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

/* 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
extern int chess_storage(PFREG:0x6D9) HSUR0;  //<2>
extern int chess_storage(PFREG:0x6DA) HSUR1;  //<2>
extern int chess_storage(PFREG:0x6D0) IuAD;   //<2>
extern int chess_storage(PFREG:0x6D1) IvAD;    //<2>
extern int chess_storage(PFREG:0x6D0) OUTPT;   //<2>
extern int chess_storage(PFREG:0x6D1) WDT1L;   //<2>
extern int chess_storage(PFREG:0x6DD) CTSTR;   //<2>
extern int chess_storage(PFREG:0x6DD) CTSTW;   //<2>
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:0x7D0) IuAD_2; //<2>
extern int chess_storage(PFREG:0x7D1) IvAD_2; //<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>

#define USE_CMOVE //<2> t-yamada

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

INITWK  IniWk;/* for dubug */
DBGWORKS DebugWk; /* for debug */
COMWORKS ComWk; /* for debug */

//#define MULTI_AXIS            /* 多軸処理有効                   */
#ifdef  MULTI_AXIS            /* 多軸処理有効                   */
#define MAX_AXIS_NUM  2       /* 最大制御軸数                   */
#endif    //#ifdef  MULTI_AXIS
/****************************************************************************************************/


/****************************************************************************************************/
/*    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  MpIntAD( void );
void  MpIntEnc( void );
//USHORT  MpSQRT( INTADWK *IntAdwk, ULONG src );
//USHORT  MpSQRT( ULONG src ) clobbers(IH);   /* 2013.05.06 tanaka21 コード整理<0 20>    */
inline USHORT MpSQRT( ULONG src );    /* 2013.05.06 tanaka21 コード整理<020>     */
//void  MpOVMMODK( INTADP *IntAdP, INTADV *IntAdV, INTADWK *IntAdwk );
//void  MpOVMMODK( INTADP *IntAdP, INTADV *IntAdV, CSHORT*  pCtbl ) clobbers(IH);   /* 2013.05.06 tanaka21 コード整理<0 20>
*/
inline void MpOVMMODK( INTADP *IntAdP, INTADV *IntAdV, CSHORT*  pCtbl );    /* 2013.05.06 tanaka21 コード整理<020>     */
inline void ADConvDataLoad( INTADV *IntAdV, INTADP *IntAdP ); //<2>
inline void SetPWM( SHORT src0, SHORT src1, SHORT src2 );   //<2>

#if defined(WIN32)    /* VC用ダミーレジスタ定義   */
SVIP_READ_REG SvIpReadReg;
SVIP_WRITE_REG  SvIpWriteReg;
#endif

#if 0/* ローカル変数定義不具合によりグローバル化 --> 不具合修正によりローカルに復帰、コメントアウ
MICRO_AXIS_HANDLE *AxisRscR;      /* Initial & Round      */
MICRO_AXIS_HANDLE *AxisRscI;      /* IntAD                */
MICRO_AXIS_HANDLE *AxisRscH;      /* IntHost              */
USHORT      ax_noR;      /* Initial & Round      */
USHORT      ax_noI;      /* IntAD                */
USHORT      ax_noH;      /* IntHost              */
#endif

/* 機能レジスタ/周辺レジスタ(0x5F0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;

//  IHOSTWK     IHostWk;  /* ホスト割込みワーク   2013.05.04 tanaka21 コード整理<019>    *//* コメントアウト
  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                                   */
/*--------------------------------------------------------------------------------------------------*/
#if 0 /* ★ 追加必要★ */
  if( 取得軸数 <= MAX_AXIS_NUM )
  {
    AxisNum = 取得軸数;
  }
  else
  {
    AxisNum = MAX_AXIS_NUM;
  }
#else
  /* 暫定処置 */
  AxisInfo.AxisNum = 1;
#endif

/*--------------------------------------------------------------------------------------------------*/
/*    Set H/W Register Address Pointer                              */
/*--------------------------------------------------------------------------------------------------*/
#ifdef  MULTI_AXIS              /* 多軸処理有効                     */
  for( ax_noR = 0; (SHORT)ax_noR < AxisInfo.AxisNum; ax_noR++ )
#else   //#ifdef  MULTI_AXIS
  ax_noR = 0;
#endif    //#ifdef  MULTI_AXIS
  {
    AxisRscR = &(AxisHdl[ax_noR]);
#if defined(WIN32)
    AxisRscR->SvIpRegR = &SvIpReadReg;
    AxisRscR->SvIpRegW = &SvIpWriteReg;
#else   //#if defined(WIN32)
#if defined( FREG_DEF ) || defined( PREG_DEF )
    AxisRscR->SvIpRegR = (SVIP_READ_REG*)(0x600);
    AxisRscR->SvIpRegW = (SVIP_WRITE_REG*)(0x600);
#else //#if defined( FREG_DEF ) || defined( PREG_DEF )
    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);
    }
#endif  //#if defined( FREG_DEF ) || defined( PREG_DEF )
#endif  //#if defined(WIN32)
  }

/*------------------------------------------------------------------------------------------------*/
/*    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 < AxisInfo.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        */
#ifdef  PREG_DEF
    BBSET = AxisRscR->StsFlg.BbSetW;  /* INT1=Encoder0, BB        */
```

```
#else   //#ifdef  PREG_DEF
    AxisRscR->SvIpRegW->BBSET = AxisRscR->StsFlg.BbSetW;  /* INT1=Encoder0, BB        */
#endif    //#ifdef  PREG_DEF

#ifdef  FREG_DEF
    ISA0 = (int)MpIntAD;
//    ISA1 = (int)MpIntEnc;
#else //#ifdef  FREG_DEF
    AxisRscR->SvIpRegW->ISA0 = (INT)MpIntAD;
//    AxisRscR->SvIpRegW->ISA1 = (INT)MpIntEnc; /* JL-086で実行するため外しておく    */
#endif  //#ifdef  FREG_DEF
/*--------------------------------------------------------------------------------------------------*/
#ifdef  PREG_DEF
    PCVS0 = AxisRscR->EncIfV.DivPls.s[0]; /* パルス変換位置      (bit15-0)     */
#else   //#ifdef  PREG_DEF
    AxisRscR->SvIpRegW->PCVS0 = AxisRscR->EncIfV.DivPls.s[0]; /* パルス変換位置      (bit15-0)     */
#endif    //#ifdef  PREG_DEF
/*--------------------------------------------------------------------------------------------------*/
    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    */
#ifdef  PREG_DEF
    PCVS1 = PoSet1W;  /* パルス変換原点補正1 (bit15-0)             */
    PCVS2 = PoSet2W;  /* パルス変換原点補正2 (bit15-0)             */
#else   //#ifdef  PREG_DEF
    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)             */
#endif    //#ifdef  PREG_DEF
/*--------------------------------------------------------------------------------------------------*/
    DivSetW = AxisRscR->DivPlsV.DivSetIn; /* MpUPDATE_DIVPOS()で比較処理があるため残しておく    */
#ifdef  PREG_DEF
    DIVSET = DivSetW; /* 分周機能設定                      */
#else   //#ifdef  PREG_DEF
    AxisRscR->SvIpRegW->DIVSET = DivSetW; /* 分周機能設定                      */
#endif    //#ifdef  PREG_DEF

/*--------------------------------------------------------------------------------------------------*/
//110914tanaka21 0,1,-1は定数マクロ化するため初期化処理不要
```

```
///*-------------------------------------------------------------------------------------------------*/
///*    Power on reset Register(定数レジスタ初期化)                                              */
///*-------------------------------------------------------------------------------------------------*/
//    ZEROR = 0x00000000; /* ZeroR,ZERORH <-- 0              */
//    ONER = 0x00000001;  /* OneR,ONERH   <--  1              <V720>  */
//    NONER = 0xffffffff; /* NOneR,NONERH <-- -1              <V720>  */
/*-------------------------------------------------------------------------------------------------*/
/* 2013.05.06 tanaka21 コード整理 (マクロ化) <022>      */
//    /* 2012.12.21 Y.Oka 現状初期化必要 */
//    ZEROR = 0x00000000;
//    ONER = 0x00000001;
//    NONER = 0xffffffff;
    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                                       */
/*-------------------------------------------------------------------------------------------------*/
#ifdef  PREG_DEF
    PWMOS = 0x0A0;              /* 2level,triangle,servo(bit7: no-Saw mode for JL-056)    */
#else   //#ifdef  PREG_DEF
    AxisRscR->SvIpRegW->PWMOS = 0x0A0;              /* 2level,triangle,servo(bit7: no-Saw mode for JL-056)    */
#endif    //#ifdef  PREG_DEF
    AxisRscR->IntAdV.CrFreqW = AxisRscR->IntAdP.CrFreq;      /* Carrier set(IntAdP.CrFreq must be set before starts)   */
#ifdef  PREG_DEF
    CRSET1 = 0x10;              /* CLA=Both(unavailable on JL-056)              */
    CRFRQ = AxisRscR->IntAdV.CrFreqW;   /* Carrier 6kHz                         */
#else   //#ifdef  PREG_DEF
    AxisRscR->SvIpRegW->CRSET1 = 0x10;              /* CLA=Both(unavailable on JL-056)              */
    AxisRscR->SvIpRegW->CRFRQ = AxisRscR->IntAdV.CrFreqW;   /* Carrier 6kHz                       */
#endif    //#ifdef  PREG_DEF
    uswk = (AxisRscR->IntAdV.CrFreqW >> 1); /* TMP0 <-- IntAdV.CrFreqW /2(50p duty)           */

#ifdef  PREG_DEF
    PwmT2 = uswk;     /* T2(W) = (duty:50p)                   */
    PwmT1 = uswk;     /* T1(V) = (duty:50p)                   */
    PwmT0 = uswk;     /* T0(U) = (duty:50p)                   */
#else   //#ifdef  PREG_DEF
//<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);
#endif    //#ifdef  PREG_DEF
/*--------------------------------------------------------------------------------------------------*/
/*    Clear Register                                        */
/*--------------------------------------------------------------------------------------------------*/
    MpDataClear( AxisRscR );
/*--------------------------------------------------------------------------------------------------*/
/*    input CPORT, DLIM = QLIM = 0, output CPORT                          */
/*--------------------------------------------------------------------------------------------------*/
//<2>#ifdef PREG_DEF
#ifndef PREG_DEF
    AxisRscR->StsFlg.CtrlStsRW = CTSTR; /* StsFlg.CtrlStsRW <- Control register               */
    AxisRscR->StsFlg.CtrlStsRW = ( AxisRscR->StsFlg.CtrlStsRW & DLIMI );  /* StsFlg.CtrlStsRW <-- StsFlg.CtrlStsRW & DLIMI
    (imm_16)            */
    CTSTW = AxisRscR->StsFlg.CtrlStsRW; /* Status Set                       */
    DebugWk.CTSTR = AxisRscR->StsFlg.CtrlStsRW;

#else   //#ifdef  PREG_DEF
    AxisRscR->StsFlg.CtrlStsRW = AxisRscR->SvIpRegR->CTSTR; /* StsFlg.CtrlStsRW <- Control register               */
    DebugWk.CTSTR = AxisRscR->StsFlg.CtrlStsRW;
    AxisRscR->StsFlg.CtrlStsRW = ( AxisRscR->StsFlg.CtrlStsRW & DLIMI );  /* StsFlg.CtrlStsRW <-- StsFlg.CtrlStsRW & DLIMI
    (imm_16)            */
    AxisRscR->SvIpRegW->CTSTW = AxisRscR->StsFlg.CtrlStsRW; /* Status Set                       */
#endif    //#ifdef  PREG_DEF
/*--------------------------------------------------------------------------------------------------*/
```

```
/*    START : INTERRUPT, PWM                                      */
/*--------------------------------------------------------------------------------------------------*/
#ifdef  FREG_DEF
    EIX = 0x0;              /* Interuput start                       */
#else   //#ifdef  FREG_DEF
    AxisRscR->SvIpRegW->EIX = 0x0;              /* Interuput start                       */
#endif    //#ifdef  FREG_DEF

#ifdef  PREG_DEF
    CRST = 0x1;             /* Carrier(PWM) start                    */
    AxisRscR->StsFlg.BbSetW = ( AxisRscR->StsFlg.BbSetW & 0xFFFB ); /* Reset soft_BB                      */
    BBSET = AxisRscR->StsFlg.BbSetW;    /*                              */
#else   //#ifdef  PREG_DEF
    AxisRscR->SvIpRegW->CRST = 0x1;             /* Carrier(PWM) start                    */
    AxisRscR->StsFlg.BbSetW = ( AxisRscR->StsFlg.BbSetW & 0xFFFB ); /* Reset soft_BB                      */
    AxisRscR->SvIpRegW->BBSET = AxisRscR->StsFlg.BbSetW;    /*                              */
#endif    //#ifdef  PREG_DEF
  }

/****************************************************************************************************/
/*                                                  */
/*    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 < AxisInfo.AxisNum; ax_noR++ )
#else   //#ifdef  MULTI_AXIS
    ax_noR = 0;
#endif    //#ifdef  MULTI_AXIS
    {
      AxisRscR = &AxisHdl[ax_noR];
/*--------------------------------------------------------------------------------------------------*/
/*    A/D error check and clear                                 */
/*--------------------------------------------------------------------------------------------------*/
#ifdef  PREG_DEF
      AxisRscR->StsFlg.FccStsMon = FCCST;
      AxisRscR->StsFlg.FltStsW = FLTSTAT & 0x7FFF;
#else   //#ifdef  PREG_DEF
      AxisRscR->StsFlg.FccStsMon = AxisRscR->SvIpRegR->FCCST;
      DebugWk.FCCST = AxisRscR->SvIpRegR->FCCST;
      AxisRscR->StsFlg.FltStsW = AxisRscR->SvIpRegR->FLTSTAT & 0x7FFF;
      DebugWk.FLTSTAT = AxisRscR->SvIpRegR->FLTSTAT;
#endif    //#ifdef  PREG_DEF
//      AxisRscR->StsFlg.FltStsW = ( AxisRscR->StsFlg.FltStsW & 0x7FFF );

//for chessde, 20121115
//      if ( AxisRscR->StsFlg.FltStsW != 0x0 )
//      {
//      //---------------------------
//      // insert error sequence
//      //---------------------------
//      }
    }

/*--------------------------------------------------------------------------------------------------*/
/*    Host port check for host INT                                */
/*      現在、WREG100～WREG104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
#ifndef PREG_DEF
    if ( HSUR0 != 0x0 )
    {
      MpIntHost( );       /*                              */
    }
#else   //#ifdef  PREG_DEF
    if ( AxisHdl[0].SvIpRegR->HSUR0 != 0x0 )
    {
      MpIntHost( );       /*                              */
    }
#endif    //#ifdef  PREG_DEF

/*----------------------------------------------------------------------------------------------------*/
/*    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
#ifndef PREG_DEF
    if ( HSUR1 != 0x0 )
#else   //#ifdef  PREG_DEF
    if ( AxisHdl[0].SvIpRegR->HSUR1 != 0x0 )
#endif    //#ifdef  PREG_DEF
    {
#ifdef  FREG_DEF
      DIX = 0x0;    /* disable interupt   <V112>                */
#else   //#ifdef  FREG_DEF
      AxisHdl[0].SvIpRegW->DIX = 0x0;   /* disable interupt   <V112>                */
#endif    //#ifdef  FREG_DEF

#ifdef  MULTI_AXIS              /* 多軸処理有効                        */
      for( ax_noR = 0; (SHORT)ax_noR < AxisInfo.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FB比例ゲイン(下位16bit)<V214>          */
        AxisRscR->WeakFV.WfKpV.s[1] = AxisRscR->WeakFV.WfKpVHIn;    /* 電圧FB比例ゲイン(上位16bit)<V214>        */
        AxisRscR->WeakFV.WfKiV.s[0] = AxisRscR->WeakFV.WfKiVLIn;    /* 電圧FB積分ゲイン(下位16bit)<V214>          */
        AxisRscR->WeakFV.WfKiV.s[1] = AxisRscR->WeakFV.WfKiVHIn;    /* 電圧FB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>      */
      }

#ifdef  FREG_DEF
      EIX = 0x0;    /* enable interupt    <V112>                */
#else   //#ifdef  FREG_DEF
      AxisHdl[0].SvIpRegW->EIX = 0x0;   /* enable interupt    <V112>                */
#endif    //#ifdef  FREG_DEF
    }
    DebugWk.HSUR1 = AxisHdl[0].SvIpRegR->HSUR1;
    ComWk.WREG82 = AxisHdl[0].SvIpRegR->CRFRQI;
    ComWk.WREG83 = AxisHdl[0].IntAdV.CrFreqW;
    ComWk.WREG87 = AxisRscR->SvIpRegR->IuAD;
    ComWk.WREG88 = AxisRscR->SvIpRegR->IvAD;
  }
  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;
```

```
//  IHOSTWK     IHostWk;  /* ホスト割込みワーク  2013.05.04 tanaka21 コード整理<019>   *//* コメントアウト
  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★    */
//<2>#ifdef PREG_DEF
#ifndef PREG_DEF
  WDT1L = 0x1;    /* Watch dog set                     */
  OUTPT = 0x1;    /* 1.13                          */
#else   //#ifdef  PREG_DEF
  AxisHdl[0].SvIpRegW->WDT1L = 0x1;   /* Watch dog set                     */
  AxisHdl[0].SvIpRegW->OUTPT = 0x1;   /* 1.13                          */
#endif    //#ifdef  PREG_DEF


#ifdef  DEBUG_OUTPT
  AxisHdl[0].SvIpRegW->OUTPT = 0x00;    /* for check progress */
#endif  //#ifdef  DEBUG_OUTPT

#ifdef  MULTI_AXIS              /* 多軸処理有効                    */
  for( ax_noH = 0; (SHORT)ax_noH < AxisInfo.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                  */
#ifdef  PREG_DEF
      CRFRQ = AxisRscH->IntAdV.CrFreqW; /* Carrier Freq. Change                    */
#else   //#ifdef  PREG_DEF
      AxisRscH->SvIpRegW->CRFRQ = AxisRscH->IntAdV.CrFreqW; /* Carrier Freq. Change                   */
#endif    //#ifdef  PREG_DEF
    }
  }

/*----------------------------------------------------------------------------------------------*/
/*    input from host                                       */
/*----------------------------------------------------------------------------------------------*/
//    swk0 = CTSTR; /* HTMP5 <-- CTSTR                    */

  /* Check Current Ajust Request */
  ActiveAxis = 0;
#ifdef  MULTI_AXIS              /* 多  軸  処  理有効                     */
  for( ax_noH = 0; (SHORT)ax_noH < AxisInfo.AxisNum; ax_noH++ )
#else   //#ifdef  MULTI_AXIS
  ax_noH = 0;
#endif    //#ifdef  MULTI_AXIS
  {
    AxisRscH = &AxisHdl[ax_noH];
//<2>#ifdef PREG_DEF
#ifndef PREG_DEF
    if ( ( CTSTR & RLOCK ) == 0 )
    {
      ActiveAxis |= 0x01 << ax_noH; /* ビ  ッ  ト 登録 */
    }
    DebugWk.CTSTR = AxisRscH->SvIpRegR->CTSTR;
#else   //#ifdef  PREG_DEF
    if ( ( AxisRscH->SvIpRegR->CTSTR & RLOCK ) == 0 )
    {
      ActiveAxis |= 0x01 << ax_noH; /* ビ  ッ  ト 登録 */
    }
    DebugWk.CTSTR = AxisRscH->SvIpRegR->CTSTR;
#endif    //#ifdef  PREG_DEF
  }


