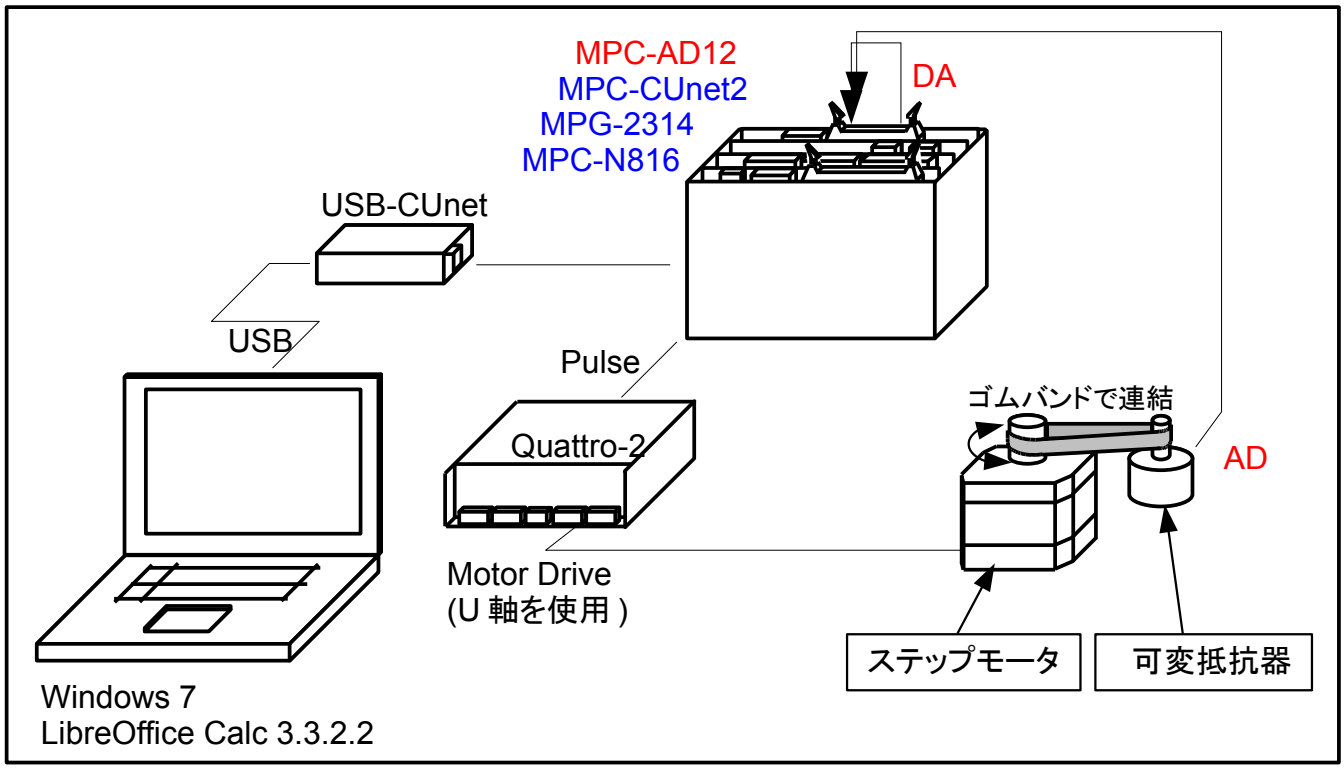
Application Note		資料作成 110518	資料番号 an2k-032
テーマ	MPC-AD12 の AD,DA 変換例		
使用機器	MPC-N816(MPC-2000 シリーズどれでも可)、MPC-AD12、MPG-2314、MPC-CUnet2、USB-CUnet 他		

機器構成



- AD 変換はステップモータを回転させて電圧の変化を調べるという計測器のようなものを想定しています。
- モータと可変抵抗器はゴムバンドで連結されていますが、可変抵抗器の摺動抵抗によりゴムバンドが伸縮するので追従が遅れます。
- MPC のサンプルプログラムは似ていますが、AD 変換をするタスク、モータ回転速度(ACCEL)、パスル数(RMVS)や往復の間隔などが異なります。

(上に乗っているのは弊社マスコット「ポン」君→)

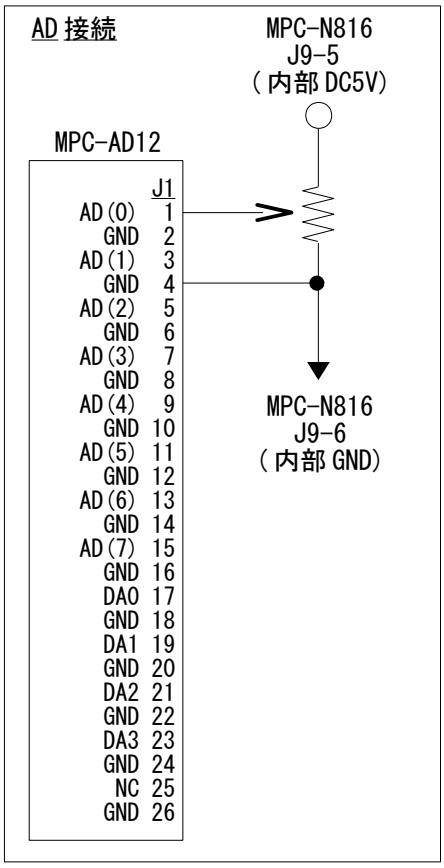
- DA 変換は DA 出力ポートの電圧を AD 入力ポートにループバックして電圧を確認しています。
- AD、DA 変換と CUnet～LibreOffice は直接関係ありません 。LibreOffice は実行結果をグラフ化するために用いています。
- PC と MPC のインターフェースは CUnet で行います。作業手順は LibreOffice から MPC スタート→MPC 動作・完了→LibreOffice へ読込・グラフ作成
- LibreOffice のマクロ、CUnet メールについては次のアプリケーションノートもご参照ください。
  http://departonline.jp/mpc2000/ref/headline/appendix/pdf/an2k-031.pdf
  http://departonline.jp/mpc2000/ref/headline/appendix/pdf/an2k-030.pdf

