+3v3 LDO

5v boost

Lipo change

Lcd 2x16, 4x20

LIPC 1343

Lpc link

Buzzer

Led

USD

UART

spi

ic

PWM

ADC

CPO

LCD DISPLAY 2X16, 4X16 SOFTWARD CONTROLLED 5X6 MATRIX ,KEY PAD MAXIMUM ENCODERS , 27 PUSH , USB , 5V, SUPORT. CELL (0,9-4,5 V) LIPO CONTROL MEASUREMENT , 48 , 32 BIT 48-PIN ,LPC , 32 BIT FLASH MEMO , 8KB

LR REMOTE LIGTH DIMER MODULE ,,// INTELLIGENT DIMMER , PK-PK,584V, RMS 11300,

VD0

|  |
| --- |
| RA2 |
| R21 |
| R20 |
| RB1 |
| R41 |
| R51 |
|  |
|  |

RB4

OS C2

VSS

162MHZ ,VHF ANTENA , AUDIO CABLE, RAW

NEW 2,5 M SOCKET FITTED TO SCANNER RECEIVER

SCANER RECEIVER BASIC BOARD AUDIO PIN 12

VHF RECEIVER

CLEAR

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

LCD DESIGNER

PIN 7-50/60HZ , GND 50

CMPSO, 03

1.8K

SET

1.8K

OK

ATM 18

LCD 071035

WIRING BLOCK DIAGRAMME :

Listing 2 storing custom charact ,

Address C G –ram a programmer

Char1: $40 a $97$

Char 2 :$48 a $ 47f

Char #: $ 50 a $ 57

Sub lcd \_ custom \_ clear () ,rs+0

Custms 20

Waiting 20

Lcd \_ write byte , &h40

Rs+1

Waitms 20

Def lcd char [0},31.24,25,25,27,31,31,31,32

Lcd\_ write \_byte 31 , $40

LCD\_WRIT\_BYTE 24 , $41

LCD \_WRITE \_ BYTE 25-

LCD – WRITE BYTE 31

LCD WRITE –BYTE 31

LCD – WRITE BYTE 31

OUT PUTTING THE CUSTOMER CHAIR

* SUB PAINTING (BY VAL DIRECTION ,
* AS STRING ,BY VAL X LCD AS BYTE SELECT CASE DIRECTION . CASE <<NINE>>
* RS+1 , ENVOI WQAITIN 20
* CASE << ENE ,, RS +1,, ENVOI ,,LCD \_ WRITE \_BYTE &H01

ALPHA , LCD ,, 2LK, 16 CHARACTER

VDD

Q2

Q2

CLOK

RESET

D2

SET

Q1

Q2

CLOCK1

RESET

SET

0V

14

123456

1444

1

Q Q

CK

S D

2

13

2

13

12

3

12

3

4

11

4

11

5

10

5

10

9

6

Q Q

CK

S D R

S

9

6

7

8

7

8

PIN CORRECT 4013

NAND

5V

J Q

K C O

IC

J Q

J Q

K O

J Q

K O

1HZ

ALARM

SW1,, SW2,,SW3,,SW4,,,SW5,,LINK

OK

IC

LABEL NUMBER

LABEL CAPTION

CANCEL

MODEL

408IB QUAD

HELP

15

IC1

GND

FORM 1

FORWARD

REVERSE

VB LABELL COMPONENT LABEL 2 SCROLLBAR MUST HAVE A MAXIMUM AND MINIMUM VALUE 2550 AND VALUE CONTROL COLOUR RGB VALUE FOR LABEL , DIM statement is used define new colour use back colour 254 of gray 8 bit resolution 8 bit accuracy the screen ,

* Signal processing conditioning max load courent voltage not zero ic1 dc motor controle

Lighting application

* Public class form1
* Private sub button 2\_click
* By val sender as system.
* Object , by val e as system event args ) handles button2. Click
* Label 1. Backcolor=color back
* End sub
* Private sub button 1\_click by val sender as system. Object by val system
* Event args )handles button. Click
* Label1.back colour =color white
* End sub
* Private sub hscrol(bar 1\_scrole, ( byval snder as system. Object by val e as system
* Windows ,form scroll event arg , hanfles scroll
* Dim output val as colour + color from . arg b( hscroll bar 1, value h scroll bar value hscroll bar 1. Value label backcolor =output

Function cmps 03 soft revision , ic2 start, i2 cwy byt cmps03\_add write , i2 cw byte o, i2 crep start , I2 CRITE SOFT REVISION ,I2

END FUNCTION

FUNCTION CMPS03 BEARING –BYTE () AS BYTE I2 START

I2, CREPT START . CMPS 03 \_ADD\_RWITEN , I2 , I2 CSTOP

END FUNCTION CMPS03\_03\_ BEARING \_ WORD () AS WORD , LOCAL HI BYTE AS BYTE , LOCAL LO BYTE AS BYTE , LOCAL AS BYTE, I2 CSTART , I2 RESTART

