

# New and Enhanced MapBasic Statements and Functions

These are the new statements and functions available for the MapInfo Professional 9.0 product.

## Sections in this Chapter:

- ♦ **New in MapBasic** .....2
- ♦ **Enhanced Functions and Statements** .....12

## New in MapBasic

We have added a new set of date and time data types which increase your ability to create date-based and time-based queries and thematics.

The following new functions have been added to MapBasic for the current release of MapInfo Professional. These functions are included in their place in the logical order of the reference section. We are duplicating them here for your reference.

- **CoordSysName\$( ) function**
- **CurDateTime function**
- **CurTime function**
- **FormatTime\$ function**
- **FME Refresh Table statement**
- **GetDate function**
- **GetTime function**
- **HotlinkInfo function**
- **Hour function**
- **MakeDateTime function**
- **Minute function**
- **NumberToDateTime function**
- **NumberToTime function**
- **RegionInfo( ) function**
- **Second function**
- **StringToDateTime function**
- **StringToTime function**

---

### CoordSysName\$( ) function

#### Purpose

Returns coordinate system name string from MapBasic Coordinate system clause.

#### Syntax

```
CoordSysName$ ( string )
```

#### Return Value

String

#### Example

```
Note CoordSysName$("Coordsys Earth Projection 1, 62")
```

Returns this string in the MapInfo dialog box:

```
Longitude / Latitude (NAD 27 for Continental US)
```

**Note:** If a coordinate system name does not exist in the MapInfo.prj file, such as when the map is in NonEarth system in Survey Feet, then function will return an empty string.

```
Note CoordSysName$("CoordSys NonEarth Units " + ""survey ft"" +  
"Bounds (0, 0) (10, 10)")
```

If an invalid CoordSys clause is passed such as this (using invalid units):

```
Note CoordSysName$("CoordSys Earth Projection 3, 74, " + ""foo"" +  
"-90, 42, 42.7333333333, 44.0666666667, 1968500, 0")
```

Then an Error regarding the Invalid Coordinate System should be returned (Error #727).

```
Invalid Coordinate System: CoordSys Earth Projection <content>
```

---

## CurDateTime function

### Purpose

Returns the current date and time.

### Syntax

```
CurDateTime
```

### Return Value

DateTime

### Example

Copy this example into the MapBasic window for a demonstration of this function.

```
dim X as datetime  
X = CurDateTime()  
Print X
```

---

## CurTime function

### Purpose

Returns the current time.

### Syntax

```
CurTime
```

### Return Value

Time

## FormatTime\$ function

---

### Example

Copy this example into the MapBasic window for a demonstration of this function.

```
dim Y as time
Y = CurTime()
Print Y
```

---

## FormatTime\$ function

### Purpose

Returns a string representing a time using the format specified in the second argument. The format string should follow the same Microsoft standards as for setting the locale time format:

Hours	Meaning
h	Hours without leading zeros for single-digit hours (12-hour clock).
hh	Hours with leading zeros for single-digit hours (12-hour clock).
H	Hours without leading zeros for single-digit hours (24-hour clock).
HH	Hours with leading zeros for single-digit hours (24-hour clock).
Minutes	Meaning
m	Minutes without leading zeros for single-digit minutes.
mm	Minutes with leading zeros for single-digit minutes.
Seconds	Meaning
s	Seconds without leading zeros for single-digit seconds.
ss	Seconds with leading zeros for single-digit seconds.
Time marker	Meaning
t	One-character time marker string. <b>Note:</b> Do not to use this format for certain languages, for example, Japanese (Japan). With this format, the application always takes the first character from the time marker string, defined by LOCALE_S1159 (AM) and LOCALE_S2359 (PM). Because of this, the application can create incorrect formatting with the same string used for both AM and PM.
tt	Multi-character time marker string.

Source: <http://msdn2.microsoft.com/en-us/library/ms776320.aspx>

**Note:** In the preceding formats, the letters m, s, and t must be lowercase, and the letter h must be lowercase to denote the 12-hour clock or uppercase to denote the 24-hour clock.

Our code follows the rules for specifying the system local time format. In addition, we also allow the user to specify f, ff, or fff for tenths of a second, hundredths of a second, or milliseconds.

### Syntax

```
FormatTime$ (Time, String)
```

### Return Value

String

### Example

Copy this example into the MapBasic window for a demonstration of this function.

```
dim Z as time
Z = CurTime()
Print FormatTime$(Z, "hh:mm:ss.fff tt")
```

---

## FME Refresh Table statement

### Purpose

Refreshes a Universal Data Source (FME) table from the original data source

### Syntax

```
FME Refresh Table alias
```

*alias* is the an alias for an open registered Universal Data Source (FME) table.