#ifdef  DEBUG_OUTPT
  AxisHdl[0].SvIpRegW->OUTPT = 0x01;    /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscH->IntAdP.Kcu;
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscH->IntAdP.Kcv;
#endif  //#ifdef  DEBUG_OUTPT

  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★    */
#ifdef  FREG_DEF
    DIX = 0x0;    /* disable interupt   <V112>                */
#else   //#ifdef  FREG_DEF
    AxisHdl[0].SvIpRegW->DIX = 0x0;   /* disable interupt   <V112>                */
#endif    //#ifdef  FREG_DEF

#ifdef  MULTI_AXIS              /* 多軸処理有効                         */
    for( ax_noH = 0; (SHORT)ax_noH < AxisInfo.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★    */
#ifdef  FREG_DEF
    EIX = 0x0;    /* enable interupt    <V112>                */
#else   //#ifdef  FREG_DEF
    AxisHdl[0].SvIpRegW->EIX = 0x0;   /* enable interupt    <V112>                */
#endif    //#ifdef  FREG_DEF
  }
/*----------------------------------------------------------------------------------------------------*/
/*    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★    */
//<2>#ifdef PREG_DEF
#ifndef PREG_DEF
  OUTPT = 0x0;    /* <H>                          */
#else   //#ifdef  PREG_DEF
  AxisHdl[0].SvIpRegW->OUTPT = 0x0;   /* <H>                          */
#endif    //#ifdef  PREG_DEF

  /* ★ 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
  DIX = 0x0;    /* disable interupt   <V112>                */
#else   //#ifdef  FREG_DEF
  AxisHdl[0].SvIpRegW->DIX = 0x0;   /* disable interupt   <V112>                */
#endif    //#ifdef  FREG_DEF

#ifdef  MULTI_AXIS              /* 多軸処理有効                      */
  for( ax_noH = 0; (SHORT)ax_noH < AxisInfo.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FB比例ゲイン(下位16bit) <V214>          */
    AxisRscH->WeakFV.WfKpV.s[1] = AxisRscH->WeakFV.WfKpVHIn;    /* 電圧FB比例ゲイン(上位16bit) <V214>        */
    AxisRscH->WeakFV.WfKiV.s[0] = AxisRscH->WeakFV.WfKiVLIn;    /* 電圧FB積分ゲイン(下位16bit) <V214>          */
    AxisRscH->WeakFV.WfKiV.s[1] = AxisRscH->WeakFV.WfKiVHIn;    /* 電圧FB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★    */
#ifdef  FREG_DEF
  EIX = 0x0;    /* enable interupt    <V112>                */
#else   //#ifdef  FREG_DEF
  AxisHdl[0].SvIpRegW->EIX = 0x0;   /* enable interupt    <V112>                */
#endif    //#ifdef  FREG_DEF

#ifdef  DEBUG_OUTPT
  AxisHdl[0].SvIpRegW->OUTPT = 0x02;    /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscH->AdinV.TLimPIn;
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscH->AdinV.TLimMIn;
#endif  //#ifdef  DEBUG_OUTPT


/*--------------------------------------------------------------------------------------------------*/
/*    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 < AxisInfo.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_WKOH++;    /* 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 < AxisInfo.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)*/
#ifdef  PREG_DEF
        SDMECLR = ( FCCST | 8 );
#else   //#ifdef  PREG_DEF
        AxisRscH->SvIpRegW->SDMECLR = ( AxisRscH->SvIpRegR->FCCST | 8 );
        DebugWk.FCCST = AxisRscH->SvIpRegR->FCCST;
#endif    //#ifdef  PREG_DEF
        AxisRscH->AdStop.ADRst = AxisRscH->IntAdP.FccRst;
        AxisRscH->IntAdP.FccRst = 0;
      }
    }
#ifdef  PREG_DEF
    ADSYNC = 1;
#else   //#ifdef  PREG_DEF
```

```
    AxisRscH->SvIpRegW->ADSYNC = 1;
#endif    //#ifdef  PREG_DEF
  }

/*--------------------------------------------------------------------------------------------------*/
//    swk0 = CTSTR; /* HTMP5 <-- control register             */
  /* Check BB Status */
  ActiveAxis = 0;
#ifdef  MULTI_AXIS             /* 多 軸 処 理有効                     */
  for( ax_noH = 0; (SHORT)ax_noH < AxisInfo.AxisNum; ax_noH++ )
#else   //#ifdef  MULTI_AXIS
  ax_noH = 0;
#endif    //#ifdef  MULTI_AXIS
  {
    AxisRscH = &AxisHdl[ax_noH];
//<2>#ifdef PREG_DEF
#ifndef PREG_DEF
    if ( CTSTR & BB )
    {
      ActiveAxis |= 0x01 << ax_noH; /* ビ ッ ト 登録 */
    }
    DebugWk.CTSTR = AxisRscH->SvIpRegR->CTSTR;
#else   //#ifdef  PREG_DEF
    if ( AxisRscH->SvIpRegR->CTSTR & BB )
    {
      ActiveAxis |= 0x01 << ax_noH; /* ビ ッ ト 登録 */
    }
    DebugWk.CTSTR = AxisRscH->SvIpRegR->CTSTR;
#endif    //#ifdef  PREG_DEF
  }

  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★    */
#ifdef  FREG_DEF
    DIX = 0x0;    /* disable interupt   <V112>               */
#else   //#ifdef  FREG_DEF
```

```
    AxisHdl[0].SvIpRegW->DIX = 0x0;   /* disable interupt   <V112>                */
#endif    //#ifdef  FREG_DEF

#ifdef  MULTI_AXIS              /* 多軸処理有効                        */
    for( ax_noH = 0; (SHORT)ax_noH < AxisInfo.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                 */
#ifdef  PREG_DEF
          CRFRQ = AxisRscH->IntAdV.CrFreqW; /* Carrier Freq. Change                   */
#else   //#ifdef  PREG_DEF
          AxisRscH->SvIpRegW->CRFRQ = AxisRscH->IntAdV.CrFreqW; /* Carrier Freq. Change                 */
#endif    //#ifdef  PREG_DEF
        }
      }
    }

    /* ★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
    EIX = 0x0;    /* enable interupt    <V112>                */
#else   //#ifdef  FREG_DEF
    AxisHdl[0].SvIpRegW->EIX = 0x0;   /* enable interupt    <V112>                */
#endif    //#ifdef  FREG_DEF
  }
```

```
#ifdef  DEBUG_OUTPT
  AxisHdl[0].SvIpRegW->OUTPT = 0x03;    /* for check progress */
#endif  //#ifdef  DEBUG_OUTPT

#ifdef  MULTI_AXIS              /* 多軸処理有効                          */
  for( ax_noH = 0; (SHORT)ax_noH < AxisInfo.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];


#ifdef  DEBUG_OUTPT
      AxisHdl[0].SvIpRegW->OUTPT = 0x04;  /* for Micro Debug */
#endif  //#ifdef  DEBUG_OUTPT

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

```
#ifdef  WIN32   /* 加 減 演 算 リ ミ ッ ト判 別用処理(VC用)   */
        IxADDSUBLMTCHKRDY( (LONG)AxisRscH->IntAdP.Kf12 * (LONG)AxisRscH->AdinV.IqIn, (LONG)AxisRscH->IntAdP.Kf11 *
        AxisRscH->IntAdV.IqIn1PL.l );
        lwk1 = ((LONG)AxisRscH->IntAdP.Kf12 * (LONG)AxisRscH->AdinV.IqIn) + ((LONG)AxisRscH->IntAdP.Kf11 *
        AxisRscH->IntAdV.IqIn1PL.l);
        IxADDLMTCHK( lwk1 )

        IxADDSUBLMTCHKRDY( lwk1, (LONG)AxisRscH->IntAdP.Kf14 * AxisRscH->IntAdV.IqIn1PPL.l );
        lwk1 = lwk1 + ((LONG)AxisRscH->IntAdP.Kf14 * AxisRscH->IntAdV.IqIn1PPL.l);
        IxADDLMTCHK( lwk1 )
#else
//<1>       lwk1 = (((LONG)AxisRscH->IntAdP.Kf12 * (LONG)AxisRscH->AdinV.IqIn)
//<1>                   + ((LONG)AxisRscH->IntAdP.Kf11 * AxisRscH->IntAdV.IqIn1PL.l)
//<1>                   + ((LONG)AxisRscH->IntAdP.Kf14 * AxisRscH->IntAdV.IqIn1PPL.l));
        lwk1 = mul(AxisRscH->IntAdP.Kf12, AxisRscH->AdinV.IqIn);
        lwk1 = mac((LONG)AxisRscH->IntAdP.Kf11, AxisRscH->IntAdV.IqIn1PL.l, lwk1);
#endif
//<1>       lwk1 = IxLmtCBS32( lwk1 );      /* 符 号 付32b i t 制 限処理             */
        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)               */
/*-----------------------------------------------------------------------------------------------------*/
//#ifdef WIN32
//       lwk2 = (LONG)(( (INT64)AxisRscH->IntAdP.Kf11 * (INT64)AxisRscH->IntAdV.IqOut1PL.l ) >> 13); /* ゲ イ ン 乗 算 後 整数化
*/
//#elif defined(ASIP_CC)
//       lwk2 = asr( AxisRscH->IntAdP.Kf11 * AxisRscH->IntAdV.IqOut1PL.l, 13); /* ゲ イ ン 乗 算 後 整数化        */
//#endif
//       lwk2 = (LONG)IlibASR64(( (INT64)AxisRscH->IntAdP.Kf11 * (INT64)AxisRscH->IntAdV.IqOut1PL.l ) , 13); /*
ゲ イ ン 乗 算 後整数化         */
//<1>       dlwk = mul((LONG)AxisRscH->IntAdP.Kf11, AxisRscH->IntAdV.IqOut1PL.l);   /* AxisRscH->IntAdP.Kf11 *
AxisRscH->IntAdV.IqOut1PL.l   */
//<1>       lwk2 = (LONG)IlibASR64(dlwk , 13);  /* ゲ イ ン 乗 算 後整数化         */
//<1>       lwk2 = IxLmtCBS32( lwk2 );      /* <V502> 追 加                */
        lwk2 = mulshr_limitf((LONG)AxisRscH->IntAdP.Kf11, AxisRscH->IntAdV.IqOut1PL.l, 13);
```

```
//#ifdef WIN32
//        lwk3 = (LONG)(( (INT64)AxisRscH->IntAdP.Kf13 * (INT64)AxisRscH->IntAdV.IqOut1PPL.l ) >> 13);  /* ゲ  イ  ン  乗  算  後整数化
*/
//#elif defined(ASIP_CC)
//        lwk3 = asr( AxisRscH->IntAdP.Kf13 * AxisRscH->IntAdV.IqOut1PPL.l, 13);  /* ゲ  イ  ン  乗  算  後整数化         */
//#endif
//        lwk3 = (LONG)IlibASR64(( (INT64)AxisRscH->IntAdP.Kf13 * (INT64)AxisRscH->IntAdV.IqOut1PPL.l ) , 13);  /*
ゲ  イ  ン  乗  算 後整数化          */
//<1>       dlwk = mul((LONG)AxisRscH->IntAdP.Kf13, AxisRscH->IntAdV.IqOut1PPL.l);    /* AxisRscH->IntAdP.Kf13 *
AxisRscH->IntAdV.IqOut1PPL.l    */
//<1>       lwk3 = (LONG)IlibASR64(dlwk , 13);  /* ゲ  イ  ン  乗  算  後整数化         */
//<1>       lwk3 = IxLmtCBS32( lwk3 );      /* <V502> 追  加                   */
        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             */

//<1><4>        AxisRscH->IntAdV.IqOut1L.l = AxisRscH->IntAdV.IqOut1BufL.l >> 13;
//<1>       AxisRscH->IntAdV.IqOut1L.s[0] = IxLmtCBS16( AxisRscH->IntAdV.IqOut1L.s[0] );  /*        <V502> 追  加       */
        AxisRscH->IntAdV.IqOut1L.s[0] = asr_limitf( AxisRscH->IntAdV.IqOut1BufL.l, 13 );  /*        <V502> 追  加       */

      }

#ifdef  DEBUG_OUTPT
  AxisHdl[0].SvIpRegW->OUTPT = 0x05;    /* for check progress */
#endif  //#ifdef  DEBUG_OUTPT
```

```
/****************************************************************************************************/
/*    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                     */
/*------------------------------------------------------------------------------------------------*/
#ifdef  WIN32
        IxADDSUBLMTCHKRDY( (LONG)AxisRscH->IntAdP.Kf32 * (LONG)AxisRscH->IntAdV.IqOut1L.s[0], (LONG)AxisRscH->IntAdP.Kf31 *
        AxisRscH->IntAdV.IqIn3PL.l );
        lwk1 = (LONG)AxisRscH->IntAdP.Kf32 * (LONG)AxisRscH->IntAdV.IqOut1L.s[0]
                + (LONG)AxisRscH->IntAdP.Kf31 * AxisRscH->IntAdV.IqIn3PL.l;
        IxADDLMTCHK( lwk1 );

        IxADDSUBLMTCHKRDY( lwk1, (LONG)AxisRscH->IntAdP.Kf34 + AxisRscH->IntAdV.IqIn3PPL.l );
        lwk1 = lwk1
                + (LONG)AxisRscH->IntAdP.Kf34 + AxisRscH->IntAdV.IqIn3PPL.l;
        IxADDLMTCHK( lwk1 );
#else
//<1>       lwk1 = (((LONG)AxisRscH->IntAdP.Kf32 * (LONG)AxisRscH->IntAdV.IqOut1L.s[0])
//<1>                   + ((LONG)AxisRscH->IntAdP.Kf31 * AxisRscH->IntAdV.IqIn3PL.l)
//<1><4>                  + ((LONG)AxisRscH->IntAdP.Kf34 + AxisRscH->IntAdV.IqIn3PPL.l));
        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);
#endif
//<1>       lwk1 = IxLmtCBS32( lwk1 );        /* 32bit制 限                   */
```

```
/*---------------------------------------------------------------------------------------------------*/
/*    HTMP0  = HTMP0 - (IntAdP.Kf31 * IQO3P + IntAdP.Kf33 * IQO3PP)                        */
/*---------------------------------------------------------------------------------------------------*/
//#ifdef WIN32
//        lwk2 = (LONG)(( (INT64)AxisRscH->IntAdP.Kf31 * (INT64)AxisRscH->IntAdV.IqOut3PL.l ) >> 13); /* ゲイン乗算後整数化
*/
//#elif defined(ASIP_CC)
//        lwk2 = asr( AxisRscH->IntAdP.Kf31 * AxisRscH->IntAdV.IqOut3PL.l, 13); /* ゲイン乗算後整数化        */
//#endif
//        lwk2 = (LONG)IlibASR64(( (INT64)AxisRscH->IntAdP.Kf31 * (INT64)AxisRscH->IntAdV.IqOut3PL.l ) , 13); /*
ゲイン乗算後整数化        */
//<1>       dlwk = mul((LONG)AxisRscH->IntAdP.Kf31, AxisRscH->IntAdV.IqOut3PL.l);   /* AxisRscH->IntAdP.Kf31 *
AxisRscH->IntAdV.IqOut3PL.l   */
//<1>       lwk2 = (LONG)IlibASR64(dlwk , 13);  /* ゲイン乗算後整数化        */
//<1>       lwk2 = IxLmtCBS32( lwk2 );          /* 桁あふれ確認              */
        lwk2 = mulshr_limitf((LONG)AxisRscH->IntAdP.Kf31, AxisRscH->IntAdV.IqOut3PL.l, 13);

//#ifdef WIN32
//        lwk3 = (LONG)(( (INT64)AxisRscH->IntAdP.Kf33 * (INT64)AxisRscH->IntAdV.IqOut3PPL.l ) >> 13);  /* ゲイン乗算後整数化
*/
//#elif defined(ASIP_CC)
//        lwk3 = asr( AxisRscH->IntAdP.Kf33 * AxisRscH->IntAdV.IqOut3PPL.l, 13);  /* ゲイン乗算後整数化        */
//#endif
//        lwk3 = (LONG)IlibASR64(( (INT64)AxisRscH->IntAdP.Kf33 * (INT64)AxisRscH->IntAdV.IqOut3PPL.l ) , 13);  /*
ゲイン乗算後整数化        */
//<1>       dlwk = mul((LONG)AxisRscH->IntAdP.Kf33, AxisRscH->IntAdV.IqOut3PPL.l);    /* AxisRscH->IntAdP.Kf33 *
AxisRscH->IntAdV.IqOut3PPL.l   */
//<1>       lwk3 = (LONG)IlibASR64( dlwk , 13 );  /* ゲイン乗算後整数化        */
//<1>       lwk3 = IxLmtCBS32( lwk3 );          /* 桁あふれ確認              */
        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              */

//<1><4>        AxisRscH->IntAdV.IqOut3L.l = lwk1 >> 13;      /* 出力値の整数化*/
//<1>       AxisRscH->IntAdV.IqOut3L.s[0] = IxLmtCBS16( AxisRscH->IntAdV.IqOut3L.s[0] );  /*        <V502> 追加        */
        AxisRscH->IntAdV.IqOut3L.s[0] = asr_limitf(lwk1, 13);

      }

#ifdef  DEBUG_OUTPT
  AxisHdl[0].SvIpRegW->OUTPT = 0x06;    /* for check progress */
#endif  //#ifdef  DEBUG_OUTPT

/****************************************************************************************************/
/*    Low Pass Filter                                                 */
/****************************************************************************************************/
/*    IntAdP.TLpf    : Time-constant                                          */
/*    IntAdV.IqOut1Lpf : Output(32 bit) .. IQ01F: 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   */

#ifdef  WIN32
        IxADDSUBLMTCHKRDY( AxisRscH->IntAdV.IqOut3, AxisRscH->IntAdV.IqOut1Lpf.s[1] );
#endif
//<1>       swk1 = AxisRscH->IntAdV.IqOut3 - AxisRscH->IntAdV.IqOut1Lpf.s[1]; /* HTMP0 <-- IntAdV.IqOut3 - IQ01FH          */
#ifdef  WIN32
        IxSUBLMTCHK( swk1 );
#endif
```

```
//<1>       swk1 = IxLmtCBS16( swk1 );  /* HTMP0 <-- limit(HTMP0, 2^15 - 1)           */
        swk1 = sub_limitf(AxisRscH->IntAdV.IqOut3, AxisRscH->IntAdV.IqOut1Lpf.s[1]);

//<1>       lwk2 = ((LONG)AxisRscH->IntAdP.TLpf * (LONG)swk1) << 2;
        lwk2 = mul(AxisRscH->IntAdP.TLpf, swk1) << 2;

#ifdef  WIN32
        IxADDSUBLMTCHKRDY( lwk2, AxisRscH->IntAdV.IqOut1Lpf.l );
#endif
//<1>       lwk2 = lwk2 + AxisRscH->IntAdV.IqOut1Lpf.l;
#ifdef  WIN32
        IxADDLMTCHK( lwk2 );
#endif
//<1>       AxisRscH->IntAdV.IqOut1Lpf.l = IxLmtCBS32( lwk2 );  /* HTMP0 <-- limit(HTMP0, 2^15 - 1)           */
        AxisRscH->IntAdV.IqOut1Lpf.l = add_limitf(lwk2, AxisRscH->IntAdV.IqOut1Lpf.l);  /* HTMP0 <-- limit(HTMP0, 2^15 - 1)
        */
      }

#ifdef  DEBUG_OUTPT
  AxisHdl[0].SvIpRegW->OUTPT = 0x07;    /* for check progress */
#endif  //#ifdef  DEBUG_OUTPT

/****************************************************************************************************/
/*    notch filter (before data input)                                */
/****************************************************************************************************/
/*    input    : IQ01F (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
      {
/*--------------------------------------------------------------------------------------------------*/
```

```
/*    HTMP0 = IntAdP.Kf22 * IQO1F + IntAdP.Kf21 * IQI2P + IntAdP.Kf24 * IQI2PP                      */
/*------------------------------------------------------------------------------------------------*/
#ifdef  WIN32
        IxADDSUBLMTCHKRDY( (LONG)AxisRscH->IntAdP.Kf22 * (LONG)AxisRscH->IntAdV.IqOut1Lpf.s[1], (LONG)AxisRscH->IntAdP.Kf21 *
        AxisRscH->IntAdV.IqOut2PL.l );
        lwk1 = (LONG)AxisRscH->IntAdP.Kf22 * (LONG)AxisRscH->IntAdV.IqOut1Lpf.s[1]
                + (LONG)AxisRscH->IntAdP.Kf21 * AxisRscH->IntAdV.IqOut2PL.l;
        IxADDLMTCHK( lwk1 );

        IxADDSUBLMTCHKRDY( lwk1, (LONG)AxisRscH->IntAdP.Kf34 + AxisRscH->IntAdV.IqIn2PPL.l );
        lwk1 = lwk1
                + (LONG)AxisRscH->IntAdP.Kf34 + AxisRscH->IntAdV.IqIn2PPL.l;
        IxADDLMTCHK( lwk1 );
#else
//<1>       lwk1 = (((LONG)AxisRscH->IntAdP.Kf22 * (LONG)AxisRscH->IntAdV.IqOut1Lpf.s[1])
//<1>                   + ((LONG)AxisRscH->IntAdP.Kf21 * AxisRscH->IntAdV.IqOut2PL.l)
//<1><4>                  + ((LONG)AxisRscH->IntAdP.Kf34 + AxisRscH->IntAdV.IqIn2PPL.l));
        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);
#endif
//<1>       lwk1 = IxLmtCBS32( lwk1 );        /* 32bit制限                    */

/*------------------------------------------------------------------------------------------------*/
/*    HTMP0  = HTMP0 - (IntAdP.Kf21 * IQOP + IntAdP.Kf23 * IQOPH)                       */
/*------------------------------------------------------------------------------------------------*/
//#ifdef WIN32
//        lwk2 = (LONG)(( (INT64)AxisRscH->IntAdP.Kf21 * (INT64)AxisRscH->IntAdV.IqOut2PL.l ) >> 13); /* ゲイン乗算後整数化
*/
//#elif defined(ASIP_CC)
//        lwk2 = asr( AxisRscH->IntAdP.Kf21 * AxisRscH->IntAdV.IqOut2PL.l, 13); /* ゲイン乗算後整数化          */
//#endif
//        lwk2 = (LONG)IlibASR64(( (INT64)AxisRscH->IntAdP.Kf21 * (INT64)AxisRscH->IntAdV.IqOut2PL.l ) , 13); /*
ゲイン乗算後整数化          */
//<1>       dlwk = mul((LONG)AxisRscH->IntAdP.Kf21, AxisRscH->IntAdV.IqOut2PL.l);   /* AxisRscH->IntAdP.Kf21 *
AxisRscH->IntAdV.IqOut2PL.l   */
//<1>       lwk2 = (LONG)IlibASR64( dlwk , 13 );  /* ゲイン乗算後整数化         */
```

```
//<1>		lwk2 = IxLmtCBS32( lwk2 );		/* 桁あふれ確認			*/
	lwk2 = mulshr_limitf((LONG)AxisRscH->IntAdP.Kf21, AxisRscH->IntAdV.IqOut2PL.l, 13);

//#ifdef WIN32
//		lwk3 = (LONG)(( (INT64)AxisRscH->IntAdP.Kf23 * (INT64)AxisRscH->IntAdV.IqOut2PPL.l ) >> 13);	/* ゲイン乗算後整数化
*/
//#elif defined(ASIP_CC)
//		lwk3 = asr( AxisRscH->IntAdP.Kf23 * AxisRscH->IntAdV.IqOut2PPL.l, 13);	/* ゲイン乗算後整数化		*/
//#endif
//		lwk3 = (LONG)IlibASR64(( (INT64)AxisRscH->IntAdP.Kf23 * (INT64)AxisRscH->IntAdV.IqOut2PPL.l ) , 13);	/*
ゲイン乗算後整数化		*/
//<1>		dlwk = mul((LONG)AxisRscH->IntAdP.Kf23, AxisRscH->IntAdV.IqOut2PPL.l);	/* AxisRscH->IntAdP.Kf23 *
AxisRscH->IntAdV.IqOut2PPL.l	*/
//<1>		lwk3 = (LONG)IlibASR64( dlwk , 13); /* ゲイン乗算後整数化		*/
//<1>		lwk3 = IxLmtCBS32( lwk3 );		/* 桁あふれ確認			*/
	lwk3 = mulshr_limitf((LONG)AxisRscH->IntAdP.Kf23, AxisRscH->IntAdV.IqOut2PPL.l, 13);

	lwk1 = lwk1 - lwk2 - lwk3;

/*--------------------------------------------------------------------------------------------------*/
/*	IQI2PP = IQI2P, IQI2P = IQO1F, IQO2PP = IQO2P, IQO2P = HTMP0				*/
/*--------------------------------------------------------------------------------------------------*/
	AxisRscH->IntAdV.IqIn2PPL.l = AxisRscH->IntAdV.IqIn2PL.l;	/* 前々回値保存				*/
	AxisRscH->IntAdV.IqIn2PL.l = (LONG)AxisRscH->IntAdV.IqOut1Lpf.s[0]; /* 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		*/

//<1>		AxisRscH->IntAdV.IqOut2L.l = lwk1 >> 13;	/* 出力値の整数化*/
//<1>		AxisRscH->IntAdV.IqOut2L.s[0] = IxLmtCBS16( AxisRscH->IntAdV.IqOut2L.s[0] );	/*	<V502> 追加	*/
	AxisRscH->IntAdV.IqOut2L.s[0] = asr_limitf(lwk1, 13);

      }
    }

#ifdef	DEBUG_OUTPT
/* for debug */
```

```
    else
    {
      AxisHdl[0].SvIpRegW->OUTPT = 0xff;  /* for Micro Debug */
    }
/* for debug */


  AxisHdl[0].SvIpRegW->OUTPT = 0x08;    /* for check progress */
#endif  //#ifdef  DEBUG_OUTPT

/*--------------------------------------------------------------------------------------------------*/
/*    omega calculation                                     */
/*--------------------------------------------------------------------------------------------------*/
//<1>   swk0 = (SHORT)((( (LONG)AxisRscH->IntAdP.Ld * (LONG)AxisRscH->WeakFV.Vel) >> 15) * AxisRscH->IntAdV.KEangle);
//<1>   swk0 = IxLmtCBS16( swk0 );
    swk0 = mulshr(AxisRscH->IntAdP.Ld, AxisRscH->WeakFV.Vel, 15);
    lwk1 = mul(swk0, AxisRscH->IntAdV.KEangle);
    swk0 = asr_limitf( lwk1, 0 );

//<1>   swk1 = (SHORT)((( (LONG)AxisRscH->IntAdP.Lq * (LONG)AxisRscH->WeakFV.Vel) >> 15) * AxisRscH->IntAdV.KEangle);
//<1>   swk1 = IxLmtCBS16( swk1 );
    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★    */
#ifdef  FREG_DEF
  DIX = 0x0;    /* disable interupt   <V112>                */
#else   //#ifdef  FREG_DEF
  AxisHdl[0].SvIpRegW->DIX = 0x0;   /* disable interupt   <V112>                */
#endif    //#ifdef  FREG_DEF
```

