRscI->IntAdP.CtrlSw & OVMSEL2) == 0 )
        {
          if( ( AxisRscI->IntAdV.V1 >= 0x2000 )&&( (AxisRscI->IntAdP.CtrlSw & OVMSEL1) != 0 ) )
          {
/*****************************************************************************************************/
```

```
/*    Over modulation1                                          */
/*******************************************************************************************************/
            IxSetCtblAdr( pCtbl, &(OVMODTBLG[0][0]) );  /* gain type                    */
            MpOVMMODK( &AxisRscI->IntAdP, &AxisRscI->IntAdV, pCtbl );
            AxisRscI->VcmpV.VuOut = mulshr_limitf(AxisRscI->VcmpV.VuOut, AxisRscI->IntAdP.Kmod, 13);
            AxisRscI->VcmpV.VvOut = mulshr_limitf(AxisRscI->VcmpV.VvOut, AxisRscI->IntAdP.Kmod, 13);
            AxisRscI->VcmpV.VwOut = mulshr_limitf(AxisRscI->VcmpV.VwOut, AxisRscI->IntAdP.Kmod, 13);
/*-------------------------------------------------------------------------------------------------------*/
/*    TMP1 = |VcmpV.VuOut|,      TMP2 = |VcmpV.VvOut|,      TMP3 = |VcmpV.VwOut|           */
/*    TMP4 = sign(VcmpV.VuOut), TMP5 = sign(VcmpV.VvOut), TMP6 = sign(VcmpV.VwOut)         */
/*-------------------------------------------------------------------------------------------------------*/
            swk0 = 1;
            swk4 = IxLIMIT( AxisRscI->VcmpV.VuOut, swk0 );
            swk1 = swk4 * AxisRscI->VcmpV.VuOut;
            swk5 = IxLIMIT( AxisRscI->VcmpV.VvOut, swk0 );
            swk2 = swk5 * AxisRscI->VcmpV.VvOut;
            swk6 = IxLIMIT( AxisRscI->VcmpV.VwOut, swk0 );
            swk3 = swk6 * AxisRscI->VcmpV.VwOut;
            if( swk1 >= swk2 )
            {
              if( swk1 >= swk3 )
              {
                swk1 = swk1 - 0x2000; /* TMP1 <-- |VcmpV.VuOut|-2000h                */
                IxLmtzImm16( swk1, 0x7fff );  /* zero limit                    */
                swk0 = swk4 * swk1;
              }
              else
              {
                swk3 = swk3 - 0x2000; /* TMP0 <-- |VcmpV.VwOut|-2000h                */
                IxLmtzImm16( swk3, 0x7fff );  /* zero limit                    */
                swk0 = swk6 * swk3;
              }
            }
            else
            {
              if( swk2 >= swk3 )
              {
```

```
            swk2 = swk2 - 0x2000; /* TMP0 <-- |VcmpV.VvOut|-2000h                  */
            IxLmtzImm16( swk2, 0x7fff );  /* zero limit                        */
            swk0 = swk5 * swk2;
          }
          else
          {
            swk3 = swk3 - 0x2000; /* TMP0 <-- |VcmpV.VwOut|-2000h                  */
            IxLmtzImm16( swk3, 0x7fff );  /* zero limit                        */
            swk0 = swk6 * swk3;
          }
        }
        AxisRscI->VcmpV.VuOut = sub_limitf(AxisRscI->VcmpV.VuOut, swk0);      /*                                  */
        AxisRscI->VcmpV.VvOut = sub_limitf(AxisRscI->VcmpV.VvOut, swk0);      /*                                  */
        AxisRscI->VcmpV.VwOut = sub_limitf(AxisRscI->VcmpV.VwOut, swk0);      /*                                  */
        AxisRscI->IntAdV.Vcent = swk0;
      }
    }
/*****************************************************************************************************/
/*    Over modulation2                                                       */
/*****************************************************************************************************/
    else
    {
      IxSetCtblAdr( pCtbl, &(OVMODTBL0[0][0]) );  /* ofset type                   */
      MpOVMMODK( &AxisRscI->IntAdP, &AxisRscI->IntAdV, pCtbl );
/*---------------------------------------------------------------------------------------------------*/
/*    MAX = TMP1, MIN = TMP2                                                 */
/*    OFS = (TMP1+TMP2)/2                                                    */
/*---------------------------------------------------------------------------------------------------*/
      if( AxisRscI->VcmpV.VuOut >= AxisRscI->VcmpV.VvOut )
      {
        swk1 = AxisRscI->VcmpV.VuOut;
        swk2 = AxisRscI->VcmpV.VvOut;
      }
      else
      {
        swk1 = AxisRscI->VcmpV.VvOut;
        swk2 = AxisRscI->VcmpV.VuOut;
```

```
        }
        if( swk1 < AxisRscI->VcmpV.VwOut )
        {
          swk1 = AxisRscI->VcmpV.VwOut;
        }
        else
        {
          if( AxisRscI->VcmpV.VwOut < swk2 )
          {
            swk2 = AxisRscI->VcmpV.VwOut;
          }
        }
        swk0 = add_limitf(swk2, swk1);  /*                              */
        swk0 = mulshr(swk0, ONE, 1);
/*-------------------------------------------------------------------------------------------------*/
        AxisRscI->VcmpV.VuOut = sub_limitf(AxisRscI->VcmpV.VuOut, swk0);      /*                    */
        AxisRscI->VcmpV.VvOut = sub_limitf(AxisRscI->VcmpV.VvOut, swk0);      /*                    */
        AxisRscI->VcmpV.VwOut = sub_limitf(AxisRscI->VcmpV.VwOut, swk0);      /*                    */
        AxisRscI->IntAdV.Vcent = swk0;
/*-------------------------------------------------------------------------------------------------*/
        swk0 = 1;
/*-------------------------------------------------------------------------------------------------*/
        swk0 = IxLIMIT( AxisRscI->VcmpV.VuOut, swk0 );  /* TMP1= -1/0/+1                  */
        swk1 = swk1 | 1;    /* TMP1 = -1/+1 -----sign(VcmpV.VuOut)                */
        swk2 = swk1 * AxisRscI->IntAdP.Kmod;
        AxisRscI->VcmpV.VuOut = add_limitf( swk2, AxisRscI->VcmpV.VuOut );      /*                    */
/*-------------------------------------------------------------------------------------------------*/
        swk1 = IxLIMIT( AxisRscI->VcmpV.VvOut, swk0 );
        swk1 = swk1 | 1;    /* sign(VcmpV.VvOut)                          */
        swk2 = swk1 * AxisRscI->IntAdP.Kmod;
        AxisRscI->VcmpV.VvOut = add_limitf( swk2, AxisRscI->VcmpV.VvOut );      /*                    */
/*-------------------------------------------------------------------------------------------------*/
        swk1 = IxLIMIT( AxisRscI->VcmpV.VwOut, swk0 );
        swk1 = swk1 | 1;    /* sign(VcmpV.VwOut)                          */
        swk2 = swk1 * AxisRscI->IntAdP.Kmod;
        AxisRscI->VcmpV.VwOut = add_limitf( swk2, AxisRscI->VcmpV.VwOut );      /*                    */
      }
```

```
    }
#endif  //#if 0 /* for debug 2013.07.17 tanaka21*/

/*****************************************************************************************************/
/*    On-Delay                                                 */
/*****************************************************************************************************/
/*---------------------------------------------------------------------------------------------------*/
/*  IU, IV reference calc                                      */
/*---------------------------------------------------------------------------------------------------*/
    swk1 = mulshr(AxisRscI->WeakFV.IdOut, AxisRscI->SinTbl.CosT, 14 );  /* TMP1 <-- ACC >> 14                  */
    swk2 = mulshr(AxisRscI->IntAdV.IqRef, AxisRscI->SinTbl.SinT, 14 );    /* TMP2 <-- ACC >> 14                  */
    AxisRscI->IntAdV.IuOut = swk1 - swk2; /* IntAdV.IuOut  <--  TMP1 - TMP2               */

    swk3 = mulshr(AxisRscI->WeakFV.IdOut, AxisRscI->SinTbl.CosT3, 14 ); /* TMP3 <-- ACC >> 14                  */
    swk4 = mulshr(AxisRscI->IntAdV.IqRef, AxisRscI->SinTbl.SinT3, 14 );   /* TMP4 <-- ACC >> 14                  */
    AxisRscI->IntAdV.IvOut = swk3 - swk4; /* IntAdV.IvOut  <--  TMP3 - TMP4               */
/*****************************************************************************************************/
//     if ( |IntAdV.IuInData| < IntAdP.OnDelayLvl ) TMP1 = IntAdV.IuOut /* Reference */
//     else                 TMP1 = IntAdV.IuInData
//     if ( |IntAdV.IvInData| < IntAdP.OnDelayLvl ) TMP2 = IntAdV.IvOut /* Reference */
//     else                 TMP2 = IntAdV.IvInData
//     if ( |IWD| < IntAdP.OnDelayLvl ) TMP2 = IWO  /* Reference */
//     else                 TMP2 = IWD
/*****************************************************************************************************/
    swk5 = AxisRscI->IntAdP.OnDelayLvl;
    if(LPX_ABS(AxisRscI->IntAdV.IuInData) > LPX_ABS(swk5))  //110530tanaka21作 業 メモ s w k 2 を 以 降 使 わ な い ため代入 は行な
    {
      swk1 = AxisRscI->IntAdV.IuInData; /* TMP1 <-- IntAdV.IuInData                     */
    }
    else
    {
      swk1 = AxisRscI->IntAdV.IuOut;  /* TMP1 <-- IntAdV.IuOut                       */
    }
    if( LPX_ABS(AxisRscI->IntAdV.IvInData) > LPX_ABS(swk5 ) ) //110530tanaka21作 業 メモ
    swk2を 以 降 使 わ な い た め 代入は行なわない
    {
      swk2 = AxisRscI->IntAdV.IvInData; /* TMP2 <-- IntAdV.IvInData                     */
```

```
      }
      else
      {
        swk2 = AxisRscI->IntAdV.IvOut;  /* TMP2 <-- IntAdV.IvOut                          */
      }
      swk3 = -AxisRscI->IntAdV.IuInData - AxisRscI->IntAdV.IvInData;  /* TMP3(IWD) <-- - TMP1 - TMP2          */
      if( LPX_ABS(swk3) <= LPX_ABS(swk5) )  //110530tanaka21作 業 メモ s w k 4 を 以 降 使 わ な い ため代入 は行なわない
      {
        swk3 = -AxisRscI->IntAdV.IuOut - AxisRscI->IntAdV.IvOut;  /* TMP3                      */
      }
      swk7 = 0x2000;  /* TMP7 <-- 2000h                    */
      swk5 = 1; /* TMP5 <-- 1                        */
/*-----------------------------------------------------------------------------------------------------*/
/*    if(IntAdP.OnDelaySlope != 0) trapezoid type else rectangle type                    */
/*-----------------------------------------------------------------------------------------------------*/
      if( AxisRscI->IntAdP.OnDelaySlope == 0 )
      {
/*-----------------------------------------------------------------------------------------------------*/
/*    TMP1(ONDVU) = sign(IU)*IntAdP.OnDelayComp                                          */
/*-----------------------------------------------------------------------------------------------------*/
        swk6 = IxLIMIT( swk1, swk5 ); /* TMP6 = -1/0/+1             */
        swk1 = AxisRscI->IntAdP.OnDelayComp * swk6;
/*-----------------------------------------------------------------------------------------------------*/
/*    TMP2(ONDVU) = sign(IV)*IntAdP.OnDelayComp                                          */
/*-----------------------------------------------------------------------------------------------------*/
        swk6 = IxLIMIT( swk2, swk5 );
        swk2 = AxisRscI->IntAdP.OnDelayComp * swk6;
/*-----------------------------------------------------------------------------------------------------*/
/*    TMP3(ONDVU) = sign(IW)*IntAdP.OnDelayComp                                          */
/*-----------------------------------------------------------------------------------------------------*/
        swk6 = IxLIMIT( swk3, swk5 );
        swk3 = AxisRscI->IntAdP.OnDelayComp * swk6;
      }
/*-----------------------------------------------------------------------------------------------------*/
/*    trapezoid type                                          */
/*-----------------------------------------------------------------------------------------------------*/
      else
```

```
      {
        swk0 = mulshr_limitf(AxisRscI->IntAdP.OnDelaySlope, swk1, 8 );    /* TMP0 <-- IU*IntAdP.OnDelaySlope>>8
        */
        swk0 = IxLIMIT( swk0, 8192 ); /* TMP0 = limit(TMP0,8192)                    */
        swk1 = mulshr(AxisRscI->IntAdP.OnDelayComp, swk0, 13 ); /* TMP1(ONDVU) = (IntAdP.OnDelayComp*TMP0)>>13                */
/*-------------------------------------------------------------------------------------------------*/
        swk0 = mulshr_limitf(AxisRscI->IntAdP.OnDelaySlope, swk2, 8); /* TMP0 = limit(TMP0,2^15-1)                */
        swk0 = IxLIMIT( swk0, 8192 ); /* TMP0 = limit(TMP0,8192)                    */
        swk2 = mulshr(AxisRscI->IntAdP.OnDelayComp, swk0, 13 ); /* TMP1(ONDVU) = (IntAdP.OnDelayComp*TMP0)>>13                */
/*-------------------------------------------------------------------------------------------------*/
        swk0 = mulshr_limitf(AxisRscI->IntAdP.OnDelaySlope, swk3, 8); /* TMP0 = limit(TMP0,2^15-1)                */
        swk0 = IxLIMIT( swk0, 8192 ); /* TMP0 = limit(TMP0,8192)                    */
        swk3 = mulshr(AxisRscI->IntAdP.OnDelayComp, swk0, 13 ); /* TMP1(ONDVU) = (IntAdP.OnDelayComp*TMP0)>>13                */
      }
/*-------------------------------------------------------------------------------------------------*/
/*************************************************************************************************/
/*    Voltage conversion to Carrier count range                                                  */
/*************************************************************************************************/
/*    -2000h..2000h  ---> 0h..4000h  ---> 0h..CRFRQ                                              */
/*************************************************************************************************/
      AxisRscI->VcmpV.VuOut = IxLIMIT( AxisRscI->VcmpV.VuOut, swk7 ); /* limit +-2000h                    */
      AxisRscI->VcmpV.VvOut = IxLIMIT( AxisRscI->VcmpV.VvOut, swk7 );
      AxisRscI->VcmpV.VwOut = IxLIMIT( AxisRscI->VcmpV.VwOut, swk7 );