基本 AD 変換

AD(n)による基本的な AD 変換です。
モータ回転と AD 変換は別々のタスクで行っています。
AD(n)はマルチタスクで同時に使用できます。

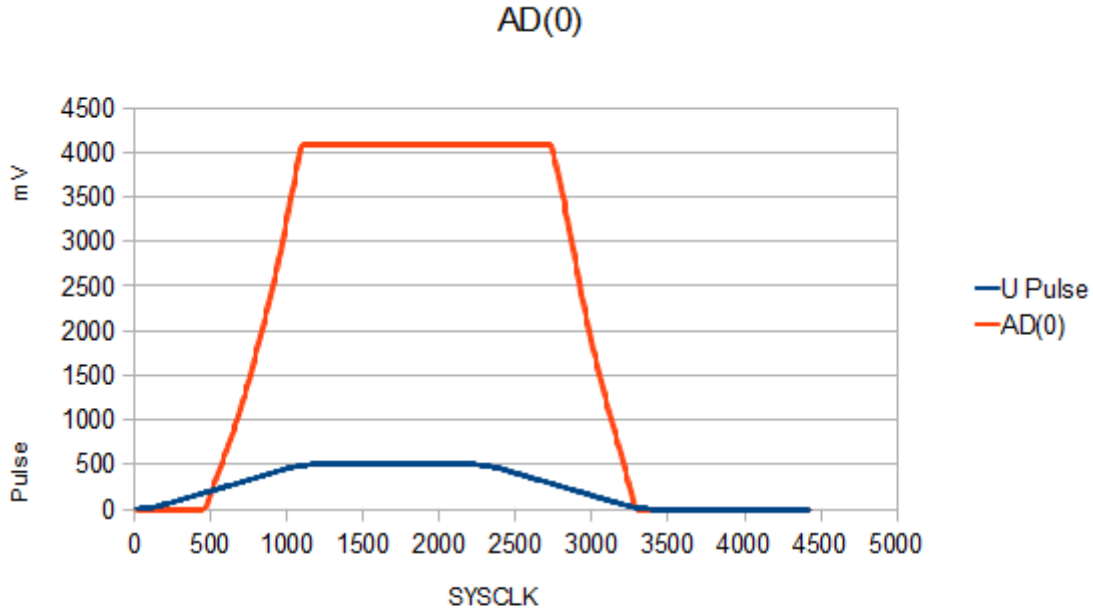
■MPC プログラム

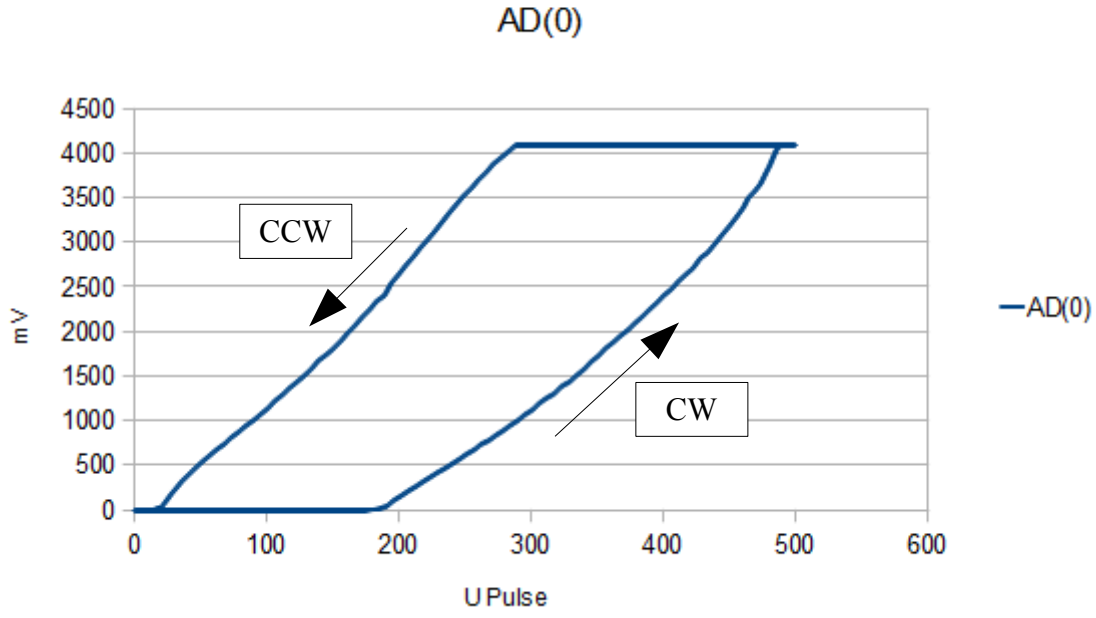
```
  PG 0
  axis=U_A
  ACCEL axis 500
  FEED axis 100
  CLRPOS
  CUNET  12 12 23        /* sets CUNET
  CU_POST 30             /* Cunet mail server
  OFF -1
  TIME 1000

  QUIT 1
  FORK 1 *STORAGE_AD     /* start a recording task

  DO
    PRINT "Wait Start"
    WAIT SW(2000)==1     /* waiting start from the PC application
    ON -1                /* start recording to task1
    GOSUB *ROTATE
    OFF -1               /* stop recording to task1
    TIME 500
    ON 2832              /* finish flag to the PC application
    WAIT SW(2000)==0
    OFF 2832
  LOOP

*ROTATE                  /* ステップモータ回転
  RMVS axis 500
  WAIT RR(axis)==0
  TIME 1000
  RMVS axis -500
  WAIT RR(axis)==0
  TIME 1000
  RETURN

*STORAGE_AD
  PG 0
  DO
    WAIT SW(-1)==1       /* 記録開始待ち
    SYSCLK=0
    sp=1000
    FILL X(1000) 9000 0 /* 点データクリア
    FILL Y(1000) 9000 0
    FILL U(1000) 9000 0
    FILL Z(1000) 9000 0
    OUT 0 2096~Lng

    WHILE SW(-1)==1      /* SW(-1)がオンの間記録を続ける
      X(sp)=SYSCLK       /* 経過時間
      Y(sp)=U(0)         /* U 軸座標
      Z(sp)=AD(0)        /* AD(0)値
      TIME 10            /* 記録間隔
      INC sp
    WEND
    pc=sp-1001
    OUT pc 2096~Lng      /* データ数, This example is 401
    PRINT "point count " pc
  LOOP
```



■結果

上：経過時間に対するAD(mV)とモータ座標(Pulse)の変化、下：モータ座標に対するADの変化です。
ゴムバンドの伸びでモータ回転より可変抵抗器の回転が遅れているのが分かります。
可変抵抗器はDC5VプルアップですがADの最大レンジは4095mVです(標準仕様)。



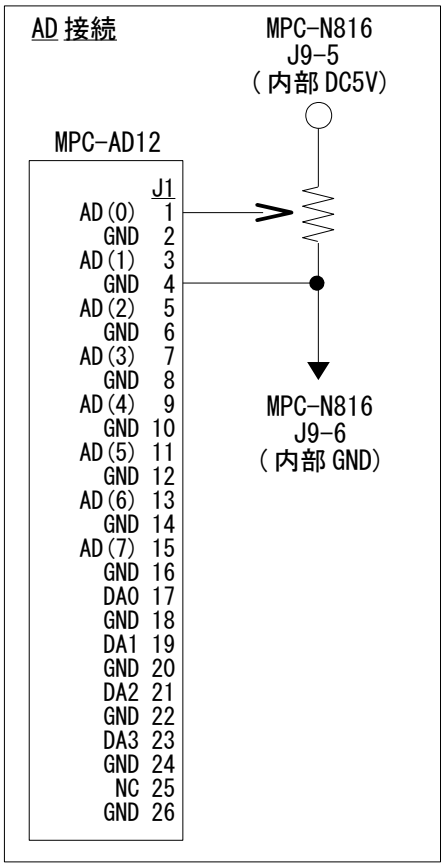


8msec 平均 AD 変換

AD(1,n)による 8msec 毎の平均 AD 変換です。
MPC プログラムも AD(0)が AD(1,0)になっている以外は同じです。
AD(1,n)はマルチタスクで同時に使用できます。

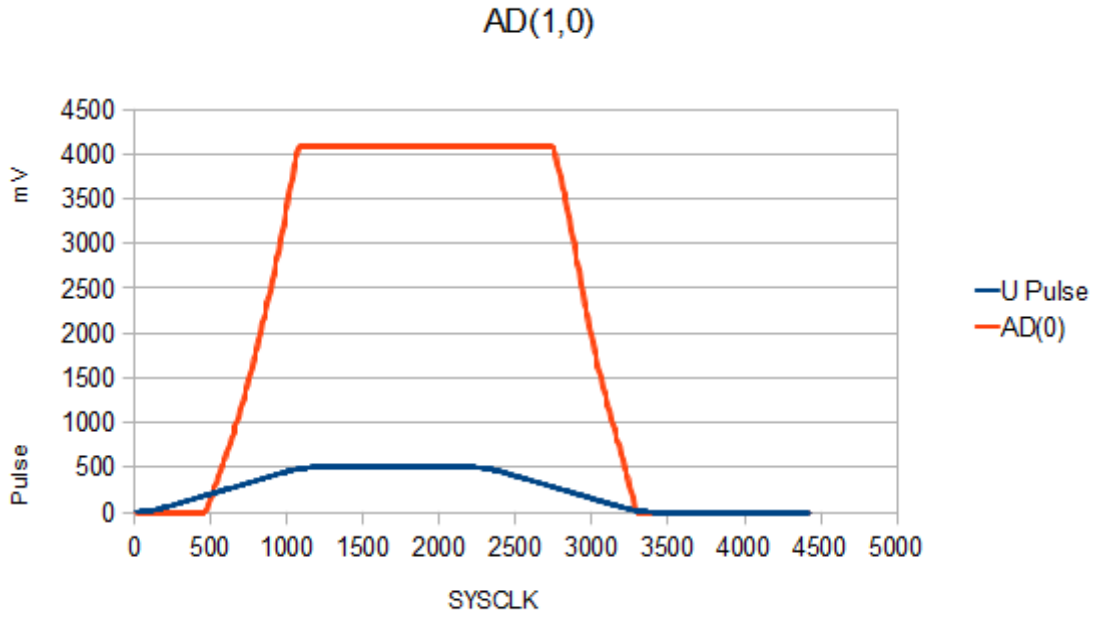
■MPC プログラム

```
  PG 0
  axis=U_A
  ACCEL axis 500
  FEED axis 100
  CLRPOS
  CUNET  12 12 23        /* setup the CUNET Global Memory
  CU_POST 30             /* initiate the Cunet mail server
  OFF -1
  TIME 1000

  QUIT 1
  FORK 1 *STORAGE_AD     /* start a recording task

  DO

    PRINT "Wait Start"
    WAIT SW(2000)==1     /* waiting start from the PC application

    ON -1                /* start recording to task1
    GOSUB *ROTATE
    OFF -1               /* stop recording to task1

    TIME 500
    ON 2832              /* finish flag to the PC application
    WAIT SW(2000)==0
    OFF 2832

  LOOP
  END

*ROTATE
  RMVS axis 500
  WAIT RR(axis)==0
  TIME 1000
  RMVS axis -500
  WAIT RR(axis)==0
  TIME 1000
  RETURN

*STORAGE_AD
  PG 0
  DO
    WAIT SW(-1)==1       /* 記録開始待ち
    SYSCLK=0
    sp=1000
    FILL X(1000) 9000 0 /* 点データクリア
    FILL Y(1000) 9000 0
    FILL U(1000) 9000 0
    FILL Z(1000) 9000 0
    OUT 0 2096~Lng
    WHILE SW(-1)==1      /* SW(-1)がオンの間記録を続ける
      X(sp)=SYSCLK
      Y(sp)=U(0)
      Z(sp)=AD(1,0)      /* AD(0) 8msec 平均値
      TIME 10            /* 記録間隔
      INC sp
    WEND
    pc=sp-1001
    OUT pc 2096~Lng      /* データ数, This example is 401
    PRINT "point count " pc
  LOOP
```



■結果

この実験では前述のAD(0)と変りませんでした。

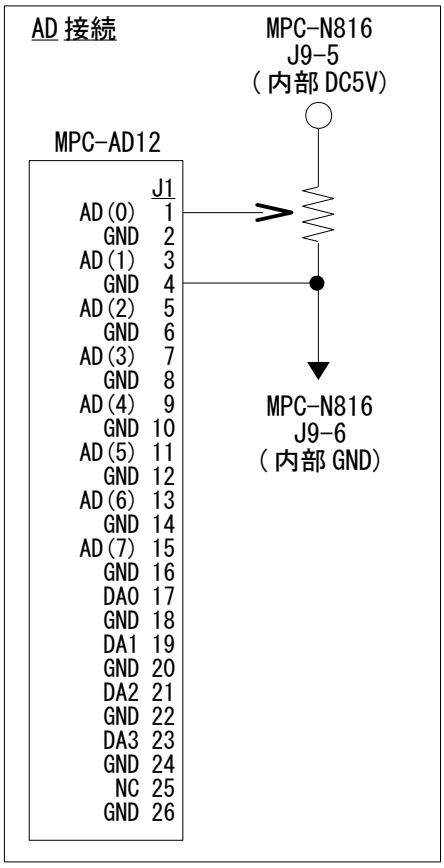


1msec、2msec 連続 AD 変換

AD(2,n)による 1msec 連続 AD 変換、AD(4,n)による 2msec 連続 AD 変換です。
サンプリングが終了するまで a=AD(3,0) で止まります。
モータ回転と AD 変換は別々のタスクで行っています。
他の AD 機能と同時・マルチタスク使用できません。