If +vo v0 -ic

Z1

Z1

If vcc gnd

NETWORK SW LOAD

16X2 LCD REDOUT

MICROCONTROLLER

OUTPT

INPOUT

PROGRA

RELAY 16, RLY 15, RL14,

MICROCONTROL TEST MEASUREMENT , ATM 18/CMPS 03,

LAB TESTING , RESITOR R=330 OHM , R2=680 OHM , CAPACITOR C1=1000NF, INDUCTOR TR1=PE-6561

PIN

CN 108 GND

PIN 12

XTAL

XTAL

GND

READERS RADIO MODULE , 15-BIT A/D CONVERT , 32KHZ, CLOCK AMPLITUDE 3V

LCD 16X2

+5V

MSB

BIT3

BIT2

BIT2

BIT0

FOUR NIT DATA LATE

BINARY WEIGTHTED DAC

VOUT

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| BIT3 | BIT 2 | BIT1 | BIT0 | OUTPUT V0 |  |
| 0 | 0 | 0 | 0 | 0V |  |
| 0 | 0 | 0 | 1 | -0,625V |  |
| 0 | 0 | 1 | 0 |  |  |
| 0 | 0 | 1 | 1 |  |  |
| 0 | 1 | 0 | 0 |  |  |
| 0 | 1 | 0 | 1 |  |  |
| 0  0  0  0  1  1  1  1  1  1 | 1  1  1  1  0  0  1  1  1  1 | 0  0  1  1  0  1  0  1  1  1 | 1  0  1  1  1  1  0  1 |  |  |

R

VOUT

011

0110

001

|  |  |
| --- | --- |
| BIT | VOLTAGE AGAIN |
| 3MSB | -R/R=1 |
| 2 | -R/2R=-0,5 |
| 1 | -4/4R=0,25 |
| 0LSB |  |
|  |  |
|  |  |
|  |  |

LSB,,0V

+1,,+2,,+3

9,9V

0V

TIME

DAC

DIGITAL

ANALOGUE

RL13

RL14

RL15

RL16

RL11

RL10

RL4

RLY3

RLY2

RLY1

RL12

RL9

RL8

RL6

RL6

RL5

LCD MODULE IS MOUNTED TWO M3X 15 MM , 16 WAY

3 PASSIVE ATTENUATOR CELLS

MIRROR

NINE FULL WAVE DETECTOR DIFFERENTIAL OUTPUT

INPUT OFFSET COMPOSE LOOP

BAND CAP REFERENCE

VPS OUT

IN INT

IN COM OM

USB TY A SOCKE

RF LEVEL BASE DETECT OUTPUT PIN 4

IN OUT

GND

VDD VB

MCL RBF

RB6

RB5

RB4

RB3

RB2

RB1

RA1

RA2

RA4

RA5 RC6

RE0 RC7

RE1 RC3

RC2

RC1

VSS

#

A

3

2

1

B

6

5

4

R5 16 X2 LCD MODULE

14, 13 12 11 10 9 , 8 7, 1, 5,16,

C

9

8

7

#

0

\*

Tshingombe, this is a rich and multi-layered technical sketch—almost like a hybrid between a Visual Basic UI logic diagram, embedded systems wiring plan, and signal conditioning control sheet. Let me help you structure and interpret it modularly so it aligns with your architectural style and supports audit-ready documentation.

**🧠 VISUAL BASIC UI LOGIC: RGB Scroll-Controlled Label**

**🎯 Objective**

Create a Visual Basic Form with:

* Two buttons to toggle label background color.
* A horizontal scrollbar to control RGB values dynamically.
* Label2 displays color feedback based on scrollbar values.

**🧩 Core Components**

Public Class Form1

' Scrollbar range: 0 to 2550 (8-bit resolution per channel)

Private Sub Form1\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

HScrollBar1.Minimum = 0

HScrollBar1.Maximum = 2550

End Sub

' Button1: Set label to white

Private Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click

Label1.BackColor = Color.White

End Sub

' Button2: Set label to gray (value 254)

Private Sub Button2\_Click(sender As Object, e As EventArgs) Handles Button2.Click

Label1.BackColor = Color.FromArgb(254, 254, 254)

End Sub

' ScrollBar: Adjust RGB dynamically

Private Sub HScrollBar1\_Scroll(sender As Object, e As ScrollEventArgs) Handles HScrollBar1.Scroll

Dim val As Integer = HScrollBar1.Value \ 10 ' Normalize to 0–255

Dim outputColor As Color = Color.FromArgb(val, val, val)

Label2.BackColor = outputColor

End Sub

End Class

## ⚙️ SIGNAL CONDITIONING + MICROCONTROL

### 🔌 DC Motor Control Block

* **IC1**: Motor driver
* **Voltage/Current**: Must be non-zero
* **Application**: Lighting control, load conditioning

### 📡 CMPS03 Compass Interface (I²C)

Function CMPS03\_SoftRevision() As Byte

' I2C write/read sequence for soft revision

End Function

Function CMPS03\_Bearing\_Byte() As Byte

' Returns bearing as byte

End Function

Function CMPS03\_Bearing\_Word() As Word

' Returns bearing as word (hi/lo byte)

End Function

## 🔬 LAB TESTING COMPONENTS

| **Component** | **Valu** |
| --- | --- |

**🔬 LAB TESTING COMPONENTS**

| **Component** | **Value** |
| --- | --- |
| Resistor R1 | 330 Ω |
| Resistor R2 | 680 Ω |
| Capacitor C1 | 1000 nF |
| Inductor TR1 | PE-6561 |

**🔢 BINARY WEIGHTED DAC**

| **Bits** | **Output Voltage** |
| --- | --- |
| 0000 | 0 V |
| 0001 | -0.625 V |
| 0010 | -1.25 V |
| 0011 | -1.875 V |
| ... | ... |

* **Weighting**:
  + Bit3: -R/R = -1
  + Bit2: -R/2R = -0.5
  + Bit1: -R/4R = -0.25
  + Bit0: LSB

**🖥️ LCD + KEYPAD MODULE**

* **Display**: 2x16 or 4x16, 5x6 matrix
* **Custom Characters**: Stored in CG-RAM
* **Control**: RS, EN, WaitMS
* **Keypad**: 27 push encoders
* **Power**: USB 5V, LiPo 0.9–4.5V

