

Inputs:

```
Test1(False),  
LinesToAdd(98 { > 98 would cause an error line, try it, the big line test adds one Line too many}),  
MasterLoopCount(5),          { To Loop though the code ? times }  
CreateTestOn(False);         { Do the Test3 Create File Test...True on Chart Symbol 1 - False on the Chart Symbol 2 File}
```

Inputs: { The Real-time Only Test }

```
Test2(False),  
DoFlush(False), { True = Will Flush the Handle Contents of our  
Test File ...TS9 }  
NumberOfExchanges(5), { Swap Variables between the two charts this number  
of times }  
GoTime(835), { you can set this to someother time to wait for the  
test start }  
SymbolToGet("EBAY"), { What the second Chart Symbol is }  
MyTestFName4("C:\DLLTestFile\DLLTestFile.TS9"); { Our Create Test3 File Name }
```

Inputs:

```
MyTestFName("C:\DLLTestFile\DLLTestFile.TS6"),  
MyTestFName2("C:\DLLTestFile\DLLTestFile.TS7"),  
MyTestFName3("C:\DLLTestFile\DLLTestFile.TS8");
```

Inputs:

TSDisplayLimit(250); { This is the string length that you wish the maximum display string to be..250 is the max}

Vars:

 Hold1("Empty"),Hold2("Empty"),Hold3("Empty"),Hold4("Empty"), { You do not need to initialize the string, I am doing it just for
this demo }

 Hold5("Empty"),Hold6("Empty"),Hold7("Empty"),Hold8("Empty"),
 Hold9("Empty"),Hold10("Empty"),Hold11("Empty"),Hold12("Empty"),
 Hold13("Empty"),Hold14("Empty"),Hold15("Empty"),Hold16("Empty"),
 Hold17("Empty"),Hold18("Empty"),Hold19("Empty"),Hold20("Empty");

Vars:

 MyMainLoop(0),
 MyMasterLoop(0),
 LineNumber(0),
 MyLinesToAdd(LinesToAdd),
 MaxLinesAllowed(0),
 MaxFilesAllowed(0),
 MaxLineLength(0),
 HandlesUsed(0),
 VersionNumber(0);

Vars:

 Iterations(0),
 TestRetStr(""),

```
TestStr("");
```

Vars:

```
{ This would normally be set to NoErrMsg("")}  
{ If not defined (meaning...the variable NoErrMsg is not even here) an empty string will be returned}  
{ It's set to No Error just to show up in this demo}  
NoErrMsg("No Error"),  
LabelLine(0),  
LabelLine2(0),  
LabelLine3(0),  
LabelLine4(0),  
MyTSDisplayLimit(TSDisplayLimit),  
TrueDisplayLimit(250);
```

Vars:

```
MyFileHandle(0),  
MyFileHandle2(0),  
MyFileHandle3(0),  
MyFileHandle4(0),  
MyFileHandle5(0);    { Maybe Extra's here }
```

Vars:

```
RecCount(0),
```

```
RecCount2(0),  
RecCount3(0),  
RecCount4(0),  
RecCount5(0);    { Maybe Extra's here }
```

Vars:

```
FileClose(False),  
FileClose2(False),  
FileClose3(False),  
FileClose4(False),  
FileClose5(False);    { Maybe Extra's here }
```

Vars:

```
WhatError(""),  
WhatError2(""),  
WhatError3(""),  
WhatError4(""),  
WhatError5("");    { Maybe Extra's here }
```

Vars:

```
UpDate(1),  
AddNew(2);    { Line Function Actions }
```

Vars:{ Def for Handle Ops }

DeleteFile(1),

{ ok, I know, TS also has a FileDelete. but it won't clear the File and Handle information }

DeleteHandle(2),

{ Action code to clear a Handle's File information and make the Handle available again. It does not delete the

file }

ClearAll(3),

ClearHandle(4);

Vars:

BuildOk(True),

TripCount(0),

Test2OLineNum(0),

Test2HLineNum(0),

Test2LLineNum(0),

Test2CLineNum(0);

DefineDLLFunc: "CodetalkerFR2.Dll",LpStr,"CharString",Lpstr,Long;

{ String to Repeat, Number of Repeats }

DefineDLLFunc: "CodetalkerFR2.Dll",Bool,"CT_ErrMsgSet",LpStr;

{ Error String Msg..This is the string returned

on an error...Limit < 50 chars}

DefineDLLFunc: "CodetalkerFR2.Dll",Float,"CT_FRVersion";

{ No Params }

DefineDLLFunc: "CodetalkerFR2.Dll",LpStr,"CT_RetLastError";

{ No Params }

DefineDLLFunc: "CodetalkerFR2.Dll",Long,"CT_MaxLinesAllowed";

{ No Params }

DefineDLLFunc: "CodetalkerFR2.Dll",Long,"CT_MaxFilesAllowed";

{ No Params }

DefineDLLFunc: "CodetalkerFR2.Dll",Long,"CT_MaxLineLength";