■MPC プログラム

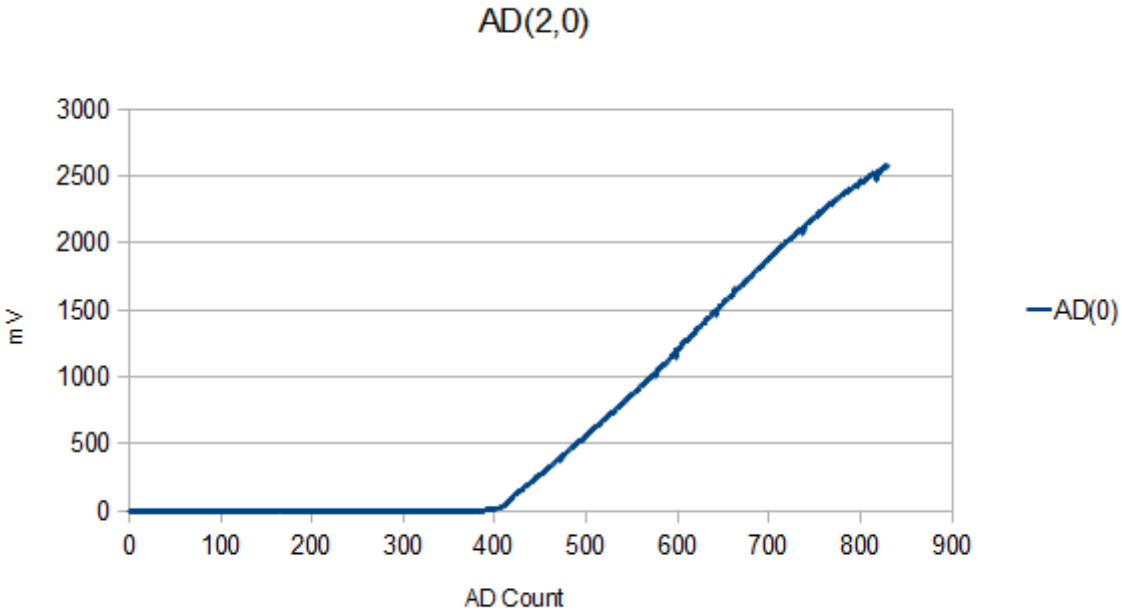
```
  PG 0
  axis=U_A
  ACCEL axis 500
  FEED axis 100
  CLRPOS
  CUNET  12 12 23       /* setup the CUNET Global Memory
  CU_POST 30            /* initiate the Cunet mail server
  OFF -1
  TIME 1000

  QUIT 1
  FORK 1 *AD

  DO

    PRINT "Wait Start"
    WAIT SW(2000)==1    /* waiting start from the PC application

    ON -1
    GOSUB *ROTATE
    OFF -1

    GOSUB *STORAGE_AD

    TIME 500
    ON 2832             /* finish flag to the PC application
    WAIT SW(2000)==0
    OFF 2832

  LOOP
  END

*AD
  DO
    WAIT SW(-1)==1      /* 記録開始待ち from task 0
    PR "AD start"
    SYSCLK=0
    a=AD(2,0)           /* 1 msec * 832 sampling
    'a=AD(4,0)          /* 2 msec * 832 sampling
    a=AD(3,0)           /* Wait for end of AD Conversion
    PR "AD end"  SYSCLK /* SYSCLK: AD(2,0)=831, AD(4,0)=1664
    WAIT SW(-1)==0
  LOOP

*ROTATE
  RMVS axis 300
  WAIT RR(axis)==0
  RMVS axis -300
  WAIT RR(axis)==0
  TIME 1000
  RETURN

*STORAGE_AD
    sp=1000
    FILL X(1000) 9000 0 /* 点データクリア
    FILL Y(1000) 9000 0
    FILL U(1000) 9000 0
    FILL Z(1000) 9000 0
    OUT 0 2096~Lng

    FOR i=0 TO 831
      X(sp)=i
      Y(sp)=AD_D(0,i)
```

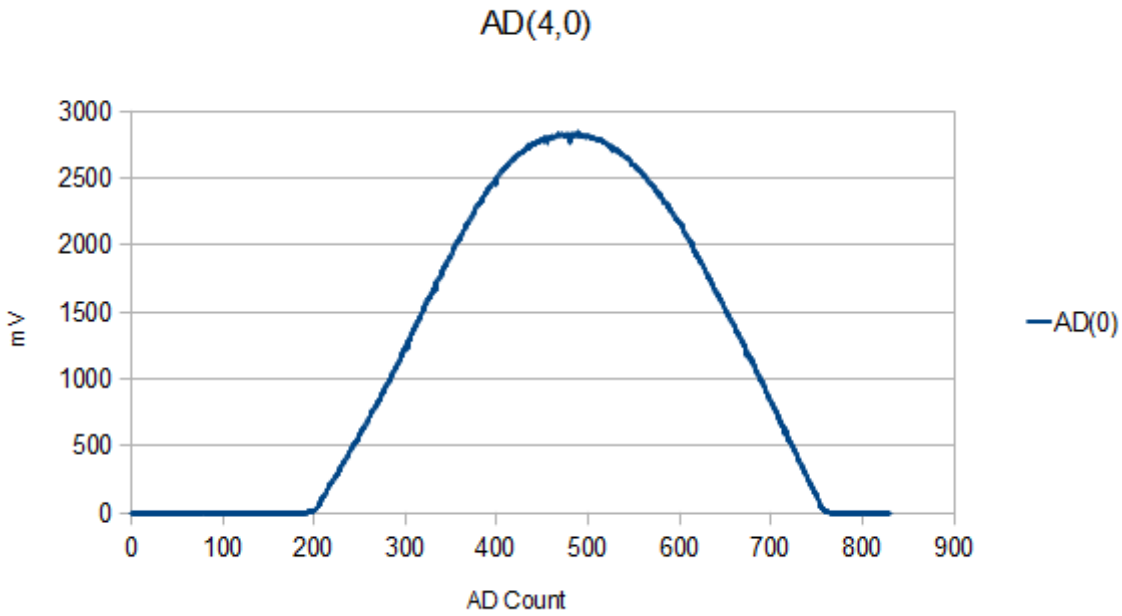


```
      INC sp
    NEXT
    pc=sp-1001
    OUT pc 2096~Lng     /* データ数
    PRINT "point count " pc
  RETURN
```

■結果

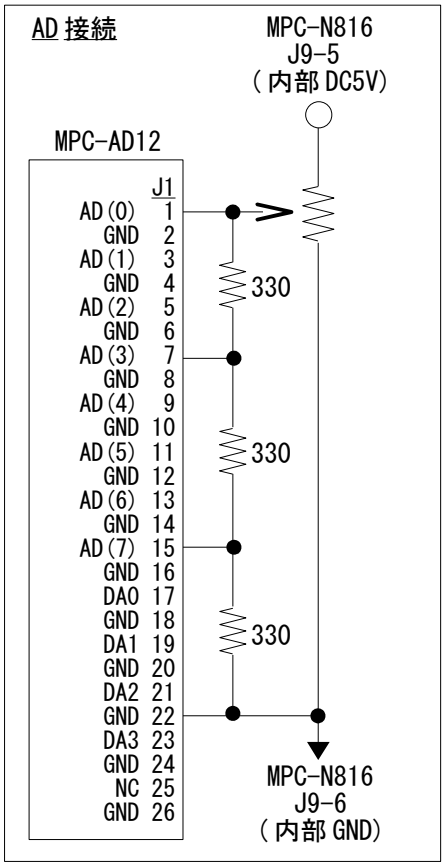
上：1msec 連続変換、1 動作の約半分のデータが取れました。
下：2msec 連続変換。1 動作分のデータが取れました。





8 チャンネルパラレル AD 変換

AD(2,8)による 1msec8 チャンネルパラレル AD 変換です。
AD(0)の電圧をカーボン抵抗で分圧して AD(3)と AD(7)へ入れています。
サンプリングが終了するまで a=AD(3,0) で止まります。
モータ回転と AD 変換は別々のタスクで行っています。
他の AD 機能と同時・マルチタスク使用できません。

