CHAPTER 3

SOFTWARE DEVELOPMENT

3.1 Introduction

MATLAB has a wide range of applications, including signal and image processing, communications, control design, test and measurement, financial modeling and analysis, and also apply for power system in Transmission and Distribution. Add-on toolboxes (collections of special-purpose MATLAB functions, available separately) extend the MATLAB environment to solve particular classes of problems, Calculation in Engineering and another areas of application line.

MATLAB® contains mathematical, statistical, and engineering functions to support all common engineering and science operations. These functions, developed by experts in mathematics, are the foundation of the MATLAB language. The core math functions use the LAPACK and BLAS linear algebra subroutine libraries and the FFTW Discrete Fourier Transform library. Because these processor-dependent libraries are optimized to the different platforms that MATLAB supports, they execute faster than the equivalent C or C++ code. MATLAB provides a number of features for documenting and sharing of work. It can integrate your MATLAB code with other languages and applications, and distribute your MATLAB algorithms and applications (Stephen J.Chapman, 2004).

3.2 Graphical User Interface (GUI)

3.2.1 Introduction of GUI

A graphical user interface (GUI) is a pictorial interface to a program. A good GUI can make programs easier to use by providing them with a consistent appearance and with intuitive controls like pushbuttons, list boxes, sliders, menus and so forth. The GUI should behave in an understandable and predictable manner, so that a user knows what to expect when an action is taken. There does not contain a complete description of components or GUI features, but it does provide the basics required to create functional GUIs for our programs (Stephen J.Chapman, 2004).

This Environment contains pushbuttons, toggle buttons , lists, menus , text boxes, and so forth, all of which are already familiar to the user, so that can concentrate on using the application rather than on the mechanics involved in doing things. However, GUIs are harder for the programmer because a GUI-based program must be prepared for mouse clicks for any GUI element at any time. The two principle elements required to create a MATLAB graphical User interface are

1. Components - The type of components include graphical controls, static elements, menus and axes. Graphical controls and static elements are created by the function unicontrol, and menus are created by the functions unimenu and unicontex tmenu. Axes, which are used to display graphical data, are created by the function axes.
2. Callbacks- There must be some way to perform an action if a user clicks a mouse on a button or types information on a keyboard. A mouse click or a key press is an event, and the MATLAB program must respond to each event if the program is to perform its function. When clicked on a button that event must cause the

MATLAB code that implements the function of the button to be executed. The code executed in response to an event is known as a call back.

The basic steps required to create a MATLAB GUI are:

1. Decide the elements which are required for our program to be used in GUI and it's function. Make a rough layout of the components by hand on a piece of paper.
2. Use the MATLAB tool called guide (GUI Development Environment) to lay out the components on a figure. The size of the figure as well as the alignment and spacing of components on the figure can be adjusted using the tools built into guide as shown in Figure 3.1.
3. Use the MATLAB tool called the property Inspector (built in guide) to give each component a name ("tag") and to set the characteristics of each component, such as its color or the text it displays.
4. Save the Figure to a file. When the figure is saving, two files will be created on disk with the same name but different extentions. The fig file contains the GUI layout and the components of the GUI; the M-file contains the code to load the figure along with the skeleton callback functions for each GUI element.
5. Write code to implement the behavior associated with each callback function (Stephen J.Chapman, 2004).

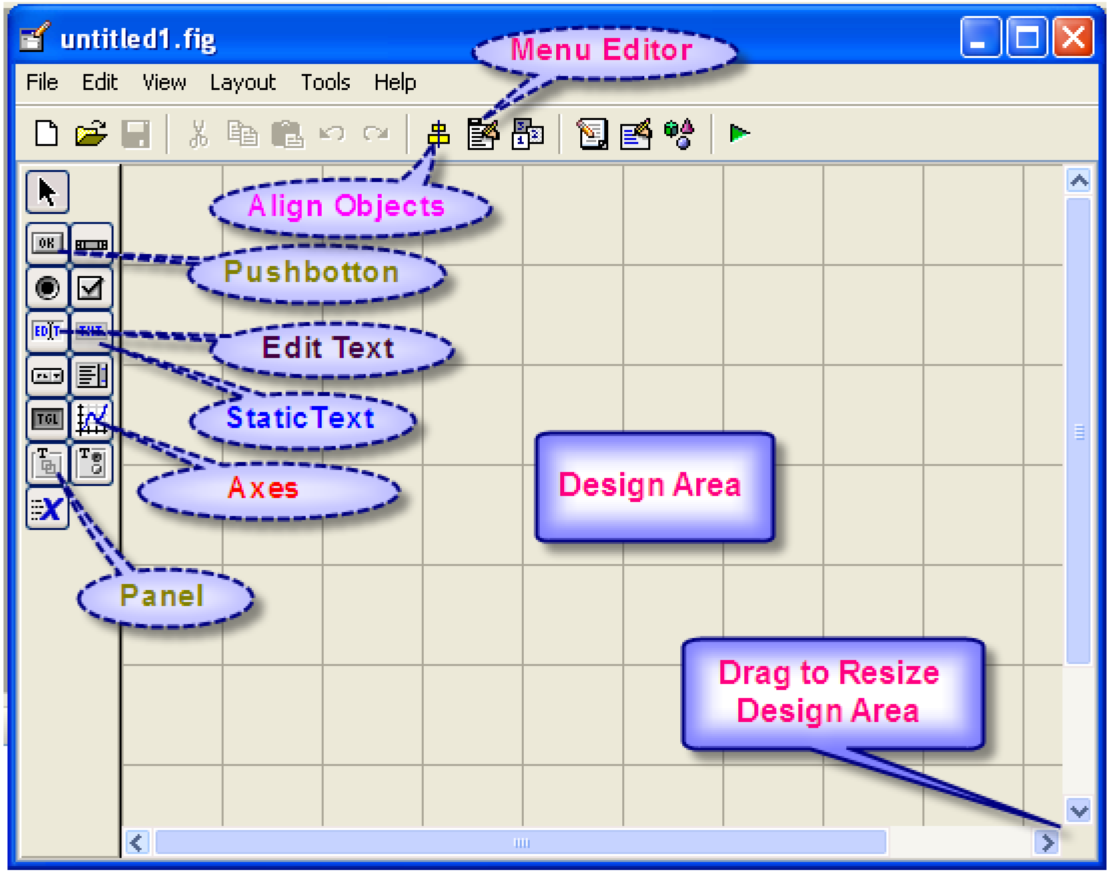


Figure 3.1 The guide Tool Window.

3.2.2 Step by step Using GUIDE

Below is a step-by-step description of the procedures used to make a program using GUIDE. It will necessary to have MATLAB editor and GUIDE open simultaneously.

Step 1: Open GUIDE. Type 'guide' in the Command window. Choose 'blank GUI' (default) option at the opening dialog window show in Figure 3.2. The GUIDE Quick Start dialog box contains two tabs:

- To create New GUI — asks we start creating the new GUI by choosing a template for it. We can also specify the name by which the GUI is saved.
- To Open Existing GUI — enables to open an existing GUI in GUIDE and choose a GUI from your current directory or browse other directories.

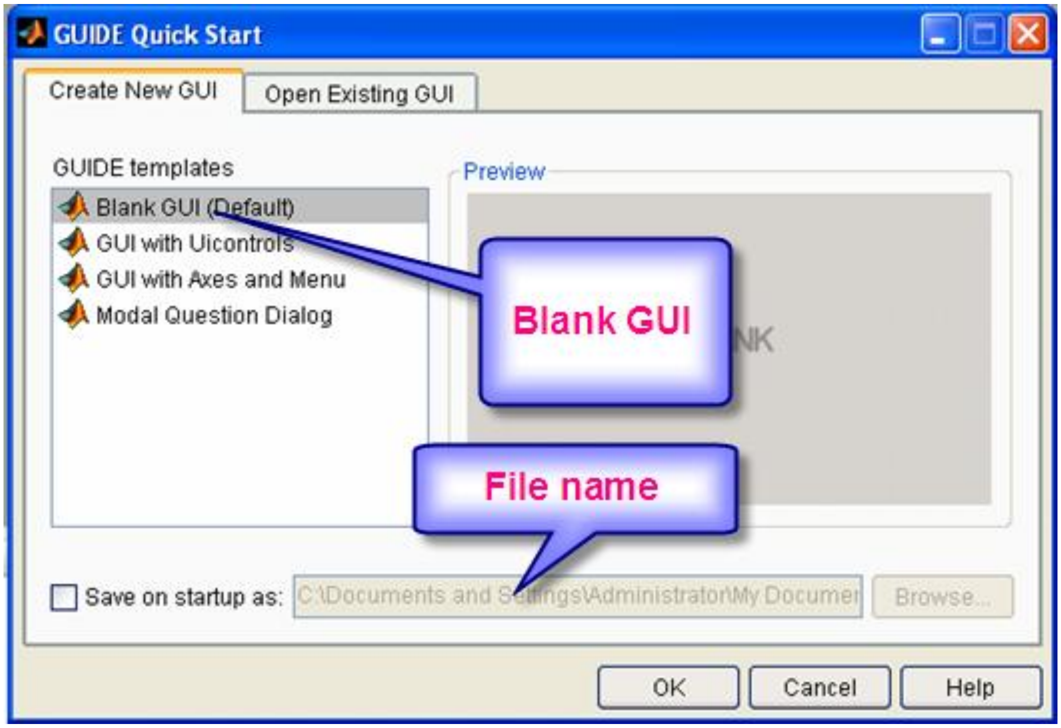


Figure 3.2 Guide of command window

Step 2: When open a GUI, It is displayed in the Layout Editor, which is the control panel for all of the GUIDE tools. Figure 3.3 shows the Layout Editor with a blank GUI template. Set the size of the GUI by resizing the grid area in the Layout Editor. Click the lower-right corner and drag it until the GUI is the desired size. If necessary, make the window larger. It can layout GUI by dragging components, such as panels, push buttons, pop-up menus, or axes, from the component palette, at the left side of the Layout Editor, into the layout area.

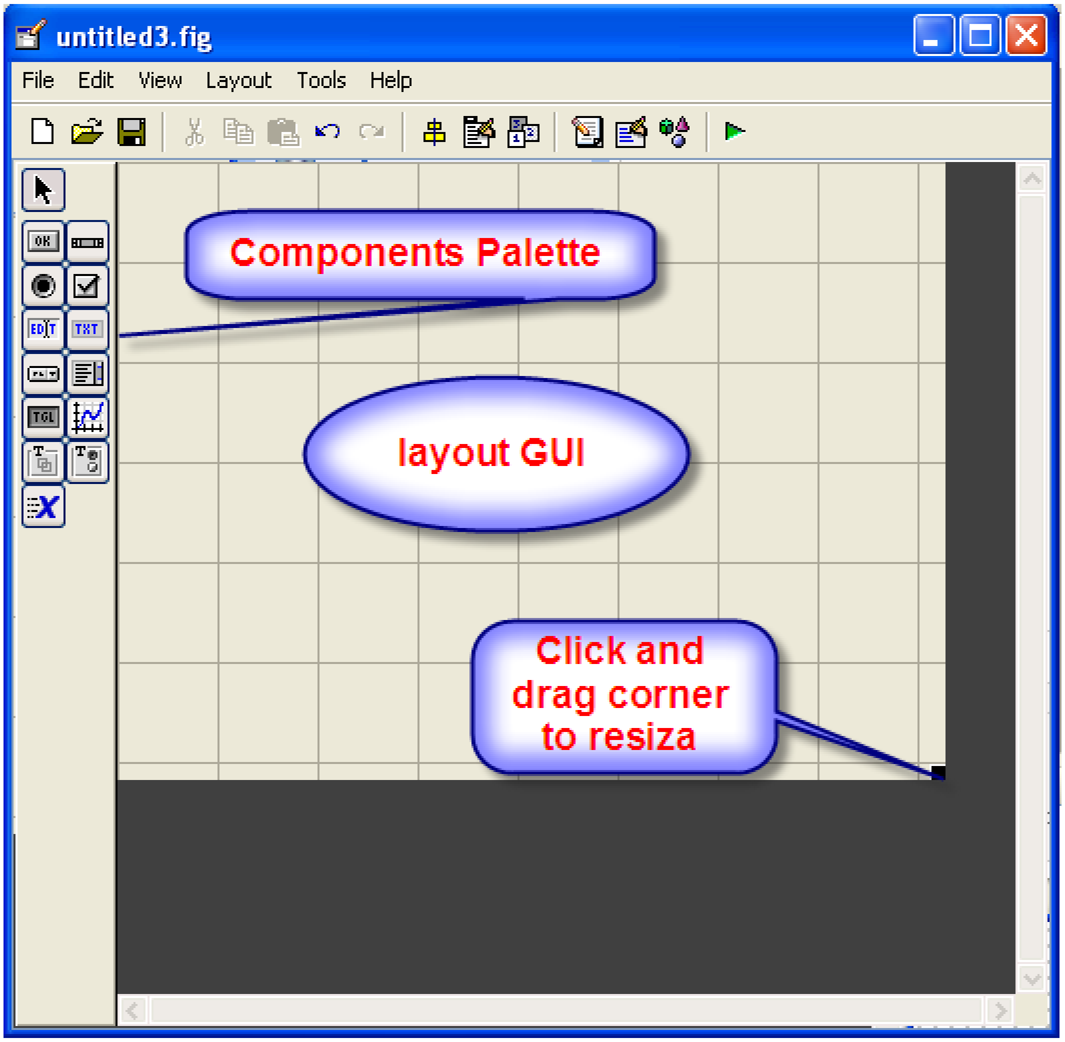


Figure 3.3 control panel GUI Tools and Blank GUI template.