### Example

The following example refreshes the local table named watershed.

```
FME Refresh Table watershed
```

---

## GetDate function

### Purpose

Returns the Date component of a DateTime.

### Syntax

```
GetDate (DateTime)
```

### Return Value

Date

## GetTime function

---

### Example

Copy this example into the MapBasic window for a demonstration of this function.

```
dim dtX as datetime
dim Z as date
dtX = "03/07/2007 12:09:09.000 AM"
Z = GetDate(dtX)
Print FormatDate$(Z)
```

---

## GetTime function

### Purpose

Returns the Time component of a DateTime.

### Syntax

```
GetTime (DateTime)
```

### Return Value

Time

### Example

Copy this example into the MapBasic window for a demonstration of this function.

```
dim dtX as datetime
dim Z as time
dtX = "03/07/2007 12:09:09.000 AM"
Z = GetTime(dtX)
Print FormatTime$(Z, "hh:mm:ss.fff tt")
```

---

## HotlinkInfo function

### Purpose

Returns information about a HotLink definition in a map layer.

### Syntax

```
HotlinkInfo ( map_window_id, layer_number, hotlink_number, attribute )
```

*map\_window\_id* is a Map window identifier.

*layer\_number* is the number of a layer in the current Map window (for example, 1 for the top layer); to determine the number of layers in a Map window, call the *MapperInfo()* function.

*hotlink\_number* - the index of the hotlink definition being queried. The first hotlink definition in a layer has index of 1.

*attribute* - the following attribute values are allowed:

HOTLINK_INFO_EXPR	Returns the filename expression for this hotlink definition.
HOTLINK_INFO_MODE	Returns the mode for this hotlink definition, one of the following predefined values: <ul style="list-style-type: none"><li>• HOTLINK_MODE_LABEL</li><li>• HOTLINK_MODE_OBJ</li><li>• HOTLINK_MODE_BOTH</li></ul>
HOTLINK_INFO_RELATIVE	Returns TRUE if the relative path option is on for this hotlink definition.
HOTLINK_INFO_ENABLED	Returns TRUE if this hotlink definition is enabled.

---

## Hour function

### Purpose

Returns the hour component of a Time.

### Syntax

```
Hour (Time)
```

### Return Value

Number

### Example

Copy this example into the MapBasic window for a demonstration of this function.

```
dim Z as time
dim iHour as integer
Z = CurDateTime()
iHour = Hour(Z)
Print iHour
```

---

## MakeDateTime function

### Purpose

Returns a DateTime made from the specified Date and Time.

### Syntax

```
MakeDateTime (Date, Time)
```

## Minute function

---

### Return Value

DateTime

### Example

Copy this example into the MapBasic window for a demonstration of this function.

```
dim tX as time
dim dX as date
dim dtX as datetime
tX = 105604123
dX = 20070908
dtX = MakeDateTime(dX,tX)
Print FormatDate$(GetDate(dtX))
Print FormatTime$(GetTime(dtX), "hh:mm:ss.fff tt")
```

---

## Minute function

### Purpose

Returns the minute component of a Time.

### Syntax

```
Minute (Time)
```

### Return Value

Number

### Example

Copy this example into the MapBasic window for a demonstration of this function.

```
dim X as time
dim iMin as integer
X = CurDateTime()
iMin = Minute(X)
Print iMin
```

---

## NumberToDateTime function

### Purpose

Returns a DateTime value.

### Syntax

```
NumberToDateTime( numeric_datetime )
```

*numeric\_datetime* is an seventeen-digit integer in the form YYYYMMDDHHMMSSFFF. For example, 20070301214237582 represents March 1, 2007 9:42:37.582 PM.

### Return Value

Date/Time

### Example

Copy this example into the MapBasic window for a demonstration of this function.

```
dim fNum as float
dim Y as datetime
fNum = 20070301214237582
Y = NumberToDateTime (fNum)
Print FormatDate$(Y)
Print FormatTime$(Y, "hh:mm:ss.fff tt")
```

---

## NumberToTime function

### Purpose

Returns a Time value.

### Syntax

```
NumberToTime( numeric_time )
```

*numeric\_time* is an nine-digit integer in the form HHMMSSFFF. For example, 214237582 represents 9:42:37.582 P.M.

