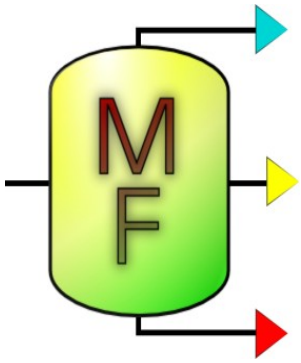
PROGRAMMER'S GUIDE FOR THE MULTIFLASH VISUAL BASIC INTERFACE

KBC ADVANCED TECHNOLOGIES LTD

Version 6.1
JUNE 2016



KBC Advanced Technologies ltd
42-50 Hersham Road
Walton on Thames
Surrey
KT12 1RZ
UK
T +44 (0) 1932 242 424
F +44 (0) 1932 224 214
e-mail: software@kbcat.com

DISCLAIMER

This document and the information contained within it is the copyright of KBC Advanced Technologies plc.

While every effort has been made to ensure that the information contained in this document is correct and that the software and data to which it relates are free from errors, no guarantee is given or implied as to their correctness or accuracy. Neither Infochem/KBC Advanced Technologies plc nor any of its employees, contractors or agents shall be liable for direct, indirect or consequential losses, damages, costs, expenses, claims or fee of any kind resulting from any deficiency, defect or error in this document, the software or the data.

CONTENTS

Introduction ***2***
- System requirements 2
- Installation instructions 2
- Path settings 3
- Technical support 3

Changes in Version 6.1 ..... ***4***

The Visual Basic interface functions ***5***
- Function summary 5
- Argument passing 5
- Return values 7
- Multiflash error codes 7

Function specifications ***8***
- MFSTRT 8
- MFIN 10
- MFOUT 10
- MFFILE 11
- MFDFLA 11
- MFDPRP 15
- MFDPHN 17
- MFDERR 20
- MFDEMS 20
- MFDUN 21
- MFDCMP 21
- MFDCMS 22
- MFDPDS 23
- MFDPDT 23
- MFDPTF 24
- MFEND 25

Examples ***26***
- Numerical mode 26
- Text mode (not recommended) 28
- VB.NET Examples 30

INTRODUCTION

Multiflash is an advanced software package for performing complex equilibrium calculations quickly and reliably.

This document describes the simplified API for the Multiflash Windows DLL which is designed to be easy to use from Microsoft Visual Basic (VB) or VBA applications. Other Multiflash interfaces include a Microsoft Excel add-in (XLL), a COM module that supports the CAPE-OPEN thermodynamics interfaces, a Matlab/Simulink interface and a more extensive low-level API. Separate documentation is available for these interfaces.

SYSTEM REQUIREMENTS

Operating System: Microsoft Windows 10, 8.1, 8, 7.

A Pentium-class processor.

Visual Basic 6, VB.Net or Visual basic for Applications.

The files supplied require about 130Mb of disk space.

INSTALLATION INSTRUCTIONS

The Multiflash DLL is supplied as part of the standard Multiflash installation. See the *Installation Guide for Multiflash.* The version 6.1 DLL is called MFPVT32.DLL (or MFPVT64.DLL for 64bit).

Support files for VB developers are supplied in the \API\VB\ directory in the Multiflash Installation directory.

Files supplied:

MFVBIntPVT.pdf this document (Adobe Acrobat PDF format).

Examples directory

Ex1.NET/vbp use of interface in numerical model.

Ex2.NET/vbp use of interface in text model.

Paths.bas module defining pathnames used by above examples (must be changed to be consistent with your installation).

Header Files directory

MFVBId.bas Multiflash function declarations and constant definitions for VB.

PATH SETTINGS

The directory in which MFPVT32.DLL (or MFPVT64.DLL) is placed must be on the current search path. The installation program for the Multiflash Windows program places the dll in the Multiflash installation directory, *C:\Program Files (x86)\KBC\MF61* by default. If you have installed the dll in another directory, you must ensure that this directory is on the path or use the LoadLibrary Windows API function.

To set the PATH:

Windows 10/8/7: use Control Panel and click on System. Select the Advanced tab and the Environment button.

TECHNICAL SUPPORT

Documentation

The VB interface is described in this document. You should also have a copy of the *Multiflash Command Reference* that describes the Multiflash command language and the *User Guide for Multiflash for Windows* that contains background information. These documents are supplied in PDF format in the \User Manuals directory branch on the CD ROM

Contact details

If you need further assistance contact us at:

Infochem/KBC Advanced Technologies plc
42-50 Hersham Rd,
Walton-on-Thames
KT12 1RZ
UK
Telephone: +44 (0)19 3224 2424
Fax: +44 (0)20 7407 3927
email: software@kbcat.com

CHANGES IN VERSION 6.1

No changes have been made for the VB interface for version 6.1, but the following should be considered when using the interface:

Argument passing:

- Unlike previous versions the string length argument is always used, so it must be correctly set to the exact length of the string passed.

Processing commands:

-Due to changes in the command processor, not all commands will work as before in text mode. Therefore is it recommended to **not** use text mode to perform flashes. Use MFDFLA instead.

See the general list of changes in the *Release Notes* for a detailed list of what is new for version 6.1.

THE VISUAL BASIC INTERFACE FUNCTIONS

FUNCTION SUMMARY

The VB interface consists of the following functions:

MFSTRT	starts Multiflash
MFIN	processes a Multiflash command
MFOUT	gets output from Multiflash
MFFILE	processes a file of Multiflash commands
MFDFLA	does a flash calculation
MFDPRP	gets a property previously calculated by MFDFLA
MFDPHN	calculates a series of points on a phase envelope
MFDERR	gets Multiflash error codes
MFDEMS	gets message associated with Multiflash error code
MFDUN	returns unit names
MFDCMP	returns a component name
MFDCMS	returns all component names
MFDPDS	returns phase descriptor names
MFDPDT	returns pure component constant properties
MFDPTF	returns pure component T-dependent properties
MFEND	ends a Multiflash run

Each of these functions is described in detail in the following sections. The VB declarations are provided in the file MFVBId.bas and are also given with each function description. The constants needed to call some of the functions are also defined in MFVBId.bas.