Step 3: In GUI components, it contains a single “Pushbutton” and ‘Static text ' field in the toolbar at the left and draw at the layout area .Double click on left mouse and then selected the “Property Inspector” from the toolbar . Set properties (color, string, etc.) of objects that you create, using the Property Inspector. The image shows the Property Inspector for the slider in the GUIDE window below. Click on any object that you have created, and the Property Inspector will display its properties. Just click on any property in the list and type new values. The Property Inspector window shown in Figure 3.4 will appear.

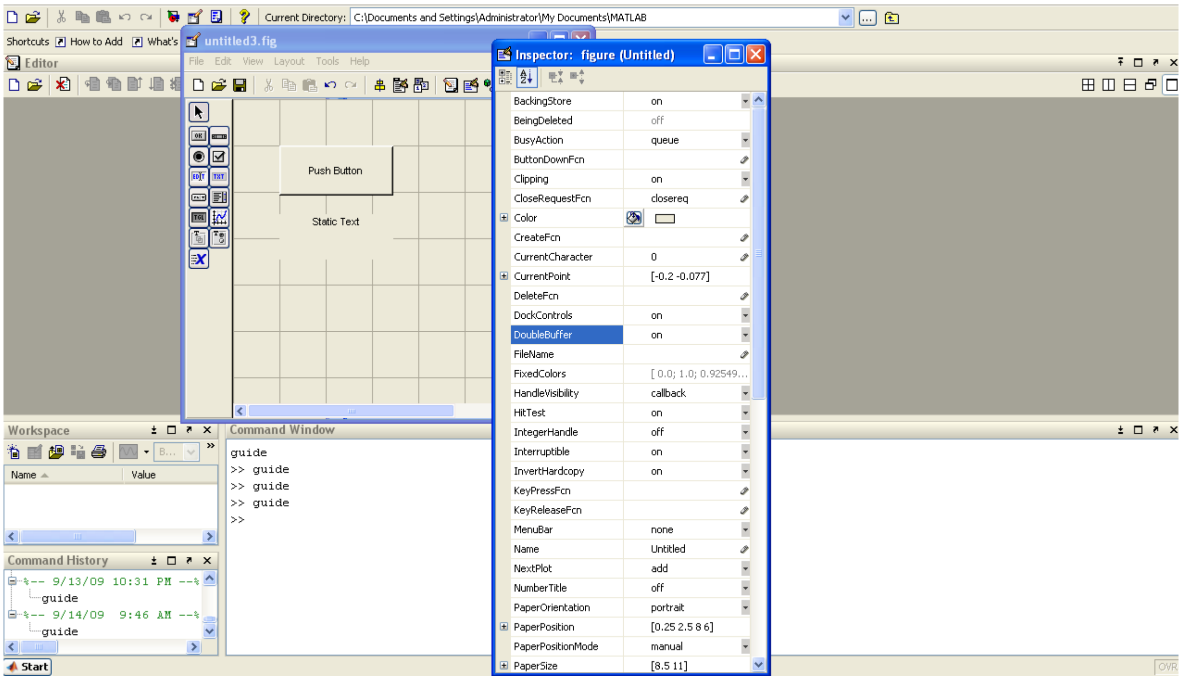


Figure 3.4 The Property Inspector showing the properties of the pushbutton. Note that the String is set to “click here”, and the Tag is set to “My First Button”.

Step 4: Each object has a property named “tag”. This can use any name that resembling. MATLAB uses the “tag” to name the **handle** of the object. Thus, if make a slider used to control some variable (say, amplitude of something or other), user might want to make the tag be “UM”. Then, the handle of the slider will be stored with all other handles in the structure “UM”, and this one will be 'UM’. The Property Inspector window shown in Figure 3.5 will appear.

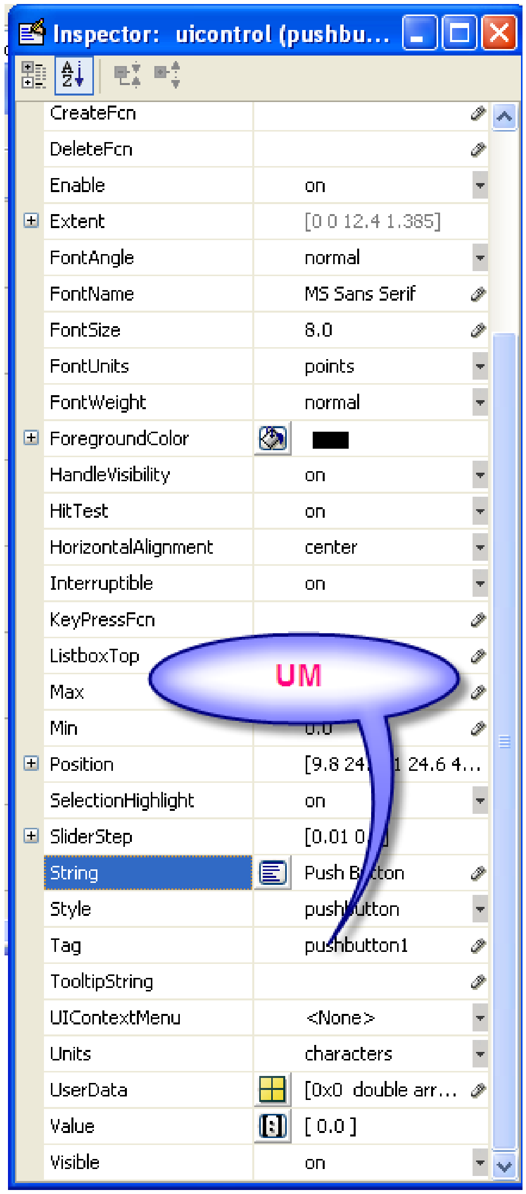


Figure 3.5 The layout of property Inspector showing the "Tag"

Step 5: For each object that has a callback, must enter a two-word callback in the Property Inspector. The first word is just the name of the program. The second word is the unique callback word. Example, if user program is called "UM" and user are creating a pushbutton that quits the program, the callback might be "UM quit". Then, at the end of m-file ("UM"), must create a function named "quit", and the code to service the callback. The Property Inspector window shown in Figure 3.6 will appear

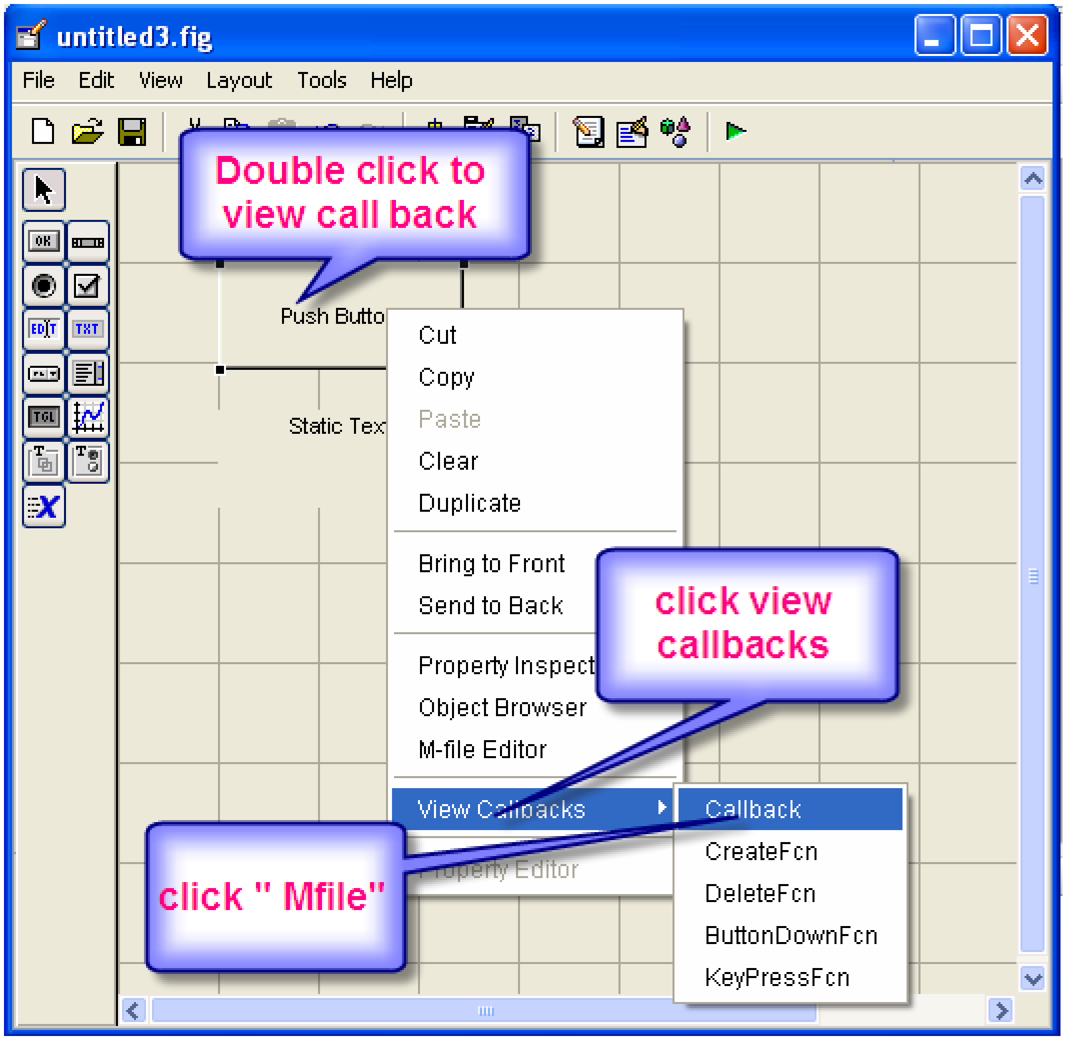


Figure 3.6 The property Inspector showing the call back

Step 6: To implement the callback sub function for the pushbutton. This function will include a persistent variable that can be used to count the number of clicks that have occurred. When a click occurs on the pushbutton, MATLAB will call the function MyfirstButton_callback as the first argument. Then function My firstGUI will call sub function My firstButton_callback as shown in Figure 3.7. This function should increases the count of clicks by one, create a new text string containing the count, and store the new string in the string property of the text field Myfirst Text.

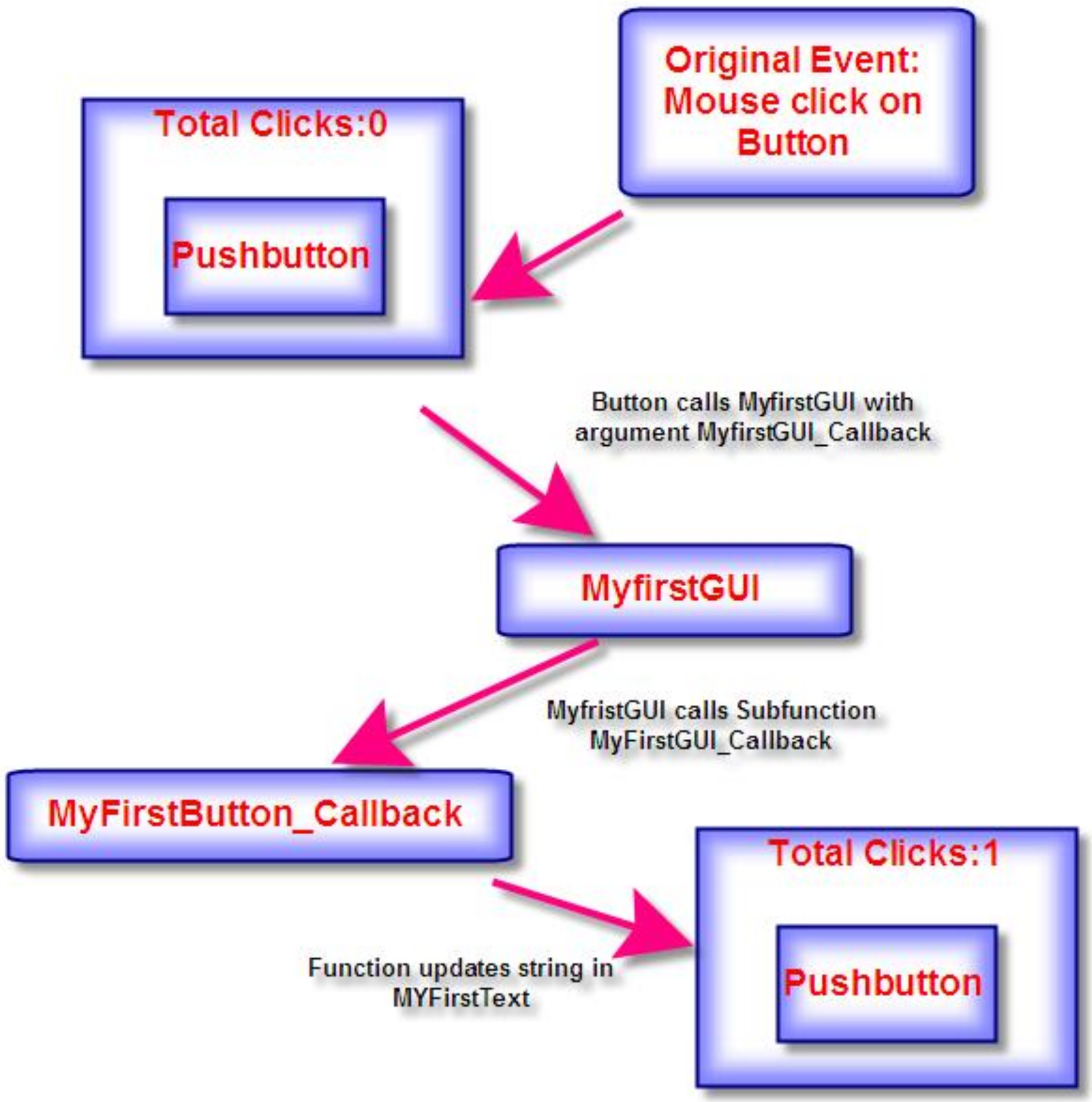


Figure 3.7 Event handling in program *MyFirstGUI*.