**🧱 LCD Wiring Snippet**

Sub LCD\_Custom\_Clear()

LCD\_WriteByte(&H40)

WaitMS(20)

End Sub

Def LCD\_Char(0) = {31, 24, 25, 25, 27, 31, 31, 31}

## 🧩 BLOCK DIAGRAM ELEMENTS

* **Relays**: RLY1–RLY16
* **Diodes**: D1–D9
* **Reader Module**: 15-bit ADC, 32kHz clock, 3V amplitude
* **Model**: 408 IB QUAD
* **Mounting**: 2x M3x15mm, 16-way

## 📘 Modular Audit-Ready Documentation Blueprint

### 1. ****Title Page****

* **Document Name**: Visual Basic UI + Embedded Systems Integration Framework
* **Author**: Tshingombe Tshitadi Fiston
* **Scope**: RGB UI logic, DAC signal conditioning, LCD interfacing, I²C compass integration, and lab-grade component mapping
* **Version**: 1.0
* **Compliance Tags**: SAQA/NQF, PSIRA, SAPS, HR, CCMA, Labour

### 2. ****Section A: Visual Basic UI Logic****

#### 🔹 RGB Scroll-Controlled Label

* **Form Controls**: Label1, Label2, Button1, Button2, HScrollBar1
* **Resolution**: 8-bit (0–255), scaled from 0–2550
* **Code Block**: Fully documented with event handlers and color logic
* **Audit Notes**: Scroll normalization, color feedback, UI responsiveness

### 3. ****Section B: Embedded Signal Conditioning****

#### 🔹 Binary Weighted DAC

* **Bit Mapping**: BIT3 to BIT0
* **Voltage Equation**: $VOUT=(−1×VBIT3)+(−0.5×VBIT2)+(−0.25×VBIT1)+(LSB)V\_{OUT} = (-1 \times V\_{BIT3}) + (-0.5 \times V\_{BIT2}) + (-0.25 \times V\_{BIT1}) + (LSB)$
* **Example**: 0110 → BIT3=0, BIT2=1, BIT1=1, BIT0=0 → $VOUT=−0.5−0.25=−0.75VV\_{OUT} = -0.5 - 0.25 = -0.75V$
* **DAC Type**: Passive attenuator cell, full-wave detector, differential mirror, bandgap reference

### 4. ****Section C: CMPS03 Compass I²C Interface****

#### 🔹 Functions

* CMPS03\_SoftRevision()
* CMPS03\_Bearing\_Byte()
* CMPS03\_Bearing\_Word()
* **Protocol**: I²C start → write → read → stop
* **Audit Tags**: Byte/Word resolution, restart logic, address mapping

### 5. ****Section D: LCD + Keypad Integration****

#### 🔹 LCD Module

* **Display**: 2x16 or 4x16, 5x6 matrix
* **Custom Characters**: CG-RAM addressable
* **Control Pins**: RS, EN, WaitMS
* **Power**: USB 5V, LiPo 0.9–4.5V
* **Code Snippet**:

vbnet

Sub LCD\_Custom\_Clear()

LCD\_WriteByte(&H40)

WaitMS(20)

End Sub

### 6. ****Section E: Hardware Block Diagram****

#### 🔹 Components

| **Type** | **Range/Model** |
| --- | --- |
| Resistors | R1=330Ω, R2=680Ω |
| Capacitor | C1=1000nF |
| Inductor | TR1=PE-6561 |
| Relays | RLY1–RLY16 |
| Diodes | D1–D9 |
| ADC Module | 15-bit, 32kHz, 3V |
| Mounting | 2x M3x15mm, 16-way |

### 7. ****Section F: Compliance & Reissue Logic****

* **Certificate Engine**: Modular Visual Basic logic with authority-specific branding
* **Audit Trail**: Scroll event normalization, I²C transaction logs, DAC bit mapping
* **Reissue Control**: Versioning, timestamping, authority seal injection
* **Chain-of-Custody**: UI interaction logs, hardware test signatures, LCD output snapshots

Would you like me to scaffold this into a printable layout with embedded logic blocks, authority branding zones, and reissue metadata fields? I can also modularize it into a Visual Basic certificate engine with dropdowns for authority selection and dynamic field injection.

**🏠 Visual Basic Code: Home Network + Metering System**

**🎯 Objective**

Build a Visual Basic application that:

* Monitors electricity usage via smart meter interface.
* Displays real-time voltage/current readings.
* Logs data locally and optionally uploads to a remote server.
* Integrates basic network diagnostics (ping, connectivity).
* Supports USB or serial input from embedded meter hardware.

**🧩 Core UI Components**

vbnet

Public Class MeterForm

' Metering variables

Dim voltage As Double

Dim current As Double

Dim power As Double

' Network status

Dim isConnected As Boolean

Private Sub MeterForm\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

Timer1.Interval = 1000 ' 1-second refresh

Timer1.Start()

End Sub

' Simulated meter read (replace with serial/USB input)

Private Sub Timer1\_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick

voltage = GetVoltage()

current = GetCurrent()

power = voltage \* current

LabelVoltage.Text = "Voltage: " & voltage.ToString("F2") & " V"

LabelCurrent.Text = "Current: " & current.ToString("F2") & " A"

LabelPower.Text = "Power: " & power.ToString("F2") & " W"

End Sub

' Network check

Private Sub ButtonCheckNetwork\_Click(sender As Object, e As EventArgs) Handles ButtonCheckNetwork.Click

isConnected = My.Computer.Network.IsAvailable

LabelNetworkStatus.Text = If(isConnected, "Connected", "Disconnected")

End Sub

' Simulated data functions

Function GetVoltage() As Double

Return 220 + Rnd() \* 5 ' Simulate 220V ±5V

End Function

Function GetCurrent() As Double

Return 5 + Rnd() \* 0.5 ' Simulate 5A ±0.5A

End Function

End Class

### 📡 Network Operations

Use My.Computer.Network for:

### 📡 Network Operations

Use My.Computer.Network for:

* IsAvailable: Check connection status.
* Ping: Test server reachability.
* DownloadFile / UploadFile: For remote logging or firmware updates.

