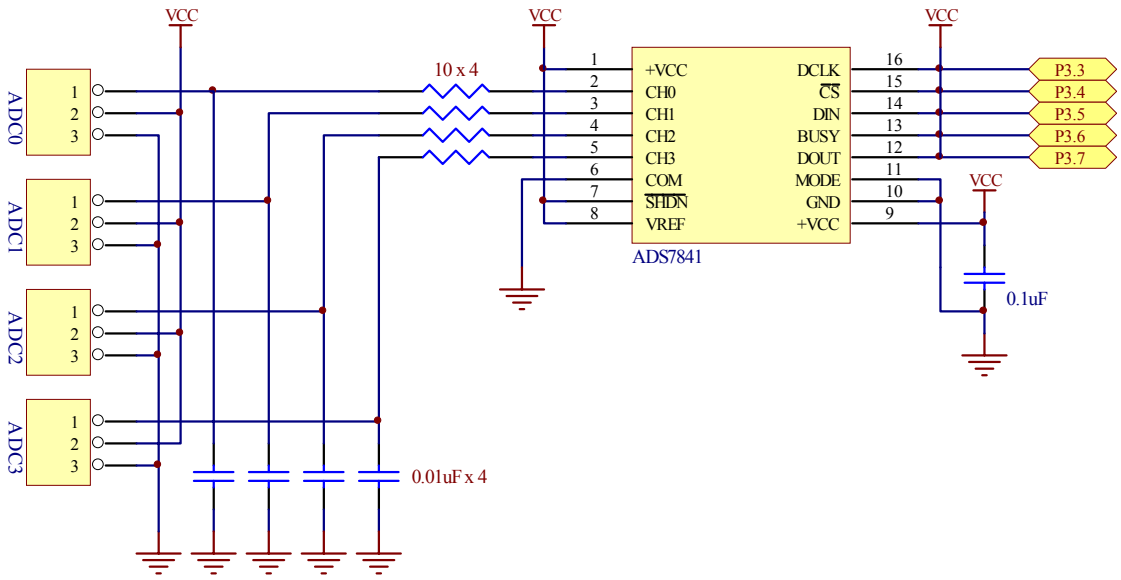
การตรวจสอบ Input แบบ Analog (A/D)

ระบบ Input / Output ของ ไมโครคอนโทรลเลอร์ P89C51RD2 ของ Philips นั้น ตามปรกติแล้วจะเป็นแบบ Logic TTL ซึ่งสามารถเชื่อมต่อโดยตรงกับ Input / Output แบบ TTL เท่านั้น แต่ในการประยุกต์ใช้งานบางอย่าง อาจมีความจำเป็นต้องติดตั้งอุปกรณ์ตรวจจับ ซึ่งให้ Output การทำงานเป็นสัญญาณแบบ Analog นั้น ตามปรกติแล้วจะไม่สามารถเชื่อมต่อสัญญาณแบบ Analog เข้ากับ CPU ได้โดยตรง ดังนั้นจำเป็นต้องจัดหาอุปกรณ์ที่จะทำหน้าที่เป็นสื่อกลางในการรับค่าสัญญาณ Input แบบ Analog เพื่อส่งต่อให้ CPU ได้รับรู้ โดยสำหรับบอร์ดควบคุมของ ET-ROBOT-RD2 นั้น ได้จัดเตรียมวงจรสำหรับทำหน้าที่แปลงสัญญาณ Input แบบ Analog ให้เป็นสัญญาณข้อมูลแบบดิจิตอลขนาด ความละเอียด 12 บิต จำนวน 4 ช่อง Input โดยใช้ชิพ A/D สำเร็จรูป เบอร์ ADS7841 เป็นตัวแปลงสัญญาณ

โดยวงจรการเชื่อมต่อของ ADS7841 กับไมโครคอนโทรลเลอร์ P89C51RD2 ในบอร์ดควบคุมรถหุ่นยนต์ ET-ROBOT-RD2 นั้น เป็นดังรูป