ARGUMENT PASSING

The Multiflash DLL is produced using Intel Fortran Compiler using Digital Fortran compatibility mode which imposes its own conventions on function names and the way that arguments are passed. The VB function declarations are provided in the file MFVBId.bas.

Integers are 4-byte integers (VB6 type Long, VB.Net type Integer) and are passed by reference.

Reals are 8-byte reals (VB type Double) and are passed by reference.

Strings are null-terminated C strings passed by reference for which Visual Basic requires the declaration ‘ByVal’. This is a feature of Visual Basic; a VB string declared as ‘ByVal’ is equivalent to a C string passed by reference. Every string argument is followed by a 4 byte integer string length (VB6 type Long, VB.Net type Integer) which is passed by value. In 64bit mode the string length must be a 64bit integer. This is a ‘hidden’ string length argument in Fortran which is inserted by the compiler. The string length argument is used by all of the procedures described in this document so it must be always correctly set.

If the string argument is an output from the Multifash function, the way it should be handled depends if it a single string or a string array:

1. Single string output:

The string must be pre-filled with some characters up to the required length. For example to get a component name:

```
Dim comp as string
Comp = Space(72)
```

2. Array of strings output

Multiflash does not support direct output of string arrays from its functions. A trick to over come this is to use a large string that will hold all the elements of the array and then process the result. Example:

```
Dim comp(40) as string * 20 '
Dim allcomps as string * (800)    ' 40 components * 20 character per component
allcomps = Space(800)
ret =  mfdcms(nco, allcomps, 20) ' here the string length is important to tell Multiflash the size of each element in the array.
' process the result
For i = 1 To nco
    comp(i) = Mid(allcomps, (i - 1) * 20 + 1, 20)
Next i
```

All functions return a 4 byte integer (VB6 type Long, VB.Net type Integer) value. Function names exported from the DLL are ‘decorated’ by the compiler. A code that indicates the number of arguments is appended to the original name. For example, the function MFIN becomes _MFIN@8 because the argument list requires 4 bytes to pass a string by reference plus 4 bytes to pass a Long by value. The function declarations in MFVBId.bas use an Alias to restore the original name.

RETURN VALUES

The standard return values for the Multiflash functions are shown in the following table. If any other values are returned, they are defined along with the function description.

0	Successful
-1	Multiflash warning state
1	Multiflash error state

MULTIFLASH ERROR CODES

The function MFDERR returns Multiflash error and warning codes. The text associated with a numeric code can be obtained by calling MFDEMS. Alternatively, the errors may be looked up in the help file MFErr.chm that is supplied in the installation directory.

FUNCTION SPECIFICATIONS

The functions that can be called via the Visual Basic interface are described in this section. The argument types are listed for VB6. For VB.NET the following correspondences are to be considered:

VB6	VB.NET	Comments
Integer	Short	16 bits
Long	Integer	32 bits
N/A	Long	64 bits
String	String	VB.NET does not support fixed length strings

MFSTRT

MFSTRT(mode, password, len_Password, app_path, len_app_path, usr_path, len_usr_path)

Starts the Multiflash command processor.

Arguments

Name	input/ output	type	description
Mode	in	Long	operating mode for DLL command processor (see note 1)
Password	in	String	depending on the type of software license a password may need to be supplied. Must be set to a single blank character if unused.
len_Password	in	Long (ByVal)	length of Password string
app_path	in	String (up to 256 characters)	directory path used to search for application-specific configuration files (see note 3)
len_app_path	in	Long (ByVal)	length of app_path string
usr_path	in	String (up to 256 characters)	directory path used to search for user-specific configuration files (see note 4). Must be set to a single blank character if unused.
len_usr_path	in	Long (ByVal)	length of usr_path string

Return values

0	Successful
-209	cannot read databank configuration file

-228	Multiflash already loaded (see note 5)
1	Multiflash error state
205	Password error
684	cannot initialise security device
685	status checks of security device failed
686	software licence has expired
687	serial number of software/security device not valid
688	software version number is not compatible with the licence
690	No more available network licenses (network version only)
691	cannot initialise security device (either missing or network server not running)

Notes

The **mode** argument can be one of the following constants defined in MFVBId.bas:

```
MF_MODE_VBNUMERIC
MF_MODE_VBTEXT
```

If MF_MODE_VBNUMERIC is specified, the program will operate in numerical mode in which case function MFDERR should be used to collect the error status information. In this mode, function MFOUT is inoperative. Calculations are initiated by calling MFDFLA and results are collected by calling MFDPRP.

If MF_MODE_VBTEXT is specified, the program will operate in text mode in which case function MFOUT should be used to collect the error status information and all output generated by calling MFIN or MFFILE. In this mode, function MFDERR is inoperative. This mode is not recommended if the intention is to do flash calculations, using `ptflash` command for example.

Whenever a function returns an error or warning status code, the error information must be obtained by calling either MFDERR or MFOUT as appropriate, otherwise the output buffer will fill up and could overflow.

The **app_path** argument should contain a directory specification that Multiflash uses to search for all databank files (infodata etc.) and other support files. This will normally be the directory where Multiflash was installed.. For example:

```
c:\Program Files (x86)\KBC\mf61
```

The **usr_path** argument is kept for compatibility. The usage of a MFCONFIG.DAT to set command processor default settings is not supported anymore.

A return value of -228 from MFSTRT indicates that the Multiflash DLL has already been initialised by your application. This can happen when running an application repeatedly from the Visual Basic environment because the DLL is not released until Visual Basic itself is shut down. If MFSTRT is called again without a call to MFEND to close the session a warning -228 is also returned.

MFIN

MFIN(input_text, len_input)

Passes a command or series of commands to Multiflash and attempts to execute the commands entered.