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

//<1>   AxisRscH->VcmpV.MagC = (SHORT)(( (LONG)AxisRscH->IntAdP.Mag * (LONG)AxisRscH->WeakFV.Vel) >> 15);   /* VcmpV.MagC <--
ACC >> 15                   */
    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                            */

#ifdef  DEBUG_OUTPT
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscH->VcmpV.MagC;    /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscH->VcmpV.LdC;   /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscH->VcmpV.LqC;   /* for check progress */
#endif  //#ifdef  DEBUG_OUTPT

/*--------------------------------------------------------------------------------------------------*/
    AxisRscH->WeakFV.IqOut = AxisRscH->IntAdV.IqOut2L.s[0]; /* <V388> 追  加                      */
//    swk0 = IntAdP.CtrlSw; /*                              */

    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
#ifndef PREG_DEF
    AxisRscH->StsFlg.CtrlStsRW = CTSTR; /* StsFlg.CtrlStsRW <- Control register              */
    DebugWk.CTSTR = AxisRscH->SvIpRegR->CTSTR;
#else   //#ifdef  PREG_DEF
    AxisRscH->StsFlg.CtrlStsRW = AxisRscH->SvIpRegR->CTSTR; /* StsFlg.CtrlStsRW <- Control register              */
#endif    //#ifdef  PREG_DEF
    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★    */
#ifdef  FREG_DEF
  EIX = 0x0;    /* enable interupt    <V112>                */
#else   //#ifdef  FREG_DEF
  AxisHdl[0].SvIpRegW->EIX = 0x0;    /* enable interupt    <V112>                */
#endif    //#ifdef  FREG_DEF


#ifdef  DEBUG_OUTPT
  AxisHdl[0].SvIpRegW->OUTPT = 0x09;  /* for Micro Debug */
#endif  //#ifdef  DEBUG_OUTPT

  return;
}




//INTADWK     IntAdwk;  /* 電流割込みワーク 2013.05.04 tanaka21 コード整理<019>     *//* コメントアウト<02
```

```
/****************************************************************************************************/
/*                                                    */
/*    AD Interupt Procedure                                      */
/*                                                    */
/*    マイクロ分周機能にてエンコーダ割込(@INT_ENC)追加のため割込レベル(INTLVWR )マスク処理変更    */
/****************************************************************************************************/
void  MpIntAD( void ) property(isr)
//void  MpIntAD( 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_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>

  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) */
#ifdef  FREG_DEF
  INTLVWR &= 0x00F0;
#else   //#ifdef  FREG_DEF
  AxisHdl[0].SvIpRegW->INTLVWR &= 0x00F0;
#endif    //#ifdef  FREG_DEF

//<2>#ifdef PREG_DEF
#ifndef PREG_DEF
  OUTPT = 0x1;
  WDT1L = 0x0;    /* Watch dog reset */
#else   //#ifdef  PREG_DEF
  AxisHdl[0].SvIpRegW->OUTPT = 0x1;
  AxisHdl[0].SvIpRegW->WDT1L = 0x0;   /* Watch dog reset */
#endif    //#ifdef  PREG_DEF

#ifdef  DEBUG_OUTPT
  AxisHdl[0].SvIpRegW->OUTPT = 0x10;    /* for check progress */
#endif  //#ifdef  DEBUG_OUTPT


  /* Get Current Feedback Data from A/D */
#ifdef  MULTI_AXIS              /* 多  軸  処  理有効                    */
  for( ax_noI = 0; (SHORT)ax_noI < AxisInfo.AxisNum; ax_noI++ )
#else   //#ifdef  MULTI_AXIS
  ax_noI = 0;
#endif    //#ifdef  MULTI_AXIS
  {
    AxisRscI = &AxisHdl[ax_noI];
/*--------------------------------------------------------------------------------------------------*/
/*    A/D convert data loading                                  */
/*--------------------------------------------------------------------------------------------------*/
/*    IntAdV.IuInData = IntAdP.Kcu * ( IUS + IntAdV.IuOffset ) / 2^8                              */
/*    IntAdV.IvInData = IntAdP.Kcv * ( IVS + IntAdV.IvOffset ) / 2^8                              */
/*----------------------------------------------------------------------------------------------------*/
//    IntAdV.IuInData = ( ( (IuAD >> 2) + IntAdV.IuOffset ) * IntAdP.Kcu ) >> 8;
#if 0 //<2>
#ifdef  PREG_DEF
    swk0 = (SHORT)IlibASR32( IuAD, 2);
#else   //#ifdef  PREG_DEF
//<1><4>    swk0 = (SHORT)IlibASR32(AxisRscI->SvIpRegR->IuAD, 2);
    swk0 = mulshr(AxisRscI->SvIpRegR->IuAD, ONE, 2);
#endif    //#ifdef  PREG_DEF
//<1>   AxisRscI->IntAdV.IuInData = (SHORT)IlibASR32( (LONG)(swk0 + AxisRscI->IntAdV.IuOffset) * (LONG)AxisRscI->IntAdP.Kcu, 8
);
    AxisRscI->IntAdV.IuInData = mulshr((swk0 + AxisRscI->IntAdV.IuOffset), AxisRscI->IntAdP.Kcu, 8 );
/*----------------------------------------------------------------------------------------------------*/
//    IntAdV.IvInData = ( ( (IvAD >> 2) + IntAdV.IvOffset ) * IntAdP.Kcv ) >> 8;
#ifdef  PREG_DEF
    swk0 = (SHORT)IlibASR32( IvAD, 2);
#else   //#ifdef  PREG_DEF
//<1><4>    swk0 = (SHORT)IlibASR32((LONG)AxisRscI->SvIpRegR->IvAD, 2);
    swk0 = mulshr(AxisRscI->SvIpRegR->IvAD, ONE, 2);
#endif    //#ifdef  PREG_DEF
//<1>   AxisRscI->IntAdV.IvInData = (SHORT)IlibASR32( (LONG)(swk0 + AxisRscI->IntAdV.IvOffset) * (LONG)AxisRscI->IntAdP.Kcv, 8
);
    AxisRscI->IntAdV.IvInData = mulshr((swk0 + AxisRscI->IntAdV.IvOffset), AxisRscI->IntAdP.Kcv, 8 );
#else //<2>
    ADConvDataLoad(&AxisRscI->IntAdV, &AxisRscI->IntAdP);
    DebugWk.IuAD = AxisRscI->SvIpRegR->IuAD;
#endif  //<2>
  }

  /* Execute Current Loop Main Operation */
#ifdef  MULTI_AXIS              /* 多  軸  処  理有効                     */
  for( ax_noI = 0; (SHORT)ax_noI < AxisInfo.AxisNum; ax_noI++ )
#else   //#ifdef  MULTI_AXIS
  ax_noI = 0;
#endif    //#ifdef  MULTI_AXIS
```

```
  {
    AxisRscI = &AxisHdl[ax_noI];
//===================================================================
// 位相補間処理     <V112>
//===================================================================
#ifndef USE_CMOVE
    if( AxisRscI->PhaseV.PhaseIpF != 1 )
    {
      /* フラグをセット */
      AxisRscI->PhaseV.PhaseIpF = 1;
    }
    else
    {
      /* 位相に位相補間値を足し込む */
      AxisRscI->PhaseV.PhaseH = AxisRscI->PhaseV.PhaseH + AxisRscI->PhaseV.PhaseIp;
    }
#else   //<2>
                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);
#endif  //<2>
//===================================================================
// 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 */
//<1>   swk2 = (SHORT)IlibASR32(( (LONG)swk1 * (LONG)AxisRscI->IntAdV.IuInData ) , 14 );
    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 */
//<1>   swk1 = (SHORT)IlibASR32(( (LONG)swk1 * (LONG)AxisRscI->IntAdV.IvInData ) , 14 );
    swk1 = mulshr(swk1, AxisRscI->IntAdV.IvInData, 14 );
    /* TMP2 <-- TMP2 + TMP1 */
    swk2 = swk1 + swk2;
    /* ACC <-- IntAdP.Kc * TMP2 */
```

```
//<1>   AxisRscI->IntAdV.IdInData = (SHORT)IlibASR32(( (LONG)AxisRscI->IntAdP.Kc * (LONG)swk2 ) , 9 );
    AxisRscI->IntAdV.IdInData = mulshr(AxisRscI->IntAdP.Kc, swk2, 9 );
/*--------------------------------------------------------------------------------------------------*/
    swk1 = AxisRscI->SinTbl.SinT2 - AxisRscI->SinTbl.SinT;                        /* TMP1 <-- sin(th+2pi/3) - sin(th)
    */
//<1>   swk2 = (SHORT)IlibASR32(( (LONG)swk1 * (LONG)AxisRscI->IntAdV.IuInData ) , 14 );  /* ACC <-- TMP1 * iu
*/
    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)
    */
//<1>   swk1 = (SHORT)IlibASR32(( (LONG)swk1 * (LONG)AxisRscI->IntAdV.IvInData ) , 14 );  /* ACC <-- TMP1 * iv
*/
    swk1 = mulshr(swk1, AxisRscI->IntAdV.IvInData, 14 );  /* ACC <-- TMP1 * iv                   */
    swk2 = swk1 + swk2;                       /* TMP2 <-- TMP2 + TMP1             */
//<1>   AxisRscI->IntAdV.IqInData = (SHORT)IlibASR32(( (LONG)AxisRscI->IntAdP.Kc * (LONG)swk2 ) , 9 );    /* ACC <-- IntAdP.Kc
* TMP2                    */
    AxisRscI->IntAdV.IqInData = mulshr(AxisRscI->IntAdP.Kc, swk2, 9 );    /* ACC <-- IntAdP.Kc * TMP2                  */

#ifdef  DEBUG_OUTPT
  AxisHdl[0].SvIpRegW->OUTPT = 0x11;    /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->PhaseV.PhaseH;
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->IntAdV.IuInData;
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->IntAdP.Kc;
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->IntAdV.IdInData;
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->IntAdV.IqInData;
#endif  //#ifdef  DEBUG_OUTPT
/*--------------------------------------------------------------------------------------------------*/
/*    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 )
      {
//<2>       swk3 = 0;   /* TMP4 = 0 ( OverFlowCheck = OK )               */
        swk3 = ZERO;    /* TMP4 = 0 ( OverFlowCheck = OK )               */
      }
      else
      {
//<2>       swk3 = 1;   /* TMP4 = 1 ( OverFlowCheck = NG )               */
        swk3 = ONE;   /* TMP4 = 1 ( OverFlowCheck = NG )               */
      }
//========================================
//  d軸オブザーバ部
//========================================
//<1>     swk0 = (SHORT)IlibASR32(( (LONG)AxisRscI->DobsP.TsPerL * (LONG)AxisRscI->VcmpV.VdOut ) , 15 );    /* TMP0 <-- ACC >>
15    ( TMP0 = Ts/L * Vd_out >> 15 )  */
      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>            */
//<1>     if( swk2 > 15000 )
//<1>     {
//<1>       swk2 = 15000;
//<1>     }
```