รูปแสดง วงจร A/D ที่ใช้กับบอร์ด ET-ROBOT-RD2

โดยวงจร A/D นี้จะสามารถรับสัญญาณ Analog ได้ระหว่าง 0-5V จำนวน 4 ช่องสัญญาณ โดยค่าผลลัพธ์ของแต่ละช่องจะมีค่าความละเอียดขนาด 12บิต (0-4095 Step) เช่น ถ้า Analog Input มีขนาด 0V จะได้ผลลัพธ์เป็น 0 (0000H) แต่ถ้า Analog Input มีขนาด 2.5V จะได้ผลลัพธ์เป็น 2047 (07FFH) หรือถ้าสัญญาณ Analog Input มีขนาดเป็น 5V ก็จะได้ผลลัพธ์เป็น 4095 (0FFFH) ดังนี้เป็นต้น

โดยจะเห็นได้ว่าลักษณะของสัญญาณของ Analog Input นั้นจะรับเป็นแบบแรงดันเท่านั้น โดยค่าผลลัพธ์ที่ได้จากการแปลงค่า Analog Input ก็จะได้เป็นค่า Step ดังนั้นถ้าต้องการนำค่าผลลัพธ์ที่อ่านได้จาก A/D มาทำงานแปลงเป็นค่าแบบแรงดัน ก็จะต้องทำการนำค่า Step ที่ได้มาทำการคำนวณเพื่อเปลี่ยนจากค่า Step เป็นค่าแรงดันเสียก่อนดังนี้

ขนาดสูงสุดของสัญญาณ Analog Input	=	5V
ขนาดผลลัพธ์สูงสุดของการแปลงข้อมูลจาก A/D	=	4095
ค่าแรงดันใน 1 Step ของผลลัพธ์	=	5V / 4095
	=	0.001221001 V

ดังนั้นเมื่อต้องการเปลี่ยนค่าผลลัพธ์ที่อ่านได้จาก A/D ให้เป็นค่าแรงดัน จะต้องนำค่าผลลัพธ์นั้นไปคูณด้วย 0.001221001V เพื่อให้ได้ผลลัพธ์เป็นค่าแรงดันที่ต้องการ

สำหรับการสั่งอ่านค่าข้อมูลจาก A/D ของ ADS7841 ด้วยโปรแกรม BASCOM-8051 นั้น ตามปรกติแล้วจะยังไม่มีคำสั่งสำหรับสั่งอ่านค่าผลลัพธ์จาก ADS7841 ไว้ให้ใช้งานโดยตรง ซึ่งจำเป็นที่เราจะต้องสร้าง Library คำสั่งขึ้นมาใช้งานเอง ซึ่งถ้าต้องการให้การทำงานของโปรแกรมมีความเร็วและเกิดประสิทธิภาพสูงสุดนั้น จะต้องสร้าง Library คำสั่งด้วยภาษาแอสแซมบลี้ เพราะถ้าสร้างเป็น Library คำสั่งโดยใช้ภาษาเบสิกเองจะได้การทำงานที่ช้าลงไปกว่าปรกติอยู่พอสมควร

สำหรับในที่นี้ผู้เขียนจะขอสร้างเป็น Library คำสั่งไว้ให้เรียกใช้งานกันเลย โดยผู้อ่านไม่จำเป็นต้องทำความเข้าใจกับการทำงานของภาษาแอสแซมบลี้ก็ได้ เพียงแต่ทำความเข้าใจในข้อกำหนดและวิธีการเรียกใช้งานคำสั่งใน Library ก็เพียงพอแล้ว โดย Library ที่จะสร้างขึ้นใช้งานกับ ADS7841 ของบอร์ด ET-ROBOT-RD2 จะมีข้อกำหนดดังนี้