/* for debug */
      swk4 = swk7 - AxisRscI->VcmpV.VuOut;
      swk4 = mulshr(swk4, AxisRscI->IntAdV.CrFreqW, 14 );
      swk5 = swk7 - AxisRscI->VcmpV.VvOut;
      swk5 = mulshr(swk5, AxisRscI->IntAdV.CrFreqW, 14 );
      swk6 = swk7 - AxisRscI->VcmpV.VwOut;
      swk6 = mulshr(swk6, AxisRscI->IntAdV.CrFreqW, 14 );


/*-------------------------------------------------------------------------------------------------*/
/*    Deat-time compensation (timer) : if(Vx == 0 || Vx == IntAdV.CrFreqW) No compensation          */
/*-------------------------------------------------------------------------------------------------*/
      if( ( swk4 != ZEROR ) && (swk4 != AxisRscI->IntAdV.CrFreqW ) )
```

```
      {
        swk4 = swk4 - swk1; /* VcmpV.VuOut <-- VcmpV.VuOut+ONDVU                */
        IxLmtzReg16( swk4, swk4, AxisRscI->IntAdV.CrFreqW );  /* VcmpV.VuOut <-- limitz( VcmpV.VuOut , IntAdV.CrFreqW )
        */
      }
      if( ( swk5 != ZEROR ) && (swk5 != AxisRscI->IntAdV.CrFreqW ) )
      {
        swk5 = swk5 - swk2; /* VcmpV.VvOut <-- VcmpV.VvOut+ONDVV              */
        IxLmtzReg16( swk5, swk5, AxisRscI->IntAdV.CrFreqW );  /* VcmpV.VvOut <-- limitz( VcmpV.VvOut , IntAdV.CrFreqW )
        */
      }
      if( ( swk6 != ZEROR ) && (swk6 != AxisRscI->IntAdV.CrFreqW ) )
      {
        swk6 = swk6 - swk3; /* VcmpV.VwOut <-- VcmpV.VwOut+ONDVW                */
        IxLmtzReg16( swk6, swk6, AxisRscI->IntAdV.CrFreqW );  /* VcmpV.VwOut <-- limitz( VcmpV.VwOut , IntAdV.CrFreqW )
        */
      }

      AxisRscI->PwmV.PwmCntT2 = swk6;
      AxisRscI->PwmV.PwmCntT1 = swk5;
      AxisRscI->PwmV.PwmCntT0 = swk4;

/*-------------------------------------------------------------------------------------------------------*/
/*    Output Voltage & status                                    */
/*-------------------------------------------------------------------------------------------------------*/
    }
//<2>#ifdef PREG_DEF
//    CTSTW = AxisRscI->StsFlg.CtrlStsRW; /* Status Set                        */
    AxisRscI->CtrlStsOut = AxisRscI->StsFlg.CtrlStsRW;  /* Status Set            *//* <Oka01> */
  }

#ifdef  MULTI_AXIS              /* 多  軸  処  理有効                    */
  {/* Axis1 start */
  /* Execute Current Loop Main Operation */
    AxisRscI = &AxisHdl[1];
//=================================================================
// 位  相  補  間処理     <V112>
```

```
//=================================================================
    swk10 = AxisRscI->PhaseV.PhaseH + AxisRscI->PhaseV.PhaseIp;
    AxisRscI->PhaseV.PhaseIpF = cmove((AxisRscI->PhaseV.PhaseIpF != 1), ONE, AxisRscI->PhaseV.PhaseIpF);
    AxisRscI->PhaseV.PhaseH   = cmove((AxisRscI->PhaseV.PhaseIpF != 1), AxisRscI->PhaseV.PhaseH, swk10);
//=================================================================
// PHASE_UPDATE処　理　<V112>
//=================================================================
/*-------------------------------------------------------------------------------------------------*/
/*    theta calculation                                            */
/*-------------------------------------------------------------------------------------------------*/
    swk0 = AxisRscI->PhaseV.PhaseH;
    swk0 = swk0 + 32;           /* TMP3 <-- PhaseV.PhaseH + 2^5 */
    swk1 = PI23;
    swk2 = swk1 + swk0; /* TMP4 <-- PhaseV.PhaseH + 2PI/3 */
    swk3 = swk0 - swk1; /* TMP5 <-- PhaseV.PhaseH - 2PI/3 */
/*-------------------------------------------------------------------------------------------------*/
/*    table read and get iu,iv by Id,Iq reference                  */
/*-------------------------------------------------------------------------------------------------*/
    swk1 = swk0 >> 6;       /* TMP1 <-- TMP3 >> 6 */
    IxTblSin16( AxisRscI->SinTbl.SinT, swk1 );    /* SinTbl.SinT <-- stable[ TMP1 ] *//* tanaka21,要　コ　メ　ン　ト解除 */
    swk0 = swk0 + PI2;        /* TMP3 <-- TMP3 + PI/2 */
    swk1 = swk0 >> 6;       /* TMP1 <-- TMP3 >> 6 */
    IxTblSin16( AxisRscI->SinTbl.CosT, swk1 );    /* SinTbl.CosT <-- stable[ TMP1 ] *//* tanaka21,要　コ　メ　ン　ト解除 */

    swk1 = swk3 >> 6;       /* TMP1 <-- TMP5 >> 6 */
    IxTblSin16( AxisRscI->SinTbl.SinT3, swk1 );   /* SinTbl.SinT3 <-- stable[ TMP1 ] *//* tanaka21,要　コ　メ　ン　ト解除　*/
    swk3 = swk3 + PI2;        /* TMP5 <-- TMP5 + PI/2 */
    swk1 = swk3 >> 6;       /* TMP1 <-- TMP5 >> 6 */
    IxTblSin16( AxisRscI->SinTbl.CosT3, swk1 );   /* SinTbl.CosT3 <-- stable[ TMP1 ] *//* tanaka21,要　コ　メ　ン　ト解除　*/

    swk1 = swk2 >> 6;       /* TMP1 <-- TMP4 >> 6 */
    IxTblSin16( AxisRscI->SinTbl.SinT2, swk1 );   /* SinTbl.SinT2 <-- stable[ TMP1 ] *//* tanaka21,要　コ　メ　ン　ト解除　*/
    swk2 = swk2 + PI2;        /* TMP4 <-- TMP4 + PI/2 */
    swk1 = swk2 >> 6;       /* TMP1 <-- TMP4 >> 6 */
    IxTblSin16( AxisRscI->SinTbl.CosT2, swk1 );   /* SinTbl.CosT2 <-- stable[ TMP1 ] *//* tanaka21,要　コ　メ　ン　ト解除　*/