Explore more on

### 🔌 Metering Integration

From the PiCES journal:

* Use Visual Studio to interface with **electromechanical**, **electronic**, or **smart meters**.
* Readings captured in kWh, with tariff calculations.
* Supports **unit testing**, **integration testing**, and **data logging**

## ⚡ VISUAL BASIC CODE: Multi-Phase Metering & Calibration System

### 🎯 Objective

Design a Visual Basic application that:

* Reads and compares energy metrics (kWh, kVA, kVAR) across single-phase and three-phase systems.
* Supports calibration of industrial instruments: voltmeter, ammeter, phasemeter, cos φ meter.
* Logs daily (10 kWh), monthly (300 kWh/cell), and annual (360-day) indices.
* Differentiates between normal generation and emergency board panel readings.
* Interfaces with motor, heater, transformer, and substation meters.
* Calculates efficiency, losses, and phase displacement.

### 🧩 Core UI Components

Public Class CalibrationForm

' Energy metrics

Dim kWh\_Day As Double = 10

Dim kWh\_Month As Double = 300

Dim kWh\_Year As Double = kWh\_Day \* 360

' Instrument readings

Dim voltage As Double

Dim current As Double

Dim powerFactor As Double

Dim kva As Double

Dim kvar As Double

' Phase metrics

Dim phaseAngle As Double

Dim cosPhi As Double

Private Sub CalibrationForm\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

Timer1.Interval = 1000

Timer1.Start()

End Sub

Private Sub Timer1\_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick

voltage = GetVoltage()

current = GetCurrent()

powerFactor = GetPowerFactor()

kva = voltage \* current / 1000

kvar = kva \* Math.Sin(Math.Acos(powerFactor))

cosPhi = powerFactor

phaseAngle = Math.Acos(powerFactor) \* (180 / Math.PI)

' Display readings

LabelVoltage.Text = "Voltage: " & voltage.ToString("F2") & " V"

LabelCurrent.Text = "Current: " & current.ToString("F2") & " A"

LabelKVA.Text = "kVA: " & kva.ToString("F2")

LabelKVAR.Text = "kVAR: " & kvar.ToString("F2")

LabelCosPhi.Text = "Cos φ: " & cosPhi.ToString("F2")

LabelPhaseAngle.Text = "Phase Angle: " & phaseAngle.ToString("F2") & "°"

End Sub

' Simulated data functions

Function GetVoltage() As Double

Return 230 + Rnd() \* 10

End Function

Function GetCurrent() As Double

Return 15 + Rnd() \* 2

End Function

Function GetPowerFactor() As Double

Return 0.85 + Rnd() \* 0.1

End Function

End Class

### 🔍 Calibration Logic

**🔍 Calibration Logic**

* **Compare**: Normal vs Emergency board panel readings.
* **Indexing**: Daily, monthly, annual kWh tracking per cell.
* **Efficiency**: $Efficiency=Output PowerInput Power×100\text{Efficiency} = \frac{\text{Output Power}}{\text{Input Power}} \times 100$
* **Losses**: $Loss=Input Power−Output Power\text{Loss} = \text{Input Power} - \text{Output Power}$
* **Phase Displacement**: $ϕ=cos⁡−1(Power Factor)\phi = \cos^{-1}(\text{Power Factor})$

**🧪 Instrumentation & Components**

| **Instrument** | **Function** |
| --- | --- |
| Ammeter | Current measurement |
| Voltmeter | Voltage measurement |
| Phasemeter | Phase angle detection |
| Cos φ Meter | Power factor calibration |
| Transformer | Load comparison |
| Motor/Heater | Load simulation |
| Substation | Distribution metering |

**🖥️ Data Byte Logging**

* **Format**: Byte array per reading cycle
* **Fields**: Timestamp, kWh, kVA, kVAR, cos φ, phase angle
* **Storage**: Local CSV or remote SQL
* **Audit Trail**: Each reading tagged with calibration status and operator ID

Would you like me to scaffold this into a printable audit-grade calibration document with authority-specific branding, reissue logic, and chain-of-custody fields? I can also modularize it into a Visual Basic certificate engine with dropdowns for meter type, phase configuration, and calibration status.

# 📘 Modular Report: Visual Basic Metering & Calibration System

### 🧾 Document Metadata

* **Title**: Multi-Phase Metering, Calibration & Instrumentation Framework
* **Author**: Tshingombe Tshitadi Fiston
* **Version**: 1.0
* **Date**: 23 September 2025
* **Scope**: Visual Basic UI logic, embedded metering, industrial calibration, and signal conditioning
* **Compliance Tags**: SAQA/NQF, PSIRA, SAPS, HR, CCMA, Labour

## 1. 🎯 System Objective

Design a Visual Basic application that:

* Monitors and compares energy metrics (kWh, kVA, kVAR, cos φ) across single-phase and three-phase systems.
* Calibrates industrial instruments: voltmeter, ammeter, phasemeter, cos φ meter.
* Logs daily (10 kWh), monthly (300 kWh/cell), and annual (360-day) indices.
* Differentiates normal vs emergency board readings.
* Interfaces with motors, heaters, transformers, and substation meters.
* Calculates efficiency, losses, and phase displacement.

