รายการอ้างอิง

1. นายพรยุทธ ชินมหาวงศ์. การเปลี่ยนไมโครคอมพิวเตอร์ 8 บิต ให้เป็นอุปกรณ์วิเคราะห์ความสูงของพัลส์. วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชานิวเคลียร์เทคโนโลยี บัณฑิตวิทยาลัยจุฬาลงกรณ์มหาวิทยาลัย,พ.ศ. 2530.
2. นายธเนส ศิริไตรวัฒนาพร. การพัฒนาระบบรับส่งข้อมูลด้วยแสงอินฟราเรดสำหรับอุปกรณ์นิวคลีออนิกส์. วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชานิวเคลียร์เทคโนโลยี บัณฑิตวิทยาลัยจุฬาลงกรณ์มหาวิทยาลัย,พ.ศ.2541.
3. นายหัสฤกษ์ เนียมอินทร์. แผ่นวงจรเชื่อมโยงสัญญาณแบบเอนกประสงค์สำหรับระบบวัดนิวเคลียร์. วิทยานิพนธ์ วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชานิวเคลียร์เทคโนโลยี บัณฑิตวิทยาลัยจุฬาลงกรณ์มหาวิทยาลัย,พ.ศ.2535.
4. รศ.นเรศร์ จันทน์ขาว. การวิเคราะห์ธาตุด้วยวิธีเรืองรังสีเอกซ์เชิงปฏิบัติ, พ.ศ.2522. 129 หน้า
5. รศ. วิรุฬห์ มังคละวิรัช และ ผศ. สุวิทย์ ปุณณชัยยะ. อุปกรณ์วิเคราะห์แบบหลายช่อง. รายงานผลการประดิษฐ์เงินอุดหนุนโครงการสิ่งประดิษฐ์, พ.ศ.2536
6. นายกอบเกียรติ กาญจนพงศ์กุล. การอินเตอร์เฟสพอร์ต USB ที่ง่ายขึ้นด้วย FT8U232AM และ FT8U245AM. วารสารเซมิคอนดักเตอร์อิเล็กทรอนิกส์, ฉบับที่236 ประจำเดือนมิถุนายน พ.ศ.2545. หน้าที่ 212 – 221.

7. Bertin, Eugene P. Principle and Practice of X-ray Spectrometric Analysis. 3rd ed. New York: Plenum Press,1975.
8. Grieken, R. Van, and Markowicz, Amdrzej. Handbook of X-ray spectrometry: methods and Techniques. New York: Marcel Dekker,1993.
9. Jenkins, Ron. X-ray Fluorescence Spectrometry. New York: Wiley,1988.
10. Jenkins, Ron. Gould, R.W. and Gedcke, Dale. Quantitative X-ray Spectrometry. 2rd ed. New York: Marcel Dekker, 1995.
11. Knoll, Glenn F. Radiation Detection and Measurement. New York: McGraw-Hill Book Company, 1999.
12. Tsoulfanidis N. Measurement and Detection of Radiation. New York: McGraw-Hill Book Company, 1983.
13. Gillmore G. and Hemingway J. Practical Gamma-Ray Spectrometry. New York: John Wiley & Son, 1995.

14. A Guide to Energy Dispersive X-ray Analysis. Link Analytical Limited.

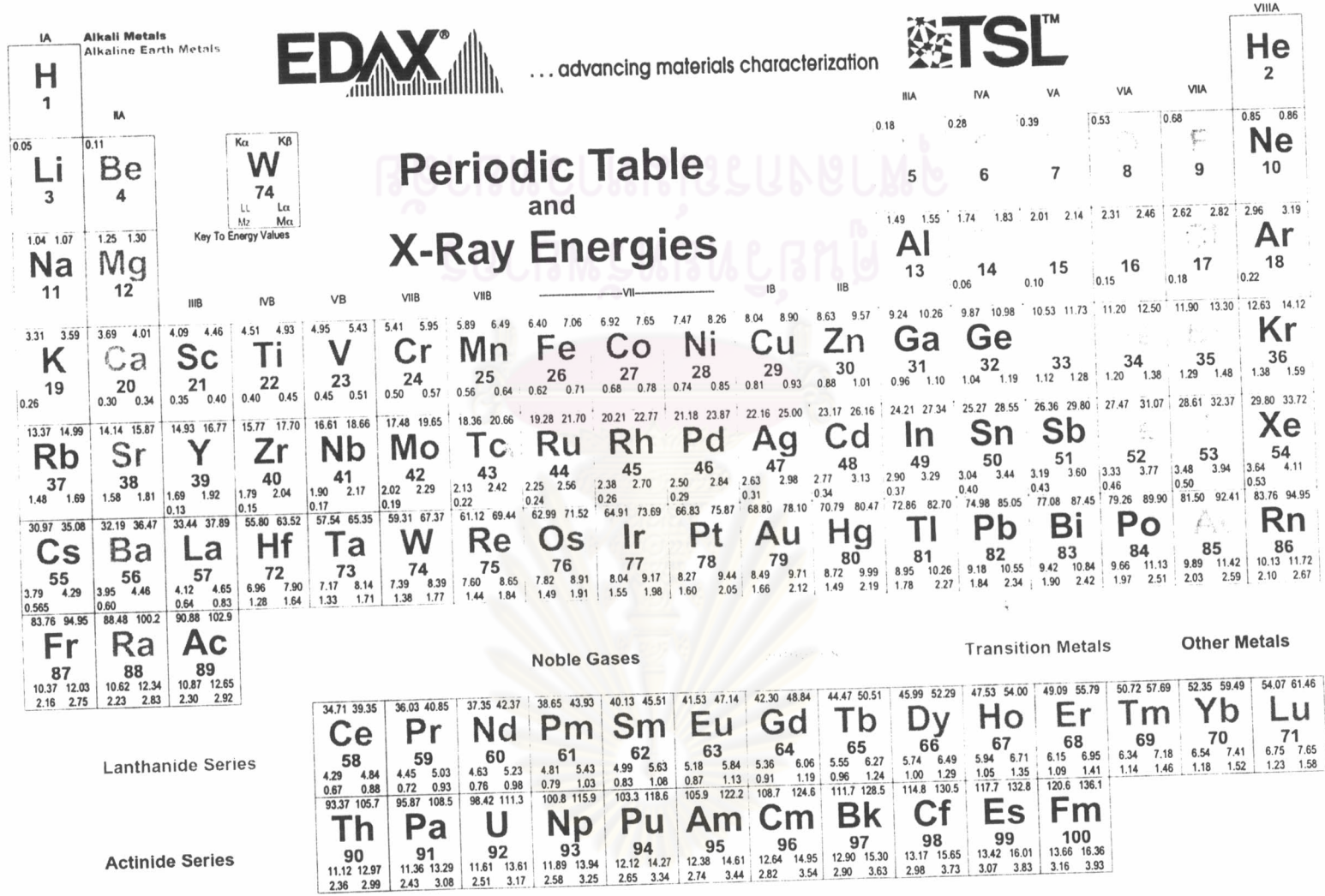
15. Trainning Notes Detectors/Processors. Link Analytical Limited.

ภาคผนวก

ภาคผนวก ก.

คุณสมบัติของต้นกำเนิดรังสีชนิดไอโซโทปบางชนิด

RADIO ISOTOPE	HALF-LIFE (YEAR)	MODE OF DECAY	USEFUL RADIATION EMITTED (keV)	%PHOTON PER DISINTEGRATION	TYPICAL ACTIVITY	
					FOR SCINTILLA-TION & PROPORTIONAL COUNTER	FOR HIGH RESOLUTION Si AND Ge DETECTORS
^{55}Fe	2.7	EC	5.9(Mn K X-rays)	28.5	2 mCi	5 mCi
^{3}H / Zr	12.3	β	2-12 (Bremss.)	$4x10^{-3}$	2 Ci	2 Ci
			2(Zr L X-rays)	$10^{-3} - 10^{-2}$		
^{3}H / Ti	12.3	β	2-12 (Bremss.)	10^{-2}	2 Ci	2 Ci
			4.5(Ti K X-rays)	10^{-2}		
^{238}Pu	86.4	$\alpha.\gamma$	12-22(U L X-rays)	13	10 mCi	30 mCi
^{109}Cd	1.3	EC	22.1(Ag K X-rays)	107	1 mCi	3 mCi
			88 (γ -rays)	4		
^{147}Pm/Al	2.6	β	10-100(Bremss.)	0.4	0.5 Ci	10 Ci
^{241}Am	458	$\alpha.\gamma$	12-22(Np L X-rays)	37	10 mCi	100 mCi
			26 (γ-rays)	2		
			59.6 (γ-rays)	3		
^{153}Gd	0.65	EC	41 (Eu K X-rays)	110	1 mCi	3 mCi
			70 (γ-rays)	2.6		
			97 (γ-rays)	30		
			103 (γ-rays)	20		
^{57}Co	0.74	EC	6.4(Fe K X-rays)	48	1 mCi	3 mCi
			14 (γ-rays)	8.2		
			122 (γ-rays)	85		
			136 (γ-rays)	11		
			700 (γ-rays)	0.2		

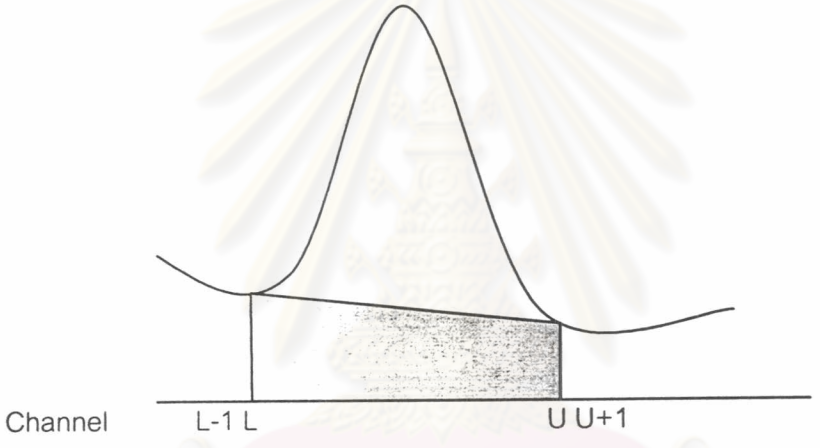


EDAX Inc. - 91 McKee Drive - Mahwah, NJ, 07430 - 201-529-4880 - www.edax.com - e-mail: edax.info@ametek.com

ภาคผนวก ค

Covell Method

Covell Method เป็นกระบวนการในการประมาณค่าระดับแบ็กกราวนด์ภายใต้พีคสเปกตรัมเพื่อที่จะนำเอาค่าดังกล่าวไปหักลบกับค่าพื้นที่ทั้งหมดจะได้พื้นที่ของสเปกตรัมพลังงานเฉพาะที่เราสนใจ ค่าพื้นที่ของสเปกตรัมสุทธิจะนำไปคำนวณหาค่าทาง Quantitative เพื่อหาปริมาณของธาตุนั้นๆในตัวอย่าง หลักการคำนวณของ Covell Method จะเริ่มจากการหาตำแหน่งแชนแนลที่มีค่าพีคสูงสุด ณ จุดนี้จะเป็นจุดศูนย์กลางของสเปกตรัมจากนั้นจะทำการหาขอบเขตของสเปกตรัมโดยกำหนดขอบเขตทางด้านซ้ายและขวาของพีค ดังรูปที่1



รูปที่ 1 แสดงการคำนวณพื้นที่พีคโดยใช้การคำนวณแบบ Covell Method

เมื่อหาขอบเขตได้แล้วเราจะนับค่า Count ในแต่ละแชนแนลรวมกันตั้งแต่ขอบเขตซ้ายไปจนถึงขอบเขตขวาเราจะได้ค่าพื้นที่ทั้งหมดหรือ Integral Area ดังสมการที่ 1.1

$$G = \sum_{I=L}^{U} C_i \qquad \text{.....1.1}$$

จากสมการดังกล่าวเป็นค่าพื้นที่ใต้พีคที่รวมเอาค่าพื้นที่แบ็กกราวนด์เอาไว้ด้วย ดังนั้นถ้าต้องการหาเฉพาะเพียงค่าพื้นที่ใต้พีคจริงๆจะต้องหักลบค่าแบ็กกราวนด์ออกไปดังสมการที่ 1.2

$$A = G - B \qquad \text{.....1.2}$$

การคำนวณพื้นที่แบ็กกราวนด์ภายใต้พีคที่เราสนใจจะหาได้จากสมการที่ 1.3

$$B = n(C_{L-1} + C_{U+1})/2 \qquad \text{.....1.3}$$

เมื่อ n เท่ากับจำนวนแชนแนลระหว่างพื้นที่ใต้พีค

C_{L-1} เท่ากับค่า Count ในแชนแนลก่อนหน้าแชนแนลขอบเขตซ้าย1 ช่อง

C_{U+1} เท่ากับค่า Count ในแชนแนลหลังแชนแนลขอบเขตขวา 1 ช่อง

ค่าที่ได้จากสมการ 1.3 เป็นค่าพื้นที่แบ็กกราวนด์โดยได้จากค่าเฉลี่ยแบ็กกราวนด์ต่อหนึ่งแชนแนลคูณด้วยจำนวนแชนแนลทั้งหมดภายใต้พีคที่สนใจ จากสมการ 1.1, 1.2 และ 1.3 นำมารวมกันจะได้สมการที่ 1.4

$$A = G - B = \sum_{i=L}^{U} C_i - n(C_{L-1} + C_{U+1})/2 \qquad \text{.....1.4}$$

จากสมการ 1.4 เราจะสามารถหาค่าพื้นที่ใต้พีคได้จากการหักลบค่าแบ็กกราวด์ออกไป แต่ค่าที่ได้จากสมการนี้อาจได้ผลลัพธ์ไม่ตรงนักเนื่องจากการคำนวณค่าแบ็กกราวนด์ดังกล่าวอาจผิดพลาดได้เนื่องจากเป็นการประมาณค่าโดยใช้เพียงสองจุด ดังนั้นเพื่อเพิ่มความแม่นยำในการคำนวณมากขึ้นเราจะเพิ่มเติมสมการเป็นสมการที่ 1.5 โดย ค่า m จะเป็นจำนวนจุดที่ประมาณค่า