When user clicks on the button with the mouse, the function *MyFirstGUI* is call automatically with the argument *MyFirstButton_Callback*. Function *MyFirstGUI* in turn calls sub-function *MyFirstButton_Callback*. This function increments *count*, and then saves the new count in the text field on the GUI (Stephen J. Chapman, 2004).

3.2.3 Available Components Graphical User Interface Component (GUI Component)

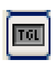
The component palette summarizes the basic characteristics of common graphical user interface components. It describes how to create and use each component, as well as the types of events for each component can generate.

- Edit Text Buttons 

Edit Text components are fields that enable users to enter or modify text strings. Use edit text when you want text as input. Users can enter numbers but you must convert them to their numeric equivalents.

- Pushbuttons 

Pushbuttons generate and action when clicked. For example , an Ok button might apply settings and close a dialog box. When you click a push button, it appears depresed; when you release the mouse button, the push button appears raised.

- Toggle Buttons 

Toggle buttons generate an action and indicate whether they are turned on or off. When you click a toggle button, it appears depressed, showing that it is on.When you do so, the button returns to the raised state, showing that it is off. Use a button group to manage mutually exclusive toggle buttons.

- Static Text buttons 

Static text controls display lines of text. Static text is typically used to label other controls, provide directions to the user, or indicate values associated with a slider. Users cannot chnage static text interactively.

- Axes Buttons 

Axes enable your GUI to display graphics such as graphs and images. Like all graphics objects, axes have properties that you can set to control many aspects of its behavior and appearance. See axes properties in the MATLAb graphics documentation and commands such as the following for more informations on axes objects:plot, surf, line , bar, polar, pie ,contour and mesh

- Radio Buttons 

Radio buttons are similar to check boxes,but radio buttons are typically mutually exclusive within a group of related radio buttons. That is when you select one button the previously selected button is deselected. To activate a radio button, click the mouse button on the object, the display indicates the state of the button. Use a button group to manage mutually exclusive radia buttons.

3.2.4 Programming GUI

This topic shows you how to generate the data to be plotted when the user clicks a button. This data is generated in the opening function. The opening function is the first callback in every GUIDE-generated GUI M-file. User can use it to perform tasks that need to be done before the user has access to the GUI. In this example, user add code that creates three data sets to the opening function. The code uses the MATLAB functions peaks, membrane, and sinc. Display the opening function in the M-file editor. If the GUI M-file, simple_gui.m, is not already open in your editor, open it by selecting M-file Editor from the View menu. In the editor, click the functionicon on the toolbar, and then select simple_gui_OpeningFcn in the pop-up menu that displays. A portion of the M-file automatically created by guide is shown in Appendixe A. This file is containing the main function MyfirstGUI, plus sub functions to specify the behavior of the active GUI components (Stephen J. Chapman, 2004).

Programming GUI contain in Pushbutton

This is the push button's Callback by double click push button icon. It displays the string Goodbye at the command line and then closes the GUI and will available at Mfile in MATLAB.

Programming GUI contain in Edit Text

To obtain the string user types in an edit box, get the String property in the Callback MATLAB returns the value of the edit text String property as a character string. User can enter the numeric values by converting the characters to numbers. User can do this using the str2double command, which converts strings to doubles. If the user enters nonnumeric characters, str2double returns NaN and can use the following code in the edit text callback. It gets the value of the String property and converts it to a double. After that, user can check the converted value is NaN (isnan), indicating the user entered a nonnumeric character and displays an error dialog (errordlg).

Programming GUI contain in Panel

Panels group GUI components and can make a GUI easier to understand by visually grouping related controls. A panel can contain panels and button groups as well as axes and user interface controls such as push buttons, sliders, pop-up menus, etc. The position of each component within a panel is interpreted relative to the lower-left corner of the panel (Stephen J. Chapman, 2004).

3.3 Current Transformer Algorithm

3.3.1 Current Transformer

The Role of the CT is particularly important with differential protection. The current comparison carried out by the differential protection only functions correctly when the primary currents are transformed to the secondary side with correct polarity and sufficient accuracy by the CTs (Gerhard Ziegler, 2005).

All current transformers used in protection are basically similar in construction to standard transformers in that they consist of magnetically coupled primary and secondary windings, wound on a common iron core, the primary winding being connected in series with the network unlike voltage transformers.

They must therefore withstand the networks short circuit currents (Les Hewitson, Mark Brown, Ben Ramesh, 2004). A current transformer is used to transform a primary current quantity in terms of its magnitude and phase to a secondary value such that in normal conditions the secondary value is substantially proportional to the primary value (Dr. C.R bayliss, 1999). Several terms are used in connection with CTs and these are described below:

- **Rated primary (or secondary) current**. This value, marked on the rating plate of the CT, is the primary or secondary current upon which the performance of the transformer is based.
- **Rated transformation ratio**. The rated transformation ratio is the ratio of rated primary current to rated secondary current and is not necessarily exactly equal to the turn’s ratio.

The magnetizing current depends upon the magnitude of the primary voltage which in turn depends upon the magnitude and power factor of the burden. It is possible partially to compensate for the magnetizing current ratio error in CT designs by slightly reducing the number of turns on the secondary. The standards to which the CTs are specified may not detail a continuous overload rating. It is therefore prudent to choose a primary current rating at least equal to the circuit rating. An Accuracy class 5P (P stands for protection) is usually specified for large system where accurate grading of several stages of protection is required. An Accuracy class 10P is also often acceptable and certainly satisfactory for thermal overload relays on motor circuits. These accuracy classes correspond to 5% or 10% composite error with rated secondary burden connected at all currents up to the primary current corresponding to the rated accuracy limit factor (Dr. C.R bayliss, 1999).

- Rated output at rated secondary current. The value, marked on the rating plate, of the apparent power in VA that the transformer is intended to supply to the secondary circuit at the rated secondary current (Dr.C.R bayliss, 1999).

The rated VA should be specified to correspond to the relay and connecting lead burden at rated CT secondary current. If relays are mounted on the switchgear adjacent to the CTs then the lead burden can often be neglected, It is best to allow a margin for greater than anticipated burden but this should be included in the specification for the rated accuracy limit factor.

- Rated accuracy limits Factor (RALF). The primary current up to which the CT is required to maintain its specified accuracy with rated secondary burden connected, expressed as a multiple of rated primary current.

Ideally the RALF current should not be less than the maximum fault current of the circuit up to which IDMTL relay grading is required, and should be based upon transient reactance fault calculations. If a switchboard is likely to have future additional fault in feeds then it is sensible to specify a RALF corresponding to the switchgear fault-breaking capacity. Rated outputs higher than 15VA and rated accuracy limit factor higher than 10 are not recommended for general purposes. It is possible to make a trade off between RALF and rated output but when the product exceeds 150 the CT becomes uneconomic with large physical dimensions. An RALF of 25 is an economic maximum figure. A reduction in RALF is not always possible and therefore the following measures should be considered

- Use the highest possible CT ratio
- Investigate relays with a lower burden. Solid state relays have burdens of 0.5VA or less and do not change with tap setting.
- A lower system voltage levels consider the use of fuses on circuits of low rating but high fault level.

For Protective purpose current transformer application may be defined in terms of the “knee point”. This is the voltage applied to the secondary terminals of the CT with all other windings being open circuited, which, when increased by 10% causes the existing current to be increased by 50 % (Dr.C.R bayliss, 1999).

Class PX is the definition in IEC 60044-1 for the quasi-transient current transformers formerly covered by Class X of BS 3938, commonly used with unit protection schemes. A typical CT magnetizing characteristic is shown in Figure 3.8. Other standards (BS3938) catered for the specification of such “Class X” CTs in terms of:

- Rated primary current
- Turns Ratio
- Rated knee point EMF
- Maximum exciting current at a stated percentage of rated knee point EMF
- Maximum resistance of the secondary winding.

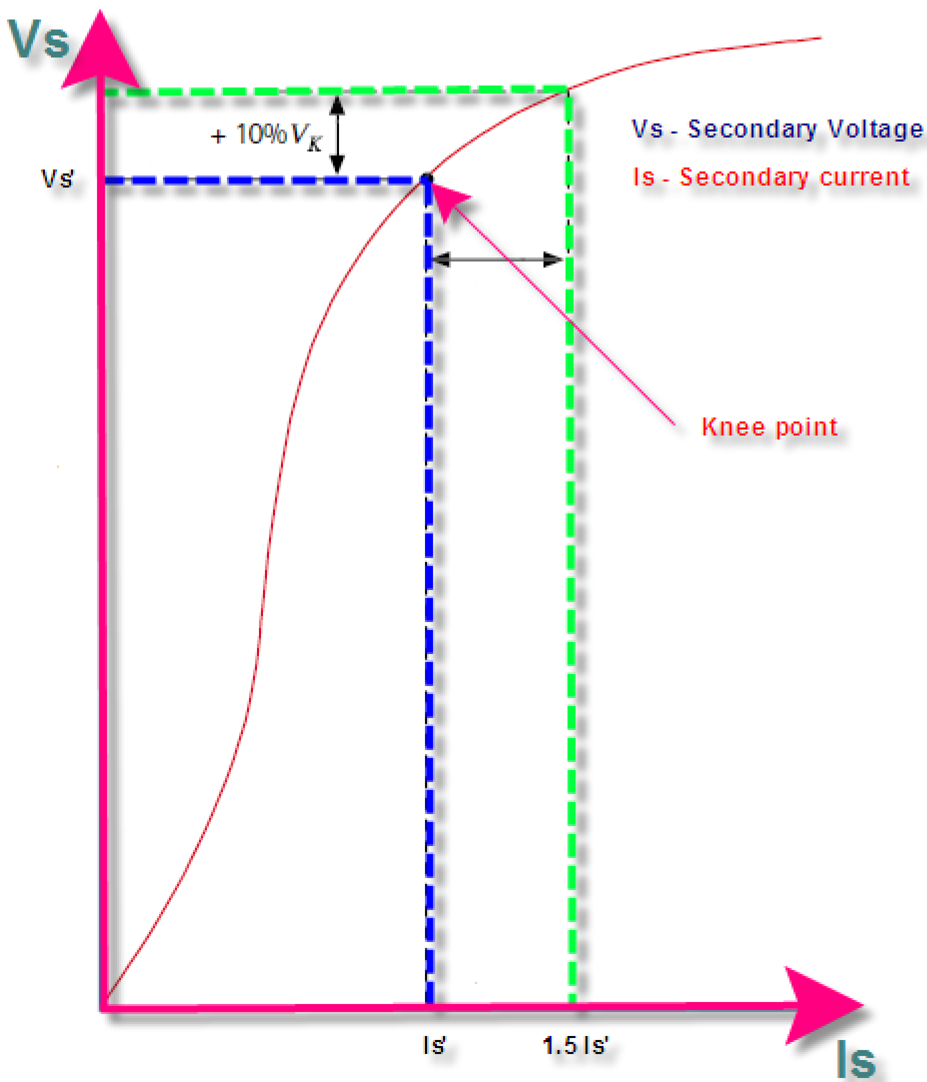


Figure 3.8 Typical Magnetizing characteristic of PX class

3.3.2 Technical Implementations

3.3.2.1 **Transformer Biased Differential Protection**. According to the guidance provided by the Manufacturer AREVA KBCH , The Knee point Voltage Requirement for transformer can be calculated using the following equations:

$$Vk>24\ In\ (Rct + Rl + Rb) \tag{3.1}$$

3.3.2.2 **Transformer High Impedance Differential Protection**. The current transformer used in high impedance circulating current differential protection systems must be equal to the turns ratio and have reasonably low secondary winding resistance. Current transformer of similar magnetizing characteristic with low reactance construction such as IEC60044 class PX or similar are preferred. The Relay requirements are based upon a calculation of the required knee point voltage it being the point on the magnetization curve at which a 10% increases in excitation current (Areva, 2001). The required stability voltage setting(Vs') and minimum knee point voltage (Vk) are calculated as follows:

$$Vs' \geq If(Rs+Rp) \tag{3.2}$$

$$VsA=VA/Ir + Ir\ Rsr \tag{3.3}$$

$$Vk \geq 2VsA \tag{3.4}$$

3.3.2.3 **Back-up Distance Protection**. Two calculations must be performed – One for the three phase fault current, and another for earth (ground) faults. The higher of the two calculated Vk voltages must be used: For Class X current transformers with a knee point Voltage greater than or equal to that calculated can be used. Class 5P protection CTs can be used, noting that the knee point Voltage equivalent these offer can be approximated from:

$$Vk= (VA \times ALF)/In + (Rct \times ALF \times In) \tag{3.5}$$

3.3.3 Flow Chart Current transformer calculations.

The following shows the Flow Chart of the Current transformer calculations.

