AlphaCAM

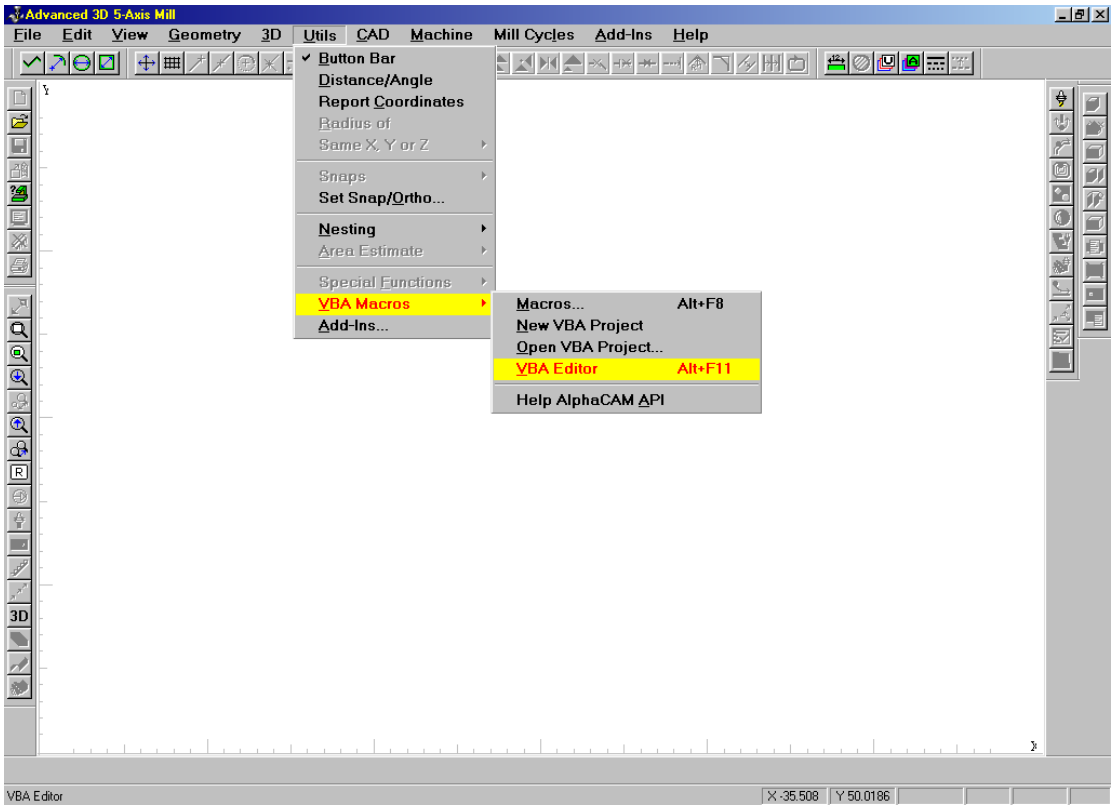
Introduction to

VBA

AlphaCAM Introduction to VBA

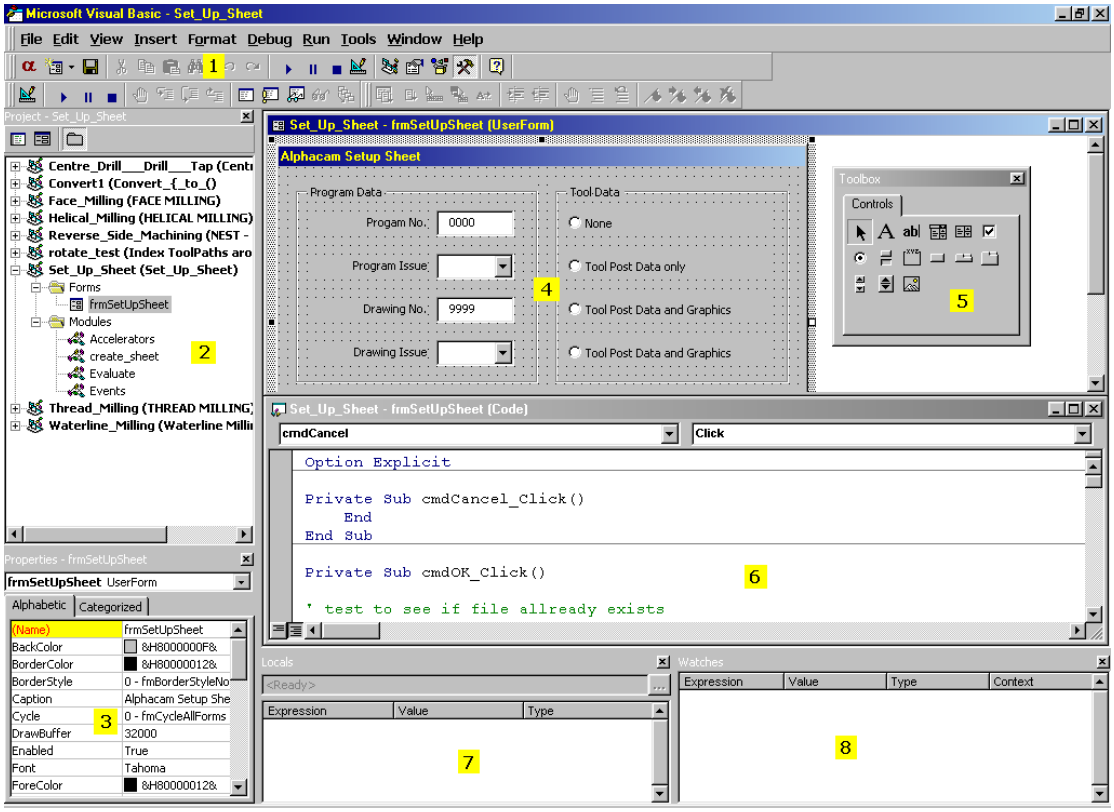
The Interface

The VBA editor is available in all Advanced modules of AlphaCAM. The VBA interface can be accessed from the Utils pull down menu as shown below



AlphaCAM Introduction to VBA

The VBA interface is shown below



1. VBA menu and toolbars

2. **Project Explorer Window** ~ displays a list of the currently loaded projects

3. **Properties Window** ~ displays a list of properties for the active control

4. **Userform Window** ~ used to design and modify dialog boxes for the active project

5. **Toolbox Window** ~ used to add new controls to the active form i.e. command buttons, labels, text boxes, etc….

6. **Code Window** ~ the program for the VBA macro is written in this window as a series of subroutines and functions

7. **Local Watch** ~ displays values for all local variables at runtime, used in debugging

8. **Watch Window** ~ displays values for specified variable at runtime, used for debugging

AlphaCAM Introduction to VBA

VBA Directory Structure

The following structure has been established so that AlphaCam and programmers can find their VBA projects easily.

There are two locations available for saving, the first is primarily for projects written by Licom Systems and the second is for projects written by customers.

Licom Projects

Alph99\
- Add-ins\ — *Licom DLL's selected with UTILS | Add-ins*
  - Autotas
  - Microscribe
  - Swsketch
- StartUp\ — *Licom VB projects selected with UTILS | Add-ins*
- SysMacro\ — *Licom VB projects loaded up when AlphaCam is initialised*

Customer Projects

Licomdir\
- VBMacros\ — *Customer VBA projects selected with UTILS | VBA .. | Open VBA project*
  - StartUp\ — *Customer VBA projects loaded up when AlphaCam is initialised*

VBA Extensions

MacroName.aab	*AlphaEdit VBA project*
MacroName.amb	*AlphaCAM milling VBA project*
MacroName.arb	*AlphaCAM routing VBA project*
MacroName.atb	*AlphaCAM turning VBA project*