{ No Params }

DefinedDLLFunc:	"CodetalkerFR2.Dll",Long,"CT_HandlesInUse";	{ No Params }
DefineDLLFunc:	"CodetalkerFR2.Dll",LpStr,"CT_ReadLine",Long,Long;	{ File Handle,LineNumber}
DefinedDLLFunc:	"CodetalkerFR2.Dll",LpStr,"CT_ReadLastLine",Long;	{ File Handle} { <- returns the line that is the
	LAST line in the file}	
DefineDLLFunc:	"CodetalkerFR2.Dll",LpStr,"CT_WriteLine",Long,Long,LpStr,LpFloat;	{ File Handle,WriteType(1=Update,2=New
	Write),String to Add,Return Record(line) #}	
DefinedDLLFunc:	"CodetalkerFR2.Dll",Long,"CT_RetRecordCount",Long;	{ File Handle}
DefineDLLFunc:	"CodetalkerFR2.Dll",LpStr,"CT_FileWrite",Long,LpBool;	{ File Handle,Boolean Success T/F}
DefineDLLFunc:	"CodetalkerFR2.Dll",LpStr,"CT_ReadFile",LpStr,LpFloat;	{ FileName,FHandle}
DefineDLLFunc:	"CodetalkerFR2.Dll",LpStr,"CT_CreateFile",LpStr,LpFloat;	{ FileName,FHandle}
DefinedDLLFunc:	"CodetalkerFR2.Dll",LpStr,"CT_FRLLabel",Long,LpStr,LpFloat;	{ File Handle,String Label Name,LINE NUMBER OF
	THE LABEL}	
DefinedDLLFunc:	"CodetalkerFR2.Dll",LpStr,"CT_FRMultiRet",Long,LpStr,LpStr,Long,Long;	{ File Handle,String Label Name,String Separator
	Char,Start #,# Lines to Ret as 1 line}	
DefinedDLLFunc:	"CodetalkerFR2.Dll",LpStr,"CT_FRLLabelRet",Long,LpStr;	{ File Handle,String Label Name}
DefinedDLLFunc:	"CodetalkerFR2.Dll",LpStr,"CT_FRLLabelSpecial",Long,LpStr,LpStr,Long;	{ File Handle,String Label Name,String Separator
	Char,Field # to return}	
DefinedDLLFunc:	"CodetalkerFR2.Dll",LpStr,"CT_HandleFileClear",Long,Long;	{ File Handle,ActionCode (DeleteFile or
	DeleteHandle}	
DefineDLLFunc:	"CodetalkerFR2.Dll",Long,"CT_RetHandle",LpStr;	{ FileName} { > File Handle is file active in
	memory else -1 }	
	{	

If CurrentBar = 1 Then TripCount=0;

If CurrentBar = 1 and CreateTestOn Then ClearDebug;

If Test1 and LastBarOnChart Then

Begin

For MyMasterLoop = 1 to MasterLoopCount { If more then Once, you will see the Output change }

Begin

{ Set what we would like the "GOOD...ie no errors" returned string message to be,}

{ Normally "" (EMPTY STRING)..for this DEMO I have set it to "No Error", or I will get..hey..HOLD5 is blank!}

{ No set will return an EMPTY STRING automatically }

Condition1=CT_ErrMsgSet(NoErrMsg);

VersionNumber=CT_FRVersion;

{ The above will return: A string contending the DLL's current version number}

MaxFilesAllowed=CT_MaxFilesAllowed;	{ Return Maximum Files ...ie handles allowed }
MaxLinesAllowed=CT_MaxLinesAllowed;	{ Return Maximum number of lines per file(handle) allowed }
MaxLineLength=CT_MaxLineLength;	{ Return Maximum lines length allowed, after which, line will be truncated}

WhatError=CT_ReadFile(MyTestFName,&MyFileHandle); { Notice ... the & is the TS command to send the address of the variable. MUST HAVE}

{ The above will: Return a HANDLE to use on all subsequent file operations }

{ This is VERY useful as we can now read,update, and or write, new lines in pseudo real-time and do not have to re-read the file each time }

{ WhatError is the variable that will receive any Error Message}

{ MyTestFName is file path and name }

{ &MyFileHandle is variable that will receive the working file HANDLE...That & symbol is a must have add to whatever variable name you use }

WhatError2=CT_ReadFile(MyTestFName2,&MyFileHandle2);	{ Load both of these files and return an access handle and an error msg if any }
--	--

{ Repeat for File #2 }

HandlesUsed=CT_HandlesInUse;	{ Returns Number of Handles (Open Files) in Use }
------------------------------	---

{ The above will return: A Long containing the current number of Handles in use..ie..the new of files that have been opened}

{ How many are we using now?Look at the print out}

LineNumber=11; { I pick Line 11 }

Hold1=CT_ReadLine(MyFileHandle,LineNumber); { Read LineNumber 11 from the file using Handle MyFileHand, put Line in Hold1 }

Hold2=CT_WriteLine(MyFileHandle,UpDate,"QQQ,"+NumToStr(LineNumber,0),&LineNumber); { Update(Change) Line "LineNumber"
(11) to QQQ,11 }

{ Update Line 11 and return any Error Msg in Hold2....&LineNumber will receive the line number it did }

{ OK, now we already know what line it did but, this same command is also used with Update and it will return the NEW line number then}

Hold3=CT_ReadLine(MyFileHandle,LineNumber);

{ ok, read back that changed line }

RecCount=CT_RetRecordCount(MyFileHandle);

{ The above will return: A Long containing the current Line or record this Handle has access to}

Hold4=CT_ReadLine(MyFileHandle,RecCount+1);

{ Now, to show you a line access error, this line will ask for 1 line > the Handle has }

```
Hold5=CT_WriteLine(MyFileHandle,AddNew,"I Added This Line "+NumToStr(RecCount+1,0),&LineNumber);
```

{ Let's ADD a new line and return any Error Msg into Hold5....&LineNumber will receive the line number it did }

{ OK, now see, this &LineNumber variable, is very useful as it now contends the NEW line number }

```
Hold6=CT_ReadLine(MyFileHandle,LineNumber);    { Read Back this line we just added }
```