```
//<1>     else if( swk2 < (-15000) )
//<1>     {
//<1>       swk2 = -15000;
//<1>     }
      swk2 = limit(swk2, 15000);
      swk1 = swk2 - AxisRscI->DobsV.IdObsOut; /*                <V076>      */
//<1>     swk1 = (SHORT)IlibASR32(( (LONG)AxisRscI->DobsP.Gobs * (LONG)swk1 ) , 16 ); /* ACC  <-- TMP2*DobsP.Gobs   ( TMP2 = g
* ( Id - Id_obs ) ) */
      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) )  */
//<1>     swk1 = (SHORT)IlibASR32(( (LONG)AxisRscI->DobsP.RLTs * (LONG)AxisRscI->DobsV.IdObsOut ) , 12 ); /* TMP2 <--
DobsV.IqObsOut      ( TMP2 = Id_obs )       */
      swk1 = mulshr(AxisRscI->DobsP.RLTs, AxisRscI->DobsV.IdObsOut, 12 ); /* TMP2 <-- DobsV.IqObsOut      ( TMP2 = Id_obs )
      */
#ifdef  WIN32
      IxADDSUBLMTCHKRDY( swk1, swk0 );
#endif
//<1>     AxisRscI->DobsV.IdObsOut = swk1 + swk0; /* DobsV.IdObsOut <-- TMP0 + TMP2   ( TMP2 = Id_obs[k+1] )  */
#ifdef  WIN32
      IxADDLMTCHK( AxisRscI->DobsV.IdObsOut );
#endif
//<1>     AxisRscI->DobsV.IdObsOut = IxLmtCBS16( AxisRscI->DobsV.IdObsOut );  /* DobsV.IdObsOut <-- limit( DobsV.IdObsOut,
2^15-1 )          */
      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
//----------------------------------------
#ifdef  WIN32
      IxADDSUBLMTCHKRDY( swk0, AxisRscI->DobsV.LpfIld.s[1] );
#endif
```

```
//<1>     swk0 = swk0 - AxisRscI->DobsV.LpfIld.s[1];  /*                      */
#ifdef  WIN32
      IxSUBLMTCHK( swk0 );
#endif
//<1>     swk0 = IxLmtCBS16( swk0 );  /*                            */
      swk0 = sub_limitf(swk0, AxisRscI->DobsV.LpfIld.s[1]);
//<1>     lwk2 = ((LONG)AxisRscI->DobsP.FilObsGain * (LONG)swk0 ) << 2; /*                            */
      lwk2 = mul(AxisRscI->DobsP.FilObsGain, swk0 ) << 2; /*                            */
#ifdef  WIN32
      IxADDSUBLMTCHKRDY( lwk2, AxisRscI->DobsV.LpfIld.l );
#endif
//<4>     lwk2 = lwk2 + AxisRscI->DobsV.LpfIld.l; /*                        */
//<4>     dlwk = mul( (LONG)AxisRscI->DobsP.FilObsGain, (LONG)swk0 );
//<4>     lwk2 = (LONG)IlibASR64( dlwk, 2 );  /*                            */

#ifdef  WIN32
      IxADDLMTCHK( lwk2 );
#endif
//<4>     AxisRscI->DobsV.LpfIld.l = IxLmtCBS32( lwk2 );  /*                            */
      AxisRscI->DobsV.LpfIld.l = add_limitf(lwk2, AxisRscI->DobsV.LpfIld.l);
//--------------------------------------
//  high pass filter
//--------------------------------------
#ifdef  WIN32
      IxADDSUBLMTCHKRDY( AxisRscI->DobsV.LpfIld.s[1], AxisRscI->DobsV.HpfIld.s[1] );
#endif
//<1>     swk0 = AxisRscI->DobsV.LpfIld.s[1] - AxisRscI->DobsV.HpfIld.s[1]; /*                    */
#ifdef  WIN32
      IxSUBLMTCHK( swk0 );
#endif
//<1>     swk0 = IxLmtCBS16( swk0 );  /*                            */
      swk0 = sub_limitf(AxisRscI->DobsV.LpfIld.s[1], AxisRscI->DobsV.HpfIld.s[1]);
//<1>     dlwk = mul( (LONG)AxisRscI->DobsP.FilObsGain, (LONG)swk0 ); /*                            */
//<1>     lwk2 = (LONG)( dlwk << 2 ); /*                            */
//<4>     lwk2 = mul(AxisRscI->DobsP.FilObsGain, swk0 );  /*                            */
      lwk2 = mul(AxisRscI->DobsP.FilObsGain, swk0 ) << 2; /*                            */
```

```
#ifdef  WIN32
      IxADDSUBLMTCHKRDY( lwk2, AxisRscI->DobsV.HpfIld.l );
#endif
//<1>     lwk2 = lwk2 + AxisRscI->DobsV.HpfIld.l; /*                          */
#ifdef  WIN32
      IxADDLMTCHK( lwk2 );
#endif
//<1>     AxisRscI->DobsV.HpfIld.l = IxLmtCBS32( lwk2 );  /*                            */
      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
//--------------------------------------
//<2>     AxisRscI->DobsV.IdObsFreq = AxisRscI->DobsV.IdObsFreq * AxisRscI->DobsV.DmpGain;    /* ACC  <-- DobsV.IdObsFreq *
DobsV.DmpGain                 */
      AxisRscI->DobsV.IdObsFreq = AxisRscI->DobsV.IdObsFreq * 2;    /* ACC  <-- DobsV.IdObsFreq * DobsV.DmpGain
      */
#ifndef USE_CMOVE //<2>
//#if 1   //err
      if( swk3 != 0 )
      {
        AxisRscI->DobsV.IdObsFreq = 0;    /* DobsV.IdObsFreqを 0 と  する       */
      }
#else //<2>
      AxisRscI->DobsV.IdObsFreq = cmove((swk3 != 0), ZERO, AxisRscI->DobsV.IdObsFreq);
#endif  //<2>
      AxisRscI->IntAdV.IdInData = AxisRscI->IntAdV.IdInData - AxisRscI->DobsV.IdObsFreq;  /*                            */
//=====================================
//  q軸  オ  ブ  ザ ーバ 部
//======================================
//<1>     swk0 = (SHORT)IlibASR32(( (LONG)AxisRscI->DobsP.TsPerL * (LONG)AxisRscI->VcmpV.VqOut ) , 15 );  /* ACC  <--
TMP0*Ts/L    ( TMP0 = Ts/L * Vq_out)     */
      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>
      */
//<1>     swk2 = IxLIMITUL(swk2, 15000, -15000 );                       /* TMP3 <-- Limit(15000)  <V076>            */
```

```
      swk2 = limit(swk2, 15000);                      /* TMP3 <-- Limit(15000)  <V076>            */
      swk1 = swk2 - AxisRscI->DobsV.IqObsOut;                     /*              <V076>        */
//<1>     swk1 = (SHORT)IlibASR32( (LONG)AxisRscI->DobsP.Gobs * (LONG)swk1 , 16 );    /* TMP2 <-- ACC >> 16    ( TMP2 = g * (
Iq - Iq_obs ) >> 16 ) */
      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) )
      */
//<1>     swk1 = (SHORT)IlibASR32( (LONG)AxisRscI->DobsP.RLTs * (LONG)AxisRscI->DobsV.IqObsOut , 12 );    /* TMP2 <-- ACC >>
12    ( TMP2 = (1-R*Ts/L)*Iq_obs >> 12 )  */
      swk1 = mulshr(AxisRscI->DobsP.RLTs, AxisRscI->DobsV.IqObsOut, 12 );   /* TMP2 <-- ACC >> 12   ( TMP2 = (1-R*Ts/L)*Iq_obs
      >> 12 ) */
#ifdef  WIN32
      IxADDSUBLMTCHKRDY( swk1, swk0 );
#endif
//<1>     AxisRscI->DobsV.IqObsOut = swk1 + swk0;                   /* DobsV.IqObsOut <-- TMP0 + TMP2   ( TMP2 = Iq_obs[k+1]
) */
#ifdef  WIN32
      IxADDLMTCHK( AxisRscI->DobsV.IqObsOut );
#endif
//<1>     AxisRscI->DobsV.IqObsOut = IxLmtCBS16( AxisRscI->DobsV.IqObsOut );                          /* DobsV.IqObsOut <--
limit( DobsV.IqObsOut, 2^15-1 )           */
      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
//----------------------------------------
#ifdef  WIN32
      IxADDSUBLMTCHKRDY( swk0, AxisRscI->DobsV.LpfIlq.s[1] );
#endif
//<1>     swk0 = swk0 - AxisRscI->DobsV.LpfIlq.s[1];  /*                       */
```

```
#ifdef  WIN32
      IxSUBLMTCHK( swk0 );
#endif
//<1>            swk0 = IxLmtCBS16( swk0 ); /*                            */
            swk0 = sub_limitf(swk0, AxisRscI->DobsV.LpfIlq.s[1]); /*                            */
//<1>     lwk2 = ( (LONG)AxisRscI->DobsP.FilObsGain * (LONG)swk0) << 2; /*                            */
      lwk2 = mul(AxisRscI->DobsP.FilObsGain, swk0) << 2;  /*                            */
#ifdef  WIN32
      IxADDSUBLMTCHKRDY( lwk2, AxisRscI->DobsV.LpfIlq.l );
#endif
//<1>     lwk2 = lwk2 + AxisRscI->DobsV.LpfIlq.l; /*                        */
#ifdef  WIN32
      IxADDLMTCHK( lwk2 );
#endif
//<1>     AxisRscI->DobsV.LpfIlq.l = IxLmtCBS32( lwk2 );  /*                            */
      AxisRscI->DobsV.LpfIlq.l = add_limitf(lwk2, AxisRscI->DobsV.LpfIlq.l);  /*                            */
//--------------------------------------
//  high pass filter
//--------------------------------------
#ifdef  WIN32
      IxADDSUBLMTCHKRDY( AxisRscI->DobsV.LpfIlq.s[1], AxisRscI->DobsV.HpfIlq.s[1] );
#endif
//<1>     swk0 = AxisRscI->DobsV.LpfIlq.s[1] - AxisRscI->DobsV.HpfIlq.s[1]; /*                    */
#ifdef  WIN32
      IxSUBLMTCHK( swk0 );
#endif
//<1>     swk0 = IxLmtCBS16( swk0 );  /*                            */
      swk0 = sub_limitf(AxisRscI->DobsV.LpfIlq.s[1], AxisRscI->DobsV.HpfIlq.s[1]);  /*                            */
//<1>     lwk2 = ( (LONG)AxisRscI->DobsP.FilObsGain * (LONG)swk0) << 2; /*                            */
      lwk2 = mul(AxisRscI->DobsP.FilObsGain, swk0) << 2;  /*                            */
#ifdef  WIN32
      IxADDSUBLMTCHKRDY( lwk2, AxisRscI->DobsV.HpfIlq.l );
#endif
//<1>     lwk2 = lwk2 + AxisRscI->DobsV.HpfIlq.l; /*                        */
#ifdef  WIN32
      IxADDLMTCHK( lwk2 );
#endif
```

```
//<1>     AxisRscI->DobsV.HpfIlq.l = IxLmtCBS32( lwk2 );  /*                            */
      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
//----------------------------------------
//<2>     AxisRscI->DobsV.IqObsFreq = AxisRscI->DobsV.IqObsFreq * AxisRscI->DobsV.DmpGain;    /* ACC  <-- DobsV.IqObsFreq *
DobsV.DmpGain                 */
      AxisRscI->DobsV.IqObsFreq = AxisRscI->DobsV.IqObsFreq * 2;    /* ACC  <-- DobsV.IqObsFreq * DobsV.DmpGain
      */
#ifndef USE_CMOVE //<2>
      if( swk3 != 0 )
      {
        AxisRscI->DobsV.IqObsFreq = 0;    /* DobsV.IdObsFreqを 0 と  する       */
      }
#else //<2>
      AxisRscI->DobsV.IqObsFreq = cmove((swk3 != 0), ZERO, AxisRscI->DobsV.IqObsFreq);
#endif  //<2>
      AxisRscI->IntAdV.IqInData = AxisRscI->IntAdV.IqInData - AxisRscI->DobsV.IqObsFreq;  /*                            */
    }

#ifdef  DEBUG_OUTPT
  AxisHdl[0].SvIpRegW->OUTPT = 0x12;    /* for check progress */
#endif  //#ifdef  DEBUG_OUTPT

/*----------------------------------------------------------------------------------------------------*///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 確認
/*----------------------------------------------------------------------------------------------------*/
    if( AxisRscI->AdStop.ADRst != 0 )
    {
      AxisRscI->AdStop.ADRst = 0;
      swk6 = AxisRscI->IntAdV.CrFreqW >> 1;
      swk5 = swk6;
      swk4 = swk6;
/*----------------------------------------------------------------------------------------------------*/
    }
    /* 2012.12.20 Y.Oka 誤り修正 */
//    else if( AxisRscI->StsFlg.CtrlStsRW == BB )
    else if( (AxisRscI->StsFlg.CtrlStsRW & BB) != 0 )
    {
/*-------------------------------------------------------------------------------------------------*/
      swk6 = AxisRscI->IntAdV.CrFreqW >> 1;
      swk5 = swk6;
      swk4 = 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-Vd^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> */
#ifdef  WIN32
        IxADDSUBLMTCHKRDY( lwk2, lwk4 );
#endif
//<1>       lwk2 = lwk2 - lwk4; /* IntAdP.Vmax^2 - Vd^2               */
#ifdef  WIN32
        IxSUBLMTCHK( lwk2 );
```

```
#endif
//<1>       lwk2 = IxLmtCBS32( lwk2 );  /*                            */
        lwk2 = sub_limitf(lwk2, lwk4);
//<4>       lwk2 = IxLIMITUL( lwk2, LPX_REG32_MAX, LPX_REG32_MIN ); /* if (IntAdP.Vmax^2 - Vd^2)< 0, then (IntAdP.Vmax^2 -
Vd^2) = 0 */
        lwk2 = limitz( lwk2, LPX_REG32_MAX ); /* if (IntAdP.Vmax^2 - Vd^2)< 0, then (IntAdP.Vmax^2 - Vd^2) = 0  */
//        swk0 = MpSQRT( &IntAdwk, lwk2 );          /* √ ( IntAdP.Vmax^2 - Vd^2 )                    */
        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|                */
        }
//<1>       swk0 = AxisRscI->WeakFV.WfVqMax - swk1; /* TMP0 = Vqmax - Vq = Δ Vq          */
//<1>       swk0 = IxLmtCBS16( swk0 );  /*                            */
        swk0 = sub_limitf(AxisRscI->WeakFV.WfVqMax, swk1);
/*--------------------------------------------------------------------------------------------------*/
/*    比  例  項 計 算                                        */
/*--------------------------------------------------------------------------------------------------*/
        lwk1 = (LONG)swk0;  /* TMP1,0 = 符  号  拡張(TMP0)                 */
//<1>       dlwk = mul( lwk1, AxisRscI->WeakFV.WfKpV.l );
//<1>       swk2 = (SHORT)IlibASR64( dlwk , 32 );
        swk2 = (SHORT)mulshr( lwk1 ,AxisRscI->WeakFV.WfKpV.l, 32 );
//<4>       if( swk2 > 0 )
        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
        {
//<1>         dlwk = mul( lwk1, AxisRscI->WeakFV.WfKpV.l );
//<1>         lwk2 = (LONG)IlibASR64( dlwk , 16 );
//<4>         lwk2 = mulshr( lwk1, AxisRscI->WeakFV.WfKpV.l, 16 );
//<4>         swk2 = (SHORT)IlibASR32(( lwk2 * 256 ) , 16 );
          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                     */
//<1>       dlwk = mul( lwk1, AxisRscI->WeakFV.WfKiV.l );
//<1>       lwk6 = (LONG)IlibASR64( dlwk , 32 );  /* 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)         */
        }
#ifdef  WIN32
        IxADDSUBLMTCHKRDY( lwk4, AxisRscI->WeakFV.WfIntgl.l );
#endif
```

```
//<1>       AxisRscI->WeakFV.WfIntgl.l = lwk4 + AxisRscI->WeakFV.WfIntgl.l; /*                      */
#ifdef  WIN32
        IxADDLMTCHK( AxisRscI->WeakFV.WfIntgl.l );
#endif
//<1>       AxisRscI->WeakFV.WfIntgl.l = IxLmtCBS32( AxisRscI->WeakFV.WfIntgl.l );  /*                          */
        AxisRscI->WeakFV.WfIntgl.l = add_limitf(lwk4, AxisRscI->WeakFV.WfIntgl.l);  /*                          */
//<022>       lwk8 = (LONG)AxisRscI->WeakFV.WfIntegLim << 16; /* TMP9,8 = WeakFV.WfIntegLim * 2^16            */
//<022>       AxisRscI->WeakFV.WfIntgl.l = IxLIMITUL( AxisRscI->WeakFV.WfIntgl.l, lwk8, -lwk8 );  /* WFINTEGH = Δ Vq * Kiv (*
2^16 / 2^16)  */
//<4>       lwk6 = (LONG)AxisRscI->WeakFV.WfIntegLim << 16; /* TMP9,8 = WeakFV.WfIntegLim * 2^16            */
        lwk6 = (ULONG)AxisRscI->WeakFV.WfIntegLim << 16;  /* TMP9,8 = WeakFV.WfIntegLim * 2^16            */
//<1>       AxisRscI->WeakFV.WfIntgl.l = IxLIMITUL( AxisRscI->WeakFV.WfIntgl.l, lwk6, -lwk6 );  /* WFINTEGH = Δ Vq * Kiv (*
2^16 / 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                                              */
/*--------------------------------------------------------------------------------------------------*/
#ifdef  WIN32
        IxADDSUBLMTCHKRDY( AxisRscI->WeakFV.WfIntgl.s[1], swk2 );
#endif
//<1>       swk4 = AxisRscI->WeakFV.WfIntgl.s[1] + swk2;  /* Idref = TMP4 = 比 例 項 +  積 分項     */
#ifdef  WIN32
        IxADDLMTCHK( swk4 );
#endif
//<1>       swk4 = IxLmtCBS16( swk4 );  /*                               */
        swk4 = add_limitf(AxisRscI->WeakFV.WfIntgl.s[1], swk2);
//<1>       swk4 = IxLIMITUL( swk4, AxisRscI->WeakFV.WfIdRefLim, -AxisRscI->WeakFV.WfIdRefLim );  /* IdrefLimで リ ミ ット
*/
        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;
#ifndef USE_CMOVE //<2>
        if( AxisRscI->WeakFV.IdOut > 0 )
        {
```

```
     AxisRscI->WeakFV.IdOut = 0;    /* Idref > 0 の 場 合、Idref = 0             */
     AxisRscI->WeakFV.WfIntgl.l = ZEROR; /* Idref > 0 の 場 合 、積分 = 0        */
    }
#else //<2>
    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);
#endif  //<2>
   }

#ifdef  DEBUG_OUTPT
 AxisHdl[0].SvIpRegW->OUTPT = 0x13;    /* for check progress */
#endif  //#ifdef  DEBUG_OUTPT


/***************************************************************************************/
/*                                                         */
/*   ACRd(d軸 電 流 制 御)                                              */
/*                                                         */
/***************************************************************************************/
/*-------------------------------------------------------------------------------------*/
/*   TMP1 = limit( WeakFV.IdOut - IntAdV.IdInData , 2^15 - 1)                      */
/*-------------------------------------------------------------------------------------*/
#ifdef  WIN32
   IxADDSUBLMTCHKRDY( AxisRscI->WeakFV.IdOut, AxisRscI->IntAdV.IdInData );
#endif
//<1>    swk1 = AxisRscI->WeakFV.IdOut - AxisRscI->IntAdV.IdInData;  /* TMP1 <-- WeakFV.IdOut - IntAdV.IdInData
*/
#ifdef  WIN32
   IxSUBLMTCHK( swk1 );
#endif
//<1>    swk1 = IxLmtCBS16( swk1 );  /* TMP1 <-- limit( TMP1 , 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 )                        */
/*-------------------------------------------------------------------------------------*/
//<1>    swk2 = (SHORT)IlibASR32(( (LONG)AxisRscI->IntAdP.KdP * (LONG)swk1 ) , 9); /* ACC <-- IntAdP.KdP * TMP1
```