- กำหนดชื่อสัญญาณที่เชื่อมต่อกับ DCLK ของ ADS7841 เป็น ADS7841_DCLK
- กำหนดชื่อสัญญาณที่เชื่อมต่อกับ CS ของ ADS7841 เป็น ADS7841_CS
- กำหนดชื่อสัญญาณที่เชื่อมต่อกับ DIN ของ ADS7841 เป็น ADS7841_DIN
- กำหนดชื่อสัญญาณที่เชื่อมต่อกับ DOUT ของ ADS7841 เป็น ADS7841_DOUT
- ต้องสร้างตัวแปรชนิด Byte เพื่อรองรับการทำงานของคำสั่งในการผ่านค่า Channel ของ A/D ที่ต้องการจะอ่านค่า
- ต้องสร้างตัวแปรชนิด Word เพื่อรองรับการทำงานของคำสั่งในการผ่านค่าผลลัพธ์ของ A/D

```
[_ADS7841]
;Input  : R0 ^Variable Save Data(Low of Word)
;       : R1 = Channel(0-3)
;Output : Save Data to Variable
_ADS7841:
     SETB      {ADS7841_CS}        ;Initial CS
     CLR       {ADS7841_DCLK}      ;Initial DCLK

     MOV       A,R1                ;Get Channel
     CJNE      A,#0,_CH1_7841
     MOV       A,#&B10010111       ;CH0,SGL,POWER
     SJMP      _START_7841
     ;
_CH1_7841:
     CJNE      A,#1,_CH2_7841
     MOV       A,#&B11010111       ;CH1,SGL,POWER
     SJMP      _START_7841
_CH2_7841:
     CJNE      A,#2,_CH3_7841
     MOV       A,#&B10100111       ;CH2,SGL,POWER
     SJMP      _START_7841

_CH3_7841:
     MOV       A,#&B11100111       ;CH3,SGL,POWER

_START_7841:
     MOV       R2,#8               ;LOOP CONTROL BYTE
     CLR       {ADS7841_CS}
_WRITE_ADS7841:
     RLC       A
     MOV       {ADS7841_DIN},C
     SETB      {ADS7841_DCLK}
     CLR       {ADS7841_DCLK}
     DJNZ      R2,_WRITE_ADS7841

     SETB      {ADS7841_DCLK}      ;Skip Busy
     CLR       {ADS7841_DCLK}

     MOV       R2,#8               ;Loop Read High Byte Data
_READ_HIGH:
     SETB      {ADS7841_DCLK}
     MOV       C,{ADS7841_DOUT}
     CLR       {ADS7841_DCLK}
     RLC       A
     DJNZ      R2,_READ_HIGH
     MOV       DPH,A               ;Save Data High Byte

     MOV       R2,#8               ; Loop Read Low Byte Data
```

```
_READ_LOW:
     SETB       {ADS7841_DCLK}
     MOV        C,{ADS7841_DOUT}
     CLR        {ADS7841_DCLK}
     RLC        A
     DJNZ       R2,_READ_LOW
     MOV        DPL,A               ;Save Data Low Byte
     SETB       {ADS7841_CS}        ;DISABLE ADC

     MOV        R2,#4               ; Shift Result
_SHIFT_RESULT:
     MOV        A,DPH
     RRC        A
     MOV        DPH,A
     MOV        A,DPL
     RRC        A
     MOV        DPL,A
     DJNZ       R2,_SHIFT_RESULT
     ;
     MOV        A,DPL               ; Get Result Low Byte
     MOV        @R0,A               ; Save Low Byte Result
     INC        R0                  ; Next Point to High Byte
     MOV        A,DPH               ; Get Result High Byte
     ANL        A,#&H0F
     MOV        @R0,A               ; Save High Byte Result
     RET
[END]
```

แสดง Source Code ใน Library คำสั่งของ ASD7841 (ET-7841.LIB)

ซึ่งผู้เขียนได้สร้าง Library ไฟล์นี้ไว้ให้เรียบร้อยแล้ว โดยกำหนดชื่อไฟล์เป็น “ET-7841.LIB” แต่ถ้าต้องการทดลองสร้าง Library เองก็ให้พิมพ์คำสั่งภาษาแอสแซมบลี้ดังตัวอย่างข้างต้น จากนั้นก็ทำการสั่งบันทึกเป็นไฟล์ชื่อ ET-7841.LIB แล้วทำการสั่งคัดลอกไฟล์ดังกล่าวไปไว้ใน Directory ชื่อ LIB ของโปรแกรม BASCOM-8051 ก็พร้อมใช้งานแล้ว

โดยในการเรียกใช้งานคำสั่งใน Library จะใช้วิธีการสั่ง Include ไฟล์ Library เข้ามาใช้งานพร้อมกับประกาศให้ BASCOM-8051 ทราบว่าเราต้องการเรียกใช้คำสั่งใดใน Library บ้าง ดังตัวอย่าง

```
$lib "et-7841.lib"          'เรียกใช้งาน Library ไฟล์ ชื่อ “ET-7541.LIB”
$external _ads7841          'เรียกใช้คำสั่งโปรแกรมย่อยใน Library
```