### Return Value

Time

### Example

Copy this example into the MapBasic window for a demonstration of this function.

```
dim fNum as float
dim Y as time
fNum = 214237582
Y = NumberToTime (fNum)
Print FormatTime$(Y, "hh:mm:ss.fff tt")
```

---

## RegionInfo( ) function

### Purpose:

This function was created to determine the orientation of points in polygons -- whether they are ordered clockwise, or counter-clockwise. The only attribute the function reports on is the 'direction' of the points in a specified polygon.

### Syntax:

```
RegionInfo(object, REGION_INFO_IS_CLOCKWISE, polygon_num)
```

## Second function

---

Only one parameter exists for this function:

```
REGION_INFO_IS_CLOCKWISE 1
```

### Example:

If you were to select the state of Utah from States mapper and issued the following command in the MapBasic window, you would get a result of `F` or `False`, since the nodes in the single region of Utah are drawn in counter-clockwise order. Colorado's nodes are drawn in clockwise order and return `T` or `True`.

```
print RegionInfo(selection.obj,1,1)
```

---

## Second function

### Purpose

Returns the second and millisecond component of a `Time` as a floating-point number.

### Syntax

```
Second (Time)
```

### Return Value

Number

### Example

Copy this example into the MapBasic window for a demonstration of this function.

```
dim X as time
dim iSec as integer
X = CurDateTime()
Sec = Second(X)
Print iSec
```

---

## StringToDateTime function

### Purpose

Returns a `DateTime` value given a string that represents a date and time. MapBasic interprets the date/time string according to the date and time-formatting options that are set up on the user's computer. At least one space must be between the date and time. See *StringToDate()* function and *StringToTime* function for details.

### Syntax

```
StringToDateTime (String)
```

### Return Value

`DateTime`

### Example

Copy this example into the MapBasic window for a demonstration of this function.

```
dim strX as string
dim Z as datetime
strX = "19990912041345789"
Z = StringtoDateTime(strX)
Print FormatDate$(Z)
Print FormatTime$(Z,"hh:mm:ss.fff tt")
```

---

## StringToTime function

### Purpose

Returns a Time value given a string that represents a time. MapBasic interprets the time string according to the time-formatting options that are set up on the user's computer. However, either 12 or 24-hour time representations are accepted. In addition, the less significant components of a time may be omitted. In other words, the hour must be specified, but the minutes, seconds, and milliseconds are optional.

### Syntax

```
StringToTime (String)
```

### Return Value

Time

### Example

Copy this example into the MapBasic window for a demonstration of this function.

```
dim strY as string
dim X as time
strY = "010203000"
X = StringtoTime(strY)
Print FormatTime$(X,"hh:mm:ss.fff tt")
```

## Enhanced Functions and Statements

The following functions and statements have been enhanced to provide additional functionality to :MapBasic:

- **Commit Table statement**
- **GeocodeInfo( ) function**
- **IsogramInfo( ) function**
- **LabelInfo( ) function**
- **LayerInfo( ) function**
- **Register Table statement**
- **Server Create Table statement**
- **Set Map statement**
- **SystemInfo() function**
- **TableInfo( ) function**

---

### Commit Table statement

#### Purpose

Saves recent edits to disk, or saves a copy of a table. The new clause contains datetime option.

#### Syntax

```
Commit Table table
  [ As filespec
    [ Type { NATIVE |
            DBF [ Charset char_set ] |
            Access Database database_filespec
              Version version
              Table tablename
              [ Password pwd ] [ Charset char_set ] |
            QUERY |
            ODBC Connection ConnectionNumber Table tablename
              [ ConvertDateTime {ON | OFF | INTERACTIVE}} ] } ]
    [ CoordSys... ]
    [ Version version ] ]
  [ { Interactive | Automatic commit_keyword } ]
  [ ConvertObjects {ON | OFF | INTERACTIVE}} ]
```

*tableName* is the name of the table as you want it to appear in database. Starting release 9.0, the name can include a schema name which specifies the schema that the table belongs to. If no schema name is provided, the table belongs to the default schema as in 8.5 and earlier versions. The user is responsible for providing an eligible schema name and must know if the login user has the proper permissions on the given schema. This extension is for SQL Server 2005 only.

*ConvertDateTime* If the source table contains Time or Date type columns, these columns will be converted to DATETIME or TIMESTAMP depending on whether the server supports the data types. However, you can control this behavior using the clause *ConvertDateTime*. If the source table does

not contain a Time or Date type, this clause is a non-operational. If *ConvertDateTime* is set to ON (which is the default setting), Time or Date type columns will be converted to DATETIME or TIMESTAMP. If *ConvertDateTime* is set to OFF, the conversion is not done and the operation will be cancelled if necessary. If *ConvertDateTime* is set to INTERACTIVE a dialog box will pop up to prompt the user and the operation will depend on the user's choice. If the user chooses to convert, then the operation will convert and continue; if the user chooses to cancel, the operation will be cancelled.