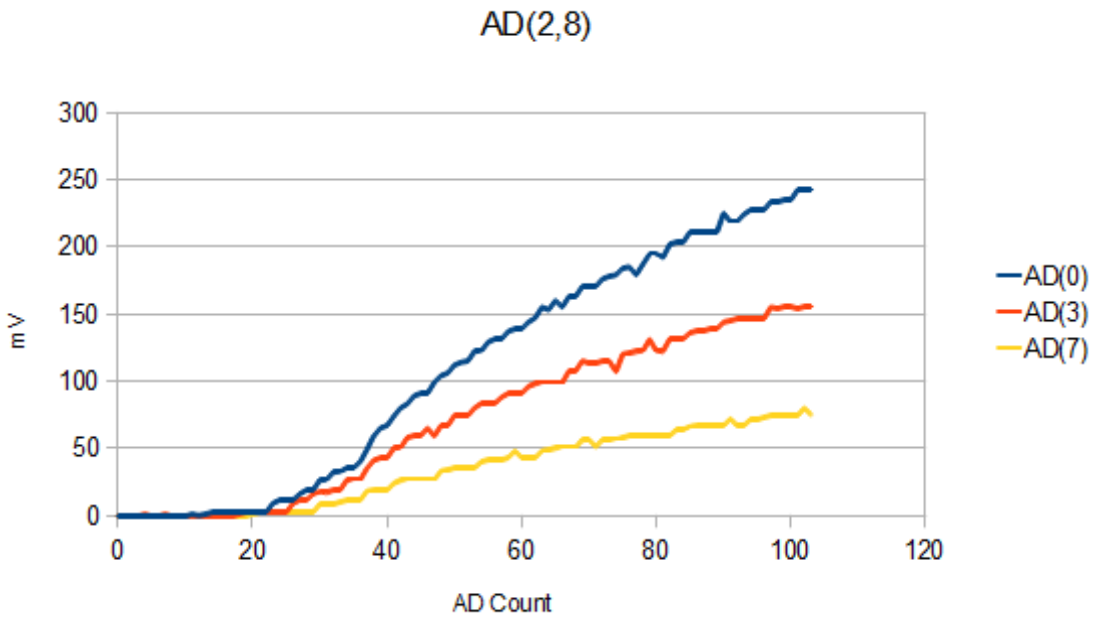


■MPC プログラム

```
  PG 0
  axis=U_A
  ACCEL axis 500
  FEED axis 100
  CLRPOS
  CUNET  12 12 23        /* setup the CUNET Global Memory
  CU_POST 30             /* initiate the Cunet mail server
  OFF -1
  TIME 1000

  QUIT 1
  FORK 1 *AD

  DO

    PRINT "Wait Start"
    WAIT SW(2000)==1     /* waiting start from the PC application

    GOSUB *ROTATE
    GOSUB *STORAGE_AD

    TIME 500
    ON 2832              /* finish flag to the PC application
    WAIT SW(2000)==0
    OFF 2832

  LOOP

*AD
  DO
    WAIT SW(-1)==1       /* 記録開始待ち from task 0
    PR "AD start"
    SYSCLK=0
    a=AD(2,8)            /* 1 msec * 104 * 8ch sampling
    a=AD(3,0)            /* wait for end of AD Conversion
    PR "AD end"  SYSCLK /* SYSCLK: AD(2,8)=106
    WAIT SW(-1)==0
  LOOP

*ROTATE
  RMVS axis 300
  TIME 350
  ON -1
  WAIT RR(axis)==0
  RMVS axis -300
  WAIT RR(axis)==0
  TIME 1000
  OFF -1
  RETURN

*STORAGE_AD
    sp=1000
    FILL X(1000) 9000 0 /* 点データクリア
    FILL Y(1000) 9000 0
    FILL U(1000) 9000 0
    FILL Z(1000) 9000 0
    OUT 0 2096~Lng

    FOR i=0 TO 103
      X(sp)=i
      Y(sp)=AD_D(0,i)    /* AD(0) data
```

```
      Z(sp)=AD_D(3,i)    /* AD(3) data
      U(sp)=AD_D(7,i)    /* AD(7) data
      INC sp
    NEXT
    pc=sp-1001
    OUT pc 2096~Lng      /* データ数
    PRINT "point count " pc
  RETURN
```

■結果

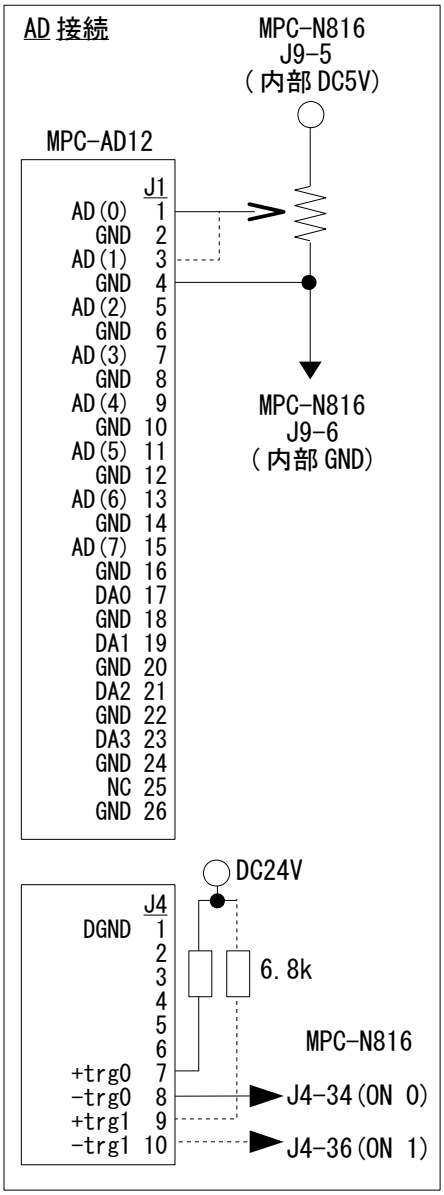


トリガ信号同期 AD 変換

TRG0(または TRG1)によるトリガ信号同期 AD 変換です。
MPC-N816 の出力を PULSE_OUT コマンドで ON/OFF してトリガー信号にしています。
モータ回転と AD 変換は同じタスクで行っています。
他の AD 機能と同時・マルチタスク使用できません。

■MPC プログラム

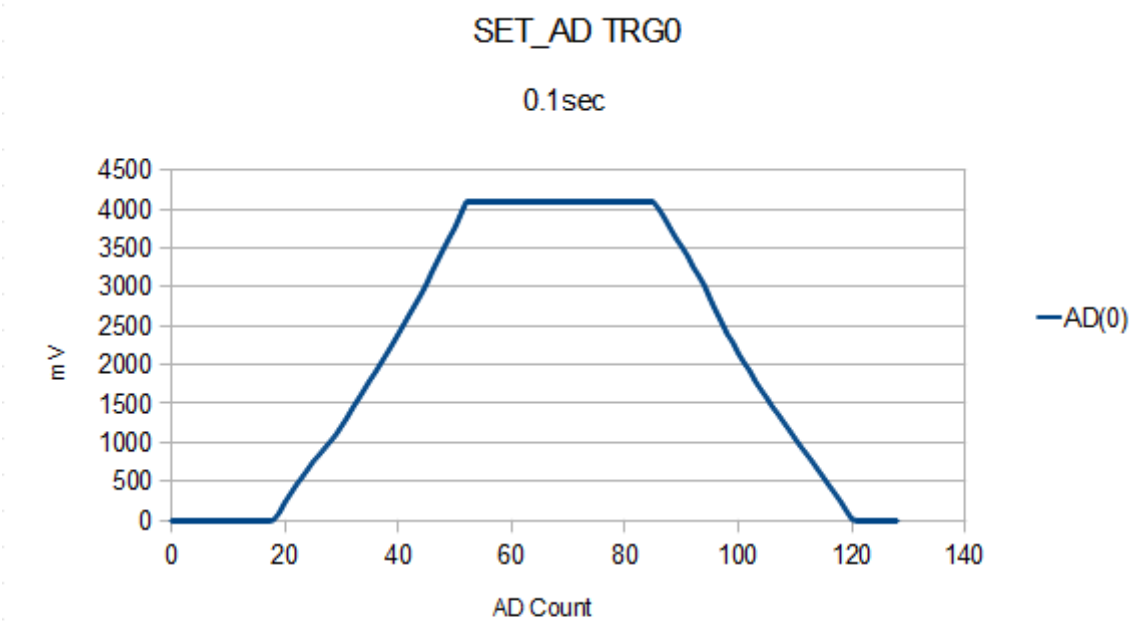
```
  PG 0
  axis=U_A
  ACCEL axis 50
  FEED axis 100
  CLRPOS
  CUNET  12 12 23       /* setup the CUNET Global Memory
  CU_POST 30            /* initiate the Cunet mail server
  OFF -1
  OFF 0
  TIME 1000

  DO

    PRINT "Wait Start"
    WAIT SW(2000)==1    /* waiting start from the PC application

    GOSUB *ROTATE
    GOSUB *STORAGE_AD

    TIME 500
    ON 2832             /* finish flag to the PC application
    WAIT SW(2000)==0
    OFF 2832

  LOOP

*ROTATE
  PR "AD(0)"
  trg_str=TRG0          /* trigger AD start
  trg_end=TRG0_END      /* trigger AD stop
  ad_array=20           /* data array number
  trg_sig=0             /* trigger pulse port
  trg_int=1             /* trigger interval *0.1sec

/*  PR "AD(1)"
/*  trg_str=TRG1
/*  trg_end=TRG1_END
/*  ad_array=21
/*  trg_sig=1
/*  trg_int=2

  SET_AD trg_str        /* AD synchronizes with the TRG0 or TRG1
  PULSE_OUT trg_sig trg_int    /* trigger pulse output #0 or 1 --> TRG0 or TRG1
  RMVS axis 300
  WAIT RR(axis)==0
  RMVS axis -300
  WAIT RR(axis)==0
  PULSE_OUT VOID
  AD_COUNT=AD_D(ad_array,1024) /* the number of sampling
  SET_AD trg_end
  TIME 1000
  PRINT "AD_COUNT" AD_COUNT

  RETURN

*STORAGE_AD
    sp=1000
    FILL X(1000) 9000 0 /* clear the point data area
    FILL Y(1000) 9000 0
    FILL U(1000) 9000 0
    FILL Z(1000) 9000 0
    OUT 0 2096~Lng
```

