mEagle Programming Guide

Feynman

November 28, 2006

Contents

1 Introduction

This chapter will introduce mEagle and organization of this guide.

1.1 About mEagle

MEagle is a geography information system for embedded system, which is developed by Feynman Software. mEagle is based on MiniGUI embedded graphics middleware, so it is able to run on more than 10 embedded OSes which MiniGUI supported.

In view of the common application of geographic information system in embedded devices, for example, electronic map operations (roam, zoom, rotation), GPS positioning, path tracking, and so on, Feynman Software has carried on a careful optimized design to mEagle, and makes it have a fast speed, low resource consumption.

The main features of mEagle:

- Based on MiniGUI graphics library, it is suitable for all kinds of embedded systems, and it has good portability, as well as fast running speed.
- Mainly support for MapInfo, ArcInfo and other popular map formats to meet the requirements of different users.
- Support for common GIS functions, including map navigation, zoom, distance measurement, map rotation, and so on.
- Support for eagle eye function.
- Support for keyword search and regional search for map information, and it can locate the searching result as well.
- Support for the map coordinate data processing function, and it can be integrated with GPS module to provide GPS navigation and location functions.

1.2 Organization of this Guide

This text is divided into four chapters:

- Chapter one: Introduce mEagle and describes the organization of this guide.
- Chapter two: Give an example for programming with mEagle to help developer start quickly.
- Chapter three: Describe the use of two controls provided by mEagle: meagle and meagleeye.
- Chapter four: Describe all messages of the two controls.

2 Getting Started

This chapter will give a simple program to describe how to develop applictions with mEagle.

2.1 A Simpel Program

The quickest approach to understand the programming method of mEagle is to analyze a simple program. Listing 1 shows a program loading and displaying map,which will be discussed in detail.

Listing 1: A Simple Program

```
/*MiniGUI header file .*/
#include <minigui/common.h>
#include <minigui/minigui.h>
#include <minigui/gdi.h>
#include <minigui/window.h>
#include <minigui/control.h>

/*mEagle header file .*/
#include "meagle.h"

#define IDC_MEVIEW     101
#define IDC_MEEYEVIEW  102

/*Handle of meagle cotrol .*/
HWND meagle_hwnd;

/*Handle of meagle eye control .*/
HWND meagleeye_hwnd;

static int mEagleProc(HWND hWnd, int message, WPARAM wParam, LPARAM lParam)
{
    switch (message)
    {
        case MSG_CREATE:

            char **layers_name = NULL;

            /*Create a mEagle control class instance . */
            meagle_hwnd = CreateWindow (MEAGLE_CTRL, "", WS_VISIBLE,
                                IDC_MEVIEW, 0, 0, 600, 480, hWnd, 0);

            /*Create a mEagle Eye control class instance .*/
            meagleeye_hwnd = CreateWindow (MEAGLEEYE_CTRL, "", WS_VISIBLE,
                                IDC_MEEYEVIEW, 600, 480, 150, 150, hWnd, 0);

            /*Open a data source.*/
            SendMessage(meagle_hwnd, ME_OPEN_MAP, 0, (LPARAM)"./res/map");

            /*Get layer names*/
            SendMessage(meagle_hwnd, ME_GET_MAPINFO, 0, (LPARAM)&layers_name);

            /*Load map layers which are specified by layers_name.*/
            SendMessage(meagle_hwnd, ME_LOAD_MAP, 0, (LPARAM)layers_name);

            /*Add eagle eye window*/
            SendMessage(meagleeye_hwnd, ME_ADD_EYE, 0, meagle_hwnd);
            /* Set current operating state to be panning*/
            SendMessage (meagle_hwnd, ME_CHANGE_TOOL, ME_STATUS_DRAG, 0);

            if (layers_name)
                free (layers_name);
            break;

        case MSG_DESTROY:

            /*Close map.*/
            SendMessage(meagle_hwnd, ME_CLOSE_MAP, 0, 0);

            DestroyAllControls (hWnd);
            return 0;

        case MSG_CLOSE:

            DestroyMainWindow (hWnd);
            PostQuitMessage (hWnd);
            return 0;
    }
    return DefaultMainWinProc(hWnd, message, wParam, lParam);
}

int MiniGUIMain (int args, const char* argv [])
{
    MSG Msg;
    HWND hMainWnd;

    MAINWINCREATE CreateInfo;

    /* register meagle control class */
    RegisterMEagleControl () ;

    /* register meagle eye control class */
    RegisterMEagleEyeControl ();

    CreateInfo.dwStyle = WS_VISIBLE | WS_BORDER;
    CreateInfo.dwExStyle = WS_EX_NONE;
    CreateInfo.spCaption = "mEagle";
    CreateInfo.hMenu = 0;
    CreateInfo.hCursor = GetSystemCursor(0);
    CreateInfo.hIcon = 0;
    CreateInfo.MainWindowProc = mEagleProc;
    CreateInfo.lx = 0;
    CreateInfo.ty = 0;
    CreateInfo.rx = 750;
    CreateInfo.by = 630;
    CreateInfo.iBkColor = COLOR_lightwhite;
    CreateInfo.dwAddData = 0;
    CreateInfo.hHosting = HWND_DESKTOP;

    hMainWnd = CreateMainWindow (&CreateInfo);

    if (hMainWnd == HWND_INVALID)
        return -1;

    ShowWindow(hMainWnd, SW_SHOWNORMAL);
    SetWindowBkColor (hMainWnd, RGB2Pixel (HDC_SCREEN, 230, 246, 255));
    InvalidateRect (hMainWnd, NULL, TRUE);

    while (GetMessage(&Msg, hMainWnd)) {
        TranslateMessage(&Msg);
        DispatchMessage(&Msg);
    }

    /* Unregister meagle control class */
    UnregisterMEagleControl () ;

    /* Unregister meagle eye control class . */
    UnregisterMEagleEyeControl () ;

    MainWindowThreadCleanup (hMainWnd);

    return 0;

}
```