ซึ่งหลังการประกาศคำสั่งดังกล่าวข้างต้นไว้ในส่วนหัวโปรแกรมของภาษาเบสิกแล้วก็สามารถเรียกใช้งานคำสั่งใน Library ได้แล้ว โดยวิธีการใช้งานให้ศึกษาจากตัวอย่างโปรแกรมต่อไปนี้

```
$regfile = "89c51rd.dat"        'กำหนดใช้งานกับ P89C51RD2(Philips)
$ramstart = 0
$ramsize = 256
$crystal = 36864000             'กำหนด XTAL = 18.432MHz แบบ X2 Mode

Config Lcdpin = Pin , Db4 = P2.0 , Db5 = P2.1 , Db6 = P2.2 ,
Db7 = P2.3 , E = P2.4 , Rs = P2.5  'กำหนดการเชื่อมต่อกับ LCD
Config Lcd = 16 * 2             'กำหนดใช้งานกับ LCD ขนาด 16 ตัวอักษร 2 บรรทัด
Cursor Off                      'ไม่ต้องแสดง Cursor LCD

$lib "et-7841.lib"              'เรียกใช้งาน Library ไฟล์ ชื่อ "ET-7541.LIB"
$external _ads7841              'เรียกใช้คำสั่งโปรแกรมย่อยใน Library
Ads7841_dclk Alias P3.3         'กำหนดการเชื่อมต่อกับ Clock Input
Ads7841_cs Alias P3.4           'กำหนดการเชื่อมต่อกับ Chips Select
Ads7841_din Alias P3.5          'กำหนดการเชื่อมต่อกับ Data Input
Ads7841_dout Alias P3.7         'กำหนดการเชื่อมต่อกับ Data Output

Declare Sub Ads7841(ad_channel As Byte,Ad_result As Word)
Dim Ad_channel As Byte          'กำหนดตัวแปรสำหรับผ่านค่า Channel
Dim Ad_result As Word           'กำหนดตัวแปรสำหรับผ่านค่าผลลัพธ์
Dim Adc_calculate As Single
Dim Adc_display As String * 8

Cls                             'สั่ง Clear หน้าจอ Display
Lcd "DEMO ADS7841 A/D"          'แสดงข้อความที่บรรทัดแรก
Locate 2 , 1                    'กำหนด Cursor ไว้ที่ตัวอักษรแรกของบรรทัดที่2
Lcd "ADC 0 =     Volt"          'แสดงข้อความที่บรรทัดที่2
Do                              'เริ่มต้นวงรอบการอ่านค่า A/D
  Ads7841 , 0 , Ad_result       'สั่งอ่านค่า  A/D ของ CH0 ไว้ในตัวแปร AD_RESULT
  Adc_calculate = Ad_result     'เปลี่ยนค่า Word เป็นเลขทศนิยม
  Adc_calculate = Adc_calculate * 0.0012210   'คูณผลลัพธ์กับ Step
  Adc_display = Fusing(adc_calculate , #.#)   'แสดงผลทศนิยม 1 หลัก
  Locate 2 , 9                                'กำหนด Cursor แสดงผล
  Lcd Adc_display                             'แสดงค่าผลลัพธ์เป็น Volt
Loop                            'กลับไปอ่านค่า A/D ใหม่ไม่รู้จบ
```

```
Sub Ads7841(ad_channel As Byte , Ad_result As Word)'โปรแกรมอ่าน A/D
    $asm
      mov r0,#{ad_result}   ;ให้ R0 ชี้ตำแหน่งแอดเดรสไบท์ต่ำของตัวแปรที่เก็บผลลัพธ์
      mov r1,{ad_channel}   ;ให้ R1 = หมายเลขช่อง A/D ที่ต้องการอ่านค่า (0-3)
      Lcall _ads7841        ;เรียกใช้โปรแกรมย่อยใน Library เพื่ออ่านค่าผลลัพธ์
    $end Asm
End Sub                     'จบโปรแกรมย่อย
```

