```
Private Sub CommandButton1_Click()

End Sub

Private Sub CommandButton2_Click()

End Sub

Private Sub CommandButton3_Click()

End Sub

Private Sub Frame1_Click()

End Sub

Private Sub Label1_Click()

End Sub

Private Sub Label3_Click()

End Sub

Private Sub Label4_Click()

End Sub

Private Sub ScrollBar1_Change()

End Sub

Private Sub ScrollBar2_Change()

End Sub

Private Sub ScrollBar4_Change()

End Sub

Private Sub TextBox1_Change()

End Sub

Private Sub TextBox2_Change()

End Sub

Private Sub TextBox4_Change()

End Sub

Private Sub TextBox5_Change()

End Sub

Private Sub UserForm_Activate()

End Sub

Private Sub UserForm_AddControl(ByVal Control As MSForms.Control)

End Sub

Private Sub UserForm_BeforeDragOver(ByVal Cancel As MSForms.ReturnBoolean, ByVal Control As MSForms.Co
ntrol, ByVal Data As MSForms.DataObject, ByVal X As Single, ByVal Y As Single, ByVal State As MSForms.
fmDragState, ByVal Effect As MSForms.ReturnEffect, ByVal Shift As Integer)

End Sub

Private Sub UserForm_BeforeDropOrPaste(ByVal Cancel As MSForms.ReturnBoolean, ByVal Control As MSForms
.Control, ByVal Action As MSForms.fmAction, ByVal Data As MSForms.DataObject, ByVal X As Single, ByVal
```

```
 Y As Single, ByVal Effect As MSForms.ReturnEffect, ByVal Shift As Integer)

End Sub

Private Sub UserForm_Click()

End Sub

Private Sub UserForm_Deactivate()

End Sub

Private Sub UserForm_Error(ByVal Number As Integer, ByVal Description As MSForms.ReturnString, ByVal S
Code As Long, ByVal Source As String, ByVal HelpFile As String, ByVal HelpContext As Long, ByVal Cance
lDisplay As MSForms.ReturnBoolean)

End Sub

Private Sub UserForm_Initialize()

End Sub

Private Sub UserForm_KeyDown(ByVal KeyCode As MSForms.ReturnInteger, ByVal Shift As Integer)

End Sub

Private Sub UserForm_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)

End Sub

Private Sub UserForm_Layout()

End Sub

Private Sub UserForm_MouseDown(ByVal Button As Integer, ByVal Shift As Integer, ByVal X As Single, ByV
al Y As Single)

End Sub

Private Sub UserForm_MouseUp(ByVal Button As Integer, ByVal Shift As Integer, ByVal X As Single, ByVal
 Y As Single)

End Sub

Private Sub UserForm_QueryClose(Cancel As Integer, CloseMode As Integer)

End Sub

Private Sub UserForm_Resize()

End Sub

Private Sub UserForm_Terminate()

End Sub

Private Sub UserForm_Zoom(Percent As Integer)

End Sub



    ' 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-gr
ade 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

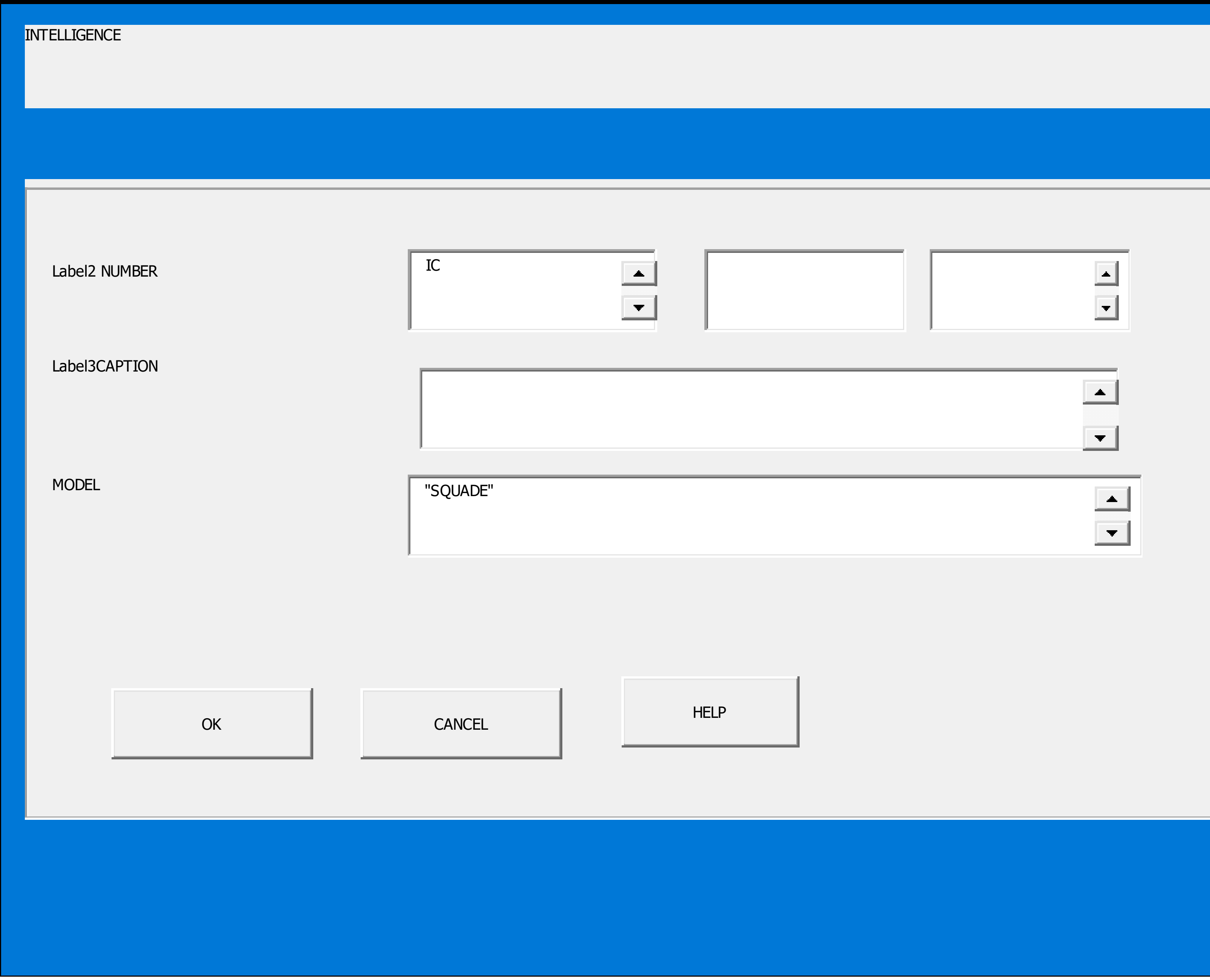
?? Microsoft PowerPoint
" Best for: Visual logic diagrams, wiring plans, and training modules
" Why: Great for presenting embedded systems and UI flows
" Features:
o Slide layouts for signal conditioning blocks
o Icons and shapes for relays, meters, transformers
o Export to PDF for distribution

?? 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
```


INTELLIGENCE

Label2 NUMBER IC

Label3CAPTION

MODEL "SQUADE"