Arguments

Name	input/ output	type	description
input_text	in	String	Multiflash commands to execute (see note 1)
len_input	in	Long (ByVal)	number of characters in input_text (including terminating null)

Notes

The **`input_text`** string can contain any valid Multiflash command. For a description of the command language, see the *Multiflash Command Reference*. The string length must be passed in **`len_input`** .

Whenever a function returns an error or warning status code, the error information must be obtained by calling either MFDERR or MFOUT as appropriate, otherwise the output buffer will fill up and could overflow.

In text mode a call to MFIN must be followed by a call to MFOUT to return the output in order to avoid buffer overflow whether or not an error has occurred. This output can be displayed visually, for example in a text box or other device in the language environment, if required.

MFOUT

MFOUT(output_text, len_output)

Returns (text) output from Multiflash to the calling program.

Arguments

Name	input/ output	type	description
output_text	out	String	text output from Multiflash (see note 1)
len_output	in	Long (ByVal)	maximum number of characters that may be returned in output_text (including terminating null)

Notes

The **`output_text`** string contains output from Multiflash. It is formatted in exactly the same way as output from the DOS version of Multiflash. Lines of output are separated by carriage return + line feed characters. The end of MF output in the

string is marked by a null character.

Before calling MFOUT the string must be initialised in the VB application to allocate space for the return value. A convenient way of initialising the string is to use the VB Space function, *e.g.*

```
Dim output_text As String
const len_output As Long = 10000
output_text = Space( len_output )
```

MFFILE

MFFILE(file_name, fname_len)

Passes a file containing Multiflash commands to Multiflash and attempts to execute the commands found.

Arguments

Name	input/ output	Type	description
file_name	in	String (up to 256 characters)	File containing Multiflash commands to execute (see note 1)
fname_len	in	Long (ByVal)	String length of file name

Notes

The **`file_name`** string can contain any valid filename (with full path as necessary) for a file containing Multiflash commands. For a description of the command language, see the *Multiflash Command Reference*.

Whenever a function returns an error or warning status code, the error information must be obtained by calling either MFDERR.

MFDFLA

MFDFLA(flash_type, solution_type, fixed_phase, fixed_phase_len, constraint_1, constraint_2, constraint_3, no_comps, amounts, stream_type, stream_type_len)

Makes Multiflash do a phase equilibrium (flash) calculation. MFDFLA must be called irrespective of the number of phases that has been defined so that Multiflash will evaluate the properties of the phases. To return phase properties use MFDPRP. The properties calculated by MFDFLA are controlled by the `'set physprops'` command. See Note 1 to the description of MFDPRP.

Arguments

Name	input/ output	type	Description

flash_type	in	Long	code specifying which flash calculation to perform (see note 1)
solution_type	in	Long	code specifying the type of solution and basis (mole fraction, mass fraction etc.) required for dew point, bubble point and fixed phase fraction flashes, Ignored for other flashes (see note 2).
fixed_phase	in	String (up to 40 characters)	name of phase to be used in fixed phase fraction flash calculation, Ignored for other flashes (see note 3).
fixed_phase	in	Long (ByVal)	String length of phase name
constraint_1 constraint_2 constraint_3	in	Double	values of constraints to be used for flash calculations - depend on flash type specified (see note 4)
no_comps	in	Long	number of components in the array amounts; this number should be consistent with the number of components defined using MFIN or MFFILE
Amounts	in	array of Double	overall number of moles or masses of each component; whether this array is interpreted as moles or masses depends on the input units specified
Stream_type	in	String (up to 40 characters)	name of Multiflash stream type to be used in calculation. If stream types have not been defined this argument should be set to an empty string
Stream_type	in	Long (ByVal)	String length of stream type name

Notes

The `flash_type` argument can be one of the following constants defined in MFVBId.bas:

MF_PTF	flash at constant temperature and pressure
MF_PHF	flash at constant pressure and enthalpy
MF_THF	flash at constant temperature and enthalpy
MF_PSF	flash at constant pressure and entropy
MF_TSF	flash at constant temperature and entropy
MF_PVF	flash at constant pressure and volume/density
MF_TVF	flash at constant temperature and volume/density
MF_PUF	flash at constant pressure and internal energy
MF_TUF	flash at constant temperature and internal energy
MF_PFRACF	fixed phase fraction flash at constant pressure
MF_TFRACF	fixed phase fraction flash at constant temperature
MF_PDEWF	dew point at fixed pressure

MF_TDEWF	dew point at fixed temperature
MF_PBUBF	bubble point at fixed pressure
MF_TBUBF	bubble point at fixed temperature
MF_UVF	flash at constant internal energy and volume
MF_HSF	flash at constant enthalpy and entropy
MF_PTR	chemical equilibrium at constant temperature and pressure
MF_PDEWR	dew point at fixed pressure with chemical reaction
MF_TDEWR	dew point at fixed temperature with chemical reaction
MF_PBUBR	bubble point at fixed pressure with chemical reaction
MF_TBUBR	bubble point at fixed temperature with chemical reaction

The **`solution_type`** argument can be one of the following constants defined in MFVBId.bas:

MF_NORMAL	default solution type, molar fraction of phase
MF_LOWER_NORMAL	normal solution at lower T/P (if more than one), molar fraction of phase
MF_UPPER_NORMAL	normal solution at upper T/P (if more than one), molar fraction of phase
MF_LOWER_RETROGRADE	retrograde solution at lower T/P (if more than one), molar fraction of phase
MF_UPPER_RETROGRADE	retrograde solution at upper T/P (if more than one), molar fraction of phase
MF_UNSPECIFIED	solution closest to initial estimate of T/P, molar fraction of phase (not recommended)
MF_NORMAL_MASSFRACTION	default solution type, mass fraction of phase
MF_LOWER_NORMAL_MASSFRACTION	normal solution at lower T/P (if more than one), mass fraction of phase
MF_UPPER_NORMAL_MASSFRACTION	normal solution at upper T/P (if more than one), mass fraction of phase
MF_LOWER_RETROGRADE_MASSFRACTION	retrograde solution at lower T/P (if more than one), mass fraction of phase
MF_UPPER_RETROGRADE_MASSFRACTION	retrograde solution at upper T/P (if more than one), mass fraction of phase
MF_UNSPECIFIED_MASSFRACTION	solution closest to initial estimate of T/P, mass fraction of phase (not recommended)
MF_NORMAL_VOLUMEFRACTION	default solution type, volume fraction of phase