```
*/
//<1>     swk2 = IxLmtCBS16( swk2 );  /* TMP2 <-- limit( TMP2 , 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 )                               */
/*---------------------------------------------------------------------------------------------------*/
//<4>     lwk4 = ((LONG)AxisRscI->IntAdP.VdLim) << 16;  /*                          */
      lwk4 = ((ULONG)AxisRscI->IntAdP.VdLim) << 16; /*                          */
//<1>     lwk6 = ( (LONG)AxisRscI->IntAdP.KdI * (LONG)swk1 ) << 3;  /*                        */
      lwk6 = mul(AxisRscI->IntAdP.KdI, swk1 ) << 3; /*                          */
#ifdef  WIN32
      IxADDSUBLMTCHKRDY( lwk6, AxisRscI->AcrV.IdIntgl.l );
#endif
//<1>     AxisRscI->AcrV.IdIntgl.l = lwk6 + AxisRscI->AcrV.IdIntgl.l; /*                        */
#ifdef  WIN32
      IxADDLMTCHK( AxisRscI->AcrV.IdIntgl.l );
#endif
//<1>     AxisRscI->AcrV.IdIntgl.l = IxLmtCBS32( AxisRscI->AcrV.IdIntgl.l );  /* AcrV.IdIntgl <-- limit( AcrV.IdIntgl , 2^31 -
1 )         */
      AxisRscI->AcrV.IdIntgl.l = add_limitf(lwk6, AxisRscI->AcrV.IdIntgl.l);  /* AcrV.IdIntgl <-- limit( AcrV.IdIntgl , 2^31 -
      1 )         */
//      AxisRscI->AcrV.IdIntgl.l = limit(AxisRscI->AcrV.IdIntgl.l, lwk4); //<4>
      if( LPX_ABS(AxisRscI->AcrV.IdIntgl.l) > LPX_ABS(lwk4) )
      {
        AxisRscI->StsFlg.CtrlStsRW = AxisRscI->StsFlg.CtrlStsRW | DLIM;   /*                              */
        swk0 = AxisRscI->IntAdP.CtrlSw;
//<4>       if( swk0 != AxisRscI->IntAdP.CtrlSw )
#ifndef USE_CMOVE //<2>
        if( (AxisRscI->IntAdP.CtrlSw & ICLR) != 0 )
        {
          AxisRscI->AcrV.IdIntgl.l = ZEROR; /* else integral clear                  */
        }
#else                   //<2>
        AxisRscI->AcrV.IdIntgl.l = cmove(((AxisRscI->IntAdP.CtrlSw & ICLR) != 0), (LONG)ZEROR, AxisRscI->AcrV.IdIntgl.l);
#endif
      }
```

```
/*--------------------------------------------------------------------------------------------------*/
/*    VcmpV.VdOut = limit( TMP2 + IDIH +TMP3, 2^15 - 1 )                            */
/*--------------------------------------------------------------------------------------------------*/
#ifdef  WIN32
      IxADDSUBLMTCHKRDY( AxisRscI->AcrV.IdIntgl.s[1], swk2 );
#endif
//<1>     swk1 = AxisRscI->AcrV.IdIntgl.s[1] + swk2;  /* TMP1 <-- TMP2 + IDIH           */
#ifdef  WIN32
      IxADDLMTCHK( swk1 );
#endif
//<1>     swk1 = IxLmtCBS16( swk1 );  /* TMP1 <-- limit( TMP1 , 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                  */
/*--------------------------------------------------------------------------------------------------*/
#ifdef  WIN32
      IxADDSUBLMTCHKRDY( swk1, AxisRscI->AcrV.VdFil.s[1] );
#endif
//<1>     swk1 = swk1 - AxisRscI->AcrV.VdFil.s[1];  /* TMP1 <-- TMP1 - VDFH           */
#ifdef  WIN32
      IxSUBLMTCHK( swk1 );
#endif
//<1>     swk1 = IxLmtCBS16( swk1 );  /* TMP1 <-- limit( TMP1 , 2^15 - 1 )        */
      swk1 = sub_limitf(swk1, AxisRscI->AcrV.VdFil.s[1]); /* TMP1 <-- limit( TMP1 , 2^15 - 1 )        */
//<1>     lwk0 = ((LONG)AxisRscI->IntAdP.Tfil * (LONG)swk1) << 2; /*                            */
      lwk0 = mul(AxisRscI->IntAdP.Tfil, swk1) << 2; /*                            */
#ifdef  WIN32
      IxADDSUBLMTCHKRDY( AxisRscI->AcrV.VdFil.l, lwk0 );
#endif
//<1>     lwk2 = AxisRscI->AcrV.VdFil.l + lwk0; /* AcrV.VdFil <-- AcrV.VdFil + TMP0             */
#ifdef  WIN32
      IxADDLMTCHK( lwk2 );
#endif
//<1>     AxisRscI->AcrV.VdFil.l = IxLmtCBS32( lwk2 );  /*                            */
      AxisRscI->AcrV.VdFil.l = add_limitf(AxisRscI->AcrV.VdFil.l, lwk0);  /*                            */

#ifdef  DEBUG_OUTPT
```

```
  AxisHdl[0].SvIpRegW->OUTPT = 0x14;    /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->WeakFV.IqOut;
#endif  //#ifdef  DEBUG_OUTPT



/****************************************************************************************************/
/*                                                  */
/*    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
      {
#ifdef  WIN32
        IxADDSUBLMTCHKRDY( AxisRscI->WeakFV.IqOut, AxisRscI->IntAdV.IqOut2Lpf.s[1] );
#endif
//<1>       swk0 = AxisRscI->WeakFV.IqOut - AxisRscI->IntAdV.IqOut2Lpf.s[1];  /* TMP0 <-- WeakFV.IqOut - IQOF             */
#ifdef  WIN32
        IxSUBLMTCHK( swk0 );
#endif
//<1>       swk0 = IxLmtCBS16( swk0 );  /* TMP0 <-- limit( TMP0, 2^15 - 1 )          */
        swk0 = sub_limitf(AxisRscI->WeakFV.IqOut, AxisRscI->IntAdV.IqOut2Lpf.s[1]); /* TMP0 <-- limit( TMP0, 2^15 - 1 )
        */
//<1>       lwk2 = ( (LONG)AxisRscI->IntAdP.TLpf2 * (LONG)swk0 ) << 2;
        lwk2 = mul(AxisRscI->IntAdP.TLpf2, swk0 ) << 2;
#ifdef  WIN32
        IxADDSUBLMTCHKRDY( AxisRscI->IntAdV.IqOut2Lpf.l, lwk2 );
#endif
//<1>       lwk2 = AxisRscI->IntAdV.IqOut2Lpf.l + lwk2; /* IntAdV.IqOut2Lpf <-- IntAdV.IqOut2Lpf + TMP2           */
#ifdef  WIN32
        IxADDLMTCHK( lwk2 );
#endif
//<1>       AxisRscI->IntAdV.IqOut2Lpf.l = IxLmtCBS32( lwk2 );
        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>
      */
#ifdef  WIN32
      IxADDSUBLMTCHKRDY( AxisRscI->IntAdV.IqOut2Lpf.s[1], AxisRscI->IntAdV.IqDist );
#endif
//<1>     AxisRscI->IntAdV.IqOfRef = AxisRscI->IntAdV.IqOut2Lpf.s[1] + AxisRscI->IntAdV.IqDist; /* IntAdV.IqOfRef = IQOF +
IntAdV.IqDist (外乱トルク加算) <V224>  */
#ifdef  WIN32
      IxADDLMTCHK( AxisRscI->IntAdV.IqOfRef );
#endif
//<1>     AxisRscI->IntAdV.IqOfRef = IxLmtCBS16( AxisRscI->IntAdV.IqOfRef );  /* IntAdV.IqOfRef <-- limit( IntAdV.IqOfRef ,
2^15 - 1 )  <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 Limit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 )
      {
//<1>       lwk4 = (LONG)AxisRscI->WeakFV.IdOut * (LONG)AxisRscI->WeakFV.IdOut;     /* Idref^2              ;
削除<V309 > 復活<V531>  */
        lwk4 = mul(AxisRscI->WeakFV.IdOut, AxisRscI->WeakFV.IdOut);     /* Idref^2              ; 削除<V309 > 復活<V531>  */
      }
      else
      {
//<1>       lwk4 = (LONG)AxisRscI->WeakFV.WfIdRefLim * (LONG)AxisRscI->WeakFV.WfIdRefLim; /* IdrefLim^2          ; <V309>
*/
        lwk4 = mul(AxisRscI->WeakFV.WfIdRefLim, AxisRscI->WeakFV.WfIdRefLim); /* IdrefLim^2          ; <V309>    */
      }
      lwk2 = lwk2 - lwk4; /* Imax^2 - Id^2                */
//      swk0 = MpSQRT( &IntAdwk, lwk2 );          /*                             */
      swk0 = MpSQRT( lwk2 );          /*                             */
      swk1 = swk0;  /* TMP0 = √ ( Imax^2 - Id^2 )            */

#ifdef  DEBUG_OUTPT
      /* 2012.12.21 Y.Oka for ROMSIM なぜかルート計算の出力が不定となる。 */
      AxisHdl[0].SvIpRegW->OUTPT = swk1;
      AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->IntAdV.TLimP;
      AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->IntAdV.TLimM;
      AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->IntAdP.KqP;
      AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->IntAdP.KqI;
      AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->IntAdP.Tfil;
#endif  //#ifdef  DEBUG_OUTPT

/*--------------------------------------------------------------------------------------------------*/
/*    Torque Limit                                      */
/*--------------------------------------------------------------------------------------------------*/
      if( AxisRscI->IntAdV.IqOfRef >= 0 )
      {
//<1>       swk1 = IxLIMITUL( swk1, AxisRscI->IntAdV.TLimP, -AxisRscI->IntAdV.TLimP );  /* 正側トルクリミット            */
        swk1 = limit( swk1, AxisRscI->IntAdV.TLimP ); /* 正側トルクリミット           */
//<1>       AxisRscI->IntAdV.IqRef = IxLIMITUL( AxisRscI->IntAdV.IqOfRef, swk1, -swk1 );  /* <V224>
外 乱 ト ル ク 加 算 後 のq軸電流指令     */
        AxisRscI->IntAdV.IqRef = limit( AxisRscI->IntAdV.IqOfRef, swk1 ); /* <V224> 外 乱 ト ル ク 加 算 後 のq軸電流指令       */
#ifndef USE_CMOVE //<2>
        if( AxisRscI->IntAdV.IqRef == swk1 )
        {
          AxisRscI->StsFlg.CtrlStsRW = AxisRscI->StsFlg.CtrlStsRW | TLIM;   /* TLIM flag set                      */
        }
#else //<2>
        swk10 = AxisRscI->StsFlg.CtrlStsRW | TLIM;    /* TLIM flag set                      */
        AxisRscI->StsFlg.CtrlStsRW = cmove((AxisRscI->IntAdV.IqRef == swk1), swk10, AxisRscI->StsFlg.CtrlStsRW);
#endif  //<2>
      }
      else
      {
//<1>       swk1 = IxLIMITUL( swk1, AxisRscI->IntAdV.TLimM, -AxisRscI->IntAdV.TLimM );  /* 負 側 ト ル ク リミット          */
        swk1 = limit( swk1, AxisRscI->IntAdV.TLimM ); /* 負 側 ト ル ク リミット          */
//<1>       AxisRscI->IntAdV.IqRef = IxLIMITUL( AxisRscI->IntAdV.IqOfRef, swk1, -swk1 );  /* <V224>
外 乱 ト ル ク 加 算 後 のq軸電流指令     */
        AxisRscI->IntAdV.IqRef = limit( AxisRscI->IntAdV.IqOfRef, swk1 ); /* <V224> 外 乱 ト ル ク 加 算 後 のq軸電流指令       */
#ifndef USE_CMOVE //<2>
        if( (AxisRscI->IntAdV.IqRef + swk1) == 0 )
        {
          AxisRscI->StsFlg.CtrlStsRW = AxisRscI->StsFlg.CtrlStsRW | TLIM;   /* TLIM flag set                      */
        }
#else //<2>
        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
        */
#endif  //<2>
      }
/*----------------------------------------------------------------------------------------------------*/
/*    TMP1 = limit( IntAdV.IqRef - IntAdV.IqInData , 2^15 - 1 )                                      */
/*----------------------------------------------------------------------------------------------------*/
#ifdef  WIN32
      IxADDSUBLMTCHKRDY( AxisRscI->IntAdV.IqRef, AxisRscI->IntAdV.IqInData );
#endif
```

```
//<1>     swk1 = AxisRscI->IntAdV.IqRef - AxisRscI->IntAdV.IqInData;  /* TMP1 <-- IQFEF - IntAdV.IqInData               */
#ifdef  WIN32
      IxSUBLMTCHK( swk1 );
#endif
//<1>     swk1 = IxLmtCBS16( swk1 );  /* TMP1 <-- limit( TMP1 , 2^15 - 1 )        */
      swk1 = sub_limitf(AxisRscI->IntAdV.IqRef, AxisRscI->IntAdV.IqInData); /* TMP1 <-- limit( TMP1 , 2^15 - 1 )        */

#ifdef  DEBUG_OUTPT
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->IntAdV.IqInData;
  AxisHdl[0].SvIpRegW->OUTPT = swk1;
#endif  //#ifdef  DEBUG_OUTPT

/*--------------------------------------------------------------------------------------------------*/
/*    TMP2 = limit( IntAdP.KqP * TMP1 / 2^9 , 2^15 - 1 )                                      */
/*--------------------------------------------------------------------------------------------------*/
//<1>     swk2 = (SHORT)IlibASR32( (LONG)AxisRscI->IntAdP.KqP * (LONG)swk1  , 9);   /* TMP2 <-- ACC >> 9                   */
//<1>     swk2 = IxLmtCBS16( swk2 );  /* TMP2 <-- limit( TMP2 , 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 ) )
      {
//<1>       lwk6 = (LONG)AxisRscI->IntAdP.KqI * (LONG)swk1; /* ACC <-- IntAdP.KqI * TMP1                   */
        lwk6 = mul(AxisRscI->IntAdP.KqI, swk1); /* ACC <-- IntAdP.KqI * TMP1                   */
//<4>       lwk4 = (LONG)AxisRscI->IntAdP.VqLim;  /*                               */
        lwk4 = (ULONG)AxisRscI->IntAdP.VqLim; /*                               */
        lwk4 = lwk4 << 16;  /*                           */
        lwk6 = lwk6 << 3; /*                             */
#ifdef  WIN32
        IxADDSUBLMTCHKRDY( lwk6, AxisRscI->AcrV.IqIntgl.l );
#endif
//<1>       AxisRscI->AcrV.IqIntgl.l = lwk6 + AxisRscI->AcrV.IqIntgl.l; /* AcrV.IqIntgl <-- AcrV.IqIntgl + (IntAdP.KqI*TMP1)
*/
#ifdef  WIN32
        IxADDLMTCHK( AxisRscI->AcrV.IqIntgl.l );
#endif
//<1>       AxisRscI->AcrV.IqIntgl.l = IxLmtCBS32( AxisRscI->AcrV.IqIntgl.l );  /* AcrV.IqIntgl <-- limit( AcrV.IqIntgl , 2^32
- 1 )         */
        AxisRscI->AcrV.IqIntgl.l = add_limitf(lwk6, AxisRscI->AcrV.IqIntgl.l);  /* AcrV.IqIntgl <-- limit( AcrV.IqIntgl , 2^32
        - 1 )         */
//        AxisRscI->AcrV.IqIntgl.l = limit(AxisRscI->AcrV.IqIntgl.l, lwk4); //<4>
        if( LPX_ABS(AxisRscI->AcrV.IqIntgl.l) > LPX_ABS(lwk4) )
        {
          AxisRscI->StsFlg.CtrlStsRW = AxisRscI->StsFlg.CtrlStsRW | QLIM;   /* IMM3 <-- STAT | QLIM (imm_16)               */
#ifndef USE_CMOVE //<2>
          if( (AxisRscI->IntAdP.CtrlSw & ICLR) != 0 )
          {
            AxisRscI->AcrV.IqIntgl.l = ZEROR; /* else integral clear                  */
          }
#else //<2>
          swk10 = AxisRscI->IntAdP.CtrlSw & ICLR;
          AxisRscI->AcrV.IqIntgl.l = cmove((swk10 != 0), (LONG)ZEROR, AxisRscI->AcrV.IqIntgl.l);
#endif  //<2>
        }
      }
/*-------------------------------------------------------------------------------------------------*/
/*    VcmpV.VqOut = limit( TMP2 + IQIH +TMP3 , 2^15 - 1 )                                 */
/*-------------------------------------------------------------------------------------------------*/
#ifdef  WIN32
      IxADDSUBLMTCHKRDY( AxisRscI->AcrV.IqIntgl.s[1], swk2 );
#endif
//<1>     swk1 = AxisRscI->AcrV.IqIntgl.s[1] + swk2;  /* TMP1 <-- TMP2 + IQIH           */
#ifdef  WIN32
      IxADDLMTCHK( swk1 );
#endif
//<1>     swk1 = IxLmtCBS16( swk1 );  /* TMP1 <-- limit( TMP1 , 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               */
/*-------------------------------------------------------------------------------------------------*/
//<1>     swk1 = swk1 - AxisRscI->AcrV.VqFil.s[1];  /* TMP1 <-- TMP1 - VQFH           */
//<1>     swk1 = IxLmtCBS16( swk1 );  /* TMP1 <-- limit( TMP1 , 2^15 - 1 )       */
```

```
      swk1 = sub_limitf(swk1, AxisRscI->AcrV.VqFil.s[1]); /* TMP1 <-- limit( TMP1 , 2^15 - 1 )        */
//<1>     lwk0 = ( (LONG)AxisRscI->IntAdP.Tfil * (LONG)swk1 ) << 2; /*                           */
      lwk0 = mul(AxisRscI->IntAdP.Tfil, swk1 ) << 2;  /*                           */
#ifdef  WIN32
      IxADDSUBLMTCHKRDY( AxisRscI->AcrV.VqFil.l, lwk0 );
#endif
//<1>     lwk2 = AxisRscI->AcrV.VqFil.l + lwk0; /* AcrV.VdFil <-- AcrV.VdFil + TMP0             */
#ifdef  WIN32
      IxADDLMTCHK( lwk2 );
#endif
//<1>     AxisRscI->AcrV.VqFil.l = IxLmtCBS32( lwk2 );
      AxisRscI->AcrV.VqFil.l = add_limitf(AxisRscI->AcrV.VqFil.l, lwk0);




#ifdef  DEBUG_OUTPT
  AxisHdl[0].SvIpRegW->OUTPT = 0x15;    /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->IntAdV.IqOut2Lpf.s[1];
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->IntAdV.IqOfRef;
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->IntAdV.IqRef;
  AxisHdl[0].SvIpRegW->OUTPT = swk2;
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->AcrV.IqIntgl.s[1];
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->AcrV.VqFil.s[1];   /* for check progress */
#endif  //#ifdef  DEBUG_OUTPT

/****************************************************************************************************/
/*                                                    */
/*    Voltage Compensation(電　圧　補償 )                                   */
/*                                                    */
/****************************************************************************************************/
      if( (AxisRscI->IntAdP.CtrlSw & ISEL) != 0 )
      {
        swk1 = AxisRscI->WeakFV.IdOut;  /* TMP1 <-- reference ID                  */
        swk2 = AxisRscI->IntAdV.IqRef;  /*                              */

#ifdef  DEBUG_OUTPT
```

```
  AxisHdl[0].SvIpRegW->OUTPT = swk1;    /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = swk2;    /* for check progress */
#endif  //#ifdef  DEBUG_OUTPT