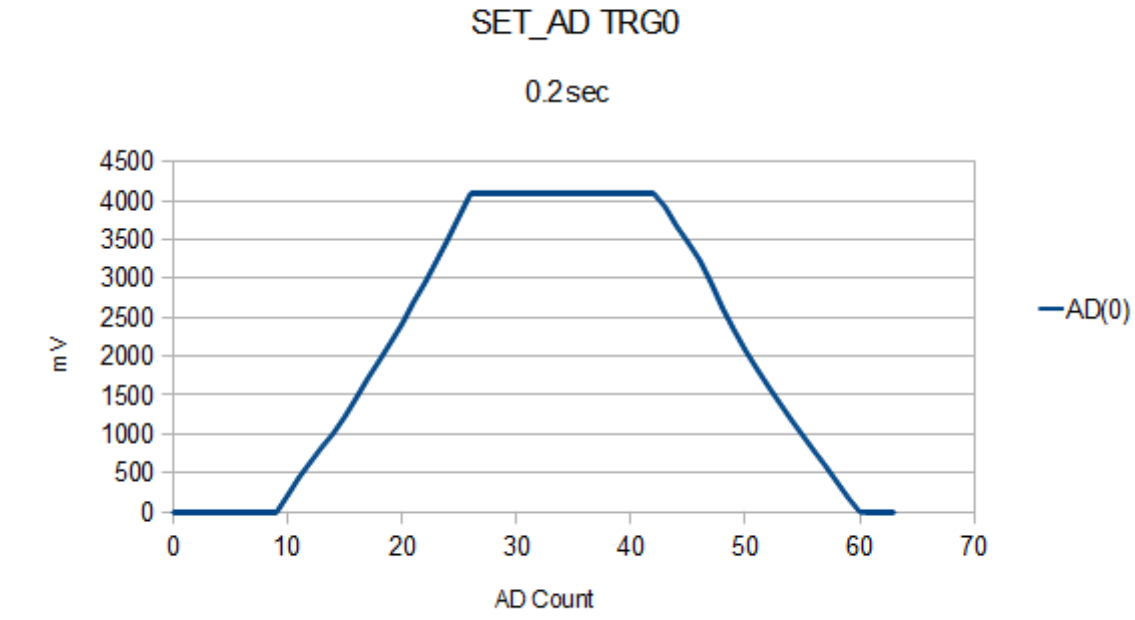


```
    AD_COUNT=AD_COUNT-1
    FOR i=0 TO AD_COUNT
      X(sp)=i
      Y(sp)=AD_D(ad_array,i)     /* data array of TRG0 or TRG1
      INC sp
    NEXT
    OUT AD_COUNT 2096~Lng        /* the number of sampling
  RETURN
```

■結果

上：トリガー間隔0.1sec(PULSE_OUT 0 1)、下：0.2sec(PULSE_OUT 0 2)



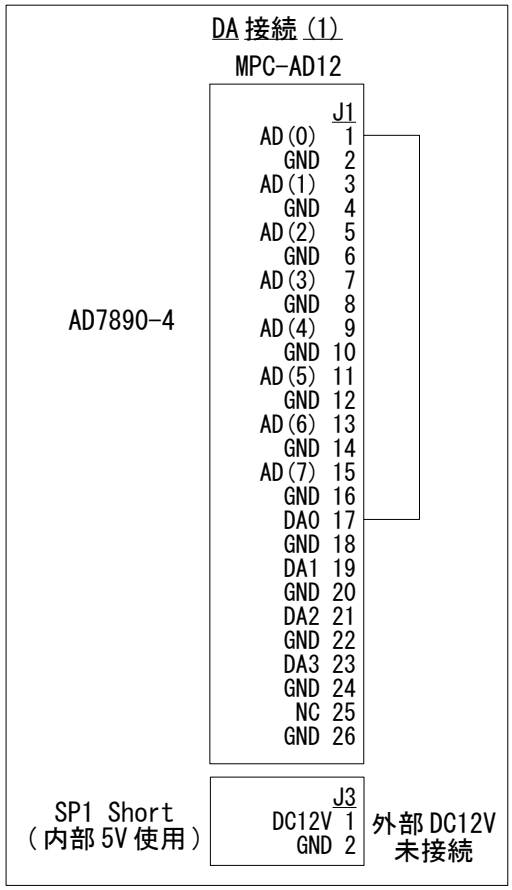


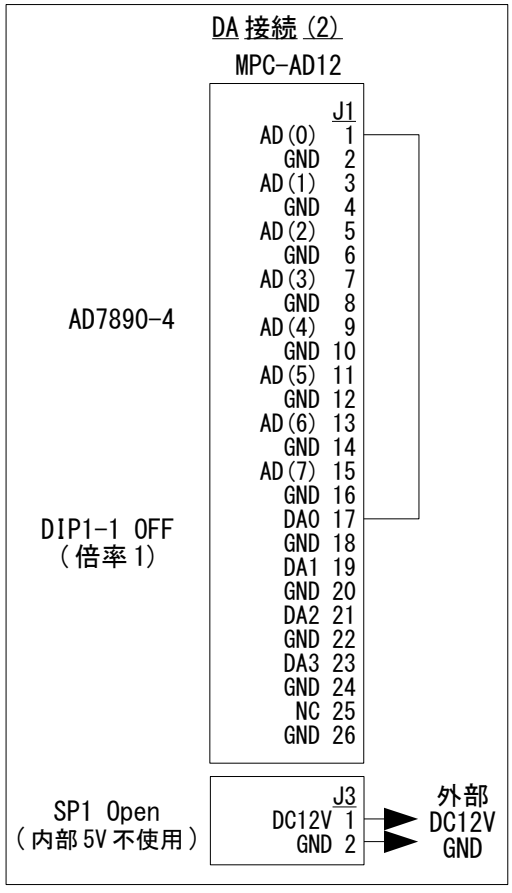
DA 変換 倍率 1 倍

DA0 の出力を AD(0) (標準搭載 AD7890-4)へ接続しています。
電源は内部電源または外部 DC12V です。

■MPC プログラム

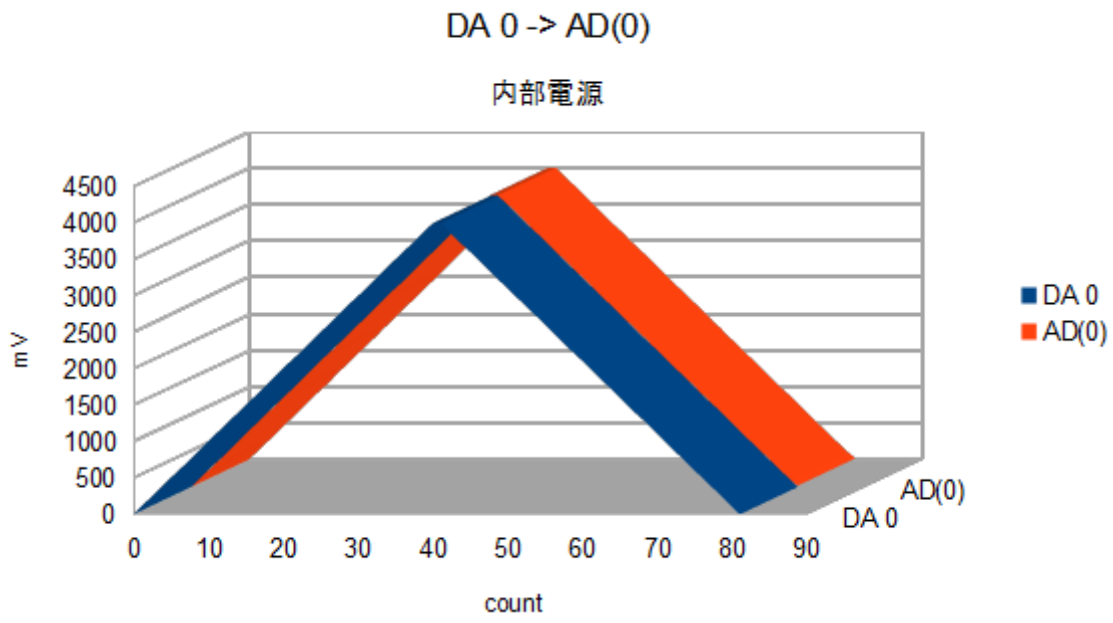
```
CUNET  12 12 23        /* setup the CUNET Global Memory
CU_POST 30             /* initiate the Cunet mail server
TIME 1000

DA 0 0                 /* 0V ch0
DO


  PRINT "Wait Start"
  WAIT SW(2000)==1     /* waiting start from the PC

  sp=1000
  FILL X(1000) 9000 0 /* clear the point data area
  FILL Y(1000) 9000 0
  FILL U(1000) 9000 0
  FILL Z(1000) 9000 0
  OUT 0 2096~Lng

  FOR v=0 TO 4000 STEP 100
    DA v 0             /* v mV ch0
    TIME 10
    X(sp)=sp-1000
    Y(sp)=v
    Z(sp)=AD(0)
    PRINT "count DA AD " sp-1000 v Z(sp)
    INC sp
  NEXT

  FOR v=4000 TO 0 STEP -100
    DA v 0             /* v mV ch0
    TIME 10
    X(sp)=sp-1000
    Y(sp)=v
    Z(sp)=AD(0)
    PRINT "count DA AD " sp-1000 v Z(sp)
    INC sp
  NEXT

  dc=sp-1001           /* データ数
  OUT dc 2096~Lng
  PRINT "data count " dc

  TIME 500
  ON 2832              /* finish to the PC application
  WAIT SW(2000)==0
  OFF 2832

LOOP
```

