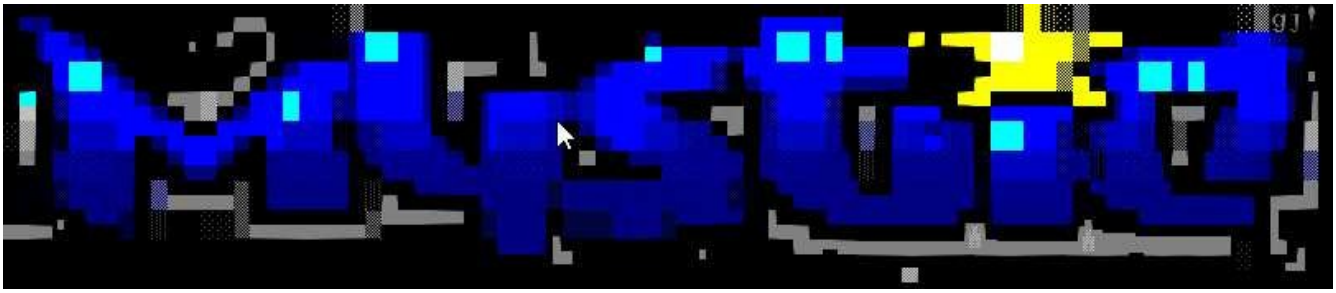

MPL Training
Lesson 12
Black Panther

Mystic BBS Related Functions

-=-=-=-=-=-=-=-

Let’s take a look at some of the Mystic BBS specific functions that are available for use in MPL programs.

-=-=-=-=-=-=-=-

ACS (S: String) : Boolean

This function will check a users ACS levels. If you want to make sure the user is at least security level 20 and is over age 18, this function can be used to check it.

If ACS(‘s20a18’) Then

-=-=-=-=-=-=-=-

CheckPw(PW: String) : Boolean

This function will check the supplied password against what is saved in the users record.

-=-=-=-=-=-=-=-

SetPw(PW: String)

This function will let the MPL set or change the users password.

-=-=-=-=-=-=-=-

ValidPW(PW: String): Byte

This will check to make sure the user supplied password is compliant with the current password policy set by the Sysop. This function will return a result depending on the status:

1 – Password does not meet min length

2 – Password does not meet min cap letters
3 – Password does not meet min symbols
4 – Password does not meet min numbers

-=-=-=-=-=-=-=-=-

DispFile(FN:String)

This is the function that is used to display any files to the user in your MPL program. When using this function, do not supply the file extension, as Mystic will display the one appropriate to the callers graphic settings.

DispFile('main')

Mystic will send the 'main.ans' file if the user has ansi enabled on their call. Otherwise, it will look for 'main.asc' if the are connected via an ascii terminal.

-=-=-=-=-=-=-=-=-

FillChar(R:Record;S:Integer;V:Value)

This function will fill a record with whichever character you provide. This way, if there is a call to that records variables, it won't display random characters on the screen.

```
Type
  MyUserRecord = Record
  UserName   : String[30]
  SomeValue  : Array[1..5] of Byte
End
Var
  u : MyUserRecord
Begin
  FillChar(u, sizeof(u), #0)
End
```

-=-=-=-=-=-=-=-=-

GetCfg

This function will load the current Mystic configuration data, so you will be able to access many of the paths. In the beginning of your code, include the 'Uses CFG'.

| | |
|---|---|
| CfgSysPath | System Path |
| CfgDataPath | Data Path |
| CfgMsgPath | Message Base Path |
| CfgProtPath | Protocol Path |
| CfgQWKPath | Local QWK Path |
| CfgMPEPath | Script (MPL) Path |
| CfgAttPath | File Attach Path |

| | |
|---|---|
| CfgLogsPath | System Logs Path |
| CfgTextPath | Text Files Path (User Theme) |
| CfgTempPath | Temp Path for Current Node |
| CfgMenuPath | Menu Files Path (User Theme) |
| CfgTimeOut | System Timeout |
| CfgSeeInvis | Give the ‘See Invisible’ ACS Value |
| CfgTNNodes | Number of Max Telnet Nodes to Allow |
| CfgNetDesc[1..30] | Gives Network Descriptions |
| CfgDefTheme | Returns Configured Default Theme |
| CfgTextFB | Returns Theme’s Text Fallback Theme |
| CfgScriptFB | Returns Theme’s Scrip Fallback Theme |
| CfgFallback | Returns T/F if Theme’s Fallback is Enabled |

```
Uses
  CFG
GetCfg
WriteLn(‘Mystic BBS is installed in ‘+CfgSysPath)
```

-=-=-=-=-=-=-=-=-

GetAttrXY(X, Y : Byte): Byte

The GetAttrXY function will return the attributes of the character at the X,Y position indicated. The attributes would consist of the MCI codes active at those coordinates.

```
Var
  Attr : Byte
Attr := GetAttrXY(1,1)
WriteLn(‘The attribute of the character at 1,1 is: ‘+Int2Str(Attr))
```

-=-=-=-=-=-=-=-=-

GetCharXY(X, Y : Byte): Char

This function will get the character at the given X,Y location on the user’s screen.

```
Var
  Ch : Char
Ch:=GetCharXY(1,1)
WriteLn(‘The user has the following character at 1,1: ‘+Ch)
```

-=-=-=-=-=-=-=-=-

GetPrompt(N: Word):String

This function will return the given prompt within the users theme. So, if you’d like to utilize the Mystic prompts within your MPL program, you could use this function.

```
WriteLn(GetPrompt(1))
```

This example would display prompt #1 within the users current theme.

