Enigma Version 3.1

Check out the Interface

Enigma is a 32 bit shareware registration and source code generation program for visual basic developers.

It offers three unique levels (types) of shareware registration protection, from simple numeric codes up to nine characters in length, to complex alpha numeric codes near 20 characters in length. Enigma stores information that you enter about each project. When your customer information is entered, it is encrypted with the project information to generate individual registration codes that will only work for that program. Level three is unique in that it thwarts a bogus registration process before the actual registration information is entered. It does this by reading an alpha numeric indicator called a "kickout" which is unique to each project. If the kickout is wrong, the registration is stopped.

You can write all your programs information to a file or the system registry simply by clicking either one on the interface and entering a path.

Easily cripple and/or date limit your shareware.

Saves all important information about each projectfor future use. It cannot be accidentally altered or destroyed. You simply click a button and the information is ready to be pasted into a project.

Pastes all the registration information to your customers email response to avoid inadvertant errors in transfering information.

Comments & Bug Reporting

Comments

Even with all the testing that gets done on software, it is always possible that something can still go wrong. Some of the software is currently in beta so any comments, with the exception of meaningless flames are encouraged and appreciated. The goal is to create a product that is easy enough for the everyday user to understand yet powerful enough for everyone. Please let me know if there is a feature that is not in the current version that you would like to see in future versions, a feature that is in this version that you would like to see expanded or one that should be dropped entirely.

Bug Reporting

If you find the software performing inconsistantly, irradically or just plain doing something it shouldn't do, drop me a line as soon as you can. Here are a few things you should try to include in your note to make fixing the problem easier. The more information you can provide the better.

1. If possible try to outline the steps you performed to cause the problem.
2. Obviously what the problem was.
3. Any error messages and numbers that may have popped up.
4. What other programs that may have been running, or any background programs.
5. The type of computer system you are using, Operating system if not Windows 95 processor speed etc.
6. Anything else you can think of that may be of help in correcting the problem.

Copyrights & Permissions

Credit Card Registration

IMPORTANT

NorthStar processess registrations only, please contact the author for any product/technical support.

E-mailed and Faxed registrations are encouraged, but all registrations are very much appreciated

For technical support or comments about this program, you may contact me at:

armor@webwrite.com

or

PO Box 32706-184
Tucson, AZ 85750-2706

Please note Northstar Solutions does not provide this information to registrants since

We have no way to identify what information you may have provided to us that is confidential and not intended for public use.

We can not be an answering service for non-orders (neither time nor money would permit us to).

For your convenience we have contracted another company, NorthStar Solutions, to process any orders you may wish to place with your Visa, MasterCard, or Discover card. NorthStar Solutions can be easily contacted **for orders only** via any of the following methods.

Phone Numbers

Calls are taken 10 am - 8 pm EST, Monday thru Saturday.

1-800-699-6395 (Calls from US only)
1-803-699-6395

Fax Orders

1-803-699-5465 (Available 24 hours. International and business orders encouraged)

Internet Orders

Simply fill out the online order form at:

http://ourworld.compuserve.com/homepages/starmail

Email Orders

America Online : **starmail**
Compuserve : **71561,2751**
Internet :**71561.2751@compuserve.com**

Please provide (or be prepared to provide) the following information when ordering:

The program you are registering
Your mailing address
Your Visa, Mastercard or Discover card number and its expiration date (if using credit card)
Your drive type (if other than 3.5")
Your email address (so NorthStar Solutions can send you an E-Mail confirming your order and so I can contact you easily with
any important follow-up information, ugrade announcements etc.).

Mail Orders (Author's Address)

You may register with a check or money order (U.S. currency).

Make them payable to:

Mike Dzicek
PO Box 32706-184
Tucson, AZ 85750-2706

Include:

Name
Mailing Address
Email address
Program name and version
Where you downloaded it

Enigma Registration

Registration is by check, money order or credit card. If paying by check, mail it to the address below. If paying by credit card, see the Credit Card.Registration.

Please do not mail checks or money orders to Northstar Solutions or contact them for technical support, they are for credit card registrations only !!

Mail your registrations to:

Mike Dzicek
P.O. Box 32706-184
Tucson, AZ 85750-2706

In addition to the $22.95 registration fee, please include:

Name
Mailing address
Email address
Name of product and version
Where you download it.

General Information

Enigma Version 3.1

Program Name	**:**	**Enigma**
Version	**:**	**3.1**
Program Last Update	**:**	**December 1, 1996**
Original Release Date	**:**	**April 17, 1996**
Price	**:**	**$22.95**
Operating System	**:**	**Windows 95***
Design Resolution	**:**	**800 X 600****
Distribution Catagory	**:**	**Shareware, Freely Distribute by all channels**
Author	**:**	**Mike Dzicek**
Organization Name	**:**	**Savant Guarde**
Snail Mail	**:**	**PO Box 32706-184 Tucson, AZ 85750-2706**
Email Address	**:**	**armor@webwrite.com**
Page URL	**:**	**http://webwrite.com/savantguarde/enigma/enigma.htm**
File URL	**:**	**http://webwrite.com/savantguarde/enigma/enigma31.zip**
Support URL	**:**	**http://webwrite.com/savantguarde/vbasic/support.htm**
Company Web Site	**:**	**http://webwrite.com/savantguarde/**

---

*This program was written and designed on Windows 95, It may or may not work an Windows NT or some other 32 bit Operating Systems. If you use it on an Operating System other than Windows 95 I make no guarantees about performance.

**The screen resolution for development was 800 X 600 and was chosen because it is the average setting by users. The program should be usuable with other settings, but if you notice some fonts difficult to read, that is probably the reason.

Installation

Installation from setup.

Simply unzip the contents into a temporary directory and run the setup.

You must be running Windows 95.

Installation from an update.

Unzip the update into the programs directory.

When asked if you would like to replace the existing exe file, select "yes"

Product Support

Support is currently limited to email. In some cases I can be reached by phone. Barring any unforseen you will have a response within 12 to 24 hours except (maybe) on weekends.

Each program comes with a Readme.txt and Help file that will outline any specific information about the software. Please consult those two files, as well as the information onthe support page before contacting me. Most of the questions I receive are fully covered in the product documentation.

Support page URL: http://webwrite.com/savantguarde/vbasic/support.htm

This software was written and designed for Windows 95, It may or may not work an Windows NT or some other 32 bit Operating Systems. If you use it on an Operating System other than Windows 95, I make no guarantees about performance.

The screen resolution for development was 800 X 600 and was chosen because it is the average setting by users. The program should be usuable with other settings, but if you notice some fonts difficult to read, that is probably the reason.