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

```
/*    dq-trans(UVW to DQ)                                          */
/*-----------------------------------------------------------------------------------------------*/
/*    ID = IntAdP.Kc * ( (SinTbl.CosT-SinTbl.CosT2)*IntAdV.IuInData/2^14 + (SinTbl.CosT3-SinTbl.CosT2)*IntAdV.IvInData/2^14 )
/2^9              */
/*    IQ = IntAdP.Kc * ( (SinTbl.SinT2-SinTbl.SinT)*IntAdV.IuInData/2^14 + (SinTbl.SinT2-SinTbl.SinT3)*IntAdV.IvInData/2^14 )
/2^9              */
/*-----------------------------------------------------------------------------------------------*/
    /* TMP1 <-- cos(th) - cos(th-2pi/3) */
    swk1 = AxisRscI->SinTbl.CosT - AxisRscI->SinTbl.CosT2;
    /* ACC <-- TMP1 * iu */
    swk2 = mulshr(swk1, AxisRscI->IntAdV.IuInData, 14 );
    /* TMP1 <-- cos(th-2pi/3)-cos(th+2pi/3) */
    swk1 = AxisRscI->SinTbl.CosT3 - AxisRscI->SinTbl.CosT2;
    /* ACC <-- TMP1 * iv */
    swk1 = mulshr(swk1, AxisRscI->IntAdV.IvInData, 14 );
    /* TMP2 <-- TMP2 + TMP1 */
    swk2 = swk1 + swk2;
    /* ACC <-- IntAdP.Kc * TMP2 */
    AxisRscI->IntAdV.IdInData = mulshr(AxisRscI->IntAdP.Kc, swk2, 9 );
/*-----------------------------------------------------------------------------------------------*/
    swk1 = AxisRscI->SinTbl.SinT2 - AxisRscI->SinTbl.SinT;                        /* TMP1 <-- sin(th+2pi/3) - sin(th)
    */
    swk2 = mulshr(swk1, AxisRscI->IntAdV.IuInData, 14 );  /* ACC <-- TMP1 * iu                    */
    swk1 = AxisRscI->SinTbl.SinT2 - AxisRscI->SinTbl.SinT3;                       /* TMP1 <-- sin(th+2pi/3)-sin(th-2pi/3)
    */
    swk1 = mulshr(swk1, AxisRscI->IntAdV.IvInData, 14 );  /* ACC <-- TMP1 * iv                    */
    swk2 = swk1 + swk2;                          /* TMP2 <-- TMP2 + TMP1            */
    AxisRscI->IntAdV.IqInData = mulshr(AxisRscI->IntAdP.Kc, swk2, 9 );    /* ACC <-- IntAdP.Kc * TMP2                    */
/*-----------------------------------------------------------------------------------------------*/
/*    Current Observer  <V038>                                     */
/*-----------------------------------------------------------------------------------------------*/
//=============================================
//  電  流  オ  ブ  ザ  ー バ スイッチ
//=============================================
    if( AxisRscI->IntAdP.CtrlSw & OBSSEL )
    {
//=============================================
```

```
//  ダ  ン  ピ  ン  グ  ゲ イ ンの設定  <V076>
//=============================================
//<2>     AxisRscI->DobsV.DmpGain = 2;
//=============================================
//  q軸  電  流  の  飽  和 チェック  <V076>
//=============================================
      if( AxisRscI->IntAdV.IqInData >= 0 )
      { /* 0以  上  の とき */
        /* TMP3 = IntAdV.IqInData */
        swk2 = AxisRscI->IntAdV.IqInData;
      }
      else          /* 負  の  とき       */
      {
        swk2 = ~AxisRscI->IntAdV.IqInData;  /* TMP3 = ~IntAdV.IqInData;
        *///110530tanaka21作  業  メ モ 、  - 1 掛  け  る  の と ど っ  ちが速い？
        swk2 = swk2 + 1;  /* TMP3 = TMP3 + 1                        */
      }
      if( swk2 <= 14250 )
      {
        swk3 = ZERO;    /* TMP4 = 0 ( OverFlowCheck = OK )              */
      }
      else
      {
        swk3 = ONE;   /* TMP4 = 1 ( OverFlowCheck = NG )              */
      }
//=====================================
//  d軸  オ  ブ  ザ ーバ 部
//=====================================
      swk0 = mulshr(AxisRscI->DobsP.TsPerL, AxisRscI->VcmpV.VdOut, 15 );    /* TMP0 <-- ACC >> 15   ( TMP0 = Ts/L * Vd_out >>
      15 )  */
      swk2 = AxisRscI->IntAdV.IdInData; /* TMP3 <-- IntAdV.IdInData         <V076>            */
      swk2 = limit(swk2, 15000);
      swk1 = swk2 - AxisRscI->DobsV.IdObsOut; /*                 <V076>      */
      swk1 = mulshr(AxisRscI->DobsP.Gobs, swk1, 16 ); /* ACC  <-- TMP2*DobsP.Gobs   ( TMP2 = g * ( Id - Id_obs ) )  */
      swk0 = swk1 + swk0; /* TMP0 <-- TMP0 + TMP2   ( TMP0 = ( g*(Id-Id_obs)>>16 ) + (Ts/L*Vd_out>>15) )  */
      swk1 = mulshr(AxisRscI->DobsP.RLTs, AxisRscI->DobsV.IdObsOut, 12 ); /* TMP2 <-- DobsV.IqObsOut      ( TMP2 = Id_obs )
      */
```

```
      AxisRscI->DobsV.IdObsOut = add_limitf(swk1, swk0);  /* DobsV.IdObsOut <-- limit( DobsV.IdObsOut, 2^15-1 )           */
//====================================
//  d軸  フ  ィ  ルタ部
//====================================
//------------------------------------
//  error obs
//------------------------------------
      swk0 = AxisRscI->IntAdV.IdInData - AxisRscI->DobsV.IdObsOut;  /*                              */
//------------------------------------
//  low pass filter
//------------------------------------
      swk0 = sub_limitf(swk0, AxisRscI->DobsV.LpfIld.s[1]);
      lwk2 = mul(AxisRscI->DobsP.FilObsGain, swk0 ) << 2; /*                                */
      AxisRscI->DobsV.LpfIld.l = add_limitf(lwk2, AxisRscI->DobsV.LpfIld.l);
//------------------------------------
//  high pass filter
//------------------------------------
      swk0 = sub_limitf(AxisRscI->DobsV.LpfIld.s[1], AxisRscI->DobsV.HpfIld.s[1]);
      lwk2 = mul(AxisRscI->DobsP.FilObsGain, swk0 ) << 2; /*                                */

      AxisRscI->DobsV.HpfIld.l = add_limitf(lwk2, AxisRscI->DobsV.HpfIld.l);  /*                                */
      AxisRscI->DobsV.IdObsFreq = AxisRscI->DobsV.LpfIld.s[1] - AxisRscI->DobsV.HpfIld.s[1];  /*                    */
//------------------------------------
//  IntAdV.IdInData = IntAdV.IdInData - DobsV.IdObsFreq
//------------------------------------
      AxisRscI->DobsV.IdObsFreq = AxisRscI->DobsV.IdObsFreq * 2;    /* ACC  <-- DobsV.IdObsFreq * DobsV.DmpGain
      */
      AxisRscI->DobsV.IdObsFreq = cmove((swk3 != 0), ZERO, AxisRscI->DobsV.IdObsFreq);
      AxisRscI->IntAdV.IdInData = AxisRscI->IntAdV.IdInData - AxisRscI->DobsV.IdObsFreq;  /*                              */
//====================================
//  q軸  オ  ブ  ザ ーバ 部
//====================================
      swk0 = mulshr(AxisRscI->DobsP.TsPerL, AxisRscI->VcmpV.VqOut, 15 );  /* ACC  <-- TMP0*Ts/L    ( TMP0 = Ts/L * Vq_out)
      */
      swk2 = AxisRscI->IntAdV.IqInData;                                  /* TMP3 <-- IntAdV.IqInData       <V076>
      */
      swk2 = limit(swk2, 15000);                        /* TMP3 <-- Limit(15000)  <V076>           */
```

```
      swk1 = swk2 - AxisRscI->DobsV.IqObsOut;                          /*              <V076>        */
      swk1 = mulshr(AxisRscI->DobsP.Gobs, swk1, 16 );   /* TMP2 <-- ACC >> 16   ( TMP2 = g * ( Iq - Iq_obs ) >> 16 )  */
      swk0 = swk1 + swk0;                      /* TMP0 <-- TMP0 + TMP2   ( TMP0 = ( g*(Iq-Iq_obs)>>16 ) + (Ts/L*Vq_out>>15) )
      */
      swk1 = mulshr(AxisRscI->DobsP.RLTs, AxisRscI->DobsV.IqObsOut, 12 );   /* TMP2 <-- ACC >> 12   ( TMP2 = (1-R*Ts/L)*Iq_obs
      >> 12 ) */
      AxisRscI->DobsV.IqObsOut = add_limitf(swk1, swk0);                          /* DobsV.IqObsOut <-- limit( DobsV.IqObsOut,
      2^15-1 )            */
//====================================
//  q軸  フ  ィ  ルタ部
//====================================
//------------------------------------
//  error obs
//------------------------------------
      swk0 = AxisRscI->IntAdV.IqInData - AxisRscI->DobsV.IqObsOut;  /*                          */
//------------------------------------
//  low pass filter
//------------------------------------
      swk0 = sub_limitf(swk0, AxisRscI->DobsV.LpfIlq.s[1]); /*                            */
      lwk2 = mul(AxisRscI->DobsP.FilObsGain, swk0) << 2;  /*                           */
      AxisRscI->DobsV.LpfIlq.l = add_limitf(lwk2, AxisRscI->DobsV.LpfIlq.l);  /*                              */
//------------------------------------
//  high pass filter
//------------------------------------
      swk0 = sub_limitf(AxisRscI->DobsV.LpfIlq.s[1], AxisRscI->DobsV.HpfIlq.s[1]);  /*                              */
      lwk2 = mul(AxisRscI->DobsP.FilObsGain, swk0) << 2;  /*                           */
      AxisRscI->DobsV.HpfIlq.l = add_limitf(lwk2, AxisRscI->DobsV.HpfIlq.l);  /*                              */
      AxisRscI->DobsV.IqObsFreq = AxisRscI->DobsV.LpfIlq.s[1] - AxisRscI->DobsV.HpfIlq.s[1];  /*                    */
//------------------------------------
//  IntAdV.IqInData = IntAdV.IqInData - DobsV.IqObsFreq
//------------------------------------
      AxisRscI->DobsV.IqObsFreq = AxisRscI->DobsV.IqObsFreq * 2;    /* ACC  <-- DobsV.IqObsFreq * DobsV.DmpGain
      */
      AxisRscI->DobsV.IqObsFreq = cmove((swk3 != 0), ZERO, AxisRscI->DobsV.IqObsFreq);
      AxisRscI->IntAdV.IqInData = AxisRscI->IntAdV.IqInData - AxisRscI->DobsV.IqObsFreq;  /*                          */
   }
/*-------------------------------------------------------------------------------------------*///110526tanaka21,BBチ  ェ
```

```
ッ　ク　処　理　、　処　理　順　をいろいろ変更。
/*    Base Block Check                                        *///if-else if-elseの　形　で　書　き　換　え　。　正　し　く動　作するか要確
/*--------------------------------------------------------------------------------------------------*/
    if( AxisRscI->AdStop.ADRst != 0 )
    {
      AxisRscI->AdStop.ADRst = 0;
      swk6 = AxisRscI->IntAdV.CrFreqW >> 1;
      AxisRscI->PwmV.PwmCntT2 = swk6;
      AxisRscI->PwmV.PwmCntT1 = swk6;
      AxisRscI->PwmV.PwmCntT0 = swk6;
/*--------------------------------------------------------------------------------------------------*/
    }
    /* 2012.12.20 Y.Oka 誤　り　修正 */
    else if( (AxisRscI->StsFlg.CtrlStsRW & BB) != 0 )
    {
/*--------------------------------------------------------------------------------------------------*/
      swk6 = AxisRscI->IntAdV.CrFreqW >> 1;
      AxisRscI->PwmV.PwmCntT2 = swk6;
      AxisRscI->PwmV.PwmCntT1 = swk6;
      AxisRscI->PwmV.PwmCntT0 = swk6;
/*--------------------------------------------------------------------------------------------------*/
    }
    else
    {
/****************************************************************************************************/
/*                                                  */
/*    弱　め　界 磁 用　Ｉｄ指 令計算処 理　　<V214>                        */
/*                                                  */
/****************************************************************************************************/
/*--------------------------------------------------------------------------------------------------*/
/*    弱　め　界　磁　方式選 択                                    */
/*--------------------------------------------------------------------------------------------------*/
      if( AxisRscI->IntAdP.CtrlSw & V_FB )
      {
/*--------------------------------------------------------------------------------------------------*/
/*    差　分　電　圧 作成                                    */
/*      Vq*と　基　準 電 圧(√(IntAdP.Vmax^2-V ｄ ^ ２ ) ) を　比　較　し 、差 分電圧を作る。                */
```

```
/*-------------------------------------------------------------------------------------------------*/
/*-------------------------------------------------------------------------------------------------*/
//    Vqmax = √ ( VmaxX^2 - Vd^2 )                                  *
/*-------------------------------------------------------------------------------------------------*/
        lwk2 = AxisRscI->WeakFV.WfV1Max * AxisRscI->WeakFV.WfV1Max; /* IntAdP.Vmax^2                  */
        lwk4 = AxisRscI->WeakFV.WfVdRef * AxisRscI->WeakFV.WfVdRef; /* Vd^2          ; 削  除 <V309>  復  活<V531>  */
        lwk2 = sub_limitf(lwk2, lwk4);
        lwk2 = limitz( lwk2, LPX_REG32_MAX ); /* if (IntAdP.Vmax^2 - Vd^2)< 0, then (IntAdP.Vmax^2 - Vd^2) = 0  */
        swk0 = MpSQRT( lwk2 );          /* √ ( IntAdP.Vmax^2 - Vd^2 )                  */
        if( swk0 > 0x7FFF )
        {
          swk0 = 0x7FFF;  /*                                */
        }
        AxisRscI->WeakFV.WfVqMax = swk0;  /* Vqmax = √ ( IntAdP.Vmax^2 - Vd^2 )            */
/*-------------------------------------------------------------------------------------------------*/
//    TMP0 = Vqmax - Vq                                  *
/*-------------------------------------------------------------------------------------------------*/
        swk1 = AxisRscI->WeakFV.WfVqRef;
        if( swk1 < 0 )
        {
          swk1 = (SHORT)ZEROR - swk1; /* TMP1 = |Vq|                */
        }
        swk0 = sub_limitf(AxisRscI->WeakFV.WfVqMax, swk1);
/*-------------------------------------------------------------------------------------------------*/
/*    比  例  項 計 算                                          */
/*-------------------------------------------------------------------------------------------------*/
        lwk1 = (LONG)swk0;  /* TMP1,0 = 符  号  拡張(TMP0)                */
        swk2 = (SHORT)mulshr( lwk1 ,AxisRscI->WeakFV.WfKpV.l, 32 );
        if( swk2 > (SHORT)0x0080 )
        {
          swk2 = LPX_REG16_MAX; /* 正  の  最 大値                        */
        }
        else if( swk2 < (SHORT)0xFF80 )
        {
          swk2 = LPX_REG16_MIN; /* 負  の  最 大値                        */
        }
        else
```

```
        {
          lwk2 = mulshr16( lwk1, AxisRscI->WeakFV.WfKpV.l);
          swk2 = mulshr( lwk2, (LONG)256, 16 );
        }
/*-------------------------------------------------------------------------------------------------*/
/*    積　分　項 計 算                                          */
/*-------------------------------------------------------------------------------------------------*/
        lwk4 = lwk1 * AxisRscI->WeakFV.WfKiV.l; /* Δ Vq * Kiv                    */