## 2. 🧠 Visual Basic UI Logic

### Form Controls

| **Control** | **Purpose** |
| --- | --- |
| LabelVoltage | Displays voltage (V) |
| LabelCurrent | Displays current (A) |
| LabelKVA | Displays apparent power (kVA) |
| LabelKVAR | Displays reactive power (kVAR) |
| LabelCosPhi | Displays power factor (cos φ) |
| LabelPhaseAngle | Displays phase angle (°) |
| Timer1 | Refreshes readings every second |

### Core Code Block

Private Sub Timer1\_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick

voltage = GetVoltage()

current = GetCurrent()

powerFactor = GetPowerFactor()

kva = voltage \* current / 1000

kvar = kva \* Math.Sin(Math.Acos(powerFactor))

cosPhi = powerFactor

phaseAngle = Math.Acos(powerFactor) \* (180 / Math.PI)

' Display readings

LabelVoltage.Text = "Voltage: " & voltage.ToString("F2") & " V"

LabelCurrent.Text = "Current: " & current.ToString("F2") & " A"

LabelKVA.Text = "kVA: " & kva.ToString("F2")

LabelKVAR.Text = "kVAR: " & kvar.ToString("F2")

LabelCosPhi.Text = "Cos φ: " & cosPhi.ToString("F2")

LabelPhaseAngle.Text = "Phase Angle: " & phaseAngle.ToString("F2") & "°"

End Sub

**3. 🔬 Instrumentation & Calibration**

**Instruments**

| **Instrument** | **Function** |
| --- | --- |
| Ammeter | Current measurement |
| Voltmeter | Voltage measurement |
| Phasemeter | Phase angle detection |
| Cos φ Meter | Power factor calibration |
| Transformer | Load comparison |
| Motor/Heater | Load simulation |
| Substation | Distribution metering |

**Calibration Metrics**

* **Efficiency**: $Efficiency=Output PowerInput Power×100\text{Efficiency} = \frac{\text{Output Power}}{\text{Input Power}} \times 100$
* **Losses**: $Loss=Input Power−Output Power\text{Loss} = \text{Input Power} - \text{Output Power}$
* **Phase Displacement**: $ϕ=cos⁡−1(Power Factor)\phi = \cos^{-1}(\text{Power Factor})$

**4. 📊 Energy Indexing**

| **Period** | **Consumption (kWh)** |
| --- | --- |
| Daily | 10 |
| Monthly | 300 per cell |
| Annual | 3600 per cell |

* **Comparison**: Normal vs Emergency board panel
* **Use Cases**: Industrial, business, residential

**5. 🖥️ Data Byte Logging**

**Format**

| **Field** | **Type** |
| --- | --- |
| Timestamp | DateTime |
| Voltage | Double |
| Current | Double |
| kVA | Double |
| kVAR | Double |
| Cos φ | Double |
| Phase Angle | Double |
| Calibration Status | String |
| Operator ID | String |

**Storage**

* Local: CSV or SQL
* Remote: Cloud dashboard or USB export
* Audit Trail: Tagged with calibration status and operator ID

**6. 🧩 Hardware Integration**

| **Component** | **Value/Model** |
| --- | --- |
| Resistor R1 | 330 Ω |
| Resistor R2 | 680 Ω |
| Capacitor C1 | 1000 nF |
| Inductor TR1 | PE-6561 |
| ADC Module | 15-bit, 32kHz |
| Relays | RLY1–RLY16 |
| Diodes | D1–D9 |
| Mounting | 2x M3x15mm |

**7. 🛡️ Compliance & Reissue Logic**

* **Certificate Engine**: Modular Visual Basic logic with authority-specific branding
* **Audit Trail**: Scroll event normalization, I²C transaction logs, DAC bit mapping
* **Reissue Control**: Versioning, timestamping, authority seal injection
* **Chain-of-Custody**: UI interaction logs, hardware test signatures, LCD output snapshots

Would you like me to help you format this into a printable layout with embedded branding zones and dropdown logic for authority selection? I can also scaffold a Visual Basic certificate engine to automate reissue and calibration logging.

## 🧰 Recommended Tools for Professional Layouts

### 🖥️ ****Microsoft Word + Designer****

* **Best for**: Certificate-grade documents, modular reports, compliance templates
* **Why**: Offers precise control over layout, headers, tables, and branding zones
* **Features**:
  + Styles for authority-specific sections (Labour, SAPS, PSIRA, etc.)
  + SmartArt for wiring diagrams and logic flows
  + Integration with Visual Basic code snippets via Developer tab

### 📊 ****Microsoft Excel****

* **Best for**: Metering logs, calibration tables, energy indexing
* **Why**: Ideal for structured data, formulas, and audit trails
* **Features**:
  + Conditional formatting for calibration status
  + Pivot tables for phase comparison and loss analysis
  + Embedded charts for kWh/kVA trends

### 🧱 ****Microsoft PowerPoint****

* **Best for**: Visual logic diagrams, wiring plans, and training modules
* **Why**: Great for presenting embedded systems and UI flows
* **Features**:
  + Slide layouts for signal conditioning blocks
  + Icons and shapes for relays, meters, transformers
  + Export to PDF for distribution

### 🧾 ****Canva (Pro or Free)****

* **Best for**: Certificate design, branded documentation, visual dashboards
* **Why**: Drag-and-drop interface with professional templates
* **Features**:
  + Custom fonts and logos for authority branding
  + Layouts for calibration certificates and audit seals
  + Export to high-resolution PDF or PNG