MacroName.aeb	*AlphaCAM wire EDM VBA project*
MacroName.alb	*AlphaCAM laser VBA project*
MacroName.afb	*AlphaCAM flame VBA project*
MacroName.apb	*AlphaCAM punch VBA project*
MacroName.asb	*AlphaCAM marble VBA project*
FormName.frm	*Exported VBA form*
FormName.frx	*Exported VBA form additional information*
ModuleName.bas	*Exported VBA module*

VBA Macro Structure

A VBA Macro is known as a project and can be split into 2 sections

1. UserForms

Forms are essentially dialog boxes used to obtain information from the user executing the macro. This information is passed into the module. A project can contain more than one form.

Forms contain a series of controls which are placed on the form at design time by the programmer using the toolbox. A control can be divided into 3 sections.

(i) **Properties** ~ these define the appearance of each control. Some of these can only be set at design time, and some of can be modified at runtime. The most important property is the name property, it is essential that this is set to a meaningful name so that it can be referenced at any time later in the code.

Recommended VBA prefixes for control names

Cmb	Combo box	**Lbl**	Label
Chk	Check box	**Lst**	List box
Cmd	Command button	**Opt**	Option button
Fra	Frame	**Pic**	Picture box
Frm	Form	**Txt**	Text box
Img	Image box		

(ii) **Methods** ~ these define the behaviour of each control. Example methods are Move Method (Moves a form or control, or moves all the controls in the Controls collection) , SetFocus Method (Moves the focus to this instance of an object) and Zorder Method (Places the object at the front or back of the Z order).

(iii) **Events** ~ these define the actions performed by the control i.e. mouse events. Events are small subroutines which are performed when the control is selected. Examples of events are Click Event (performed code when the user clicks on the control with the mouse), BeforeUpdate Events (perform code before the display of the control is updated) and Change Event (perform code when the value of the control changes).