MF_LOWER_NORMAL_VOLUMEFRACTION	normal solution at lower T/P (if more than one), volume fraction of phase
MF_UPPER_NORMAL_VOLUMEFRACTION	normal solution at upper T/P (if more than one), volume fraction of phase
MF_LOWER_RETROGRADE_VOLUMEFRACTION	retrograde solution at lower T/P (if more than one), volume fraction of phase
MF_UPPER_RETROGRADE_VOLUMEFRACTION	retrograde solution at upper T/P (if more than one), volume fraction of phase
MF_UNSPECIFIED_VOLUMEFRACTION	solution closest to initial estimate of T/P, volume fraction of phase (not recommended)
MF_NORMAL_NUCLEATION	default solution type, nucleation of phase
MF_LOWER_NORMAL_NUCLEATION	normal solution at lower T/P (if more than one), nucleation of phase
MF_UPPER_NORMAL_NUCLEATION	normal solution at upper T/P (if more than one), nucleation of phase
MF_LOWER_RETROGRADE_NUCLEATION	retrograde solution at lower T/P (if more than one), nucleation of phase
MF_UPPER_RETROGRADE_NUCLEATION	retrograde solution at upper T/P (if more than one), nucleation of phase
MF_UNSPECIFIED_NUCLEATION	solution closest to initial estimate of T/P, nucleation of phase (not recommended)

The **`fixed_phase`** argument is the *name* of the phase, the amount of which is being held fixed in the flash calculation. The phase name is user-defined by the Phasedescriptor (PD) command. The default PD names used in the supplied problem configuration files (*.mfl) and generated by the GUI are:

GAS	gas/vapour phase
LIQUID1	first hydrocarbon liquid phase
LIQUID2	second hydrocarbon liquid phase
WATER	aqueous liquid phase
ICE	ice phase
HYDRATE1	hydrate structure I phase
HYDRATE2	hydrate structure II phase

The quantities held constant for the different flash types are

Flash type	First constraint	Second constraint	Third constraint
MF_PTF	pressure	temperature	ignored
MF_PHF	pressure	enthalpy	ignored
MF_THF	temperature	enthalpy	ignored
MF_PSF	pressure	entropy	ignored

MF_TSF	temperature	entropy	ignored
MF_PVF	pressure	volume/density	ignored
MF_TVF	temperature	volume/density	ignored
MF_PUF	pressure	internal energy	ignored
MF_TUF	temperature	internal energy	ignored
MF_PFRACF	pressure	molar phase fraction	ignored
MF_TFRACF	temperature	molar phase fraction	ignored
MF_PDEWF	pressure	ignored	ignored
MF_TDEWF	temperature	ignored	ignored
MF_PBUBF	pressure	ignored	ignored
MF_TBUBF	temperature	ignored	ignored
MF_UVF	internal energy	volume/density	ignored
MF_HSF	enthalpy	entropy	ignored
MF_PTR	pressure	temperature	ignored
MF_PDEWR	pressure	ignored	ignored
MF_TDEWR	temperature	ignored	ignored
MF_PBUBR	pressure	ignored	ignored
MF_TBUBR	temperature	ignored	ignored

MFDPRP

MFDPRP(phase_wanted, phase_wanted_len, no_props_wanted, props_wanted, comp_no, properties)

Returns specified properties for the last phase equilibrium calculation

Arguments

Name	input/ output	type	description
Phase_wanted	in	String (up to 40 characters)	name of phase for which properties required - the name is the PD name used when setting up phases (see fixed_phase above). If phase_wanted is set to OVERALL then the total values for all phases, summed appropriately if possible, are reported
phase_wanted_len	in	Long (ByVal)	String length of phase name
no_props_wanted	in	Long	the number of properties required

Props_wanted	in	array of Long	array of codes defining the properties required (see note 2).
comp_no	in	array of Long	component numbers associated with props_wanted (see note 2).
Properties	out	array of Double	array of values corresponding to the properties requested (see note 3). The units used are the current Multiflash output units.

Notes

By default, Multiflash calculates the compositions and amounts of each phase. The properties calculated are controlled by the `set physprops` command. The default is equivalent to `'set physprops 0'`. If you require molecular weight, volume, enthalpy, entropy, Gibbs and internal Energies use `'set physprops 1'`. To calculate the heat capacities Cp, Cv or the speed of sound use `'set physprops 2'`.*before* initiating a flash calculation. If the letter 'V', 'C' and/or 'S' are added to the physprops level the transport properties viscosity, thermal conductivity and surface tension, respectively, are calculated, if the letter 'E' is added, thermal properties relative to the elements are also calculated. To calculate all properties use `'set physprops 2EVCS'`. Calculation time will be saved if you set the property calculation level to the minimum required. To calcualte diffusion coefficients add a 'D', for example: `'set physprops 2EVCSD'`. For a description of the `set physprops` command, see the *Multiflash Command Reference*.