แสดง ตัวอย่างโปรแกรมอ่านค่า A/D โดยใช้ Library คำสั่งที่สร้างขึ้น

สำหรับการทำงานของโปรแกรมตัวอย่างจะเป็นการสั่งอ่านค่า A/D ช่อง0 แล้วนำค่ามาแสดงผลที่หน้าจอ LCD ในรูปแบบของ Volt Meter โดยแสดงค่าผลลัพธ์ในหน่วยของ Volt โดยการทำงานส่วนที่เกี่ยวข้องกับ Library (ET-7841.LIB) นั้นจะเริ่มจากการสั่ง Include ไฟล์ “ET-7841.LIB” ขึ้นมาใช้งานในโปรแกรมซึ่งจะต้องนำไฟล์ดังกล่าวไปใส่ไว้ใน Directory ชื่อ LIB เสียก่อน เนื่องจากเมื่อพบคำสั่ง $LIB โปรแกรม BASCOM-8051 จะไปค้นหาไฟล์จาก Directory นี้เสมอ จากนั้นจะเป็นการประกาศเรียกใช้คำสั่ง _ADS7841 จาก Library ดังกล่าว

โดยจะเห็นได้ว่าในส่วนของโปรแกรมหลักจะมีการประกาศ Pin Port สำหรับใช้ในการเชื่อมต่อของ CPU กับ ADS7841 ไว้ในส่วนหัวโปรแกรมด้วย เนื่องจาก Library จะถูกออกแบบให้มีความอ่อนตัวในการนำไปใช้งาน สามารถเปลี่ยนแปลง Pin Port ในการเชื่อมต่อได้เอง

จากนั้นก็จะมีการประกาศสร้างโปรแกรมย่อยสำหรับเป็นใช้ทำหน้าที่เป็นตัวกลางในการผ่านค่าพารามิเตอร์ระหว่างโปรแกรมหลักและคำสั่งใน Library โดยจะมีการสร้างตัวแปรขึ้นมารองรับการทำงานของคำสั่งใน Library จำนวน 2 ตัวแปร คือ

- AD_CHANNEL เป็นตัวแปรแบบ Byte ใช้สำหรับผ่านค่า Channel ของ A/D ที่ต้องการสั่งอ่านค่าผลลัพธ์ ซึ่งสามารถผ่านค่าเป็นตัวเลขระหว่าง 0-3
- AD_RESULT เป็นตัวแปรแบบ Word ใช้สำหรับผ่านค่าผลลัพธ์ที่ได้จากการอ่านค่า A/D กลับมายังโปรแกรม

โดยในส่วนของโปรแกรมย่อยที่สร้างขึ้นมาใช้งานนั้น ในการเรียกใช้แต่ละครั้งจะต้องมีการผ่านค่าตัวแปรจำนวน 2 ค่าให้กับโปรแกรมย่อยด้วย โดยค่าแรก (AD_CHANNEL) จะเป็นค่าตัวเลข 0-3 เพื่อกำหนดช่องของ A/D ที่ต้องการอ่านค่า ส่วนค่าที่2 (AD-RESULT) จะเป็นตัวแปรสำหรับใช้เก็บผลลัพธ์ที่ได้จาก A/D ของช่องที่กำหนด

โดยในการผ่านค่าให้กับคำสั่งใน Library นั้นจะใช้คำสั่งของภาษาแอสแซมบลี้ทั้งหมด โดยจะกำหนดให้รีจิสเตอร์ R0 ชี้ไปที่ตำแหน่งแอดเดรสของหน่วยความจำไบท์ต่ำที่ถูกใช้เป็นตัวแปรสำหรับเก็บค่าผลลัพธ์ ส่วนรีจิสเตอร์ R1 จะใช้ผ่านค่าที่ผ่านมาให้กับตัวแปร AD_CHANNEL ไว้ จากนั้นจึงทำการเรียกใช้โปรแกรมย่อย ADS7841 ใน Library โดยการทำงานของโปรแกรมย่อยหรือคำสั่งใน Library นี้ จะทำให้ได้ผลลัพธ์เก็บไว้ในตัวแปร ADRESULT ทันที