The Time type requires conversion for all supported servers (Oracle, IBM Informix, MS SQL Server and Access) and the Date type requires conversion for MS SQL Server and Access database servers.

**Note:** For MS SQL Server and Access database servers, this restriction could be an backward compatibility issue. In previous releases, we did the conversion without explaining it. In this release, we suggest you use the DateTime data type instead of Date data type. If you still use the Date data type, the conversion operation will fail.

*ConvertObjects ON* automatically converts any unsupported objects encountered in supported objects.

*ConvertObjects OFF* This does not convert any unsupported objects. If they are encountered, an error message is displayed saying the table can not be saved. (Before implementation of this feature this was the only behavior.)

*ConvertObjects Interactive* If any unsupported objects are encountered in a table, ask the user what she wants to do.

### Example 1

```
Commit Table DATETIME90 As "D:\MapInfo\Data\Remote\DATETIME90CPY.TAB"
Type ODBC Connection 1 Table ""EAZYLOADER"."DATETIME90CPY""
ConvertDateTime Interactive
```

### Example 2

```
Server 1 Create Table ""EAZYLOADER"."CITY_125AA"" (Field1
Char(10),Field2 Char(10),Field3 Char(10),MI_STYLE Char(254)) KeyColumn
SW_MEMBER ObjectColumn SW_GEOMETRY
or
Server 1 Create Table "EAZYLOADER.CITY_125AA" (Field1 Char(10),Field2
Char(10),Field3 Char(10),MI_STYLE Char(254)) KeyColumn SW_MEMBER
ObjectColumn SW_GEOMETRY
```

```
Commit Table City_125aa As
"C:\Projects\Data\TestScripts\English\remote\City_125aacpy.tab" Type ODBC
Connection 1 Table ""EAZYLOADER"."CITY_125AACPY""
or
Commit Table City_125aa As
"C:\Projects\Data\TestScripts\English\remote\City_125aacpy.tab" Type ODBC
Connection 1 Table "EAZYLOADER.CITY_125AACPY"
```

### GeocodeInfo( ) function

#### Purpose

We have added a new attribute to this function to handle the maximum number of addresses that the server will permit to be sent to the service at a time.

#### Syntax

```
GeoCodeInfo( connection_handle, attribute )
```

*connection\_handle* is an Integer.

*attribute* is an Integer code, indicating which type of information should be returned.

#### Return Value

Float, Integer, SmallInt, Logical, or String, depending on the attribute parameter.

#### Description

The **GeoCodeInfo( )** function returns the properties defaulted by the connection or the properties that have been changed using Set GeoCode. Like many functions of this type in MapBasic, the return values vary according to the *attribute* parameter. All the codes for these values are listed in MAPBASIC.DEF.

#### GEOCODE\_MAX\_BATCH\_SIZE

Integer value representing the maximum number of records (i.e., addresses) that the server will permit to be sent to the service at one time.

#### Example

The following MapBasic snippet will print the Envinsa Location Utility Constraints to the message window in MapInfo Professional:

```
Include "MapBasic.Def"
declare sub main
sub main
dim iConnect as integer

Open Connection Service Geocode Envinsa
  URL
"http://envinsa_server:8066/LocationUtility/services/LocationUtility"
  User "john"
  Password "green"
  into variable iConnect

Print "Geocode Max Batch Size: " +
GeoCodeInfo(iConnect,GEOCODE_MAX_BATCH_SIZE)
end sub
```

## IsogramInfo( ) function

### Purpose

New attributes have been added to this function to handle the maximum number of records for server, time, and distance values.

### Syntax

`IsogramInfo( connection_handle, attribute )`

*connection\_handle* is an integer signifying the number of the connection returned from the *Open Connection* statement.

*attribute* is an Integer code, indicating which type of information should be returned.

### Return Value

Float, Logical, or String, depending on the *attribute* parameter.

### Description

This function returns the properties defaulted by the connection or the properties that have been changed using the [Set Connection Isogram statement](#).

There are several attributes that **IsogramInfo( )** can return. Codes are defined in MAPBASIC.DEF.