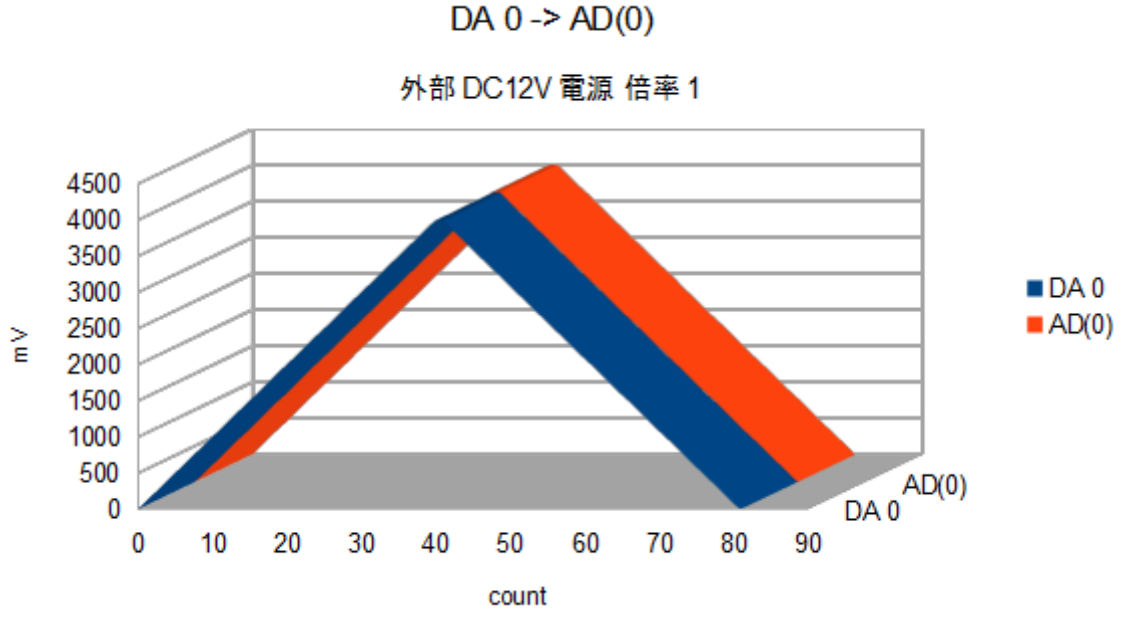




■結果
DA0 は DA コマンドに与えた数値、AD(0)は得られた電圧(mV)です。

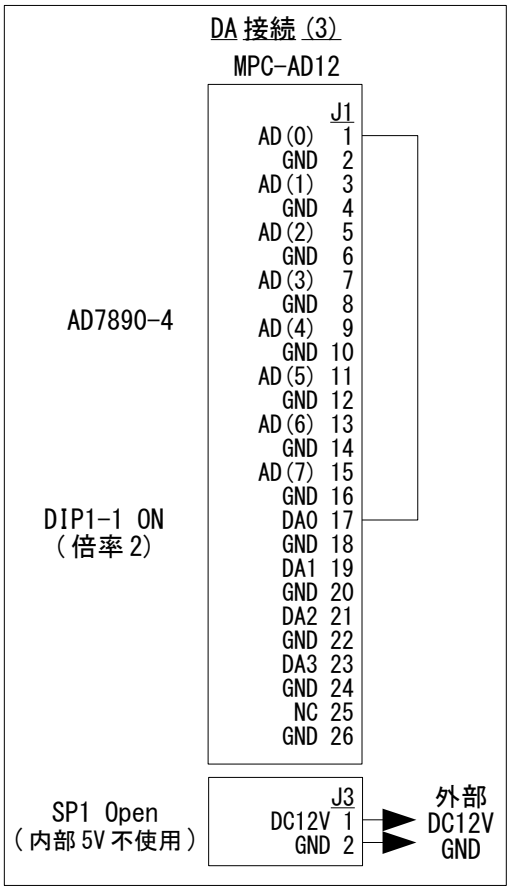
上：DA 接続(1)、下：DA 接続(2)。同じ結果です。



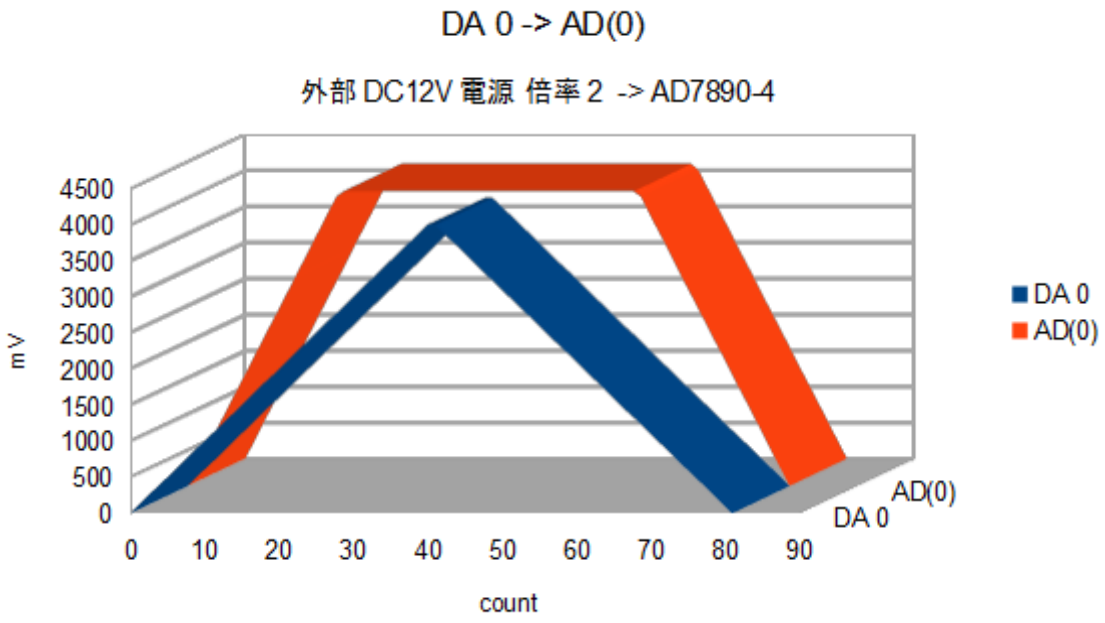


DA 変換 倍率２倍

■MPC プログラム、電源は DA 接続(2)と同じですが
DA の倍率を２倍にしてみました。



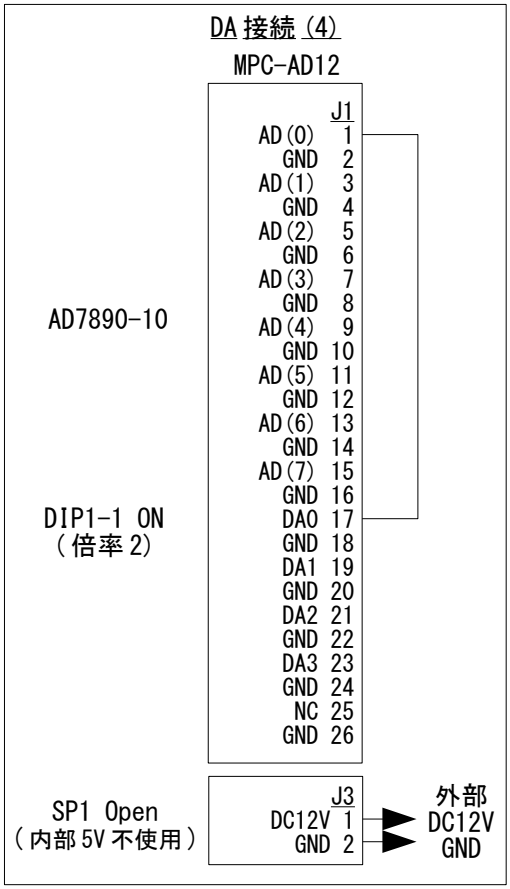
■結果
出力電圧が AD7890-4 の入力レンジを越えてしまいました。



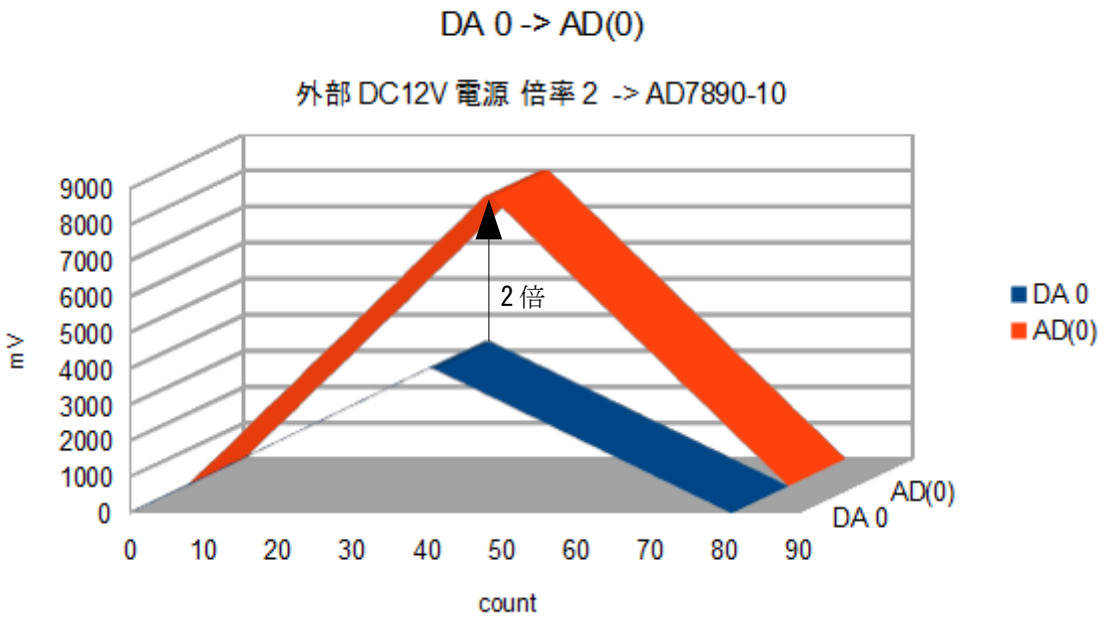
DA 倍率 2 倍で AD コンバータを AD7890-10(±10V)に交換しました。