2.2 Header file

The header file included in beginning of the simple program, namely meagle.h, should be included for all applictions with mEagle control. The header file includes definitions of data types, structure, messages and interfaces of meagle and meagleeye control.

2.3 Register Control class

```
RegisterMEagleControl () ;
RegisterMEagleEyeControl () ;
```

Each control of MiniGUI is an instance of a certain control class. Before creating an control instance, you must first register control class. RegisterMEagleControl and RegisterMEagleEyeControl are functions to register meagle control class and meagleeye control class.

2.4 Create Control Instance

```
meagle_hwnd = CreateWindow (MEAGLE_CTRL,‘‘’’,WS_VISIBLE,IDC_MEVIEW,30, 0, mectlw, mectlh
            , hWnd, 0);
meagleeye_hwnd = CreateWindow (MEAGLEEYE_CTRL,‘‘’’,WS_VISIBLE,IDC_MEEYEVIEW
            ,600, 370, 150, 150, hWnd, 0);
```

In MiniGUI,an instance of a certain control class can be created by calling CreateWindow function. It specifies the control class name, the control catption, the control style, the control identifier, and the initial position and size of the control.This function also specifies the parent window of the child window.

2.5 Open Data Source

```
SendMessage(meagle_hwnd, ME_OPEN_MAP, 0, (LPARAM)‘‘./res/map’’);
```

After meagle control instance and meagleeye control instance are created, you can send messages to them ,which is message driven, to make a lot of mapping operation such as loading, zooming, panning, rotating,etc. A lot of messages are defined for the two controls. They will be detailed in chapter 4. When meagle control receives message ME_OPEN_MAP, it will open a data source which is specified by lParam parameter.