2. Modules

Modules contain the main bulk of the code and are split into subroutines and functions. This is done so that the programmer can reuse code in different userforms without having to rewrite it. A project can contain more than one module.

A subroutine is a program which performs a set task without returning a value.

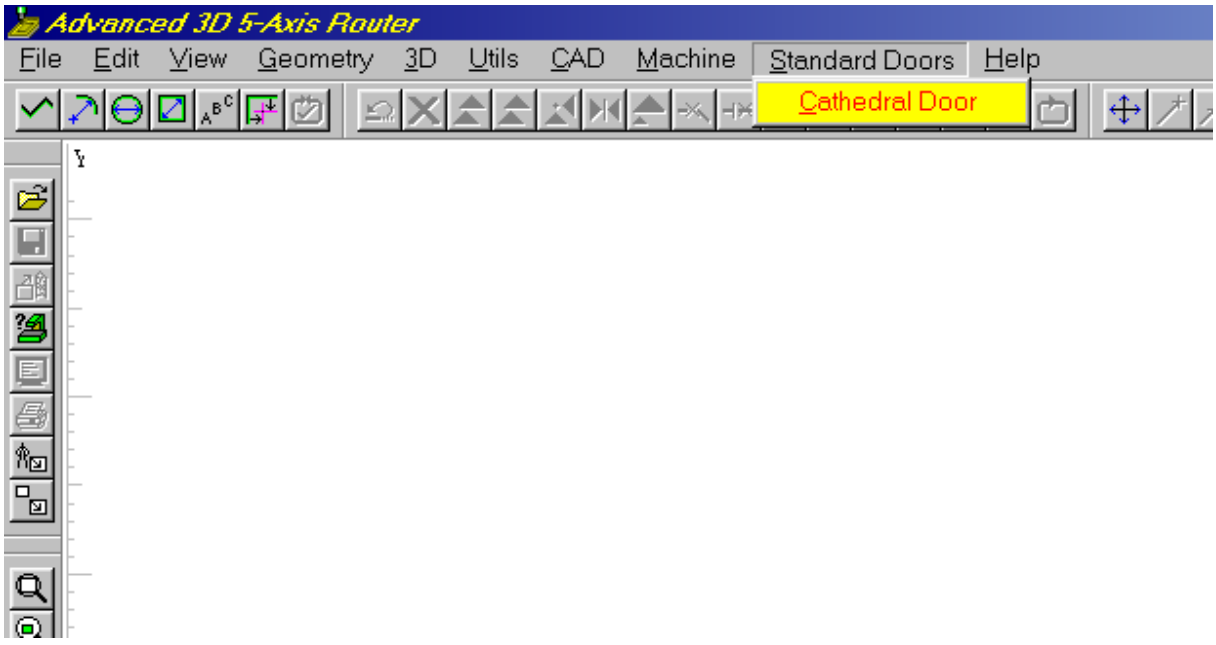
A function is a program which performs a set task and can return one or more values for use in another part of the project.

An important module is the Events module. This module is used when the programmer wishes to add a new item to one of the AlphaCAM menus. Below is an example of an events module to add a new menu.

```
Option Explicit

Function InitAlphacamAddIn(acamversion As Long) As Integer
   Dim fr As Frame
   Set fr = App.Frame
   Dim PopupName As String, ItemName As String, MenuName As String
   MenuName = "Standard Doors": ItemName = "Cathedral Door"
  With fr
          .AddMenuItem2 ItemName, "ShowfrmMain", acamMenuNEW, MenuName
  End with
  InitAlphacamAddIn = 0
End Function

Sub ShowForm()
   Load frmMain
   frmMain.Show
End Sub
```



AlphaCAM Introduction to VBA

General Layout and Styles

Naming Conventions

~ Do not include any spaces in the name of the project
~ Set name for project and text file the same

Text

~ MS Sans Serif 8 points

Control Dimensions

~ If a form is a replacement for an AlphaCAM form make it look the same as the original form
~ Textbox Height 18 points, Width 55 points
~ Label Height 12 points
~ Checkbox Height 15 points
~ Option Button Height 15 points
~ Command Button Height 18 points, Width 55 points

Alignment

~ Textboxes and labels align middles
~ Labels, set text right justified
~ Textboxes, set text left justified
~ Set selection margin to False

Modules

~ Always import the Accelerators and Evaluate modules

The Accelerators module is used when translating the text file into other languages, and is used at the end of a form initialise event by typing ***SetAccelerators Me.***

The Evaluate module is when you want to ensure the contents of a text box only contains numeric values and will also enable the text box to perform full mathematical functionality. Its use is shown below for a textbox named textbox1.

```
Private Sub textbox1_BeforeUpdate(ByVal Cancel As MSForms.ReturnBoolean)
    TextBoxCalculate textbox1, Cancel
End Sub
```