■MPC プログラム

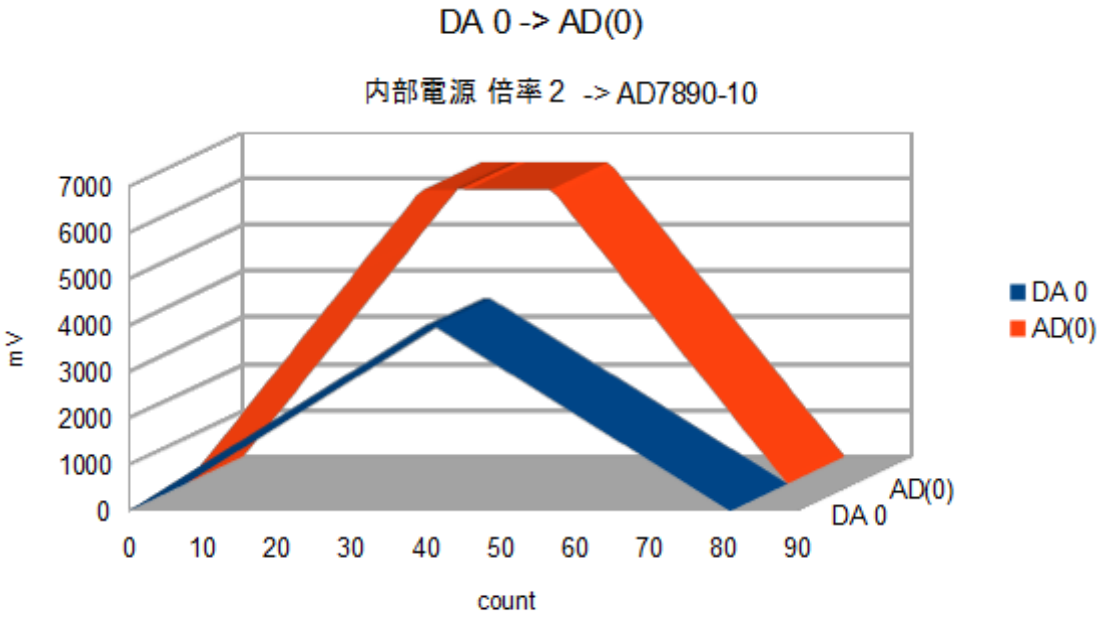
```
CUNET  12 12 23        /* setup the CUNET Global Memory
CU_POST 30             /* initiate the Cunet mail server
TIME 1000

SET_AD AD7890_10       /* using AD7890-10 minus voltage
DA 0 0                 /* 0V ch0
DO


  PRINT "Wait Start"
  WAIT SW(2000)==1     /* waiting start from the PC

  sp=1000
  FILL X(1000) 9000 0 /* clear the point data area
  FILL Y(1000) 9000 0
  FILL U(1000) 9000 0
  FILL Z(1000) 9000 0
  OUT 0 2096~Lng

  FOR v=0 TO 4000 STEP 100
    DA v 0             /* v*2 mV ch0
    TIME 10
    X(sp)=sp-1000
    Y(sp)=v
    Z(sp)=AD(0)*10000/2047
    PRINT "count DA AD " sp-1000 v Z(sp)
    INC sp
  NEXT

  FOR v=4000 TO 0 STEP -100
    DA v 0             /* v*2 mV ch0
    TIME 10
    X(sp)=sp-1000
    Y(sp)=v
    Z(sp)=AD(0)*10000/2047
    PRINT "count DA AD " sp-1000 v Z(sp)
    INC sp
  NEXT

  dc=sp-1001           /* データ数
  OUT dc 2096~Lng
  PRINT "data count " dc

  TIME 500
  ON 2832              /* finish flag to the PC application
  WAIT SW(2000)==0
  OFF 2832

LOOP
```