The **`props_wanted`** argument can be one of the following constants defined in MFVBId.bas:

MF_AMOUNTS	concentration of a component in the phase requested as total amount
MF_FRACTIONS	concentration of a component in the phase requested as mass/mole fraction
MF_PHASE	total amount of that phase
MF_TEMPERATURE	temperature
MF_PRESSURE	pressure
MF_VOLUME (or MF_DENSITY)	volume/density
MF_ENTHALPY	Enthalpy
MF_ENTROPY	Entropy
MF_FUGCOEF	fugacity coefficient of specified component(s)
MF_LNFUGCOEF	log fugacity coefficient of specified component(s)
MF_DIFFCOEF	Diffusion coefficients
MF_INTERNALENERGY	internal energy
MF_GIBBS	Gibbs energy
MF_ZFACTOR	compressibility factor

MF_ENTHALPY_ELEM	enthalpy relative to elements
MF_ENTROPY_ELEM	entropy relative to elements
MF_INTERNALENERGY_ELEM	internal energy relative to elements
MF_GIBBS_ELEM	Gibbs energy relative to elements
MF_AVMOLWT	average molecular weight
MF_CP	Cp
MF_CV	Cv
MF_SSOUND	speed of sound
MF_VISCOSITY	Viscosity
MF_THCOND	thermal conductivity
MF_SURTEN	surface tension

Where MF_AMOUNTS , MF_FRACTIONS , MF_FUGCOEF or MF_LNFUGCOEF is specified, the corresponding element in array comp_no is used to define which component is required. For all other properties, comp_no is ignored. A property value will only be returned if a model for the property has been specified for the phase requested.

If the property requested is MF_DIFFCOEF, comp_no has to contain the two component ids necessary to calculate the property. For example ICOMP = (1,2) to get diffusion coefficient between components 1 and 2.

There are two special cases when the value returned is equal to either of the following constants defined in MFVBId.bas:

MF_UNDEFINED	value unknown as flash calculation has failed or input is invalid
MF_NOT_AVAILABLE	value unknown as phase in question does not exist under the conditions specified

MFDPHN

MFDPHN(solution_type, fixed_phase, fixed_phase_len, flash_type, direction, initial_pressure, initial_temperature, spec, no_comps, amounts, stream_type, stream_type_len, no_points, n_skip, point_id, p_points, t_points, v_points, h_points, s_points, u_points, nphout, phfout)

Calculates a: locus of temperature/pressure points at a specified constraint value. To calculate a phase boundary the constraint should be the fraction of a particular phase. Alternatively this function can calculate lines of constant enthalpy, entropy, volume or internal energy. The user can specify the number of points on the locus to be returned but the spacing of points is determined by the calculation algorithm. A non-zero status will be returned by MFDPHN if it is called with unacceptable arguments but once a calculation has been started a zero status is always returned because the calculation will usually be terminated by a convergence failure at

extreme conditions of temperature/pressure or because the locus cannot be followed. It is therefore necessary to call MFDERR or MFOUT as appropriate to clear the error buffer even is a zero status is returned.

Arguments

Name	input/ output	type	Description
solution_type	in	Long	code specifying the type of solution required and whether the specified phase fraction is on a molar, mass or volume fraction basis or whether the calculation is a search for phase nucleation (see note 2 in the description of MFDFLA).
fixed_phase	in	String (up to 40 characters)	name of phase to be used in fixed phase fraction flash calculation, Ignored for other flashes (see note 3 in the description of MFDFLA).
fixed_phase	in	Long (ByVal)	String length of fixed phase name
flash_type	in	Long	code specifying initial variable (P or T) to be incremented and variable to be constrained (phase fraction, enthalpy, entropy, volume or internal energy). See note 1 below.
direction	in	Long	specifies whether independent variable is to be increased or decreased initially (1=increase, 0 = decrease)
initial_pressure initial_temperature	in	Double	values of pressure and temperature at initial point on phase envelope (the unknown value should be set to MF_UNDEFINED)
spec	in	Double	value of constraint (phase fraction, enthalpy, entropy, volume or internal energy depending on setting of flash_type)
no_comps	in	Long	number of components in the array amounts; this number should be consistent with the number of components defined using MFIN or MFFILE
Amounts	in	array of Double	overall number of moles or masses of each component; whether this array is interpreted as moles or masses depends on the input units specified

Stream_type	in	String (up to 40 characters)	name of Multiflash stream type to be used in calculation. If stream types have not been defined this argument should be set to an empty string
stream_type_len	in	Long (ByVal)	String length of stream type name
no_points	in	Long	number of points on phase envelope to return
n_skip	in	Long	number of points on phase envelope to skip between each reported point - this argument provides some control over the range of points returned. A value of 1 returns every point, 2 returns every second point, etc.
point_id	out	array of Long	code for each point returned in p_points and t_points (see note 2)
p_points t_points v_points h_points s_points u_points	out	array of Double	value of calculated pressure, temperature, volume, enthalpy, entropy and internal energy for each point on the phase envelope
nphout	out	array of Long	number of phases for each point on phase envelope
phfout	out	array of Double	molar phase fraction for each phase at each point on phase envelope (see note 3)

Notes

The flash_type argument may have the following values(defined in MFVBId.bas)

Flash type	initial variable	constraint
MF_PFRACF	pressure	phase fraction
MF_TFRACF	temperature	phase fraction
MF_PHF	pressure	enthalpy
MF_THF	temperature	enthalpy
MF_PSF	pressure	entropy
MF_TSF	temperature	entropy
MF_PVF	pressure	volume/density
MF_TVF	temperature	volume/density
MF_PUF	pressure	internal energy
MF_TUF	temperature	internal energy

The following point ids (defined in MFVBId.bas) may be returned:

MF_PENORMAL	normal point
MF_PECRITICAL	critical point (estimated from adjacent points on

	phase boundary)
MF_PEDISCONTINUITY	discontinuity on phase boundary where there is a change in the number or types of phases (estimated from adjacent points on phase boundary)

The phase fractions are returned as a 1-dimensional array with the values for point 1 on the phase envelope stored in elements 1 to MF_MAXPD, values for point 2 stored in elements MF_MAXPD +1 to 2*MF_MAXPD and so on. MF_MAXPD is the maximum number of phase descriptors that may be defined – the value is set in MFVBId.bas.

MFDERR

MFDERR(nerr, mxerr, ierr)

Returns the error/warning codes currently stored

Arguments

Name	input/ output	type	Description
nerr	out	Long	number of error/warning codes set by Multiflash
mxerr	in	Long	maximum number of error codes to return in ierr array (array dimension)
ierr	out	array of Long	array of Multiflash error/warning codes; the meaning of each code can be looked up in the Multiflash error help file (MFErr.hlp) or MFDEMS can be called to return the text

MFDEMS

MFDEMS(ierr, errmess, errmess_len)

Returns the message associated with an error/warning code.