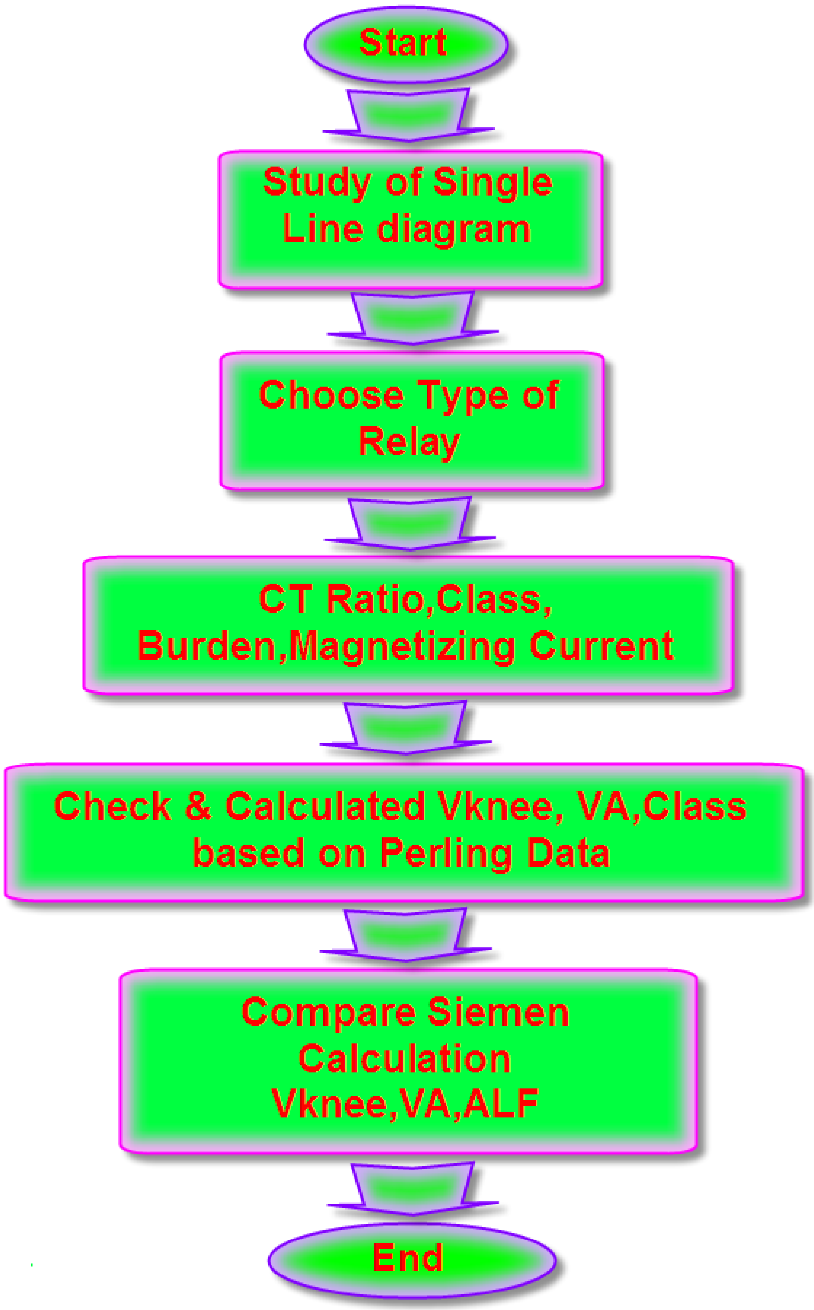
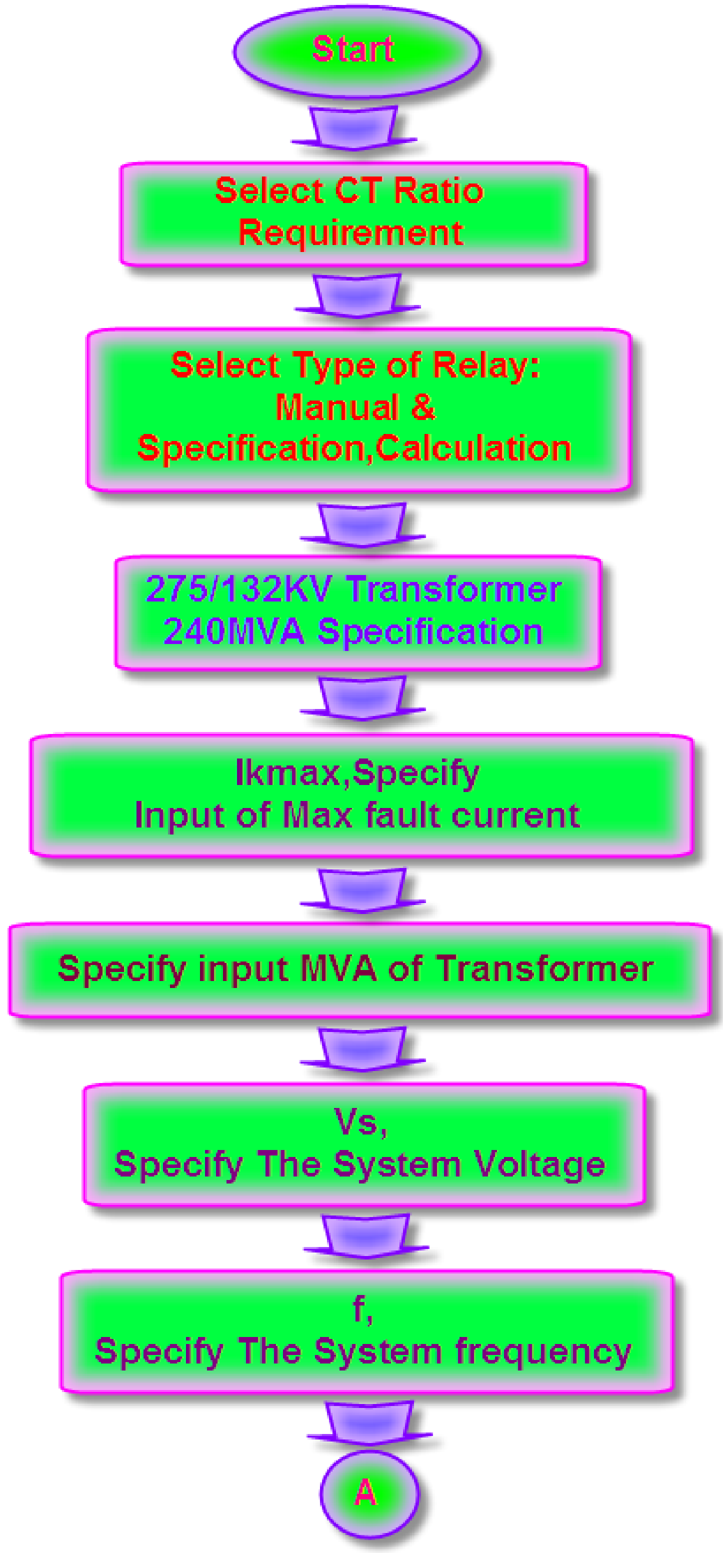
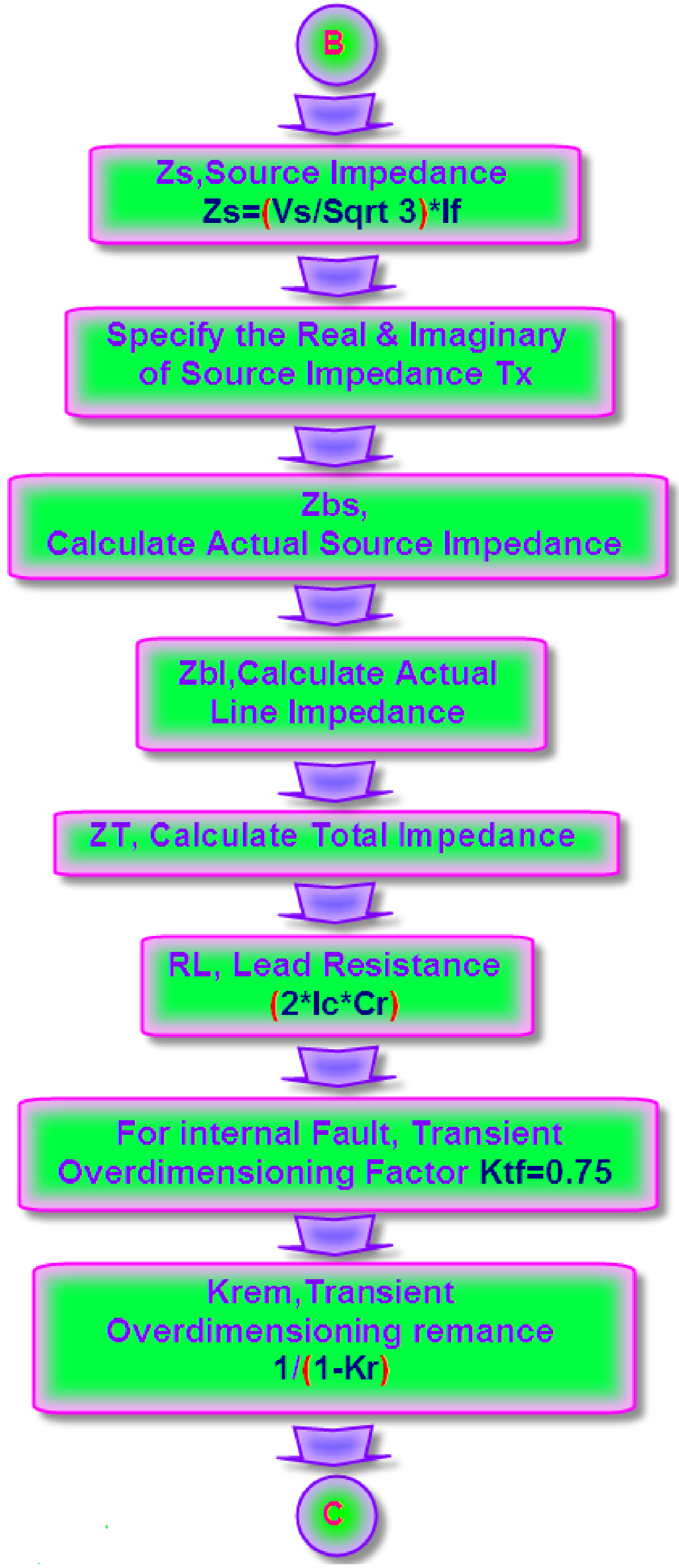


Figure 3.9 Flow chart of Choosing of Current Transformer

3.3.4 Flow Chart of proposed program.




A
X/R, Input of Dc time Constant for 275KV
Ic,Specify Cable Length in Km
Cc,Specify the Cable Saiz in mm square
Cr,Specify Cable Resistance in Ohm/km
L,Distance of Transmission Line
R/T,Turn per ratio in Ohm perturn
ALF,Specify From CT Plate
Tn,DC Time constant of Applicable Fault loop
Xr/2*Pi*f
B


B
Zs,Source Impedance
Zs=(Vs/Sqrt 3)*If
Specify the Real & Imaginary
of Source Impedance Tx
Zbs,
Calculate Actual Source Impedance
Zbl,Calculate Actual
Line Impedance
ZT, Calculate Total Impedance
RL, Lead Resistance
(2*Ic*Cr)
For internal Fault, Transient
Overdimensioning Factor Ktf=0.75
Krem,Transient
Overdimensioning remance
1/(1-Kr)
C