{ ok, read back that new line }

```
Hold7=CT_ReadLastLine(MyFileHandle);    { Should be the same as the line returned above .. }
```

{ Because we know this is the LAST Line (the one at the bottom) in the file, we could also use the ReadLastLine command to read it back. }

{ This command will ONLY return the LAST Line in the file! }

{ This line just happens to be the LAST LINE, because we just added it. }

{ It may be useful when you want to: read-write-mod-write.....the last line in the file..like a new addition }

```
Hold8=CT_FRLabel(MyFileHandle,"[Label]",&LabelLine);
```

{ The above will return: A string containing the line directly following (below) the "label" name, the name is whatever your Label maybe }

```
Hold9=CT_FRLabel(MyFileHandle,"[Label2]",&LabelLine);
```

{ The above will return: A string containing the line directly following (below) the "label2" name, the name is whatever your Label maybe }

```
Hold10=CT_FRMultiRet(MyFileHandle,"[Label]","~",1,1); {Label Name,Multiple Line separator,Starting x line after label,for x number of lines }
```

{ The above will return: A string containing 1 line, starting 1 line following (below) the "label" name, this effectively returns the same }
{ as the line above, but is more versatile, as it will be seen in examples below. A user defined separator must be used, even if only asking for 1 line.}

{ I have chosen the tilde "~" for this example. The separator (delimiter) is placed in front of any multi line return to denote (delimit) the next line.}

```
Hold11=CT_FRMultiRet(MyFileHandle,"[Label2]","|",1,2);
```

{ The above will return: A string containing the contents of 2 lines, starting 1 line following (below) the "label2" name.}

{ A user defined separator (delimiter) must be used. I have chosen the split bar "|" for this example. }

{ The separator is placed in front of any multi line return to denote (delimit) the next line. }

```
Hold12=CT_FRMultiRet(MyFileHandle,"[Label2]","|",0,2); { Only this Command "CT_FRMultiRet" can do zero (0)}
```

{ The above will return: A string containing the contents of 2 lines, starting AT (Including) the line of the "label2" name.}
{ A user defined separator (delimiter) must be used. I have chosen the split bar "|" for this example. }
{ The separator is placed in front of any multi line return to denote (delimit) the next line. }

```
Hold13=CT_FRMultiRet(MyFileHandle,"[Label2]","|",3,1);
```

{ The above will return: A string containing the contents of 1 line, starting 3 lines following (below) the "label2" name.}
{ A user defined separator (delimiter) must be used. I have chosen the split bar "|" for this example. }
{ The separator is placed in front of any multi line return to denote (delimit) the next line. }

```
Hold14=CT_FRMultiRet(MyFileHandle,"[Label2]","|",4,2);
```

{ The above will return: A string containing the contents of 2 lines, starting 4 lines following (below) the "label2" name.}
{ A user defined separator (delimiter) must be used. I have chosen the split bar "|" for this example. }
{ The separator is placed in front of any multi line return to denote (delimit) the next line. }

```
Hold15=CT_FRLabelRet(MyFileHandle,"Big Fat Label");
```

{ The above will return: A string containing the contents OF THE LINE THAT CONTENTS THE "Big Fat Label"}
{ Note this is a different call then the top example}
{ This is here just to show, you can get the string contents of a string containing the "label" name, if this string "label" occurs more than once}

{ ANYWHERE in your entire FILE, it will only return the FIRST one found!}

```
Hold16=CT_FRMultiRet(MyFileHandle,"[Label]", "|", 4, 4);
```

{ The above will return: A string containing the contents of 4 lines, starting 4 lines following (below) the "label" name. }

{ A user defined separator (delimiter) must be used. I have chosen the split bar "|" for this example. }

{ The string will consist of the 4 string lines following the "label", which in this case will include our second label name }

```
Hold17=CT_FRMultiRet(MyFileHandle, "", "|", 1, 5);
```

{ The above will return: A string containing the contents of 5 lines, starting AT Line 1, as there is not "label" name specified }

{ A user defined separator (delimiter) must be used. I have chosen the split bar "|" for this example. }

{ The string will consist of the 5 string lines, 1-5 }

```
Hold18=CT_FRLabelSpecial(MyFileHandle, "Big Fat Label", ",", 2);    { HELLO, THIS ONE MAYBE VERY USEFUL }
```

{ Examples for below:

Big Fat Label, CSCO, 1, 2, 3, 4, 5 }

{ ^ }

{ | }

{ The above will return: A string containing the contents of "field" 2 of the line that starts with the "label" - "Big Fat Label" }