Attribute Setting	IsogramInfo Return Value
ISOGRAM_MAX_BATCH_SIZE	Integer value representing the maximum number of records (i.e., points) that the server will permit to be sent to the service at one time.
ISOGRAM_MAX_BANDS	Integer value representing the maximum number of Iso bands (i.e., distances or times) allowed.
ISOGRAM_MAX_DISTANCE	Float value representing the maximum distance permitted for an Isodistance request. The distance units are specified by ISOGRAM_MAX_DISTANCE_UNITS.
ISOGRAM_MAX_DISTANCE_UNITS	String value representing the units for ISOGRAM_MAX_DISTANCE.
ISOGRAM_MAX_TIME	Float value representing the maximum time permitted for an Isochrone request. The time units are specified by ISOGRAM_MAX_TIME_UNITS.
ISOGRAM_MAX_TIME_UNITS	String value representing the units for ISOGRAM_MAX_TIME.

## LabelInfo( ) function

---

### Example

The following MapBasic snippet will print the Envinsa Routing Constraints to the message window in MapInfo Professional:

```
Include "MapBasic.Def"
declare sub main
sub main
dim iConnect as integer
Open Connection Service Isogram
    URL "http://envinsa_server:8062/Route/services/Route"
    User "john"
    Password "green"
    into variable iConnect

Print "Isogram_Max_Batch_Size: " +
IsogramInfo(iConnect,Isogram_Max_Batch_Size)
Print "Isogram_Max_Bands: " + IsogramInfo(iConnect, Isogram_Max_Bands)
Print "Isogram_Max_Distance: " + IsogramInfo(iConnect,
Isogram_Max_Distance)
Print "Isogram_Max_Distance_Units: " + IsogramInfo(iConnect,
Isogram_Max_Distance_Units)
Print "Isogram_Max_Time: " + IsogramInfo(iConnect,Isogram_Max_Time)
Print "Isogram_Max_Time_Units: " +
IsogramInfo(iConnect,Isogram_Max_Time_Units)
Close Connection iConnect
end sub
```

---

## LabelInfo( ) function

### Purpose

Returns information about a label in a map. LabelInfo can return a label as a text object. However, if the label is curved, it will be returned as rotated flat text.

### Syntax

```
LabelInfo( map_window_id, layer_number, attribute )
```

### Description

The attribute parameter must be one of the codes from the following table; codes are defined in MAPBASIC.DEF.

attribute code	Labelinfo( ) Return Value
LABEL_INFO_ORIENTATION	<p>Returns Smallint value indicating the 'current' label's orientation. The current label is initialized by using one of the following Label functions: LabelFindFirst, LabelFindByID, or LabelFindNext. The Return value will be one of these:</p> <ul style="list-style-type: none"> <li>• LAYER_INFO_LABEL_ORIENT_HORIZONTAL (label has angle equal to 0)</li> <li>• LAYER_INFO_LABEL_ORIENT_PARALLEL (label has non-zero angle)</li> <li>• LAYER_INFO_LABEL_ORIENT_CURVED (label is curved)</li> </ul>

## LayerInfo( ) function

### Purpose of the New Attributes

New attribute value LAYER\_INFO\_HOTLINK\_COUNT will allow programmers to query the number of hotlink definitions in a layer.

For backwards compatibility, the original set of attributes will still work, and will return the values for the layer's first hotlink definition. If no hotlinks are defined when the function is called, then the following values are returned:

LAYER\_INFO\_HOTLINK\_EXPR - empty string ("")

LAYER\_INFO\_HOTLINK\_MODE - returns default value HOTLINK\_MODE\_LABEL

LAYER\_INFO\_HOTLINK\_RELATIVE - returns default value FALSE

New Attribute:

LAYER\_INFO\_LABEL\_ORIENTATION - Returns Smallint value indicating the setting for the layer's auto label orientation. Return value will be one of these values:

LAYER\_INFO\_LABEL\_ORIENT\_HORIZONTAL (labels have angle equal to 0)

LAYER\_INFO\_LABEL\_ORIENT\_PARALLEL (labels have non-zero angle)

LAYER\_INFO\_LABEL\_ORIENT\_CURVED (labels are curved)

**Note:** If LAYER\_INFO\_LABEL\_ORIENT\_PARALLEL is returned, then LAYER\_INFO\_LABEL\_PARALLEL will return TRUE.

## Register Table statement

### Purpose

Builds a MapInfo Professional table from a spreadsheet, database, text file, raster, or grid image.