Arguments

Name	input/ output	type	Description
ierr	in	Long	Multiflash error/warning code
Errmess	out	String (80 characters)	error/warning message associated with the code ierr
Errmess_len	in	Long (ByVal)	String length of error message: always set to 80

Return values

-2	the error message may not fit in Errmess output

MFDUN

MFDUN(property_id, io, unit_name, unit_name)

Returns name of unit currently set for a physical property.

Arguments

Name	input/ output	type	Description
property_id	in	Long	code defining property for which units are required - the codes are as described for MFDPRP
io	in	Long	code for type of unit: 0 = input units (*e.g.* as used in MFDFLA call) 1 = output unit (*e.g.* for values returned by MFDPRP)
unit_name	out	String (up to 12 characters)	unit name
unit_name	in	Long (ByVal)	String length of unit name

Return values

-2	the unit name may not fit in unit_name output

MFDCMP

MFDCMP(component_no, component_name, component_name_len, stream_type, streamtype_len)

Returns name of a component currently defined in Multiflash.

Arguments

Name	input/ output	type	description

component_no	in	Long	serial number of component for which name is required; the serial number refers to components as defined in Multiflash and is not related to any component i.d. numbers for databanks
component_name	out	String (up to 72 characters)	component name
component_name_len	in	Long (ByVal)	String length of component name
Stream_type	in	String (up to 40 characters)	name of Multiflash stream type for which component name is required. If stream types have not been defined this argument should be set to an empty string
Stream_type_len	in	Long (ByVal)	String length of stream type name

Return values

-2	The component name may not fit in the component_name output

MFDCMS

MFDCMS(no_components, component_names, component_names_len)

Returns number of components currently defined in Multiflash and their names (the maximum number of components is currently 50).

Arguments

Name	input/ output	type	description
no_components	out	Long	total number of component which have been defined in Multiflash
component_names	out	Array of Strings (up to 72 characters)	component names
component_names_len	in	Long (ByVal)	length of each array element

Return values

-2	The array of component names may not fit in the component_names var

MFDPDS

MFDPDS(no_PDs, PD_names, PD_names_len)

Returns number of phase descriptors (PDs) currently defined in Multiflash and their names (the maximum number of PDs is given by the constant MAX_PD in MFVBId.bas)).

Arguments

Name	input/ output	type	description
no_PDs	out	Long	total number of phase descriptors which have been defined in Multiflash
PD_names	out	Array of Strings (up to 40 characters)	PD names
PD_names_len	in	Long (ByVal)	length of each array element

Return values

-2	The array of PD names may not fit in the PD_names var

MFDPDT

MFDPDS(component_no, property, property_len, offset, property_value)

Returns a pure component constant property for one component.

Arguments

Name	input/ output	type	description
component_no	in	Long	serial number of component for which property is required
property	in	String (up to 40 characters)	name of property (see note 1)
property_len	in	Long (ByVal)	String length of property name
offset	in	Long	should be set to 0
property_value	out	Double	property value in current output units

Notes

The **`property`** argument can be one of the following character strings:

MOLECULARWEIGHT TCRIT PCRIT

VCRIT	ACENTRICFACTOR	TBOIL
HFORMATION	SSTANDARD	TMELT
HMELT	SMELT	CPMELT
VMELT	RUNIQUAC	QUNIQUAC
THLWATER	VHLWATER	DIPOLEMOMENT
PARACHOR	RADGYR	HOCASS
GFORMATION	TTRIPLE	PTRIPLE
HCOMBUSTION	V25	SOLUPAR
SOLIDSOLUPAR	ZCRIT	REFRACTINDEX
TFLASH	TAUTO	FLAMLOWER
FLAMUPPER	SPGRAVITY	REFVISCOSITY

For more information on the properties stored, see the *Multiflash Command Reference*.

MFDPTF

MFDPTF(component_no, property, property_len, temperature, property_value)

Returns a pure component temperature-dependent property for one component.

Arguments

Name	input/ output	type	description
component_no	in	Long	serial number of component for which property is required
property	in	String (up to 40 characters)	name of property (see note 1)
property_len	in	Long (ByVal)	String length of property name
temperature	in	Double	Temperature at which property is evaluated
property_value	out	Double	property value in current output units

Notes

The **`property`** argument can be one of the following character strings:

CPIDEAL	Perfect gas heat capacity Cp
CPLIQUID	Heat capacity Cp for saturated liquid

CPSOLID	Heat capacity Cp for solid
PSAT	Saturated vapour pressure
HVAP	Enthalpy change on evaporation (latent heat) for liquid on saturation line
LDENS	Saturated liquid density
LTHCOND	Thermal conductivity for saturated liquid
VTHCOND	Thermal conductivity for gas at low pressure
LVISC	Viscosity for saturated liquid
VVISC	Viscosity for gas at low pressure
STENSION	Surface tension for saturated liquid
VIRIALCOEFF	Second virial coefficient for gas

For more information on the properties stored, see the *Multiflash Command Reference*.

MFEND

MFEND( )

Closes all Multiflash files and resets internal storage. MFEND should be called when your application has finished using Multiflash. It is highly advisable to place a call to MFEND at all points where it is possible to exit from your program. This prevents opened files being left in Windows, as can happen with some applications that do not close files properly.

EXAMPLES

NUMERICAL MODE

The files EX1.vbp and EX1.frm contain an example of how to use Multiflash from Visual Basic in numerical mode. The steps that should to be followed are described below with reference to this example.

Include the file MFVBId.bas in your project using the Project, Add Module menu option. This file contains all the Multiflash function declarations described above plus definitions of the constants required to call them. This example also requires the file Paths.bas to be added to the project in a similar way.

The ‘Start’ button runs the example calculation by calling the StartMF procedure.