Full support is offered on all software and all means will be exhausted to resolve any problems you may encounter, within reason. If you use the software in a manner not intended, on an unsupported operating system or something of that nature, I will do my best to help you resolve the problem. You must understand that such support may go beyond the scope and I reserve the right to terminate support at any time.

Occaisionally software will not work on certain systems even if everything is done correctly. As unfortunate as this may be, I have limited resources and the above condition applies.

Trouble Shooting

For the sake of download time and distribution size, some of the files that may be necessary to run this software have been intentionally left out of this zip. This wasn't a haphazard decision, most systems will already have these files, or most of them. Including them would double the download time. So for those of you that already have them, I am sure you appreciate the reduction in size, for those that don't the problem is easily resolved.

This a list of the missing files.

VB40032.dll
VEN2232.dll
OLEPRO32.olb
MSVCRT20.dll
MSVCRT40.dll
CTL3D32.dll

If you encounter any errors pertaining to these files, I have them zipped up at my web page.

The URL is:

http://webwrite.com/savantguarde/vbasic/runtime.zip

If you downloaded this from msn, you can get them in the "Shareware Tools Library". If you downloaded this from winsite or some other archive, they too have these files zipped up, look for vb40032.zip or something of that nature. Just unzip them all into your Windows\System directory. The only exception may be the CTL3D32.dll, which may have to be placed in the Enigma directory.

Common Errors

Below are some common errors that may occur during the installation process. The install will be aborted, a screen will display with the cause of the error and all installed files will be removed.

•Error: Install cannot find the st4unst.00x file.
•Error: Install cannot find the st4unst.exe file.

•Solution: The above errors can be caused by either old runtime files or lack of them. In either case it is recommended that you download the most current files.

•Error: Install cannot find vb40032.dll or one of the files listed above.

•Solution: This is caused by the runtime files not being on your system. Download the runtime.zip and follow the instructions in the Readme.txt.

Warranties & Disclaimers

THIS SOFTWARE AND THE ACCOMPANYING FILES ARE SOLD "AS IS" AND WITHOUT WARRANTIES AS TO PERFORMANCE OF MERCHANTABILITY OR ANY OTHER WARRANTIES WHETHER EXPRESSED OR IMPLIED.

Common sense dictates that any program be thoroughly tested with non-critical data before relying on it. The user must assume the entire risk of using the program. Any liability of the seller will be limited, in the case of non registered shareware, to the support outlined. In the case of the registered program, to support outlined and/or product replacement or purchase price.

PRICES SUBJECT TO CHANGE WITHOUT NOTICE.

Code For Registration Screen

The form code that you need to work with Enigma is very simple. For Levels 1 & 2 you only need 2 text boxes and 1 command button, For Level 3 you need 3 text boxes and 1 command button. You are not limited to one button, I normally use 3, 1 to register, 1 to continue and 1 to exit. Only one is necessary for the registration process, so I will only deal with that. If you don't feel like dealing with coding the forms, I have included two sample registration screens. 1 is for levels 1 &2, the other is for level 3. Customize them as you wish.

Levels 1 & 2

This is sample code for the command button.

```
Private Sub Command1_Click()

Dim stext1 As String
Dim stext2 As Variant

stext1 = Text1.Text
stext2 = Text2.Text

    If Jumbler(stext1, stext2) = False Then

        MsgBox "Registration Information Is Invalid", 48, "Shareware"

        Text1.Text = ""
        Text2.Text = ""
        Text1.SetFocus

    Else

        MsgBox "Thank You, Registration Accepted", , "Shareware"


            WriteIt stext1, stext2
            benable = True
            Unload Me

            Form2.Show

    End If

End Sub
```

Level 3

This is sample code for the "Kickout", the rest of the code is the same as that of Levels 1 & 2.

```
txtKickOut_Change()

Dim LenText As Integer

        LenText = Len(txtKickOut.Text)

        If LenText = 3 Then

            If Kickout(txtKickOut.Text) = False Then


                txtKickOut.Text = ""
                txtKickOut.SetFocus

            Else

                Text2.SetFocus

            End If
            End If
```

It is important to note that it is possible to generate a two digit "Kickout", it is not likely unless Your Information is one character long. In most cases it will be 3 digits. Not to worry, Enigma prints the kickout before the registration number when you generate it. The Kickout will not change once it is generated, as the source of it is Your Information.

Crippling

This is accomplished with a variable called benable. This variable is set to false and is not changed to true until a valid registration is accepted. You use the state of the benable variable to enable the controls or keep them disabled. Obviously some programs will be more complicated than others and the approach taken below will not suffice. For simple programs use something along these lines.

```
Form1_Load()

        If benable = True Then

            Command1.Enabled = True
            Command2.Enabled = True
            Form1.Caption = "My APP"

        End If
```

For more complicated programs you can include a Select Case or IF Then in the events for the perticular controls, that will dictate the state of the controls depending on the benable variable. This is similiar to what I did to cripple Enigma

Date Limiting

Date limiting is a good way to let the user get the feel of your program and still register. The only drawback to it is that all the end user has to do to continue using it is delete the file that holds the information about the date of creation. For this reason I have made no effort to encrypt the contents of the date file. You can ship a file with your program that it needs in order to run, but if the user knows the location, they simply delete and/or reinstall. The best way to safeguard your program is to place the file somewhere the user won't look, and name it something that may not be related to the name of your program. Be careful when naming your file, especially if you put it in the windows or windows system directory, you wouldn't want to write over some important file of the same name, like win.ini.

The variations of date limiting are endless and I won't get into them, but suffice to say there were too many for me to configure Enigma to handle them all, and if I did, someone would want to do it some way I didn't think of. There are a couple of pieces of information in the code that will help you set up your date limiting. The first is a variable called "***YourDayCount***", which is the number of total days you want your program to run. The second is a function called "***DateReadIt***", which is described in "Functions & Subs", in short it returns the number of days that have passed since the file was created. There are two points in the code where you will need to add your own code to make the date limiting come to life. Both are in Sub Main(), the first is right after the line "If Not FileExists(TheFileName) Then", it is commented " 'DateReadIt Now". This is there because the first time the program is run there is no file, so obviously it can't read one. So if you have the day count or something to that effect being shown in the caption of a form, you will have to put that here also, or the first time the user runs your program, your caption will be missing. Calling " DateReadIt" here to return the number of days that have passed since the file was created is not necessary as the file was created a nanosecond before, I put that there to grab your attention. So place whatever you want to happen in the spot where the date read it is located:

```
Form1.Label1.Caption = "Thank You For Trying My Widget"
```