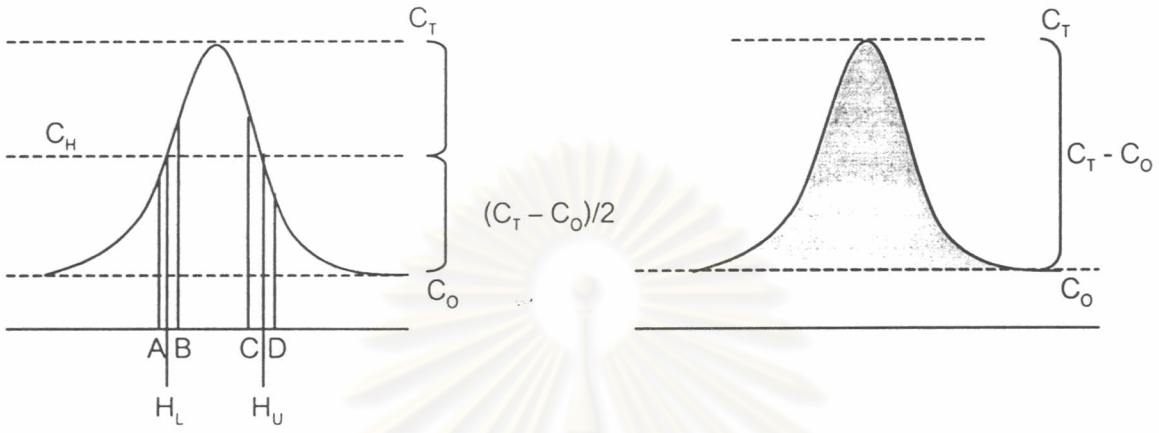
$$A = G - B = \sum_{i=L}^{U} C_i - n\left[\sum_{i=L-m}^{L-1} + \sum_{i=U+1}^{U+m}\right]/2m \qquad \text{.....1.5}$$

ค่าแบ็กกราวนด์เฉลี่ยต่อหนึ่งแชนแนลเท่ากับ

$$C_B = \left[\sum_{i=L-m}^{L-1} + \sum_{i=U+1}^{U+m}\right]/2m \qquad \text{.....1.6}$$

อัลกอริทึมในการประมาณค่าความกว้างพีค (FWHM)

กระบวนการในการประมาณค่าความกว้างของพีคจะมีอยู่สองแบบ คือแบบ Interpolation กับแบบ Area/height ratio ดังแสดงในรูปที่ 1



รูปที่ 1 กระบวนการการประมาณค่า FWHM ซ้าย: Interpolation ขวา: Area/height ratio

กระบวนการประมาณค่าแบบ Interpolation จะมีหลักการดังนี้

(ก) ประมาณค่าความสูงของพีค C_T ณ จุดยอดของพีค

(ข) หักลบค่า C_t ด้วยค่าจำนวนนับของแบ็กกราวนด์ C_O ซึ่งเป็นค่าเฉลี่ยแบ็กกราวนด์ที่ได้จากการคำนวณจากสมการ Covell Method

(ค) นำค่าที่หักลบแบ็กกราวนด์แล้วมาหารสองแล้วบวกเพิ่มค่า C_O อีกครั้งจะได้ค่าจำนวนนับโดยประมาณ ณ ตำแหน่งครึ่งหนึ่งของความสูงพีค (C_H)

(ง) หาแชนแนลทางด้านซ้ายของยอดพีคสองจุดที่มีค่าจำนวนนับใกล้เคียงกับค่า C_H โดยแชนแนลที่มีจำนวนนับใกล้เคียงกับค่า C_H แต่มีค่าน้อยกว่าให้กำหนดเป็นแชนแนล A ส่วนแชนแนลที่มีจำนวนนับใกล้เคียงกับค่า C_H แต่มีค่ามากกว่าให้กำหนดเป็นแชนแนล B ค่าจำนวนนับ ณ ตำแหน่ง A และ B จะกำหนดให้เท่ากับ C_A และ C_B ตามลำดับ ค่าแชนแนลทางด้านซ้ายที่มีความสูงเท่ากับครึ่งหนึ่งของพีคจะมีค่าเท่ากับ

$$H_L = A + (C_H - C_A)/(C_B - C_A)$$

(จ) ตำแหน่งของแชนแนลทางด้านขวาที่มีความสูงเท่ากับครึ่งหนึ่งของพีคจะมีวิธีการหาเหมือนกับข้อ (ง) มีค่าเท่ากับ

$$H_U = C + (C_C - C_H)/(C_C - C_D)$$

(ฉ) ค่าความต่างระหว่าง $H_U - H_L$ จะเป็นค่า FWHM ในหน่วยแชนแนล

กระบวนการประมาณค่าแบบ area/height ratio จะสมมุติว่ารูปร่างของพีคเป็นแบบ Guassian ดังนั้นค่าความกว้างของพีคจะหาได้จากสมการ

$$FWHM = 0.939 \times A / (C_T - C_O)$$

เมื่อ A คือพื้นที่ของพีค

C_T คือความสูงของพีค

C_O คือความสูงของแบ็กกราวนด์

สมการที่ได้นี้มาจากการคำนวณและวิเคราะห์ทางสถิติของการแจกแจงข้อมูลแบบ Gaussian แต่ค่าที่ได้จากสมการนี้จะเป็นค่าประมาณที่ให้ผลไม่แม่นยำเท่ากับแบบ Interpolate แต่ให้ความรวดเร็วในการคำนวณมากกว่า