### Syntax

```
Register Table source_file
{ Type "NATIVE" |
  Type "DBF" [ Charset char_set ] |
  Type "ASCII" [ Delimiter delim_char ][ Titles ][ CharSet char_set ] |
  Type "WKS" [ Titles ] [ Range range_name ] |
  Type "WMS" Coordsys...
  Type "WFS" [ Charset char_set ] Coordsys... [ Symbol... ]
    [ Linestyle Pen(...) ] [ Regionstyle Pen(...) Brush(...) ]
  Type "XLS" [ Titles ] [ Range range_name ] [ Interactive ] |
  Type "Access" Table table_name [ Password pwd ] [ CharSet char_set ] }
Type ODBC
  Connection { Handle ConnectionNumber | ConnectionString }
  Toolkit toolkitname
  Cache { ON | OFF }
  [ Autokey { ON | OFF } ]
  Table SQLQuery
  [ Versioned { ON | OFF } ]
  [ Workspace WorkspaceName ]
  [ ParentWorkspace ParentWorkspaceName ]
Type "GRID" | Type "RASTER"
[ ControlPoints ( MapX1, MapY1 ) ( RasterX1, RasterY1 ),
  ( MapX2, MapY2 ) ( RasterX2, RasterY2 ),
  ( MapX3, MapY3 ) ( RasterX3, RasterY3 )
  [, ... ]
]
[ CoordSys ... ]
Type "FME" [ Charset char_set ]
  CoordSys...
  Format format type
  Schema featuretype
  [ Use Color ]
  [ Database ]
  [ SingleFile ]
  [ Symbol... ]
  [ Linestyle Pen(...) ]
  [ Regionstyle Pen(...) Brush(...) ]
  [ Font ... ]
  Settings string1 [, string2 .. ]
Type "SHAPEFILE" [ Charset char_set ] CoordSys...
  [ PersistentCache { ON | OFF } ]
  [ Symbol... ] [ Linestyle Pen(...) ]
  [ Regionstyle Pen(...) Brush(...) ]
  [ Into destination_file ]
```

*Charset char\_set* - this is optional parameter. If character set is not specified in MapBasic, then system character set is used. This is similar to all other formats.

*CoordSys...* - CoordSys clause is required.

*Format formattype* - formattype is a string that is used by FME to identify format that is opened.

*Schema featuretype* - specifies a featuretype (essentially schema name).

*Settings string1 [, string2 .. ]* - These are Safe Software FME-specific settings that vary depending upon the format and settings options the user selects.

*Use Color* - specifies if color information from dataset is used

*Database* - specifies if referenced data source is from a database

*SingleFile* - specifies if referenced data source consist of a single file

### Example

```
Register Table "D:\MUT\DWG\Data\afrika_miller.DWG" Type "FME" CoordSys
Earth Projection 11, 104, "m", 0 Format "ACAD" Schema "afrika_miller" Use
Color SingleFile Symbol (35,0,16) Linestyle Pen (1,2,0) RegionStyle Pen
(1,2,0) Brush (2,16777215,16777215) Font ("Arial",0,9,0) Settings
"RUNTIME_MACROS","METAFILE,acad,_EXPAND_BLOCKS,yes,ACAD_IN_USE_BLOCK_HEAD
ER_LAYER,yes,ACAD_IN_RESOLVE_ENTITY_COLOR,yes,_EXPAND_VISIBLE,yes,_BULGES
_AS_ARCS,no,_STORE_BULGE_INFO,no,_READ_PAPER_SPACE,no,ACAD_IN_READ_GROUPS
,no,_IGNORE_UCS,no,_ACADPreserveComplexHatches,no,_MERGE_SCHEMAS,YES",
"META_MACROS","Source_EXPAND_BLOCKS,yes,SourceACAD_IN_USE_BLOCK_HEADER_LA
YER,yes,SourceACAD_IN_RESOLVE_ENTITY_COLOR,yes,Source_EXPAND_VISIBLE,yes,
Source_BULGES_AS_ARCS,no,Source_STORE_BULGE_INFO,no,Source_READ_PAPER_SPA
CE,no,SourceACAD_IN_READ_GROUPS,no,Source_IGNORE_UCS,no,Source_ACADPreser
veComplexHatches,no", "METAFILE","acad", "COORDSYS","", "IDLIST","" Into
"C:\Temp\afrika_miller.tab"
Open table "C:\Temp\afrika_miller.tab"
Map From "afrika_miller"
```

---

## Server Create Table statement

There is no specific change in terms of syntax. We do have following restrictions for the some data types:

In release 9.0, there are two new MapInfo date types, Time and DateTime, which were added to meet customers needs. However, most databases do not have a corresponding DBMS TIME types. Before this release, we only supported the Date type. Even the Date was converted to server type If the server did not support Date type. Starting 9.0, this statement only supports the types that the server also supports. Therefore, the Time type is prohibited from this statement for all supported servers (such as Oracle, IBM Informix, MS SQL Server and Access) and the Date type data type is prohibited for MS SQL Server and Access. Those "unsupported" types should be replaced with DateTime if you still want to create the table that contains time information on a column.