        lwk6 = mulshr( lwk1, AxisRscI->WeakFV.WfKiV.l, 32 );  /* Δ Vq * Kiv                    */
        if( (SHORT)lwk6 > 0x08 )
        {
          lwk4 = LPX_REG32_MAX; /* 正　の　最 大値                    */
        }
        else if( (USHORT)lwk6 > 0xFFF8 )
        {
          lwk4 = LPX_REG32_MIN; /* 負　の　最 大値                    */
        }
        else
        {
          lwk4 = lwk4 >> 4; /*                              */
          lwk4 = lwk4 & 0x0fffffff; /*                              */
          lwk6 = lwk6 << 28;  /*                              */
          lwk4 = lwk6 | lwk4; /* TMP5,4 = Δ Vq * Kiv (* 2^16)         */
        }
        AxisRscI->WeakFV.WfIntgl.l = add_limitf(lwk4, AxisRscI->WeakFV.WfIntgl.l);  /*                    */
        lwk6 = (ULONG)AxisRscI->WeakFV.WfIntegLim << 16;  /* TMP9,8 = WeakFV.WfIntegLim * 2^16          */
        AxisRscI->WeakFV.WfIntgl.l = limit( AxisRscI->WeakFV.WfIntgl.l, lwk6 ); /* WFINTEGH = Δ Vq * Kiv (* 2^16 / 2^16)  */
/*-------------------------------------------------------------------------------------------------*/
/*    比　例 項　+　積分 項                                      */
/*-------------------------------------------------------------------------------------------------*/
        swk4 = add_limitf(AxisRscI->WeakFV.WfIntgl.s[1], swk2);
        swk4 = limit( swk4, AxisRscI->WeakFV.WfIdRefLim );  /* IdrefLimで　リ　ミ ット          */
/*-------------------------------------------------------------------------------------------------*/
/*    Idref > 0 な　ら　ば、Idref =　0,積分 =　0                    */
/*      Idref(d軸　電　流 指　令　) が　正　に　な　る　こ　と　は　無　い 。正になった 場合は０にする。          */
/*-------------------------------------------------------------------------------------------------*/
        AxisRscI->WeakFV.IdOut = swk4;
```

```
        swk10 = AxisRscI->WeakFV.IdOut;
        AxisRscI->WeakFV.IdOut = cmove((swk10 > 0), ZERO, AxisRscI->WeakFV.IdOut);
        AxisRscI->WeakFV.WfIntgl.l = cmove((swk10 > 0), (LONG)ZEROR, AxisRscI->WeakFV.WfIntgl.l);
      }

/*****************************************************************************************************/
/*                                                   */
/*    ACRd(d軸  電  流 制 御)                                  */
/*                                                   */
/*****************************************************************************************************/
/*---------------------------------------------------------------------------------------------------*/
/*    TMP1 = limit( WeakFV.IdOut - IntAdV.IdInData , 2^15 - 1)                                       */
/*---------------------------------------------------------------------------------------------------*/
      swk1 = sub_limitf(AxisRscI->WeakFV.IdOut, AxisRscI->IntAdV.IdInData); /* TMP1 <-- limit( TMP1 , 2^15 - 1 )        */
/*---------------------------------------------------------------------------------------------------*/
/*    TMP2 = limit( IntAdP.KdP * TMP1 / 2^9 , 2^15 - 1 )                                             */
/*---------------------------------------------------------------------------------------------------*/
      swk2 = mulshr_limitf(AxisRscI->IntAdP.KdP, swk1, 9);  /* ACC <-- IntAdP.KdP * TMP1                    */
/*---------------------------------------------------------------------------------------------------*/
/*    IdIntgl(32) = (IntAdP.KdI * TMP1)<<3 + IdIntgl(32)                                             */
/*    IDIH = limit( IDIH , IntAdP.VdLim )                                                            */
/*---------------------------------------------------------------------------------------------------*/
      lwk4 = ((ULONG)AxisRscI->IntAdP.VdLim) << 16; /*                                               */
      lwk6 = mul(AxisRscI->IntAdP.KdI, swk1 ) << 3; /*                                               */
      AxisRscI->AcrV.IdIntgl.l = add_limitf(lwk6, AxisRscI->AcrV.IdIntgl.l);  /* AcrV.IdIntgl <-- limit( AcrV.IdIntgl , 2^31 -
      1 )          */
      if( LPX_ABS(AxisRscI->AcrV.IdIntgl.l) > LPX_ABS(lwk4) )
      {
        AxisRscI->StsFlg.CtrlStsRW = AxisRscI->StsFlg.CtrlStsRW | DLIM;   /*                          */
        swk0 = AxisRscI->IntAdP.CtrlSw;
        AxisRscI->AcrV.IdIntgl.l = cmove(((AxisRscI->IntAdP.CtrlSw & ICLR) != 0), (LONG)ZEROR, AxisRscI->AcrV.IdIntgl.l);
      }
/*---------------------------------------------------------------------------------------------------*/
/*    VcmpV.VdOut = limit( TMP2 + IDIH +TMP3, 2^15 - 1 )                                             */
/*---------------------------------------------------------------------------------------------------*/
      swk1 = add_limitf(AxisRscI->AcrV.IdIntgl.s[1], swk2); /* TMP1 <-- limit( TMP1 , 2^15 - 1 )        */
/*---------------------------------------------------------------------------------------------------*/
```

```
/*    filter : AcrV.VdFil =  ( ( ( TMP1 - VDFH ) * IntAdP.Tfil ) << 2 ) + AcrV.VdFil                  */
/*-------------------------------------------------------------------------------------------------*/
      swk1 = sub_limitf(swk1, AxisRscI->AcrV.VdFil.s[1]); /* TMP1 <-- limit( TMP1 , 2^15 - 1 )        */
      lwk0 = mul(AxisRscI->IntAdP.Tfil, swk1) << 2; /*                                  */
      AxisRscI->AcrV.VdFil.l = add_limitf(AxisRscI->AcrV.VdFil.l, lwk0);  /*                          */

/*************************************************************************************************/
/*                                                 */
/*    ACRq(q軸  電  流 制 御)                                    */
/*                                                 */
/*************************************************************************************************/
/*-------------------------------------------------------------------------------------------------*/
/*    Low Pass Filter                                          */
/*-------------------------------------------------------------------------------------------------*/
/*    IntAdP.TLpf2 : Time-constant                                  */
/*    IntAdV.IqOut2Lpf : Output(32 bit) .. IQOF: High 16 bit                        */
/*    WeakFV.IqOut   : Input                                    */
/*-------------------------------------------------------------------------------------------------*/
/*    IQOF(32) =  ( ( ( WeakFV.IqOut - IQOF(16) ) * IntAdP.TLpf2 ) << 2 ) + IntAdV.IqOut2Lpf(32)          */
/*-------------------------------------------------------------------------------------------------*/
      if( (AxisRscI->IntAdP.CtrlSw & LPFCDSABL) != 0 )
      {
        AxisRscI->IntAdV.IqOut2Lpf.s[1] = AxisRscI->WeakFV.IqOut; /* disable LPF                    */
      }
/*-------------------------------------------------------------------------------------------------*/
      else
      {
        swk0 = sub_limitf(AxisRscI->WeakFV.IqOut, AxisRscI->IntAdV.IqOut2Lpf.s[1]); /* TMP0 <-- limit( TMP0, 2^15 - 1 )
        */
        lwk2 = mul(AxisRscI->IntAdP.TLpf2, swk0 ) << 2;
        AxisRscI->IntAdV.IqOut2Lpf.l = add_limitf(AxisRscI->IntAdV.IqOut2Lpf.l, lwk2);
      }
/*-------------------------------------------------------------------------------------------------*/
      AxisRscI->IntAdV.IqMonFil = AxisRscI->IntAdV.IqOut2Lpf.s[1];  /* IntAdV.IqMonFil:フ  ィ  ル  タ 後  の q 軸  電流 (モ  ニ タ用) <V224>
      */
      AxisRscI->IntAdV.IqOfRef = add_limitf(AxisRscI->IntAdV.IqOut2Lpf.s[1], AxisRscI->IntAdV.IqDist);  /* IntAdV.IqOfRef <--
      limit( IntAdV.IqOfRef , 2^15 - 1 )  <V224>    */
```

```
/*-------------------------------------------------------------------------------------------------*/
/*    Torque Limit:     <V214>                                                                     */
/*      電 圧 フ ィ ー ド バ ッ ク 弱 め 界 磁 制 御 で d 軸 電 流 指 令 が作られるので、q軸電流指令は以下の式で    */
/*      求 め た 値 と ト ル ク リ ミ ッ ト 設 定 値 の いずれか小さい方で リミッ トする。          */
/*        Iq*リ ミ ッ ト値  = √(Imax^2-Id* ^2)                                                    */
/*-------------------------------------------------------------------------------------------------*/
/*    Id*に よ るTorque Lim it値       ;                                                          */
/*-------------------------------------------------------------------------------------------------*/
      lwk2 = 0x0d693a40;  /* 15000^2                  */
      swk0 = AxisRscI->IntAdP.CtrlSw;
      swk1 = V_FB | V_FB2;
      swk0 = swk0 & swk1; /* TMP0の bit11,bit13 以 外 を マ スクする     */
      if( swk0 != V_FB )
      {
        lwk4 = mul(AxisRscI->WeakFV.IdOut, AxisRscI->WeakFV.IdOut);     /* Idref^2              ; 削 除<V309 > 復活<V531>  */
      }
      else
      {
        lwk4 = mul(AxisRscI->WeakFV.WfIdRefLim, AxisRscI->WeakFV.WfIdRefLim); /* IdrefLim^2           ; <V309>    */
      }
      lwk2 = lwk2 - lwk4; /* Imax^2 - Id^2                */
      swk0 = MpSQRT( lwk2 );          /*                              */
      swk1 = swk0;  /* TMP0 = √ ( Imax^2 - Id^2 )             */

/*-------------------------------------------------------------------------------------------------*/
/*    Torque Limit                                          */
/*-------------------------------------------------------------------------------------------------*/
      if( AxisRscI->IntAdV.IqOfRef >= 0 )
      {
        swk1 = limit( swk1, AxisRscI->IntAdV.TLimP ); /* 正 側 ト ル ク リミット              */
        AxisRscI->IntAdV.IqRef = limit( AxisRscI->IntAdV.IqOfRef, swk1 ); /* <V224> 外 乱 ト ル ク 加 算 後 のq軸電流指令       */
        swk10 = AxisRscI->StsFlg.CtrlStsRW | TLIM;    /* TLIM flag set                    */
        AxisRscI->StsFlg.CtrlStsRW = cmove((AxisRscI->IntAdV.IqRef == swk1), swk10, AxisRscI->StsFlg.CtrlStsRW);
      }
      else
      {
        swk1 = limit( swk1, AxisRscI->IntAdV.TLimM ); /* 負 側 ト ル ク リミット              */
```

```
        AxisRscI->IntAdV.IqRef = limit( AxisRscI->IntAdV.IqOfRef, swk1 ); /* <V224> 外　乱　ト　ル　ク 加　算　後 のq軸電流指令       */
        swk10 = AxisRscI->IntAdV.IqRef + swk1;
        swk11 = AxisRscI->StsFlg.CtrlStsRW | TLIM;    /* TLIM flag set                    */
        AxisRscI->StsFlg.CtrlStsRW = cmove((swk10 == 0), swk11, AxisRscI->StsFlg.CtrlStsRW);    /* TLIM flag set
        */
      }
/*-------------------------------------------------------------------------------------------------*/
/*    TMP1 = limit( IntAdV.IqRef - IntAdV.IqInData , 2^15 - 1 )                    */
/*-------------------------------------------------------------------------------------------------*/
      swk1 = sub_limitf(AxisRscI->IntAdV.IqRef, AxisRscI->IntAdV.IqInData); /* TMP1 <-- limit( TMP1 , 2^15 - 1 )       */

/*-------------------------------------------------------------------------------------------------*/
/*    TMP2 = limit( IntAdP.KqP * TMP1 / 2^9 , 2^15 - 1 )                    */
/*-------------------------------------------------------------------------------------------------*/
      swk2 = mulshr_limitf(AxisRscI->IntAdP.KqP, swk1, 9);  /* TMP2 <-- limit( TMP2 , 2^15 - 1 )          */
/*-------------------------------------------------------------------------------------------------*/
/*    AcrV.IqIntgl(32) = (IntAdP.KqI * TMP1)<<3 + AcrV.IqIntgl(32)                    */
/*    IQIH = limit( IQIH , IntAdP.VqLim )                    */
/*-------------------------------------------------------------------------------------------------*/
      if( ( (AxisRscI->IntAdP.CtrlSw & INT_ST) == 0) || ( (AxisRscI->StsFlg.IntglFlg & 1) == 0 ) )
      {
        lwk6 = mul(AxisRscI->IntAdP.KqI, swk1); /* ACC <-- IntAdP.KqI * TMP1                    */
        lwk4 = (ULONG)AxisRscI->IntAdP.VqLim; /*                    */
        lwk4 = lwk4 << 16;  /*                    */
        lwk6 = lwk6 << 3; /*                    */
        AxisRscI->AcrV.IqIntgl.l = add_limitf(lwk6, AxisRscI->AcrV.IqIntgl.l);  /* AcrV.IqIntgl <-- limit( AcrV.IqIntgl , 2^32
        - 1 )          */
        if( LPX_ABS(AxisRscI->AcrV.IqIntgl.l) > LPX_ABS(lwk4) )
        {
          AxisRscI->StsFlg.CtrlStsRW = AxisRscI->StsFlg.CtrlStsRW | QLIM;    /* IMM3 <-- STAT | QLIM (imm_16)              */
          swk10 = AxisRscI->IntAdP.CtrlSw & ICLR;
          AxisRscI->AcrV.IqIntgl.l = cmove((swk10 != 0), (LONG)ZEROR, AxisRscI->AcrV.IqIntgl.l);
        }
      }
/*-------------------------------------------------------------------------------------------------*/
/*    VcmpV.VqOut = limit( TMP2 + IQIH +TMP3 , 2^15 - 1 )                    */
/*-------------------------------------------------------------------------------------------------*/
```

```
      swk1 = add_limitf(AxisRscI->AcrV.IqIntgl.s[1], swk2); /* TMP1 <-- limit( TMP1 , 2^15 - 1 )        */
/*-------------------------------------------------------------------------------------------------*/
/*    filter : AcrV.VqFil = ( ( ( TMP1 - VQFH ) * IntAdP.Tfil ) << 2 ) + AcrV.VqFil                */
/*-------------------------------------------------------------------------------------------------*/
      swk1 = sub_limitf(swk1, AxisRscI->AcrV.VqFil.s[1]); /* TMP1 <-- limit( TMP1 , 2^15 - 1 )        */
      lwk0 = mul(AxisRscI->IntAdP.Tfil, swk1 ) << 2;  /*                                          */
      AxisRscI->AcrV.VqFil.l = add_limitf(AxisRscI->AcrV.VqFil.l, lwk0);

/*************************************************************************************************/
/*                                                 */
/*    Voltage Compensation(電　圧　補償 )                                  */