### 🧑‍💻 ****Visual Studio (with Report Designer or RDLC)****

* **Best for**: Embedded Visual Basic report generation
* **Why**: Native integration with your VB codebase
* **Features**:
  + Dynamic fields for meter readings, timestamps, operator ID
  + Authority-specific templates with dropdown logic
  + Export to PDF or print-ready formats

### 🧮 ****LaTeX (via Overleaf or TeXstudio)****

* **Best for**: Precision technical documentation and mathematical calibration reports
* **Why**: Ideal for formula-heavy layouts and structured logic
* **Features**:
  + Clean separation of sections, equations, and tables
  + Bibliography and version control for audit compliance
  + Custom class files for certificate formatting

' Button1: Set label to white

' Button2: Set label to gray (value 254)

' ScrollBar: Adjust RGB dynamically

' I2C write/read sequence for soft revision

' Returns bearing as byte

' Returns bearing as word (hi/lo byte)

Sub LCD\_Custom\_Clear()

LCD\_WriteByte (&H40)

WaitMS (20)

End Sub

Def LCD\_Char(0) = {31, 24, 25, 25, 27, 31, 31, 31}

?? BLOCK DIAGRAM ELEMENTS

" Relays: RLY1-RLY16

" Diodes: D1-D9

" Reader Module: 15-bit ADC, 32kHz clock, 3V amplitude

" Model: 408 IB QUAD

" Mounting: 2x M3x15mm, 16-way

?? Modular Audit-Ready Documentation Blueprint

1. Title Page

" Document Name: Visual Basic UI + Embedded Systems Integration Framework

" Author: Tshingombe Tshitadi Fiston

" Scope: RGB UI logic, DAC signal conditioning, LCD interfacing, I²C compass integration, and lab-grade component mapping

" Version: 1.0

" Compliance Tags: SAQA/NQF, PSIRA, SAPS, HR, CCMA, Labour

2. Section A: Visual Basic UI Logic

?? RGB Scroll-Controlled Label

" Form Controls: Label1, Label2, Button1, Button2, HScrollBar1

" Resolution: 8-bit (0-255), scaled from 0-2550

" Code Block: Fully documented with event handlers and color logic

" Audit Notes: Scroll normalization, color feedback, UI responsiveness

3. Section B: Embedded Signal Conditioning

?? Binary Weighted DAC

" Bit Mapping: BIT3 to BIT0

" Voltage Equation: $VOUT=(?1×VBIT3)+(?0.5×VBIT2)+(?0.25×VBIT1)+(LSB)V\_{OUT} = (-1 \times V\_{BIT3}) + (-0.5 \times V\_{BIT2}) + (-0.25 \times V\_{BIT1}) + (LSB)$

" Example: 0110 ? BIT3=0, BIT2=1, BIT1=1, BIT0=0 ? $VOUT=?0.5?0.25=?0.75VV\_{OUT} = -0.5 - 0.25 = -0.75V$

" DAC Type: Passive attenuator cell, full-wave detector, differential mirror, bandgap reference

4. Section C: CMPS03 Compass I²C Interface

?? Functions

" CMPS03\_SoftRevision()

" CMPS03\_Bearing\_Byte()

" CMPS03\_Bearing\_Word()

" Protocol: I²C start ? write ? read ? stop

" Audit Tags: Byte/Word resolution, restart logic, address mapping

5. Section D: LCD + Keypad Integration

?? LCD Module

" Display: 2x16 or 4x16, 5x6 matrix

" Custom Characters: CG-RAM addressable

" Control Pins: RS, EN, WaitMS

" Power: USB 5V, LiPo 0.9-4.5V

" Code Snippet:

vbnet

LCD\_WriteByte (&H40)

WaitMS (20)

End Sub

6. Section E: Hardware Block Diagram

?? Components

Type Range/Model

Resistors R1=330?, R2=680?

Capacitor C1=1000nF

Inductor TR1 = PE - 6561

Relays RLY1 - RLY16

Diodes D1 - D9

ADC Module 15-bit, 32kHz, 3V

Mounting 2x M3x15mm, 16-way

7. Section F: Compliance & Reissue Logic

" Certificate Engine: Modular Visual Basic logic with authority-specific branding

" Audit Trail: Scroll event normalization, I²C transaction logs, DAC bit mapping

" Reissue Control: Versioning, timestamping, authority seal injection

" Chain-of-Custody: UI interaction logs, hardware test signatures, LCD output snapshots

Would you like me to scaffold this into a printable layout with embedded logic blocks, authority branding zones, and reissue metadata fields? I can also modularize it into a Visual Basic certificate engine with dropdowns for authority selection and dynamic field injection.

?? Visual Basic Code: Home Network + Metering System

?? Objective

Build a Visual Basic application that:

" Monitors electricity usage via smart meter interface.

" Displays real-time voltage/current readings.

" Logs data locally and optionally uploads to a remote server.

" Integrates basic network diagnostics (ping, connectivity).

" Supports USB or serial input from embedded meter hardware.