2.6 Get Layer Name

```
SendMessage(meagle_hwnd, ME_GET_MAPINFO, 0, (LPARAM)&layers_name);
```

After meagle control open a data source, you should get all layer names of the datasource. You can send message ME_GET_MAPINFO to meagle control and the returned names are stored in parameter lParam.

2.7 Load Map

```
SendMessage(meagle_hwnd, ME_LOAD_MAP, 0, (LPARAM)layers_name);
```

With all layer names, the map can be loaded and displayed when meagle control receives message ME_LOAD_MAP.

2.8 Change Operating Status

```
SendMessage(meagle_hwnd, ME_CHANGE_TOOL, ME_STATUS_DRAG, 0);
```

Mapping operation meagle control supports is specified by operating status such as pan, zoom, rotate, etc. You can set current status by sending message ME_CHANGE_TOOL to meagle control. The parameter wParam specifies the new status.

2.9 Add Eye

```
SendMessage(meagleeye_hwnd, ME_ADD_EYE, 0, meagle_hwnd);
```

In order to make meagleeye take effect, message ME_ADD_EYE should be sent to meagleeye control and parameter lParam specifies handle of meagle control.

2.10 Close Map

```
SendMessage(meagle_hwnd, ME_CLOSE_MAP, 0, 0);
```

When meagle control receives message ME_CLOSE_MAP, it will unload the map.

2.11 Unregister Control class

```
UnregisterMEagleControl () ;
UnregisterMEagleEyeControl () ;
```

When exiting application, you must unregister control class. UnregisterMEagleControl function and UnregisterMEagleEyeControl function unregister meagle control class and meagleeye control class.

3 mEagle controls

There are two predefined controls in mEagle,including meagle and meagleeye. They are the main elements for mapping operation. Table 1 1ists the predefined control classes and the corresponding classs names in mEagle. This chapter will describe in detail the predefined control classes in mEagle.

Table 1: mEagle controls

Control Class	Class Name	C Macro for Control Name
meagle	meagle	MEAGLE_CTRL
eye	meagleeye	MEAGLEEYE_CTRL

3.1 meagle

Meagle control is a MiniGUI control which provides a lot of mapping functionality including loading, panning, zooming, rotating, distance measurement, searching,etc. With meagle control, developers can develop self GIS or GPS navagation application easily and quickly.

To develop application with mEagle control, you should first register a control class with function RegisterMEagleControl.It returns TRUE if success, otherwise return FALSE.

```
BOOL RegisterMEagleControl (void)
```

Secondly you should create an instance of meagle control class with function CreateWindow.

```
meagle_hwnd = CreateWindow (MEAGLE_CTRL, ‘‘’’,WS_VISIBLE, IDC_MEVIEW, 0, 0, 200, 200,
            hWnd, 0);
```

The function specifies the control class, the control caption, the control style, the control identifier, the initial position, size of the control and the parent window of the child window.

When exiting the application, you should unregister control class with function UnregisterMEagleControl.

```
void UnregisterMEagleControl (void)
```

Meagle control provides a lot of mapping functionality which is message driven. By sending messages to meagle control, you can drive the control to operate the map. Table 2 lists all messages of meagle control. More details are described in Chapter 4.

3.2 meagleeye

Meagleeye is a MiniGUI control which shows a full extent view of a map. It must be used with map control and can not be used singly. To develop application with meagleeye control, you shoud first register a control class with function RegisterMEagleEyeControl. It returns TRUE if success, otherwise return FALSE.