โดยในส่วนของโปรแกรมหลักนั้นจะใช้คำสั่ง DO..LOOP สำหรับทำหน้าที่วนรอบอ่านค่า A/D โดยจะเรียกใช้โปรแกรมย่อยที่สร้างขึ้น ดังนี้

```
Ads7841 , 0 , Ad_result          'สั่งอ่านค่า A/D ของ CH0 ไว้ในตัวแปร AD_RESULT
```

ซึ่งบรรทัดคำสั่งดังกล่าวจะเป็นการเรียกใช้โปรแกรมย่อย ADS7841 โดยมีการผ่านค่าตัวเลข ศูนย์ไปให้กับโปรแกรมย่อย ซึ่งหมายถึง A/D ช่อง 0 ซึ่งสามารถเปลี่ยนเป็นค่าตัวเลข 1 หรือ 2 หรือ 3 แทนได้ ส่วนตัวแปรที่2 จะเป็นการผ่านค่าตัวแปรชื่อ AD_RESULT สำหรับเก็บค่าผลลัพธ์ของ A/D ซึ่งเมื่อโปรแกรมทำงานตามบรรทัดคำสั่งนี้แล้วจะได้ค่าผลลัพธ์เก็บไว้ในตัวแปร AD_RESULT ทันที โดยค่าในตัวแปรจะมีค่าระหว่าง 0-4095 ซึ่งเป็นค่า Step ของ A/D ที่อ่านได้

โดยในโปรแกรมตัวอย่างจะทำการนำค่านั้นไปแปลงเป็นค่าแรงดันเสียก่อน โดยนำไปคูณกับค่าคงที่ 0.0012210 เพื่อเปลี่ยนเป็นค่าแรงดันตามต้องการ จากนั้นจะใช้คำสั่ง Fusing เพื่อจัดรูปแบบการแสดงผลค่าตัวเลขทศนิยมให้แสดงผลทศนิยมเพียงหลักเดียว

ซึ่งในการนำไปประยุกต์ใช้งานนั้น อาจไม่จำเป็นต้องทำการคำนวณค่าผลลัพธ์ของ A/D ให้เป็นค่าแรงดันก็ได้ แต่อาจใช้วิธีการเปรียบเทียบค่า Step ที่อ่านได้จาก A/D ว่าเท่ากับหรือ มากกว่า หรือน้อยกว่าค่าอ้างอิงที่มีอยู่เพื่อใช้ประกอบการตัดสินใจแทนก็ได้