ภาคผนวก ง

โปรแกรมอิมูเลเตอร์

```
Dim COMP, BPS, DB, PAR, SB As String
Dim CountSB As Integer
Dim A, B As Long
Dim CommData As Long
Dim DataChannel(2048) As Long
Dim i, J, K, L, M, T As Integer
Dim MouseWK As Integer
Dim Xpos, Ypos As Integer
Dim Xpointer, Ypointer, Zpointer As Integer
Dim X1, Y1 As Integer
Dim H1, W1 As Integer
Dim ROLindex(100), RORindex(100) As Integer
Dim ROIindex As Integer
Dim ROI, ROII As Boolean
Dim ROL, ROR As Integer
Dim XStart, XXStart As Integer
Dim pausetime As Integer

Private Sub Command1_Click()
Dim eq, sq As Integer
Dim tq As Single
   eq = 8
   sq = 3
   tq = eq / sq
   Debug.Print tq
End Sub

Private Sub Energy_Click()
   Form2.Visible = True
End Sub

Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
   Select Case KeyCode
      Case vbKeyF1: MsgBox "F1 is your friend."
      Case vbKeyInsert
            For i = 1 To 100
               If ROLindex(i) = 0 Then
```

```
                ROIindex = i
                Exit For
            End If
        Next i
        ROLindex(ROIindex) = ROL
        RORindex(ROIindex) = ROR
        Call ROIA(ROLindex(ROIindex), RORindex(ROIindex))
Case vbKeyDelete
        If ROIindex <> 0 Then
            For J = ROLindex(ROIindex) To RORindex(ROIindex)
                X_PIC.PSet (J - Xpointer, X1 - (DataChannel(J) * H1)), vbGreen
                X_PIC.Line (J - Xpointer, X1)-(J - Xpointer, X1 - (DataChannel(J) * H1)), vbYellow
            Next J
            For i = ROIindex To 99
                ROLindex(i) = ROLindex(i + 1)
                RORindex(i) = RORindex(i + 1)
                ROLindex(i + 1) = 0
                RORindex(i + 1) = 0
            Next i
            ROIindex = 1
            If ROLindex(ROIindex) <> 0 Then
            Call ROIA(ROLindex(ROIindex), RORindex(ROIindex))
            End If
        End If
Case vbKeyAdd
        If (ROIindex < 100) And (ROLindex(ROIindex + 1) <> 0) Then
            ROIindex = ROIindex + 1
            Call ROIA(ROLindex(ROIindex), RORindex(ROIindex))
            Debug.Print "Position "; ROIindex
        End If
Case vbKeySubtract
        If (ROIindex > 0) And (ROLindex(ROIindex - 1) <> 0) Then
            ROIindex = ROIindex - 1
            Call ROIA(ROLindex(ROIindex), RORindex(ROIindex))
            Debug.Print "Position"; ROIindex
        End If
Case vbKeyLeft
    Zpointer = Zpointer - 1
    If Shift = 1 Then
        ROI = True
    End If
```

```
    Call Cursor(Shift)
    ROI = False
  Case vbKeyRight
    Zpointer = Zpointer + 1
    If Shift = 1 Then
      ROI = True
    End If
    Call Cursor(Shift)
    ROI = False
  Case vbKeyUp
    H1 = H1 * 2
    X_PIC.Cls
    For J = Xpointer To 2048
      X_PIC.PSet (J - Xpointer, X1 - (DataChannel(J) * H1)), vbGreen
      X_PIC.Line (J - Xpointer, X1)-(J - Xpointer, X1 - (DataChannel(J) * H1)), vbYellow
    Next J
    For i = 1 To 100
      If ROLindex(i) <> 0 Then
      Call ROIA(ROLindex(i), RORindex(i))
      Else
        Exit For
      End If
    Next i
    txtStatus.Text = txtStatus.Text & "Stretch " & H1 & vbCrLf
    txtStatus.SelStart = Len(txtStatus.Text)
  Case vbKeyDown
    H1 = H1 / 2
    X_PIC.Cls
    For J = Xpointer To 2048
      X_PIC.PSet (J - Xpointer, X1 - (DataChannel(J) * H1)), vbGreen
      X_PIC.Line (J - Xpointer, X1)-(J - Xpointer, X1 - (DataChannel(J) * H1)), vbYellow
    Next J
    For i = 1 To 100        'Show ROI Area
      If ROLindex(i) <> 0 Then
        Call ROIA(ROLindex(i), RORindex(i))
      Else
        Exit For
      End If
    Next i
    txtStatus.Text = txtStatus.Text & "Stretch " & H1 & vbCrLf
    txtStatus.SelStart = Len(txtStatus.Text)
```

```
    Case vbKeyShift
      ROI = True
      XStart = Zpointer
      Debug.Print "xstart", XStart
  End Select
End Sub
Private Sub Cursor(Shift As Integer)
  X_PIC.DrawMode = 13
  If ROI Then
    X_PIC.Line (Zpointer + 1 - Xpointer, X1)-(Zpointer + 1 - Xpointer, X1 - (DataChannel(Zpointer + 1) * H1)),
&HFF8000
    X_PIC.Line (Ypointer - Xpointer, (X1 - 5) - (DataChannel(Ypointer) * H1))-(Ypointer - Xpointer, (X1 - 30) -
(DataChannel(Ypointer) * H1)), &HFF8000   '&H80000002
    'XStart = Zpointer
    Debug.Print XStart
  Else
    X_PIC.Line (Ypointer - Xpointer, (X1 - 5) - (DataChannel(Ypointer) * H1))-(Ypointer - Xpointer, (X1 - 30) -
(DataChannel(Ypointer) * H1)), RGB(255, 130, 0)
  End If
  If (Zpointer >= 0) Then
      If (Zpointer < 2048) Then
        txtX.Text = " (" & Zpointer & ") "
        txtY.Text = DataChannel(Zpointer)
        txtE.Text = (Zpointer) * Gain + Intercept & "keV"
        If Shift = 1 And ROI Then
          X_PIC.DrawMode = 13
          X_PIC.MousePointer = vbSizeWE
          If XStart <= Zpointer Then
            Debug.Print "print"
            Debug.Print "X", Xpointer, "Zpointer", Zpointer, "X1", X1, "H1", H1
            X_PIC.Line (Zpointer - Xpointer, X1)-(Zpointer - Xpointer, X1 - (DataChannel(Zpointer) * H1)),
vbMagenta
            ROL = XStart: ROR = Zpointer
            Debug.Print "ROL", ROL, "ROR", ROR
          End If
          X_PIC.Line (Zpointer + 1 - Xpointer, X1)-(Zpointer + 1 - Xpointer, X1 - (DataChannel(Zpointer + 1) * H1)),
vbYellow
        Else
          X_PIC.DrawMode = 7
          X_PIC.Line (Zpointer - Xpointer, (X1 - 5) - (DataChannel(Zpointer) * H1))-(Zpointer - Xpointer, (X1 - 30) -
(DataChannel(Zpointer) * H1)), vbWhite
```

```
            X_PIC.Line (Ypointer - Xpointer, (X1 - 5) - (DataChannel(Ypointer) * H1))-(Ypointer - Xpointer, (X1 - 30) -
(DataChannel(Ypointer) * H1)), vbWhite
          End If
        Else
          Zpointer = 2048
        End If
      Else
        Zpointer = 0
      End If
      Ypointer = Zpointer
      X_PIC.DrawMode = 13
End Sub
Private Sub ROIA(ByVal ROLL As Integer, ByVal RORR As Integer)
Dim ACL, ACU, ACT, N As Single
Dim Integral, Area, Background, O As Single
Dim MaxCount, MaxChannel As Single
Dim A, B, C, D As Single
Dim P, Q, R, S As Single
Dim Ca, Cb, Cc, Cd, Ct, Co, Ch, Ch10 As Single
Dim Cp, Cq, Cr, Cs As Single
Dim Chl, Chu As Single
Dim Hl, Hu, Hl10, Hu10, FWHM, FWTM As Single
Dim ind As Integer
   Integral = 0: Area = 0
   MaxCount = 0: MaxChannel = 0
   N = 0: ACL = 0: ACU = 0
   A = 0: B = 0: C = 0: D = 0
   P = 0: Q = 0: R = 0: S = 0
   Ca = 0: Cb = 0: Cc = 0: Cd = 0
   Cp = 0: Cq = 0: Cr = 0: Cs = 0
   Hl = 0: Hu = 0: Hl10 = 0: Hu10 = 0
   For J = ROLL To RORR
     N = N + 1
     If DataChannel(J) > MaxCount Then      'Finding Peak
       MaxCount = DataChannel(J)
       MaxChannel = J
     End If
       Integral = Integral + DataChannel(J) 'Finding Integral Area
       X_PIC.PSet (J - W1, X1 - (DataChannel(J) * H1)), vbGreen
       X_PIC.Line (J - W1, X1)-(J - W1, X1 - (DataChannel(J) * H1)), vbRed
   Next J
```

```
'============================
For K = 1 To 4                    'Calculate AverageCount
   ACL = ACL + DataChannel(ROLL - K)
   ACU = ACU + DataChannel(RORR + K)
Next K
ACT = (ACL + ACU) / 8
Background = N * (ACT)            'Calculate Background
Area = Integral - Background      'Calculate Area
'============================
Ct = MaxCount                     'Count High
Co = ACT
Ch = ((Ct - Co) / 2) + Co
Ch10 = ((Ct - Co) / 10) + Co
'============================
ind = ROLL
While (ind < RORR)
   If (DataChannel(ind - 1) < Ch) And (DataChannel(ind) >= Ch) And (ind < MaxChannel) Then    'Find Left
channel for FWHM
      A = ind - 1: B = ind
   End If
   If (DataChannel(ind - 1) >= Ch) And (DataChannel(ind) < Ch) And (ind > MaxChannel) Then    'Find Right
channel for FWHM
      C = ind - 1: D = ind
   End If
   If (DataChannel(ind - 1) < Ch10) And (DataChannel(ind) >= Ch10) And (ind < MaxChannel) Then   'Find Left
channel for FWTM
      P = ind - 1: Q = ind
   End If
   If (DataChannel(ind - 1) >= Ch10) And (DataChannel(ind) < Ch10) And (ind > MaxChannel) Then   'Find Right
channel for FWTM
      R = ind - 1: S = ind
   End If
   ind = ind + 1
Wend
Ca = DataChannel(A): Cb = DataChannel(B): Cc = DataChannel(C): Cd = DataChannel(D)
Cp = DataChannel(P): Cq = DataChannel(Q): Cr = DataChannel(R): Cs = DataChannel(S)
Hl = A + ((Ch - Ca) / (Cb - Ca))          'Left Channel FWHM
Hu = C + ((Cc - Ch) / (Cc - Cd))          'Right Channel FWHM
Hl10 = P + ((Ch10 - Cp) / (Cq - Cp))      'Left Channel FWTM
Hu10 = R + ((Cr - Ch10) / (Cr - Cs))      'Right Channel FWTM
FWHM = Hu - Hl
```

```
    FWTM = Hu10 - Hl10
    Chl = Ca + ((Hl - A) * (Cb - Ca)) / (B - A)        ' Left Count FWHM
    Chu = Cc + ((Hu - C) * (Cd - Cc)) / (D - C)       'Right Count FWHM
    'Debug.Print "var="; (FWHM / 2.35)                 'Standard Deviation
    'Debug.Print "FWTM="; (4.29 * (FWHM / 2.35)) 'FWTM = 4.29 * Standard; Deviation
    Debug.Print "ACL="; ACL; "ACU="; ACU; "ACT="; ACT; "Background="; Background
    Debug.Print "Ct"; Ct; "Co"; Co; "Ch"; Ch; "Ch10"; Ch10
    Debug.Print "A"; A; "B"; B; "C"; C; "D"; D
    'Debug.Print "P"; P; "Q"; Q; "R"; R; "S"; S
    Debug.Print "Ind"; ind; "Ca"; Ca; "Cb"; Cb; "Cc"; Cc; "Cd"; Cd
    'Debug.Print "Ind"; ind; "Cp"; Cp; "Cq"; Cq; "Cr"; Cr; "Cs"; Cs
    Debug.Print "Hl"; Hl; "Hu"; Hu; "Hl10"; Hl10; "Hu10"; Hu10; "FWHM"; FWHM; "FWTM"; FWTM
    Debug.Print "Chl"; Chl; "Chu"; Chu
    'Debug.Print "Gain"; Gain; "Intercept"; Intercept
    X_PIC.Line (Int(Hl) - W1, X1 - (Chl * H1))-(Int(Hu) - W1, X1 - (Chu * H1)), vbBlack      'FWHM line
    '==================================
    X_PIC.Line (Ypointer - Xpointer, (X1 - 5) - (DataChannel(Ypointer) * H1))-(Ypointer - Xpointer, (X1 - 30) -
(DataChannel(Ypointer) * H1)), &HFF8000  '&H80000002
    Zpointer = MaxChannel
    Ypointer = Zpointer
    X_PIC.Line (Zpointer - Xpointer, (X1 - 5) - (DataChannel(Zpointer) * H1))-(Zpointer - Xpointer, (X1 - 30) -
(DataChannel(Zpointer) * H1)), RGB(255, 130, 0)
    '====================================
    txtROL.Text = " " & ROLL
    txtROR.Text = " " & RORR
    txtInt.Text = " " & Integral
    txtArea.Text = " " & Area
    txtPeak.Text = " (" & MaxChannel & ") " & MaxCount
    Debug.Print "FWHM"; FWHM; "GAIN"; Gain; "RES"; FWHM * Gain
    txtFWHM.Text = " " & FWHM * Gain '+ Intercept
    txtFWTM.Text = " " & FWTM * Gain '+ Intercept
    txtGauss.Text = " " & (FWTM / FWHM) / 1.82
End Sub
Private Sub DrawPic()
    X_PIC.Cls
    For J = 0 To 2047
        X_PIC.PSet (J - W1, X1 - (DataChannel(J) * H1)), vbGreen
        X_PIC.Line (J - W1, X1)-(J - W1, X1 - (DataChannel(J) * H1)), vbYellow
    Next J
End Sub
Private Sub DrawLine()
```

```
    Y_PIC.ForeColor = vbWhite
    Y_PIC.Line (250, 50)-(250, 500), vbWhite
    Y_PIC.Line (250, 500)-(5800, 500), vbWhite
    Y_PIC.CurrentX = 2700: Y_PIC.CurrentY = 560: Y_PIC.Print "Channel"
    For i = 1 To 110
        Y_PIC.Line ((i * 50) + 250, 500)-((i * 50) + 250, 510), vbWhite
    Next i
    For J = 1 To 29
        Y_PIC.Line (220, (J * 15) + 50)-(250, (J * 15) + 50), vbWhite
    Next J
    For i = 0 To 11
        Y_PIC.Line ((i * 500) + 500, 500)-((i * 500) + 500, 520), vbWhite
        Y_PIC.Line ((i * 500) + 250, 500)-((i * 500) + 250, 530), vbWhite
        Y_PIC.CurrentX = (i * 500) + 130
        Y_PIC.CurrentY = 530
        Y_PIC.Print i * 200
    Next i
End Sub

Private Sub cmdClear_Click()
On Error GoTo Err_cmdClear_Click
    txtStatus.Text = " ลบข้อมูล " & vbCrLf
    X_PIC.Cls
    DrawLine
    For K = 0 To 2047
        DataChannel(K) = 0
    Next K
    MouseWK = 0
    X_PIC.ScaleWidth = 2150: X_PIC.ScaleHeight = 500
    Xpointer = 0: Ypointer = 0: Zpointer = 0
Exit_cmdClear_Click:
    Exit Sub
Err_cmdClear_Click:
    txtStatus.Text = txtStatus.Text & Err.Description & " - Error number: " & Err.Number & vbCrLf
    txtStatus.SelStart = Len(txtStatus.Text)
    Resume Exit_cmdClear_Click
End Sub

Private Sub cmdClose_Click()
On Error GoTo Err_cmdClose_Click
    txtStatus.Text = " ปิดพอร์ตข้อมูล " & vbCrLf
```

```
    MSComm1.PortOpen = False
Exit_cmdClose_Click:
    Exit Sub
Err_cmdClose_Click:
    txtStatus.Text = txtStatus.Text & Err.Description & " - Error number: " & Err.Number & vbCrLf
    txtStatus.SelStart = Len(txtStatus.Text)
    Resume Exit_cmdClose_Click
End Sub

Private Sub cmdDec_Click()
    H1 = H1 / 2
    X_PIC.Cls
    For J = Xpointer To 2048
        X_PIC.PSet (J - Xpointer, X1 - (DataChannel(J) * H1)), vbGreen
        X_PIC.Line (J - Xpointer, X1)-(J - Xpointer, X1 - (DataChannel(J) * H1)), vbYellow
    Next J
    For i = 1 To 100          'Show ROI Area
        If ROLindex(i) <> 0 Then
            Call ROIA(ROLindex(i), RORindex(i))
        Else
            Exit For
        End If
    Next i
    txtStatus.Text = txtStatus.Text & "Stretch " & H1 & vbCrLf
    txtStatus.SelStart = Len(txtStatus.Text)
End Sub

Private Sub cmdExit_Click()
    If MSComm1.PortOpen Then
        MSComm1.PortOpen = False
    End If
    End
End Sub

Private Sub cmdExpand_Click()
    MouseWK = 1
End Sub

Private Sub cmdInc_Click()
    H1 = H1 * 2
    X_PIC.Cls
```

```
    For J = Xpointer To 2048
        X_PIC.PSet (J - Xpointer, X1 - (DataChannel(J) * H1)), vbGreen
        X_PIC.Line (J - Xpointer, X1)-(J - Xpointer, X1 - (DataChannel(J) * H1)), vbYellow
    Next J
    For i = 1 To 100
        If ROLindex(i) <> 0 Then
            Call ROIA(ROLindex(i), RORindex(i))
        Else
            Exit For
        End If
    Next i
    txtStatus.Text = txtStatus.Text & "Stretch " & H1 & vbCrLf
    txtStatus.SelStart = Len(txtStatus.Text)
End Sub

Private Sub cmdOpenPort_Click()
On Error GoTo Err_cmdOpenPort_Click
        txtStatus.Text = " เปิดพอร์ตข้อมูล " & vbCrLf
        MSComm1.PortOpen = True
Exit_cmdOpenPort_Click:
    Exit Sub
Err_cmdOpenPort_Click:
    txtStatus.Text = txtStatus.Text & Err.Description & " - Error number: " & Err.Number & vbCrLf
    txtStatus.SelStart = Len(txtStatus.Text)
    Resume Exit_cmdOpenPort_Click
End Sub

Private Sub cmdRestore_Click()
    MouseWK = 0
    X_PIC.ScaleWidth = 2150: X_PIC.ScaleHeight = 500
    Xpointer = 0: Ypointer = 0: Zpointer = 0
    X1 = 495: H1 = 1: W1 = 0
    DrawPic
End Sub

Private Sub cmdRoll_Click()
    MouseWK = 2
End Sub

Private Sub cmdStart_Click()
On Error GoTo Err_cmdStart_Click
```

```
If Not Timer1.Enabled Then
      txtStatus.Text = " เริ่มอ่านข้อมูล " & vbCrLf
      Timer1.Enabled = True
      CountSB = 0
      A = 0: B = 0
      For i = 0 To 2047
         DataChannel(i) = 0
      Next i
      pausetime = Val(txtSetTime.Text)    ' Set duration.
      start = Timer   ' Set start time.
      Do While (Timer < start + pausetime) And Timer1.Enabled
         DoEvents    ' Yield to other processes
         txtShowTime.Text = " เหลือเวลาอีก " & pausetime - Int(Timer - start) & " วินาที "
      Loop
      Timer1.Enabled = False
   End If
Exit_cmdStart_Click:
   Exit Sub
Err_cmdStart_Click:
   txtStatus.Text = txtStatus.Text & Err.Description & " - Error number: " & Err.Number & vbCrLf
   txtStatus.SelStart = Len(txtStatus.Text)
   Resume Exit_cmdStart_Click
End Sub

Private Sub cmdStop_Click()
On Error GoTo Err_cmdStop_Click
   txtStatus.Text = " หยุดการทำงาน " & vbCrLf
   Timer1.Enabled = False
Exit_cmdStop_Click:
   Exit Sub
Err_cmdStop_Click:
   txtStatus.Text = txtStatus.Text & Err.Description & " - Error number: " & Err.Number & vbCrLf
   txtStatus.SelStart = Len(txtStatus.Text)
   Resume Exit_cmdStop_Click
End Sub

Private Sub Form_Load()
   '===== Comm Port Initial Setting ============================
   BPS = "9600"
   PAR = "N"
   DB = "8"
```

```
SB = "1"
COMP = "4"
MSComm1.Settings = BPS & PAR & DB & SB
MSComm1.CommPort = COMP
MSComm1.InputLen = 1
MSComm1.RThreshold = 1
'============================================================
DrawLine
Xpointer = 0: Ypointer = 0: Zpointer = 0
X1 = 500: H1 = 1: W1 = 0
ROI = False
KeyPreview = True
For i = 0 To 100
   ROLindex(i) = 0
   RORindex(i) = 0
Next i
ROIindex = 1
Intercept = 0: Gain = 1
End Sub

Private Sub MSComm1_OnComm()
Dim tempbuffer As Variant
Dim bytebuffer() As Byte

  If (MSComm1.PortOpen) And (MSComm1.InBufferCount <> 1) Then
    If CountSB = 0 Then
      B = CInt(Asc(MSComm1.Input) - 48)
      CountSB = 1
    Else
      A = CInt(Asc(MSComm1.Input) - 48)
      CommData = (A * 256) + B
      If (CommData > 0) And (CommData < 2048) Then
        DataChannel(CommData) = DataChannel(CommData) + 1
        X_PIC.PSet (CommData, X1 - DataChannel(CommData)), vbGreen
      End If
      CountSB = 0
    End If
  End If
End Sub

Private Sub Timer1_Timer()
```

```
On Error GoTo Err_Timer1_Click
   For i = 1 To 1000
      DoEvents
      MSComm1.Output = "0"
   Next i
Exit_Timer1_Click:
   Exit Sub
Err_Timer1_Click:
   txtStatus.Text = txtStatus.Text & Err.Description & " - Error number: " & Err.Number & vbCrLf
   txtStatus.SelStart = Len(txtStatus.Text)
   Resume Exit_Timer1_Click
End Sub

Private Sub X_PIC_MouseDown(Button As Integer, Shift As Integer, x As Single, y As Single)
   X_PIC.DrawMode = 13
   Xpos = x: Ypos = y
   If Shift = 1 Then
      X_PIC.Line (Zpointer + 1 - Xpointer, X1)-(Zpointer + 1 - Xpointer, X1 - (DataChannel(Zpointer + 1) * H1)),
&HFF8000  '&H80000002
      X_PIC.Line (Ypointer - Xpointer, (X1 - 5) - (DataChannel(Ypointer) * H1))-(Ypointer - Xpointer, (X1 - 30) -
(DataChannel(Ypointer) * H1)), &HFF8000  '&H80000002
      'ROI = True
      XXStart = Zpointer
   Else
      X_PIC.Line (Ypointer - Xpointer, (X1 - 5) - (DataChannel(Ypointer) * H1))-(Ypointer - Xpointer, (X1 - 30) -
(DataChannel(Ypointer) * H1)), RGB(255, 130, 0)
   End If
End Sub

Private Sub X_PIC_MouseMove(Button As Integer, Shift As Integer, x As Single, y As Single)
   If Button = 1 Then
      '====================
      If (Xpos < x) Then
         Zpointer = Zpointer + 1
      Else
         Zpointer = Zpointer - 1
      End If
      'Debug.Print Zpointer, Ypointer, Xpointer
      '====================
      If (Zpointer >= 0) Then
         If (Zpointer < 2048) Then
```

```
            txtX.Text = " (" & Zpointer & ") "
            txtY.Text = DataChannel(Zpointer)
            txtE.Text = (Zpointer) * Gain + Intercept & "keV"
            If Shift = 1 And ROI Then
              X_PIC.DrawMode = 13
              X_PIC.MousePointer = vbSizeWE
              If XXStart <= Zpointer Then
                X_PIC.Line (Zpointer - Xpointer, X1)-(Zpointer - Xpointer, X1 - (DataChannel(Zpointer) * H1)),
vbMagenta
                ROL = XXStart: ROR = Zpointer
                Debug.Print "ROL", ROL, "ROR", ROR
              End If
              X_PIC.Line (Zpointer + 1 - Xpointer, X1)-(Zpointer + 1 - Xpointer, X1 - (DataChannel(Zpointer + 1) * H1)),
vbYellow
            Else
              X_PIC.DrawMode = 7
              X_PIC.Line (Zpointer - Xpointer, (X1 - 5) - (DataChannel(Zpointer) * H1))-(Zpointer - Xpointer, (X1 - 30) -
(DataChannel(Zpointer) * H1)), vbWhite
              X_PIC.Line (Ypointer - Xpointer, (X1 - 5) - (DataChannel(Ypointer) * H1))-(Ypointer - Xpointer, (X1 - 30) -
(DataChannel(Ypointer) * H1)), vbWhite
            End If
          Else
            Zpointer = 2048
          End If
        Else
          Zpointer = 0
        End If
        Ypointer = Zpointer
      '=====================================================================
      Else
        If Button = 2 Then
        Select Case MouseWK
          Case 1
              If (X_PIC.ScaleWidth > 50) Then
                If (Xpos <= x) Then
                  X_PIC.ScaleWidth = X_PIC.ScaleWidth - 50
                Else
                  X_PIC.ScaleWidth = X_PIC.ScaleWidth + 50
                End If
                DrawPic  ' ================== Draw Spectrum ====================
                For i = 1 To 100                    ' Draw ROI area
```

```
            If ROLindex(i) <> 0 Then
               Call ROIA(ROLindex(i), RORindex(i))
            Else
               Exit For
            End If
         Next i
      Else
         MsgBox ("Out of Range")
         X_PIC.ScaleWidth = 100
      End If
      txtStatus.Text = txtStatus.Text & "Expand " & X_PIC.ScaleWidth & vbCrLf
      txtStatus.SelStart = Len(txtStatus.Text)
Case 2
      If (Xpos <= x) Then
         Xpointer = Xpointer - 10
      Else
         Xpointer = Xpointer + 10
      End If
      If Xpointer > 0 Then
         If Xpointer < 2048 Then
            X_PIC.Cls
            W1 = Xpointer
            For T = Xpointer To 2048
               X_PIC.PSet (T - Xpointer, X1 - (DataChannel(T) * H1)), vbGreen
               X_PIC.Line (T - Xpointer, X1)-(T - Xpointer, X1 - (DataChannel(T) * H1)), vbYellow
            Next T
            For i = 1 To 100                    ' Draw ROI area
            If ROLindex(i) <> 0 Then
               Call ROIA(ROLindex(i), RORindex(i))
            Else
               Exit For
            End If
         Next i
         Else
            Xpointer = 2048
         End If
      Else
         Xpointer = 1
      End If
      txtStatus.Text = txtStatus.Text & "Roll" & vbCrLf
      txtStatus.SelStart = Len(txtStatus.Text)
```

```
    End Select
    End If
  End If
End Sub

Private Sub X_PIC_MouseUp(Button As Integer, Shift As Integer, x As Single, y As Single)
  X_PIC.MousePointer = vbDefault
  X_PIC.DrawMode = 13
  ROI = False
  Xcursor = Zpointer
End Sub

Private Sub XExit_Click()
  End
End Sub

Private Sub XOpen_Click()
Dim Myvalue As Long
Dim Graphfile As String
Dim InputData As String
Dim MAX, YValue As Integer
On Error GoTo Err_XOpen_Click

  X_PIC.Cls
  txtStatus.Text = "Open Spectrum Files"
  For i = 0 To 2047
    DataChannel(i) = 0
  Next i
    '================== Set Value ==============================
  MouseWK = 0
  X_PIC.ScaleWidth = 2150: X_PIC.ScaleHeight = 500
  Xpointer = 0: Ypointer = 0: Zpointer = 0
  X1 = 495: H1 = 1: W1 = 0
  '==================== Open Files ==============================
  i = 0: J = 0
  CommonDialog1.ShowOpen
  Graphfile = CommonDialog1.filename
  Open Graphfile For Input As #1    ' Open file for input.
  Do While Not EOF(1) ' Check for end of file.
    Line Input #1, InputData    ' Read line of data.
    Myvalue = Val(InputData)
```

```
    DataChannel(i) = Myvalue
    i = i + 1
  Loop
  i = i - 1
  Debug.Print "OK"
  '================== Find Maximum Value ====================
  MAX = 0
  For J = 0 To i
    If DataChannel(J) > MAX Then
      MAX = DataChannel(J)
    End If
  Next J
  '================== Plot Graph ==============================
  YValue = MAX \ 100
  For J = 0 To i
    X_PIC.PSet (J - W1, X1 - DataChannel(J)), vbGreen
    X_PIC.Line (J - W1, X1)-(J - W1, X1 - DataChannel(J)), vbYellow
  Next J
  '================== Close Files ============================
  Close #1    ' Close file.

Exit_XOpen_Click:
  Exit Sub
Err_XOpen_Click:
  txtStatus.Text = txtStatus.Text & Err.Description & " - Error number: " & Err.Number & vbCrLf
  txtStatus.SelStart = Len(txtStatus.Text)
  Resume Exit_XOpen_Click
End Sub

Private Sub XSave_Click()
On Error GoTo Err_XSave_Click
Dim SaveGraph
  CommonDialog1.Filter = "All Files (*.*)|*.*|Text Files (*.txt)|*.txt"
  CommonDialog1.FilterIndex = 2
  CommonDialog1.ShowSave
  SaveGraph = Trim(CommonDialog1.filename)
  If Not (SaveGraph = "") Then
    Open SaveGraph For Output As #1    ' Open file for output.
    For i = 0 To 2047
      Print #1, DataChannel(i) 'write datat to file
    Next i
```

```
        Close #1
    End If
Exit_XSave_Click:
    Exit Sub
Err_XSave_Click:
    txtStatus.Text = txtStatus.Text & Err.Description & " - Error number: " & Err.Number & vbCrLf
    txtStatus.SelStart = Len(txtStatus.Text)
    Resume Exit_XSave_Click
End Sub
```