/*                                                 */
/*************************************************************************************************/
      if( (AxisRscI->IntAdP.CtrlSw & ISEL) != 0 )
      {
        swk1 = AxisRscI->WeakFV.IdOut;  /* TMP1 <-- reference ID                  */
        swk2 = AxisRscI->IntAdV.IqRef;  /*                                  */
      }
      else
      {
        swk1 = AxisRscI->IntAdV.IdInData; /* TMP1 <-- feedback ID                    */
        swk2 = AxisRscI->IntAdV.IqInData; /* TMP2 <-- feedback IQ                    */
      }
/*-------------------------------------------------------------------------------------------------*/
/*    TMP4(VcmpV.VdComp) = IntAdP.MotResist*TMP1/2^15 - VcmpV.LqC * TMP2 / 2^15                    */
/*-------------------------------------------------------------------------------------------------*/
      swk4 = mulshr(AxisRscI->VcmpV.LqC, swk2, 15 );    /* VcmpV.VdComp <-- ACC >> 15                */
      swk0 = mulshr(AxisRscI->IntAdP.MotResist, swk1, 15 );
      swk4 = swk0 - swk4;
/*-------------------------------------------------------------------------------------------------*/
/*    TMP5(VcmpV.VqComp) = VcmpV.LdC * TMP1 / 2^15 + VcmpV.MagC + IntAdP.MotResist*TMP2/2^15                    */
/*-------------------------------------------------------------------------------------------------*/
      swk3 = mulshr(AxisRscI->VcmpV.LdC, swk1, 15 );  /* TMP3 <-- ACC >> 15                  */
      swk0 = mulshr(AxisRscI->IntAdP.MotResist, swk2, 15 );
      swk3 = swk3 + AxisRscI->VcmpV.MagC;
      swk5 = swk3 + swk0; /* VcmpV.VqComp <-- VcmpV.MagC + TMP3 + TMP0        */
```

```
/*-------------------------------------------------------------------------------------------------------*/
/*    if(IntAdP.CtrlSw & DIDTSET) VcmpV.VdComp = TMP4 + KDD * (IntAdV.IdDataP - IntAdV.IdInData),
IntAdV.IdDataP=IntAdV.IdInData                    */
/*              VcmpV.VqComp = TMP5 + KQD * (IntAdV.IqDataP - IntAdV.IqRef), IntAdV.IqDataP=IntAdV.IqRef                */
/*-------------------------------------------------------------------------------------------------------*/
      if( (AxisRscI->IntAdP.CtrlSw & DIDTSEL) == 0 )
      {
        AxisRscI->VcmpV.VdComp = swk4;  /*                                  */
        AxisRscI->VcmpV.VqComp = swk5;  /*                                  */
      }
/*-------------------------------------------------------------------------------------------------------*/
/*    filter : I*FL =  ( ( ( TMP1 - I*FH ) * IntAdP.Tfil ) << 2 ) + I*FL                  */
/*-------------------------------------------------------------------------------------------------------*/
      else
      {
        swk1 = AxisRscI->WeakFV.IdOut;  /*                                  */
        swk1 = sub_limitf(swk1, AxisRscI->IntAdV.IdLfil.s[1]);  /*                          */
        lwk0 = mul(AxisRscI->IntAdP.Tfil, swk1) << 2; /*                          */
        AxisRscI->IntAdV.IdLfil.l = add_limitf(AxisRscI->IntAdV.IdLfil.l, lwk0);  /*                                */
/*-------------------------------------------------------------------------------------------------------*/
        swk1 = AxisRscI->IntAdV.IqRef;  /*                                  */
        swk1 = sub_limitf(swk1, AxisRscI->IntAdV.IqLfil.s[1]);  /*                          */
        lwk0 = mul(AxisRscI->IntAdP.Tfil, swk1 ) << 2;  /*                          */
        AxisRscI->IntAdV.IqLfil.l = add_limitf(AxisRscI->IntAdV.IqLfil.l, lwk0);  /*                                */
/* ------------------------------------------------------------------------------------------------- */
        swk2 = AxisRscI->IntAdV.IdLfil.s[1] - AxisRscI->IntAdV.IdDataP; /*                          */
        AxisRscI->IntAdV.IdDataP = AxisRscI->IntAdV.IdLfil.s[1];  /*                          */
        swk2 = mulshr_limitf(AxisRscI->IntAdP.L_dIdt, swk2, 9); /* limit( VDL , 2^15 - 1 )              */
        AxisRscI->VcmpV.VdComp = add_limitf(swk2, swk4);  /* VcmpV.VdComp <-- limit( VcmpV.VdOut , 2^15 - 1 )              */
/*-------------------------------------------------------------------------------------------------------*/
        swk2 = AxisRscI->IntAdV.IqLfil.s[1] - AxisRscI->IntAdV.IqDataP; /*                          */
        AxisRscI->IntAdV.IqDataP = AxisRscI->IntAdV.IqLfil.s[1];
        swk2 = mulshr_limitf(AxisRscI->IntAdP.L_dIdt, swk2, 9); /* limit( VQL , 2^15 - 1 )              */
        AxisRscI->VcmpV.VqComp = add_limitf(swk2, swk5);  /* VcmpV.VqComp <-- limit( VcmpV.VqOut , 2^15 - 1 )              */
      }
/*-------------------------------------------------------------------------------------------------------*/
/*    TMP1 = limit( VDFH + VcmpV.VdComp , 2^15 - 1 )                          */
```

```
/*    TMP2 = limit( VQFH + VcmpV.VqComp , 2^15 - 1 )                                  */
/*-------------------------------------------------------------------------------------*/
      swk1 = add_limitf(AxisRscI->AcrV.VdFil.s[1], AxisRscI->VcmpV.VdComp); /* VcmpV.VdOut <-- limit( VcmpV.VdOut , 2^15 - 1 )
      */
      swk2 = add_limitf(AxisRscI->AcrV.VqFil.s[1], AxisRscI->VcmpV.VqComp); /* VcmpV.VqOut <-- limit( VcmpV.VqOut , 2^15 - 1 )
      */
/*-------------------------------------------------------------------------------------*/
/*    TMP1 = limit( VcmpV.VdRef + TMP1 , 2^15 - 1 )                                   */
/*    TMP2 = limit( VcmpV.VqRef + TMP2 , 2^15 - 1 )                                   */
/*-------------------------------------------------------------------------------------*/
      swk1 = add_limitf(AxisRscI->VcmpV.VdRef, swk1); /* VcmpV.VdOut <-- limit( VcmpV.VdOut , 2^15 - 1 )            */
      swk2 = add_limitf(AxisRscI->VcmpV.VqRef, swk2); /* VcmpV.VqOut <-- limit( VcmpV.VqOut , 2^15 - 1 )            */
/*-------------------------------------------------------------------------------------*/
/*    VcmpV.VdOut = limit( IntAdP.Kvv * TMP1 / 2^13 , 2^15 - 1 )                      */
/*    VcmpV.VqOut = limit( IntAdP.Kvv * TMP2 / 2^13 , 2^15 - 1 )                      */
/*-------------------------------------------------------------------------------------*/
      AxisRscI->VcmpV.VdOut = mulshr_limitf(AxisRscI->IntAdP.Kvv, swk1, 13);  /* VcmpV.VdOut   <-- limit( TMP1 , 2^15 - 1 )
      */
      AxisRscI->VcmpV.VqOut = mulshr_limitf(AxisRscI->IntAdP.Kvv, swk2, 13);    /* VcmpV.VqOut   <-- limit( TMP2 , 2^15 - 1 )
      */
      AxisRscI->WeakFV.WfVdRef = AxisRscI->VcmpV.VdOut;   /* d軸  電  圧  指 令保存     <V531>             */
      AxisRscI->WeakFV.WfVqRef = AxisRscI->VcmpV.VqOut;   /* q軸  電  圧  指 令保存     <V531>             */
/*************************************************************************************/
/*    電  圧  ベ  ク  ト  ル 補 正値計算     <V537> 新  弱  め  界  磁  制  御  以  外  は  こ  の処理 をジャンプする      */
/*************************************************************************************/
      if( (AxisRscI->IntAdP.CtrlSw & V_FB2) != 0 )
      {
/*************************************************************************************/
/*    Get modulation              <V531> 変  調  率  計  算を移動                    */
/*************************************************************************************/
        lwk2 = mul(AxisRscI->VcmpV.VdOut, AxisRscI->VcmpV.VdOut);
        lwk2 = mac(AxisRscI->VcmpV.VqOut, AxisRscI->VcmpV.VqOut, lwk2);
        swk0 = MpSQRT( lwk2 );          /* TMP0 = √ (VcmpV.VdOut^2 + VcmpV.VqOut^2)                */
        AxisRscI->IntAdV.V1 = swk0;   /* IntAdV.V1   = TMP0                                */
/*************************************************************************************/
/*    飽  和  判断              <V531> IntAdV.V1 > 8192*127%(10403.8) -> 飽  和  状態         */
/*************************************************************************************/
```

```
        AxisRscI->VcmpV.Vmax2 = 10403;  /* VcmpV.Vmax2 = 8192 * 1.27                    */
        AxisRscI->VcmpV.V12 = AxisRscI->IntAdV.V1;      /* VcmpV.V12 = √ (VcmpV.VdOut^2 + VcmpV.VqOut^2)                  */
        swk10 = AxisRscI->VcmpV.Vmax2 >> 1;   /* VcmpV.Vmax2 = 8192 * 1.27 / 2                */
        swk11 = AxisRscI->IntAdV.V1 >> 1;   /* VcmpV.V12 = √ (VcmpV.VdOut^2 + VcmpV.VqOut^2) / 2              */
        AxisRscI->VcmpV.Vmax2 = cmove((AxisRscI->IntAdV.V1 < 0), swk10, AxisRscI->VcmpV.Vmax2);
        AxisRscI->VcmpV.V12   = cmove((AxisRscI->IntAdV.V1 < 0), swk11, AxisRscI->VcmpV.V12);
        if( AxisRscI->VcmpV.Vmax2 < AxisRscI->VcmpV.V12 )
        {
          AxisRscI->IntAdV.V1 = 10403;    /* IntAdV.V1 = IntAdP.Vmax( 8192 * 1.27 )            */
          AxisRscI->StsFlg.IntglFlg = AxisRscI->StsFlg.IntglFlg | 1;      /* 積　分　停　止　フ　ラグセット                  */
//;*************************************************************************************************
//;*    電　圧　ベ　ク　ト　ル 補 正値計算        <V531> VcmpV.VdOut',VcmpV.VqOut' = IntAdP.Vmax / IntAdV.V1 * VcmpV.VdOut, VcmpV.VqOut
<V537> 削　除     *
//;*************************************************************************************************
/*-------------------------------------------------------------------------------------------------*/
/*    電　圧　制　限　テ　ー　ブ　ルアドレ ス取 得                                 */
/*-------------------------------------------------------------------------------------------------*/
        lwk2 = mul(AxisRscI->VcmpV.V12, AxisRscI->VcmpV.V12); /* TMP3,2 = VcmpV.V12^2                  */
        lwk2 = lwk2 -  0x00400000;  /* TMP3,2 = IntAdV.V1^2 - 2^22                  */
        lwk2 = lwk2 >> 4; /* TMP3,2 = (VcmpV.V12^2 - 2^22) / 2^4            */
        swk0 = (USHORT)( lwk2 >> 16 );  /* TMP0 = (VcmpV.V12^2 - 2^22) / 2^4 / 2^16 = addr      */
        lwk2 = lwk2 & 0x0000ffff; /* TMP2 = { (VcmpV.V12^2 - 2^22) / 2^4 } & 0x0000ffff       */
/*-------------------------------------------------------------------------------------------------*/
/*    電　圧　制　限　ベ　ク　ト　ル　直　線補 間用デー タ取得                          */
/*-------------------------------------------------------------------------------------------------*/
        lwk4 = 65536; /* TMP5,TMP4 = 65536                    */
        lwk6 = lwk4 - lwk2; /* TMP7,6 = 10000h - Table Index (Lo) -> (addr*2^16-low)  */
        IxTblVlmt16( swk8, swk0 );  /* TMP8 : テ　ー　ブ　ル　デ　ー　タ　読　み　出　し(読み出 し　ア ドレスad dr)  *//* tanaka21,コ　ン
        */
        lwk6 = (ULONG)swk8 * lwk6;  /* TMP6 = tblrv(addr)*(2^16-low)            */
        swk0 = swk0 + 1;    /* TMP0 = addr+1                    */
        IxTblVlmt16( swk8, swk0 );  /* TMP8 : テ　ー　ブ　ル　デ　ー　タ　読　み　出　し(読み出 し　ア ドレスad dr+1)  *//*
        tanaka21,コ　ン　パ　イ　ラ 対応待ち    */
        lwk4 = (ULONG)swk8 * lwk2;  /* TMP4 = tblrv(addr+1)*low              */
        lwk0 = lwk6 + lwk4; /* TMP0 = tblrv(addr)*(2^16-low) + tblrv(addr+1)*low  */
/*-------------------------------------------------------------------------------------------------*/
/*    電　圧　電　圧　ベ　ク　ト ル補正値 計算                                  */
```

```
/*-----------------------------------------------------------------------------------------------*/
          swk8 = AxisRscI->VcmpV.Vmax2; /* TMP8 = VcmpV.Vmax2                                    */
          lwk2 = mulshr((ULONG)swk8, lwk0, 28 );    /* TMP2 = MAC / 2^28                         */
          AxisRscI->VcmpV.VdOut = mulshr(swk2, AxisRscI->VcmpV.VdOut, 14 );     /* VcmpV.VdOut = IntAdP.Vmax / VcmpV.V12 *
          VcmpV.VdOut * 2^(13+13+16) / 2^(28+14)      */
          AxisRscI->VcmpV.VqOut = mulshr(swk2, AxisRscI->VcmpV.VqOut, 14 );     /* VcmpV.VqOut = IntAdP.Vmax / VcmpV.V12 *
          VcmpV.VqOut * 2^(13+13+16) / 2^(28+14)      */
        }
        else
        {
          AxisRscI->StsFlg.IntglFlg = AxisRscI->StsFlg.IntglFlg & 0xFFFE;   /* 積　分　停　止　フ　ラグクリア                    */
        }
      }
/*************************************************************************************************/
/*                                                */
/*    UVW transform : dq( 2phase ) to UVW( 3phase ) Transform                    */
/*                                                */
/*************************************************************************************************/
/*-----------------------------------------------------------------------------------------------*/
/*    VcmpV.VuOut = limit( SinTbl.CosT * VcmpV.VdOut / 2^14 - SinTbl.SinT * VcmpV.VqOut / 2^14 , 2^15 - 1 )               */
/*-----------------------------------------------------------------------------------------------*/
      swk4 = AxisRscI->IntAdP.Vmax; /*                                  */
      swk1 = mulshr(AxisRscI->SinTbl.CosT, AxisRscI->VcmpV.VdOut, 14 ); /* TMP1 <-- ACC >> 14                  */
      swk2 = mulshr(AxisRscI->SinTbl.SinT, AxisRscI->VcmpV.VqOut, 14 );   /* TMP2 <-- ACC >> 14                  */
      AxisRscI->VcmpV.VuOut = sub_limitf(swk1, swk2);     /* VcmpV.VuOut <-- limit( VcmpV.VuOut , 2^15 - 1 )               */
      AxisRscI->VcmpV.VuOut = IxLIMIT( AxisRscI->VcmpV.VuOut, swk4 ); /*                                  */
/*-----------------------------------------------------------------------------------------------*/
/*    VcmpV.VvOut = limit( SinTbl.CosT3 * VcmpV.VdOut / 2^14 - SinTbl.SinT3 * VcmpV.VqOut / 2^14 , 2^15 - 1 )               */
/*-----------------------------------------------------------------------------------------------*/
      swk1 = mulshr(AxisRscI->SinTbl.CosT3, AxisRscI->VcmpV.VdOut, 14 );  /* TMP1 <-- ACC >> 14                  */