Table 2: Messages of meagle

Message	Notes
ME_OPEN_MAP	Open a data source
ME_GET_MAPINFO	Get all layer name
ME_LOAD_MAP	Load and display map data.
ME_CLOSE_MAP	Unload map data.
ME_LOAD_MEAGLE_MAP	Load and display MEG map file.
ME_SAVE_MEAGLE_MAP	Save map data in RAM to MEG file
ME_CHANGE_TOOL	Change current operation
ME_GET_LAYERCOUNT	Get count of layers.
ME_GET_LAYERNAME	Get certain layer's name
ME_GET_LAYERVISIBLE	Decide whether certain layer is visible.
ME_CHANGE_LAYERINDEX	Change certain layer's index.
ME_CHANGE_LAYERVISIBLE	Make certain layer visible or not.
ME_WIN_TO_LL	Convert screen coordinate to geographic coordinate.
ME_LL_TO_WIN	Convert geographic coordinate to screen coordinate.
ME_REFRESH	Repaint map.
ME_GLOBAL	Zoom map to full extent.
ME_CLEAR	Clear all custom features.
ME_NORTH	North seeking.
ME_SET_SCALE	Set current scale.
ME_GET_SCALE	Get current scale.
ME_MOVE_X	Move map in horizontal direction.
ME_MOVE_Y	Move map in vertical directon.
ME_ROTATE	Rotate map some angle.
ME_KEY_SEARCH	Search features on name.
ME_AREA_SEARCH	Search features on area.
ME_GET_KEYRESULTCOUNT	Get count of keyword search result.
ME_GET_KEYRESULTITEM	Get certain item of keyword search result.
ME_GET_AREARESULTCOUNT	Get count of area search result.
ME_GET_AREARESULTITEM	Get certain item of area search result.
ME_ADD_RESULT	Add a custom feature.
ME_REMOVE_RESULT	Remove certain custom feature.
ME_SETCENTER_LAT	Set center point.

```
BOOL RegisterMEagleEyeControl (void)
```

Secondly you should build a meagleeye control class instance with CreateWindow.

```
meagleeye_hwnd = CreateWindow (MEAGLEEYE_CTRL,‘‘’’,WS_VISIBLE,IDC_MEEYEVIEW
            ,600, 370, 150, 150, hWnd, 0);
```

When exiting the application, you should unregister the control class with function UnregisterMEagleEyeControl.

```
void  UnregisterMEagleEyeControl (void)
```

Meagleeye control is also message driven as meagle control. Table 3 lists all messages of meagleeye control.

Table 3: Messages of meagleeye

Message	Notes
ME_ADD_EYE	connect eagle eye with map view.
ME_REMOVE_EYE	disconnect eagle eye.
ME_UPDATE_EYE	Repaint eagle eye.

More details are described in Chapter 4.

4 Messages

This chapter will describe messages of meagle control and meagleeye control in detail.

4.1 Open Map

Sending ME_OPEN_MAP message to meagle control will open a map file/source data. The parameter lParam is the pointer to the buffer which stores the path of the map or the data source. It should be noted that lParam is a directory name which includes all map layers.

```
char *dir = ‘‘ mapdir’’;
SendMessage(meagle_hwnd, ME_OPEN_MAP, 0, (LPARAM)dir);
```

4.2 Load Map

Sending ME_LOAD_MAP message to meagle control will load and display map. The parameter lParam is the pointer to the buffer which stores all layer names.

```
char **layers = {‘‘ river ’’ ,‘‘ city ’’ ,‘‘ road’’};
SendMessage(meagle_hwnd, ME_LOAD_MAP, 0, (LPARAM)layers);
```

4.3 Close Map

Sending ME_CLOSE_MAP message to meagle control will unload and close the map.

```
SendMessage(meagle_hwnd, ME_CLOSE_MAP, 0, 0);
```

4.4 Save Meagle Map

Sending ME_SAVE_MEAGLE_MAP message to meagle control will save current map data in RAM to a MEG file with .meg extent whose format is defined by Feynman,Inc. The parameter lParam is the pointer to the buffer which specifies the MEG file’s name.

```
char *file_name = {‘‘ beijing .meg’’};
SendMessage(meagle_hwnd, ME_SAVE_MEAGLE_MAP, 0, (LPARAM)file_name);
```