      }
      else
      {
        swk1 = AxisRscI->IntAdV.IdInData; /* TMP1 <-- feedback ID                   */
        swk2 = AxisRscI->IntAdV.IqInData; /* TMP2 <-- feedback IQ                   */

#ifdef  DEBUG_OUTPT
  AxisHdl[0].SvIpRegW->OUTPT = swk1;    /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = swk2;    /* for check progress */
#endif  //#ifdef  DEBUG_OUTPT

      }
/*--------------------------------------------------------------------------------------------------*/
/*    TMP4(VcmpV.VdComp) = IntAdP.MotResist*TMP1/2^15 - VcmpV.LqC * TMP2 / 2^15                       */
/*--------------------------------------------------------------------------------------------------*/
//<1>     swk4 = (SHORT)IlibASR32( ( (LONG)AxisRscI->VcmpV.LqC * (LONG)swk2 ) , 15 );   /* VcmpV.VdComp <-- ACC >> 15
*/
      swk4 = mulshr(AxisRscI->VcmpV.LqC, swk2, 15 );    /* VcmpV.VdComp <-- ACC >> 15                   */
//<1>     swk0 = (SHORT)IlibASR32( ( (LONG)AxisRscI->IntAdP.MotResist * (LONG)swk1 ) , 15 );
      swk0 = mulshr(AxisRscI->IntAdP.MotResist, swk1, 15 );
      swk4 = swk0 - swk4;

#ifdef  DEBUG_OUTPT
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->VcmpV.LqC;   /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->IntAdP.MotResist;    /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = swk4;    /* for check progress */
#endif  //#ifdef  DEBUG_OUTPT

/*--------------------------------------------------------------------------------------------------*/
/*    TMP5(VcmpV.VqComp) = VcmpV.LdC * TMP1 / 2^15 + VcmpV.MagC + IntAdP.MotResist*TMP2/2^15                      */
/*--------------------------------------------------------------------------------------------------*/
//<1>     swk3 = (SHORT)IlibASR32( ( (LONG)AxisRscI->VcmpV.LdC * (LONG)swk1 ) , 15 ); /* TMP3 <-- ACC >> 15
*/
```

```
      swk3 = mulshr(AxisRscI->VcmpV.LdC, swk1, 15 );  /* TMP3 <-- ACC >> 15                   */
//<1>     swk0 = (SHORT)IlibASR32( ( (LONG)AxisRscI->IntAdP.MotResist * (LONG)swk2 ) , 15 );
      swk0 = mulshr(AxisRscI->IntAdP.MotResist, swk2, 15 );
      swk3 = swk3 + AxisRscI->VcmpV.MagC;
      swk5 = swk3 + swk0; /* VcmpV.VqComp <-- VcmpV.MagC + TMP3 + TMP0        */

#ifdef  DEBUG_OUTPT
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->VcmpV.LdC;   /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->IntAdP.MotResist;    /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->VcmpV.MagC;    /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = swk5;    /* for check progress */
#endif  //#ifdef  DEBUG_OUTPT

/*----------------------------------------------------------------------------------------------------*/
/*    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;  /*                              */

#ifdef  DEBUG_OUTPT
  AxisHdl[0].SvIpRegW->OUTPT = 0xf0;    /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->VcmpV.VdComp;    /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->VcmpV.VqComp;    /* for check progress */
#endif  //#ifdef  DEBUG_OUTPT

      }
/*----------------------------------------------------------------------------------------------------*/
/*    filter : I*FL =  ( ( ( TMP1 - I*FH ) * IntAdP.Tfil ) << 2 ) + I*FL                 */
/*----------------------------------------------------------------------------------------------------*/
      else
      {
        swk1 = AxisRscI->WeakFV.IdOut;  /*                              */
#ifdef  WIN32
```

```
        IxADDSUBLMTCHKRDY( swk1, AxisRscI->IntAdV.IdLfil.s[1] );
#endif
//<1>       swk1 = swk1 - AxisRscI->IntAdV.IdLfil.s[1]; /*                      */
#ifdef  WIN32
        IxSUBLMTCHK( swk1 );
#endif
//<1>       swk1 = IxLmtCBS16( swk1 );  /*                          */
        swk1 = sub_limitf(swk1, AxisRscI->IntAdV.IdLfil.s[1]);  /*                          */
//<1>       lwk0 = ( (LONG)AxisRscI->IntAdP.Tfil * (LONG)swk1 ) << 2; /*                            */
        lwk0 = mul(AxisRscI->IntAdP.Tfil, swk1) << 2; /*                            */
#ifdef  WIN32
        IxADDSUBLMTCHKRDY( AxisRscI->IntAdV.IdLfil.l, lwk0 );
#endif
//<1>       lwk2 = AxisRscI->IntAdV.IdLfil.l + lwk0;  /*                          */
#ifdef  WIN32
        IxADDLMTCHK( lwk2 );
#endif
//<1>       AxisRscI->IntAdV.IdLfil.l = IxLmtCBS32( lwk2 ); /*                            */
        AxisRscI->IntAdV.IdLfil.l = add_limitf(AxisRscI->IntAdV.IdLfil.l, lwk0);  /*                            */
/*--------------------------------------------------------------------------------------------------*/
        swk1 = AxisRscI->IntAdV.IqRef;  /*                            */
#ifdef  WIN32
        IxADDSUBLMTCHKRDY( swk1, AxisRscI->IntAdV.IqLfil.s[1] );
#endif
//<1>       swk1 = swk1 - AxisRscI->IntAdV.IqLfil.s[1]; /*                      */
#ifdef  WIN32
        IxSUBLMTCHK( swk1 );
#endif
//<1>       swk1 = IxLmtCBS16( swk1 );  /*                          */
        swk1 = sub_limitf(swk1, AxisRscI->IntAdV.IqLfil.s[1]);  /*                          */
//<1>       lwk0 = ( (LONG)AxisRscI->IntAdP.Tfil * (LONG)swk1 ) << 2; /*                            */
        lwk0 = mul(AxisRscI->IntAdP.Tfil, swk1 ) << 2;  /*                            */
#ifdef  WIN32
        IxADDSUBLMTCHKRDY( AxisRscI->IntAdV.IqLfil.l, lwk0 );
#endif
//<1>       lwk2 = AxisRscI->IntAdV.IqLfil.l + lwk0;  /*                          */
#ifdef  WIN32
```

```
        IxADDLMTCHK( lwk2 );
#endif
//<1>       AxisRscI->IntAdV.IqLfil.l = IxLmtCBS32( lwk2 ); /*                            */
        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];  /*                              */
//<1>       swk2 = (SHORT)IlibASR32(( (LONG)AxisRscI->IntAdP.L_dIdt * (LONG)swk2 ) , 9 ); /*                              */
//<1>       swk2 = IxLmtCBS16( swk2 );  /* limit( VDL , 2^15 - 1 )              */
        swk2 = mulshr_limitf(AxisRscI->IntAdP.L_dIdt, swk2, 9); /* limit( VDL , 2^15 - 1 )              */
#ifdef  WIN32
        IxADDSUBLMTCHKRDY( swk2, swk4 );
#endif
//<1>       swk0 = swk2 + swk4; /* VcmpV.VdComp <-- TMP4 + TMP3             */
#ifdef  WIN32
        IxADDLMTCHK( swk0 );
#endif
//<1>       AxisRscI->VcmpV.VdComp = IxLmtCBS16( swk0 );  /* VcmpV.VdComp <-- limit( VcmpV.VdOut , 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];
//<1>       swk2 = (SHORT)IlibASR32( ( (LONG)AxisRscI->IntAdP.L_dIdt * (LONG)swk2 ) , 9 );  /*                              */
//<1>       swk2 = IxLmtCBS16( swk2 );  /* limit( VQL , 2^15 - 1 )              */
        swk2 = mulshr_limitf(AxisRscI->IntAdP.L_dIdt, swk2, 9); /* limit( VQL , 2^15 - 1 )              */
#ifdef  WIN32
        IxADDSUBLMTCHKRDY( swk2, swk5 );
#endif
//<1>       swk0 = swk2 + swk5; /* VcmpV.VqComp <-- TMP5 + TMP3           */
#ifdef  WIN32
        IxADDLMTCHK( swk0 );
#endif
//<1>       AxisRscI->VcmpV.VqComp = IxLmtCBS16( swk0 );  /* VcmpV.VqComp <-- limit( VcmpV.VqOut , 2^15 - 1 )             */
        AxisRscI->VcmpV.VqComp = add_limitf(swk2, swk5);  /* VcmpV.VqComp <-- limit( VcmpV.VqOut , 2^15 - 1 )             */

#ifdef  DEBUG_OUTPT
  AxisHdl[0].SvIpRegW->OUTPT = 0xf1;    /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->VcmpV.VdComp;    /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->VcmpV.VqComp;    /* for check progress */
#endif  //#ifdef  DEBUG_OUTPT

      }
/*--------------------------------------------------------------------------------------------------*/
/*    TMP1 = limit( VDFH + VcmpV.VdComp , 2^15 - 1 )                            */
/*    TMP2 = limit( VQFH + VcmpV.VqComp , 2^15 - 1 )                            */
/*--------------------------------------------------------------------------------------------------*/
#ifdef  WIN32
      IxADDSUBLMTCHKRDY( AxisRscI->AcrV.VdFil.s[1], AxisRscI->VcmpV.VdComp );
#endif
//<1>     swk0 = AxisRscI->AcrV.VdFil.s[1] + AxisRscI->VcmpV.VdComp;  /* VcmpV.VdOut <-- VDFH + VcmpV.VdComp                */
#ifdef  WIN32
      IxADDLMTCHK( swk0 );
#endif
//<1>     swk1 = IxLmtCBS16( swk0 );  /* VcmpV.VdOut <-- limit( VcmpV.VdOut , 2^15 - 1 )           */
      swk1 = add_limitf(AxisRscI->AcrV.VdFil.s[1], AxisRscI->VcmpV.VdComp); /* VcmpV.VdOut <-- limit( VcmpV.VdOut , 2^15 - 1 )
      */
#ifdef  WIN32
      IxADDSUBLMTCHKRDY( AxisRscI->AcrV.VqFil.s[1], AxisRscI->VcmpV.VqComp );
#endif
//<1>     swk0 = AxisRscI->AcrV.VqFil.s[1] + AxisRscI->VcmpV.VqComp;  /* VcmpV.VqOut <-- VQFH + VcmpV.VqComp                */
#ifdef  WIN32
      IxADDLMTCHK( swk0 );
#endif
//<1>     swk2 = IxLmtCBS16( swk0 );  /* VcmpV.VqOut <-- limit( VcmpV.VqOut , 2^15 - 1 )           */
      swk2 = add_limitf(AxisRscI->AcrV.VqFil.s[1], AxisRscI->VcmpV.VqComp); /* VcmpV.VqOut <-- limit( VcmpV.VqOut , 2^15 - 1 )
      */

#ifdef  DEBUG_OUTPT
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->AcrV.VdFil.s[1];   /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = swk0;    /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = swk1;    /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = swk2;    /* for check progress */
#endif  //#ifdef  DEBUG_OUTPT
```

```
/*--------------------------------------------------------------------------------------------------*/
/*    TMP1 = limit( VcmpV.VdRef + TMP1 , 2^15 - 1 )                           */
/*    TMP2 = limit( VcmpV.VqRef + TMP2 , 2^15 - 1 )                           */
/*--------------------------------------------------------------------------------------------------*/
#ifdef  WIN32
      IxADDSUBLMTCHKRDY( AxisRscI->VcmpV.VdRef, swk1 );
#endif
//<1>     swk0 = AxisRscI->VcmpV.VdRef + swk1;  /* VcmpV.VdOut <-- VcmpV.VdRef + TMP1             */
#ifdef  WIN32
      IxADDLMTCHK( swk0 );
#endif
//<1>     swk1 = IxLmtCBS16( swk0 );  /* VcmpV.VdOut <-- limit( VcmpV.VdOut , 2^15 - 1 )              */
      swk1 = add_limitf(AxisRscI->VcmpV.VdRef, swk1); /* VcmpV.VdOut <-- limit( VcmpV.VdOut , 2^15 - 1 )            */
#ifdef  WIN32
      IxADDSUBLMTCHKRDY( AxisRscI->VcmpV.VqRef, swk2 );
#endif
//<1>     swk0 = AxisRscI->VcmpV.VqRef + swk2;  /* VcmpV.VqOut <-- VcmpV.VqRef + TMP2               */
#ifdef  WIN32
      IxADDLMTCHK( swk0 );
#endif
//<1>     swk2 = IxLmtCBS16( swk0 );  /* VcmpV.VqOut <-- limit( VcmpV.VqOut , 2^15 - 1 )              */
      swk2 = add_limitf(AxisRscI->VcmpV.VqRef, swk2); /* VcmpV.VqOut <-- limit( VcmpV.VqOut , 2^15 - 1 )            */

#ifdef  DEBUG_OUTPT
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->VcmpV.VdRef;   /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = swk0;    /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = swk1;    /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = swk2;    /* for check progress */
#endif  //#ifdef  DEBUG_OUTPT

/*--------------------------------------------------------------------------------------------------*/
/*    VcmpV.VdOut = limit( IntAdP.Kvv * TMP1 / 2^13 , 2^15 - 1 )                          */
/*    VcmpV.VqOut = limit( IntAdP.Kvv * TMP2 / 2^13 , 2^15 - 1 )                          */
/*--------------------------------------------------------------------------------------------------*/
//<1>     swk1 = (SHORT)IlibASR32( ( (LONG)AxisRscI->IntAdP.Kvv * (LONG)swk1 ) , 13 );  /* TMP1 <-- ACC >> 13
*/
//<1>     AxisRscI->VcmpV.VdOut = IxLmtCBS16( swk1 ); /* VcmpV.VdOut   <-- limit( TMP1 , 2^15 - 1 )          */
```

```
      AxisRscI->VcmpV.VdOut = mulshr_limitf(AxisRscI->IntAdP.Kvv, swk1, 13);  /* VcmpV.VdOut   <-- limit( TMP1 , 2^15 - 1 )
      */
//<1>     swk2 = (SHORT)IlibASR32( ( (LONG)AxisRscI->IntAdP.Kvv * (LONG)swk2 ) , 13 );    /* TMP2 <-- ACC >> 13
*/
//<1>     AxisRscI->VcmpV.VqOut = IxLmtCBS16( swk2 );   /* VcmpV.VqOut   <-- limit( TMP2 , 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>             */

#ifdef  DEBUG_OUTPT
  AxisHdl[0].SvIpRegW->OUTPT = 0x16;    /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = swk1;    /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = swk2;    /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->VcmpV.VdOut;   /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->VcmpV.VqOut;   /* for check progress */
#endif  //#ifdef  DEBUG_OUTPT


/****************************************************************************************************/
/*    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               */
/****************************************************************************************************/
//<1>       lwk2 = (LONG)AxisRscI->VcmpV.VdOut * (LONG)AxisRscI->VcmpV.VdOut;
        lwk2 = mul(AxisRscI->VcmpV.VdOut, AxisRscI->VcmpV.VdOut);
//<1>       lwk4 = (LONG)AxisRscI->VcmpV.VqOut * (LONG)AxisRscI->VcmpV.VqOut;
//<2>       lwk4 = mul(AxisRscI->VcmpV.VqOut, AxisRscI->VcmpV.VqOut);
//<2>       lwk2 = lwk2 + lwk4; /* TMP2 = VcmpV.VdOut^2 + VcmpV.VqOut^2            */
        lwk2 = mac(AxisRscI->VcmpV.VqOut, AxisRscI->VcmpV.VqOut, lwk2);
//        swk0 = MpSQRT( &IntAdwk, lwk2 );          /* TMP0 = √ (VcmpV.VdOut^2 + VcmpV.VqOut^2)                */
        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)                  */
#ifndef USE_CMOVE
        if( AxisRscI->IntAdV.V1 < 0 )
        {
          AxisRscI->VcmpV.Vmax2 = AxisRscI->VcmpV.Vmax2 >> 1; /* VcmpV.Vmax2 = 8192 * 1.27 / 2                 */
          AxisRscI->VcmpV.V12 = AxisRscI->IntAdV.V1 >> 1;   /* VcmpV.V12 = √ (VcmpV.VdOut^2 + VcmpV.VqOut^2) / 2
          */
        }
#else //<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);
#endif  //<2>
        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  べ  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                                   */
/*---------------------------------------------------------------------------------------------------*/
//<1>         lwk2 = (LONG)AxisRscI->VcmpV.V12 * (LONG)AxisRscI->VcmpV.V12; /* TMP3,2 = VcmpV.V12^2                  */
          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  べ  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,コンパ
     */
//<4>  lwk6 = (LONG)swk8 * lwk6; /* TMP6 = tblrv(addr)*(2^16-low)          */
     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    */
//<4>  lwk4 = (LONG)swk8 * lwk2; /* TMP4 = tblrv(addr+1)*low             */
     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                   */
//<1>  dlwk = mul( (LONG)swk8, lwk0 );
//<1><4>  lwk2 = (LONG)IlibASR64( dlwk , 28 );    /* TMP2 = MAC / 2^28                 */
     lwk2 = mulshr((ULONG)swk8, lwk0, 28 );    /* TMP2 = MAC / 2^28                 */
//<1>  AxisRscI->VcmpV.VdOut = (SHORT)IlibASR32( ( (LONG)swk2 * (LONG)AxisRscI->VcmpV.VdOut ) , 14 );      /*
VcmpV.VdOut = IntAdP.Vmax / VcmpV.V12 * VcmpV.VdOut * 2^(13+13+16) / 2^(28+14)      */
     AxisRscI->VcmpV.VdOut = mulshr(swk2, AxisRscI->VcmpV.VdOut, 14 );     /* VcmpV.VdOut = IntAdP.Vmax / VcmpV.V12 *
     VcmpV.VdOut * 2^(13+13+16) / 2^(28+14)      */
//<1>  AxisRscI->VcmpV.VqOut = (SHORT)IlibASR32( ( (LONG)swk2 * (LONG)AxisRscI->VcmpV.VqOut ) , 14 );      /*
VcmpV.VqOut = IntAdP.Vmax / VcmpV.V12 * VcmpV.VqOut * 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                 */
    }
   }


#ifdef  DEBUG_OUTPT
 AxisHdl[0].SvIpRegW->OUTPT = 0x17;    /* for check progress */
```

```
#endif  //#ifdef  DEBUG_OUTPT

/****************************************************************************************************/
/*                                                    */
/*    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; /*                              */
//<1>     swk1 = (SHORT)IlibASR32( ( (LONG)AxisRscI->SinTbl.CosT * (LONG)AxisRscI->VcmpV.VdOut ) , 14 );  /* TMP1 <-- ACC >>
14                    */
      swk1 = mulshr(AxisRscI->SinTbl.CosT, AxisRscI->VcmpV.VdOut, 14 ); /* TMP1 <-- ACC >> 14                    */
//<1>     swk2 = (SHORT)IlibASR32( ( (LONG)AxisRscI->SinTbl.SinT * (LONG)AxisRscI->VcmpV.VqOut ) , 14 );    /* TMP2 <-- ACC >>
14                    */
      swk2 = mulshr(AxisRscI->SinTbl.SinT, AxisRscI->VcmpV.VqOut, 14 );   /* TMP2 <-- ACC >> 14                    */
#ifdef  WIN32
      IxADDSUBLMTCHKRDY( swk1, swk2 );
#endif
//<1>     AxisRscI->VcmpV.VuOut = swk1 - swk2;  /* VcmpV.VuOut <-- TMP1 - TMP2                */
#ifdef  WIN32
      IxSUBLMTCHK( AxisRscI->VcmpV.VuOut );
#endif
//<1>     AxisRscI->VcmpV.VuOut = IxLmtCBS16( AxisRscI->VcmpV.VuOut );      /* VcmpV.VuOut <-- limit( VcmpV.VuOut , 2^15 - 1 )
*/
      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 )             */
/*-----------------------------------------------------------------------------------------------*/
//<1>     swk1 = (SHORT)IlibASR32( ( (LONG)AxisRscI->SinTbl.CosT3 * (LONG)AxisRscI->VcmpV.VdOut ) , 14 ); /* TMP1 <-- ACC >>
14                    */
      swk1 = mulshr(AxisRscI->SinTbl.CosT3, AxisRscI->VcmpV.VdOut, 14 );  /* TMP1 <-- ACC >> 14                   */
//<1>     swk2 = (SHORT)IlibASR32( ( (LONG)AxisRscI->SinTbl.SinT3 * (LONG)AxisRscI->VcmpV.VqOut ) , 14 );   /* TMP2 <-- ACC >>
14                    */
      swk2 = mulshr(AxisRscI->SinTbl.SinT3, AxisRscI->VcmpV.VqOut, 14 );    /* TMP2 <-- ACC >> 14                   */
```