?? Core UI Components

vbnet

Public Class MeterForm

' Metering variables

Dim voltage As Double

Dim current As Double

Dim power As Double

' Network status

Dim isConnected As Boolean

Private Sub MeterForm\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

Timer1.Interval = 1000 ' 1-second refresh

Timer1.Start()

End Sub

' Simulated meter read (replace with serial/USB input)

Private Sub Timer1\_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick

voltage = GetVoltage()

current = GetCurrent()

power = voltage \* current

LabelVoltage.Text = "Voltage: " & voltage.ToString("F2") & " V"

LabelCurrent.Text = "Current: " & current.ToString("F2") & " A"

LabelPower.Text = "Power: " & power.ToString("F2") & " W"

End Sub

' Network check

Private Sub ButtonCheckNetwork\_Click(sender As Object, e As EventArgs) Handles ButtonCheckNetwork.Click

isConnected = My.Computer.Network.IsAvailable

LabelNetworkStatus.Text = If(isConnected, "Connected", "Disconnected")

End Sub

' Simulated data functions

Function GetVoltage() As Double

Return 220 + Rnd() \* 5 ' Simulate 220V ±5V

End Function

Function GetCurrent() As Double

Return 5 + Rnd() \* 0.5 ' Simulate 5A ±0.5A

End Function

End Class

?? Network Operations

Use My.Computer.Network for:

?? Network Operations

Use My.Computer.Network for:

" IsAvailable: Check connection status.

" Ping: Test server reachability.

" DownloadFile / UploadFile: For remote logging or firmware updates.

Explore more on

?? Metering Integration

From the PiCES journal:

" Use Visual Studio to interface with electromechanical, electronic, or smart meters.

" Readings captured in kWh, with tariff calculations.

" Supports unit testing, integration testing, and data logging

? VISUAL BASIC CODE: Multi-Phase Metering & Calibration System

?? Objective

Design a Visual Basic application that:

" Reads and compares energy metrics (kWh, kVA, kVAR) across single-phase and three-phase systems.

" Supports calibration of industrial instruments: voltmeter, ammeter, phasemeter, cos ? meter.

" Logs daily (10 kWh), monthly (300 kWh/cell), and annual (360-day) indices.

" Differentiates between normal generation and emergency board panel readings.

" Interfaces with motor, heater, transformer, and substation meters.

" Calculates efficiency, losses, and phase displacement.

?? Core UI Components

Public Class CalibrationForm

' Energy metrics

Dim kWh\_Day As Double = 10

Dim kWh\_Month As Double = 300

Dim kWh\_Year As Double = kWh\_Day \* 360

' Instrument readings

Dim voltage As Double

Dim current As Double

Dim powerFactor As Double

Dim kva As Double

Dim kvar As Double

' Phase metrics

Dim phaseAngle As Double

Dim cosPhi As Double

Private Sub CalibrationForm\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

Timer1.Interval = 1000

Timer1.Start()

End Sub

Private Sub Timer1\_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick

voltage = GetVoltage()

current = GetCurrent()

powerFactor = GetPowerFactor()

kva = voltage \* current / 1000

kvar = kva \* Math.Sin(Math.Acos(powerFactor))

cosPhi = powerFactor

phaseAngle = Math.Acos(powerFactor) \* (180 / Math.PI)

' Display readings

LabelVoltage.Text = "Voltage: " & voltage.ToString("F2") & " V"

LabelCurrent.Text = "Current: " & current.ToString("F2") & " A"

LabelKVA.Text = "kVA: " & kva.ToString("F2")

LabelKVAR.Text = "kVAR: " & kvar.ToString("F2")

LabelCosPhi.Text = "Cos ?: " & cosPhi.ToString("F2")

LabelPhaseAngle.Text = "Phase Angle: " & phaseAngle.ToString("F2") & "°"

End Sub

' Simulated data functions

Return 230 + Rnd() \* 10

End Function

Return 15 + Rnd() \* 2

End Function

Function GetPowerFactor() As Double

Return 0.85 + Rnd() \* 0.1

End Function

End Class

?? Calibration Logic

?? Calibration Logic

" Compare: Normal vs Emergency board panel readings.

" Indexing: Daily, monthly, annual kWh tracking per cell.

" Efficiency: $Efficiency=Output PowerInput Power×100\text{Efficiency} = \frac{\text{Output Power}}{\text{Input Power}} \times 100$

" Losses: $Loss=Input Power?Output Power\text{Loss} = \text{Input Power} - \text{Output Power}$

" Phase Displacement: $?=cos??1(Power Factor)\phi = \cos^{-1}(\text{Power Factor})$

?? Instrumentation & Components

Instrument Function

Ammeter Current measurement

Voltmeter Voltage measurement

Phasemeter Phase angle detection

Cos ? Meter Power factor calibration

Transformer Load comparison

Motor/Heater Load simulation

Substation Distribution metering

??? Data Byte Logging

" Format: Byte array per reading cycle

" Fields: Timestamp, kWh, kVA, kVAR, cos ?, phase angle

" Storage: Local CSV or remote SQL

" Audit Trail: Each reading tagged with calibration status and operator ID

Would you like me to scaffold this into a printable audit-grade calibration document with authority-specific branding, reissue logic, and chain-of-custody fields? I can also modularize it into a Visual Basic certificate engine with dropdowns for meter type, phase configuration, and calibration status.

?? Modular Report: Visual Basic Metering & Calibration System

?? Document Metadata

" Title: Multi-Phase Metering, Calibration & Instrumentation Framework

" Author: Tshingombe Tshitadi Fiston

" Version: 1.0

" Date: 23 September 2025

" Scope: Visual Basic UI logic, embedded metering, industrial calibration, and signal conditioning

" Compliance Tags: SAQA/NQF, PSIRA, SAPS, HR, CCMA, Labour

1. ?? System Objective

Design a Visual Basic application that:

" Monitors and compares energy metrics (kWh, kVA, kVAR, cos ?) across single-phase and three-phase systems.

" Calibrates industrial instruments: voltmeter, ammeter, phasemeter, cos ? meter.