{ The "," tells the command this is the field delimiter character. }

{ Hummm, let's see, than maybe I could use this command to do global actions???????.....See Test2 }

```
Hold19=CT_FRMultiRet(MyFileHandle,"[label2]","|",0,1); { Only this Command "CT_FRMultiRet" can do zero (0)}
```

```
{ This is here to show that the search is case sensitive }
```

```
Hold20=CT_FRLLabelSpecial(MyFileHandle,"Big Fat Label","",9); { This will produce a field error }
```

```
{ ***** Message Below relates to ALL ***** }
```

```
{ BIG NOTE: If you use the separator (delimiter) character anywhere else in your file, it will mess up any line separation later }  
{ as this, separator (delimiter) character, will be interpid as a line break. The separator (delimiter) character must be unique}
```

```
Print(GetSymbolName," ",Date:7:0,Time:5:0," CurrentTime ",CurrentTime:5:0,  
      " VersionNumber ",VersionNumber:2:2,  
      " MaxFilesAllowed ",MaxFilesAllowed:3:0,  
      " MaxLinesAllowed ",MaxLinesAllowed:3:0,  
      " MaxLineLength ",MaxLineLength:3:0,  
      " Handles In Use ",HandlesUsed:3:0);
```

```
Print("File ",MyTestFName," Handle Number ",MyFileHandle:2:0," Any Errors --> ",WhatError);  
Print("Hold1  ",Hold1);
```

```
Print("Hold2  ",Hold2);
Print("Hold3  ",Hold3);
Print("Hold4  ",Hold4);
Print("Hold5  ",Hold5);
Print("Hold6  ",Hold6);
Print("Hold7  ",Hold7);
Print("Hold8  ",Hold8);
Print("Hold9  ",Hold9);
Print("Hold10 ",Hold10);
Print("Hold11 ",Hold11);
Print("Hold12 ",Hold12);
Print("Hold13 ",Hold13);
Print("Hold14 ",Hold14);
Print("Hold15 ",Hold15);
Print("Hold16 ",Hold16);
Print("Hold17 ",Hold17);
Print("Hold18 ",Hold18);
Print("Hold19 ",Hold19);
Print("Hold20 ",Hold20);
Print(); { just spacing the window }
```

{Number Two}

LineNumber=11;

Hold1=CT_ReadLine(MyFileHandle2,LineNumber);

Hold2=CT_WriteLine(MyFileHandle2,UpDate,"XXX,"+NumtoStr(LineNumber,0),&LineNumber); { Update(Change) Line "LineNumber"
(11) to XXX,11 }

Hold3=CT_ReadLine(MyFileHandle2,LineNumber);

RecCount2=CT_RetRecordCount(MyFileHandle2);

Hold4=CT_ReadLine(MyFileHandle2,RecCount2+1);

Hold5=CT_WriteLine(MyFileHandle2,AddNew,"What Line did I Add "+NumtoStr(RecCount2+1,0),&LineNumber);

Hold6=CT_ReadLine(MyFileHandle2,LineNumber);

Hold7=CT_ReadLastLine(MyFileHandle2); { Should be the same as the line returned above ..}

Hold8=CT_FRLLabel(MyFileHandle2,"[Label7]",&LabelLine2);

Hold9=CT_FRLLabel(MyFileHandle2,"[Label72]",&LabelLine2);

Hold10=CT_FRMultiRet(MyFileHandle2,"[Label7]", "~",1,1);

Hold11=CT_FRMultiRet(MyFileHandle2,"[Label72]", " |",1,2);

Hold12=CT_FRMultiRet(MyFileHandle2,"[Label72]", " |",0,2); { Only this Command "CT_FRMultiRet" can do zero (0)}

Hold13=CT_FRMultiRet(MyFileHandle2,"[Label72]", " |",3,1);

Hold14=CT_FRMultiRet(MyFileHandle2,"[Label72]", " |",4,2);

Hold15=CT_FRLLabelRet(MyFileHandle2,"Big Fat Label7");

Hold16=CT_FRMultiRet(MyFileHandle2,"[Label7]", " |",4,4);

Hold17=CT_FRMultiRet(MyFileHandle2,"", " |",1,5);

Hold18=CT_FRLLabelSpecial(MyFileHandle2,"Big Fat Label", "",2);