```
#ifdef  WIN32
      IxADDSUBLMTCHKRDY( swk1, swk2 );
#endif
//<1>     AxisRscI->VcmpV.VvOut = swk1 - swk2;  /* VcmpV.VvOut <-- TMP1 - TMP2                */
#ifdef  WIN32
      IxSUBLMTCHK( AxisRscI->VcmpV.VvOut );
#endif
//<1>     AxisRscI->VcmpV.VvOut = IxLmtCBS16(AxisRscI-> VcmpV.VvOut );      /* VcmpV.VvOut <-- limit( VcmpV.VvOut , 2^15 - 1 )
*/
      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                  */
#ifdef  WIN32
      IxADDSUBLMTCHKRDY( swk1, AxisRscI->VcmpV.VvOut );
#endif
//<1>     AxisRscI->VcmpV.VwOut = swk1 - AxisRscI->VcmpV.VvOut;
#ifdef  WIN32
      IxSUBLMTCHK( AxisRscI->VcmpV.VwOut );
#endif
//<1>     AxisRscI->VcmpV.VwOut = IxLmtCBS16( AxisRscI->VcmpV.VwOut );      /* VcmpV.VwOut <-- limit( VcmpV.VwOut , 2^15 - 1 )
*/
      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 ); /*                              */



#ifdef  DEBUG_OUTPT
  AxisHdl[0].SvIpRegW->OUTPT = 0x18;    /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->VcmpV.VuOut;   /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->VcmpV.VvOut;   /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->VcmpV.VwOut;   /* for check progress */
#endif  //#ifdef  DEBUG_OUTPT
```

```
/****************************************************************************************************/
/*    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      */
/****************************************************************************************************/
//<1>       lwk2 = (LONG)AxisRscI->VcmpV.VdOut * (LONG)AxisRscI->VcmpV.VdOut;
        lwk2 = mul(AxisRscI->VcmpV.VdOut, AxisRscI->VcmpV.VdOut);
//<1>       lwk4 = (LONG)AxisRscI->VcmpV.VqOut * (LONG)AxisRscI->VcmpV.VqOut;
//<2>       lwk4 = mul(AxisRscI->VcmpV.VqOut, AxisRscI->VcmpV.VqOut);
//<2>       lwk2 = lwk2 + lwk4;
        lwk2 = mac(AxisRscI->VcmpV.VqOut, AxisRscI->VcmpV.VqOut, lwk2);
//        swk0 = MpSQRT( &IntAdwk, lwk2 );
        swk0 = MpSQRT( lwk2 );
        if( (USHORT)swk0 > 0x7FFF )
        {
          swk0 = 0x7FFF;  /* √ の計算が32767を超えたら、32767にする。        ; <V350>  */
        }
        AxisRscI->IntAdV.V1 = swk0;
/*-----------------------------------------------------------------------------------------------*/
/*    飽和判断          <V531> <V537> 復活                */
/*-----------------------------------------------------------------------------------------------*/
#ifndef USE_CMOVE //<2>
        if( AxisRscI->IntAdV.V1 >= 9421 )
        {
          AxisRscI->StsFlg.IntglFlg = AxisRscI->StsFlg.IntglFlg | 1;      /*                              */
        }
        else
        {
          AxisRscI->StsFlg.IntglFlg = AxisRscI->StsFlg.IntglFlg & 0xFFFE;   /*                              */
        }
#else //<2>
        AxisRscI->StsFlg.IntglFlg = AxisRscI->StsFlg.IntglFlg & 0xFFFE;   /*                              */
        swk10 = AxisRscI->StsFlg.IntglFlg | 1;      /*                   */
        AxisRscI->StsFlg.IntglFlg = cmove((AxisRscI->IntAdV.V1 >= 9421), swk10, AxisRscI->StsFlg.IntglFlg);
```

```
#endif  //<2>
      }
/****************************************************************************************************/
/*    Over modulation type select                                 */
/****************************************************************************************************/
      if( AxisRscI->IntAdP.Vmax >= 0x2000 )
      {
        if( (AxisRscI->IntAdP.CtrlSw & OVMSEL2) == 0 )
        {
//<4>         if( ( AxisRscI->IntAdV.V1 >= 0x2000 )||( (AxisRscI->IntAdP.CtrlSw & OVMSEL1) != 0 ) )
          if( ( AxisRscI->IntAdV.V1 >= 0x2000 )&&( (AxisRscI->IntAdP.CtrlSw & OVMSEL1) != 0 ) )
          {
/****************************************************************************************************/
/*    Over modulation1                                      */
/****************************************************************************************************/
//            IxSetCtblAdr( pCtbl, &OVMODTBLG[0][0] );  /* gain type                       */
            IxSetCtblAdr( pCtbl, &(OVMODTBLG[0][0]) );  /* gain type                       */
//            MpOVMMODK( &AxisRscI->IntAdP, &AxisRscI->IntAdV, &IntAdwk );
            MpOVMMODK( &AxisRscI->IntAdP, &AxisRscI->IntAdV, pCtbl );
//<1>           AxisRscI->VcmpV.VuOut = (SHORT)IlibASR32( ( (LONG)AxisRscI->VcmpV.VuOut * (LONG)AxisRscI->IntAdP.Kmod ) , 13
);
//<1>           AxisRscI->VcmpV.VuOut = IxLmtCBS16( AxisRscI->VcmpV.VuOut );
            AxisRscI->VcmpV.VuOut = mulshr_limitf(AxisRscI->VcmpV.VuOut, AxisRscI->IntAdP.Kmod, 13);
//<1>           AxisRscI->VcmpV.VvOut = (SHORT)IlibASR32( ( (LONG)AxisRscI->VcmpV.VvOut * (LONG)AxisRscI->IntAdP.Kmod ) , 13
);
//<1>           AxisRscI->VcmpV.VvOut = IxLmtCBS16( AxisRscI->VcmpV.VvOut );
            AxisRscI->VcmpV.VvOut = mulshr_limitf(AxisRscI->VcmpV.VvOut, AxisRscI->IntAdP.Kmod, 13);
//<1>           AxisRscI->VcmpV.VwOut = (SHORT)IlibASR32( ( (LONG)AxisRscI->VcmpV.VwOut * (LONG)AxisRscI->IntAdP.Kmod ) , 13
);
//<1>           AxisRscI->VcmpV.VwOut = IxLmtCBS16( AxisRscI->VcmpV.VwOut );
            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 );
```

```
//<2>           swk1 = (SHORT)( (LONG)swk4 * (LONG)AxisRscI->VcmpV.VuOut );
            swk1 = swk4 * AxisRscI->VcmpV.VuOut;
            swk5 = IxLIMIT( AxisRscI->VcmpV.VvOut, swk0 );
//<2>           swk2 = (SHORT)( (LONG)swk5 * (LONG)AxisRscI->VcmpV.VvOut );
            swk2 = swk5 * AxisRscI->VcmpV.VvOut;
            swk6 = IxLIMIT( AxisRscI->VcmpV.VwOut, swk0 );
//<2>           swk3 = (SHORT)( (LONG)swk6 * (LONG)AxisRscI->VcmpV.VwOut );
            swk3 = swk6 * AxisRscI->VcmpV.VwOut;
            if( swk1 >= swk2 )
            {
              if( swk1 >= swk3 )
              {
#ifdef  WIN32
                IxADDSUBLMTCHKRDY( swk1, 0x2000 );
#endif
                swk1 = swk1 - 0x2000; /* TMP1 <-- |VcmpV.VuOut|-2000h                */
#ifdef  WIN32
                IxSUBLMTCHK( swk1 );
#endif
                IxLmtzImm16( swk1, 0x7fff );  /* zero limit                    */
//<2>               swk0 = (SHORT)( (LONG)swk4 * (LONG)swk1 );
                swk0 = swk4 * swk1;
              }
              else
              {
#ifdef  WIN32
                IxADDSUBLMTCHKRDY( swk3, 0x2000 );
#endif
                swk3 = swk3 - 0x2000; /* TMP0 <-- |VcmpV.VwOut|-2000h                */
#ifdef  WIN32
                IxSUBLMTCHK( swk3 );
#endif
                IxLmtzImm16( swk3, 0x7fff );  /* zero limit                    */
//<2>               swk0 = (SHORT)( (LONG)swk6 * (LONG)swk3 );
                swk0 = swk6 * swk3;
              }
            }
            else
            {
              if( swk2 >= swk3 )
              {
#ifdef  WIN32
                IxADDSUBLMTCHKRDY( swk2, 0x2000 );
#endif
                swk2 = swk2 - 0x2000; /* TMP0 <-- |VcmpV.VvOut|-2000h                    */
#ifdef  WIN32
                IxSUBLMTCHK( swk2 );
#endif
                IxLmtzImm16( swk2, 0x7fff );  /* zero limit                     */
//<2>               swk0 = (SHORT)( (LONG)swk5 * (LONG)swk2 );
                swk0 = swk5 * swk2;
              }
              else
              {
#ifdef  WIN32
                IxADDSUBLMTCHKRDY( swk3, 0x2000 );
#endif
                swk3 = swk3 - 0x2000; /* TMP0 <-- |VcmpV.VwOut|-2000h                  */
#ifdef  WIN32
                IxSUBLMTCHK( swk3 );
#endif
                IxLmtzImm16( swk3, 0x7fff );  /* zero limit                     */
//<2>               swk0 = (SHORT)( (LONG)swk6 * (LONG)swk3 );
                swk0 = swk6 * swk3;
              }
            }
#ifdef  WIN32
            IxADDSUBLMTCHKRDY( AxisRscI->VcmpV.VuOut, swk0 );
#endif
//<1>           AxisRscI->VcmpV.VuOut = AxisRscI->VcmpV.VuOut - swk0;
#ifdef  WIN32
            IxSUBLMTCHK( AxisRscI->VcmpV.VuOut );
#endif
//<1>           AxisRscI->VcmpV.VuOut = IxLmtCBS16( AxisRscI->VcmpV.VuOut );      /*                              */
```

```
            AxisRscI->VcmpV.VuOut = sub_limitf(AxisRscI->VcmpV.VuOut, swk0);      /*                              */
#ifdef  WIN32
            IxADDSUBLMTCHKRDY( AxisRscI->VcmpV.VvOut, swk0 );
#endif
//<1>           AxisRscI->VcmpV.VvOut = AxisRscI->VcmpV.VvOut - swk0;
#ifdef  WIN32
            IxSUBLMTCHK( AxisRscI->VcmpV.VvOut );
#endif
//<1>           AxisRscI->VcmpV.VvOut = IxLmtCBS16( AxisRscI->VcmpV.VvOut );      /*                              */
            AxisRscI->VcmpV.VvOut = sub_limitf(AxisRscI->VcmpV.VvOut, swk0);      /*                              */
#ifdef  WIN32
            IxADDSUBLMTCHKRDY( AxisRscI->VcmpV.VwOut, swk0 );
#endif
//<1>           AxisRscI->VcmpV.VwOut = AxisRscI->VcmpV.VwOut - swk0;
#ifdef  WIN32
            IxSUBLMTCHK( AxisRscI->VcmpV.VwOut );
#endif
//<1>           AxisRscI->VcmpV.VwOut = IxLmtCBS16( AxisRscI->VcmpV.VwOut );      /*                              */
            AxisRscI->VcmpV.VwOut = sub_limitf(AxisRscI->VcmpV.VwOut, swk0);      /*                              */
            AxisRscI->IntAdV.Vcent = swk0;
          }
        }
/****************************************************************************************************/
/*    Over modulation2                                                */
/****************************************************************************************************/
        else
        {
//          IxSetCtblAdr( pCtbl, &(OVMODTBL0) );  /* ofset type                    */
          IxSetCtblAdr( pCtbl, &(OVMODTBL0[0][0]) );  /* ofset type                    */
//          MpOVMMODK( &AxisRscI->IntAdP, &AxisRscI->IntAdV, &IntAdwk );
          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;
            }
          }
#ifdef  WIN32
          IxADDSUBLMTCHKRDY( swk2, swk1 );
#endif
//<1>         swk0 = swk2 + swk1;
#ifdef  WIN32
          IxADDLMTCHK( swk0 );
#endif
//<1>         swk0 = IxLmtCBS16( swk0 );  /*                            */
          swk0 = add_limitf(swk2, swk1);  /*                            */
//<1>         swk0 = (SHORT)IlibASR32((LONG)swk0 , 1);
          swk0 = mulshr(swk0, ONE, 1);
/*--------------------------------------------------------------------------------------------------*/
#ifdef  WIN32
          IxADDSUBLMTCHKRDY( AxisRscI->VcmpV.VuOut, swk0 );
#endif
//<1>         AxisRscI->VcmpV.VuOut = AxisRscI->VcmpV.VuOut - swk0;
#ifdef  WIN32
          IxSUBLMTCHK( AxisRscI->VcmpV.VuOut );
```

```
#endif
//<1>         AxisRscI->VcmpV.VuOut = IxLmtCBS16( AxisRscI->VcmpV.VuOut );      /*                              */
          AxisRscI->VcmpV.VuOut = sub_limitf(AxisRscI->VcmpV.VuOut, swk0);      /*                              */
#ifdef  WIN32
          IxADDSUBLMTCHKRDY( AxisRscI->VcmpV.VvOut, swk0 );
#endif
//<1>         AxisRscI->VcmpV.VvOut = AxisRscI->VcmpV.VvOut - swk0;
#ifdef  WIN32
          IxSUBLMTCHK( AxisRscI->VcmpV.VvOut );
#endif
//<1>         AxisRscI->VcmpV.VvOut = IxLmtCBS16( AxisRscI->VcmpV.VvOut );      /*                              */
          AxisRscI->VcmpV.VvOut = sub_limitf(AxisRscI->VcmpV.VvOut, swk0);      /*                              */
#ifdef  WIN32
          IxADDSUBLMTCHKRDY( AxisRscI->VcmpV.VwOut, swk0 );
#endif
//<1>         AxisRscI->VcmpV.VwOut = AxisRscI->VcmpV.VwOut - swk0;
#ifdef  WIN32
          IxSUBLMTCHK( AxisRscI->VcmpV.VwOut );
#endif
//<1>         AxisRscI->VcmpV.VwOut = IxLmtCBS16( AxisRscI->VcmpV.VwOut );      /*                              */
          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)                */
//<1>         AxisRscI->VcmpV.VuOut = (SHORT)( (LONG)swk1 * (LONG)AxisRscI->IntAdP.Kmod ) + AxisRscI->VcmpV.VuOut;
//<1>         AxisRscI->VcmpV.VuOut = IxLmtCBS16( AxisRscI->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)                          */
//<1>         AxisRscI->VcmpV.VvOut = (SHORT)( (LONG)swk1 * (LONG)AxisRscI->IntAdP.Kmod ) + AxisRscI->VcmpV.VvOut;
//<1>         AxisRscI->VcmpV.VvOut = IxLmtCBS16( AxisRscI->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)                          */
//<1>         AxisRscI->VcmpV.VwOut = (SHORT)( (LONG)swk1 * (LONG)AxisRscI->IntAdP.Kmod ) + AxisRscI->VcmpV.VwOut;
//<1>         AxisRscI->VcmpV.VwOut = IxLmtCBS16( AxisRscI->VcmpV.VwOut );      /*                              */
          swk2 = swk1 * AxisRscI->IntAdP.Kmod;
          AxisRscI->VcmpV.VwOut = add_limitf( swk2, AxisRscI->VcmpV.VwOut );      /*                              */
        }
      }

#ifdef  DEBUG_OUTPT
  AxisHdl[0].SvIpRegW->OUTPT = 0x19;    /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->VcmpV.VuOut;   /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->VcmpV.VvOut;   /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->VcmpV.VwOut;   /* for check progress */
#endif  //#ifdef  DEBUG_OUTPT

/****************************************************************************************************/
/*    On-Delay                                          */
/****************************************************************************************************/
/*--------------------------------------------------------------------------------------------------*/
/*  IU, IV reference calc                                     */
/*--------------------------------------------------------------------------------------------------*/
//<1>     swk1 = (SHORT)IlibASR32( ( (LONG)AxisRscI->WeakFV.IdOut * (LONG)AxisRscI->SinTbl.CosT ) , 14 ); /* TMP1 <-- ACC >>
14                    */
      swk1 = mulshr(AxisRscI->WeakFV.IdOut, AxisRscI->SinTbl.CosT, 14 );  /* TMP1 <-- ACC >> 14                   */
//<1>     swk2 = (SHORT)IlibASR32( ( (LONG)AxisRscI->IntAdV.IqRef * (LONG)AxisRscI->SinTbl.SinT ) , 14 );   /* TMP2 <-- ACC >>
14                    */
      swk2 = mulshr(AxisRscI->IntAdV.IqRef, AxisRscI->SinTbl.SinT, 14 );    /* TMP2 <-- ACC >> 14                   */
      AxisRscI->IntAdV.IuOut = swk1 - swk2; /* IntAdV.IuOut  <--  TMP1 - TMP2               */