4.5 Load Meagle MAP

Sending ME_LOAD_MEAGLE_MAP message to meagle control will load a MEG file and display the map. The parameter lParam is the pointer to the buffer which stores a MEG file’s name. It should be noted that meagle can load MEG file much quicker than MapInfo or ArcInfo file. It is specially used in embedded device.

```
char *file_name = {‘‘ beijing .meg’’};
SendMessage(meagle_hwnd, ME_LOAD_MEAGLE_MAP, 0, (LPARAM)file_name);
```

4.6 Get Map Info

Sending ME_GET_MAPINFO message to meagle control will get all layer name. The parameter lParam is the pointer to the buffer which receives all layer names.

```
char **layers_name = NULL;
SendMessage(meagle_hwnd, ME_GET_MAPINFO, 0, (LPARAM)&layers_name);
```

4.7 Set Layer Font

Sending ME_SET_LAYERFONT message to meagle control will set font of a layer . The parameter wParam is the pointer to the buffer which stores certain layer's name, and lParam is the pointer to a LOGFONT structure which is a font structure of MiniGUI. More details about LOGFONT are described in MiniGUI's Program Guide.

```
char *layers_name = ‘‘ city ’’;
LOGFONT *font = CreateLogFont(NULL, ”SansSerif”, FONT_CHARSET_GB2312_0,
        FONT_WEIGHT_BOLD, FONT_SLANT_ROMAN, FONT_SETWIDTH_NORMAL,
        FONT_SPACING_CHARCELL, FONT_UNDERLINE_NONE, FONT_STRUCKOUT_NONE
        , 8, 0);
SendMessage(meagle_hwnd, ME_SET_LAYERFONT, (DWORD)layers_name, (DWORD)font);
```

4.8 Change Tool

Sending ME_CHANGE_TOOL message to meagle control will change current mapping tool. The parameter wParam specifies new tool.

```
SendMessage (meagle_hwnd, ME_CHANGE_TOOL, ME_STATUS_ZOOMIN, 0);
```

Most mapping applications provide an assortment of toolbar buttons to aid with common navigation tasks(such as panning). Meagle control provides five built-in mapping tools. Table 4 lists all tools and their C Macro.

Table 4: Map Operating Status

C Macro	Tool
ME_STATUS_DRAG	Pan
ME_STATUS_ZOOMOUT	Zoom Out
ME_STATUS_ZOOMIN	Zoom In
ME_STATUS_ROTATE	Rotate
ME_STATUS_DISTANCE	Ruler

- Pan: Repositions the map within the window.
- Zoom Out: Gets a wider area view of the map.
- Zoom In: Gets a closer area view of the map.
- Rotate: Rotates the map within the window.
- Ruler: Measures distance between two points.

4.9 Get Layer Count

Sending ME_GET_LAYERCOUNT message to meagle control will get count of map layers. The parameter lParam receives the number.

```
int  layer_count  = 0;
SendMessage(meagle_hwnd, ME_GET_LAYERCOUNT, 0, (LPARAM)&layer_count);
```

4.10 Get Layer Name

Sending ME_GET_LAYERNAME message to meagle control will get certain layer's name. The parameter wParam is the index of a layer, and lParam is the pointer to buffer which receives the name.

```
int index  = 0;
char *layer_name;
SendMessage(meagle_hwnd, ME_GET_LAYERNAME, index, (LPARAM)&layer_name);
```

4.11 Get Layer Visible

Sending ME_GET_LAYERVISIBLE message to meagle control will know whether certain layer is displayed in the map. The parameter wParam is the index of a layer, and lParam receives the return value. If the value is 1 ,the layer is visible; otherwise it is unvisible.

```
int index  = 0;
int  layer_visible  ;
SendMessage(meagle_hwnd, ME_GET_LAYERVISIBLE, index, (LPARAM)&layer_visible);
```