Call MFSTRT to start the Multiflash DLL command processor running:

```
n = 1
l_app_path = Len(APP_PATH)
status = MFSTRT(MF_MODE_VBNUMERIC, " ", n, APP_PATH, l_app_path, "
", n)
Call ChkStatus(status, "MFSTRT")
```

APP_PATH is defined in the Paths.bas module. It is used to search for the databank files and other Multiflash support files. The value of this constant should be amended so that it contains the correct directory for your installation. The length of the APP_PATH string must be obtained and passed to MFSTRT. No password or usr_path values are used so these arguments are set to a single blank character. The ChkStatus procedure (see below) checks the return value from the call and displays a message if the call was not successful.

The file pr.mfc is one of the standard Multiflash model configuration files provided. It sets up the PR eos model for fluid phases. The call to MFFILE makes Multiflash process this file:

```
status = MFFILE(MFC_PATH & "Pr.mfc", n)
```

MFC_PATH is defined in the Paths.bas module. It is used in this example to search for the Multiflash configuration files. The value of this constant should be amended so that it contains the correct directory for your installation.

The model configuration files do not define the components in the mixture. This should be done with a command like

```
Call MFIN_VB("components methane, ethane;", status)
```

this defines two components taken from the Infodata databank (which is specified in pr.mfc).

The routine MFIN_VB is used to pass the command string and its length to Multiflash:

```
Private Sub MFIN_VB( inputtext As String, status As Long )
  Dim lentext As Long
  lentext = Len(inputtext) + 1
  status = MFIN(inputtext, lentext)
End Sub
```

The units used for physical properties can be set with a command like

```
Call MFIN_VB("units amount mole temp K pres bar;", status)
```

which sets the input and output units for component amounts, temperature and pressure. For more information on the default units and unit settings see the *Multiflash Command Reference*.

The component amounts can now be set and the call to MFDFLA carries out the flash calculation. The call shown below is an isothermal, isobaric flash at 10 bar and 200K for a mixture containing 0.4 moles of methane and 0.6 moles of ethane:

```
nmol(1) = 0.4
nmol(2) = 0.6
status = MFDFLA(MF_PTF, MF_NORMAL, "", n, 10#, 200#, 0#, 2, nmol(1),
"", n)
```

Note that in Visual Basic an array reference is passed as a reference to its first element.

Call MFDPRP to obtain the physical properties generated by the previous flash calculation. For example, the following call obtains the overall mixture enthalpy and volume:

```
iprop(1) = MF_ENTHALPY
iprop(2) = MF_VOLUME
icomp (1) = 0
icomp (2) = 0
status = MFDPRP("OVERALL", n, 2, iprop(1), icomp(1), prop(1))
```

MFDPRP may be called as often as required to obtain any of the physical properties. Once all the required properties have been collected, a new flash can then be performed by calling MFDFLA and the process of obtaining the physical properties repeated.

Whenever any of the functions returns a non-zero value, the error codes should always be obtained by calling MFDERR. The example uses the procedure ChkStatus for this purpose. The essentials of the code are:

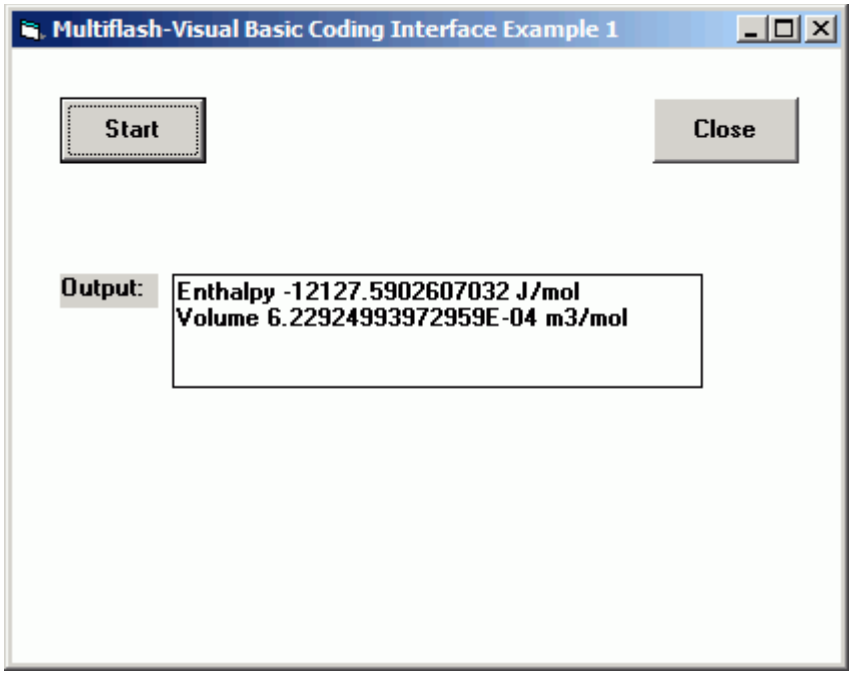
```
If status <> 0
  msg = "Status " & CStr(status) & " returned by " & RoutineName
  lstatus = MFDERR(NErr, Mxerr, Ierr(1))
  If NErr > 0 Then
    Errmes = Space$(80)
    For i = 1 To NErr
     lstatus = MFDEMS(Ierr(i), Errmes, n)
    Next
    msg = msg & vbCrLf & Errmes
  End If
  MsgBox msg
End If
```

The error numbers returned by MFDERR are used to retrieve the associated error messages in MFDEMS. All the messages are concatenated for display in a message

box.

Before ending the program the function MFEND is called when the ‘Close’ button is clicked.

Output. The example produces the following output



TEXT MODE (NOT RECOMMENDED)

The files EX2.vbp and EX2.frm contain an example of how to use Multiflash from Visual Basic in text mode. The steps that should to be followed are described below with reference to this example.

Include the files MFVBId.bas and Paths.bas in your project (see above).

The ‘Start’ button runs the example calculation by calling the StartMF procedure.