The second area is the most important, as it will be read each time the program is run (starting the second time). It is here that you make the determinations about what actions your program will take. I like to use a select case for the number that "DateReadIt" returns, and when it gets beyond the number I like, they only get a message to register, and then get a registration screen only.

```
Select Case DateReadIt(Password2)

Case 1 To 20

        Form1.Label1.Caption = "Thank You For Trying My Widget"
        Form1.Show
```

```
        Case 21 To 30

            Form1.Label1.Caption = "You Have Been This Program For Awhile Now,
You Might Consider Registering"
            Form1.Show

        Case 31 To 40

            Form1.Label1.Caption = "You Are Now In The 10 Day Grace Period, Please
Register"
            Form1.Show

      Case Else

            Form1.Caption = "Evaluation Period Complete, You Must Register To
Continue Using This Program"
            Form1.ContinueUnRegistered.enabled = False
           Form1.Show

   End Select
```

Notice I passed password2 to "DateReadIt", you can use either Password1 or Password2, it doesn't matter they are both the same in the event they hold a date. Look in "Function & Subs" for details.

Function & Subs

I will outline all the functions and what information needs to be passed to them. I didn't include source codes for the forms because most of the code needed simply passes information to these functions. See Form Code for details about what is needed in the forms.

Jumbler "Jumbler <Something1>, <Something2>"

This is the function that verifies all the registration information. When the file is read the information in Password1 and Password2 are passed. It isn't important for you to know the logistics of the function, but you will be passing information to it from your registration form. You must remember that <Something1> is the long text of your customers Registration Information, and <Something2> is the Registration Number. A true or false is returned depending on whether or not the information passed is valid. Just an informational note, When date limiting, the dates are also passed to Jumbler. They fail because the information is exactly the same so obviously the comparision will fail, and when it does it is then passed to the "DateReadIt" function which takes it from there.

FileExists

There is both a registry and a file version of this function. You personally don't have any dealings with it, but it searches for the existance of your file, and returns a true or false depending on the result of that search.

WriteIt "WriteIt <Something1>, <Something2>"

There is a registry and file version of this sub. It creates and writes the end user information to the file. You will passing information to this sub from your registration form. <Something1> and <Something2> are exactly the same for this function as in Jumbler. <Something1> being the long text of Registration Information, and <Something2> being the Registration Number.

DateReadIt "DateReadit <Something1>"

You don't pass anything directly to this function unless you choose to. It returns an integer that represents the elapsed days since the user file was created. The code will read the contents of this file and pass it to Jumbler, when jumbler fails it will then pass it here, and a day count is determined. <Someting1> must be a valid date or else the function returns an error.

Kickout "Kickout <Something1>"

This function returns a true or false depending on the outcome of its comparision. This function is only used in Level 3 encryption. But you will be passing the contents of the

first registrtion text box here before the whole registration process occurs, See Level in Settings. If the value of those 3 digits are incorrect a false is generated and the user is "Kicked" out of the registration process

Error Handling

I have included error handler in all of of the subs and functions. They are designed to handle most errors that will occur. In some cases I allowed the code to patch up the problem but in others the user will get a message box that describes the error, gives its number and prompts them to contact you. The main errors you will be contending with are probably file access errors, where someone has decided to play with the contents of the file and now the program won't run. I felt in that case it would be appropriate to let them wait until you got back to them, after all they were trying to take money from you so a little wait won't kill them. Enigma is setup so it won't create anything (that I could think of, or find in testing) that will create errors when a file is being created. So most errors will occurs after everything is set up and not during or before.

Date Limiting

Date Limit Check Box

Selecting this check box configures the source code for date limiting. If you neglect to enter a number of days in the Days input box Enigma will disregard the date check box

Days Input Box

This is the number of days you would like to limit your shareware. It changes the value of a variable called "***YourDayCount***" to this number. This variable is in the source code in case you have some use for this number. If you use the select case method with regard to your date limiting, this number may not be necessary, but it is there just in case you need it.

File or Registry (Write To)

You get a choice of where you would like to store your end user information. With the exception of date limited software, it makes no difference where the information is stored. No file is generated until the user registers, and once registered the information in the file is the exact same information that the user entered. There is no intelligence to be gleaned by accessing this file, so the choice is up to you how to store it. With regard to date limited software the story is a little different. The source code generates a file with a date that is read each time the user starts the program. Encrypting the file that holds the date is of no value as elimination of that file will just start the date limiting at the beginning anyway. So it is a good idea to put this file somewhere that the user cannot find it.

File Name Or Registry Entry

File Name

This is where you enter the file name and or path where you would like your end user information written. In an effort to eliminate errors, I have removed the ability to write to a specific drive. The code assumes that the drive that your program is installed to is the drive that you want. If you want to write a file in the same directory that your program is in you simply write a file name, "MYapp.ini". If you would like to write to a sub directory of the programs directory, you type the directory and the file name, "MySubDir\Myapp.ini". If you would like to create a new directory you simply type a "\" at the beginning of the path, "\MyNewDir\Myapp.ini". You can create your own directories and subs and go as deep as you want. While testing out the code I took it 20 subdirectories deep and it worked without a hitch. You can also write to pre existing directories like windows or windows\system.

Registry

Writing to the registry is a little different than writing to a file. The registry uses 4 settings, Appname, Section, Key and default. Enigma writes all of these for you will the exception of Appname. Don't confuse the Appname as used here with Project Name, they are two different things. Consider Appname in this case to be the path your file will be written to. The "Section" is set to "Reg", the Keys are set to "Info" and "number" respectfully and the defaults will be where your information is written. The same applies for writing a path with the registry as with a file. You type the name where you would like your settings placed, "Myapp". If you want to write to a "sub" area of Myapp, you simply write it as if you were writing a path, "Myapp\Subsettings\MySettings".

Generating Registration Information

Your Information Input Box

This is the information that is used to generate an encrypted key that uniquely identifies each project. The input box will accept up to 180 characters, and you can enter anything you like. This information is very important and should not be deleted or altered once a program is released. For this reason the information is saved and protected from accidental deletion or alteration.

Registration Information Input Box

This is your customer information. I like to Use Name, address and email address. You can use anything you like. The number is encrypted with the information stored in the "Your Information" Input box to generate a code for a specific program and user. This information is not saved or protected, as it is not important to keep. If one of your customer looses their information it is easier to just generate a new one. This is also limited to 180 characters.

Registration Number

This is the number that your end user will type in to confirm the Registration is valid. It is an encrypted code from the contents of the "Your Information" and "Registration Information" input boxes. For Levels 1 & 2 you will get one number, for level 3 you wll generate two, a kickout and a Registration number. Clicking on this label will paste the information to the clipboard for easy transfer to email.