**Note:** For MS SQL Server and Access, the restriction could be a backward compatibility issue. In the previous release, we did the conversion in the background. As of 9.0, the users must choose to use DATETIME instead of DATE. If you still use DATE, the operation will fail.

## Set Map statement

---

```
Server ConnectionNumber Create Table TableName(ColumnName ColumnType [,...])
    [KeyColumnn ColumnName]
    [ObjectColumn ColumnName]
    [StyleColumn ColumnName]
    [CoordSys... ]
```

*TableName* is the name of the table as you want it to appear in database. Starting release 9.0, the name can include a schema name which specifies the schema that the table belongs to. If no schema name is provided, the table belongs to the default schema as in 8.5 and earlier versions. The user is responsible for providing an eligible schema name and must know if the login user has the proper permissions on the given schema. This extension is for SQL Server 2005 only.

---

## Set Map statement

### LabelClause Change

This change allows for the creation of curved labels.

#### Syntax

Each LABELCLAUSE has the syntax described below:

```
[ Label [ Line { Simple | Arrow | None } ]
[ Position [ Center ] [ Above | Below ] [ Left | Right ] ] ]
[ Font... ] [ Pen... ]
[ With label_expr ]
[ Parallel { On | Off } | Follow Path ]
[ Visibility { On | Off | Zoom( min_vis, max_vis ) [ Units dist_unit ] } ]
[ Auto [ { On | Off } ] ]
[ Overlap [ { On | Off } ] ]
[ PartialSegments { On | Off } ]
[ Duplicates [ { On | Off } ] ]
[ Max [ number_of_labels ] ]
[ Offset offset_amount ]
[ Default ]
[ Object ID
    [ Table alias ]
    [ Visibility { On | Off } ]
    [ Anchor ( anchor_x, anchor_y ) ]
    Text text_string
    [ Position [ Center ] [ Above | Below ] [ Left | Right ] ]
    [ Font... ] [ Pen... ]
    [ Line { Simple | Arrow | None } ]
    [ Angle text_angle | Follow Path ]
    [ Offset offset_amount ]
    [ Callout ( callout_x, callout_y ) ] ]
[ Object... ]
]
```

*Parallel* This parameter contains the following attributes:

*Parallel Off* = creates horizontal labels, not rotated with line

*Parallel On* = creates labels rotated with line

*Follow Path* clause = creates curved label, with the path auto calculated once and stored until location edited

### Example

```
Set Map Layer 1 Label Follow Path
```

## Layer Activate clause

The hotlink settings are currently persisted via Set Map Layer Activate, which has been expanded to support a multiple hotlink definitions. This includes the ability to add new items, modify the attribute of existing items, remove and reorder items. Below is a description of the different "flavors" of the Activate clause, including syntax details. See the [Exceptions to Support Backwards Compatibility](#) section for a discussion of how MapBasic supports legacy syntax.

Note that the properties of an individual hotlink are the same as in prior version, except for the new Enable clause, which allow the user to "turn off" a hotlink while preserving the definition. (In a prior version the user disabled the hotlink by setting the expression to "", losing the original expression.)

### Purpose

The purpose of Activate is to allow you to define new hotlinks. You use a hotlink to launch a file or a URL from a Map window.

### Syntax

```
{ Using launch_expr[ On { [ [ Labels ] | [ Objects ] ] } ] | [ Relative  
Path { On | Off } ] [ Enable { On | Off } ] },  
[ Using launch_expr[ On { [ [ Labels ] | [ Objects ] ] } ] | [ Relative  
Path { On | Off } ] [ Enable { On | Off } ] ]...
```

### Example

```
Set Map Layer 1 Activate Using Url1 On Objects Relative Path Off Enable  
On, Using Url2 On Objects Relative Path On Enable On
```

### Notes

This version of the command wipes out any existing definitions and one or more new definitions. The Using clause is required and launch\_expr must not be an empty string (for example, ""). When the Enable clause is included and set to Off, the hotlink definition will be disabled. The On, Relative and Enable clauses are optional.

## Adding New HotLink Definitions

### Syntax

```
Activate Add [ First ]
{ Using launch_expr[ On { [ [ Labels ] | [ Objects ] ] } ] | [ Relative
  Path { On | Off } ] [ Enable { On | Off } ] },
[ Using launch_expr[ On { [ [ Labels ] | [ Objects ] ] } ] | [ Relative
  Path { On | Off } ] [ Enable { On | Off } ]...
```

### Examples

```
Activate Add Using URL1 On Objects Relative Path On, Using URL2 On Objects
Enabled Off
```

```
Activate Add First Using URL1 On Objects
```

### Comments

The MapBasic 9.0 version of this command adds a new hotlink def at the end of the hotlink list.