Call MFSTRT to start the Multiflash DLL command processor running:

```
n = 1
l_app_path = Len(APP_PATH)
status = MFSTRT(MF_MODE_VBTEXT, " ", n, APP_PATH, l_app_path, " ",
n)
Call ChkStatus(status, "MFSTRT")
```

The path APP_PATH (defined in Path.bas) is used to search for the databank files. The ChkStatus procedure (see below) checks the return value from the call and displays all output from Multiflash.

The model, components and units are defined just as in the previous example:

```
status = MFFILE(MFC_PATH & "Pr.mfc", n)
Call MFIN_VB("components methane, ethane;", status)
Call MFIN_VB("units amount mole temp K pres bar;", status)
```

As in the first example the routine MFIN_VB is used to pass the command string and its length.

In text mode the component amounts and flash conditions are set using MFIN:

```
Call MFIN_VB("amount 0.4, 0.6; temp 200; press 10;", status)
```

The flash calculation at fixed pressure and temperature is triggered by another call to MFIN:

```
Call MFIN_VB("ptflash;", status)
```

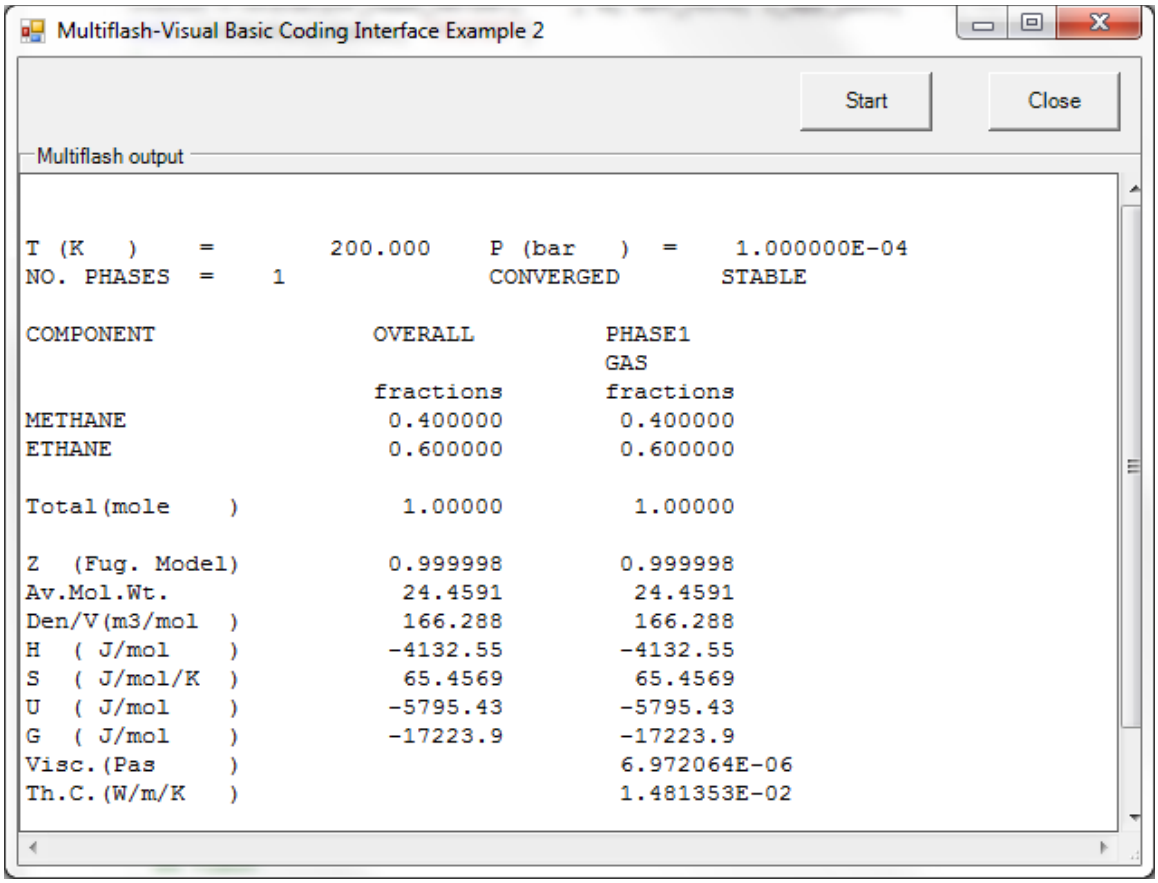
Whenever any of the functions returns a non-zero value, the error codes should always be obtained in text mode by calling MFOUT. The example uses the procedure ChkStatus to check the return status for each function call and to display any output in a scrolling text window. The essentials of the code are:

```
If status <> 0 Then
'
' form status message
'
  msg = "Status " & CStr(status) & " returned by " & RoutineName
  MsgBox msg
End If
'
' collect output from MF
'
outputtext = Space$(MF_MAXVBO)
lstatus = MFOUT(outputtext, n)
n = Len(outputtext)
i = InStr(outputtext, vbNullChar)
If n < i Then i = n
If i > 1 Then
  txtMFOutput = txtMFOutput & vbCrLf & Left(outputtext, i - 1)
'
'  Move cursor to end of output window
'
  txtMFOutput.SelStart = Len(txtMFOutput)
End If
```

Note that before output is placed in ‘outputtext’ it is essential to ensure that storage has been created to accept the output up to the internal Multiflash limit of MF_MAXVBO characters, hence the call of ‘Space$(MF_MAXVBO)’. The output is in text format as it would appear on the screen in the interactive command-driven version of Multiflash. All line breaks in ‘outputtext’ are marked by a line feed character immediately followed by a carriage return character. The end of output from a call to MFOUT is marked by a null character. In the example above this is located using the VB function InStr and searching for the vbNullChar character.

Note - in text mode MFOUT should always be called after MFIN or MFFILE regardless of whether Multiflash is in an error state or not.

Output. This example produces the following output.



VB.NET EXAMPLES

All the previously described examples can also be compiled in VB.NET. Project files are supplied in the Multiflash installation directory under API\VB\.