■結果
期待通りの電圧出力が確認できました。

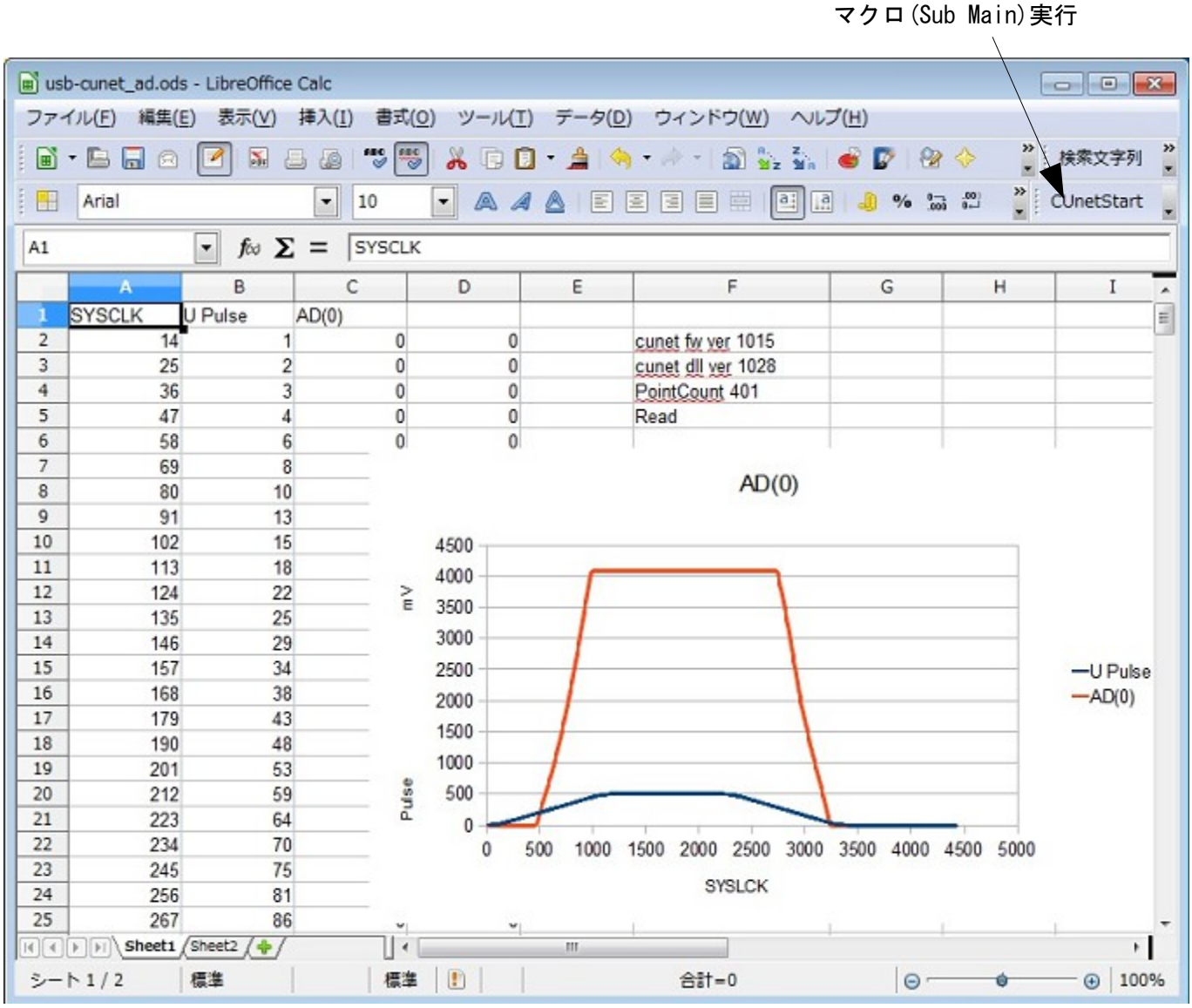


ちなみに、内部電源では倍率 2 倍でもフルレンジ出ません(Max 約 6.5V)。



LibreOffice Calcのマクロ

■LibreOffice Calc実行時の例
- マクロでデータ読み込みからグラフ作成まで連続で行います。
- MS-Excelでも同じことができると思います。
- Panasonic Let's note CF-F10 Windows7 Professional (x64) でもうまく動きました。
  (LibreOffice と usbcunet.dll は32ビットです)




```
REM  *****  BASIC  *****
Option Explicit

'usbcunet.dll関数宣言
Declare Function cunet_usb_open Lib "usbcunet.dll" () As Long
Declare Function cunet_fw_ver Lib "usbcunet.dll" () As Long
Declare Function cunet_dll_ver Lib "usbcunet.dll" () As Long
Declare Sub cunet_init Lib "usbcunet.dll" _
    (ByVal sa As Long, ByVal ow As Long, ByVal en As Long)
Declare Function cunet_sw Lib "usbcunet.dll" _
    Alias "cunet_sw_s" (ByVal adr As Long) As Long
Declare Sub cunet_on Lib "usbcunet.dll" _
    Alias "cunet_on_s" (ByVal adr As Long)
Declare Sub cunet_off Lib "usbcunet.dll" _
    Alias "cunet_off_s" (ByVal adr As Long)
Declare Function cunet_in Lib "usbcunet.dll" _
    Alias "cunet_in_s" (ByVal adr As Long, ByVal siz As Long) As Long
Declare Sub cunet_out Lib "usbcunet.dll" _
    Alias "cunet_out_s" (ByVal dat As Long, ByVal adr As Long, ByVal siz As Long)
Declare Function cunet_req_pnt Lib "usbcunet.dll" _
    Alias "cunet_req_pnt_s" (ByVal req_sa As Long, ByVal ar_top As Long, ByRef rcv_ar as Variant) As
```

```
Long

Public Const CuChr = 1    '1byte符号無
Public Const CuInt = 2    '2Byte符号付
Public Const CuWrd = 4    '2Byte符号無
Public Const CuLng = 8    '4Byte符号付

sub Main

  call Activesheet_1
  call Chart_Clear

  call Cell_Clear
  call Cell_Clear '既存のグラフがセレクト状態(四隅に□)の時、1回ではデータがクリアされないので2回やる害はないようだ。

  call CunetOpen
  call XyzStart
  call Chart_Draw

end sub

sub CunetOpen
  If cunet_usb_open <> 1 Then
    ThisComponent.Sheets(0).getCellByPosition(5,1).String="CUnet Open Error"
    End
  End If

  ThisComponent.Sheets(0).getCellByPosition(5,1).String="cunet fw ver " + Cstr(cunet_fw_ver)
  ThisComponent.Sheets(0).getCellByPosition(5,2).String="cunet dll ver " + Cstr(cunet_dll_ver)

  cunet_init 255, 0, 0  'initialize usbcunet
  Wait 500
  cunet_init 0,12,23  'sets usbcunet SA , Global memory area
  Wait 500

end sub

sub XyzStart

  Dim sa_adr , p_adr , ar(60) as long

  Dim ln , rw , j
  Dim PointCount, BlockCount

  cunet_on 2000                      ' start MPC action
  Do While cunet_sw(2832)<>1         ' wait for finish
    Wait 1
  Loop

  cunet_off 2000
  Do While cunet_sw(2832)<>0
    Wait 1
  Loop

  PointCount=cunet_in(2096,CuLng)
  ThisComponent.Sheets(0).getCellByPosition(5,3).String="PointCount " + Cstr(PointCount)
  BlockCount=PointCount/15+1

  ThisComponent.Sheets(0).getCellByPosition(0,0).String="SYSCLK"
  ThisComponent.Sheets(0).getCellByPosition(1,0).String="U Pulse"
  ThisComponent.Sheets(0).getCellByPosition(2,0).String="AD(0)"
  'ThisComponent.Sheets(0).getCellByPosition(3,0).String="AD(3)"

  sa_adr=12                          ' the number of reqest SA
  p_adr=1000                         ' reading the point number

  ln=1                               ' line number
  For j=1 To BlockCount
    cunet_req_pnt sa_adr, p_adr, ar
    For rw=0 to 59 Step 4
      IF ln <= PointCount+1 Then
        ThisComponent.Sheets(0).getCellByPosition(0,ln).value=ar(rw)
        ThisComponent.Sheets(0).getCellByPosition(1,ln).value=ar(rw+1)
        ThisComponent.Sheets(0).getCellByPosition(2,ln).value=ar(rw+2)
```

```
        ThisComponent.Sheets(0).getCellByPosition(3,ln).value=ar(rw+3)
        ThisComponent.Sheets(0).getCellByPosition(5,4).String=Cstr(ln)
      End IF
      ln=ln+1
    Next rw
    p_adr=p_adr+15
  Next j

end sub


Sub clearContents

  Dim oRange as Object
  Dim oSheet as Object
  oSheet = thisComponent.Sheets.getByName("Sheet1")
  oRange = oSheet.getCellRangeByName("A1:C100")
  oRange.clearContents(com.sun.star.sheet.CellFlags.STRING)
  oRange.clearContents(com.sun.star.sheet.CellFlags.VALUE)

End Sub

Sub Cell_Clear

  rem ----------------------------------------------------------------------
  rem define variables
  dim document   as object
  dim dispatcher as object
  rem ----------------------------------------------------------------------
  rem get access to the document
  document   = ThisComponent.CurrentController.Frame
  dispatcher = createUnoService("com.sun.star.frame.DispatchHelper")

  rem ----------------------------------------------------------------------
  dispatcher.executeDispatch(document, ".uno:SelectAll", "", 0, Array())

  wait 100

  rem ----------------------------------------------------------------------
  dim args2(0) as new com.sun.star.beans.PropertyValue
  args2(0).Name = "Flags"
  args2(0).Value = "SVDFN"

  dispatcher.executeDispatch(document, ".uno:Delete", "", 0, args2())

  wait 100

  rem ----------------------------------------------------------------------
  dim args3(0) as new com.sun.star.beans.PropertyValue
  args3(0).Name = "ToPoint"
  args3(0).Value = "$A$1"

  dispatcher.executeDispatch(document, ".uno:GoToCell", "", 0, args3())

End Sub


Sub Goto_Cell(Byval c As String )

  dim document   as object
  dim dispatcher as object
  document   = ThisComponent.CurrentController.Frame
  dispatcher = createUnoService("com.sun.star.frame.DispatchHelper")

  dim args1(0) as new com.sun.star.beans.PropertyValue
  args1(0).Name = "ToPoint"
  args1(0).Value = c
  dispatcher.executeDispatch(document, ".uno:GoToCell", "", 0, args1())

End Sub

Sub Chart_Clear

  Dim oCharts As Object
  Dim oTitle As String
```

```
  Dim i , j

  oCharts = ThisComponent.Sheets.getByName("Sheet1").Charts

  If oCharts.Count>0 Then
    i= oCharts.Count
    for j=1 to i
      oTitle= oCharts.ElementNames(0)  ' グラフ名取得
        'MsgBox oTitle
        oCharts.removeByName(oTitle)
      next j
  End If

End Sub

Sub Activesheet_1

  Dim oDoc as Object
  Dim oController as Object
  Dim oSheets as Object

  oDoc = ThisComponent
  oController = oDoc.getCurrentController()
  oSheets = oDoc.getSheets()
  oController.setActiveSheet(oSheets.getByName("Sheet1"))

End Sub

Sub Chart_Draw

  Goto_Cell "$A$2"

  Dim Ec
  Dim Rect As New com.sun.star.awt.Rectangle
  Dim RangeAddress(0) As New com.sun.star.table.CellRangeAddress

  ' グラフ位置
  Rect.X = 10000
  Rect.Y = 3000
  ' グラフサイズ
  Rect.Width = 15000
  Rect.Height = 10600

  ' 範囲
  RangeAddress(0).Sheet = 0
  RangeAddress(0).StartColumn = 0
  RangeAddress(0).StartRow = 1
  RangeAddress(0).EndColumn = 2
  Ec = Cint(ThisComponent.Sheets(0).getCellByPosition(5,4).string)
  RangeAddress(0).EndRow = Ec
  wait 2000       ' need

  ThisComponent.Sheets(0).Charts.addNewByName("datagraph", Rect, RangeAddress(), False, False)
  ' 系列名 (凡例)
  ThisComponent.Sheets(0).Charts.getByName("datagraph").embeddedObject().Data.setColumnDescriptions(Ar
ray("x","U Pulse","AD(0)")
  ' 散布図
  ThisComponent.Sheets(0).Charts.getByName("datagraph").embeddedObject.Diagram _
                                                                                =
ThisComponent.Sheets(0).Charts.getByName("datagraph").embeddedObject.createInstance("com.sun.star.char
t.XYDiagram")
  ' データポイント無し
   ThisComponent.Sheets(0).Charts.getByName("datagraph").embeddedObject.Diagram.SymbolType =-3   ' no
points
  ' 一本目の系列
  ThisComponent.Sheets(0).Charts.getByName("datagraph").embeddedObject.Diagram.getDataRowProperties(1)
.LineColor = RGB(0,0,255)
  ThisComponent.Sheets(0).Charts.getByName("datagraph").embeddedObject.Diagram.getDataRowProperties(1)
.LineWidth = 50
  ' 二本目の系列
  ThisComponent.Sheets(0).Charts.getByName("datagraph").embeddedObject.Diagram.getDataRowProperties(2)
.LineColor = RGB(255,0,0)
  ThisComponent.Sheets(0).Charts.getByName("datagraph").embeddedObject.Diagram.getDataRowProperties(2)
.LineWidth = 50
  ' x 軸ラベル
```

```
  ThisComponent.Sheets(0).Charts.getByName("datagraph").embeddedObject.Diagram.HasXAxisTitle = True
        ThisComponent.Sheets(0).Charts.getByName("datagraph").embeddedObject.Diagram.XAxisTitle.String
="SYSCLK"
  ' y 軸ラベル
  ThisComponent.Sheets(0).Charts.getByName("datagraph").embeddedObject.Diagram.HasYAxisTitle = True
        ThisComponent.Sheets(0).Charts.getByName("datagraph").embeddedObject.Diagram.YAxisTitle.String
="Pulse        mV"
  ' タイトル
  ThisComponent.Sheets(0).Charts.getByName("datagraph").embeddedObject.HasMainTitle = True
  ThisComponent.Sheets(0).Charts.getByName("datagraph").embeddedObject.Title.String ="AD(0)"

End Sub
```

MPC、MPC-AD12 バージョン

■サンプル作成に用いた MPC と MPC-AD12(CEP-125F) のバージョン

```
#VER
MPC-1000(SH7030) BL/I 1.12_70 2011/05/09
 All Rights reserved. ACCEL Corp. .T32
  [* and / performed before + or -]
#PRX AD(-1)
20110323      ← MPC-AD12 ファームウェアバージョン
```

GET_AD コマンド　(2015/01/26 追記)

(MPC-1200H BL/I 1.14_34 2015/01/09 with MPC-AD12(CEP-125F))

GET_AD コマンドはサンプリング数、サンプリング間隔を任意に設定できます。取得したデータは配列に格納されるのでデータの確認、後処理が楽です。

■基本書式

```
GET_AD Ch Array Cnt [ms]
       Ch: A/D 入力チャンネル番号
       Array: 格納する配列
       Cnt: サンプリング数
       ms:サンプリング間隔
```

■サンプルプログラム

```
SYSCLK=0
FILL X(100) 100 0          /* 配列初期化
GET_AD 0 X(100) 100 2      /* データ取得 ※1
WAIT GET_AD(0)==0          /* 終了待ち
ad=0
FOR i=100 TO 199
  ad=ad+X(i)
NEXT
average=ad/100
PR "[average elapse]" average SYSCLK
```

実行すると X(100)～X(199)にデータが格納されます。サンプルプログラムでは平均値を計算します。

```
P(99)    X= 0   Y= 0   U= 0   Z= 0
P(100)   X= 1281   Y= 0   U= 0   Z= 0
P(101)   X= 1273   Y= 0   U= 0   Z= 0
(中略)
P(198)   X= 1275   Y= 0   U= 0   Z= 0
P(199)   X= 1275   Y= 0   U= 0   Z= 0
P(200)   X= 0   Y= 0   U= 0   Z= 0
```

■実行結果

```
※1 のサンプリング間隔を省略した時 (GET_AD 0 X(100) 100)
RUN
# [average elapse] 1274 104

※1 のサンプリング間隔を 2msec にしたとき　(GET_AD 0 X(100) 100 2)
RUN
# [average elapse] 1274 204
```

- 複数のチャンネルを同時に取得することもできます→コマンドリファレンス参照。
- 関連資料 「GET_AD コマンドのマルチタスクへの影響」
  http://departonline.jp/mpc2000/ref/headline/appendix/pdf/get_ad_multitask.pdf