4.12 Change Layer Index

Sending ME_CHANGE_LAYERINDEX message to meagle control will change certain layer's index which decides sequence of displaying. The parameter wParam is new index of the layer, and lParam is the pointer to buffer which stores the layer's name.

```
int index  = 2;
char* layer_name  = `` river '' ;
SendMessage(meagle_hwnd, ME_CHANGE_LAYERINDEX, index, (LPARAM)layer_name);
```

4.13 Change Layer Visible

Sending ME_CHANGE_LAYERVISIBLE message to meagle control will control whether certain layer is displayed in the map. The parameter wParam is index of the layer, and lParam controls whether the layer is visible. If 1 ,visible; otherwise, unvisible.

```
int index  = 0;
int  layer_visible   = 1;
SendMessage(meagle_hwnd, ME_CHANGE_LAYERVISIBLE, index, layer_visible);
```

4.14 Screen Coordinate To Geographic Coordinate

Sending ME_WIN_TO_LL message to meagle control will convert coordinates of a point from screen coordinate system to geographic coordinate system. The parameter wParam is a pointer to structure which stores the point's screen coordinate, and lParam is a pointer to the structure which stores converted coordinate.

```
POINT point  = { 100, 100 };
DOUBLE_POINT dp;
SendMessage(meagle_hwnd, ME_WIN_TO_LL, (WPARAM)&point, (LPARAM)&dp);
```

4.15 Geographic Coordinate To Screen Coordinate

Sending ME_LL_TO_WIN message to meagle control will convert coordinates of a point from geographic coordinate system to screen coordinate system. The parameter wParam is a pointer to structure which stores the point's geographic coordinate, and lParam is a pointer to structure which stores converted coordinate.

```
DOUBLE_POINT dp = {116.43448,39.92272};
POINT point;
SendMessage(meagle_hwnd, ME_LL_TO_WIN, (WPARAM)&dp, (LPARAM)&point);
```

4.16 Refresh

Sending ME_REFRESH message to meagle control will repaint the map.

```
SendMessage(meagle_hwnd, ME_REFRESH, 0, 0);
```

4.17 Global

Sending ME_GLOBAL message to meagle control will zoom map to full extent.

```
SendMessage(meagle_hwnd, ME_GLOBAL, 0, 0);
```

4.18 Clear

Sending ME_CLEAR message to meagle control will clear all custom features such as search result, measuring line, etc.

```
SendMessage(meagle_hwnd, ME_CLEAR, 0, 0);
```

4.19 North Seeking

Sending ME_NORTH message to meagle control will seek north and rotate the map to it.

```
SendMessage(meagle_hwnd, ME_NORTH, 0, 0);
```

4.20 Set Scale

Sending ME_SET_SCALE message to meagle control will set current scale to be a new one. The parameter lParam is the new scale to be set. If the new scale is larger than the old one, the map will zoom in; otherwise zoom out.

```
double pixel_length ;
SendMessage(hwnd, ME_SET_SCALE, 0, (DWORD)(&pixel_length));
```

4.21 Get Scale

Sending ME_GET_SCALE message to meagle control will get current scale.

```
double pixel_length ;
SendMessage(hwnd, ME_GET_SCALE, 0, (DWORD)(&pixel_length));
```

4.22 Move X

Sending ME_MOVE_X message to meagle control will move map in horizontal direction. The parameter wParam specifies the coordinate system such as ME_COORDINATE_WIN ,ME_COORDINATE_LAT, and lParam is the offset to move. If parameter lparam is larger than zero, the map will be moved leftwards, otherwise it will be moved rightwards.

```
SendMessage (meagle_hwnd, ME_MOVE_X, ME_COORDINATE_WIN, (LPARAM)−300);
```

4.23 Move Y

Sending ME_MOVE_Y message to meagle control will move map in vertical direction. If parameter lparam is larger than zero, the map will be moved downwards, otherwise it will be moved upwards.

```
SendMessage (meagle_hwnd, ME_MOVE_Y, ME_COORDINATE_WIN, (LPARAM)−300);
```