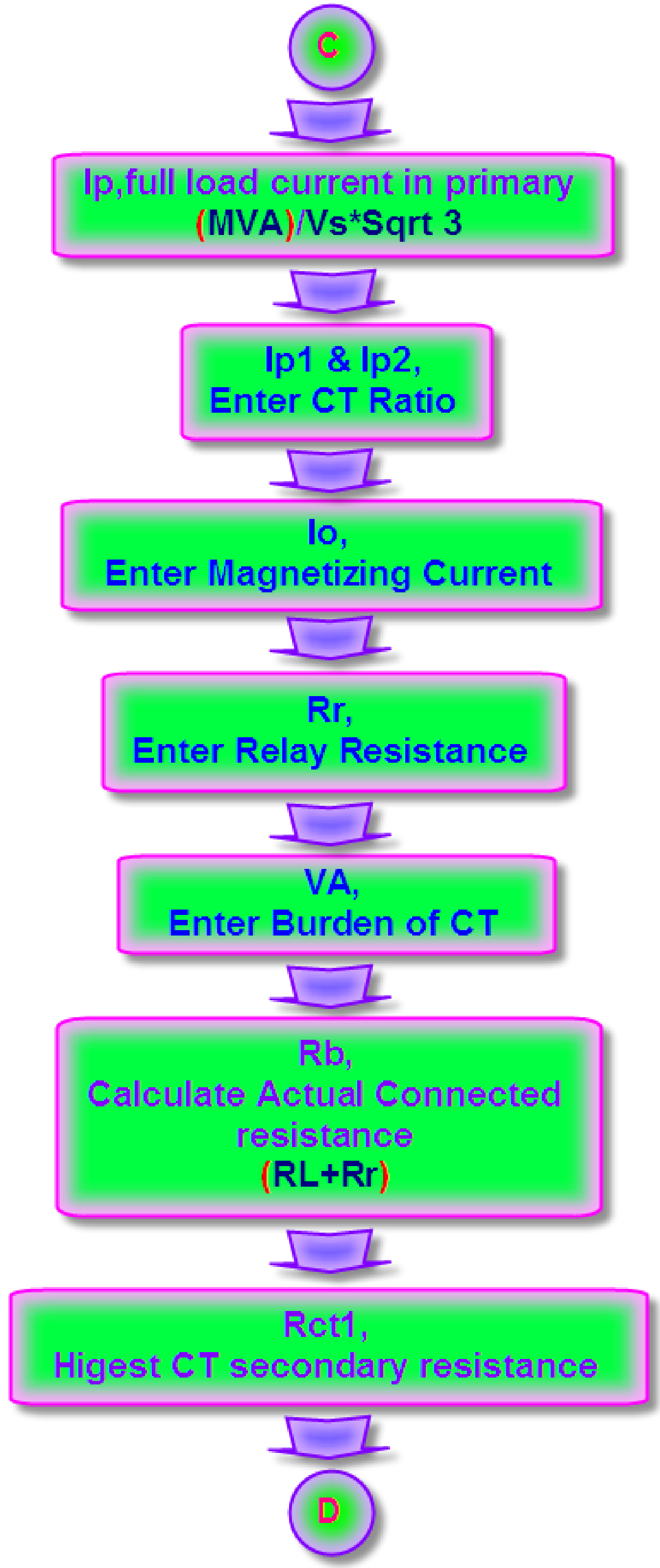

C
Ip,full load current in primary
(MVA)/Vs*Sqrt 3
Ip1 & Ip2,
Enter CT Ratio
Io,
Enter Magnetizing Current
Rr,
Enter Relay Resistance
VA,
Enter Burden of CT
Rb,
Calculate Actual Connected
resistance
(RL+Rr)
Rct1,
Higest CT secondary resistance
D

Figure 3.10 Flow chart of Technical Current Transformer.

In principle, the CT corresponds to a transformer that is current driven. During normal operation, its induction (flux density) is small in comparison with its saturation induction. The induction increases proportional to the increases of the primary current and correspondingly the voltage across the connected secondary burden. The CT in general is dimensioned such that a certain AC fault current can be transformed without saturation (Gerhard Ziegler, 2005). From the protection point of view the leakage induction of the CT may be neglected.

During the saturation free operation, the magnetizing current may also be neglected and the Ampere's law states the following:

$$I_1.w_1 = I_2.w_2 \text{ or } I_2 / I_1 = w_1/w_2 = r_{CT} \tag{3.6}$$

The voltage at the secondary terminals of the CT correspond to the voltage drop across the connected burden

$$U_2 = I_2.R_B^1 \tag{3.7}$$

The burden consists of the impedance of the CT secondary cables, the relay and if applied, interposing transformers and further devices. The reactive components may in general be neglected so that the pure resistive burden can be used for the calculation. If detailed information is not available, the indicated rating in VA is used as a purely resistive burden in the following:

$$R = P\,[VA]/I[A]^2 \tag{3.8}$$

This corresponds to the worst case. The power supplied by the CT is then

$$P_2 = U_2.I_2 = I_2^2.R_B \tag{3.9}$$

The induction in the CT is proportional to the internal EMF

The internal EMF is given by

$$E_2 = I_2.(R_{CT} + R_B) \tag{3.10}$$

The current Transformer dimension is selected such that it provides a defined accuracy for the fault current up to a threshold current (rated accuracy limit current) with the rated burden R_{BN} connected. The rated accuracy limit current is indicated as a multiple of the nominal current

$$I_{al} = ALF.\ I_N \tag{3.11}$$

The factor ALF is also known as the accuracy limit factor.

The internal EMF that arise when this rated accuracy limit factor flows then corresponds to the saturation voltage of the CT

$$E_{al} = ALF.\ I_{2N}.\ (R_{CT} + R_B) \tag{3.12}$$

With Class X according to BS 3938 this voltage is designated as the knee point voltage U_{KN} and is used to indicate the transformation capacity instead of the nominal burden and rated accuracy limit current. The accuracy is however defined somewhat differently. In general, the relation $U_{KN} \cong 0.8,.Eal$ may be applied (Gerhard Ziegler, 2005) .

The accuracy limit factor.

The accuracy limit factor of the CT only applies when rated burden is connected. If a smaller burden is connected and increased operating accuracy limit factor ALF' may be derived from the equation 2.10 and 2.11:

$$ALF' = I_F / I_N\ .\ K_{TF}.\ K_{Rem} \tag{3.13}$$

The corresponding rated accuracy limit factor is then

$$ALF = (R_{CT} + R_B / R_{CT} + R_N)\ .\ ALF' = (P_i + P_B / P_i + P_N)\ .\ ALF \tag{3.14}$$

The ratio IF/IN must consider the maximum fault current that arises. Often, the rated short circuit current of the plant is used in this context.

3.4 Transient Dimensioning Factor.

If the short circuit current is asymmetrical, due to the DC component saturation in the core will be reached much earlier. If saturation occurs during the period a protective relay is carrying out the measurement, the transformer will have to be over dimensioned. This over dimensioning factor is defined by $K_{TF.}$ The transient dimensioning factor K_{TF} is the ratio of the theoretical total secondary linked flux to the peak instantaneous value and are proportional to the corresponding magnetizing currents K_{TF} and is given by (N.E. Korponay, 1975),

$$K_{TF} = i_o / i_{o\sim max} \tag{3.15}$$

Substituting expression for i_o and $i_{o\sim max}$ the transient dimensioning factor can be written as

$$K_{TF} = \frac{\left[\frac{Tp - qTs}{Tp - Ts}. \text{Cos}\,\theta\right] e^{-t/Tp}}{[(1 - q / Ts\,\omega)^2 + g^2]^{1/2}} + \frac{\left[g\,\text{Cos}\,\theta + \frac{1 - q}{Ts\,\omega}.\text{Sin}\,\theta - \frac{Tp - qTs}{Tp - Ts}.\text{Cos}\,\theta\right]}{[(1 - q / Ts\,\omega)^2 + g^2]^{1/2}} - \frac{\left[\frac{(1 - q)}{Ts\,\omega}.\text{Sin}\,(\omega t + \theta) - \frac{Tp - qTs}{Tp - Ts}.\text{Cos}(\omega t + \theta)\right]}{[(1 - q / Ts\,\omega)^2 + g^2]^{1/2}} \tag{3.16}$$

Where $g = (q\omega^2 Ts^2 + 1)(\omega^2 Ts^2 + 1)$ and Assuming $\omega^2 Ts^2 + 1 = \omega^2 Ts^2$

If the burden is purely resistive as in the case of static or numerical protection, L2 = 0 and thus q =0 and g can be ignored. Therefore equation 3.16 can be simplified and the transient dimensioning factor becomes

$$K_{TF} = \frac{\omega.Tp\,.Ts}{Tp - Ts}. \text{Cos}\,\theta\ (e^{-t/Tp} - e^{-t/Ts)} + \text{Sin}\theta.\ e^{-t/Ts} - \text{Sin}\,(\omega t + \theta) \tag{3.17}$$

In order to achieve the dimensioning purpose, the transient factor can be calculated by using equation sin ($\omega t + \theta$) = -1. The Transient dimensioning factor K_{TF} will then become (N.E. Korponay, 1975),

$$K_{TF} = \frac{\omega . Tp . Ts}{Tp - Ts} . \cos\theta \; (e^{-t/Tp} - e^{-t/Ts)} + \sin\theta . (e^{-t/Ts} + 1) \qquad (3.18)$$

.

3.4.1 The numerical Transient Dimensioning calculations and the program follows the flowchart depicted below.

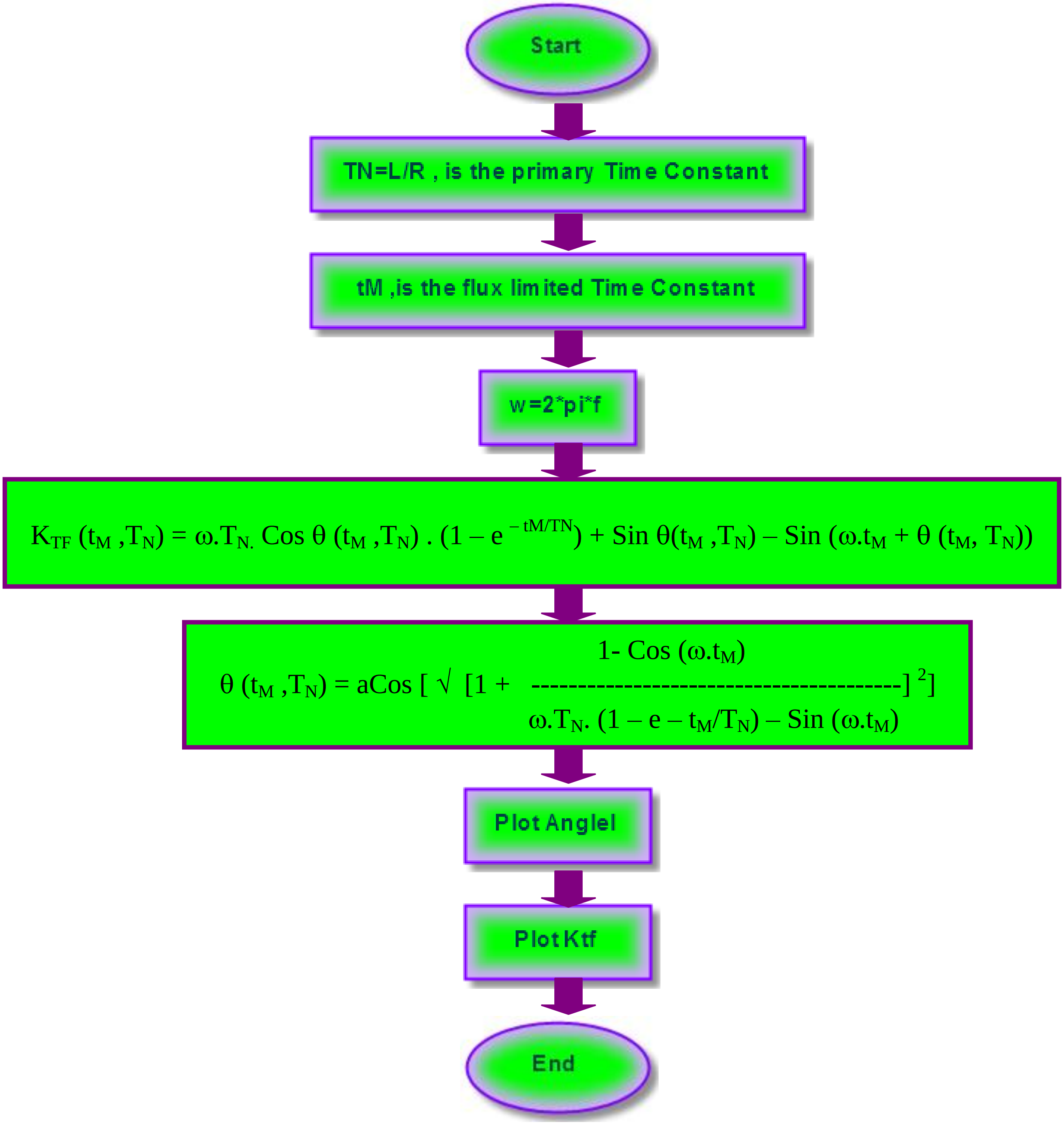


Figure 3.11 Flow chart of over dimensioning Factor K_T

Over- Dimensioning factor for remanence K_{Rem}

The remanence may be as much as 80% (Kr=0.8) in closed iron core CTs. Consequently only 20% flux increases up to saturation would remain as show in Figure 3.12 and the CT dimension would have to be increased by the remanence over dimensioning factor $K_{Rem} = 1/(1 - Kr) = 5$ (Gerhard Ziegler, 2005).