It can include the optional *First* keyword to insert the new items at the beginning of the list.

When the *Enable* clause is included and set to *Off*, the hotlink definition will be disabled.

The *On*, *Relative* and *Enable* clauses are optional.

## Modifying Existing HotLinks Definitions

### Syntax

```
Activate Modify
{ hotlink_id[ Using launch_expr ] | [ On { [ [ Labels ] | [ Objects ] ] } ]
} | [ Relative Path { On | Off } ] [ Enable { On | Off } ],
[ hotlink_id[ Using launch_expr ] | [ On { [ [ Labels ] | [ Objects ] ] } ]
} | [ Relative Path { On | Off } ] [ Enable { On | Off } ]...
```

### Examples

```
Activate Modify 1 Using URL1 On Objects, 2 Relative Path Off
```

```
Activate Modify 2 On Objects, 4 On Labels
```

```
Activate Modify 3 Relative Path On Enable Off
```

```
Activate Modify 2 Enable Off, 3 Enable On
```

### Comments

hotlink\_id is an integer index (1-based) that specifies the hotlink definition to modify

At least one of the *Using/On/Relative/Enabled* clauses is required.

launch\_expr must not be an empty string (for example, "").

When the *Enable* clause is included and set to *Off*, the hotlink definition will be disabled.

## Removing HotLink Definitions

### Syntax

```
Activate Remove { All | hotlink_id [ , hotlink_id, hotlink_id, ... ] }
```

### Examples

```
Activate Remove 2, 4  
Activate Remove All
```

### Comments

*hotlink\_id* is an integer index (1-based) that specifies the hotlink definition to modify.

At least one *hotlink\_id* must be specified.

If the *All* keyword is used, then all hotlink definitions are removed.

## Reordering HotLink Definitions

### Syntax

```
Activate Order { hotlink_id [ , hotlink_id, hotlink_id, ... ] }
```

### Examples

```
Activate Order 2, 3, 1
```

### Comments

*hotlink\_id* is an integer index (1-based) that specifies the hotlink definition to modify

At least one *hotlink\_id* must be specified.

## Exceptions to Support Backwards Compatibility

The Using clause can be omitted, but only from the first HotLink definition. The Using expression can be empty (""), but only for the first HotLink definition.

---

## SystemInfo() function

### Purpose:

This function has a new attribute to return the current build number so you can distinguish between MapInfo Professional point versions in MapBasic. For example, SystemInfo(SYS\_INFO\_MIVERSION) returns 850 for MapInfo Professional 8.5 and 8.5.2. You can further distinguish between 8.5 and 8.5.2 with SystemInfo(SYS\_INFO\_MIBUILD\_NUMBER), which returns 32 for 8.5, and 60 for 8.5.2.

## TableInfo( ) function

---

### Syntax:

```
SystemInfo(SYS_INFO_MIBUILD_NUMBER)
```

**Note:** The MapBasic.def constant for this function is 18.

---

## TableInfo( ) function

### Purpose

Returns information about an open table. Added a new define for FME tables.

### Syntax

```
TableInfo( table_id, attribute )
```

*table\_id* is a string representing a table name, a positive integer table number, or 0 (zero).

*attribute* is an integer code indicating which aspect of the table to return.

### Return Value

String, SmallInt, or logical, depending on the attribute parameter specified.

The *attribute* parameter can be any value from the table below. Codes in the left column (for example, TAB\_INFO\_NAME) are defined in MAPBASIC.DEF.

attribute code	TableInfo( ) returns
TAB_INFO_TYPE	SmallInt result, indicating the type of table. The returned value will match one of these: <ul style="list-style-type: none"><li>• TAB_TYPE_BASE (if a normal or seamless table).</li><li>• TAB_TYPE_RESULT (if results of a query).</li><li>• TAB_TYPE_IMAGE (if table is a raster image).</li><li>• TAB_TYPE_VIEW (if table is actually a view; for example, StreetInfo tables are actually views).</li><li>• TAB_TYPE_LINKED (if this table is linked).</li><li>• TAB_TYPE_WMS (if table is from a Web Map Service).</li><li>• TAB_TYPE_WFS (if table is from a Web Feature Service).</li><li>• TAB_TYPE_FME (if table is opened through FME).</li></ul>