Hold19=CT_FRMultiRet(MyFileHandle2,"[label72]", " |",0,1); { Only this Command "CT_FRMultiRet" can do zero (0)}


```
Hold20=CT_FRLabelSpecial(MyFileHandle2,"Big Fat Label","",3);
```

```
Print(GetSymbolName," ",Date:7:0,Time:5:0," CurrentTime ",CurrentTime:5:0,  
      " VersionNumber ",VersionNumber:2:2,  
      " MaxFilesAllowed ",MaxFilesAllowed:3:0,  
      " MaxLinesAllowed ",MaxLinesAllowed:3:0,  
      " MaxLineLength ",MaxLineLength:3:0,  
      " Handles In Use ",HandlesUsed:3:0);
```

```
Print("File ",MyTestFName2," Handle Number ",MyFileHandle2:2:0," Any Errors --> ",WhatError2);
```

```
Print("Hold1  ",Hold1);
```

```
Print("Hold2  ",Hold2);
```

```
Print("Hold3  ",Hold3);
```

```
Print("Hold4  ",Hold4);
```

```
Print("Hold5  ",Hold5);
```

```
Print("Hold6  ",Hold6);
```

```
Print("Hold7  ",Hold7);
```

```
Print("Hold8  ",Hold8);
```

```
Print("Hold9  ",Hold9);
```

```
Print("Hold10 ",Hold10);
```

```
Print("Hold11 ",Hold11);
```

```
Print("Hold12 ",Hold12);
```

```
Print("Hold13 ",Hold13);
```

```
Print("Hold14 ",Hold14);
Print("Hold15 ",Hold15);
Print("Hold16 ",Hold16);
Print("Hold17 ",Hold17);
Print("Hold18 ",Hold18);
Print("Hold19 ",Hold19);
```

```
Print(); { just spacing the window }
```

```
{ Do Number one again }
```

```
LineNumber=11; { I pick Line 11 }
```

```
Hold1=CT_ReadLine(MyFileHandle,LineNumber); { Read LineNumber 11 from the file using Handle MyFileHand, put Line in Hold1 }
Hold2=CT_WriteLine(MyFileHandle,UpDate,"ZZZ,"+NumtoStr(LineNumber,0),&LineNumber); { Update(Change) Line "LineNumber" (11) to
ZZZ,11 }
Hold3=CT_ReadLine(MyFileHandle,LineNumber);
RecCount=CT_RetRecordCount(MyFileHandle);
Hold4=CT_ReadLine(MyFileHandle,RecCount+1);
Hold5=CT_WriteLine(MyFileHandle,AddNew,"I Added This Line "+NumtoStr(RecCount+1,0),&LineNumber);
Hold6=CT_ReadLine(MyFileHandle,LineNumber); { Read Back this line we just added }
Hold7=CT_ReadLastLine(MyFileHandle); { Should be the same as the line returned above ..}
Hold8=CT_FRLLabel(MyFileHandle,"[Label]",&LabelLine);
```

```

Hold9=CT_FRLabel(MyFileHandle,"[Label2]",&LabelLine);
Hold10=CT_FRMultiRet(MyFileHandle,"[Label]", "~",1,1); {Label Name,Multiple Line separator,Starting x line after label,for x number of
lines }
Hold11=CT_FRMultiRet(MyFileHandle,"[Label2]", "|",1,2);
Hold12=CT_FRMultiRet(MyFileHandle,"[Label2]", "|",0,2); { Only this Command "CT_FRMultiRet" can do zero (0)}
Hold13=CT_FRMultiRet(MyFileHandle,"[Label2]", "|",3,1);
Hold14=CT_FRMultiRet(MyFileHandle,"[Label2]", "|",4,2);
Hold15=CT_FRLabelRet(MyFileHandle,"Big Fat Label");
Hold16=CT_FRMultiRet(MyFileHandle,"[Label]", "|",4,4);
Hold17=CT_FRMultiRet(MyFileHandle,"", "|",1,5);
Hold18=CT_FRLabelSpecial(MyFileHandle,"Big Fat Label","",2);
Hold19=CT_FRMultiRet(MyFileHandle,"[laBel2]", "|",0,1); { Ok, here we Capitalized the B ...return...No label found }
Hold20=CT_FRLabelSpecial(MyFileHandle,"Big Fat Label","",7); { This time we ask for the limit MAX returned in the first cycle }

Print(GetSymbolName," ",Date:7:0,Time:5:0," CurrentTime ",CurrentTime:5:0,
      " VersionNumber ",VersionNumber:2:2,
      " MaxFilesAllowed ",MaxFilesAllowed:3:0,
      " MaxLinesAllowed ",MaxLinesAllowed:3:0,
      " MaxLineLength ",MaxLineLength:3:0,
      " Handles In Use ",HandlesUsed:3:0);

Print("ReCycle on File 1");
Print("File ",MyTestFName," Handle Number ",MyFileHandle:2:0," Any Errors --> ",WhatError);

```

```
Print("Hold1  ",Hold1);
Print("Hold2  ",Hold2);
Print("Hold3  ",Hold3);
Print("Hold4  ",Hold4);
Print("Hold5  ",Hold5);
Print("Hold6  ",Hold6);
Print("Hold7  ",Hold7);
Print("Hold8  ",Hold8);
Print("Hold9  ",Hold9);
Print("Hold10 ",Hold10);
Print("Hold11 ",Hold11);
Print("Hold12 ",Hold12);
Print("Hold13 ",Hold13);
Print("Hold14 ",Hold14);
Print("Hold15 ",Hold15);
Print("Hold16 ",Hold16);
Print("Hold17 ",Hold17);
Print("Hold18 ",Hold18);
Print("Hold19 ",Hold19);
Print("Hold20 ",Hold20);
Print();  { just spacing the window }
```

```
WhatError=CT_FileWrite(MyFileHandle,&FileClose);
```

```
{ Now for the Power...Write our file back out..with any changes!}
```

```
{ Write Handle file back out...return any Errors into WhatError and a BOOLEAN True/False for success into variable FileClose}
```

```
{ NOTE:.....the & symbol on the FileClose variable. It MUST BE on your variable name! }
```

```
If FileClose Then Print("File Handle ",MyFileHandle:2:0," Closed Successfully "," File Write Close ",FileClose)
```

```
Else Print("File Handle ",MyFileHandle:2:0," Write Error --> ",WhatError," File Write Close ",FileClose);
```

```
WhatError2=CT_FileWrite(MyFileHandle2,&FileClose2);
```

```
{ Do the Write close on file Handle 2 }
```

```
If FileClose2 Then Print("File Handle ",MyFileHandle2:2:0," Closed Successfully "," File Write Close ",FileClose2)
```

```
Else Print("File Handle ",MyFileHandle2:2:0," Write Error --> ",WhatError2," File Write Close ",FileClose2,NewLine);
```

```
Print(); { just spacing the window }
```

```
If CreateTestOn Then
```

```
Begin
```

```
{ Test Create Call....The Create Test }
```

```
MyFileHandle3=CT_RetHandle(MyTestFName3); { Let see if a handle has been established for this file}
```

```
If MyFileHandle3 > 0 Then
```

```
Begin
```

```
    Print("Old Handle ",MyFileHandle3:2:0," exist...deleting files and clearing handle information",NewLine);
```

```
    WhatError3=CT_HandleFileClear(MyFileHandle3>DeleteFile);           { Kill MyFileHandle3, if there is one }
```

```
    Print("Deletion results ....Errors --> ",WhatError3,NewLine);
```

```
End;
```

```
HandlesUsed=CT_HandlesInUse; { It should now report 2 handles in Use }
```

```
Print("Attempting to creating new file ",MyTestFName3," Handles In Use ",HandlesUsed:3:0,NewLine);
```

```
WhatError3=CT_CreateFile(MyTestFName3,&MyFileHandle3);
```

```
{ Yes, that's correct, I also give you the ability to create a new file, if you so desire.}
```

```
{ Just remember to Write close the File handle when you are done or it will be an empty file }
```

```
{ Also note that write closing does not flush the Handle content, it just writes the current contents out.}
```

```
{ You can still perform read-Update-AddNew commands}
```