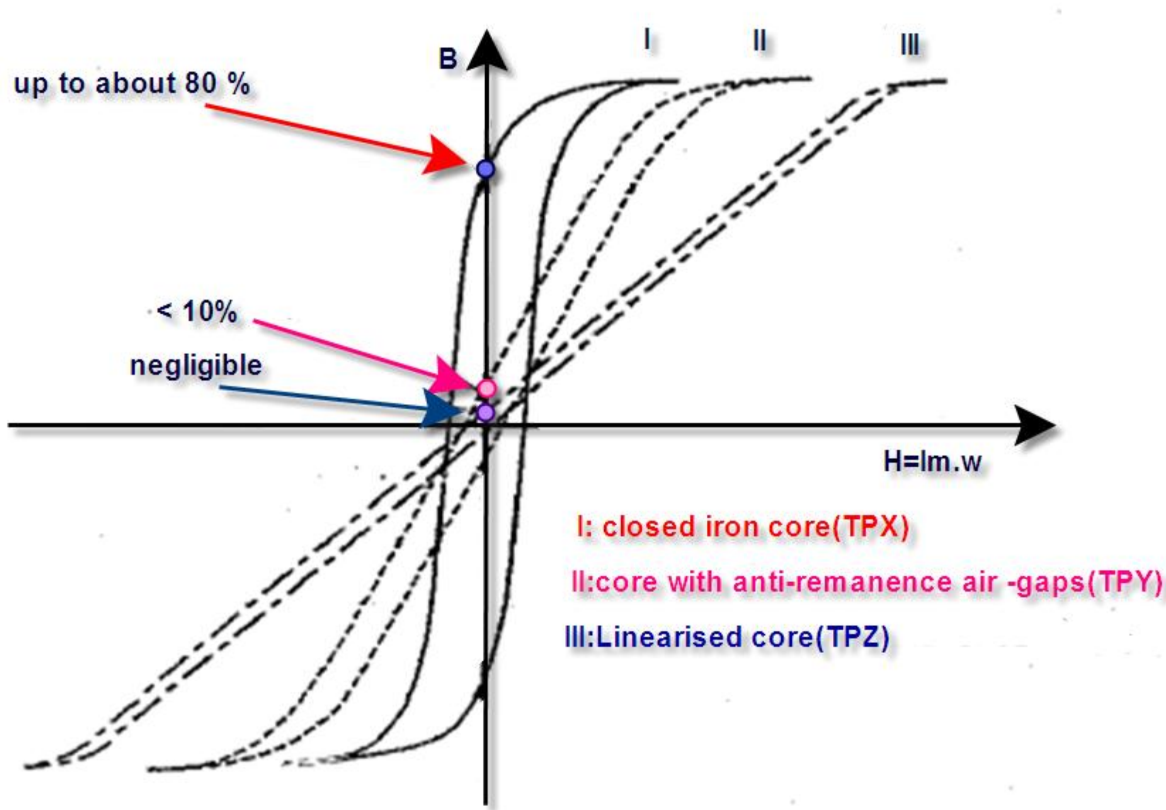


Figure 3.12 CT classes according to IEC 60044-6, magnetizing curves

3.5 Simulation Software

This section will describe the step by step procedure in order to simulate and start the program; by using the MATLAB software. The program can be started by double click the MATLAB icon and appear the MATLAB command window as shown in Figure 3.13.

After user download click the icon, the user can look up the file name at command window. Under the command window, the user type “GUIDE” and Enter. The MATLAB will show as “GUIDE QUICK START” which is shown in Figure 3.15.

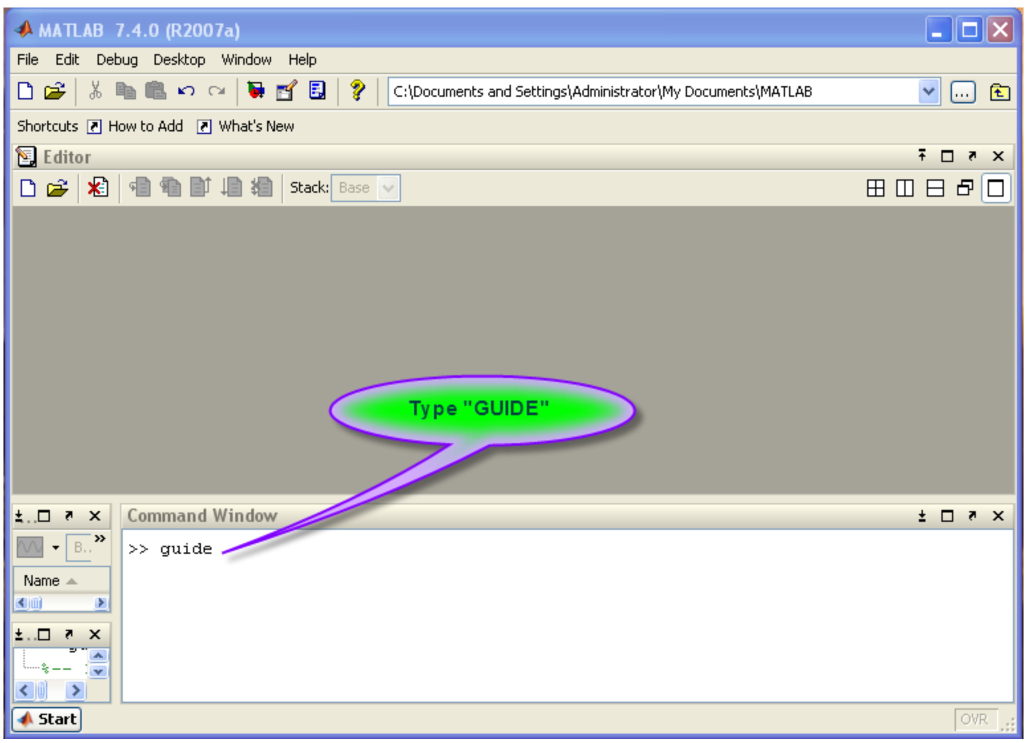


Figure 3.14 MATLAB Command Window.

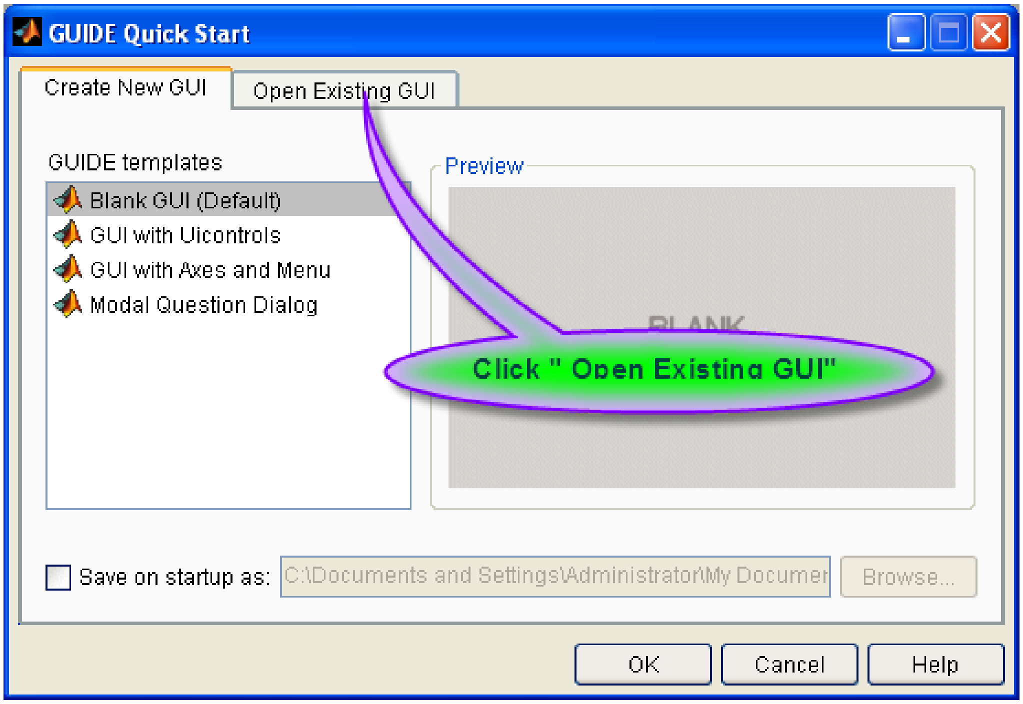


Figure 3.15 Existing GUI File.

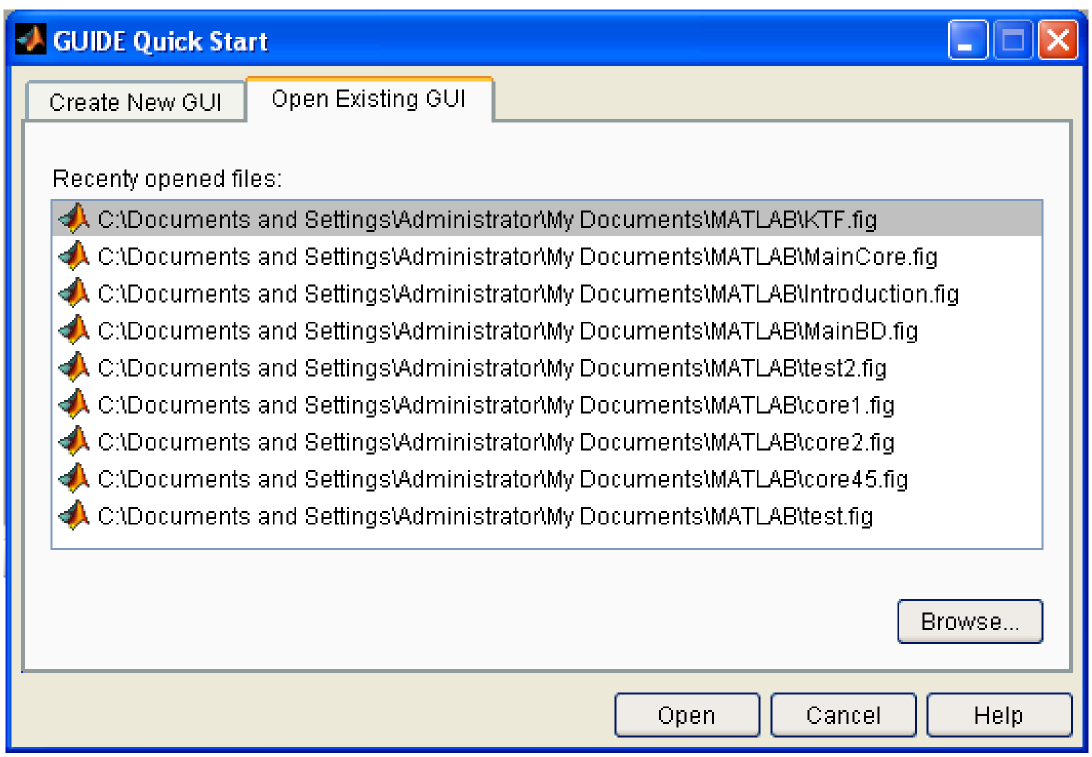


Figure 3.16 GUIDE Quick Start

User must select the file and press button open to open the file. The layout appears as shown in Figure 3.16. The completed GUI layout appears by pressing the Run button as shown in Figure 3.17. While the General layout of Mains shown in Figure 3.18. To know detail about program in Figure 3.17, refer appendix A

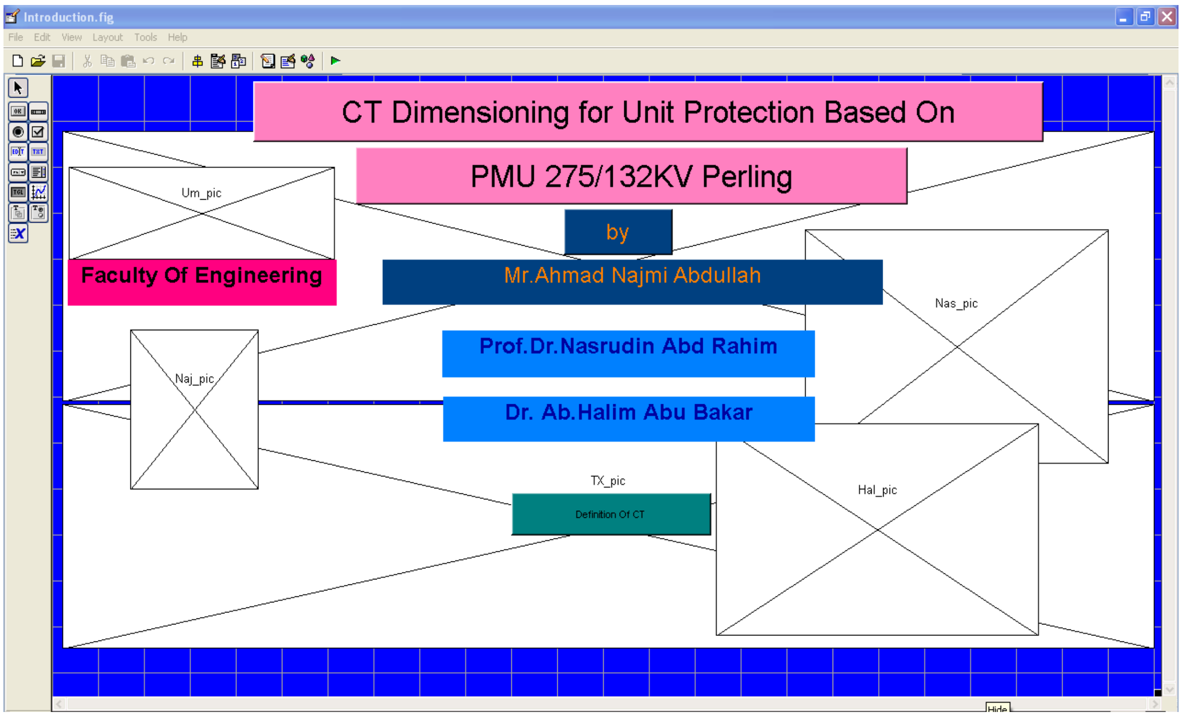


Figure 3.17 Overview layout of GUI design

Figure 3.18 The General Layout of Main GUI.