4.24 Rotate

Sending ME_ROTATE message to meagle control will rotate map some angle. The parameter lParam is the radian to rotate.

```
#define G_PI   3.14159265358979323846E0
alpha = −G_PI/8;
SendMessage (meagle_hwnd, ME_ROTATE, 0, (LPARAM)&alpha);
```

4.25 Key Search

Sending ME_KEY_SEARCH message to meagle control will search a feature according to the feature name. The parameter lParameter is a pointer to the buffer which stores the name.

```
char keyinfo [256] = ‘‘ NewYork’’;
SendMessage (meagle_hwnd, ME_KEY_SEARCH, 0, (LPARAM)keyinfo);
```

4.26 Area Search

Sending ME_AREA_SEARCH message to meagle control will search all features in a circle area in the map. The parameter lParam is a pointer to AREA_SEARCH structure which includes the center and radius of the circle area. More details about AREA_SEARCH will be described in Section A.

```
AREA_SEARCH area_search;
 area_search . latitude  = dp.x;
 area_search . longitude  = dp.y;
 area_search . distance  = 200;
SendMessage(meagle_hwnd, ME_AREA_SEARCH, 0, (LPARAM)&area_search);
```

4.27 Get Key Result Count

Sending ME_GET_KEYRESULTCOUNT message to meagle control will get count of features which are found according to feature name.

```
int   result_count ;
 result_count  = SendMessage(meagle_hwnd, ME_GET_KEYRESULTCOUNT, 0, 0);
```

4.28 Get Key Result Item

Sending ME_GET_KEYRESULTITEM message to meagle control will get a feature which is found according to feature name. The parameter wPara is result index, and lParam is a pointer to a RESULT structure which includes a feature's information such as id, name, geometry type, color, coordinate, label, etc. More details about RESULT are described in Appendex A.

```
RESULT result;
int  index  = 1;
SendMessage(meagle_hwnd, ME_GET_KEYRESULTITEM, index, (LPARAM)&result);
```

4.29 Get Area Result Count

Sending ME_GET_AREARESULTCOUNT message to meagle control will get count of features which are found in an circle area.

```
 result_count  = SendMessage(meagle_hwnd, ME_GET_AREARESULTCOUNT, 0, 0);
```

4.30 Get Area Result Item

Sending ME_GET_AREARESULTITEM message to meagle control will get a feature which is found in a circle area. The parameter wParam is result index, and lParam is a pointer to RESULT structure.

```
RESULT result;
int  index  = 1;
SendMessage(meagle_hwnd, ME_GET_AREARESULTITEM, index, (LPARAM)&result);
```

4.31 Add Result

Sending ME_ADD_RESULT message to meagle control will add a custom feature in meagle control. The parameter wParam is the custom result's index, and lParam is a pointer to RESULT structure.

```
int   result_id   = 1;
RESULT result;
 result .GEOMDATA.CIRCLE.ll.latitude = dp.x;
 result .GEOMDATA.CIRCLE.ll.longitude = dp.y;
 result .GEOMDATA.CIRCLE.color = color;
 result .GEOMDATA.CIRCLE.size = size;
 result . feature_name  =  name;
 result .geom_type = CIRCLEFILLED;
SendMessage(meagle_hwnd, ME_ADD_RESULT, (WPARAM)&result_id, (LPARAM) &result);
```

4.32 Remove Result

Sending ME_REMOVE_RESULT message to meagle control will romove a custom feature in meagle control. The parameter lParam is the index of the result to be removed.

```
int   result_id   = 1;
 SendMessage(meagle_hwnd, ME_REMOVE_RESULT, 0, (LPARAM)result_id);
```

4.33 Set Center Lat

Sending ME_SETCENTER_LAT message to meagle control will set the point which will be displayed in the center of meagle control window. The parameter wParam is the longitude of the point, and lParam is the latitude of the point.

```
DOUBLE_POINT dp = {116.43448,39.92272};
sendMessage(meagle_hwnd, ME_SETCENTER_LAT,(DWORD)&dp.x, (LPARAM)&dp.y);
```