IF MyFileHandle3 > 0 and WhatError3 = NoErrMsg THEN { If not zero, will have a Handle }

Begin

HandlesUsed=CT_HandlesInUse; { Returns Number of Handles (Open Files) in Use }

Print(GetSymbolName," ",Date:7:0,Time:5:0," Handles Out ",HandlesUsed:3:0);

MyLinesToAdd=MinList(LinesToAdd,MaxLinesAllowed);

Print("File Handle ",MyFileHandle3:2:0," WhatError3 --> ",WhatError3,"Will now write 1 Label + ",MyLinesToAdd:3:0," Lines into New file",NewLine);

Hold15=CT_WriteLine(MyFileHandle3,AddNew,"[New File Label Line 1]",&LineNumber);

{ now...you could use MaxLinesAllowed in the for loop below, to avoid a line max error. If you don't, an error message could be returned }

{ The input: LinesToAdd(99) or larger would cause a line error...Try it..Also note you can comment input lines...cool ah? }

For MyMainLoop = LineNumber+1 TO LineNumber+MyLinesToAdd

Begin

Hold15=CT_WriteLine(MyFileHandle3,AddNew,"I Added This Line "+NumToStr(MyMainLoop,0),&Value1);

```
    If Hold15 <> NoErrMsg Then Print("Error while attempting to Write Line ",MyMainLoop:2:0," <- Returned Error --> ",Hold15);  
End;
```

```
WhatError3=CT_FileWrite(MyFileHandle3,&FileClose3);    { Write Data out to TS8 }
```

```
If FileClose3 Then Print("File Handle ",MyFileHandle3:2:0," Closed Successfully "," File Write Close ",FileClose3)  
    Else Print("File Handle ",MyFileHandle3:2:0," Write Error --> ",WhatError3," File Write Close ",FileClose3,NewLine);
```

```
Print();    { just spacing the window }
```

```
If FileClose3 Then        { If we have a Clean Close...Let's do a read back }  
Begin
```

```
    RecCount3=CT_RetRecordCount(MyFileHandle3);
```

```
    Print("Current Records(Lines) for FHandle ",MyFileHandle3:2:0," ",RecCount3:3:0," Value1 ",Value1:3:0,NewLine);
```

```
    WhatError3=CT_ReadFile(MyTestFName3,&MyFileHandle3);
```

```
    IF MyFileHandle3 > 0 THEN
```

```
    Begin
```

```
        Print("Handle Read Back ",MyFileHandle3:2:0);
```

```
        Hold1=CT_FRMultiRet(MyFileHandle3,"[New File Label Line 1]","|",0,5);
```



```

Hold2=CT_FRMultiRet(MyFileHandle3,"[New File Label Line 1]","|",0,1);
Hold3=CT_FRMultiRet(MyFileHandle3,"[New File Label Line 1]","|",1,5);
Hold4=CT_FRMultiRet(MyFileHandle3,"","|",6,5); { Read from line 6 for 5 lines }
Hold5=CT_FRMultiRet(MyFileHandle3,"","|",11,5);
Hold6=CT_FRMultiRet(MyFileHandle3,"","|",16,11);
Hold7=CT_FRMultiRet(MyFileHandle3,"","|",16,13); { Notice if the Count is > the total records..return will be upto the max
count}

Hold8=CT_FRMultiRet(MyFileHandle3,"","|",0,13); { Notice if the Count is > the total records..return will be upto the max
count}

{ Also note the above line shows a zero in the start param, this is of course impossible, so it will be adjusted to 1 }

Print(GetSymbolName," ",Date:7:0,Time:5:0," CurrentTime ",CurrentTime:5:0," VersionNumber ",VersionNumber:2:2);

MyTSDisplayLimit=MinList(TSDisplayLimit,TrueDisplayLimit);

If StrLen(Hold1) <= MyTSDisplayLimit Then Print("Hold1  ",Hold1) Else
Value10=ShowLongString("Hold1",Hold1,MyTSDisplayLimit);
If StrLen(Hold2) <= MyTSDisplayLimit Then Print("Hold2  ",Hold2) Else
Value10=ShowLongString("Hold2",Hold2,MyTSDisplayLimit);
If StrLen(Hold3) <= MyTSDisplayLimit Then Print("Hold3  ",Hold3) Else
Value10=ShowLongString("Hold3",Hold3,MyTSDisplayLimit);
If StrLen(Hold4) <= MyTSDisplayLimit Then Print("Hold4  ",Hold4) Else

```

```

Value10=ShowLongString("Hold4",Hold4,MyTSDisplayLimit);
        If StrLen(Hold5) <= MyTSDisplayLimit Then Print("Hold5  ",Hold5) Else
Value10=ShowLongString("Hold5",Hold5,MyTSDisplayLimit);
        If StrLen(Hold6) <= MyTSDisplayLimit Then Print("Hold6  ",Hold6) Else
Value10=ShowLongString("Hold6",Hold6,MyTSDisplayLimit);
        If StrLen(Hold7) <= MyTSDisplayLimit Then Print("Hold7  ",Hold7) Else
Value10=ShowLongString("Hold7",Hold7,MyTSDisplayLimit);
        If StrLen(Hold8) <= MyTSDisplayLimit Then Print("Hold8  ",Hold8) Else
Value10=ShowLongString("Hold8",Hold8,MyTSDisplayLimit);