      swk2 = mulshr(AxisRscI->SinTbl.SinT3, AxisRscI->VcmpV.VqOut, 14 );    /* TMP2 <-- ACC >> 14                  */
      AxisRscI->VcmpV.VvOut = sub_limitf(swk1, swk2);     /* VcmpV.VvOut <-- limit( VcmpV.VvOut , 2^15 - 1 )               */
      AxisRscI->VcmpV.VvOut = IxLIMIT( AxisRscI->VcmpV.VvOut, swk4 ); /*                                  */
/*-----------------------------------------------------------------------------------------------*/
/*    VcmpV.VwOut = limit( - VcmpV.VuOut - VcmpV.VvOut , 2^15 - 1 )                                */
/*-----------------------------------------------------------------------------------------------*/
      swk1 = (SHORT)ZEROR - AxisRscI->VcmpV.VuOut;  /* VcmpV.VwOut <-- - VcmpV.VuOut - VcmpV.VvOut                  */
```

```
      AxisRscI->VcmpV.VwOut = sub_limitf(swk1, AxisRscI->VcmpV.VvOut);      /* VcmpV.VwOut <-- limit( VcmpV.VwOut , 2^15 - 1 )
      */
      AxisRscI->VcmpV.VwOut = IxLIMIT( AxisRscI->VcmpV.VwOut, swk4 ); /*                                    */


#if 0 /* for debug 2013.07.17 tanaka21*/
/*****************************************************************************************************/
/*    新 弱 め 界 磁 制 御 判断処理    <V537> 新 弱 め 界 磁 の 場 合 変 調 率 計 算 ,   飽和判断処理を ジャンプする  */
/*****************************************************************************************************/
      if( (AxisRscI->IntAdP.CtrlSw & V_FB2) == 0 )
      {
/*****************************************************************************************************/
/*    Get modulation          <V531> 変 調 率 計 算 は  2 相  3 相変換前に  す る <V537> 復活      */
/*****************************************************************************************************/
        lwk2 = mul(AxisRscI->VcmpV.VdOut, AxisRscI->VcmpV.VdOut);
        lwk2 = mac(AxisRscI->VcmpV.VqOut, AxisRscI->VcmpV.VqOut, lwk2);
        swk0 = MpSQRT( lwk2 );
        if( (USHORT)swk0 > 0x7FFF )
        {
          swk0 = 0x7FFF;  /* √  の  計 算が3 2 7 6 7 を 超えた  ら  、32767にす る  。          ;<V350>  */
        }
        AxisRscI->IntAdV.V1 = swk0;
/*---------------------------------------------------------------------------------------------------*/
/*    飽 和  判断           <V531> <V537> 復  活                        */
/*---------------------------------------------------------------------------------------------------*/
        AxisRscI->StsFlg.IntglFlg = AxisRscI->StsFlg.IntglFlg & 0xFFFE;   /*                          */
        swk10 = AxisRscI->StsFlg.IntglFlg | 1;      /*                        */
        AxisRscI->StsFlg.IntglFlg = cmove((AxisRscI->IntAdV.V1 >= 9421), swk10, AxisRscI->StsFlg.IntglFlg);
      }
/*****************************************************************************************************/
/*    Over modulation type select                                */
/*****************************************************************************************************/
      if( AxisRscI->IntAdP.Vmax >= 0x2000 )
      {
        if( (AxisRscI->IntAdP.CtrlSw & OVMSEL2) == 0 )
        {
          if( ( AxisRscI->IntAdV.V1 >= 0x2000 )&&( (AxisRscI->IntAdP.CtrlSw & OVMSEL1) != 0 ) )
```

```
          {
/*****************************************************************************************************************/
/*    Over modulation1                                                                                           */
/*****************************************************************************************************************/
            IxSetCtblAdr( pCtbl, &(OVMODTBLG[0][0]) );  /* gain type                                                 */
            MpOVMMODK( &AxisRscI->IntAdP, &AxisRscI->IntAdV, pCtbl );
            AxisRscI->VcmpV.VuOut = mulshr_limitf(AxisRscI->VcmpV.VuOut, AxisRscI->IntAdP.Kmod, 13);
            AxisRscI->VcmpV.VvOut = mulshr_limitf(AxisRscI->VcmpV.VvOut, AxisRscI->IntAdP.Kmod, 13);
            AxisRscI->VcmpV.VwOut = mulshr_limitf(AxisRscI->VcmpV.VwOut, AxisRscI->IntAdP.Kmod, 13);
/*---------------------------------------------------------------------------------------------------------------*/
/*    TMP1 = |VcmpV.VuOut|,      TMP2 = |VcmpV.VvOut|,      TMP3 = |VcmpV.VwOut|           */
/*    TMP4 = sign(VcmpV.VuOut), TMP5 = sign(VcmpV.VvOut), TMP6 = sign(VcmpV.VwOut)         */
/*---------------------------------------------------------------------------------------------------------------*/
            swk0 = 1;
            swk4 = IxLIMIT( AxisRscI->VcmpV.VuOut, swk0 );
            swk1 = swk4 * AxisRscI->VcmpV.VuOut;
            swk5 = IxLIMIT( AxisRscI->VcmpV.VvOut, swk0 );
            swk2 = swk5 * AxisRscI->VcmpV.VvOut;
            swk6 = IxLIMIT( AxisRscI->VcmpV.VwOut, swk0 );
            swk3 = swk6 * AxisRscI->VcmpV.VwOut;
            if( swk1 >= swk2 )
            {
              if( swk1 >= swk3 )
              {
                swk1 = swk1 - 0x2000; /* TMP1 <-- |VcmpV.VuOut|-2000h                */
                IxLmtzImm16( swk1, 0x7fff );  /* zero limit                      */
                swk0 = swk4 * swk1;
              }
              else
              {
                swk3 = swk3 - 0x2000; /* TMP0 <-- |VcmpV.VwOut|-2000h                */
                IxLmtzImm16( swk3, 0x7fff );  /* zero limit                      */
                swk0 = swk6 * swk3;
              }
            }
            else
            {
```

```
            if( swk2 >= swk3 )
            {
              swk2 = swk2 - 0x2000; /* TMP0 <-- |VcmpV.VvOut|-2000h                  */
              IxLmtzImm16( swk2, 0x7fff );  /* zero limit                        */
              swk0 = swk5 * swk2;
            }
            else
            {
              swk3 = swk3 - 0x2000; /* TMP0 <-- |VcmpV.VwOut|-2000h                  */
              IxLmtzImm16( swk3, 0x7fff );  /* zero limit                        */
              swk0 = swk6 * swk3;
            }
          }
          AxisRscI->VcmpV.VuOut = sub_limitf(AxisRscI->VcmpV.VuOut, swk0);      /*                                  */
          AxisRscI->VcmpV.VvOut = sub_limitf(AxisRscI->VcmpV.VvOut, swk0);      /*                                  */
          AxisRscI->VcmpV.VwOut = sub_limitf(AxisRscI->VcmpV.VwOut, swk0);      /*                                  */
          AxisRscI->IntAdV.Vcent = swk0;
        }
      }
/*****************************************************************************************************************/
/*    Over modulation2                                          */
/*****************************************************************************************************************/
      else
      {
        IxSetCtblAdr( pCtbl, &(OVMODTBL0[0][0]) );  /* ofset type                  */
        MpOVMMODK( &AxisRscI->IntAdP, &AxisRscI->IntAdV, pCtbl );
/*---------------------------------------------------------------------------------------------------------------*/
/*    MAX = TMP1, MIN = TMP2                                    */
/*    OFS = (TMP1+TMP2)/2                                      */
/*---------------------------------------------------------------------------------------------------------------*/
        if( AxisRscI->VcmpV.VuOut >= AxisRscI->VcmpV.VvOut )
        {
          swk1 = AxisRscI->VcmpV.VuOut;
          swk2 = AxisRscI->VcmpV.VvOut;
        }
        else
        {
```

```
            swk1 = AxisRscI->VcmpV.VvOut;
            swk2 = AxisRscI->VcmpV.VuOut;
          }
          if( swk1 < AxisRscI->VcmpV.VwOut )
          {
            swk1 = AxisRscI->VcmpV.VwOut;
          }
          else
          {
            if( AxisRscI->VcmpV.VwOut < swk2 )
            {
              swk2 = AxisRscI->VcmpV.VwOut;
            }
          }
          swk0 = add_limitf(swk2, swk1);  /*                              */
          swk0 = mulshr(swk0, ONE, 1);
/*-------------------------------------------------------------------------------------------------*/
          AxisRscI->VcmpV.VuOut = sub_limitf(AxisRscI->VcmpV.VuOut, swk0);      /*                   */
          AxisRscI->VcmpV.VvOut = sub_limitf(AxisRscI->VcmpV.VvOut, swk0);      /*                   */
          AxisRscI->VcmpV.VwOut = sub_limitf(AxisRscI->VcmpV.VwOut, swk0);      /*                   */
          AxisRscI->IntAdV.Vcent = swk0;
/*-------------------------------------------------------------------------------------------------*/
          swk0 = 1;
/*-------------------------------------------------------------------------------------------------*/
          swk0 = IxLIMIT( AxisRscI->VcmpV.VuOut, swk0 );  /* TMP1= -1/0/+1                 */
          swk1 = swk1 | 1;    /* TMP1 = -1/+1 -----sign(VcmpV.VuOut)                */
          swk2 = swk1 * AxisRscI->IntAdP.Kmod;
          AxisRscI->VcmpV.VuOut = add_limitf( swk2, AxisRscI->VcmpV.VuOut );      /*                   */
/*-------------------------------------------------------------------------------------------------*/
          swk1 = IxLIMIT( AxisRscI->VcmpV.VvOut, swk0 );
          swk1 = swk1 | 1;    /* sign(VcmpV.VvOut)                          */
          swk2 = swk1 * AxisRscI->IntAdP.Kmod;
          AxisRscI->VcmpV.VvOut = add_limitf( swk2, AxisRscI->VcmpV.VvOut );      /*                   */
/*-------------------------------------------------------------------------------------------------*/
          swk1 = IxLIMIT( AxisRscI->VcmpV.VwOut, swk0 );
          swk1 = swk1 | 1;    /* sign(VcmpV.VwOut)                          */
          swk2 = swk1 * AxisRscI->IntAdP.Kmod;
```

```
          AxisRscI->VcmpV.VwOut = add_limitf( swk2, AxisRscI->VcmpV.VwOut );      /*                                  */
        }
      }
#endif  //#if 0 /* for debug 2013.07.17 tanaka21*/

/*****************************************************************************************************************/
/*    On-Delay                                                  */
/*****************************************************************************************************************/
/*---------------------------------------------------------------------------------------------------------------*/
/*  IU, IV reference calc                                       */
/*---------------------------------------------------------------------------------------------------------------*/
      swk1 = mulshr(AxisRscI->WeakFV.IdOut, AxisRscI->SinTbl.CosT, 14 );  /* TMP1 <-- ACC >> 14                    */
      swk2 = mulshr(AxisRscI->IntAdV.IqRef, AxisRscI->SinTbl.SinT, 14 );    /* TMP2 <-- ACC >> 14                    */
      AxisRscI->IntAdV.IuOut = swk1 - swk2; /* IntAdV.IuOut  <--  TMP1 - TMP2                */

      swk3 = mulshr(AxisRscI->WeakFV.IdOut, AxisRscI->SinTbl.CosT3, 14 ); /* TMP3 <-- ACC >> 14                    */
      swk4 = mulshr(AxisRscI->IntAdV.IqRef, AxisRscI->SinTbl.SinT3, 14 );   /* TMP4 <-- ACC >> 14                    */
      AxisRscI->IntAdV.IvOut = swk3 - swk4; /* IntAdV.IvOut  <--  TMP3 - TMP4                */
/*****************************************************************************************************************/
//     if ( |IntAdV.IuInData| < IntAdP.OnDelayLvl ) TMP1 = IntAdV.IuOut /* Reference */
//     else                 TMP1 = IntAdV.IuInData
//     if ( |IntAdV.IvInData| < IntAdP.OnDelayLvl ) TMP2 = IntAdV.IvOut /* Reference */
//     else                 TMP2 = IntAdV.IvInData
//     if ( |IWD| < IntAdP.OnDelayLvl ) TMP2 = IWO  /* Reference */
//     else                 TMP2 = IWD
/*****************************************************************************************************************/
      swk5 = AxisRscI->IntAdP.OnDelayLvl;
      if(LPX_ABS(AxisRscI->IntAdV.IuInData) > LPX_ABS(swk5))  //110530tanaka21作  業  メモ s w k 2 を  以  降  使  わ  な  い ため代入 は行
      {
        swk1 = AxisRscI->IntAdV.IuInData; /* TMP1 <-- IntAdV.IuInData                      */
      }
      else
      {
        swk1 = AxisRscI->IntAdV.IuOut;  /* TMP1 <-- IntAdV.IuOut                        */
      }
      if( LPX_ABS(AxisRscI->IntAdV.IvInData) > LPX_ABS(swk5 ) ) //110530tanaka21作  業  メモ
      swk2を  以  降  使  わ  な  い  た  め 代入は行なわない
```

```
      {
        swk2 = AxisRscI->IntAdV.IvInData; /* TMP2 <-- IntAdV.IvInData                        */
      }
      else
      {
        swk2 = AxisRscI->IntAdV.IvOut;  /* TMP2 <-- IntAdV.IvOut                           */
      }
      swk3 = -AxisRscI->IntAdV.IuInData - AxisRscI->IntAdV.IvInData;  /* TMP3(IWD) <-- - TMP1 - TMP2          */
      if( LPX_ABS(swk3) <= LPX_ABS(swk5) )  //110530tanaka21作 業 メモ s w k 4 を 以 降 使 わ な い ため代入 は行なわない
      {
        swk3 = -AxisRscI->IntAdV.IuOut - AxisRscI->IntAdV.IvOut;  /* TMP3                       */
      }
      swk7 = 0x2000;  /* TMP7 <-- 2000h                    */
      swk5 = 1; /* TMP5 <-- 1                        */
/*-----------------------------------------------------------------------------------------------*/
/*    if(IntAdP.OnDelaySlope != 0) trapezoid type else rectangle type                            */
/*-----------------------------------------------------------------------------------------------*/
      if( AxisRscI->IntAdP.OnDelaySlope == 0 )
      {
/*-----------------------------------------------------------------------------------------------*/
/*    TMP1(ONDVU) = sign(IU)*IntAdP.OnDelayComp                                                  */
/*-----------------------------------------------------------------------------------------------*/