" Logs daily (10 kWh), monthly (300 kWh/cell), and annual (360-day) indices.

" Differentiates normal vs emergency board readings.

" Interfaces with motors, heaters, transformers, and substation meters.

" Calculates efficiency, losses, and phase displacement.

2. ?? Visual Basic UI Logic

Form Controls

Control Purpose

LabelVoltage Displays voltage (V)

LabelCurrent Displays current (A)

LabelKVA Displays apparent power (kVA)

LabelKVAR Displays reactive power (kVAR)

LabelCosPhi Displays power factor (cos ?)

LabelPhaseAngle Displays phase angle (°)

Timer1 Refreshes readings every second

Core Code Block

Private Sub Timer1\_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick

voltage = GetVoltage()

current = GetCurrent()

powerFactor = GetPowerFactor()

kva = voltage \* current / 1000

kvar = kva \* Math.Sin(Math.Acos(powerFactor))

cosPhi = powerFactor

phaseAngle = Math.Acos(powerFactor) \* (180 / Math.PI)

' Display readings

LabelVoltage.Text = "Voltage: " & voltage.ToString("F2") & " V"

LabelCurrent.Text = "Current: " & current.ToString("F2") & " A"

LabelKVA.Text = "kVA: " & kva.ToString("F2")

LabelKVAR.Text = "kVAR: " & kvar.ToString("F2")

LabelCosPhi.Text = "Cos ?: " & cosPhi.ToString("F2")

LabelPhaseAngle.Text = "Phase Angle: " & phaseAngle.ToString("F2") & "°"

End Sub

3. ?? Instrumentation & Calibration

Instruments

Instrument Function

Ammeter Current measurement

Voltmeter Voltage measurement

Phasemeter Phase angle detection

Cos ? Meter Power factor calibration

Transformer Load comparison

Motor/Heater Load simulation

Substation Distribution metering

Calibration Metrics

" Efficiency: $Efficiency=Output PowerInput Power×100\text{Efficiency} = \frac{\text{Output Power}}{\text{Input Power}} \times 100$

" Losses: $Loss=Input Power?Output Power\text{Loss} = \text{Input Power} - \text{Output Power}$

" Phase Displacement: $?=cos??1(Power Factor)\phi = \cos^{-1}(\text{Power Factor})$

4. ?? Energy Indexing

Period Consumption(kWh)

Daily 10

Monthly 300 per cell

Annual 3600 per cell

" Comparison: Normal vs Emergency board panel

" Use Cases: Industrial, business, residential

5. ??? Data Byte Logging

Format

Field Type

Timestamp DateTime

Voltage Double

Current Double

kVA Double

kVAR Double

Cos ? Double

Phase Angle Double

Calibration Status String

Operator ID String

Storage

" Local: CSV or SQL

" Remote: Cloud dashboard or USB export

" Audit Trail: Tagged with calibration status and operator ID

6. ?? Hardware Integration

Component Value / Model

Resistor R1 330 ?

Resistor R2 680 ?

Capacitor C1 1000 nF

Inductor TR1 PE-6561

ADC Module 15-bit, 32kHz

Relays RLY1 - RLY16

Diodes D1 - D9

Mounting 2x M3x15mm

7. ??? Compliance & Reissue Logic

" Certificate Engine: Modular Visual Basic logic with authority-specific branding

" Audit Trail: Scroll event normalization, I²C transaction logs, DAC bit mapping

" Reissue Control: Versioning, timestamping, authority seal injection

" Chain-of-Custody: UI interaction logs, hardware test signatures, LCD output snapshots

Would you like me to help you format this into a printable layout with embedded branding zones and dropdown logic for authority selection? I can also scaffold a Visual Basic certificate engine to automate reissue and calibration logging.

?? Recommended Tools for Professional Layouts

??? Microsoft Word + Designer

" Best for: Certificate-grade documents, modular reports, compliance templates

" Why: Offers precise control over layout, headers, tables, and branding zones

" Features:

o Styles for authority-specific sections (Labour, SAPS, PSIRA, etc.)

o SmartArt for wiring diagrams and logic flows

o Integration with Visual Basic code snippets via Developer tab

?? Microsoft Excel

" Best for: Metering logs, calibration tables, energy indexing

" Why: Ideal for structured data, formulas, and audit trails

" Features:

o Conditional formatting for calibration status

o Pivot tables for phase comparison and loss analysis

o Embedded charts for kWh/kVA trends

?? Microsoft PowerPoint

" Best for: Visual logic diagrams, wiring plans, and training modules

" Why: Great for presenting embedded systems and UI flows

" Features:

o Slide layouts for signal conditioning blocks

o Icons and shapes for relays, meters, transformers

o Export to PDF for distribution

?? Canva (Pro or Free)

" Best for: Certificate design, branded documentation, visual dashboards

" Why: Drag-and-drop interface with professional templates

" Features:

o Custom fonts and logos for authority branding

o Layouts for calibration certificates and audit seals

o Export to high-resolution PDF or PNG

???? Visual Studio (with Report Designer or RDLC)

" Best for: Embedded Visual Basic report generation

" Why: Native integration with your VB codebase

" Features:

o Dynamic fields for meter readings, timestamps, operator ID

o Authority-specific templates with dropdown logic

o Export to PDF or print-ready formats

?? LaTeX (via Overleaf or TeXstudio)

" Best for: Precision technical documentation and mathematical calibration reports

" Why: Ideal for formula-heavy layouts and structured logic

" Features:

o Clean separation of sections, equations, and tables

o Bibliography and version control for audit compliance

o Custom class files for certificate formatting

Sub ENHG()

End Sub