```

```

        Print(NewLine,"Now I'm sure you noticed a few printed differently","...TradeStation seems to have a display string limit of
250 chars",NewLine);

```

```

        { Now I am going to build a test string }

```

```

        TestStr="This line will be truncated as its over the limit-"; { After I auto-clone it! }

```

```

        Iterations=(MaxLineLength/StrLen(TestStr))+10;

```

```

        TestRetStr=CharString(TestStr,Iterations);          { Make a Big LONG String..string creation is from the DLL }

```

```

        If StrLen(TestRetStr) <= MyTSDisplayLimit Then Print("TestRetStr ",TestRetStr) Else Value10=ShowLongString("Initial
TestRetStr",TestRetStr,MyTSDisplayLimit);

```

{ Attempt to write into next record slot, a string longer then allowed length }

Print("Record Count Before Long string Add ",CT_RetRecordCount(MyFileHandle3):5:0," Value1 ",Value1:3:0);

Hold9=CT_WriteLine(MyFileHandle3,AddNew,TestRetStr,&Value1);

If StrLen(Hold9) <= MyTSDisplayLimit Then Print("Hold9 ",Hold9," Value1 ",Value1:3:0) Else
Value10=ShowLongString("Hold9",Hold9,MyTSDisplayLimit);

If Hold9=NoErrMsg Then

Begin

Hold10=CT_ReadLastLine(MyFileHandle3); { Read last line String back}

Print("Length of Hold10 ",StrLen(Hold10):5:0," TSDisplay Limits ",MyTSDisplayLimit:3:0);

If StrLen(Hold10) <= MyTSDisplayLimit Then

Begin

Print("Truncated TestRetStr");

Print(Hold10,NewLine);

End Else Value10=ShowLongString("Truncated TestRetStr",Hold10,MyTSDisplayLimit);

End Else Print("Error on Write: ",Hold9);

```
WhatError3=CT_FileWrite(MyFileHandle3,&FileClose3);    { re=Write Data out to TS8 }
```

```
If FileClose3 Then Print("File Handle ",MyFileHandle3:2:0," Closed Successfully "," File Write Close ",FileClose3,NewLine)
```

```
Else Print("File Handle ",MyFileHandle3:2:0," Write Error: ",WhatError3," File Write Close
```

```
",FileClose3,NewLine);
```

```
HandlesUsed=CT_HandlesInUse;        { Returns Number of Handles (Open Files) in Use }
```

```
{ Remember Closing does not delete, also be aware that if the loop was not in here}
```

```
{ TS would flush all variables at the start of each run and it would appear that the handles don't change}
```

```
Print(GetSymbolName," ",Date:7:0,Time:5:0," CurrentTime ",CurrentTime:5:0," Handles In Use ",HandlesUsed:3:0);
```

```
End;
```

```
End Else
```

```
Begin
```

```
Print("File Handle ",MyFileHandle3:2:0," File Creation Failed: ",WhatError3,NewLine);
```

```
End;
```

```
End; { MyFileHandle3 > 0 }
```

```
End; { CreateTestOn }
```

```
End; { MasterLoop }
```

```
End; { If Test1 }
```

```
{ Now, just use TradeStation's Text Manipulation words to break out multi-line strings and Bingo...you should be ok! }
```

```
{
```

This section will demonstrate a "possible" global useage, that is a little more compact than making files.

It requires a file name and create a file, but it's just a dummy file, created to get a handle.

WARNING NOTICE WARNING NOTICE!!!!

As TS currently does not provide anyway to ensure a sequence order, the read/writing does not guarantee, valid updated information.

The data, UNLESS it is STATIC, may be INCORRECT,...ie old... by one bar.

The Data is written, on entry into your study, if the writting study/chart/symbol is delayed, meaning...waiting for the opening tick of the next bar

and the reading study/chart/symbol reads first...Well, you see what I mean.

Try it....at least you have the ability to do it.

}

If Test2 and LastBarOnChart Then

Begin

If TripCount=0 Then { If we are just starting the test build the global labels }

Begin

Test2OLineNum=0;

Test2HLineNum=0;

Test2LLineNum=0;

Test2CLineNum=0;

Hold1=""; Hold2=""; Hold3=""; Hold4="";

MyFileHandle4=CT_RetHandle(MyTestFName4); { Let see if a handle has been established for this file}

If MyFileHandle4 > 0 Then

Begin

 If DoFlush Then WhatError4=CT_HandleFileClear(MyFileHandle4,ClearHandle); { Flush handle Infonew Start }

```

End Else
Begin
    WhatError4=CT_CreateFile(MyTestFName4,&MyFileHandle4); { We are going to use this Handle (a junk file) to pass globals }
    If MyFileHandle4 < 1 Then Print("Error Creating ",MyTestFName4," Error --> ",WhatError4) Else Print("Creating
",MyTestFName4," Error --> ",WhatError4);
End;

{ TripCount=0 and DoFlush=False }

Hold1=CT_FRLabel(MyFileHandle4,GetSymbolName+" Open",&Test2OLineNum);      { Get Field 2, THE LABEL IS FIELD 1 }
Hold2=CT_FRLabel(MyFileHandle4,GetSymbolName+" High",&Test2HLineNum);      { Get Field 2 }
Hold3=CT_FRLabel(MyFileHandle4,GetSymbolName+" Low",&Test2LLineNum);      { Get Field 2 }
Hold4=CT_FRLabel(MyFileHandle4,GetSymbolName+" Close",&Test2CLineNum);    { Get Field 2 }

If Test2OLineNum=0 Then Hold1=CT_WriteLine(MyFileHandle4,AddNew,GetSymbolName+"
Open"+"", "+NumtoStr(Open,2),&Test2OLineNum);      { Build the Global entry }
If Test2HLineNum=0 Then Hold2=CT_WriteLine(MyFileHandle4,AddNew,GetSymbolName+"
High"+"", "+NumtoStr(High,2),&Test2HLineNum);      { Build the Global entry }
If Test2LLineNum=0 Then Hold3=CT_WriteLine(MyFileHandle4,AddNew,GetSymbolName+"
Low"+"", "+NumtoStr(Low,2),&Test2LLineNum);      { Build the Global entry }
If Test2CLineNum=0 Then Hold4=CT_WriteLine(MyFileHandle4,AddNew,GetSymbolName+"
Close"+"", "+NumtoStr(Close,2),&Test2CLineNum); { Build the Global entry }