        swk6 = IxLIMIT( swk1, swk5 ); /* TMP6 = -1/0/+1             */
        swk1 = AxisRscI->IntAdP.OnDelayComp * swk6;
/*-----------------------------------------------------------------------------------------------*/
/*    TMP2(ONDVU) = sign(IV)*IntAdP.OnDelayComp                                                  */
/*-----------------------------------------------------------------------------------------------*/
        swk6 = IxLIMIT( swk2, swk5 );
        swk2 = AxisRscI->IntAdP.OnDelayComp * swk6;
/*-----------------------------------------------------------------------------------------------*/
/*    TMP3(ONDVU) = sign(IW)*IntAdP.OnDelayComp                                                  */
/*-----------------------------------------------------------------------------------------------*/
        swk6 = IxLIMIT( swk3, swk5 );
        swk3 = AxisRscI->IntAdP.OnDelayComp * swk6;
      }
/*-----------------------------------------------------------------------------------------------*/
/*    trapezoid type                                          */
```

```
/*-------------------------------------------------------------------------------------------------*/
      else
      {
        swk0 = mulshr_limitf(AxisRscI->IntAdP.OnDelaySlope, swk1, 8 );    /* TMP0 <-- IU*IntAdP.OnDelaySlope>>8
        */
        swk0 = IxLIMIT( swk0, 8192 ); /* TMP0 = limit(TMP0,8192)                    */
        swk1 = mulshr(AxisRscI->IntAdP.OnDelayComp, swk0, 13 ); /* TMP1(ONDVU) = (IntAdP.OnDelayComp*TMP0)>>13    */
/*-------------------------------------------------------------------------------------------------*/
        swk0 = mulshr_limitf(AxisRscI->IntAdP.OnDelaySlope, swk2, 8); /* TMP0 = limit(TMP0,2^15-1)    */
        swk0 = IxLIMIT( swk0, 8192 ); /* TMP0 = limit(TMP0,8192)                    */
        swk2 = mulshr(AxisRscI->IntAdP.OnDelayComp, swk0, 13 ); /* TMP1(ONDVU) = (IntAdP.OnDelayComp*TMP0)>>13    */
/*-------------------------------------------------------------------------------------------------*/
        swk0 = mulshr_limitf(AxisRscI->IntAdP.OnDelaySlope, swk3, 8); /* TMP0 = limit(TMP0,2^15-1)    */
        swk0 = IxLIMIT( swk0, 8192 ); /* TMP0 = limit(TMP0,8192)                    */
        swk3 = mulshr(AxisRscI->IntAdP.OnDelayComp, swk0, 13 ); /* TMP1(ONDVU) = (IntAdP.OnDelayComp*TMP0)>>13    */
      }
/*-------------------------------------------------------------------------------------------------*/
/*************************************************************************************************/
/*    Voltage conversion to Carrier count range                                                  */
/*************************************************************************************************/
/*    -2000h..2000h  ---> 0h..4000h  ---> 0h..CRFRQ                                              */
/*************************************************************************************************/
      AxisRscI->VcmpV.VuOut = IxLIMIT( AxisRscI->VcmpV.VuOut, swk7 ); /* limit +-2000h            */
      AxisRscI->VcmpV.VvOut = IxLIMIT( AxisRscI->VcmpV.VvOut, swk7 );
      AxisRscI->VcmpV.VwOut = IxLIMIT( AxisRscI->VcmpV.VwOut, swk7 );

/* for debug */
      swk4 = swk7 - AxisRscI->VcmpV.VuOut;
      swk4 = mulshr(swk4, AxisRscI->IntAdV.CrFreqW, 14 );
      swk5 = swk7 - AxisRscI->VcmpV.VvOut;
      swk5 = mulshr(swk5, AxisRscI->IntAdV.CrFreqW, 14 );
      swk6 = swk7 - AxisRscI->VcmpV.VwOut;
      swk6 = mulshr(swk6, AxisRscI->IntAdV.CrFreqW, 14 );


/*-------------------------------------------------------------------------------------------------*/
/*    Deat-time compensation (timer) : if(Vx == 0 || Vx == IntAdV.CrFreqW) No compensation          */
```

```
/*-----------------------------------------------------------------------------------------------------*/
      if( ( swk4 != ZEROR ) && (swk4 != AxisRscI->IntAdV.CrFreqW ) )
      {
        swk4 = swk4 - swk1; /* VcmpV.VuOut <-- VcmpV.VuOut+ONDVU                */
        IxLmtzReg16( swk4, swk4, AxisRscI->IntAdV.CrFreqW );  /* VcmpV.VuOut <-- limitz( VcmpV.VuOut , IntAdV.CrFreqW )
        */
      }
      if( ( swk5 != ZEROR ) && (swk5 != AxisRscI->IntAdV.CrFreqW ) )
      {
        swk5 = swk5 - swk2; /* VcmpV.VvOut <-- VcmpV.VvOut+ONDVV                */
        IxLmtzReg16( swk5, swk5, AxisRscI->IntAdV.CrFreqW );  /* VcmpV.VvOut <-- limitz( VcmpV.VvOut , IntAdV.CrFreqW )
        */
      }
      if( ( swk6 != ZEROR ) && (swk6 != AxisRscI->IntAdV.CrFreqW ) )
      {
        swk6 = swk6 - swk3; /* VcmpV.VwOut <-- VcmpV.VwOut+ONDVW                */
        IxLmtzReg16( swk6, swk6, AxisRscI->IntAdV.CrFreqW );  /* VcmpV.VwOut <-- limitz( VcmpV.VwOut , IntAdV.CrFreqW )
        */
      }

      AxisRscI->PwmV.PwmCntT2 = swk6;
      AxisRscI->PwmV.PwmCntT1 = swk5;
      AxisRscI->PwmV.PwmCntT0 = swk4;

/*-----------------------------------------------------------------------------------------------------*/
/*    Output Voltage & status                                    */
/*-----------------------------------------------------------------------------------------------------*/
    }
//<2>#ifdef PREG_DEF
//    CTSTW = AxisRscI->StsFlg.CtrlStsRW; /* Status Set                          */
    AxisRscI->CtrlStsOut = AxisRscI->StsFlg.CtrlStsRW;  /* Status Set                          */
  }
#endif  //#ifdef  MULTI_AXIS             /* 多  軸  処  理有 効                        */


  /* Output PWM Data */
  SetPWM( &AxisHdl[0] );
```

```
/*-----------------------------------------------------------------------------------------------*/
  /* ★ H/W ア ク セ ス が 共 通 の も の を ま と め た い ！！0軸目っ て書く のが格好悪い★    */
  /* level(AD=3, INT1=0/4 HOST=0) */
  INTLVWR |= 0x0004;

//<2>#ifdef PREG_DEF
  OUTPT = 0x0;

  IniWk.IN_WK1H++;    /* for debug counter tanaka21 */


  return;
}


#if 0   /* JL086で 実 行 す る た め コメントアウト    */
/*************************************************************************************************/
/*                                                   */
/*    Encoder(SPG0) Interrupt Procedure   ; 通 常 ( 初 期 イ ン ク レ パルス出力 完了時 ):11clk 〈V720〉 */
/*                                                   */
/*    [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( 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] = SRPGORD5;   /* 今 回 値 読 込み：Position Low          */
    IEncWk.RxPos.s[1] = SRPGORD6; /* 今 回 値 読 込み：Position High          */
/*-------------------------------------------------------------------------------------------*/
    IEncWk.EncWk0 = INT1SET;      /* INT1 Acknowledge                */
/*-------------------------------------------------------------------------------------------*/
    return;           /* return                      */
}



/*********************************************************************************************/
/*                                                   */
/*    分 周 パ ル ス 更新 処理           ; 最 大:???clk,   通常:???cl k      <V720>  */
/*                                                   */
/*********************************************************************************************/
void  MpUPDATE_DIVPOS( void )
{
/*-------------------------------------------------------------------------------------------*/
    IHostWk.Divuswk = INT1SET;    /* INT1 Acknowledge             <V741>  */
/*-------------------------------------------------------------------------------------------*/
    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 N C 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 t C 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 o s 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<1.02>                                          */
/*-----------------------------------------------------------------------------------------------------*/
  AxisRsc->IntAdV.IqOut1L.l = ZEROR;  /*                  ; <V388> 追  加     */
  AxisRsc->IntAdV.IqOut1PL.l = ZEROR; /*                  ; <V388> 追  加     */
  AxisRsc->IntAdV.IqOut1PPL.l = ZEROR;  /*                   ; <V388> 追  加   */
  AxisRsc->IntAdV.IqIn1PL.l = ZEROR;  /*                  ; <V388> 追  加     */
  AxisRsc->IntAdV.IqIn1PPL.l = ZEROR; /*                  ; <V388> 追  加     */
  AxisRsc->IntAdV.IqOut2L.l = ZEROR;  /*                  ; <V388> 追  加     */
  AxisRsc->IntAdV.IqOut2PL.l = ZEROR; /*                  ; <V388> 追  加     */
  AxisRsc->IntAdV.IqOut2PPL.l = ZEROR;  /*                   ; <V388> 追  加   */
  AxisRsc->IntAdV.IqIn2PL.l = ZEROR;  /*                  ; <V388> 追  加     */
  AxisRsc->IntAdV.IqIn2PPL.l = ZEROR; /*                  ; <V388> 追  加   */
  AxisRsc->IntAdV.IqOut3L.l = ZEROR;  /*                  ; <V388> 追  加     */
  AxisRsc->IntAdV.IqOut3PL.l = ZEROR; /*                  ; <V388> 追  加     */
  AxisRsc->IntAdV.IqOut3PPL.l = ZEROR;  /*                   ; <V388> 追  加   */
  AxisRsc->IntAdV.IqIn3PL.l = ZEROR;  /*                  ; <V388> 追  加     */
  AxisRsc->IntAdV.IqIn3PPL.l = ZEROR; /*                  ; <V388> 追  加   */
/*-----------------------------------------------------------------------------------------------------*/
  AxisRsc->AcrV.IdIntgl.l = ZEROR;  /* integral(32bit) <-- 0                */
  AxisRsc->AcrV.IqIntgl.l = ZEROR;  /* integral(32bit) <-- 0                */
  AxisRsc->AcrV.VdFil.l = ZEROR;  /* vd filter out(32bit) <-- 0             */
  AxisRsc->AcrV.VqFil.l = ZEROR;  /* vq filter out(32bit) <-- 0             */
  AxisRsc->IntAdV.IqOut2Lpf.l = ZEROR;  /* iq filter out(32bit) <-- 0             */
  AxisRsc->IntAdV.IqRef = 0x0;    /* iq(after limit) <-- 0                  */
  AxisRsc->VcmpV.VdOut = 0x0;     /* vd <-- 0                          */
  AxisRsc->VcmpV.VqOut = 0x0;     /* vq <-- 0                          */
  AxisRsc->VcmpV.VuOut = 0x0;     /* vu <-- 0                          */
  AxisRsc->VcmpV.VvOut = 0x0;     /* vv <-- 0                          */
  AxisRsc->VcmpV.VwOut = 0x0;     /* vw <-- 0                          */
  AxisRsc->VcmpV.LdC = 0x0;
  AxisRsc->VcmpV.LqC = 0x0;
  AxisRsc->VcmpV.MagC = 0x0;
  AxisRsc->IntAdV.IuOut = 0x0;
```

```
  AxisRsc->IntAdV.IvOut = 0x0;
  AxisRsc->IntAdV.IdDataP = AxisRsc->IntAdV.IdInData;   /*                              */
  AxisRsc->IntAdV.IqDataP = AxisRsc->IntAdV.IqRef;    /*                              */
/*-------------------------------------------------------------------------------------*/
  AxisRsc->WeakFV.IdOut = 0;      /*                              */
  AxisRsc->VcmpV.VdOut = 0;     /*                             */
  AxisRsc->VcmpV.VqOut = 0;     /*                             */
  AxisRsc->IntAdV.IdLfil.l = ZEROR; /*                              */
  AxisRsc->IntAdV.IqLfil.l = ZEROR; /*                              */

  AxisRsc->WeakFV.WfIntgl.l = ZEROR;  /* <V214>                          */
  AxisRsc->WeakFV.WfVdRef = 0;    /* <V214>             ; 削  除<V309>  復  活<V531>  */
  AxisRsc->WeakFV.WfVqRef = 0;    /* <V214>             ; 削  除<V309>  復  活<V531>  */

/*-------------------------------------------------------------------------------------*/
  return;
}



/*****************************************************************************************/
/*                                          */
/*     SQRT(TMP2(32)) Sub-routin (MAX 1.21us)                    */
/*                                          */
/*****************************************************************************************/
/*    Input   TMP2 : Low  data                          */
/*          TMP3 : High data                        */
/*    Output    TMP0 : SQRT(dat)                          */
/*    Stack No. 0                                 */
/*    Work    TMP0,TMP1,TMP2,TMP3,TMP4,TMP5,TMP8                    */
/*          MACCL,MACCH,SACCL,SACCH                         */
/*****************************************************************************************/
//USHORT  MpSQRT( INTADWK *IntAdwk, ULONG src )
#if 0
USHORT  MpSQRT( ULONG src )   /* 2013.05.06 tanaka21 コ  ー  ド 整理<020>     */
{
    USHORT  Low;      /* 引  数  下位16 bit値          2013.05.06 tanaka21 コ  ー  ド 整理<020>    */
```

```
    USHORT  High;     /* 引  数  上位16 bit値          2013.05.06 tanaka21 コ  ー  ド 整理<020>    */
    USHORT  uswk0;      /* 平  方  根  演算用 1 6 b i t ワ ークレジスタ0    2013.05.06 tanaka21 コ  ー  ド 整理<020>    */
//    USHORT  uswk1;      /* 平  方  根  演算用 1 6 b i t ワ ークレジ スタ1     2013.05.06 tanaka21 コ  ー  ド 整理<020>    *//*
コ  メ  ン  ト  アウト  (  u swk0と統合) <022>    */
    USHORT  uswk3;      /* 平  方  根  演算用 1 6 b i t ワ ークレジスタ3    2013.05.06 tanaka21 コ  ー  ド 整理<020>    */
    USHORT  uswk4;      /* 平  方  根  演算用 1 6 b i t ワ ークレジスタ4    2013.05.06 tanaka21 コ  ー  ド 整理<020>    */
    USHORT  uswk5;      /* 平  方  根  演算用 1 6 b i t ワ ークレジスタ5    2013.05.06 tanaka21 コ  ー  ド 整理<020>    */
    USHORT  uswk6;      /* 平  方  根  演算用 1 6 b i t ワ ークレジスタ6    2013.05.06 tanaka21 コ  ー  ド 整理<020>    */
    ULONG ulwk0;      /* 平  方  根  演算用 3 2 b i t ワ ークレジスタ0    2013.05.06 tanaka21 コ  ー  ド 整理<020>    */
//    ULONG ulwk2;      /* 平  方  根  演算用 3 2 b i t ワ ークレジ スタ2     2013.05.06 tanaka21 コ  ー  ド 整理<020>    *//*
コ  メ  ン  ト  アウト  (  u swk0と統合) <022>    */
    DWREG tmp0;     /* 平  方  根  演算用16/ 3 2 b i t ワ ークレジスタ0   2013.05.06 tanaka21 コ  ー  ド 整理<020>    */

    Low = (USHORT)src;
    High = (USHORT)( src >> 16 );

/*-----------------------------------------------------------------------------------------------------*/
/*    TMP0(16) = sqrt(TMP2(32))                                    */
/*-----------------------------------------------------------------------------------------------------*/