Level

Enigma comes with three encryption levels. Level One being the lightest and Level Three being the deepeset.

Level one Generates numeric Registration Information up to 10 characters long, depending on the length of the information that you enter into Your Information and Registration Information.

Level Two generates Lengthy alpha numeric Registration Information that can be up to 20 characters long, with the shortest being 7 in length.

Level Three uses the same encryption technique as Level Two except that it uses two text boxes and generates another 2-3 digit number I call a "Kickout". The "Kickout" disallows any attempt to continue the registration process if the contents of the first text box are incorrect. It just adds an addition layer to the process.

Project Name

Project Listing

The project listing simply lists the name of all projects you have saved settings for. You enter the project name in the "***Project Name***" Input box. This name has no relation on the source code, it is simply used for you to identify which project is which

Project Name List Box

The name that identifies one saved setting from another. Choose a name that will uniquely identify each project. If you have updated a project be sure to use a name that will differentiate it from the older version, if they are still being used. Using the same name twice will just overwrite the old settings with the new ones.

Source Code and Registration Number Generation

Calculate Button

The Calculate Button will generate all the necessary registration information for a specific project. All fields must be complete or the button will not be enabled.

Clear Button

The Clear Button only clears the Registration Information Input Box. If you are generating more than one registration number, hitting the calculate button will highlight the contents of the Registration Information Input Box so you can just enter the next customers information.

Source Button

The Source Button prints the source code for a project to the clipboard. This button will not be enabled until all the information is properly entered and calculated. Once all the information is stored the Source button will remain enabled.

Help And Exit Buttons

Self Explanatory

Project Setting Buttons

New Button

The New Button clears the contents of all the input boxes and allows you to start a new project. By default every time Enigma is started you are in the new mode. New does not delete any of the current settings is simply readies the interface for a new project.

Edit Button

The Edit Button removes the safeguards that are in place to protect project settings, and allows them to be altered. It is not recommended that you edit any settings for shareware that has been released, doing so will make the registration process for that program obsolete. If you choose to alter the settings for an already released program, you can calculate numbers and generate source codes five times without fear that the new settings will be saved. The third time you will be asked if you would like to save the current settings. On the sixth you will be notified that the current settings have been saved. This is a safeguard against inadvertantly losing settings that you may wish to keep. It is advised that you experiment with new settings rather that pre-existing ones.

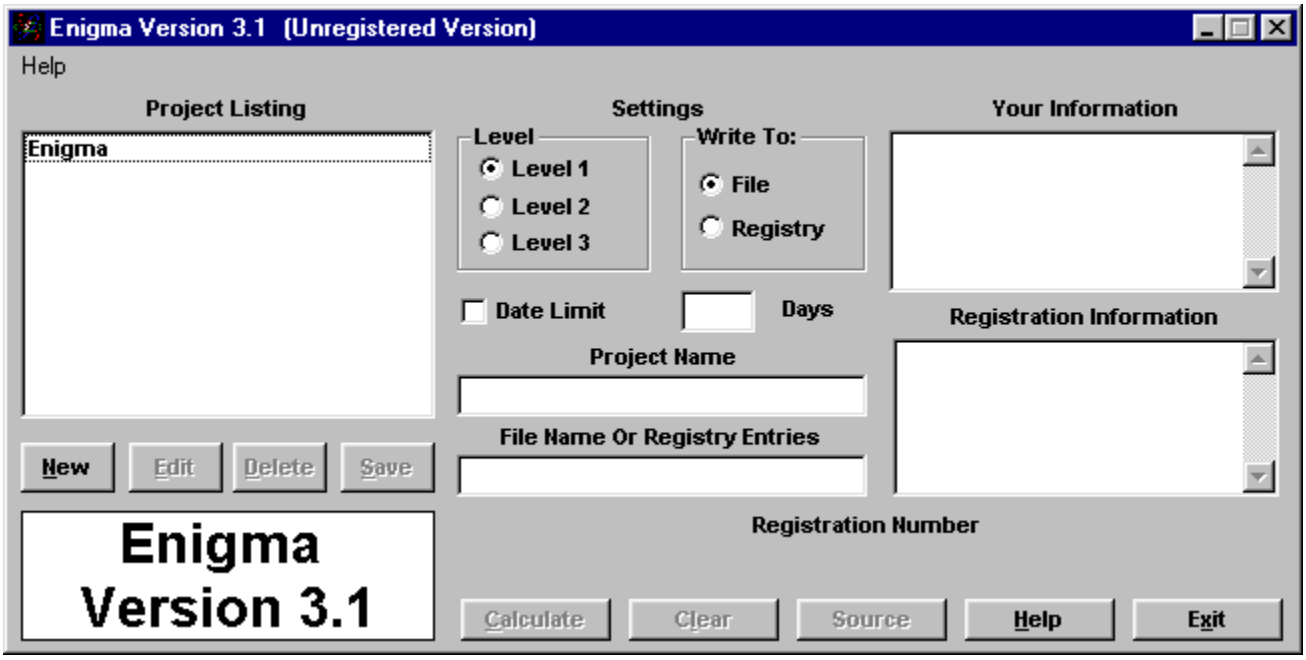
Delete Button

The Delete Button removes all reference to a specific project. It is not recommended that you delete any references to projects that still have a chance of being registered.

Save Button

The Save Button saves any new project settings, or any altered project settings.

The Interface



Click what you need help with.

Overview Of Tutorials

Tutorial 1

This tutorial deals with entering information about a specific project and the controls involved.

Tutorial 2

This tutorial deals with manipulation of information that has already been entered and the controls involved.

Tutorial 3

Putting the code where it needs to go.

Tutorial 4

This tutorial walks you through the process of date limiting your project.

Tutorial 5

This tutorial walks you through the process crippling your project.

Tutorial 6

This tutorial walks you through the process of using date limiting and crippling in one project.

Tutorial 7

Creating a registration form.

It is important to note that for the sake of these tutorials the Enigma interface has been divided into three columns. The first is all the controls under the label ***"Project Listing***", the second is all the controls under ***"Settings"*** and the third is everything under ***"Your Information"***. If you see a reference made to a column, this is the breakdown.

Tutorial 1

This tutorial is for entering information about a new project.

When entering information for a new topic you will only be concerned with the controls in columns 2 and 3. Column 1 is for manipulation of already existing project information.

Starting with column 2 (middle column), you must select a level of encryption for your shareware. For more information on this see **"Level".**

You must then select whether or not you would like your information written to a file or the system registry. It is recommended (by Microsoft) that Windows 95 programs have their configuration settings written to the registry and Windows 3.x to an "ini" file, but the choice is yours. The design of the source code generated by Enigma is such that it doesn't matter where the information is located as the user cannot get any intelligence from it, this does not apply to **date limited** software however.