โปรแกรม Energy Calibration

```
Dim ITEM(100) As Integer
Dim y, x, a0, a1 As Single
Dim sigma_y, sigma_x, sigma_xy, sigma_x2 As Long
Dim N, E, F As Integer

Private Sub LeastSquare()
   N = lstCount.ListCount
   sigma_y = 0: sigma_x = 0: sigma_xy = 0: sigma_x2 = 0
   For E = 0 To N - 1
      sigma_y = sigma_y + (lstEnergy.List(E))
      sigma_x = sigma_x + (lstChannel.List(E))
      sigma_xy = sigma_xy + ((lstChannel.List(E)) * (lstEnergy.List(E)))
      sigma_x2 = sigma_x2 + ((lstChannel.List(E)) ^ 2)
   Next E
   a1 = ((sigma_x * sigma_y) - (N * sigma_xy)) / ((sigma_x * sigma_x) - (N * sigma_x2))
   a0 = ((sigma_xy - sigma_y) - ((sigma_x2 - sigma_x) * a1)) / (sigma_x - N)
   If a0 >= 0 Then
      txta1.Text = a1: txta0.Text = "+" & a0
   Else
      txta1.Text = a1: txta0.Text = "-" & Abs(a0)
   End If
   Gain = a1: Intercept = a0
End Sub
Private Sub cmdAccept_Click()
Dim DD As Integer
   lstEnergy.AddItem txtEnergy.Text
   lstEnergy.ListIndex = lstEnergy.NewIndex
   '==================
   lstChannel.AddItem txtChannel.Text
   lstChannel.ListIndex = lstChannel.NewIndex
   '==================
   lstCount.AddItem " "
   lstCount.ListIndex = lstCount.NewIndex
End Sub

Private Sub cmdCalculate_Click()
   Call LeastSquare
End Sub
```

```
Private Sub cmdCancel_Click()
    Form2.Visible = False
End Sub

Private Sub cmdClear_Click()
    lstEnergy.Clear
    lstChannel.Clear
    lstCount.Clear
End Sub

Private Sub cmdDel_Click()
    If (lstEnergy.ListIndex <> -1) And (lstEnergy.List(lstEnergy.ListIndex) <> ">") Then
        lstEnergy.RemoveItem (lstEnergy.ListIndex)
    End If
    If (lstChannel.ListIndex <> -1) And (lstChannel.List(lstChannel.ListIndex) <> ">") Then
        lstChannel.RemoveItem (lstChannel.ListIndex)
    End If
    If (lstCount.ListIndex <> -1) Then
        lstCount.RemoveItem (lstCount.ListIndex)
    End If
End Sub

Private Sub cmdLocate_Click()
    txtChannel.Text = Xcursor
End Sub

Private Sub cmdOK_Click()
    Form2.Visible = False
End Sub

Private Sub lstCount_Click()
    lstEnergy.ListIndex = lstCount.ListIndex
    lstChannel.ListIndex = lstCount.ListIndex
    Debug.Print lstCount.ListIndex
End Sub
```

ภาคผนวก จ

USB FIFO - Fast Parallel Data Transfer IC

FEATURES

- Single Chip Fast Data Transfer Solution
- Send / Receive Data over USB at up to 1 M Bytes / sec
- 384 byte FIFO Transmit buffer / 128 byte FIFO receive buffer for high data throughput
- Simple interface to CPU or MCU bus
- No in-depth knowledge of USB required as all USB Protocol is handled automatically within the I.C.
- FTDI's Virtual COM port drivers eliminate the need for USB driver development in most cases.
- Compact 32 pin (7mm x 7mm) MQFP package
- Integrated 6MHz - 48MHz Clock Multiplier aids FCC and CE compliance
- Integrated 3.3v Regulator – No External Regulator Required
- 4.4v .. 5.25v Single Supply Operation
- UHCI / OHCI Compliant
- USB 1.1 Specification Compliant
- USB VID, PID, Serial Number and Product Description Strings in external E2PROM.

Virtual COM Port Drivers for –

Windows 98 and Windows 98 SE

Windows 2000

Windows Millennium **

Apple iMAC **

Linux **

- Application Areas

USB ISDN and ADSL Modems

High Speed USB ⇔ PDA Communications

USB I/F for Digital Cameras

USB I/F for MP3 players

High Speed USB Instrumentation

USB ⇔ USB data transfer cables

USB ⇔ USB null-modem cables

GENERAL DESCRIPTION

The FT8U245AM provides an easy cost-effective method of transferring data to / from a peripheral and a host P.C. at up to 8 Million bits (1 Megabyte) per second. It's simple FIFO-like design makes it easy to interface to any CPU (MCU) either by mapping the device into the Memory / IO map of the CPU, using DMA or controlling the device via IO ports.

To send data from the peripheral to the host P.C. simply write the byte wide data into the device when the transmitter empty status bit is not active. If the (384 byte) transmit buffer fills up, the device de-asserts transmit empty in order to stop further data being written to the device until some of the FIFO data has been transferred over USB.

When the host P.C. sends data to the peripheral over USB, the device will assert the receiver full status bit to let the peripheral know that data is available. The peripheral then reads the data until the receiver full status bit goes inactive, indicating no more data is available to read.

By using FTDI's virtual COM Port drivers, the peripheral looks like a standard COM Port to the application software. Commands to set the baud rate are ignored – the device always transfers data at it's fastest rate regardless of the application's baud rate setting.

Figure 1 – FT8U245AM Block Diagram (Simplified)

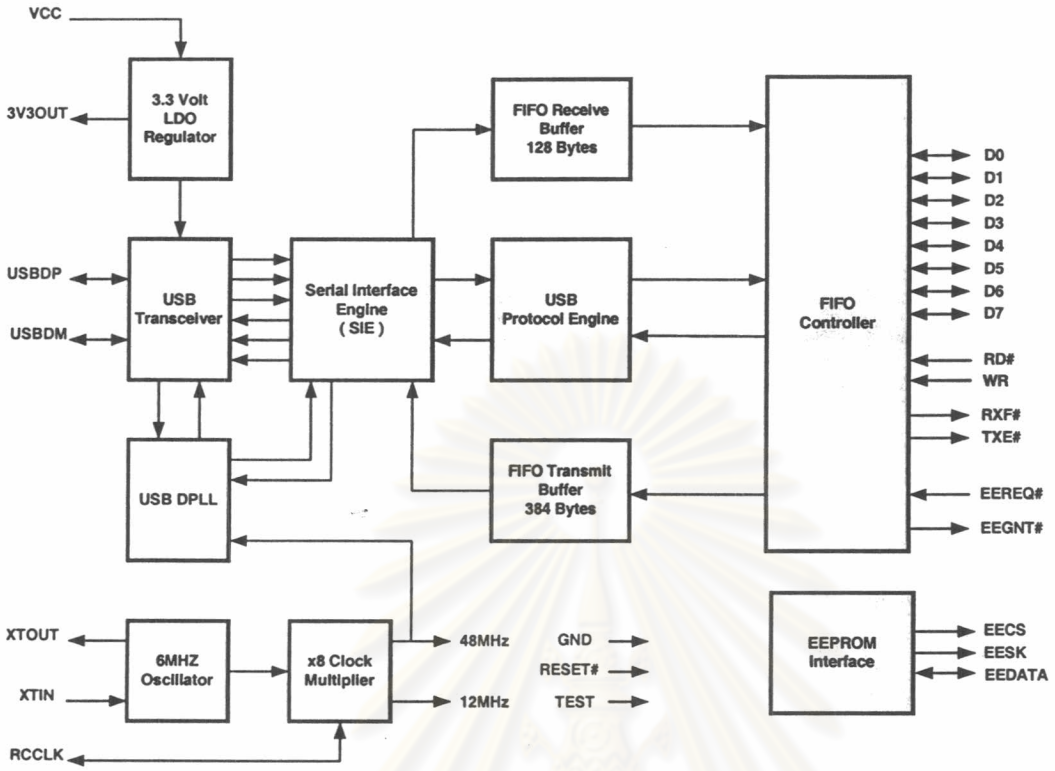
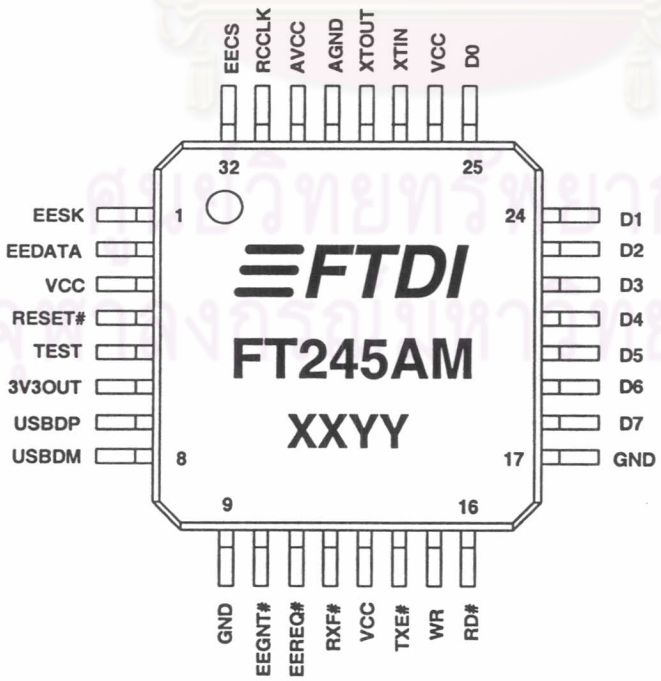


Figure 2 – FT8U245AM I.C. Pinout



FT8U245AM - FUNCTIONAL BLOCK DESCRIPTION

- **3.3V LDO Regulator**

  The 3.3V LDO Regulator generates the 3.3 volt reference voltage for driving the USB transceiver cell output buffers. It requires an external decoupling capacitor to be attached to the 3V3OUT regulator output pin.