//<1>     swk3 = (SHORT)IlibASR32( ( (LONG)AxisRscI->WeakFV.IdOut * (LONG)AxisRscI->SinTbl.CosT3 ) , 14 );  /* TMP3 <-- ACC >>
14                    */
      swk3 = mulshr(AxisRscI->WeakFV.IdOut, AxisRscI->SinTbl.CosT3, 14 ); /* TMP3 <-- ACC >> 14                   */
//<1>     swk4 = (SHORT)IlibASR32( ( (LONG)AxisRscI->IntAdV.IqRef * (LONG)AxisRscI->SinTbl.SinT3 ) , 14 );    /* TMP4 <-- 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 ｓｗｋ２を　以　降　使　わ　な　い ため代入 は行わ
      {
        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 ｓｗｋ４を　以　降　使　わ　な　い ため代入 は行なわない
      {
//<4>       swk3 = AxisRscI->IntAdV.IuOut - AxisRscI->IntAdV.IvOut; /* TMP3                     */
        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             */
//<2>       swk1 = (SHORT)( (LONG)AxisRscI->IntAdP.OnDelayComp * (LONG)swk6 );
        swk1 = AxisRscI->IntAdP.OnDelayComp * swk6;
/*------------------------------------------------------------------------------------------------*/
/*    TMP2(ONDVU) = sign(IV)*IntAdP.OnDelayComp                                 */
/*------------------------------------------------------------------------------------------------*/
        swk6 = IxLIMIT( swk2, swk5 );
//<2>       swk2 = (SHORT)( (LONG)AxisRscI->IntAdP.OnDelayComp * (LONG)swk6 );
        swk2 = AxisRscI->IntAdP.OnDelayComp * swk6;
/*------------------------------------------------------------------------------------------------*/
/*    TMP3(ONDVU) = sign(IW)*IntAdP.OnDelayComp                                 */
/*------------------------------------------------------------------------------------------------*/
        swk6 = IxLIMIT( swk3, swk5 );
//<2>       swk3 = (SHORT)( (LONG)AxisRscI->IntAdP.OnDelayComp * (LONG)swk6 );
        swk3 = AxisRscI->IntAdP.OnDelayComp * swk6;
      }
/*------------------------------------------------------------------------------------------------*/
/*    trapezoid type                                        */
/*------------------------------------------------------------------------------------------------*/
      else
      {
//<1>       swk0 = (SHORT)IlibASR32( ( (LONG)AxisRscI->IntAdP.OnDelaySlope * (LONG)swk1 ) , 8 );    /* TMP0 <--
IU*IntAdP.OnDelaySlope>>8                   */
//<1>       swk0 = IxLmtCBS16( swk0 );  /* TMP0 = limit(TMP0,2^15-1)              */
        swk0 = mulshr_limitf(AxisRscI->IntAdP.OnDelaySlope, swk1, 8 );    /* TMP0 <-- IU*IntAdP.OnDelaySlope>>8
        */
/* for debug */
  ComWk.WREG104 = swk0;
//        swk0 = IxLmtCBS16(
//                      (SHORT)IlibASR32( ( (LONG)AxisRscI->IntAdP.OnDelaySlope * (LONG)swk1 ) , 8 )
//                      );  /* TMP0 = limit(TMP0,2^15-1)              */
        swk0 = IxLIMIT( swk0, 8192 ); /* TMP0 = limit(TMP0,8192)                  */
//<1>       swk1 = (SHORT)IlibASR32( ( (LONG)AxisRscI->IntAdP.OnDelayComp * (LONG)swk0 ) , 13 );  /* TMP1(ONDVU) =
(IntAdP.OnDelayComp*TMP0)>>13             */
        swk1 = mulshr(AxisRscI->IntAdP.OnDelayComp, swk0, 13 ); /* TMP1(ONDVU) = (IntAdP.OnDelayComp*TMP0)>>13              */
/*--------------------------------------------------------------------------------------------------*/
//<1>       swk0 = (SHORT)IlibASR32( ( (LONG)AxisRscI->IntAdP.OnDelaySlope * (LONG)swk2 ) , 8 );    /* TMP0 <--
IV*IntAdP.OnDelaySlope>>8                   */
//<1>       swk0 = IxLmtCBS16( swk0 );  /* TMP0 = limit(TMP0,2^15-1)              */
        swk0 = mulshr_limitf(AxisRscI->IntAdP.OnDelaySlope, swk2, 8); /* TMP0 = limit(TMP0,2^15-1)              */
/* for debug */
  ComWk.WREG109 = swk0;
        swk0 = IxLIMIT( swk0, 8192 ); /* TMP0 = limit(TMP0,8192)                  */
//<1>       swk2 = (SHORT)IlibASR32( ( (LONG)AxisRscI->IntAdP.OnDelayComp * (LONG)swk0 ) , 13 );  /* TMP1(ONDVU) =
(IntAdP.OnDelayComp*TMP0)>>13             */
        swk2 = mulshr(AxisRscI->IntAdP.OnDelayComp, swk0, 13 ); /* TMP1(ONDVU) = (IntAdP.OnDelayComp*TMP0)>>13              */
/*--------------------------------------------------------------------------------------------------*/
//<1>       swk0 = (SHORT)IlibASR32( ( (LONG)AxisRscI->IntAdP.OnDelaySlope * (LONG)swk3 ) , 8 );    /* TMP0 <--
IV*IntAdP.OnDelaySlope>>8                   */
//<1>       swk0 = IxLmtCBS16( swk0 );  /* TMP0 = limit(TMP0,2^15-1)              */
        swk0 = mulshr_limitf(AxisRscI->IntAdP.OnDelaySlope, swk3, 8); /* TMP0 = limit(TMP0,2^15-1)              */
/* for debug */
  ComWk.Dummy = swk6;
        swk0 = IxLIMIT( swk0, 8192 ); /* TMP0 = limit(TMP0,8192)                  */
//<1>       swk3 = (SHORT)IlibASR32( ( (LONG)AxisRscI->IntAdP.OnDelayComp * (LONG)swk0 ) , 13 );  /* TMP1(ONDVU) =
(IntAdP.OnDelayComp*TMP0)>>13             */
        swk3 = mulshr(AxisRscI->IntAdP.OnDelayComp, swk0, 13 ); /* TMP1(ONDVU) = (IntAdP.OnDelayComp*TMP0)>>13              */
      }
/*--------------------------------------------------------------------------------------------------*/



#ifdef  DEBUG_OUTPT
  AxisHdl[0].SvIpRegW->OUTPT = 0x20;    /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->VcmpV.VuOut;   /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->VcmpV.VvOut;   /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = AxisRscI->VcmpV.VwOut;   /* for check progress */
```

```
#endif  //#ifdef  DEBUG_OUTPT

/****************************************************************************************************/
/*    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 );

      swk4 = swk7 - AxisRscI->VcmpV.VuOut;
//<1>     swk4 = (SHORT)IlibASR32( ( (LONG)swk4 * (LONG)AxisRscI->IntAdV.CrFreqW ) , 14 );
      swk4 = mulshr(swk4, AxisRscI->IntAdV.CrFreqW, 14 );
      swk5 = swk7 - AxisRscI->VcmpV.VvOut;
//<1>     swk5 = (SHORT)IlibASR32(( (LONG)swk5 * (LONG)AxisRscI->IntAdV.CrFreqW ) , 14 );
      swk5 = mulshr(swk5, AxisRscI->IntAdV.CrFreqW, 14 );
      swk6 = swk7 - AxisRscI->VcmpV.VwOut;
//<1>     swk6 = (SHORT)IlibASR32( ( (LONG)swk6 * (LONG)AxisRscI->IntAdV.CrFreqW ) , 14 );
      swk6 = mulshr(swk6, AxisRscI->IntAdV.CrFreqW, 14 );


/*--------------------------------------------------------------------------------------------------*/
/*    Deat-time compensation (timer) : if(Vx == 0 || Vx == IntAdV.CrFreqW) No compensation          */
/*--------------------------------------------------------------------------------------------------*/
//<4>     if( ( swk4 != ZEROR ) || (swk4 != AxisRscI->IntAdV.CrFreqW ) )
      if( ( swk4 != ZEROR ) && (swk4 != AxisRscI->IntAdV.CrFreqW ) )
      {
#ifdef  WIN32
        IxADDSUBLMTCHKRDY( swk4, swk1 );
#endif
        swk4 = swk4 - swk1; /* VcmpV.VuOut <-- VcmpV.VuOut+ONDVU                */
/* for debug */
  ComWk.WREG89 = swk4;
#ifdef  WIN32
        IxSUBLMTCHK( swk4 );
#endif
```

```
        IxLmtzReg16( swk4, swk4, AxisRscI->IntAdV.CrFreqW );  /* VcmpV.VuOut <-- limitz( VcmpV.VuOut , IntAdV.CrFreqW )
        */
/* for debug */
  ComWk.WREG101 = swk4;
      }
//<4>     if( ( swk5 != ZEROR ) || (swk5 != AxisRscI->IntAdV.CrFreqW ) )
      if( ( swk5 != ZEROR ) && (swk5 != AxisRscI->IntAdV.CrFreqW ) )
      {
#ifdef  WIN32
        IxADDSUBLMTCHKRDY( swk5, swk2 );
#endif
        swk5 = swk5 - swk2; /* VcmpV.VvOut <-- VcmpV.VvOut+ONDVV              */
/* for debug */
  ComWk.WREG95 = swk5;
#ifdef  WIN32
        IxSUBLMTCHK( swk5 );
#endif
        IxLmtzReg16( swk5, swk5, AxisRscI->IntAdV.CrFreqW );  /* VcmpV.VvOut <-- limitz( VcmpV.VvOut , IntAdV.CrFreqW )
        */
/* for debug */
  ComWk.WREG102 = swk5;
      }
//<4>     if( ( swk6 != ZEROR ) || (swk6 != AxisRscI->IntAdV.CrFreqW ) )
      if( ( swk6 != ZEROR ) && (swk6 != AxisRscI->IntAdV.CrFreqW ) )
      {
#ifdef  WIN32
        IxADDSUBLMTCHKRDY( swk6, swk3 );
#endif
        swk6 = swk6 - swk3; /* VcmpV.VwOut <-- VcmpV.VwOut+ONDVW                */
/* for debug */
  ComWk.WREG100 = swk6;
#ifdef  WIN32
        IxSUBLMTCHK( swk6 );
#endif
        IxLmtzReg16( swk6, swk6, AxisRscI->IntAdV.CrFreqW );  /* VcmpV.VwOut <-- limitz( VcmpV.VwOut , IntAdV.CrFreqW )
        */
/* for debug */
```

```
  ComWk.WREG103 = swk6;
      }

/*------------------------------------------------------------------------------------------------*/
/*    Output Voltage & status                                   */
/*------------------------------------------------------------------------------------------------*/
    }
//<2>#ifdef PREG_DEF
#ifndef PREG_DEF
    CTSTW = AxisRscI->StsFlg.CtrlStsRW; /* Status Set                       */
#else   //#ifdef  PREG_DEF
    AxisRscI->SvIpRegW->CTSTW = AxisRscI->StsFlg.CtrlStsRW; /* Status Set                       */
#endif    //#ifdef  PREG_DEF
  }

  /* Output PWM Data */
#if 0 //<2>
#ifdef  MULTI_AXIS              /* 多  軸  処  理有効                     */
  for( ax_noI = 0; (SHORT)ax_noI < AxisInfo.AxisNum; ax_noI++ )
#else   //#ifdef  MULTI_AXIS
  ax_noI = 0;
#endif    //#ifdef  MULTI_AXIS
  {
    AxisRscI = &AxisHdl[ax_noI];
/****************************************************************************************************/
/*    PWM data set(for test)                                    */
/****************************************************************************************************/
#ifdef  PREG_DEF
    PwmT2 = swk6;
    PwmT1 = swk5;
    PwmT0 = swk4;
#else   //#ifdef  PREG_DEF
    AxisRscI->SvIpRegW->PwmT2 = swk6;
    AxisRscI->SvIpRegW->PwmT1 = swk5;
    AxisRscI->SvIpRegW->PwmT0 = swk4;
#endif    //#ifdef  PREG_DEF
  }
```

```
#else //<2>
  SetPWM(swk4, swk5, swk6);
#endif  //<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★    */
  /* level(AD=3, INT1=0/4 HOST=0) */
#ifdef  FREG_DEF
  INTLVWR |= 0x0004;
#else   //#ifdef  FREG_DEF
  AxisHdl[0].SvIpRegW->INTLVWR |= 0x0004;
#endif    //#ifdef  FREG_DEF

//<2>#ifdef PREG_DEF
#ifndef PREG_DEF
  OUTPT = 0x0;
#else   //#ifdef  PREG_DEF
  AxisHdl[0].SvIpRegW->OUTPT = 0x0;
#endif    //#ifdef  PREG_DEF


#ifdef  DEBUG_OUTPT
  AxisHdl[0].SvIpRegW->OUTPT = 0x21;    /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = swk6;    /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = swk5;    /* for check progress */
  AxisHdl[0].SvIpRegW->OUTPT = swk4;    /* for check progress */
#endif  //#ifdef  DEBUG_OUTPT


  ComWk.WREG84 = swk6;
  ComWk.WREG85 = swk5;
  ComWk.WREG86 = swk4;

  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] = SRPG0RD5;    /* 今回値読込み:Position Low            */
    IEncWk.RxPos.s[1] = SRPG0RD6; /* 今回値読込み:Position High           */
/*--------------------------------------------------------------------------------------*/
    IEncWk.EncWk0 = INT1SET;      /* INT1 Acknowledge                */
/*--------------------------------------------------------------------------------------*/
    return;           /* return                    */
}



/****************************************************************************************/
/*                                                  */
/*    分周パルス更新処理             ; 最大:???clk,    通常:???clk        <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_ENC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.RxPosの値は@INT_ENC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 tCPU    */
      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 (HostCPU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 (HostCPU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 HostCPU   */
    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 );
```

```
#ifdef  DEBUG_OUTPT
    AxisHdl[0].SvIpRegW->OUTPT = 0x30;
    AxisHdl[0].SvIpRegW->OUTPT = Low;
    AxisHdl[0].SvIpRegW->OUTPT = High;
#endif  //#ifdef  DEBUG_OUTPT

/*------------------------------------------------------------------------------------------------*/
/*    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)           */

#ifdef  DEBUG_OUTPT
      AxisHdl[0].SvIpRegW->OUTPT = 0x31;
      AxisHdl[0].SvIpRegW->OUTPT = uswk5;
#endif  //#ifdef  DEBUG_OUTPT

    }
    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)            */

#ifdef  DEBUG_OUTPT
      AxisHdl[0].SvIpRegW->OUTPT = 0x32;
      AxisHdl[0].SvIpRegW->OUTPT = uswk5;
#endif  //#ifdef  DEBUG_OUTPT

    }
    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)              */

#ifdef  DEBUG_OUTPT
      AxisHdl[0].SvIpRegW->OUTPT = 0x33;
      AxisHdl[0].SvIpRegW->OUTPT = uswk5;
#endif  //#ifdef  DEBUG_OUTPT

    }
    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                       */

#ifdef  DEBUG_OUTPT
      AxisHdl[0].SvIpRegW->OUTPT = 0x34;
      AxisHdl[0].SvIpRegW->OUTPT = uswk5;
#endif  //#ifdef  DEBUG_OUTPT

    }
    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)             */

#ifdef  DEBUG_OUTPT
      AxisHdl[0].SvIpRegW->OUTPT = 0x35;
      AxisHdl[0].SvIpRegW->OUTPT = uswk5;
#endif  //#ifdef  DEBUG_OUTPT

    }
    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)             */

#ifdef  DEBUG_OUTPT
      AxisHdl[0].SvIpRegW->OUTPT = 0x36;
      AxisHdl[0].SvIpRegW->OUTPT = uswk5;
#endif  //#ifdef  DEBUG_OUTPT

    }
// |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]                */

#ifdef  DEBUG_OUTPT
      AxisHdl[0].SvIpRegW->OUTPT = uswk3;
#endif  //#ifdef  DEBUG_OUTPT

      if( uswk5 == 0x00FF )
      {
        uswk0 = 0xFFFF; /* TMP0 <-- (tbl[tmp+1])                  */

#ifdef  DEBUG_OUTPT
        AxisHdl[0].SvIpRegW->OUTPT = 0x3a;
#endif  //#ifdef  DEBUG_OUTPT

      }
      else
      {
        uswk5 = uswk5 + 1;
        IxTblSqrt16( (uswk0), uswk5 );  /* TMP0 <-- tbl[tmp+1]                  */

#ifdef  DEBUG_OUTPT
        AxisHdl[0].SvIpRegW->OUTPT = 0x3b;
        AxisHdl[0].SvIpRegW->OUTPT = uswk5;
        AxisHdl[0].SvIpRegW->OUTPT = uswk0;
#endif  //#ifdef  DEBUG_OUTPT
      }
/*------------------------------------------------------------------------------------------------*/
/*    (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             */

#ifdef  DEBUG_OUTPT
      AxisHdl[0].SvIpRegW->OUTPT = 0x37;
      AxisHdl[0].SvIpRegW->OUTPT = uswk0;
#endif  //#ifdef  DEBUG_OUTPT

    }
/*------------------------------------------------------------------------------------------------*/
/*    Scaling                                           */
/*------------------------------------------------------------------------------------------------*/
//<022>   ulwk2 = (ULONG)(uswk0);
//<022>   ulwk0 = (ulwk2 >> uswk6);
    ulwk0 = ((ULONG)(uswk0) >> uswk6);

#ifdef  DEBUG_OUTPT
    AxisHdl[0].SvIpRegW->OUTPT = 0x38;
    AxisHdl[0].SvIpRegW->OUTPT = uswk0;
#endif  //#ifdef  DEBUG_OUTPT
```

```
    return( (USHORT)ulwk0 );
}
#else
//<3> start
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 )
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>

#ifdef  DEBUG_OUTPT
  AxisHdl[0].SvIpRegW->OUTPT = 0x40;    /* for check progress */
#endif  //#ifdef  DEBUG_OUTPT

  if( IntAdV->V1 < 9459 )
  {
//<2>   IxLoadMpmem16( IntAdP->Kmod, pCtbl, 0 );  /* IntAdP->Kmod = G[0];                     */
    IxLoadMpmem16( swk4, pCtbl, 0 );  /* IntAdP->Kmod = G[0];                     */

#ifdef  DEBUG_OUTPT
    AxisHdl[0].SvIpRegW->OUTPT = 0x41;    /* for check progress */
    AxisHdl[0].SvIpRegW->OUTPT = IntAdP->Kmod;    /* for check progress */
#endif  //#ifdef  DEBUG_OUTPT

  }
  else if( (IntAdP->CtrlSw & OVMMOD) == 0 )
  {
    pCtbl = pCtbl + 15;
//<2>   IxLoadMpmem16( IntAdP->Kmod, pCtbl, 1 );
    IxLoadMpmem16( swk4, pCtbl, 1 );

#ifdef  DEBUG_OUTPT
    AxisHdl[0].SvIpRegW->OUTPT = 0x42;    /* for check progress */
    AxisHdl[0].SvIpRegW->OUTPT = IntAdP->Kmod;    /* for check progress */
#endif  //#ifdef  DEBUG_OUTPT

  }
  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 );
    }

#ifdef  DEBUG_OUTPT
    AxisHdl[0].SvIpRegW->OUTPT = 0x43;    /* for check progress */
    AxisHdl[0].SvIpRegW->OUTPT = IntAdP->Kmod;    /* for check progress */
#endif  //#ifdef  DEBUG_OUTPT

  }
  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32bit)       */
/*                        /* lswk4   : 乗 算 結 果 保 持用(上位32bit)       */
/*--------------------------------------------------------------------------------------------------*/
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
void ADConvDataLoad(INTADV *IntAdV, INTADP *IntAdP)
{
  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);
  IntAdV->IuInData = mulshr((swk + IntAdV->IuOffset), IntAdP->Kcu, 8 );
/*------------------------------------------------------------------------------*/
  swk = mulshr(IvAD, ONE, 2);
  IntAdV->IvInData = mulshr((swk + IntAdV->IvOffset), IntAdP->Kcv, 8 );
#ifdef  MULTI_AXIS
  swk = mulshr(IuAD_2, ONE, 2);
  IntAdV->IuInData = mulshr((swk + IntAdV->IuOffset), IntAdP->Kcu, 8 );
/*------------------------------------------------------------------------------*/
  swk = mulshr(IvAD_2, ONE, 2);
  IntAdV->IvInData = mulshr((swk + IntAdV->IvOffset), IntAdP->Kcv, 8 );
#endif

  return;
}

void SetPWM(SHORT src0, SHORT src1, SHORT src2)
{
  PwmT0 = src0;
  PwmT1 = src1;
  PwmT2 = src2;
#ifdef  MULTI_AXIS
```

```
  PwmT0_2 = src0;
  PwmT1_2 = src1;
  PwmT2_2 = src2;
#endif
}

//<2> end
/***************************************** end of file **********************************************/
```