ตัวอย่างโปรแกรมการอ่านค่า A/D โดยไม่ต้องแยกไฟล์ Library

```
$regfile = "89c51rd.dat"          'กำหนดใช้งานกับ P89C51RD2(Philips)
$ramstart = 0
$ramsize = 256
$crystal = 36864000               'กำหนด XTAL = 18.432MHz แบบ X2 Mode

Config Lcdpin = Pin , Db4 = P2.0 , Db5 = P2.1 , Db6 = P2.2 ,
Db7 = P2.3 , E = P2.4 , Rs = P2.5  'กำหนดการเชื่อมต่อกับ LCD
Config Lcd = 16 * 2               'กำหนดใช้งานกับ LCD ขนาด 16 ตัวอักษร 2 บรรทัด
Cursor Off                        'ไม่ต้องแสดง Cursor LCD

Ads7841_dclk Alias P3.3           'กำหนดการเชื่อมต่อกับ Clock Input
Ads7841_cs Alias P3.4             'กำหนดการเชื่อมต่อกับ Chips Select
Ads7841_din Alias P3.5            'กำหนดการเชื่อมต่อกับ Data Input
Ads7841_dout Alias P3.7           'กำหนดการเชื่อมต่อกับ Data Output

Declare Sub Ads7841(ad_channel As Byte,Ad_result As Word)
Dim Ad_channel As Byte            'กำหนดตัวแปรสำหรับผ่านค่า Channel
Dim Ad_result As Word             'กำหนดตัวแปรสำหรับผ่านค่าผลลัพธ์
Dim Adc_calculate As Single
Dim Adc_display As String * 8

Cls                               'สั่ง Clear หน้าจอ Display
Lcd "DEMO ADS7841 A/D"            'แสดงข้อความที่บรรทัดแรก
Locate 2 , 1                      'กำหนด Cursor ไว้ที่ตัวอักษรแรกของบรรทัดที่2
Lcd "ADC 0 =     Volt"            'แสดงข้อความที่บรรทัดที่2
Do                                'เริ่มต้นวงรอบการอ่านค่า A/D
  Ads7841 , 0 , Ad_result         'สั่งอ่านค่า  A/D ของ CH0 ไว้ในตัวแปร AD_RESULT
  Adc_calculate = Ad_result       'เปลี่ยนค่า Word เป็นเลขทศนิยม
  Adc_calculate = Adc_calculate * 0.0012210   'คูณผลลัพธ์กับ Step
  Adc_display = Fusing(adc_calculate , #.#)   'แสดงผลทศนิยม 1 หลัก
  Locate 2 , 9                                'กำหนด Cursor แสดงผล
  Lcd Adc_display                             'แสดงค่าผลลัพธ์เป็น Volt
Loop                              'กลับไปอ่านค่า A/D ใหม่ไม่รู้จบ
```

```
Sub Ads7841(ad_channel As Byte , Ad_result As Word)'โปรแกรมอ่าน A/D
  $ASM
     MOV R0,#{AD_RESULT}   ;ให้ R0 ชี้ตำแหน่งแอดเดรสไบท์ต่ำของตัวแปรที่เก็บผลลัพธ์
     MOV R1,{AD_CHANNEL}   ;ให้ R1 = หมายเลขช่อง A/D ที่ต้องการอ่านค่า (0-3)
      SETB      {ADS7841_CS}         ;Initial CS
     CLR        {ADS7841_DCLK}       ;Initial DCLK

     MOV        A,R1                 ;Get Channel
     CJNE       A,#0,_CH1_7841
     MOV        A,#&B10010111        ;CH0,SGL,POWER
     SJMP       _START_7841
     ;
  _CH1_7841:
     CJNE       A,#1,_CH2_7841
     MOV        A,#&B11010111        ;CH1,SGL,POWER
     SJMP       _START_7841
  _CH2_7841:
     CJNE       A,#2,_CH3_7841
     MOV        A,#&B10100111        ;CH2,SGL,POWER
     SJMP       _START_7841

  _CH3_7841:
     MOV        A,#&B11100111        ;CH3,SGL,POWER

  _START_7841:
     MOV        R2,#8                ;LOOP CONTROL BYTE
     CLR        {ADS7841_CS}
  _WRITE_ADS7841:
     RLC        A
     MOV        {ADS7841_DIN},C
     SETB       {ADS7841_DCLK}
     CLR        {ADS7841_DCLK}
     DJNZ       R2,_WRITE_ADS7841

     SETB       {ADS7841_DCLK}       ;Skip Busy
     CLR        {ADS7841_DCLK}

     MOV        R2,#8                ;Loop Read High Byte Data
  _READ_HIGH:
     SETB       {ADS7841_DCLK}
     MOV        C,{ADS7841_DOUT}
     CLR        {ADS7841_DCLK}
     RLC        A
     DJNZ       R2,_READ_HIGH
     MOV        DPH,A                ;Save Data High Byte

     MOV        R2,#8                ; Loop Read Low Byte Data
  _READ_LOW:
```

```
    SETB      {ADS7841_DCLK}
    MOV       C,{ADS7841_DOUT}
    CLR       {ADS7841_DCLK}
    RLC       A
    DJNZ      R2,_READ_LOW
    MOV       DPL,A               ;Save Data Low Byte
    SETB      {ADS7841_CS}        ;DISABLE ADC

    MOV       R2,#4               ; Shift Result
  _SHIFT_RESULT:
    MOV       A,DPH
    RRC       A
    MOV       DPH,A
    MOV       A,DPL
    RRC       A
    MOV       DPL,A
    DJNZ      R2,_SHIFT_RESULT
    ;
    MOV       A,DPL               ; Get Result Low Byte
    MOV       @R0,A               ; Save Low Byte Result
    INC       R0                  ; Next Point to High Byte
    MOV       A,DPH               ; Get Result High Byte
    ANL       A,#&H0F
    MOV       @R0,A               ; Save High Byte Result
    RET


  $END ASM
End Sub                           'จบโปรแกรมย่อย
```