- **USB Transceiver**

  The USB Transceiver Cell provides the USB 1.1 full-speed physical interface to the USB cable. The output drivers provide 3.3 volt level slew rate control signalling, whilst a differential receiver and two single ended receivers provide USB data in, SEO and USB Reset condition detection.

- **USB DPLL**

  The USB DPLL cell locks on to the incoming NRZI USB data and provides separate recovered clock and data signals to the SIE block.

- **6MHz Oscillator**

  The 6MHz Oscillator cell generates a 6MHz reference clock input to the X8 Clock multiplier from an external 6MHz crystal or ceramic resonator.

- **X8 Clock Multiplier**

  The X8 Clock Multiplier takes the 6MHz input from the Oscillator cell and generates a 12MHz reference clock for the SIE, USB Protocol Engine and UART FIFO controller blocks. It also generates a 48MHz reference clock for the USB DPPL and the Baud Rate Generator blocks.

- **Serial Interface Engine (SIE)**

  The Serial Interface Engine (SIE) block performs the Parallel to Serial and Serial to Parallel conversion of the USB data. In accordance to the USB 1.1 specification, it performs bit stuffing / un-stuffing and CRC5 / CRC16 generation / checking on the USB data stream.

- **USB Protocol Engine**

  The USB Protocol Engine manages the data stream from the device USB control endpoint. It handles the low level USB protocol (Chapter 9) requests generated by the USB host controller and the commands for controlling the functional parameters of the UART.

- **Fifo Receive Buffer (128 bytes)**

  Data sent from the USB Host to the FIFO via the USB data out endpoint is stored in the FIFO Receive Buffer and is removed from the buffer by reading the FIFO contents using RD#.

- **FIFO Transmit Buffer (384 bytes)**

  Data written into the FIFO using WR# is stored in the FIFO Transmit Buffer. The Host removes Data from the FIFO Transmit Data by sending a USB request for data from the device data in endpoint.

- **FIFO Controller**

  The FIFO Controller handles the transfer of data between the external FIFO interface pins and the FIFO Transmit and Receive buffers.

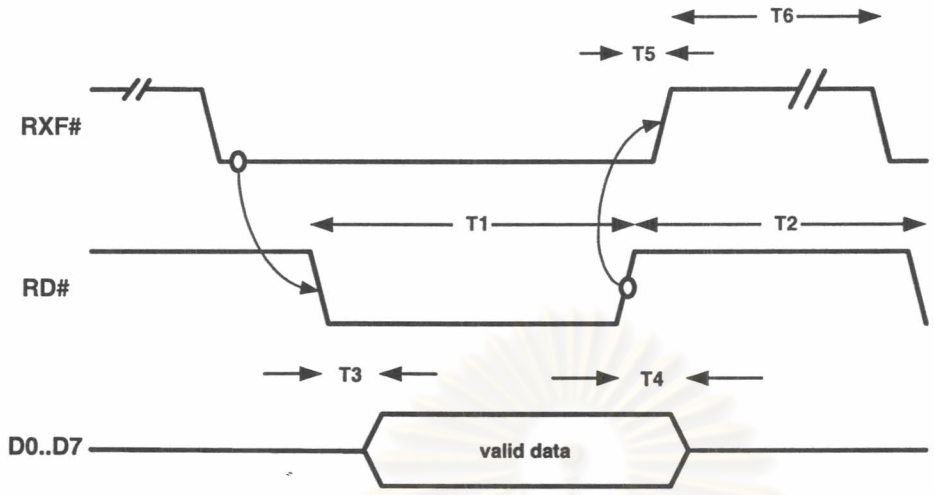
- **EEPROM Interface**

  The FT8U245AM uses an external 93C46 EEPROM to customise the USB VID, PID, Serial Number and Strings of the FT8U245AM for OEM applications. The FT8U245 Virtual Com Port Drivers rely on a unique device serial number for to bind a unique virtual COM port to each individual device.

Table 1 - FT8U245AM - PINOUT DESCRIPTION

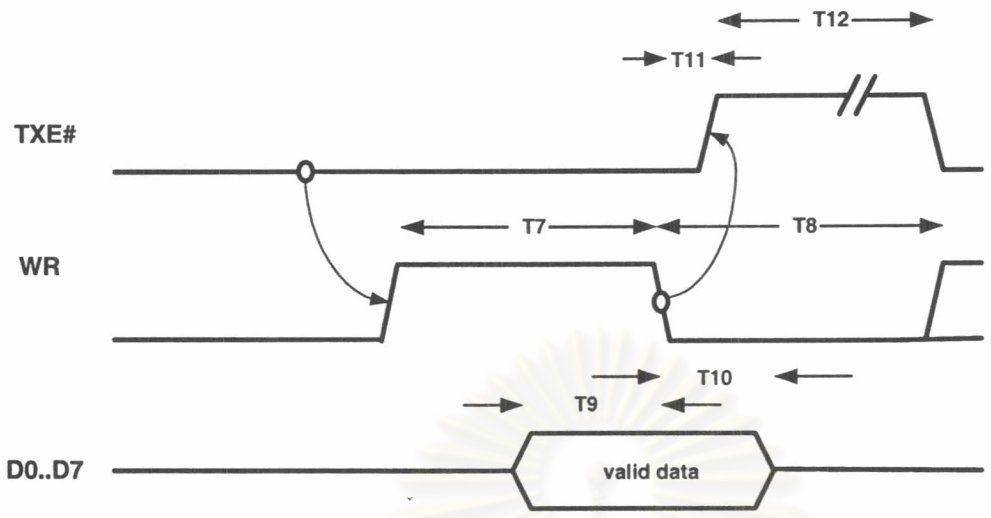
Pin #	Signal	Type	Description
7	USBDP	I/O	USB Data Signal Plus – Requires 1.5k pull-up to 3V3OUT
8	USBDM	I/O	USB Data Signal Minus
6	3V3OUT	OUT	3.3 volt Output from integrated regulator
27	XTIN	IN	Input to 6MHz Crystal Oscillator Cell
28	XTOUT	OUT	Output from 6MHz Crystal Oscillator Cell
31	RCCLK	I/O	RC timer – used to guarantee clock stability on exiting sleep mode. Clamped low during reset or sleep condition.
4	RESET#	IN	Resets entire device using external RC network
32	EECS	I/O	Optional EEPROM – Chip Select
1	EESK	I/O	Optional EEPROM – Clock
2	EEDATA	I/O	Optional EEPROM – Data I/O
5	TEST	IN	Puts device in i.c. test mode – must be tied to GND
25	D0	I/O	Bi-directional Data Bus Bit # 0
24	D1	I/O	Bi-directional Data Bus Bit # 1
23	D2	I/O	Bi-directional Data Bus Bit # 2
22	D3	I/O	Bi-directional Data Bus Bit # 3
21	D4	I/O	Bi-directional Data Bus Bit # 4
20	D5	I/O	Bi-directional Data Bus Bit # 5
19	D6	I/O	Bi-directional Data Bus Bit # 6
18	D7	I/O	Bi-directional Data Bus Bit # 7
16	RD#	IN	Enables Current FIFO Data Byte on D0..D7.when low. Fetches the next FIFO Data Byte (if available) from the Receive FIFO Buffer when RD# goes from low to high.
15	WR	IN	Writes the Data Byte on the D0..D7 into the Transmit FIFO Buffer when WR goes from high to low.
14	TXE#	OUT	When high, do not write data into the FIFO. When low, data can be written into the FIFO by strobing WR high then low.
12	RXF#	OUT	When high, do not read data from the FIFO. When low, there is data available in the FIFO which can be read by strobing RD# low then high again.
11	EEREQ#	IN	Requests the EEPROM contents to be accessed via the Data Bus.
10	EEGNT#	OUT	When low, allows the EEPROM contents to be accessed via the Data Bus.
3,13,26	VCC	PWR	Device - +4.4 volt to +5.25 volt Power Supply Pins
9.17	GND	PWR	Device – Ground Supply Pins
30	AVCC	PWR	Device - Analog Power Supply for the internal x8 clock multiplier
29	AGND	PWR	Device - Analog Ground Supply for the internal x8 clock multiplier

FT8U245AM TIMING DIAGRAM – FIFO READ CYCLE



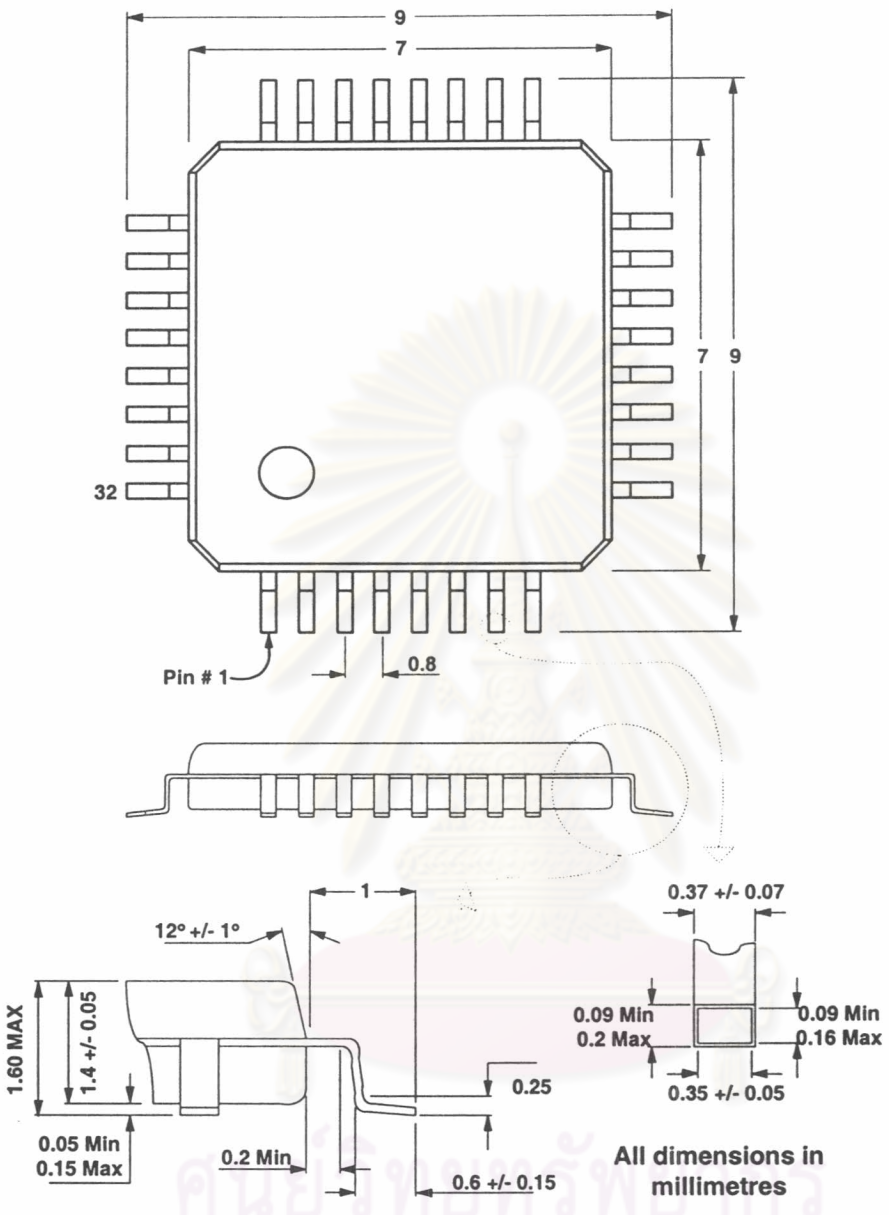
Time	Description	Min	Max	Unit
T1	RD Active Pulse Width	50		ns
T2	RD to RD Pre-Charge Time	50		ns
T3	RD Active to Valid Data		30	ns
T4	Valid Data Hold Time from RD inactive	10		ns
T5	RD Inactive to RXF#	5	25	ns
T6	RXF inactive after RD cycle	80		ns

FT8U245AM TIMING DIAGRAM – FIFO WRITE CYCLE



Time	Description	Min	Max	Unit
T7	WR Active Pulse Width	50		ns
T8	WR to WR Pre-Charge Time	50		ns
T9	Data Setup Time before WR inactive		20	ns
T10	Data Hold Time from WR inactive	10		ns
T11	WR Inactive to TXE#	5	25	ns
T12	TXE inactive after RD cycle	80		ns

Figure 3. FT8U245AM - PACKAGE DESCRIPTION – QFP 7mm x 7mm



Absolute Maximum Ratings

Storage Temperature	-65°C to + 150°C
Ambient Temperature (Power Applied)	0°C to + 70°C
VCC Supply Voltage	-0.5v to +6.00v
DC Input Voltage - Inputs	-0.5v to VCC + 0.5v
DC Input Voltage - High Impedance Bidirectionals	-0.5v to VCC + 0.5v
DC Output Current – Outputs	24mA
DC Output Current – Low Impedance Bidirectionals	24mA
Power Dissipation	500mW

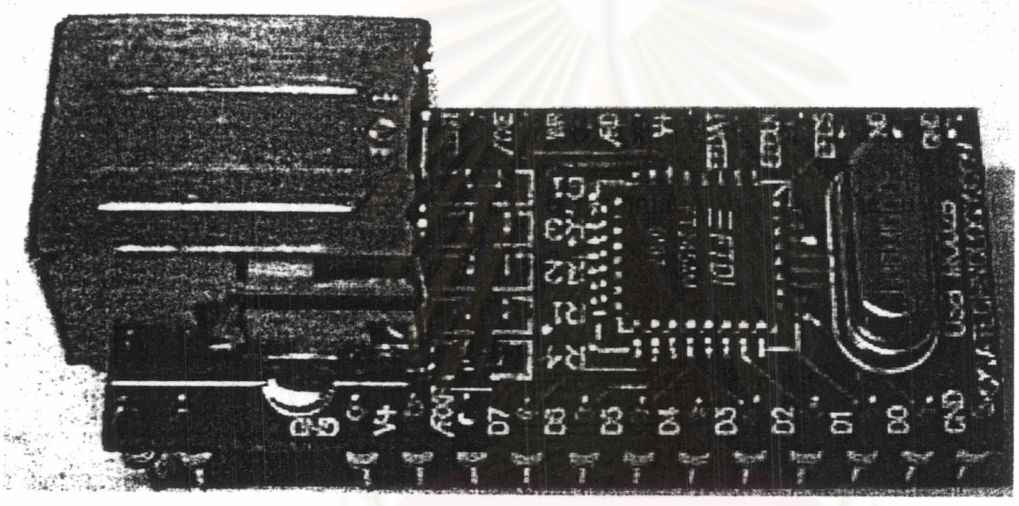
DC Characteristics (Ambient Temperature = 0 .. 70 Degrees C)

	Description	Min	Max	Units	Conditions
VCC	Operating Supply Voltage	4.4	5.25	v	
Icc1	Operating Supply Current		50	mA	Normal Operation
Icc2	Operating Supply Current		250	uA	USB Suspend
Ioh1	Digital IO Pins Source Current	4		mA	Voh = VCC – 0.5v
Iol1	Digital IO Pins Sink Current	4		mA	Vol = + 0.5v
Voh1	Input Voltage Threshold (Low)		0.6	v	
Vol1	Input Voltage Threshold (High)	2.7		v	
VDif	USB Differential Input Sensitivity	0.2		v	
VCom	USB Differential Common Mode	0.8	2.5	v	
URxt	USB Single Ended Rx Threshold	0.8	2.0	v	
UVh	USB IO Pins Static Output (Low)		0.3v		Rl = 1.5k to 3.6v
UVl	USB IO Pins Static Output (High)	2.8			Rl = 15k to GND

USB MOD2 - USB Plug and Play Parallel 8-Bit FIFO Development Module

The USBMOD2 is a low-cost integrated module for transferring data to / from a peripheral and a host P.C at up to 8 Million bit (1 Megabyte) per second. Based on the FTDI FT8U245 USB FIFO – Fast Parallel Data Transfer IC, it's simple FIFO-like design makes it easy to interface to an CPU (MCU) either by mapping the device into the memory / I/O map of the PCU, using DMA or controlling the device via IO ports.