```

Condition1=Test2OLineNum=0 OR Test2HLineNum=0 OR Test2LLineNum=0 OR Test2CLineNum=0;

If Condition1 Then

Begin

Print("Error establishing Global Label's ",GetSymbolName,
" Hold1 ",Hold1," Test2OLineNum ",Test2OLineNum:3:0,
" Hold2 ",Hold2," Test2HLineNum ",Test2HLineNum:3:0,
" Hold3 ",Hold3," Test2LLineNum ",Test2LLineNum:3:0,
" Hold4 ",Hold4," Test2CLineNum ",Test2CLineNum:3:0);

BuildOk=False;

End Else BuildOk=True;

{ Ok, now let's watch for a number of Chart Swaps }

End; { If TripCount = 0 }

If BuildOk and (Date = CurrentDate or LastBarOnChart) and TripCount < NumberOfExchanges and Time >= GoTime Then

Begin

TripCount=TripCount+1;

MyFileHandle4=CT_RetHandle(MyTestFName4); { Let see if a handle has been established for this file}


```
{ Get the other Symbols Global Values }
```

```
Hold1=CT_FRLLabelSpecial(MyFileHandle4,SymbolToGet+" Open","",2); { Get Field 2 }
```

```
Hold2=CT_FRLLabelSpecial(MyFileHandle4,SymbolToGet+" High","",2); { Get Field 2 }
```

```
Hold3=CT_FRLLabelSpecial(MyFileHandle4,SymbolToGet+" Low","",2); { Get Field 2 }
```

```
Hold4=CT_FRLLabelSpecial(MyFileHandle4,SymbolToGet+" Close","",2); { Get Field 2 }
```

```
{ Now Print Both Symbol Values }
```

```
Print(GetSymbolName," Open ",Open:4:2," ",NumToStr(Test2OLineNum,0),Spaces(10),SymbolToGet," Open
",StrtoNum(Hold1):4:2);
Print(GetSymbolName," High ",High:4:2," ",NumToStr(Test2HLineNum,0),Spaces(10),SymbolToGet," High ",StrtoNum(Hold2):4:2);
Print(GetSymbolName," Low ",Low:4:2," ",NumToStr(Test2LLineNum,0),Spaces(10),SymbolToGet," Low
",StrtoNum(Hold3):4:2);
Print(GetSymbolName," Close ",Close:4:2," ",NumToStr(Test2CLineNum,0),Spaces(10),SymbolToGet," Close ",StrtoNum(Hold4):4:2);
Print();
```

```
{ Now UpDate our Symbol Numbers }
```

```
Hold1=CT_WriteLine(MyFileHandle4,UpDate,GetSymbolName+"
Open"+", "+NumtoStr(Open,2)+" "+NumToStr(Time,0),&Test2OLineNum); { Update the Global entry }
Hold2=CT_WriteLine(MyFileHandle4,UpDate,GetSymbolName+"
High"+", "+NumtoStr(High,2)+" "+NumToStr(Time,0),&Test2HLineNum); { Update the Global entry }
```

```

        Hold3=CT_WriteLine(MyFileHandle4,UpDate,GetSymbolName+"
Low"+"", "+NumToStr(Low,2)+"", "+NumToStr(Time,0),&Test2LLineNum);          { Update the Global entry }
        Hold4=CT_WriteLine(MyFileHandle4,UpDate,GetSymbolName+"
Close"+"", "+NumToStr(Close,2)+"", "+NumToStr(Time,0),&Test2CLineNum); { Update the Global entry }

RecCount4=CT_RetRecordCount(MyFileHandle4);

Print(GetSymbolName," ",ELDatetoString(Date)," ",Time:5:0," Global Swap Test",
                                           " TripCount ",TripCount:1:0,
                                           " Record #'s ",RecCount4:5:0,NewLine);

Condition1=Hold1=NoErrMsg and Hold2=NoErrMsg and Hold3=NoErrMsg and Hold4=NoErrMsg;

If Condition1 Then Print("Error Updating Global Labels ",GetSymbolName,"
",Hold1,Spaces(2),Hold2,Spaces(2),Hold3,Spaces(2),Hold4);

If TripCount=NumberOfExchanges Then
Begin
    WhatError4=CT_FileWrite(MyFileHandle4,&FileClose4);    { re=Write Data out to TS9 }

    If FileClose4 Then Print("File Handle ",MyFileHandle4:2:0," Closed Successfully "," File Write Close ",FileClose4,NewLine)
    Else Print("File Handle ",MyFileHandle4:2:0," Write Error: ",WhatError4," File Write Close
",FileClose4,NewLine);

```

End;

End; { BuildOk and Date = CurrentDate and TripCount < NumberOfExchanges }

End; { Test2 }

{ ** Copyright (c) 2002 Codetalker, All rights reserved. ** }