/*    TMP3(High), TMP2(Low)   ---> TMP0(result)                        */
/*    table search from high 8bits                                 */
/*    and closely resemble using low 15 bits                           */
/*        |----|----|----|----|----|----|--------|                     */
/*       31   27   23   19   15   11   7        0                     */
/*    TMP8     0    2    4    6    8    10    12                      */
/*-----------------------------------------------------------------------------------------------------*/
//    uswk6 = 0;    /* 2013.05.06 tanaka21 コ  ー  ド 整理<0 20>    */
    if( High & 0xF000 )
/*-----------------------------------------------------------------------------------------------------*/
/*    TMP8 0                                                       */
/*    |xxxx|yyyy|aaaa|aaaa|aaaa|aaa-|--------|                       */
/*-----------------------------------------------------------------------------------------------------*/
    {
      uswk6 = 0;    /* 2013.05.06 tanaka21 コ  ー  ド 整理<020>     */
      tmp0.ul = ( src >> 9 ); /* TMP4 for approxmate(15bit)           */
      tmp0.us[0] = ( tmp0.us[0] & 0x7FFF ); /* mask 15bit                    */
      uswk5 = ( High >> 8 );  /* TMP5 for table search(8bit)          */
```

```
    }
    else if( High & 0x0F00 )
/*-----------------------------------------------------------------------------------------------------*/
/*    TMP8 2                                                */
/*    |0000|xxxx|yyyy|aaaa|aaaa|aaaa|aaa-----|                          */
/*-----------------------------------------------------------------------------------------------------*/
    {
      uswk6 = 2;
      tmp0.ul = ( src >> 5 ); /* TMP4 for approximate(15bit)              */
      tmp0.us[0] = ( tmp0.us[0] & 0x7FFF ); /* mask 15bit                        */
      uswk5 = ( High >> 4 );  /* TMP5 for table search(8bit)            */
    }
    else if( High & 0x00F0 )
/*-----------------------------------------------------------------------------------------------------*/
/*    TMP8 4                                                */
/*    |0000|0000|xxxx|yyyy|aaaa|aaaa|aaaaaaa-|                          */
/*-----------------------------------------------------------------------------------------------------*/
    {
      uswk6 = 4;
      uswk5 = High; /* TMP5 for table search(8bit)             */
      tmp0.us[0] = ( Low >> 1 );  /* TMP4 for approximate(15bit)              */
    }
    else if( High & 0x000F )
/*-----------------------------------------------------------------------------------------------------*/
/*    TMP8 6                                                */
/*    |0000|0000|0000|xxxx|yyyy|aaaa|aaaaaaaa|(000)                    */
/*-----------------------------------------------------------------------------------------------------*/
    {
      uswk6 = 6;
      uswk5 = (USHORT)(( src & 0x0FFFF000 ) >> 12); /* TMP5 for table search(8bit)            */
      tmp0.ul = ( src << 4 ); /* TMP5 for table search(8bit)              */
      tmp0.us[0] = ( tmp0.us[0] >> 1 ); /* TMP4 for approximate(15bit)              */
      tmp0.us[0] = ( tmp0.us[0] & 0x7FFF ); /* mask 15bit                        */
    }
    else if( Low & 0xF000 )
/*-----------------------------------------------------------------------------------------------------*/
/*    TMP8 8                                                */
```

```
/*    |0000|0000|0000|0000|xxxx|yyyy|aaaaaaaa|(0000000)                        */
/*-------------------------------------------------------------------------------------------------*/
    {
      uswk6 = 8;
      uswk5 = ( Low >> 8 ); /* TMP5 for table search (8bit)           */
      uswk4 = ( Low & 0x0FF );
      tmp0.us[0] = ( uswk4 << 7 );  /* TMP4 for approximate (15bit)             */
    }
    else if( Low & 0x0F00 )
/*-------------------------------------------------------------------------------------------------*/
/*    TMP8 10                                                 */
/*    |0000|0000|0000|0000|0000|xxxx|yyyyaaaa|(00000000000)                    */
/*-------------------------------------------------------------------------------------------------*/
    {
      uswk6 = 10;
      uswk5 = ( Low >> 4 ); /* TMP5 table search (8bit)             */
      uswk4 = ( Low & 0x00F );
      tmp0.us[0] = ( uswk4 << 11 ); /* TMP4 approximate (15bit)             */
    }
// |0000|0000|0000|0000|0000|0000|xxxxyyyy|(000000000000000)
    else
    {
      uswk6 = 12;
      IxTblSqrt16( (uswk0), Low );  /* TMP0 = table data                  */
    }
/*-------------------------------------------------------------------------------------------------*/
/*    table read and approximate                                 */
/*    TMP5(High), TMP4(Low)                                   */
/*-------------------------------------------------------------------------------------------------*/
    if( uswk6 < 12 )
    {
      IxTblSqrt16( (uswk3), uswk5 );  /* TMP3 <-- tbl[tmp]                */
      if( uswk5 == 0x00FF )
      {
        uswk0 = 0xFFFF; /* TMP0 <-- (tbl[tmp+1])                  */
      }
      else
```

```
      {
        uswk5 = uswk5 + 1;
        IxTblSqrt16( (uswk0), uswk5 );  /* TMP0 <-- tbl[tmp+1]                    */
      }
/*-------------------------------------------------------------------------------------------------*/
/*    (tbl[tmp+1] - tbl[tmp])*low/32768 + tbl[tmp]                                 */
/*---------------------------------- ----------------------------------------------------------*/
      uswk4 = uswk0 - uswk3;
//<022>     uswk1 = (USHORT)IlibASR32(( (LONG)uswk4 * (LONG)tmp0.us[0] ) , 15);
//<022>     uswk0 = uswk1 + uswk3;  /* TMP0 = read data                 */
      uswk0 = (USHORT)IlibASR32(( (LONG)uswk4 * (LONG)tmp0.us[0] ) , 15);
      uswk0 = uswk0 + uswk3;  /* TMP0 = read data                 */
    }
/*-------------------------------------------------------------------------------------------------*/
/*    Scaling                                                  */
/*-------------------------------------------------------------------------------------------------*/
//<022>   ulwk2 = (ULONG)(uswk0);
//<022>   ulwk0 = (ulwk2 >> uswk6);
    ulwk0 = ((ULONG)(uswk0) >> uswk6);
    return( (USHORT)ulwk0 );
}
#else
//<3> start
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 る 場 合 は切捨ての補正なし
    if( ulwk0 < ulwk2 ) {             // 切 捨 て 誤 差 が 平 方 根 の 結 果より大きい場 合 補 正
      uswk0 = uswk0 + 1;
    }
```

```
  }

  return ( uswk0 );
}
//<3> end
#endif

/*****************************************************************************************************/
/*                                                   */
/*    Over modulation compasation calculation                          */
/*                                                   */
/*****************************************************************************************************/
/*    INPUT:   TMP4: table address, IntAdV.V1:modulation                    */
/*    OUTPUT:  Kmod:   compensation gain/offset                        */
/*    work:    TMP0,TMP1,TMP2,TMP3                                */
/*****************************************************************************************************/
//void  MpOVMMODK( INTADP *IntAdP, INTADV *IntAdV, INTADWK *IntAdwk )
inline void MpOVMMODK( INTADP *IntAdP, INTADV *IntAdV, CSHORT*  pCtbl )   /* 2013.05.06 tanaka21 コ ー ド 整理<020>     */
{

  SHORT swk0;       /* 16bitワ ー ク レ ジスタ0    2013.05.06 tanaka21 コ ー ド 整理<020>    */
  SHORT swk1;       /* 16bitワ ー ク レ ジスタ1    2013.05.06 tanaka21 コ ー ド 整理<020>    */
  SHORT swk2;       /* 16bitワ ー ク レ ジスタ2    2013.05.06 tanaka21 コ ー ド 整理<020>    */
  SHORT swk3;       /* 16bitワ ー ク レ ジスタ3    2013.05.06 tanaka21 コ ー ド 整理<020>    */
        SHORT   swk4;   //<2>
  if( IntAdV->V1 < 9459 )
  {
//<2>   IxLoadMpmem16( IntAdP->Kmod, pCtbl, 0 );  /* IntAdP->Kmod = G[0];                    */
    IxLoadMpmem16( swk4, pCtbl, 0 );  /* IntAdP->Kmod = G[0];                    */
  }
  else if( (IntAdP->CtrlSw & OVMMOD) == 0 )
  {
    pCtbl = pCtbl + 15;
//<2>   IxLoadMpmem16( IntAdP->Kmod, pCtbl, 1 );
    IxLoadMpmem16( swk4, pCtbl, 1 );
  }
  else
```

```
  {
    if( IntAdV->V1 < 10431 )
    {
      swk0 = IntAdV->V1;
      swk0 = swk0 - 9443; /* -9439-5(margin)                        */
      swk1 = swk0;
      swk0 = swk0 >> 5; /* high                              */
      swk1 = swk1 & 0x1F; /* low                                 */
      if( swk0 >= 32 )
      {
        pCtbl = pCtbl + 15;
//<2>       IxLoadMpmem16( IntAdP->Kmod, pCtbl, 1 );
        IxLoadMpmem16( swk4, pCtbl, 1 );
      }
      else
      {
        swk2 = swk0;
        swk0 = swk0 >> 1;
        if( ( swk2 & 1 ) == 0 )
        {
          pCtbl = pCtbl + swk0;
          IxLoadMpmem16( swk2, pCtbl, 0 );
          IxLoadMpmem16( swk3, pCtbl, 1 );
        }
        else
        {
          pCtbl = pCtbl + swk0;
          IxLoadMpmem16( swk2, pCtbl, 1 );
          pCtbl = pCtbl + 1;
          IxLoadMpmem16( swk3, pCtbl, 0 );
        }
        swk0 = swk3 - swk2;
/* 2012.10.05 Y.Oka 変　換　前は% s h rなのでIli b A SR32では ？ */
//        swk0 = IlibASR16( swk0 * swk1, 5);
//<1>       swk0 = (SHORT)IlibASR32( (LONG)swk0 * (LONG)swk1, 5);
        swk0 = mulshr(swk0, swk1, 5);
/* 2012.10.05 Y.Oka 変　換　前は% s h rなのでIli b A SR32では ？ */
```

```
//<2>     IntAdP->Kmod = swk0 + swk2;
        swk4 = swk0 + swk2;
        }
    }
    else
    {
      pCtbl = pCtbl + 15;
//<2>     IxLoadMpmem16( IntAdP->Kmod, pCtbl, 1 );
        IxLoadMpmem16( swk4, pCtbl, 1 );
    }
  }
  IntAdP->Kmod = swk4;
  return;
}



#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              <V720>  */
/*                                                                                                   */
/*****************************************************************************************************/
//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.dl = DivPlsV.Argu0.l * DivPlsV.Kx.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.dl = DivPlsV.Argu0.l * DivPlsV.Kx.l;
      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)))     */
//#ifdef WIN32
      IPfbwk.lswk10 = (LONG)((INT64)IPfbwk.lswk2 >> IPfbwk.lswk1);  /* lswk10 = (xh>>(sx-24))             */
//#elif defined(ASIP_CC)
//      IPfbwk.lswk10 = asr( IPfbwk.lswk2, IPfbwk.lswk1); /* lswk10 = (xh>>(sx-24))             */
//#endif
/*-----------------------------------------------------------------------------------------------*/
      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 + ((((xl>>(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
```

```
//<2> start
/*****************************************************************************************************/
/*                                                   */
/*    処　理　高　速　化　用 ラ イブラリ                                          */
/*                                                   */
/*****************************************************************************************************/
/*                                                   */
/*    電　流　検　出　値 読み 取り                                              */
/*                                                   */
/*****************************************************************************************************/
inline void ADConvDataLoad( MICRO_AXIS_HANDLE *AxisRsc )
{
  SHORT swk;

/*-------------------------------------------------------------------------*/
/*    A/D convert data loading                       */
/*-------------------------------------------------------------------------*/
/*    IntAdV.IuInData = IntAdP.Kcu * ( IUS + IntAdV.IuOffset ) / 2^8      */
/*    IntAdV.IvInData = IntAdP.Kcv * ( IVS + IntAdV.IvOffset ) / 2^8      */
/*-------------------------------------------------------------------------*/
  swk = mulshr(IuAD, ONE, 2);
  AxisRsc->IntAdV.IuInData = mulshr((swk + AxisRsc->IntAdV.IuOffset), AxisRsc->IntAdP.Kcu, 8 );
/*-------------------------------------------------------------------------*/
  swk = mulshr(IvAD, ONE, 2);
  AxisRsc->IntAdV.IvInData = mulshr((swk + AxisRsc->IntAdV.IvOffset), AxisRsc->IntAdP.Kcv, 8 );
#ifdef  MULTI_AXIS
  AxisRsc++;
  swk = mulshr(IuAD_2, ONE, 2);
  AxisRsc->IntAdV.IuInData = mulshr((swk + AxisRsc->IntAdV.IuOffset), AxisRsc->IntAdP.Kcu, 8 );
/*-------------------------------------------------------------------------*/
  swk = mulshr(IvAD_2, ONE, 2);
  AxisRsc->IntAdV.IvInData = mulshr((swk + AxisRsc->IntAdV.IvOffset), AxisRsc->IntAdP.Kcv, 8 );
#endif

  return;
}
```

```
/*****************************************************************************************************/
/*                                                   */
/*    PWM出  力                                          */
/*                                                   */
/*****************************************************************************************************/
inline void SetPWM( MICRO_AXIS_HANDLE *AxisRsc )    /* <S015> */
{
  PwmT2 = AxisRsc->PwmV.PwmCntT2;
  PwmT1 = AxisRsc->PwmV.PwmCntT1;
  PwmT0 = AxisRsc->PwmV.PwmCntT0;

#ifdef  MULTI_AXIS              /* 多  軸  処  理有効                    */
  AxisRsc++;
  PwmT2_2 = AxisRsc->PwmV.PwmCntT2;
  PwmT1_2 = AxisRsc->PwmV.PwmCntT1;
  PwmT0_2 = AxisRsc->PwmV.PwmCntT0;
#endif /* MULTI_AXIS */           /* 多  軸  処  理有効                    */
}

//<2> end


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