-=-=-=-=-=-=-=-=-

GetScreenInfo(I, X, Y, Attr: Integer)

This is a very powerful function that will allow you and your MPL program to utilize files such as templates within Mystic. It will read the MCI codes, such as |!1 and |!2, and allow you to use those for placing text on the screen. A good example of this, is the Message Editor ansi file.

```
Var
  X, Y, Attr : Byte
Begin
  GetScreenInfo(1, X, Y, Attr)
  WriteLn('The value of the !1 MCI code was:')
  WriteLn('     X: '+Int2Str(X))
  WriteLn('     Y: '+Int2Str(Y))
  WriteLn(' Attr: '+Int2Str(Attr))
End
```

-=-=-=-=-=-=-=-=-

GetUser(N: Integer): Boolean

If you wanted to get information about other users on the BBS, you could use this function to obtain the following.

| | | |
|---|---|---|
| UserDeleted | Boolean | Is the user marked for deletion? |
| UserName | String | User's real name |
| UserAlias | String | User's BBS alias |
| UserPassword | String | User's Password (Not sure if this is still availble) |
| UserAddress | String | User's Street Address |
| UserCity | String | User's City/State |
| UserBirthday | String | User's Birth Date |
| UserSex | Char | User's Gender |
| UserSec | Byte | User's Security Level |
| UserFirstOn | LongInt | User's Date/Time of first call (Packed Date Format) |
| UserLastOn | LongInt | User's Date/Time of last call (Packed Date Format) |
| UserPosts | Integer? | Number of posts user has made |
| UserDL | Integer? | Number of downloads user has made |
| UserUL | Integer? | Number of uploads user has made |

(Not certain on the variable type for the last three, as they were not indicated in the what's new file)

```
Var
  A : Integer
Begin
  A:=1
```

```
  While GetUser(A) Do
  Begin
    WriteLn('User Alias: '+ UserAlias)
    A:=A+1
  End
End
```

-=-=-=-=-=-=-=-=-

GetThisUser

If you want to have the current user's information available in your MPL program, you will need to run this early in your program. This will obtain the 'USER' information listed above for the user currently logged in

```
Begin
  GetThisUser
  WriteLn('Welcome to the BBS, '+UserAlias)
  WriteLn('You have called '+UserCalls+' times!')
```

-=-=-=-=-=-=-=-=-

Graphics:Byte

This function will return the current user's graphics mode. It will be either a 0 or 1.

0 = ASCII graphics
1 = ANSI graphics

```
If Graphics = 1 Then
  WriteLn('ANSI')
Else
  WriteLn('ASCII')
```

-=-=-=-=-=-=-=-=-

HangUp

This will exit your MPL program, and Mystic will disconnect the user from the BBS.

```
If InputYN('Do you wish to hangup now? ') Then
  HangUp
```

-=-=-=-=-=-=-=-=-

Local:Boolean

This function will return 'True' if the caller is logged on locally, and 'False' if they are a remote caller.

```
If Local Then
  WriteLn('Local caller')
Else
  WriteLn('Remote caller')
```

-=-=-=-=-=-=-=-=-

MenuCmd(CM: String, Data: String)

This is very handy if you want your MPL program to run a Mystic command. The 'CM' variable is the command, and the 'Data' is any data the command needs to in order to run.

```
MenuCmd('NW','')
```

This would run the Mystic command 'NW', which is the Who's Online command. As this command does not require any additional data, the field if null.

-=-=-=-=-=-=-=-=-

NodeNum:Byte

This function will return the current user's node number they are on.

```
WriteLn('You are currently on node: '+NodeNum)
```

-=-=-=-=-=-=-=-=-

SysopLog(S: String)

This is very handy for having your MPL program keep a log of what the user does within your MPL. It will write the information into the NODEx.LOG file that Mystic keeps. It will automatically insert the date and time in the correct format, so you just need to enter the string you would like to have in the log file.

```
SysopLog(UserAlias+': just entered my awesome MPL program!')
```

-=-=-=-=-=-=-=-=-

I know this lesson got a bit longer than previous, but I wanted to cover most of the Mystic related functions in one lesson. There are a few others that will be covered in a later lesson, as I feel they are more in-depth, and need more time spent on those.