The source code for crippling shareware is entered into your projects by default, this is not the case with date limiting. If you would like your program to incorporate it, you must check the the "***Date Limit***" check box then add the number of days to the "***Days***" input box. If you don't enter a number of days you will be prompted to do so when you try to calculate the source. If you opt not to at that point the date limiting source code will be dropped from the project. Due to the number of variations to date limiting and crippling, Enigma only inserts the tools to accomplish either of these tasks, you customize your projects to use them as you wish. Tutorials 4, 5 and 6 cover this in detail.

The "***Project Name***" has a two fold purpose. This is the name that will show in the "***Project Listing***" list box. It is also used in the error handling events that are automatically inserted into the source code. This name will not be used in any of your source code, only the source code generated by Enigma.

The "***File Name Or Registry Entries***" is simply the file you would like your settings written, or the location in the system registry. You can write these settings anywhere you like, to any directory depth you like. See **File or Registry** for full details. Simply put, if you would like your settings written to a file called "MyApp.ini" that's what you enter. If you would like them written to the "MyApp" section of the registry, that's what you enter.

"***Your Information***" is used to generate an encryption key for a specific project. Once your program is in distribution, you should leave this alone. If it's lost, so is your ability to register your program. I recommend that you enter something long and not obvious. **Do Not** release this information to anyone. Although it's not likely that your end user will have Enigma or know Enigma was used to create your registration codes, if you use obvious information like the name of your program, you have just increased the chances

of someone "cracking" your registration if they have Enigma, common sense applies.

"***Registration Information***" is your end user information. I like to use Name, Address and email address, but you can use anything you like. It is then encrypted with the key that is generated by "***Your Information***"

"***Registration Number***" is the product of encrypting "***Your Information***" and "***Registration Information***". When the end user registers your program they need to input both the "***Registration Information***" and the "***Registration Number***", so you must include both in your email response. If you click the blue numbers under this label, all the necessary information is written to clipboard for you to insert into your email responses. This reduces the chance that you will copy the information incorrectly and cause the registration to fail.

Once you have all these fields filled in you need to hit the "***Calculate***" button to generate the "***Registration Number***" and source codes. Even the very first time you enter information for a project you must fill in all the fields. This was done for testing purposes. I normally enter a 1 or 2 characters into the "***Registration Information***" field and then save the number for future use.. After you "***Calculate***" successfully, the source is held for easy retrieval. If for some reason you change any of the information, you must calculate the new settings before they will be written to clipboard.

The clear button is used if you have a number of registrations to generate. It only clears the "***Registration Information***" field and doesn't affect any other settings. For ease of use, you don't really have to use this button. Once you calculate a users information it is highlighted, you simply type the new information and the old is overwritten.

The "***Source***" button writes all the source code for the project to the clipboard so you can insert it into your project. You paste this information to a bas file, **Tutorial 3** covers this in depth. This button is not enabled until all the necessary information is entered. It is important to note that if you make any changes to the contents of the interface , you must recalculate before the changes will be reflected in the source code.

The "***Help***" and "***Exit***" buttons are self explanatory.

Tutorial 2

<u>This tutorial deals with altering already existing information.</u>

Once you have saved and/or calculated new information, it is protected from inadvertant alteration or deletion. You may decide you would like to update some of the settings and leave others. We will be dealing with the controls in column one (all controls under the "***Project Listings***" label). Once you save the changes, all the old settings are lost, so think about what you want to do before you change anything. Enigma gives you five chances to play with new settings before it makes them permanent, warning you after the third.

Enigma starts in the new mode, so you can immediately begin entering any information pertaining to a new project. You will notice four buttons under the "***Project Listings***" list box, "***New***", "***Edit***", "***Delete***" and "**Save**". In the new mode all these buttons are disabled except "***New***". If you would like to open the settings for an already existing project you must click the title in the "***Project Listings***" list box, at which point all buttons are enabled with the exception of "***Save***".

Clicking "***New***" simply removes the contents of the interface and readies it to receive information about a new project.

"***Edit***" removes the protection and allows you to manipulate the contents of the fields. Use this option with caution, once the changes are saved there is no getting them back.

"**<u>Delete</u>**" removes and entire project from the listing.

"***Save***" saves all the changes that you have made to an existing project. This button will not be enabled until you "Calculate" the changes. You are given 5 chances to test your new settings before Enigma automatically saves them. If you do not calculate the new settings the source code button will write the **OLD** source to the clipboard, so be sure to calculate and save prior to using the new code.

Tutorial 3

Putting the code where it needs to go.

This is the easiest of the tutorials. Once you have all the information input into the interface, and you have calculated it, it is ready to be inserted into your project. The registration process requires two things, a module and a form (registration screen). Enigma only writes the source code for the module, not the registration screen. Because the registration screen simply passes information to the module, and this only requires a few lines of code, I felt it easier to include sample registration screens, and a tutorial of the process rather than have Enigma generate the code itself.

Insert a module into your project, you can name it anything you like. Remove any code from it, Like "Option Explicit", then paste. You will notice some commented areas in the source code. This is where you will be customizing the code to suit your needs. Tutorials 4,5 and 6 go into great detail about customizing your programs date limiting and crippling capabilities, we will stick to the insertion of the code itself.

Enigma refers to the registration screen as form1. If you named all your forms something other than the visual basic defaults this shouldn't be a problem. Enigma also uses form2 to reference your registered program. This is there for demonstration purposes only. As I have no idea what forms you want to display or not display, form2 was used only to show where a reference to the specified form would go. The use of form1 and 2 is limited to the initial sub and is easy to find. Please replace these these names with the proper ones.

Tutorial 4

Customizing the code for date limiting

Due to the infinite number of things that can be done with regard to date limiting, Enigma doesn't have a cookie cutter formula for doing so. It simply inserts everything you need into your source code, and you customize to your liking. In this tutorial we will go through a few options so you can get the feel of date limiting, then you can let your imagination go wild in your project. If you are curious about the code that does the date limiting look in "**Function & Subs**". For the sake of simplicity this tutorial will only deal with the code necessary to accomplish the goal of date limiting.

All of the source code that needs to be manipulated is in the Sub Main () procedure of the module. You will notice 4 lines of code that read"

```
If Not FileExists(TheFileName) Then

    WriteIt   NOW, NOW

    'DateReadIt Now

    Form1.show
```

The first line determines that a date file does not exist. The second line writes a date file. The third line does nothing as it is commented out to grab your attention. The fourth line shows your registration form.