The USBMOD2 is ideal for rapid prototyping and development by offering a complete plug and play solution.



MODULE FEATURES

- Single module High-Speed USB UART solution
- Based on FTDI FT8U245 USB FIFO – Fast Parallel Data Transfer IC
- Integrated Type-B USB Connector
- On-board 6MHz Crystal
- Provision for external EEPROM for USB enumeration data
- No external passive components required
- Module powered from USB bus (up to 60mA from USB for user application)
- 32-pin Dual In-Line Package (Ideal for prototyping)
- Fits into a standard 32-pin 600mil IC Socket

FT8U245 IC FEATURES

- Single Chip Multi-Function Data Transfer Solution
- Send / Receive Data over USB at up to 1 Mb / Sec
- 384 byte receive buffer / 128 byte transmit buffer for high data throughput
- Simple interface to CPU or MCU bus
- No in-depth knowledge of USB required as all USB Protocol is handled automatically within the I.C
- FTDI's Virtual COM port drivers eliminate the need for USB driver development in most cases.
- Compact 32 pin (7mm x 7mm) MQFP package
- Integrated 6Mhz – 48Mhz Clock Multiplier aids FCC and CE compliance
- Integrated 3.3v Regulator – No External Regulator Required
- UHCI / OHCI Compliant
- USB 1.1 Specification Compliant
- USB VID, PID, Serial Number and Product Description Strings in external E2PROM.

VIRTUAL COM PORT (VCP)
DRIVERS for

- Windows 98 and Windows 98 SE
- Windows 2000 / ME / XP
- Windows CE **
- MAC OS-8 and OS9
- MAC OS-X
- Linux 2.40 and greater

[** = In the planning or under development]

D2XX
(USB Direct Drivers + DLL S/W Interface)

- Windows 98 and Windows 98 SE
- Windows 2000 / ME / XP

For further information regarding the FTDI FT8U245AM USB FIFO – Fast Parallel Data Transfer IC please refer to the FT8U245AM Datasheet. This datasheet can be found on the Ravar website at **http://www.ravar.net**

As mentioned above in module features, the USB MOD2 is in a 32-pin Dual In-Line Package. This allows the module to fit into a standard 32-pin 600mil IC Socket which makes the module ideal for prototyping and development work. Shown in diagram 2 below is the pin out for the USB MOD2.

USB MOD2 PINOUT

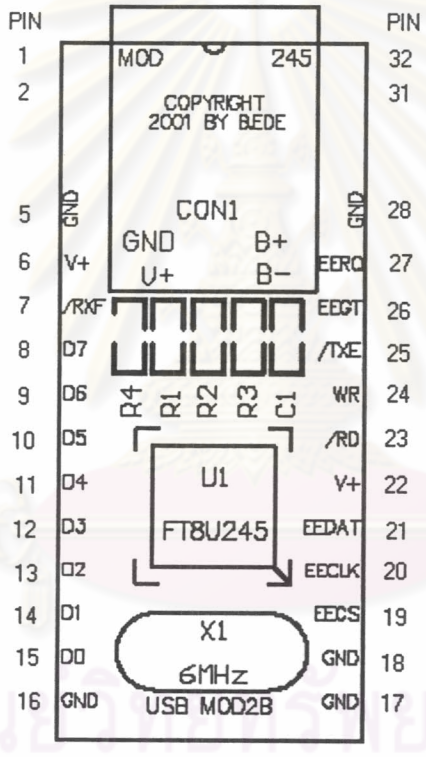


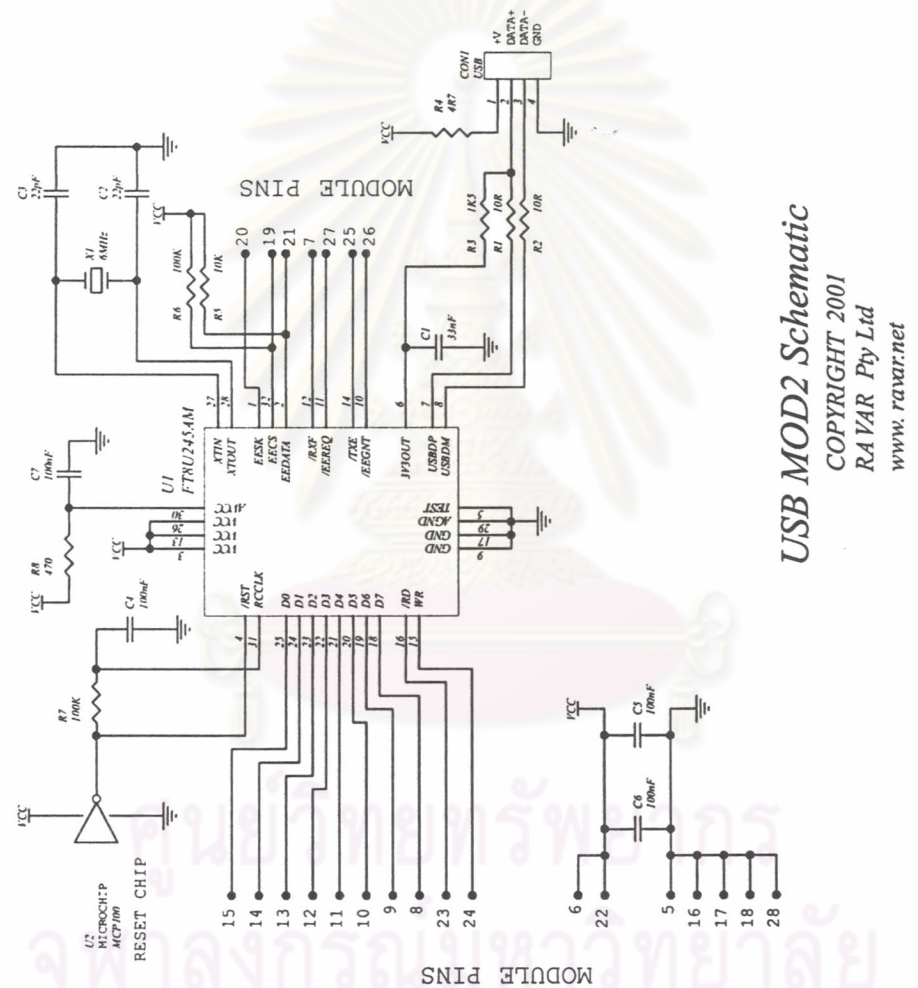
DIAGRAM 2

On the following page is the pin out table showing what the various pins are on the module.

USBMOD2 PINOUT

PIN #	SIGNAL	TYPE	DESCRIPTION
1	N/C	MOUNT	Mounting Pin for module USB connector support
2	N/C	MOUNT	Mounting Pin for module USB connector support
3	NO PIN	NO PIN	
4	NO PIN	NO PIN	
5	GND	PWR	Device – Ground Supply Pin
6	V+	PWR	Device - +4.4 volt to +5.25 volt Power Supply Pin
7	/RXF	OUT	When high, do not read data from FIFO. When low, there is data available in the FIFO which can be read by strobing RD# low the high again.
8	D7	I/O	Bi-Directional Data Bus Bit #7
9	D6	I/O	Bi-Directional Data Bus Bit #6
10	D5	I/O	Bi-Directional Data Bus Bit #5
11	D4	I/O	Bi-Directional Data Bus Bit #4
12	D3	I/O	Bi-Directional Data Bus Bit #3
13	D2	I/O	Bi-Directional Data Bus Bit #2
14	D1	I/O	Bi-Directional Data Bus Bit #1
15	D0	I/O	Bi-Directional Data Bus Bit #0
16	GND	PWR	Device – Ground Supply Pin
17	GND	PWR	Device – Ground Supply Pin
18	GND	PWR	Device – Ground Supply Pin
19	EECS	I/O	Optional EEPROM – Chip Select
20	EECLK	I/O	Optional EEPROM – Clock
21	EEDAT	I/O	Optional EEPROM – Data I/O
22	V+	PWR	Device - +4.4 volt to +5.25 volt Power Supply Pin
23	/RD	IN	Enables Current FIFO Data Byte on D0..D7 when low. Fetches the next FIFO Data Byte (if available) from the Receive FIFO Buffer when /RD goes from low to high.
24	WR	IN	Writes the Data Byte on D0..D7 into the Transmit FIFO Buffer when WR goes from high to low.
25	/TXE	OUT	When high, do not write data into the FIFO. When low, data can be written into the FIFO by strobing WR high then low.
26	EEGT	OUT	When low, allows the EEPROM contents to be accessed via the Data Bus
27	EERQ	IN	Requests the EEPROM contents to be accessed via the Data Bus.
28	GND	PWR	Device – Ground Supply Pin
29	NO PIN	NO PIN	
30	NO PIN	NO PIN	
31	N/C	MOUNT	Mounting Pin for module USB connector support
32	N/C	MOUNT	Mounting Pin for module USB connector support

USB MOD 2 SCHEMATIC



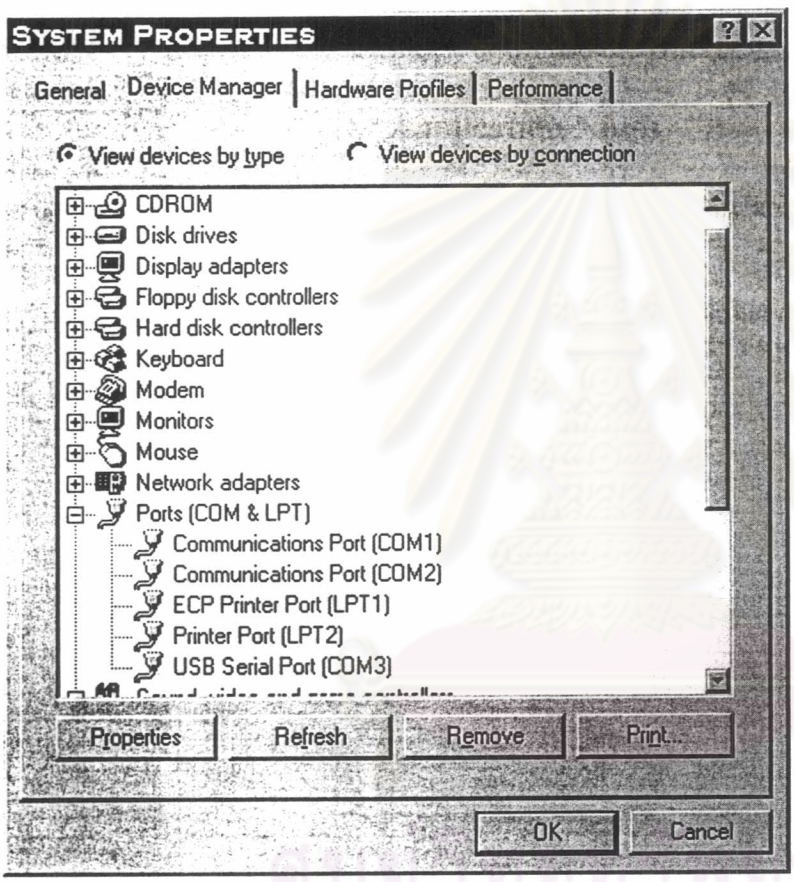
USB MOD2 Schematic

COPYRIGHT 2001
RA VAR Pty Ltd
www. ravar.net

Driver Installation.

Your first choice when using the USBMOD is whether you want to use the Virtual Com Port driver or the Direct DLL driver.

For programming simplicity the best driver is the Virtual Com Port and when installed the USBMOD will appear in the System Properties / Device Manager as an USB Serial Port (COMn) as follows.



The Com port number will vary depending on the number of existing com ports on your computer and the number of USBMODs connected to your system.

To install the Virtual Com Port drivers, download the driver from our website or the ftdichip.com website and unzip the files to a local directory. Then connect the USBMOD and windows will automatically ask for the driver, select to specify a location and browse to the directory where you have unzipped the files. (Use the Non Plug & Play driver for the USBMOD to avoid a delay on connecting the USBMOD)

Once the Virtual Com Port is installed it can be programmed exactly as a regular serial com port using the MSComm control from Visual Basic or API calls from C or other languages. Set the com port to the same number as appears in the Device Manager, the baud rate, stop bits, parity etc are not used as the device always runs at full speed.

The Direct DLL driver is installed in a similar manner but using the alternative download from the website.