4.34 Add Eye

After creating meagleeye control, you should send ME_ADD_EYE message to meagleeye control to connect it with a meagle control. The parameter lParam is the handle of meagle control.

```
SendMessage(meagleeye_hwnd, ME_ADD_EYE, 0, (LPARAM)meagle_hwnd);
```

4.35 Update Eye

Sending ME_UPDATE_EYE message to meagleeye control will repaint it.

```
SendMessage(meagle_hwnd, ME_UPDATE_EYE, 0, 0);
```

4.36 Remove Eye

Sending ME_REMOVE_EYE message to meagleeye control will disconnect it with meagle cotrol.

```
SendMessage(meagleeye_hwnd, ME_REMOVE_EYE, 0, 0);
```

A Data Structure

A.1 DOUBLE_POINT

```
typedef struct _DoublePoint
{
    double x;
    double y;
} DOUBLE_POINT;
```

A.2 KEYWORD_SEARCH

```
typedef struct _KEYWORD_SEARCH
{
    double latitude ;
    double longitude ;
    char* feature_name ;
}KEYWORD_SEARCH;
```

A.3 AREA_SEARCH

```
typedef struct _AREA_SEARCH
{
    double latitude ;
    double longitude ;
    int distance ;
}AREA_SEARCH;
```

A.4 RESULT

```
typedef struct _RESULT
{
    long id ;
    const char* feature_name ;
    int geom_type;
    union _GEOMDATA
   {
        struct _CIRCLE
       {
            int size ;
            char* color ;
            int isll ;
            union {
                struct {
                    double latitude ;
                    double longitude ;
            }       ll ;
                struct {
                    int x;
                    int y;
            }       win;
          };
       } CIRCLE;
```

```
        struct _PICTURE
        {
            int width;
            int height;

            PBITMAP pBitmap;
            int isll;
            union{
                struct{
                    double latitude;
                    double longitude;
                } ll;
                struct {
                    int x;
                    int y;
                } win;
            };
        } PICTURE;

        struct _LINES
        {
            int width;
            char* color;
            int num;
            DOUBLE_POINT* points;
        } LINES;

        struct _BIGCIRCLE
        {
            double r;
            int pen_width;
            char* color;
            double latitude;
            double longitude;
        } BIGCIRCLE;
    } GEOMDATA;
}RESULT;
```

B Messages Definition

```
enum
{
    ME_END_KEYSEARCH,
    ME_LOAD_MEAGLE_MAP,
    ME_SAVE_MEAGLE_MAP,
    ME_OPEN_MAP,
    ME_GET_MAPINFO,
    ME_LOAD_MAP,
    ME_CLOSE_MAP,
    ME_CHANGE_TOOL,
    ME_GET_LAYERCOUNT,
    ME_GET_LAYERNAME,
    ME_GET_LAYERVISIBLE,
    ME_SET_LAYERFONT,

    ME_CHANGE_LAYERINDEX,
    ME_CHANGE_LAYERVISIBLE,
    ME_ADD_EYE,
    ME_REMOVE_EYE,
    ME_UPDATE_EYE,

    ME_WIN_TO_LL,
    ME_LL_TO_WIN,

    ME_REFRESH,
    ME_GLOBAL,
    ME_CLEAR,
    ME_NORTH,
    ME_SET_SCALE,
    ME_GET_SCALE,
    ME_MOVE_X,
    ME_MOVE_Y,
    ME_ROTATE,
    ME_KEY_SEARCH,
    ME_AREA_SEARCH,
    ME_GET_KEYRESULTCOUNT,
    ME_GET_KEYRESULTITEM,
    ME_GET_AREARESULTCOUNT,
    ME_GET_AREARESULTITEM,
    ME_ADD_RESULT,
    ME_REMOVE_RESULT,
    ME_SETCENTER_LAT,
};
```