It is important to note that this code will only be **executed once**. Line 3 was designed to be an attention grabber, and unless you use a "**Continue Unregistered**" option on your registration screen, line 4 is also worthless. The reason I point this out is simple. If you are one of those people that like to show the total days left in the evaluation period, unregistered or registered etc., in the forms caption, you must put that here. If you don't then the first time your user runs your program they will see whatever default settings were set at design time.

The rest of the above code follows (notice this is for the registry and not a file):

```
    Else

On Error GoTo FileError

    GetSetting TheFileName,"Reg", "Info",Password1
    GetSetting TheFileName,"Reg", "Number",Password2

    If Jumbler(password1, password2) = False Then
```

```
        '++++++++++Select Case Or Whatever Code You Choose To Keep Track Of
Days Goes Here++++++++++

        Else

                Form2.show

        End If
        End If
```

This code is executed each time the program is started, starting the second time. The code has found the file and reads the contents. It then passes the contents to "**Jumbler**" which determines if the contents are valid registration entries. If they are not, the first part of the loop is executed (your date limiting options), if they are then the second part is executed and form2 (your registered program) is displayed.

There are two things you can use for date limiting purposes. A variable called "***YourDayCount***" or a function called "**DateReadIt**". Your day count is the total number of days, and "***DateReadIt***" returns the number of days elapsed since the file was created. Once "***Jumbler***" has determined that the contents are not valid registration codes, you pass either of them (Password1 or Password2) to "***DateReadIt***". If it is a valid date all is fine and the number of elapsed days is returned. If it's not a valid date an error is generated and the code is passed to the "**FileError**"error handler, which in the below example shows your registration screen (Form1). If you would like your program to display something else, change the error handling code in your program to reflect what you would like to do..

The below example uses the "DateReadIt" function, form1 as the registration screen and form2 as your program.

```
If Not FileExists(TheFileName) Then      'registration file does not exist

        WriteIt   NOW, NOW                      'writes a date file

                Form2.Caption = "Thank You For Trying My Program" 'these lines set your
options
                Form2.show                                      'that will be seen the
first time
                Exit Sub                                        'your program runs.

        Else                    'this code runs everytime your program runs starting with the second
time.

On Error GoTo FileError     'send the code to the error handling routine.

        GetSetting TheFileName,"Reg", "Info",Password1      'reads the contents of the
registration files
```

```
        GetSetting TheFileName,"Reg", "Number",Password2

        If Jumbler(password1, password2) = False Then          'determine that the contents are
not valid registration codes

                Select Case DateReadIt(Password2) 'sends one of the passwords to the
DateReadIt function
                'if this is a valid date a number is returned, if not this is sent to the error handler
                        Case 1 to 10          'What your program does from days 1 to 10

                                Form2.Caption = "Thank You For Trying My Program"
                                Form2.Show

                        Case 11 to 15         'What your program does from days 11 to 15

                                Form2.Caption ="You Are In the Second Half Of The
Evaluation Period"
                                Form2.Show

                        Case 16 to 25         'What your program does from days 16 to 25

                                Form2.Caption ="Please Consider Registering The Program"
                                Form2.Show

                        Case 26 to 30         'What your program does from days 26 to 30

                                Form2.Caption ="Final Days Of Program Evaluation Period."
                                Form2.Show

                        Case Else             'What your program does if the case is not between 1 to
30

                                Form1.Caption = "You Must Register To Continue Use"
                                Form1.Show      'Registration Screen

                End Select

        Else

        Form2.Caption = "MyApp Version 1.0"    'this code is executed if the registration codes
are valid
        Form2.show

        End If
        End If

        Exit Sub
```

```
FileError:

        Form1.Show          'if there is an error, for instance the date that was passed to the date reading sub, this code
                            'is executed.   In this case it loads the registration screen.
On Error Goto 0

End Sub
```

Tutorial 5

Customizing the code for crippling

The options for crippling are as extensive as those for date limiting. As in the case of date limiting you must customize the code to meet your needs. The backbone of the crippling procedure is a variable called "**benable**" which is initialized as false.

There are two code examples below. The first is the code that will go in the Sub Main () and the Form2_Load or Activate events. The second uses only the Sub Main (). The line in the source code that is important to you is marked with an (8) for this example. If you read through "**Tutorial 4**" this line will be familiar as it is the same line where the action starts for date limiting. The essential difference here is that once the contents of the registration file are determined not to be valid registration information, the process is done. Unlike date limiting where the contents are then read to determine if they are valid dates

At this point it may get a little tricky. The line after the one marked (8) is important because you can place crippling code here in addition to or replacement of the code in the Form_Load. I personally like the Form_Load event and the benable variable, especially for multiple forms. If you are using only a single form with simple crippling, it is possible to accomplish that in the Sub Main () without the Form_Load event or benable variable. Below is an example of each.

<u>This example uses the Sub Main (), benable variable and the Form_Load event.</u>

Sub Main () <snip>

```
If Not FileExists(TheFileName) Then      'determines that this is the first time the program has been run

            Form1.show  'If you use a Continue Unregistered" button you can start with the registration
                        'otherwise you may want to start with Form2 and call the registration
            Exit Sub    'screen from a menu or button.

      Else

On Error GoTo FileError     'sets up error handling

      GetSetting TheFileName,"Reg", "Info",Password1      'reads the contents of the registration file
      GetSetting TheFileName,"Reg", "Number",Password2

(8)   If Jumbler(password1, password2) = False Then      'determines the validity of reg. file contents
```

```
            Form2.show          'if contents are not valid it loads your program, remember
benable
                                'is initialized as false
        Else

            benable = True      'if contents of reg. file are valid.   sets benable to true

            Form2.show          'loads your program

        End If
        End If

        Exit Sub
```

Form2_Load <snip>

```
        If benable = False Then

            Command1.Enabled = False        'disables the controls
            Command2.Enabled = False

        Else

            Command1.Enabled = True         'enables the controls, you get the point
            Command2.Enabled = True

        End If
```

<u>This is an example of crippling in the Sub Main without the Form_Load or benable variable.</u>

<Snip>

```
(8)     If Jumbler(password1, password2) = False Then       'determines the contents of the
files as invalid

            Form2.Command1.Enabled = False      'cripples controls
            Form2.Command2.Enabled = False
            Form2.show          'shows your program, the benable variable is set to false

        Else

            Form2.Command1.Enabled = True       'enables controls
            Form2.Command2.Enabled = True
            Form2.show          'shows your program with controls enabled

        End If
```

End If

Tutorial 6

Date limiting and crippling the same project

This tutorial will pick up where **Tutorials 4** and **Tutorials 5** left off, so if you haven't looked them over please do so now.