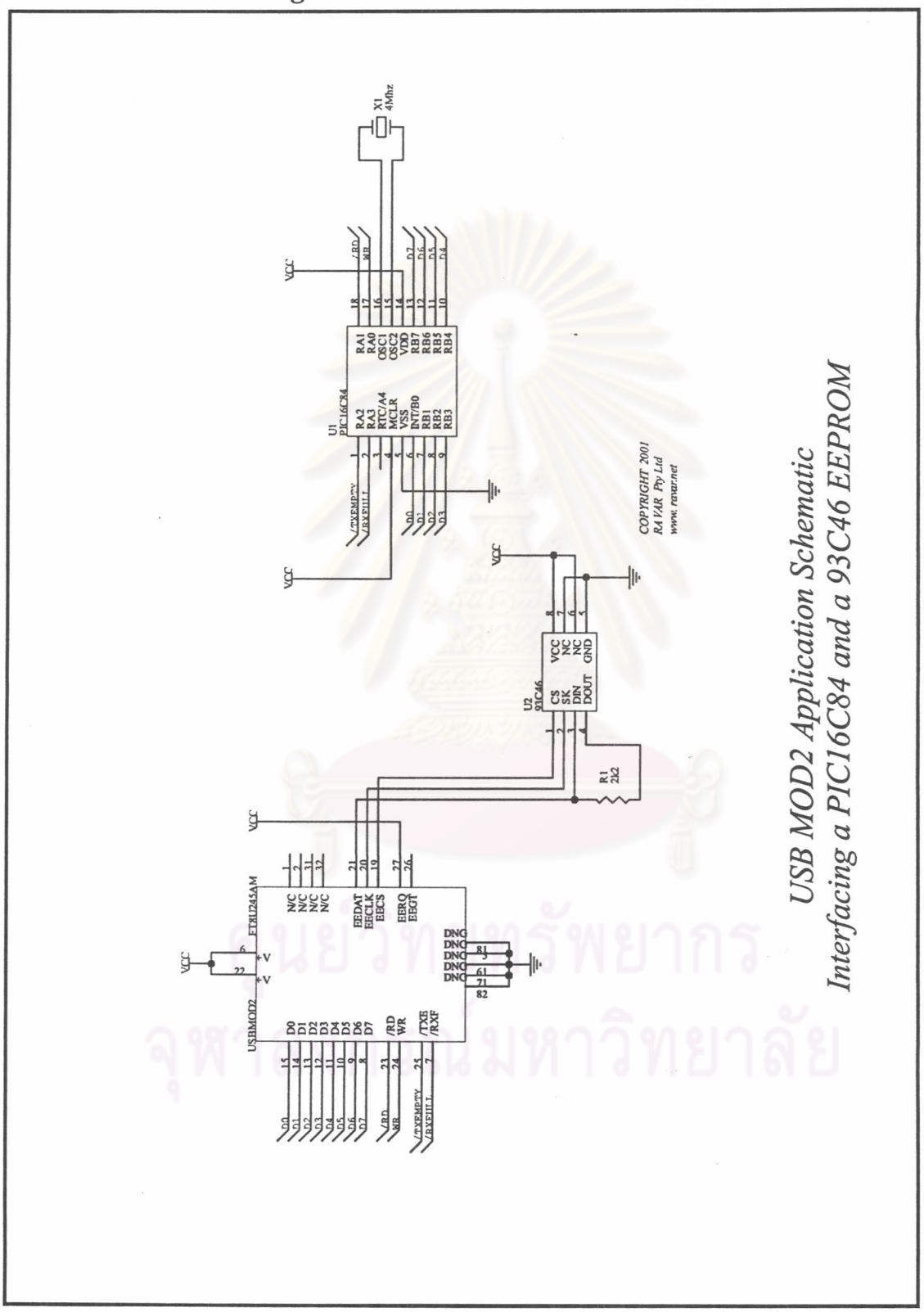
Programming the Direct DLL driver is by call to the DLL Library functions.
Please download the Direct DLL programmers guide from the Ravar website.

Application Notes

On the following pages there are schematic drawings showing a sample applications for the USB MOD2.

The application uses a micro controller, in this case we are using a PIC 16C84 as well as a 93C46 EEPROM chip.

SAMPLE APPLICATION No. 2
Interfacing a PIC16C84 and a 93C46EEPROM



Absolute Maximum Ratings

Storage Temperature	-65°C to + 150°C
Ambient Temperature (Power Applied)	0°C to + 70°C
VCC Supply Voltage	-0.5v to +6.00v
DC Input Voltage - Inputs	-0.5v to VCC + 0.5v
DC Input Voltage - High Impedance Bidirectionals	-0.5v to VCC + 0.5v
DC Output Current – Outputs	24mA
DC Output Current – Low Impedance Bidirectionals	24mA
Power Dissipation	500mW

DC Characteristics (Ambient Temperature = 0°C .. 70°C)

	Description	Min	Max	Units	Condition
VCC	Operating Supply Voltage	4.4	5.25	V	
Icc1	Operating Supply Current		50	mA	Normal Operation
Icc2	Operating Supply Current		250	uA	USB Suspend
Ioh1	Digital IO Pins Source Current	4		mA	Voh = VCC – 0.5V
Iol1	Digital IO Pins Sink Current	8		mA	Vol = +0.5V
Voh1	Input Voltage Threshold (Low)		0.6	V	
Vol1	Input Voltage Threshold (High)	2.7		V	
VDif	USB Differential Input Sensitivity	0.2		V	
VCom	USB Differential Common Mode	0.8	2.5	V	
URxt	USB Single Ended RX Threshold	0.8	2.0	V	
UVh	USB IO Pins Static Output (Low)		0.3	V	Rl = 1.5k to 3.6V
UVI	USB IO Pins Static Output (High)	2.8		V	Rl = 15k to GND

Technical Support and Further Information

For any questions relating to the USBMOD2 please contact us by Email, Fax or Phone.

email: support@ravar.net
Fax: +61 755 914364
Ph: +61 755 325688

Ravar Pty Ltd
5G Jackman Center
Jackman Street, Southport
Queensland 4215
Australia

Ravar Pty Ltd
PO Box 2514
Southport
Queensland 4215
Australia

Product Use Limitations, Warranty and Quality Statement.

The USBMOD2 should not be used in any situation where it's failure or failure of the PC or software controlling it could cause human injury or severe damage to equipment.
This device is not designed for or intended to be used in any life critical application.

The USBMOD2 is warranted to be free from manufacture defects for a period of 12 months from the date purchase.
Subjecting the device to conditions beyond the Absolute Maximum Ratings listed above will invalidate this warranty.
The USBIO24 is a static sensitive device, anti static procedures should be used in the handling of this device.

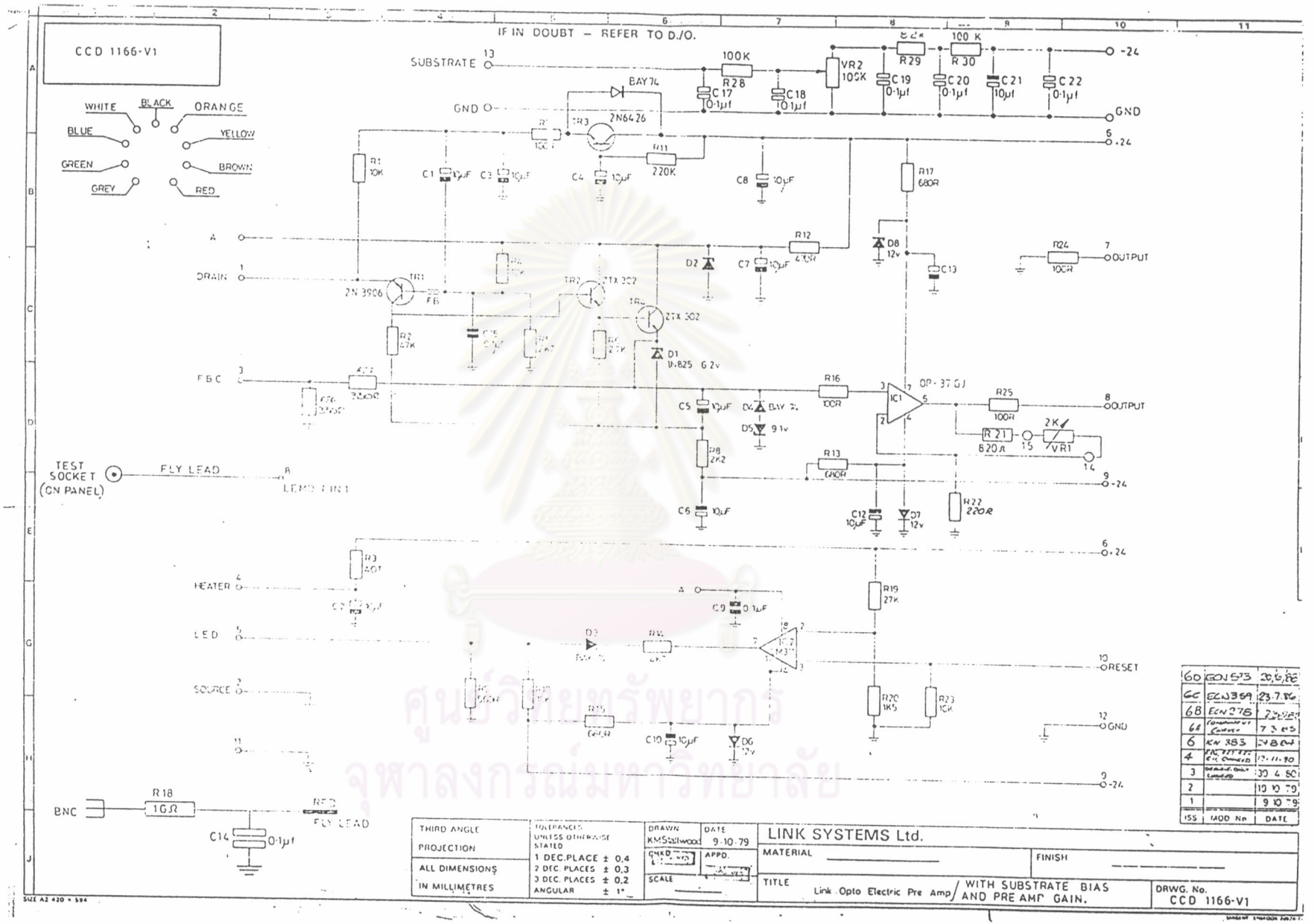
All USBIO24 units are extensively tested at time of manufacture to be free of defects.

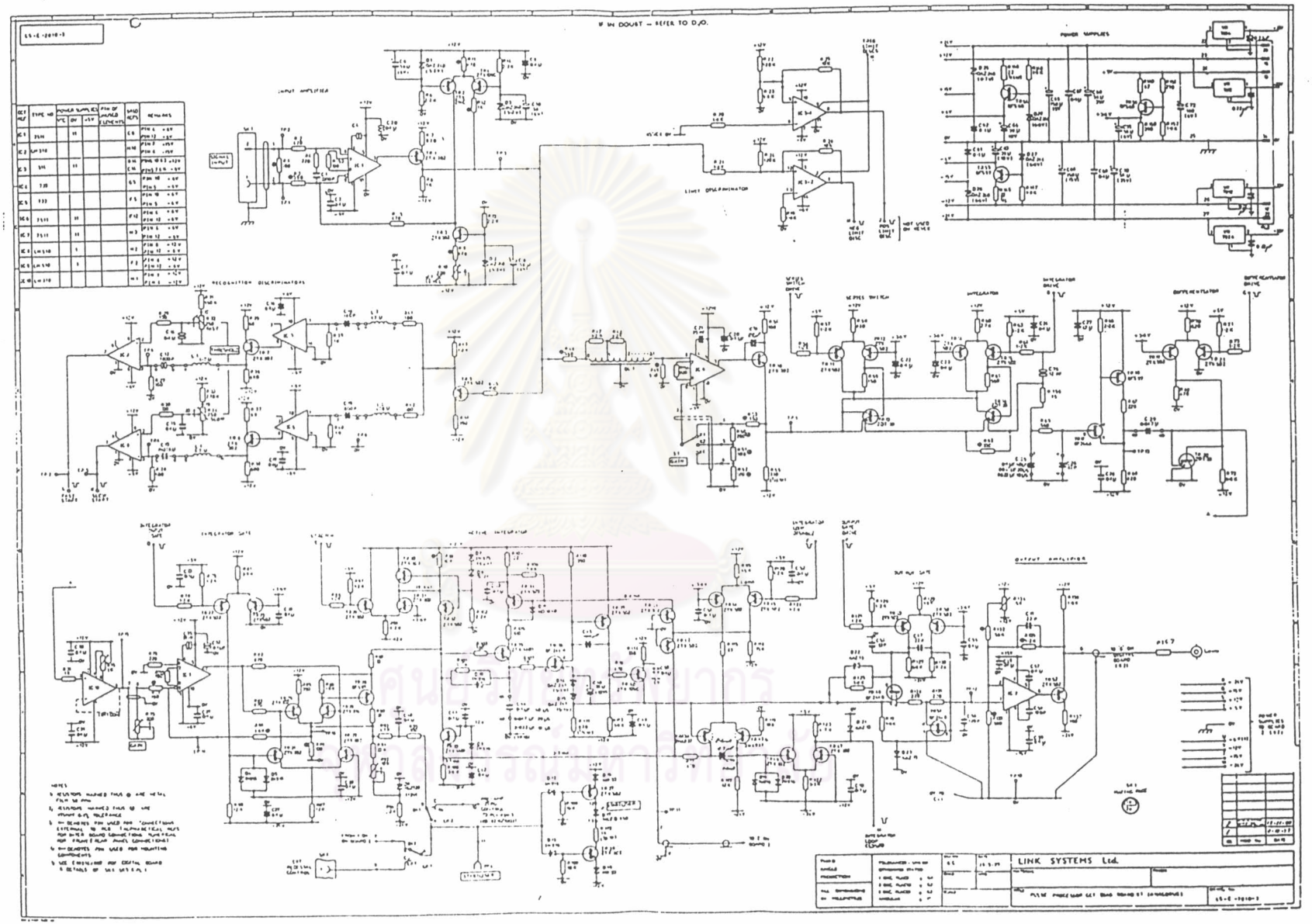
Ravar is committed to providing products of the highest quality. Should you experience any product quality issues with this product please contact our quality assurance manager at the above address.