Figure 3.18 shows the information about project including title, and supervisors. To go further, user should press the "Definition of CT" and the layout of Definition of CT will appear as show in Figure 3.19.

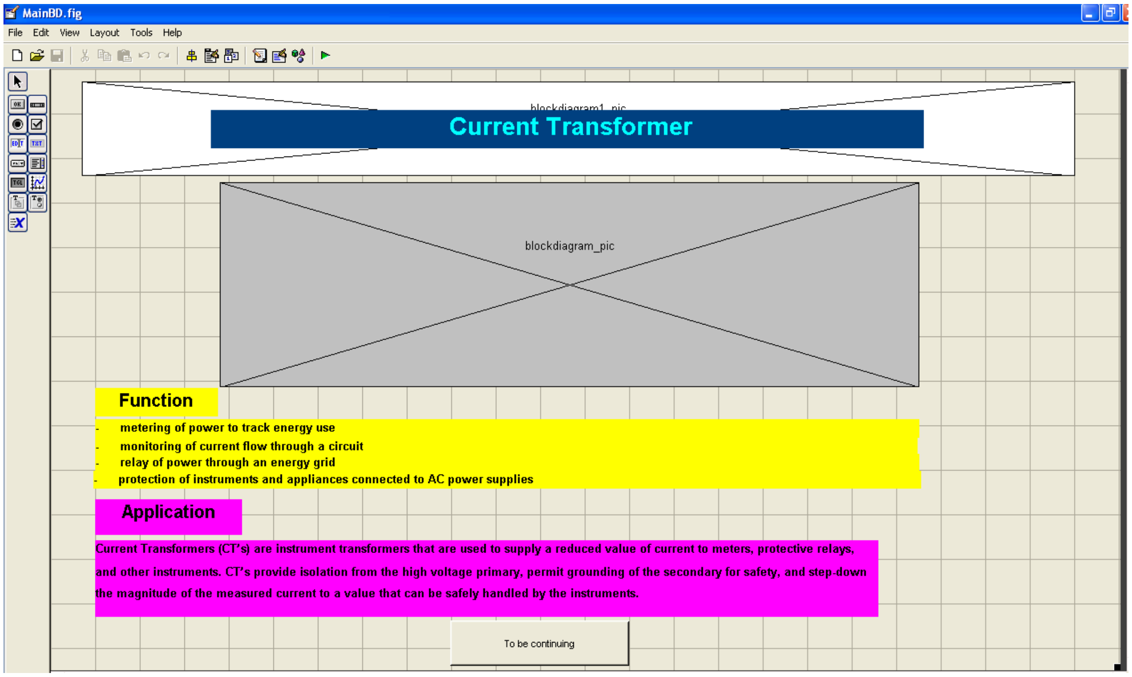


Figure 3.19 The rough layout GUI program for "CT functions and Application"

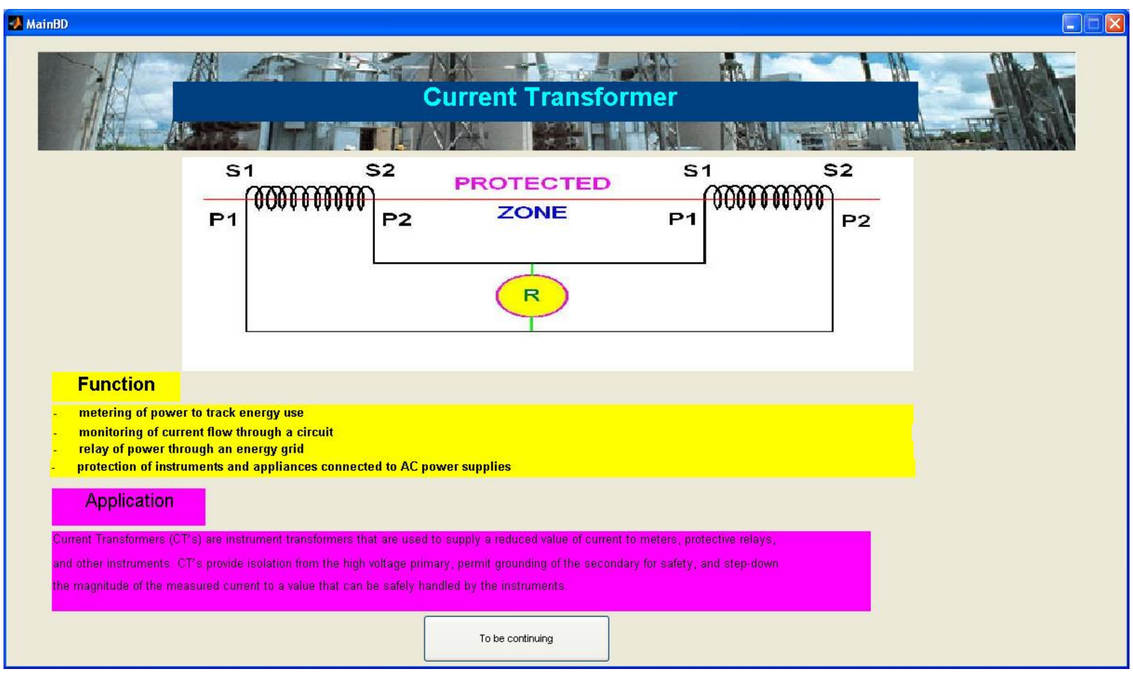


Figure 3.20 The General Layout Of GUI for Program of "CT Function and Application"

Once the user understands the function and application, user press button "to be continued" and Over dimensioning factor will appeared as show in Figure 3.21 while the General layout as shown in Figure 3.22, refer to Appendix B. Based on Figure 3.22, there are two equations which are going to be used to plot the graph in MATLAB.

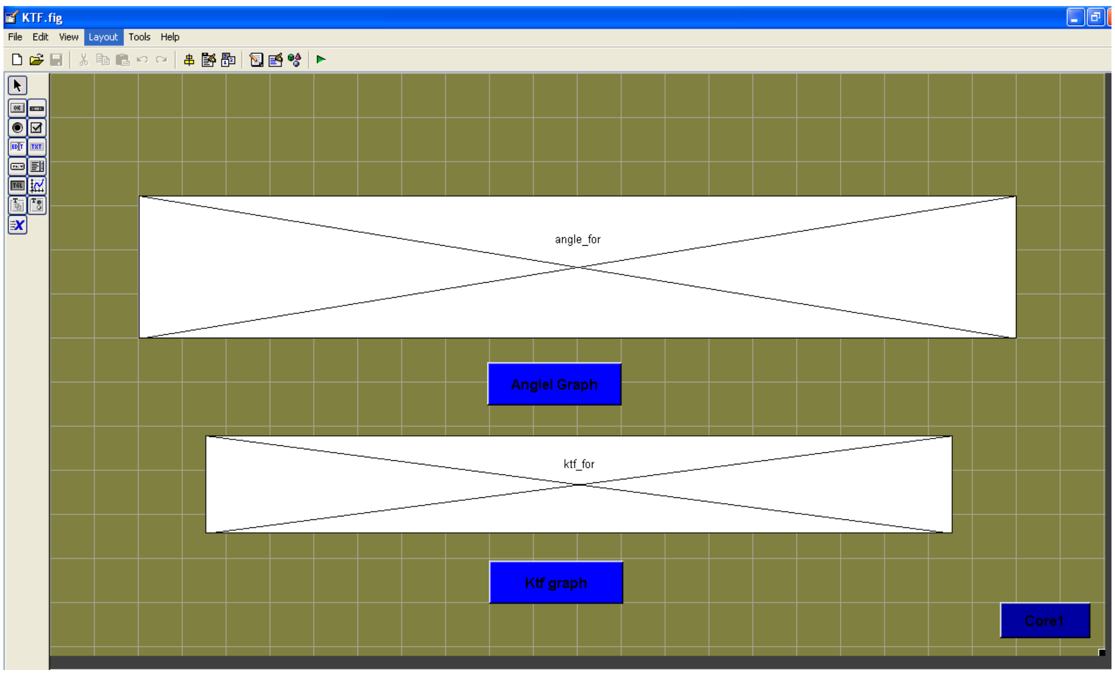


Figure 3.21 The rough layout GUI program for "Over Dimensioning Factor"

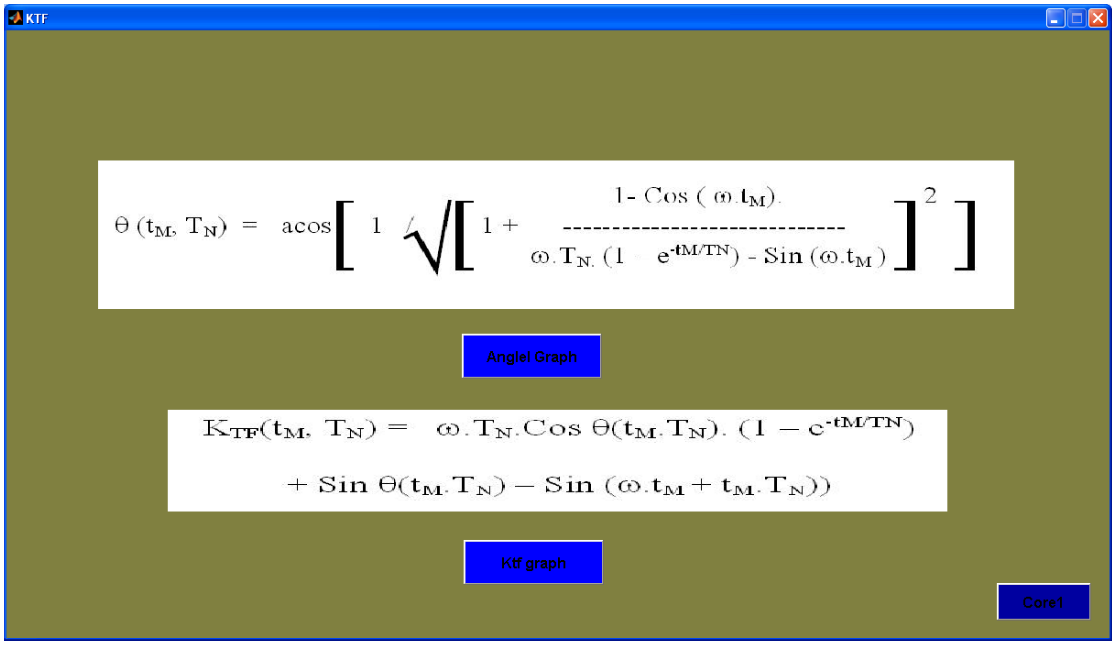


Figure 3.22 The General Layout Of GUI for Program of "Over Dimensioning Factor"

From Figure 3.21 and 3.22, we can find the value of K_{TF}. The K_{TF} value will use as input for Core 1, Core 2, Core 3, Core 4 and 5. User will press the button "Core 1" and Rough layout and General complete GUI program will be appear as shown in Figure 3.23 and Figure 3.24, refer to Appendix C. All the input value will enter and the MATLAB will be calculated. The CT is suitable or not suitable by pressing button "ALF". Once the user has simulated and calculated the operation, the user should process the next by pressing button "Core 2" as shown in Figure 3.25, refer Appendix D. Similarly, operation and simulation will take place for core 3, core 4 and 5, and are indicated as shown in Figure 3.26, refer to Appendix E and 3.27, refer to appendix F.

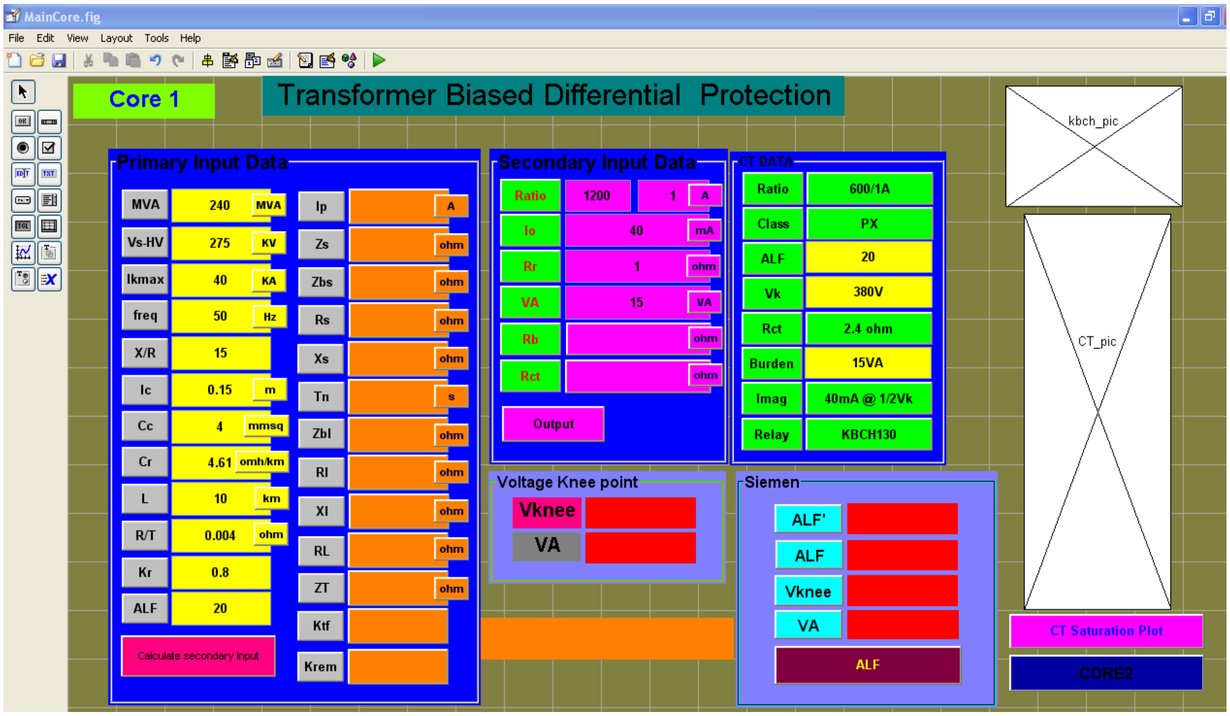


Figure 3.23 The rough layout program of Core 1 “Transformer Biased Differential”

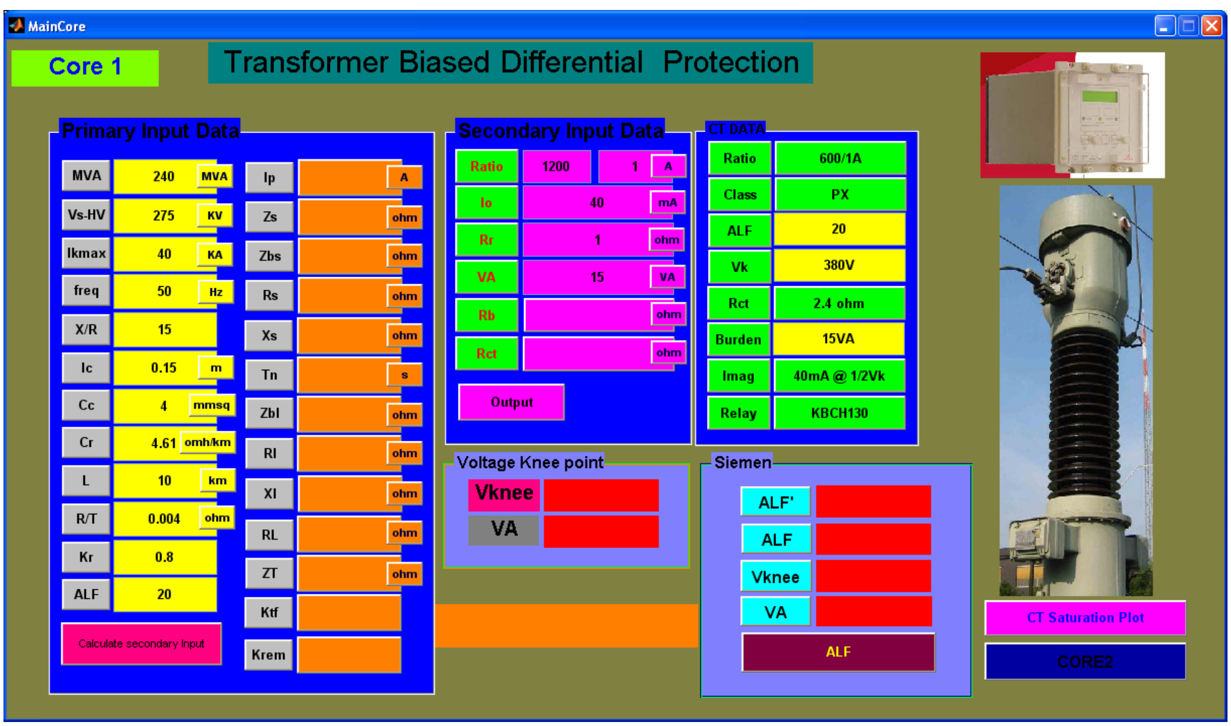


Figure 3.24 The General Layout Of GUI for Program of “Transformer Biases Differential”

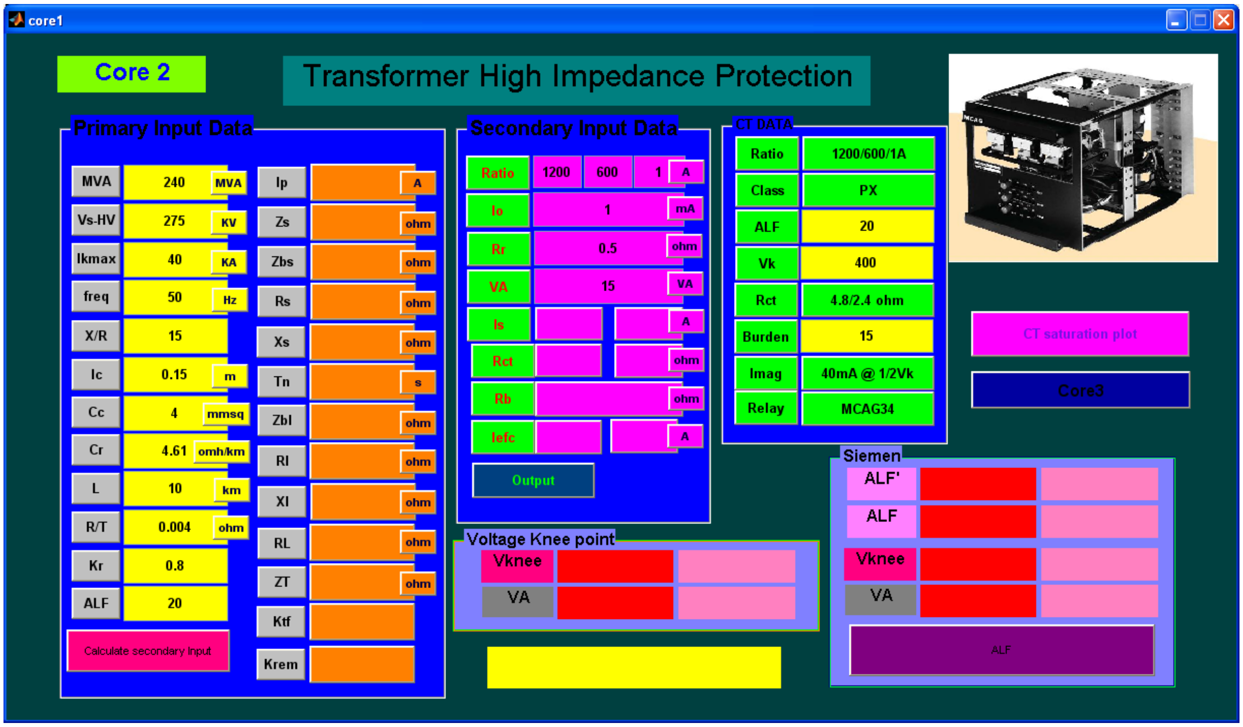


Figure 3.25 The General Layout of GUI for Program of “Transformer high Impedance“

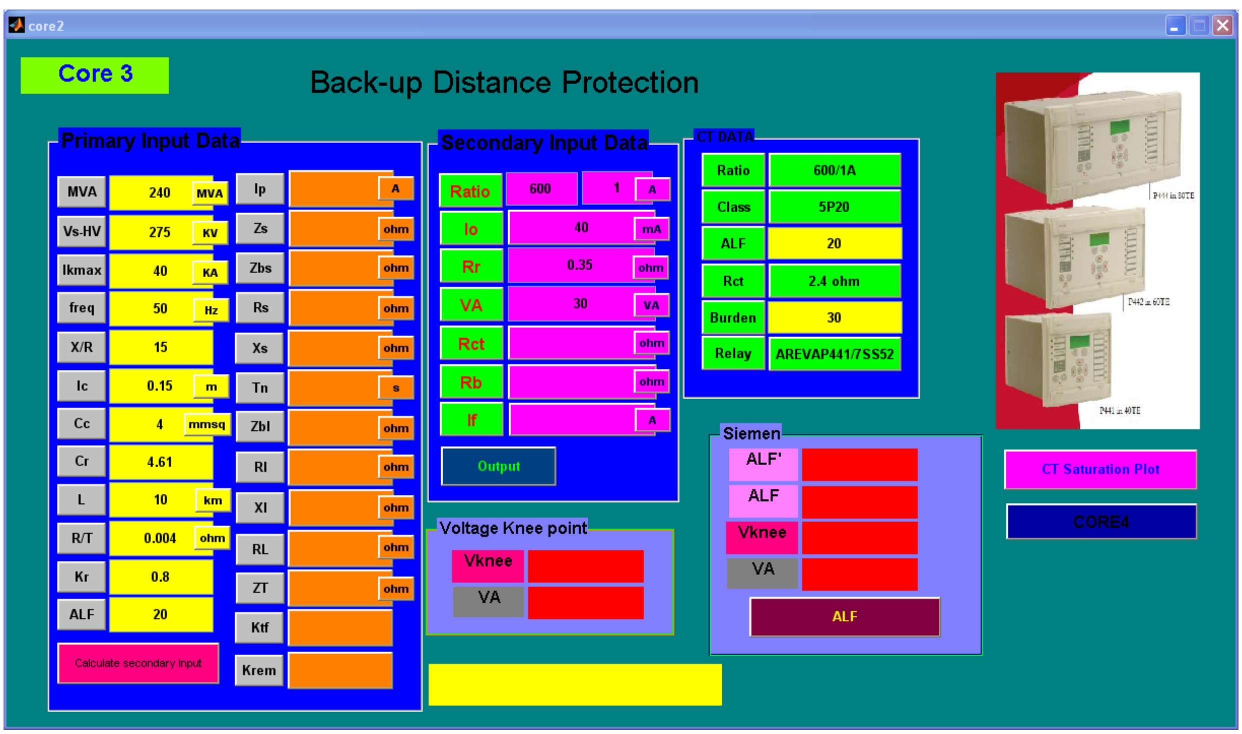


Figure 3.26 The General Layout Of GUI for Program of “Back Up Distance”

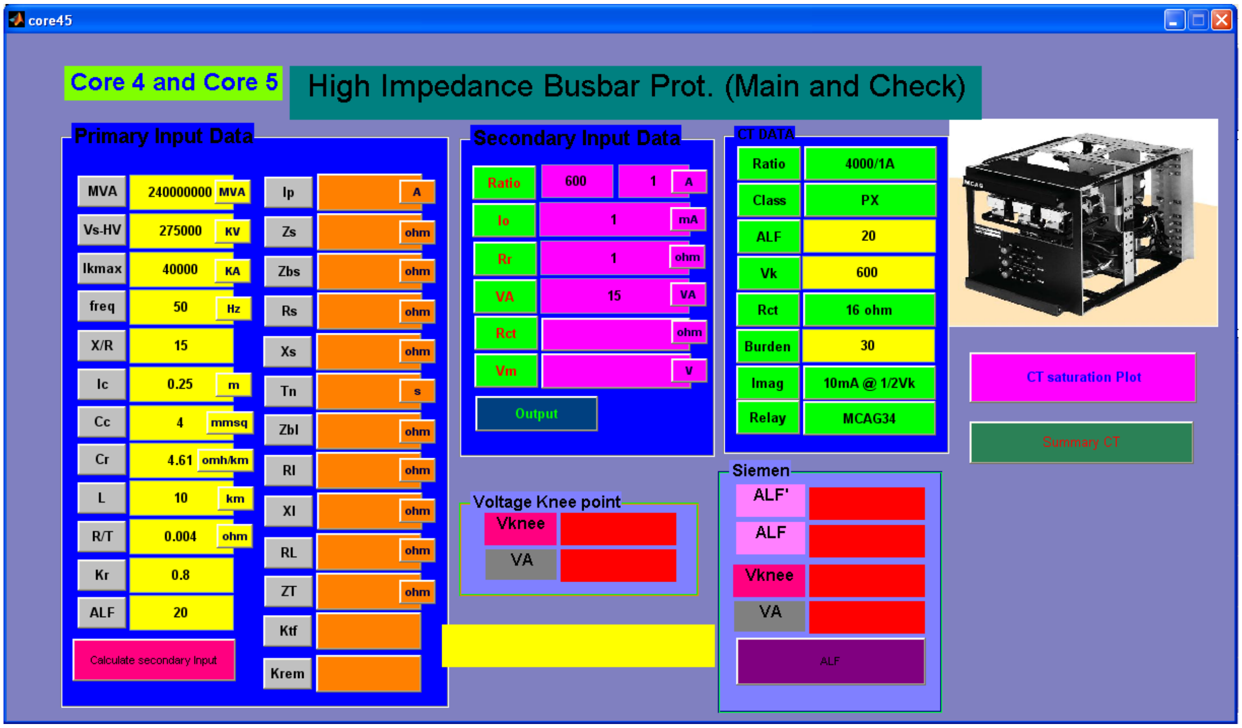


Figure 3.27 The General Layout Of GUI for Program of “High Impedance Busbar Prot. (Main and Check)