This example demonstrates a date limiting and crippling combination. It uses 3 forms, a nag/splash screen (frmSplash), Form1 (your registration screen) and Form2 (your program). In the example I am using both the Sub Main and the Form_Load events. You will notice some redundant Select Case source code. This is done for your benefit. It is easier to use the Sub Main () procedure only, when crippling single form programs. The process gets more complicated when you use multiple forms. With that in mind this example gives you enough code to do either, modify as you wish.

<Snip>

```
If Not FileExists(TheFileName) Then       'determines if the program has been run before

        WriteIt   NOW, NOW                        'writes the date file

        'DateReadIt Now

                frmSplash.Caption = "Thank You For Trying MyApp"    'set nag screen caption
                frmSplash.Show              'shows the nag screen

                Exit Sub

        Else    'this code runs each time the program starts, starting with the second time

On Error GoTo FileError

        GetSetting TheFileName,"Reg", "Info",Password1       'reads the contents of the reg file
        GetSetting TheFileName,"Reg", "Number",Password2

        If Jumbler(password1, password2) = False Then                'determines validity of reg. file contents

        Select Case DateReadIt(Password2)      'determines if the contents is a date or not
        'either Password1 or 2 can be read as they both will be dates, if they are not it goes to the error handler

                Case 1 To 15                'first 15 days

                        Form2.Caption = "First Half Of MyApp Eval Period"     'set form caption
```

```
                Form2.Show
                'crippling can be added here

        Case 16 To 30                   'last 15 days

                Form2.Caption = "Second Half Of MyApp Eval. Period"
                Form2.Show
                'crippling can be added here

    Else            'if reg. file contents are valid

        benable = True

        Form2.show

    End If
    End If

    End Select
```

This is the code for Form2 (your program). Notice the Select Case is similiar to one in the Sub Main ().

```
If benable = false Then

    Select Case DateReadIt(Password2)

        Case 1 To 15

            Command1.Enabled = False
            Command2.Enabled = False

        Case 16 To 30

            Command1.Enabled = False
            Command2.Enabled = False
            Command3.Enabled = False
            Command4.Enabled = False

    End Select

Else
```

'enable all controls that are not enabled. You may want to enable them at design time and only disable them in your code, that way enabling them requires less code. Notice that all this disabling can be accomplished in the Sub Main, it's not so easy for multiform apps.

End if

Tutorial 7

Creating a Registration Form

The source code for the registration forms is very straight forward. If you use levels 1 or 2, source code is only necessary for one control, the enter button. If you use level 3 then source code is required for two controls, the enter button and the "kickout" input box. The coding on any additional controls is not mandatory as long as it doesn't effect their purpose, which is to input data. Included with this program are two sample screens for those who don't want to create their own from scratch. The Form1 and Form2 descriptions apply, so you must change these names in the code to match the names in your project.

The only mandatory controls on the registration forms are two text boxes and one command button for levels 1 and 2 and 3 text boxes and 1 command button for level 3. The samples have "Exit" and "Continue Unregistered" buttons as well. Edit their source code to work with your program.

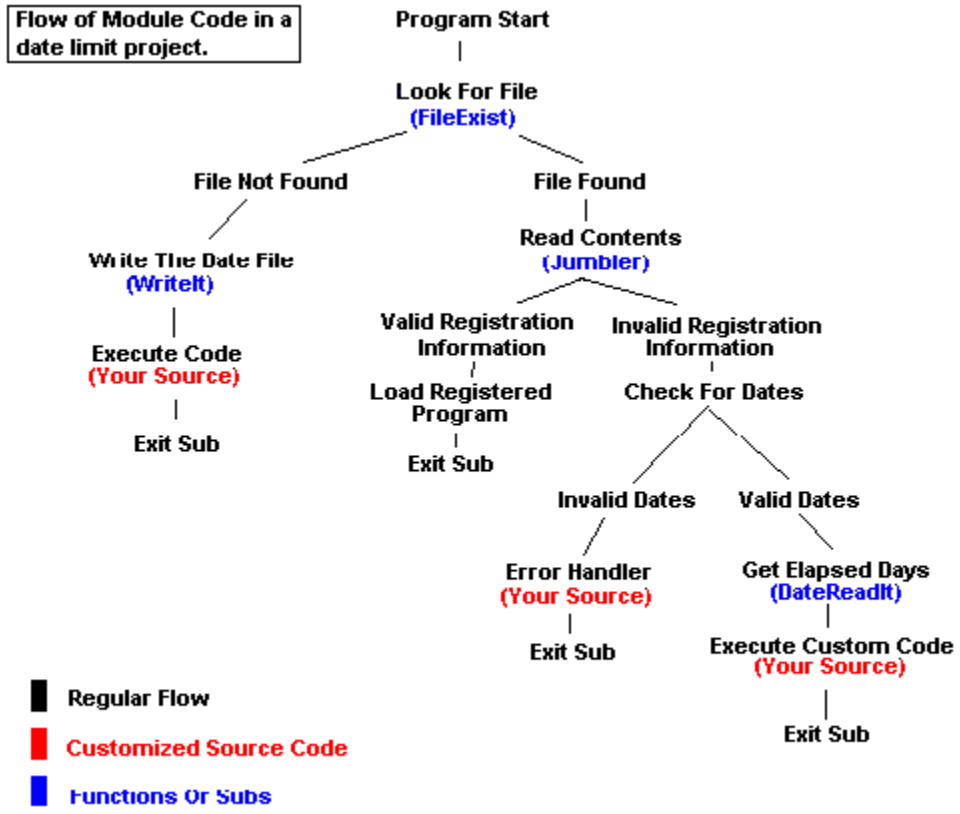
Below is an example of source code for the Enter button . Notice this code uses "Jumbler" the same way as the Sub Main () procedure in the module. That is because the same criteria must be met for the registration to work. This simply passes the contents of the text boxes to "Jumbler" and if they match, the registration is approved, if not the contents are erased and the user must try again. This source code is the same for all levels.

```
If Jumbler(Text1, Text2) = False Then      'passes the contents of the text boxes to Jumbler for verification

        MsgBox "Registration Information Is Invalid", 48, "Shareware"        'failed registration

        Text1.Text = ""              'resets the input boxes
        Text2.Text = ""
        Text1.SetFocus

    Else

        MsgBox "Thank You, Registration Accepted", , "Shareware" 'Passed Registration

            WriteIt stext1, stext2          'saves the contents to be verified each time the program runs
            benable = True                  'allows controls to now be enabled
            Unload Me                       'unloads this form and loads your registered program

            Form2.Show

    End If
```

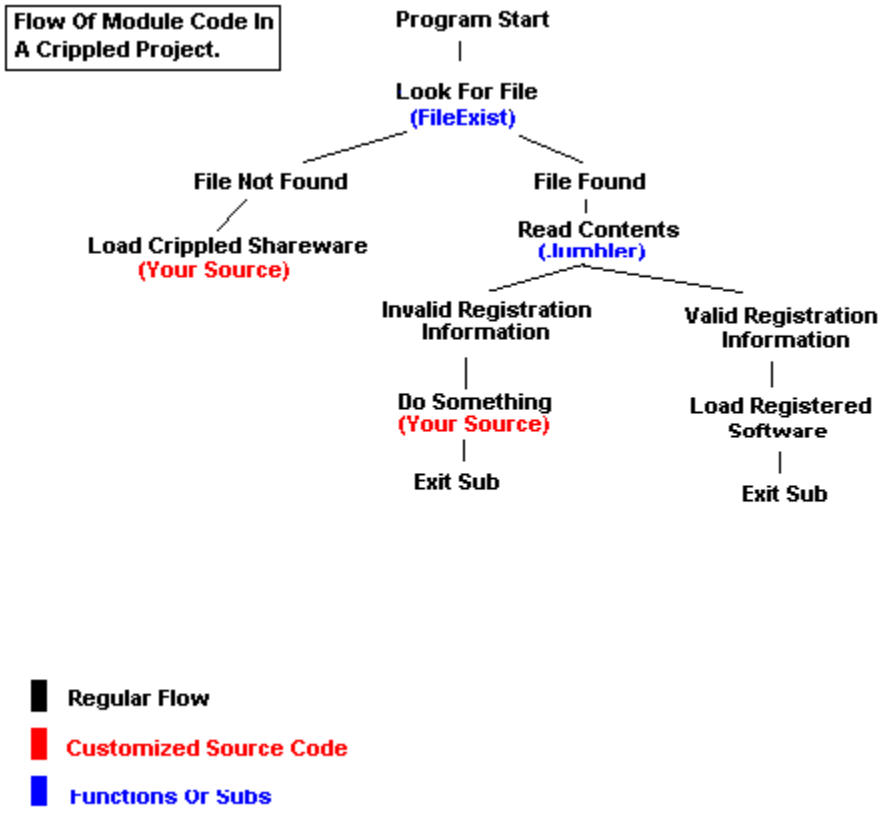
This is the source code for the "kickout". Unlike levels 1 and 2, level 3 uses 2 text boxes to input registration information. The second text box is for a "kickout". Simply put, the "kickout" does a preverification of the registration, if it fails, the user is not allowed to enter the actual registration number. The **txtKickout_Change Event** does all the work. It is important to note that most kickouts will be 3 characters long. If you enter very short information in the "Your Information" input box, it is possible that you can generate a 2 digit kickout. If that is the case you need to change the code in this event to reflect the shorter kickout length. The enclosed sample will not allow focus to be set to the next text box if nothing was entered in the previous one. The samples are rough, you should test their functionality out, and modify them to your needs.

```
Private Sub txtKickOut_Change()

        Dim LenText As Integer

          LenText = Len(txtKickOut.Text)        'measures the length of the kickout text

          If LenText = 3 Then

              If Kickout(txtKickOut.Text) = False Then              'checks for a match

                  txtKickOut.Text = ""  'failed match resets the controls on the form
            Text1.Text = ""
                  Text1..SetFocus

              Else

                  Text2.SetFocus              'passed match moves the focus to the next box

              End If
              End If

End Sub
```

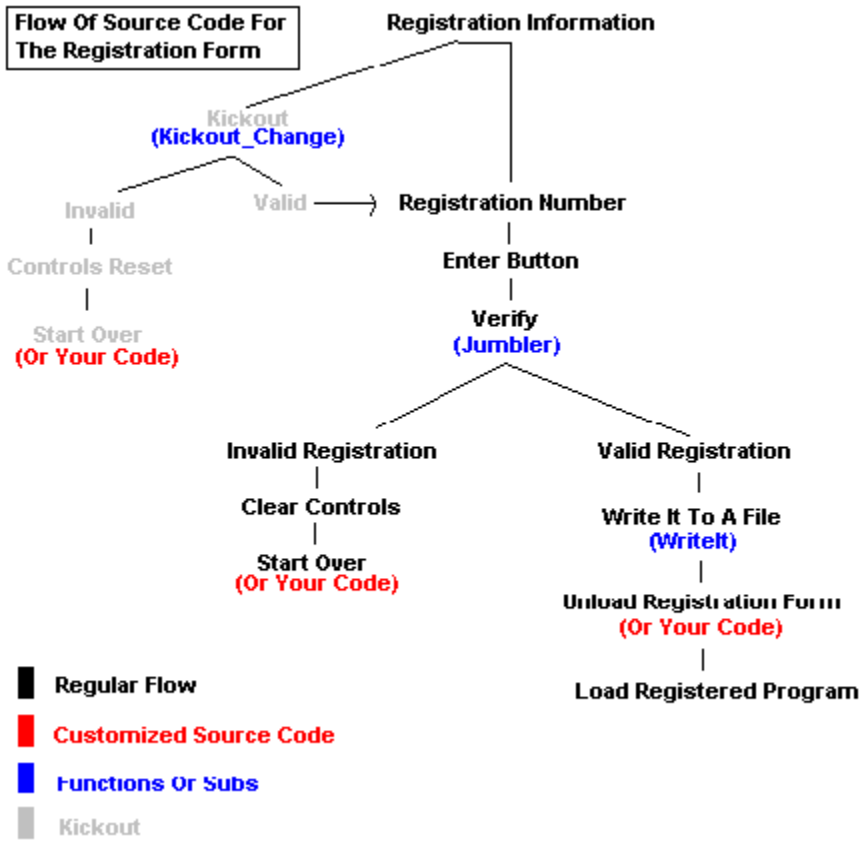
Flowchart 1- Date Limited Project



Flowchart 2 - Crippled Project



Flowchart 3 - Registration Screen



Overview Of Flowcharts

Flowchart 1

This chart shows the execution of source code for a module in a date limited project.

Flowchart 2

This chart shows the execution of source code for a module in a crippled project.

Flowchart 3

This chart shows the execution of source code for the registration screen.

There is no flowchart for a project that is using both date limiting and crippling. Flowchart 1 most closely resembles that case.

The charts are color coded for easy reference. The flow of the code is set but what the code does is not. For instance, if at some point you do not want your registration screen to display, but the flowchart shows that happening, simply change the source to do what you like. Anything coded in **red** can be changed.