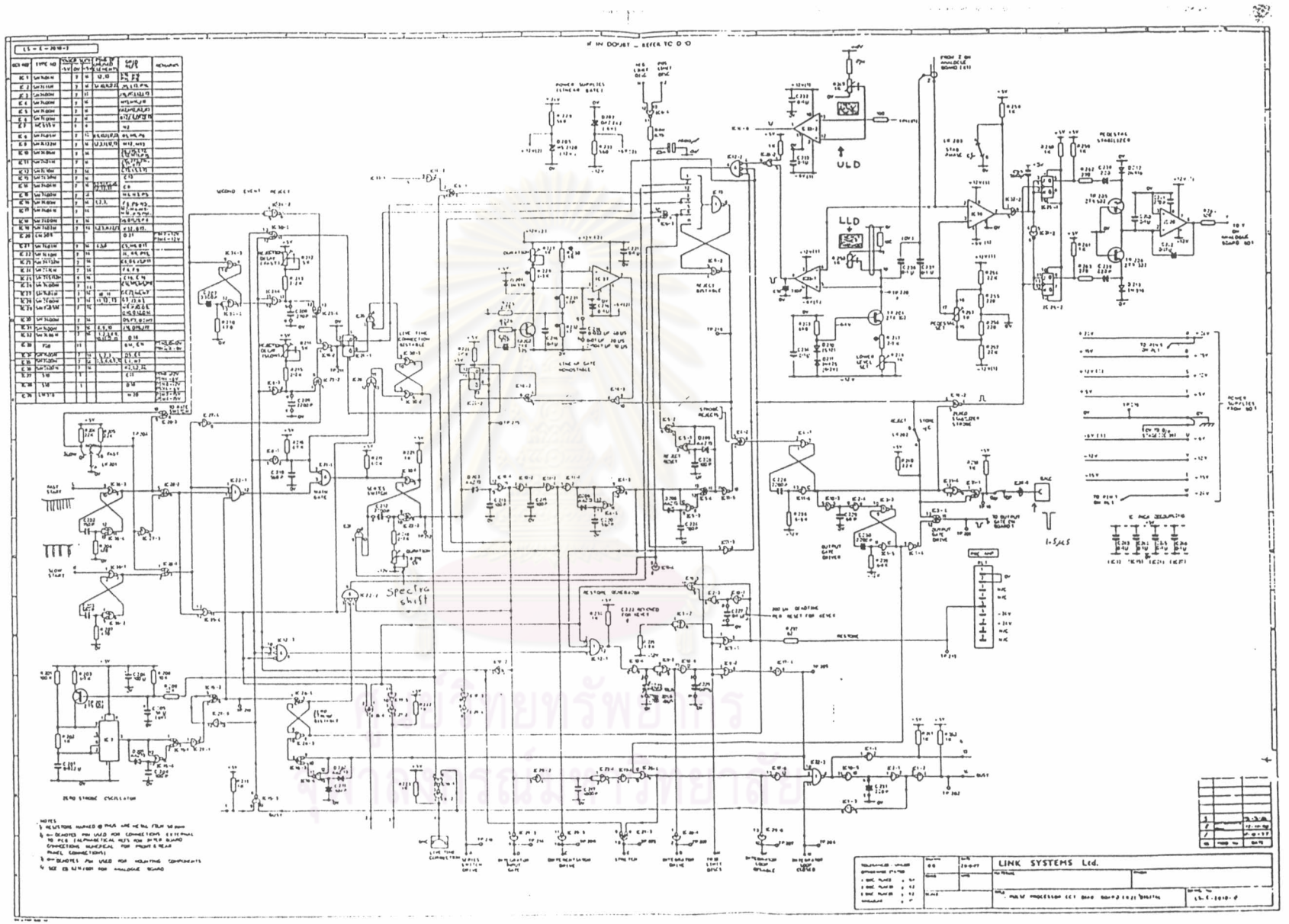
Disclaimer.

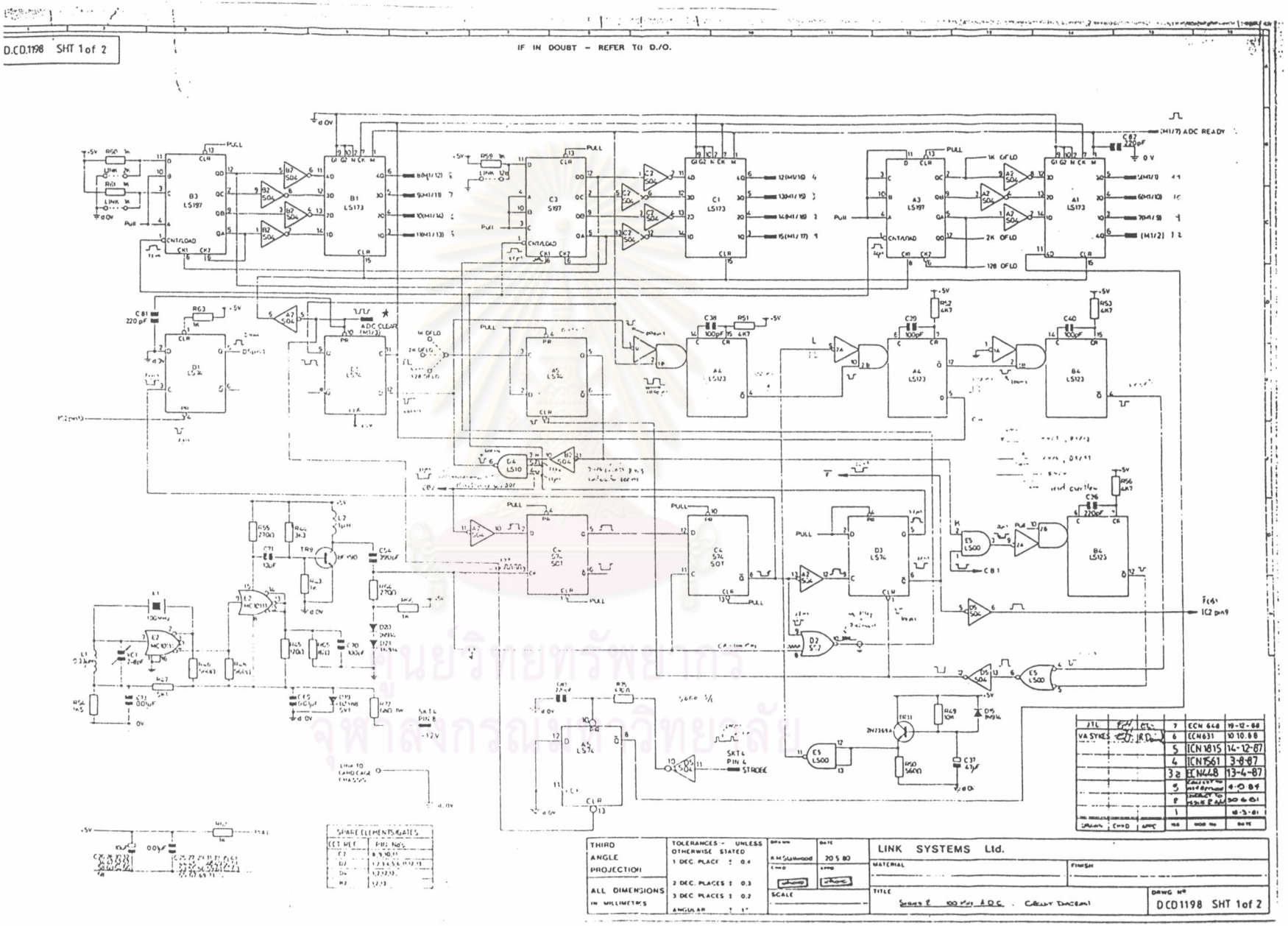
This product and its documentation are provided as-is and no warranty is made or implied as to their suitability for any particular purpose.
Ravar Pty Ltd will not accept any claim for damages arising from the use of this product or documentation.
This document provides information on our products and all efforts are made to ensure the accuracy of the information contained within. The specifications of the product are subject to change and continual improvement.

ภาคผนวก ฉ


CCD 1166-V1
IF IN DOUBT – REFER TO D./O.
WHITE
BLACK
ORANGE
BLUE
YELLOW
GREEN
BROWN
GREY
RED
SUBSTRATE
GND
DRAIN
A
FBC
HEATER
LED
SOURCE
TEST SOCKET (ON PANEL)
FLY LEAD
LEMO PIN 1
BNC
R18 1GΩ
RED FLY LEAD
C14 0·1µF
2N6426
2N3906
2TX302
OP-37 GJ
OUTPUT
ORESET
GND
-24
+24
THIRD ANGLE PROJECTION
ALL DIMENSIONS IN MILLIMETRES
TOLERANCES UNLESS OTHERWISE STATED
1 DEC.PLACE ± 0,4
2 DEC. PLACES ± 0,3
3 DEC. PLACES ± 0,2
ANGULAR ± 1°
DRAWN KMSaltwood
DATE 9-10-79
CHKD
APPD.
SCALE
LINK SYSTEMS Ltd.
MATERIAL
FINISH
TITLE Link Opto Electric Pre Amp/ WITH SUBSTRATE BIAS AND PRE AMP GAIN.
DRWG. No. CCD 1166-V1
SIZE A2 420 x 594
6D ECN 573 20.6.86
GC ECN 369 23.7.84
6B ECN 276 23.2.83
6A 7.3.83
6 ECN 383 24.8.82
4 13.11.80
3 30.4.80
2 19.10.79
1 9.10.79
ISS MOD No DATE

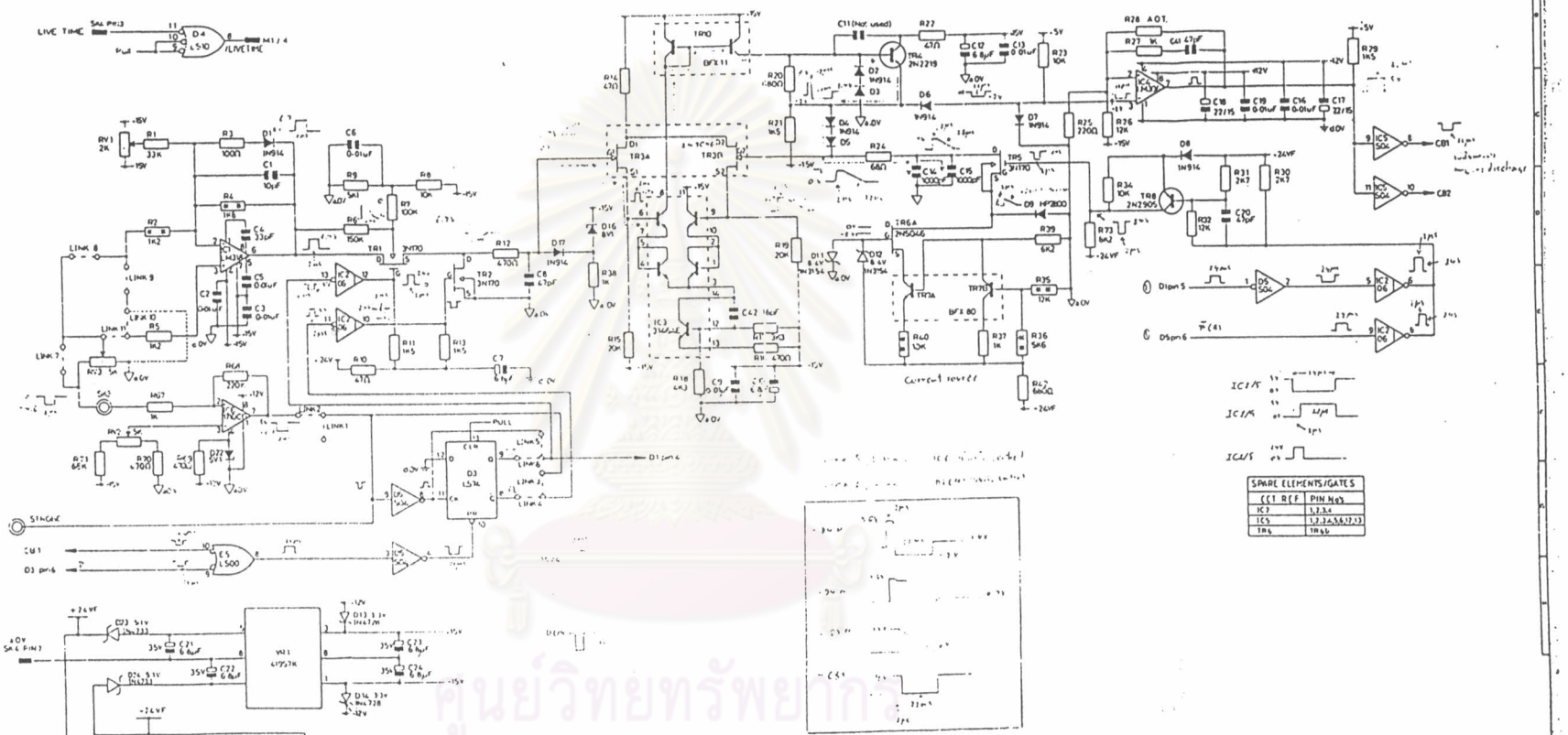

IF IN DOUBT – REFER TO D/O.
INPUT AMPLIFIER
LIMIT DISCRIMINATOR
POWER SUPPLIES
RECOGNITION DISCRIMINATORS
INTEGRATOR
OUTPUT AMPLIFIER
LINK SYSTEMS Ltd.

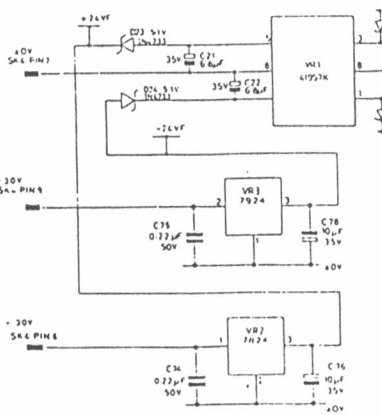

ULD
LLD
LINK SYSTEMS Ltd.


D.CD.1198 SHT 1 of 2
IF IN DOUBT - REFER TO D./O.
(M1/7) ADC READY
ADC CLEAR (M1/3)
1K OF LO
2K OF LO
128 OF LO
SKT 6 PIN 8
SKT 6 PIN 4 STROBE
LINK TO CAMCAGE CHASSIS
SPARE ELEMENTS/GATES
THIRD ANGLE PROJECTION
ALL DIMENSIONS IN MILLIMETRES
TOLERANCES - UNLESS OTHERWISE STATED
1 DEC PLACE ± 0.4
2 DEC PLACES ± 0.3
3 DEC PLACES ± 0.2
ANGULAR ± 1°
R M SUMWOOD
20 5 80
LINK SYSTEMS Ltd.
MATERIAL
FINISH
SCALE
DRWG NO
DCD1198 SHT 1 of 2
ECN 648 19-12-88
ECN 631 10.10.88
ECN 615 14-12-87
ECN 561 3-8-87
ECN 448 13-4-87
4-0 87
30 6 87
18-3-87

CD-1198 SHT 2 of 2

IF IN DOUBT - REFER TO D.J.O.





NOTES 1) RESISTORS SHOWN THUS " ** " TO BE 5ppm/°C, 0.5% TOL. METAL FILM. HOLDSWORTHY OR EQUIVALENT.

					See SHT 1
DRAWN	CHKD.	APPD	ISS	MOD No.	DATE

THIRD ANGLE PROJECTION	TOLERANCES - UNLESS OTHERWISE STATED 1 DEC. PLACE ± 0.4	DRAWN R.M.S. Garwood	DATE 22 5 80	LINK SYSTEMS Ltd.
ALL DIMENSIONS IN MILLIMETRES	2 DEC. PLACES ± 0.3 3 DEC. PLACES ± 0.1 ANGULAR ± 1°	CHKD	APPD	MATERIAL — FINISH —
		SCALE —		TITLE Series 2 100 MHz ADC Colour Diagram DRWG Nº D.CD1198 SHT 2of2

ประวัติผู้เขียนวิทยานิพนธ์

นายวสันต์ อัมพุชินี เกิดเมื่อวันที่ 22 เมษายน 2519 ที่กรุงเทพมหานคร สำเร็จการศึกษาระดับปริญญาบัณฑิตจากภาควิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยเกษตรศาสตร์ เมื่อปีการศึกษา 2540 และในปีการศึกษา 2542 ได้เข้าศึกษาระดับปริญญามหาบัณฑิตที่ภาควิชานิวเคลียร์เทคโนโลยี คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย