Explain for SDK

1. Summarize

MarkEzd.dll file is Dynamic Link Library provided by Beijing JCZ Technology CO. LTD for developing software based on ezcad2 and lmc board

MarkEzdDll.h is header file of the exports function in MarkEzd.dll

We can develop the software using VC6.0.

The calling way of MarkEzd.dll is explicitly link. Developer needs to load and free MarkEzd.dll by calling Windows API function.

The steps are as follows.

1. Call Windows' API function LoadLibrary() to load DLL dynamically;

2. Call Windows' API function GetProcAddrress() to get the pointer of the functions in the DLL and use the function pointer to finish the work;

3. Call Windows' API function FreeLibrary() to release library when you do not use DLL or the program ends.

Note: the program that calls MarkEzd.dll must be located at the same folder with ezcad2.exe. Or else the program would not work. And ezcad2.exe and markEzd.dll cannot work as usually at the same time, so you must close ezcad2.exe when MarkEzd.dll is used.

2. Description

Return value of most functions in MarkEzd.dll is a common error code of integer type. Its definitions are as follow:

```
#define LMC1_ERR_SUCCESS        0    // Success
#define LMC1_ERR_EZCADRUN       1    // Find EZCAD running
#define LMC1_ERR_NOFINDCFGFILE  2    // Can not find EZCAD.CFG
#define LMC1_ERR_FAILEDOPEN     3    // Open LMC1 board failed
#define LMC1_ERR_NODEVICE       4    // Can not find valid lmc1 device
#define LMC1_ERR_HARDVER        5    // Lmc1's version is error.
#define LMC1_ERR_DEVCFG         6    // Can not find configuration files
#define LMC1_ERR_STOPSIGNAL     7    // Alarm signal
#define LMC1_ERR_USERSTOP       8    // User stops
#define LMC1_ERR_UNKNOW         9    // Unknown error
#define LMC1_ERR_OUTTIME        10   // Overtime
#define LMC1_ERR_NOINITIAL      11   // Un-initialized
#define LMC1_ERR_READFILE       12   // Read file error
#define LMC1_ERR_OWENWNDNULL    13   // Window handle is NULL
#define LMC1_ERR_NOFINDFONT     14   // Can not find designated font
#define LMC1_ERR_PENNO          15   // Wrong pen number
#define LMC1_ERR_NOTTEXT        16   // Object is not text
#define LMC1_ERR_SAVEFILE       17   // Save file failed
#define LMC1_ERR_NOFINDENT      18   // Can not find designated object
#define LMC1_ERR_STATUE         19   // Can not run the operation in
current state
```

Note: The entire TCHAR object in MarkEzd.dll must be UNICODE.

3. Device

lmc1_Initial

INTENTION: initialize lmc1 control board

DEFINITION: int lmc1_Initial(TCHAR* strEzCadPath, BOOL bTestMode, HWND hOwenWnd)

strEzCadPath: //the full path where ezcad2.exe exists

bTestMode //Whether in test mode or not

hOwenWnd: //The window that has the focus. It is used to check the user's stop messages.

DESCRIPTION: you must first call lmc1_Initial before other function in program.
RETURN VALUE: common error code

lmc1_Initial2

INTENTION: initialize lmc1 control board
DEFINITION: int lmc1_Initial(TCHAR* strEzCadPath, BOOL bTestMode, HWND hOwenWnd)
strEzCadPath: //the full path where ezcad2.exe exists
bTestMode //Whether in test mode or not
DESCRIPTION: you must first call lmc1_Initial before other function in program.
RETURN VALUE: common error code

lmc1_Close

INTENTION: Close lmc1 board
DEFINITION: int lmc1_Close();
DESCRIPTION: you must call lmc1_Close to close the lmc1 board when exit program.
RETURN VALUE: common error code

lmc1_SetDevCfg

INTENTION: set parameter for device
DEFINITION: int lmc1_SetDevCfg( )
DISCRIPTION: call lmc1_SetDevCfg, and then a window will be popup. User could set parameters of device in it.
RETURN VALUE: common error code

lmc1_SetDevCfg2

INTENTION: set parameter for device
DEFINITION: int lmc1_SetDevCfg (BOOL bAxisShow0，BOOL bAxisShow1)
bAxisShow0 //show axis0 windows or not
bAxisShow1 //show axis1 windows or not
DISCRIPTION: call lmc1_SetDevCfg, and then a window will be popup. User could set parameters of device in it. And decided show this window or not.
RETURN VALUE: common error code

lmc1_SetRotateMoveParam

INTENTION: set parameter for rotation transform
DEFINITION：void lmc1_SetRotateMoveParam (double dMoveX, double dMoveY, dCenterX, double dCenterY,

double dRotateAng)；

dMoveX //the move distance of X direction
dMoveY //the move distance of Y direction
dCenterX: //X coordinate of rotate center
dCenterY: //Y coordinate of rotate center
dRotateAng: //rotation angle (in radian)

DISCRIPTION: call lmc1_SetRotateParam to set the parameter of rotation transform. All of the objects in database are rotated around the appointed center. And then move some distance.

RETURN VALUE: NULL

4. Mark

lmc1_Mark

INTENTION: mark all the data in database

DEFINITION: int lmc1_Mark(BOOL bFlyMark)；
bFlyMark= TRUE // enable mark on fly

DISCRIPTION: Begin to mark by calling this function after loading ezd file using lmc1_LoadEzdFile. The function will not return back until marking complete.

RETURN VALUE: common error code

lmc1_MarkEntity

INTENTION: mark the appointed named object in database

DEFINITION: int lmc1_MarkEntity(TCHAR* strEntName);

DISCRIPTION: after loading ezd file by lmc1_LoadEzdFile, user can use this function to mark the appointed object. The function will not return back until marking complete.

RETURN VALUE: common error code

lmc1_MarkFlyByStartSignal

INTENTION: mark all the data in database on the fly mode

DEFINITION: int lmc1_MarkFlyByStartSignal()；

DISCRIPTION: After use this function, the software will wait hardware signal (IN8/IN9, setting on the fly param window), start to mark after software get signal.

RETURN VALUE: common error code

lmc1_MarkEntityFly

INTENTION: mark the appointed named object in database on fly mode
DEFINITION: int lmc1_MarkEntityFly(TCHAR* strEntName);
DISCRIPTION: after loading ezd file by lmc1_LoadEzdFile, user can use this function to fly mark the appointed object. The function will not return back until marking complete.
RETURN VALUE: common error code

lmc1_MarkLine

INTENTION: mark the appointed line
DEFINITION: int lmc1_MarkLine(double x1, double y1, double x2, double y2, int pen)；
x1, y1: //the coordinate of the starting point
x2, y2: //the coordinate of end point
pen: //Pen NO.
RETURN VALUE: common error code

lmc1_MarkPoint

INTENTION: mark the appointed point
DEFINITION： int lmc1_MarkPoint(double x, double y, double delay, int pen)；
x,y: //the coordinate of point
delay: //marking time
pen: //Pen NO.
DISCRIPTION: mark a point at the appointed station.
RETURN VALUE: common error code

lmc1_MarkPointBuf2

INTENTION: mark all appointed points
DEFINITION： int lmc1_ MarkPointBuf2(double ptBuf[][2],double dJumpSpeed, double dLaserOnTimeMs)
ptBuf //the coordinate group of points
ptBuf[n][0] //the X coordinate of point [n]
ptBuf[n][1] //the Y coordinate of point [n]
dJumpSpeed //the jump speed between points
dLaserOnTimeMs //the time of mark points, ms
RETURN VALUE: common error code

lmc1_IsMarking

INTENTION: check work state of control card
DEFINITION： bool lmc1_IsMarking()
DISCRIPTION: use lmc1_IsMarking checking control card working or not
RETURN VALUE: True means working.

lmc1_StopMark

INTENTION: Stop laser marking or red light
DEFINITION： int lmc1_StopMark()
DISCRIPTION: use lmc1_StopMark for stop laser shoot or red light.

lmc1_RedLightMark

INTENTION: mark the contour using indicated red light
DEFINITION: int lmc1_RedLightMark();
DISCRIPTION: mark the contour using indicated red light
RETURN VALUE: common error code

lmc1_RedLightMarkContour

INTENTION: use red light preview contour for all data on the SDK
DEFINITION: int lmc1_RedLightMarkContour();
DISCRIPTION: the preview looks like marking result, if there have a circle, so the red light will show a circle.
RETURN VALUE: common error code

lmc1_RedLightMarkByEnt

INTENTION: use red light preview data on the SDK
DEFINITION: int lmc1_RedLightMarkByEnt(TCHAR* strEntName, BOOL bContour)
strEntName //name of project
bContour //show contour or not, true is mean show contour, false is mean show mark position.
DISCRIPTION: preview marking position
RETURN VALUE: common error code

lmc1_GetFlySpeed

INTENTION: Get fly speed
DEFINITION: int lmc1_GetFlySpeed(double & FlySpeed)
FlySpeed //the speed of convery

DISCRIPTION: check the fly speed while not working, it is mean we cannot use it while laser is marking or red light working.

RETURN VALUE: common error code

5. File

lmc1_LoadEzdFile

INTENTION: open the appointed ezd file, and clear all the object in database.

DEFINITION: int lmc1_LoadEzdFile(TCHAR* strFileName);

DESCRIPTION: this function can open an ezd file that was build by user as a template. User need not set process parameters, because they will be loaded in from the template file.

RETURN VALUE: common error code

lmc1_GetPrevBitmap

INTENTION: Get the preview picture of all the objects in database.

DEFINITION: CBitmap* lmc1_GetPrevBitmap(HWND hwnd, int nBMPWidth, int nBMPHeight)；

Hwnd: //Window Handle that the preview shows in

nBMPWidth: //the preview picture's width in pixel

nBMPHeight: //the preview picture's height in pixel

DISCRIPTION: Call lmc1_GetPrevBitmap to get the preview picture of all the objects in database. It can be used to refresh the interface.

RETURN VALUE: return picture's pointer if successful and NULL if failed

lmc1_GetPrevBitmap2

INTENTION: Get the preview picture of all the objects in database.

DEFINITION: CBitmap* lmc1_GetPrevBitmap(int nBMPWidth, int nBMPHeight)；

nBMPWidth: // the preview picture's width in pixel

nBMPHeight: //the preview picture's height in pixel

DISCRIPTION: Call lmc1_GetPrevBitmap2 to get the preview picture of all the objects in database. It can be used to refresh the interface.

RETURN VALUE: return picture's pointer if successful and NULL if failed

lmc1_GetPrevBitmapByName2

INTENTION: Get the preview picture of all the objects in database.

DEFINITION: CBitmap* lmc1_lmc1_GetPrevBitmapByName2
(TCHAR*strEntName, int nBMPWidth, int nBMPHeight);
strEntName //Name of project
nBMPWidth: //the preview picture's width in pixel
nBMPHeight: // the preview picture's height in pixel
DISCRIPTION: Call lmc1_GetPrevBitmapByName2 to get the preview picture of all the objects in database. It can be used to refresh the interface.
RETURN VALUE: return picture's pointer if successful and NULL if failed

lmc1_SaveEntLibToFile

INTENTION: save all objects in database to the appointed .ezd file.
DEFINITION: int lmc1_SaveEntLibToFile(TCHAR* strFileName);
strFileName //name of ezd file
DISCRIPTION: save all objects in database to the appointed ezd file.
RETURN VALUE: common error code

Object

lmc1_GetEntSize

INTENTION: get the maximum and minimal coordinate of the appointed object.
DEFINITION: int lmc1_GetEntSize(TCHAR* pEntName, double& dMinx, double& dMiny, double& dMaxx, double& dMaxy, double& dZ);
pEntName //name of object
dMinx //minimal coordinate X
dMiny //minimal coordinate Y
dMaxx //maximum coordinate X
dMaxy //maximum coordinate Y
dZ //coordinate Z
DISCRIPTION: get the maximum and minimal coordinate of the appointed object.
RETURN VALUE: common error code

lmc1_MoveEnt

INTENTION: move object appointed distance
DEFINITION: int lmc1_GetEntSize(TCHAR* pEntName, double dMovex, double dMovey);
pEntName: //name of object
dMovex: //the X distance of object moving
dMovey: //the Y distance of object moving
DISCRIPTION: move object appointed distance

RETURN VALUE: common error code

lmc1_ScaleEnt

INTENTION: Scaling appointed object
DEFINITION: int lmc1_ScaleEnt(TCHAR* pEntName,
double dCenx,
double dCeny
double dScalex
double dScaley);

pEntName: //name of object
dCenx: //the scale center of X
dCeny: //the scale center of Y
dScalex //the scale proportion of X
dScaley //the scale proportion of Y

DISCRIPTION: scale appointed object according to center
RETURN VALUE: common error code

lmc1_MirrorEnt

INTENTION: Mirror appointed object
DEFINITION: int lmc1_MirrorEnt (TCHAR* pEntName,
double dCenx,
double dCeny
BOOL bMirrorX
BOOL bMirrorY);

pEntName: //name of object
dCenx: //the mirror center of X
dCeny: //the mirror center of Y
bMirrorX //Mirror X or not
bMirrorY //Mirror Y or not true is mean yes,

DISCRIPTION: Mirror appointed object
RETURN VALUE: common error code

lmc1_RotateEnt

INTENTION: Rotate appointed object
DEFINITION: int lmc1_RotateEnt (TCHAR* pEntName,
double dCenx,
double dCeny
double dAngle);

pEntName: // name of object
dCenx: //the rotate center of X
dCeny: //the rotate center of Y

dAngle //the angle of rotate.

DISCRIPTION: Rotate appointed object

RETURN VALUE: common error code

lmc1_CopyEnt

INTENTION: copy appointed object

DEFINITION: int lmc1_CopyEnt (TCHAR* pEntName,
TCHAR*pNewNetName)；

pEntName: //name of object

pNewEntName //name after copy

DISCRIPTION: use lmc1_CopyEnt copy and paste object, and named new object

RETURN VALUE: common error code

lmc1_GetEntityCount

INTENTION: get the total number of objects in database.

DEFINITION: int lmc1_GetEntityCount()；

DISCRIPTION: get the total number of objects in database.

RETURN VALUE: Total count of object in database

lmc1_GetEntityName

INTENTION: get the name of the object that has appointed serial number

DEFINITION: int lmc1_GetEntityName(int nEntityIndex,
TCHAR szEntName[256])；

NEntityIndex: //appoint the serial number， 0－(total number-1). The total objects count can be got by lmc1_GetEntityCount.

szEntName: // name of appointed object

DISCRIPTION: get the name of the object that has appointed serial number

RETURN VALUE: common error code

lmc1_SetEntityName

INTENTION: set name for object
DEFINITION: int lmc1_SetEntityName(int nEntityIndex, TCHAR*pEntName)；
nEntityLndex //set serial number for object, the range is object total,(the total number will be get by lmc1_GeEntityCount)
pEntName //set name for object
DISCRIPTION: set name for object.
RETURN VALUE: Common err code

lmc1_ChangeEntName

INTENTION: change name for object
DEFINITION: int lmc1_ChangeEntName(TCHAR*pEntName,
TCHAR*pNewEntName)；
pEntName //the object name before change
pNewEntName //the object name after change
DISCRIPTION: change a new name for object, if few object have same name, all of them will be change.
RETURN VALUE: Common err code

lmc1_GroupEnt

INTENTION: Group
DEFINITION: int lmc1_GroupEnt(TCHAR*pEntName1
TCHAR*pEntName2
TCHAR*pEntNameNew
Int pen)；
pEntName1 //name of group1,
pEntName2 //name of group2
pEntNameNew //name of new group
pen //pen no for new group
DISCRIPTION: group 2 objects, and set new name and pen no for it.
RETURN VALUE: Common err code

lmc1_UnGroupEnt

INTENTION: UnGroup
DEFINITION: int lmc1_UnGroupEnt(TCHAR*pGroupEntName)；
pGroupEntName //name of group
DISCRIPTION: Ungroup object
RETURN VALUE: Common err code

lmc1_GetBitmapEntParam

INTENTION: get parameter from bitmap
DEFINITION: int lmc1_ GetBitmapEntParam（TCHAR* strEntName
TCHAR BmpPath [256],
int& nBmpAttrib,
int& nScanAttrib,
double& dBrightness,
double& dContrast,
double& dPointTime,
int& nImportDpi）;

strEntName //name of bitmap
BmpPath //path of bitmap
nBmpAttrib //parameter
nScanAttrib //scan parameter
dBrightness //brightness setting[-1,1]
dContrast //contrast ratio setting[-1,1]
dPointTime //mark point time setting
nImportDpi //DPI

const int BMPSCAN_INVERT = 0x0001; //reverse bitmap
const int BMPSCAN_GRAY = 0x0002; //Gary
const int BMPSCAN_LIGHT = 0x0004; //Brightness
const int BMPSCAN_DITHER = 0x0010; //Dither
const int BMPSCAN_BIDIR = 0x1000; //double scan
const int BMPSCAN_YDIR = 0x2000; //Y scan
const int BMPSCAN_DRILL = 0x4000; //points mode
const int BMPSCAN_POWER = 0x8000; //adjust power
const int BMPATTRIB_DYNFILE = 0x1000;//dynamic file
const int BMPATTRIB_IMPORTFIXED_WIDTH = 0x2000;//fixed width of file
const int BMPATTRIB_IMPORTFIXED_HEIGHT = 0x4000;//fixed high of file
const int BMPATTRIB_IMPORTFIXED_DPI = 0x8000;//fixed DPI

DISCRIPTION: get parameter for bitmap
RETURN VALUE: Common err code

lmc1_GetBitmapEntParam2

INTENTION: get parameter from bitmap
DEFINITION: int lmc1_ GetBitmapEntParam2（TCHAR* strEntName

TCHAR* strBmpPath,
int nBmpAttrib,
int nScanAttrib,
double dBrightness,
double dContrast,
double dPointTime,
int nImportDpi）；

strEntName //name of bitmap
strBmpPath //path of bitmap
nBmpAttrib //bitmap parameter
nScanAttrib //scan parameter
dBrightness //brightness setting[-1,1]
dContrast //contrast ratio setting[-1,1]
dPointTime //mark point time setting
nImportDpi //DPI

DISCRIPTION: get parameter for bitmap
RETURN VALUE: Common err code

lmc1_MoveEntityBefore

INTENTION: move object forward
DEFINITION: int lmc1_MoveEntityBefore(int nMoveEnt, int nGoalEnt)；
nMoveEnt //the position of the object that you want to move
nGoalEnt //the position of the object that you want to move to
DISCRIPTION: move object forward, change the mark order
RETURN VALUE: Common err code

lmc1_MoveEntityAfter

INTENTION: Backward move object
DEFINITION: int lmc1_MoveEntityAfter(int nMoveEnt, int nGoalEnt)；
nMoveEnt //the position of the object that you want to move
nGoalEnt //the position of the object that you want to move to
DISCRIPTION: backward move object, change the mark order
RETURN VALUE: Common err code

lmc1_ReverseAllEntOrder

INTENTION: change the order for all object in the list
DEFINITION: int lmc1_ReverseAllEntOrder()；
DISCRIPTION: change mark order on the marking lista
RETURN VALUE: Common err code

6. Port

lmc1_ReadPort

INTENTION: read the input port of the lmc1
DEFINITION: int lmc1_ReadPort(WORD& data);
data: //the data in input port
DISCRIPTION: call lmc1_ReadPort to read the data from input ports
RETURN VALUE: common error code

lmc1_WritePort

INTENTION: write data to output port on the lmc1
DEFINITION: int lmc1_WritePort(WORD data);
data: //the data to output ports
DISCRIPTION: call lmc1_WritePort to write data to the output port
RETURN VALUE: common error code

lmc1_GetOutPort

INTENTION: read output port from control card
DEFINITION: int lmc1_GetOutPort(WORD&data);
data //data from output IO
DISCRIPTION: call lmc1_WritePort to write data to the output port
//Bit =0 low
//Bit =1 high
RETURN VALUE: Common err code

lmc1_LaserOn

INTENTION: control laser shoot
DEFINITION: int lmc1_LaserOn(BOOL Open);
Open //control laser shoot
DISCRIPTION: call lmc1_LaserOn for control laser on
RETURN VALUE: Common err code

7. Text

lmc1_ChangeTextByName

INTENTION: change the content of the text with appointed name.
DEFINITION: int lmc1_ChangeTextByName(TCHAR* strTextName, TCHAR* strTextNew);
strTextName //the name of text object whose content will be changed
strTextNew //new content of text
DISCRIPTION: after loading ezd file by lmc_LoadEzdFile, user can use this function to change the content of appointed text object before marking it.
RETURN VALUE: common error code

lmc1_GetTextByName

INTENTION: get the content according to text name
DEFINITION: int lmc1_GetTextByName(TCHAR* strTextName, TCHAR* strEntText[256]);
strTextName //the name of text object whose content will be got
strEntText //content of text
DISCRIPTION: get the content according to text name
RETURN VALUE: common error code

lmc1_TextResetSn

INTENTION: Reset serial number
DEFINITION: int lmc1_TextResetSnTCHAR* pTextName);
pTextName //name of text
DISCRIPTION: Reset serial number to start
RETURN VALUE: common error code

lmc1_GetFontRecordCount

INTENTION: get the font count from PC system
DEFINITION: int lmc1_GetFontRecordCount(int& nFontNum);
nFontNum //count of system font
DISCRIPTION: get the number of font that software already have
RETURN VALUE: common error code

lmc1_GetFontRecord

INTENTION: get the parameter of the font that support by PC system

DEFINITION: int lmc1_GetFontRecord (int& nFontIndex, TCHAR szFontName[256], DWORD& dwFontAttrib);

nFontIndex //serial number of font
szFontName //name of font
dwFontAttrib //Type parameter of font

DISCRIPTION: Gets the name and type parameter of the font that specifies the ordinal number.

RETURN VALUE: common error code

lmc1_GetAllFontRecord

INTENTION: Gets all fonts parameters supported by the current system.

DEFINITION: int lmc1_FontRecord*lmc1_GetAllFontRecord (int& nFontNum,);

nFontNum //count of font

```
// Font type attribute definition
#define FONTATB_JSF          0x0001          //JczSingle
#define FONTATB_TTF          0x0002          //TrueType
#define FONTATB_DMF          0x0004          //DotMatrix
#define FONTATB_BCF          0x0008          //BarCode
#define FONTATB_SHX          0x0010          //SHX

//Font record
struct lmc1_FontRecord
{
    TCHAR    szFontName[256];      //name of font
    DWORD    dwFontAttrib;         //font attribute
};
```

DISCRIPTION: Gets all fonts parameters supported by the current system.

RETURN VALUE: common error code

lmc1_SetFontParam

INTENTION: set the parameter of font

DEFINITION: int lmc1_SetFontParam(

TCHAR* strFontName
double dCharHeight,
double dCharWidth,
double dCharAngle,
double dCharSpace,
double dLineSpace,
BOOL bEqualCharWidth);

strFontName: //name of font
dCharHeight: //height of character
dCharWidth: //width of character
dCharAngle: //angle of character
dCharSpace: //distance between the characters
dLineSpace: //distance between the lines.
bEqualCharWidth: //enable the same characters width mode

DISCRIPTION: call lmc1_SetFontParam to set parameters of font. The parameters will be used for the latter text object added in database.

RETURN VALUE: common error code

lmc1_SetTextEntParam

INTENTION: Sets the font parameter of the specified text

DEFINITION: int lmc1_SetTextEntParam(TCHAR* strTextName
double dCharHeight,
double dCharWidth,
double dCharAngle,
double dCharSpace,
double dLineSpace,
BOOL bEqualCharWidth);

strTextName: //name of Text
dCharHeight: //height of character
dCharWidth: //width of character
dCharAngle: //angle of character
dCharSpace: //distance between the characters
dLineSpace: //distance between the lines.
bEqualCharWidth: //enable the same characters width mode

DISCRIPTION: Sets the font parameter of the specified text

RETURN VALUE: common error code

lmc1_SetTextEntParam2

INTENTION: Sets the font parameter of the specified text

DEFINITION: int lmc1_SetTextEntParam2(TCHAR* strTextName,
TCHAR*strFontName,

double dCharHeight,
double dCharWidth,
double dCharAngle,
double dCharSpace,
double dLineSpace,
double dSpaceWidth
BOOL bEqualCharWidth);

strTextName: //name of Text
strFontName //name of font
dCharHeight: //height of character
dCharWidth: //width of character
dCharAngle: //angle of character
dCharSpace: //distance between the characters
dLineSpace: //distance between the lines.
dSpaceWidth //space width
bEqualCharWidth: //enable the same characters width mode

DISCRIPTION: Sets the font parameter of the specified text
RETURN VALUE: common error code

lmc1_GetTextEntParam

INTENTION: Gets the font parameter of the specified text
DEFINITION: int lmc1_SetTextEntParam2(TCHAR* strTextName
TCHAR sFontName[256]
double& dCharHeight,
double& dCharWidth,
double& dCharAngle,
double& dCharSpace,
double& dLineSpace,
BOOL& bEqualCharWidth);

strTextName: //name of Text
sFontName //name of font
dCharHeight: //height of character
dCharWidth: //width of character
dCharAngle: //angle of character
dCharSpace: //distance between the characters
dLineSpace: //distance between the lines.
bEqualCharWidth: //enable the same characters width mode

DISCRIPTION: Gets the font parameter of the specified text
RETURN VALUE: common error code

lmc1_GetTextEntParam2

INTENTION: Gets the font parameter of the specified text

DEFINITION: int lmc1_GetTextEntParam2(TCHAR* strTextName,
TCHAR sFontName[256]
double& dCharHeight,
double& dCharWidth,
double& dCharAngle,
double& dCharSpace,
double& dLineSpace,
double& dSpaceWidth
BOOL& bEqualCharWidth);
strTextName: //name of Text
sFontName //name of font
dCharHeight: //height of character
dCharWidth: //width of character
dCharAngle: //angle of character
dCharSpace: //distance between the characters
dLineSpace: //distance between the lines.
dSpaceWidth //space width
bEqualCharWidth: //enable the same characters width mode
DISCRIPTION: Gets the font parameter of the specified text
RETURN VALUE: common error code

8. Pen

lmc1_GetPenParam

INTENTION: get the parameter of appointed pen
DEFINITION: int lmc1_GetPenParam(
int nPenNo, // Pen's NO. (0-255)
int& nMarkLoop, //mark times
double& dMarkSpeed, //speed of marking mm/s
double& dPowerRatio, // power ratio of laser (0-100%)
double& dCurrent, //current of laser (A)
int& nFreq, // frequency of laser HZ
int& nQPulseWidth, //width of Q pulse (us)
int& nStartTC, // Start delay (us)
int& nLaserOffTC, //delay before laser off (us)
int& nEndTC, // marking end delay (us)
int& nPolyTC , //delay for corner (us)
double& dJumpSpeed, //speed of jump without laser (mm/s)
int& nJumpPosTC, //delay about jump position (us)
int& nJumpDistTC, //delay about the jump distance (us)

```
    double&     dEndComp,       // compensate for end (mm)
    double&     dAccDist,       // distance of speed up (mm)
    double&     dPointTime,     //delay for point mark (ms)
    BOOL&       bPulsePointMode,    //pulse for point mark mode
    int&        nPulseNum,      //the number of pulse
    double&     dFlySpeed       //speed of production line
    );
```

DISCRIPTION: call lmc1_GetParam to get the parameter of appointed pen in database

RETURN VALUE: common error code

lmc1_GetPenParam2

INTENTION: get the parameter of appointed pen

DEFINITION: int lmc1_GetPenParam2(

```
    int         nPenNo,             // Pen's NO. (0-255)
    int&        nMarkLoop,          //mark times
    double&     dMarkSpeed,         //speed of marking mm/s
    double&     dPowerRatio,        // power ratio of laser (0-100%)
    double&     dCurrent,           //current of laser (A)
    int&        nFreq,              // frequency of laser HZ
    double&     dQPulseWidth,       //width of Q pulse (us)
    int&        nStartTC,           // Start delay (us)
    int&        nLaserOffTC,        //delay before laser off (us)
    int&        nEndTC,             // marking end delay (us)
    int&        nPolyTC ,           //delay for corner (us)
    double&     dJumpSpeed,         //speed of jump without laser (mm/s)
    int&        nJumpPosTC,         //delay about jump position (us)
    int&        nJumpDistTC,        //delay about the jump distance (us)
    double&     dPointTime,         //delay for point mark (ms)
    int&        nSpiWave,           //Select SPI wave
    BOOL&       bWobbleMode,        //Wobble mode
    double&     bWobbleDiameter,    //Wobble diameter
    double&     bWobbleDist         //Wobble distance
    );
```

DISCRIPTION: call lmc1_GetParam2 to get the parameter of appointed pen in database

RETURN VALUE: common error code

lmc1_GetPenParam4

INTENTION: get the parameter of appointed pen

DEFINITION: int lmc1_GetPenParam4(

```
    int         nPenNo,             // Pen's NO. (0-255)
```

```
TCHAR   sPenName[256]       //pen name, default name is default
int&       clr                    // pen color
BOOL&   bDisableMark,       //enable pen or not, true is mean close
BOOL&   bUseDefParam        //enable default param or not
int&       nMarkLoop,           //mark times
double&    dMarkSpeed,          //speed of marking mm/s
double&    dPowerRatio,         // power ratio of laser (0-100%)
double&    dCurrent,            //current of laser (A)
int&       nFreq,               // frequency of laser HZ
double&    dQPulseWidth,        //width of Q pulse (us)
int&       nStartTC,            // Start delay (us)
int&       nLaserOffTC,         //delay before laser off (us)
int&       nEndTC,              // marking end delay (us)
int&       nPolyTC ,            //delay for corner (us)
double&    dJumpSpeed,          //speed of jump without laser (mm/s)
int&       nMinJumpDelayTCUs // the mix jump delay(us)
int&       nMaxJumpDelayTCUs //the max jump delay(us)
double&    dJumpLengthLimit,    //the limit length of jump
double&    dPointTime,          //time for point mark (ms)
BOOL&      nSpiSpiContinueMode, // SPI continue mode
int&       nSpiWave,            //Select SPI wave
int&       nYagMarkMode,        // YAG fast hatch mode
BOOL&      bPulsePointMode,     //pulse point mode
Int&       nPulseNum,           //pulse point count
BOOL&      bEnableACCMode,      //enable ACC mode
double &   dEndComp,            //end comp
double&    dAccDist,            //ACC distance
double&    dBreakAngle，        //break angle
BOOL&      bWobbleMode,         //Wobble mode
double&    bWobbleDiameter,     //Wobble diameter
double&    bWobbleDist          //Wobble distance
);
```

DISCRIPTION: call lmc1_GetParam4 to get the parameter of appointed pen in database

RETURN VALUE: common error code

lmc1_SetPenParam

INTENTION: Set the pan parameter

DEFINITION: int lmc1_SetPenParam(

```
int         nPenNo,          // Pen's NO. (0-255)
int&        nMarkLoop,       //mark times
double&     dMarkSpeed,      //speed of marking mm/s
```

```
double&     dPowerRatio,        // power ratio of laser (0-100%)
double&     dCurrent,           //current of laser (A)
int&        nFreq,              // frequency of laser HZ
int&        nQPulseWidth,       //width of Q pulse (us)
int&        nStartTC,           // Start delay (us)
int&        nLaserOffTC,        //delay before laser off (us)
int&        nEndTC,             // marking end delay (us)
int&        nPolyTC,            //delay for corner (us)
double&     dJumpSpeed,         //speed of jump without laser (mm/s)
int&        nJumpPosTC,         //delay about jump position (us)
int&        nJumpDistTC,        //delay about the jump distance (us)
double&     dEndComp,           //compensate for end (mm)
double&     dAccDist,           // distance of speed up (mm)
double&     dPointTime,         //delay for point mark (ms)
BOOL&       bPulsePointMode,    //pulse for point mark mode
int&        nPulseNum,          //the number of pulse
double&     dFlySpeed           //speed of production line
);
```

DISCRIPTION: call lmc1_SetPenParam to set the parameters of appointed pen in database

RETURN VALUE: common error code

lmc1_SetPenParam2

INTENTION: Set the pan parameter

DEFINITION: int lmc1_SetPenParam(

```
int         nPenNo,             // Pen's NO. (0-255)
int         nMarkLoop,          //mark times
double      dMarkSpeed,         //speed of marking mm/s
double      dPowerRatio,        // power ratio of laser (0-100%)
double      dCurrent,           //current of laser (A)
int         nFreq,              // frequency of laser HZ
double      dQPulseWidth,       //width of Q pulse (us)
int         nStartTC,           // Start delay (us)
int         nLaserOffTC,        //delay before laser off (us)
int         nEndTC,             // marking end delay (us)
int         nPolyTC,            //delay for corner (us)
double      dJumpSpeed,         //speed of jump without laser (mm/s)
int         nJumpPosTC,         //delay about jump position (us)
int         nJumpDistTC,        //delay about the jump distance (us)
double&     dPointTime,         //delay for point mark (ms)
int         nSpiWave,           //select SPI wave
BOOL&       bWobbleMode,        //Wobble mode
double&     bWobbleDiameter,    //Wobble diameter
```

double& bWobbleDist //Wobble distance);

DISCRIPTION: call lmc1_SetPenParam2 to set the parameters of appointed pen in database

RETURN VALUE: common error code

lmc1_SetPenParam4

INTENTION: Set the pan parameter

DEFINITION: int lmc1_SetPenParam4(

```
int          nPenNo,              // Pen's NO. (0-255)
TCHAR* sPenName,          //pen name, default name is default
int          clr, //pen color
  BOOL    bDisableMark,      //enable pen or not, true is mean close
  BOOL    bUseDefParam,      //enable default or not
int          nMarkLoop,         //mark times
double     dMarkSpeed,        //speed of marking mm/s
double     dPowerRatio,       // power ratio of laser (0-100%)
double     dCurrent,          //current of laser (A)
int          nFreq,             // frequency of laser HZ
double     dQPulseWidth,      //width of Q pulse (us)
int          nStartTC,          // Start delay (us)
int          nLaserOffTC,       //delay before laser off (us)
int          nEndTC,            // marking end delay (us)
int          nPolyTC,           //delay for corner (us)
double     dJumpSpeed,        //speed of jump without laser (mm/s)
int&        nMinJumpDelayTCUs // the mix jump delay(us)
int&        nMaxJumpDelayTCUs //the max jump delay(us)
double&   dJumpLengthLimit, //the limit length of jump
double&   dPointTime,        //time for point mark (ms)
BOOL&    nSpiSpiContinueMode, // SPI continue mode
int&         nSpiWave,          //Select SPI wave
int&         nYagMarkMode,    // YAG fast hatch mode
BOOL&     bPulsePointMode, //pulse point mode
Int&         nPulseNum,         //pulse point count
BOOL&      bEnableACCMode,//enable ACC mode
double &   dEndComp,          //end comp
double&    dAccDist,          //ACC distance
double&    dBreakAngle，      //break angle
BOOL&      bWobbleMode,       //Wobble mode
 double&   bWobbleDiameter,  //Wobble diameter
 double&   bWobbleDist        //Wobble distance);
```

DISCRIPTION: call lmc1_SetPenParam4 to set the parameters of appointed pen in database

RETURN VALUE: common error code

lmc1_SetPenDisableState

INTENTION: enable pen or not

DEFINITION: int lmc1_SetPenDisableState(int nPenNo, //the pen need to be set(0-255)
BOOL bDisableMark, //enable pen or not, true is mean close)

DISCRIPTION: call lmc1_SetPenDisableState to open or close pen.

RETURN VALUE: common error code

lmc1_GetPenDisableState

INTENTION: Get pen state

DEFINITION: int lmc1_GetPenDisableState(int nPenNo, //the pen need to be set(0-255)
BOOL bDisableMark, //enable pen or not, true is mean close);

DISCRIPTION: call lmc1_GetPenDisableState to open or close pen.

RETURN VALUE: common error code

lmc1_GetPenNumberFromName

INTENTION: Get pen no from pen name

DEFINITION: int lmc1_ GetPenNumberFromName(
TCHAR* sPenName); //pen name

DISCRIPTION: call lmc1_GetPenNumberFromName to get pen no

RETURN VALUE: pen no (0-255)

lmc1_GetPenNumberFromEnt

INTENTION: Get pen no from appointed object

DEFINITION: int lmc1_ GetPenNumberFromEnt(
TCHAR* sPenName); //appointed object name

DISCRIPTION: call lmc1_ GetPenNumberFromEnt to get pen no

RETURN VALUE: pen no (0-255)

lmc1_SetEntAllChildPen

INTENTION: Get pen no from appointed object

DEFINITION: int lmc1_SetEntAllChildPen(
TCHAR* sEntName, //appointed object name
Int nPenNo, // the setting pen no(0-255))

DISCRIPTION: Get pen no from appointed object

RETURN VALUE: common error code

9. Hatch

lmc1_SetHatchParam

INTENTION: Set the hatch parameter

DEFINITION: int lmc1_SetHatchParam (

```
BOOL   bEnableContour,   //enable the contour of object to be marked
int    bEnableHatch1,    //enable hatch NO. 1
int    nPenNo1,          //set the pen of hatch NO. 1
int    nHatchAttrib1,    //set the attribute of hatch NO. 1
double dHatchEdgeDist1,  //set the distance between hatch line and
                         contour of hatch NO. 1
double dHatchLineDist1,  //set the distance between two line of hatch
                           NO. 1 .
double dHatchStartOffset1,  //set the start offset of hatch NO. 1
double dHatchEndOffset1,    //set the end offset of hatch NO. 1
double dHatchAngle1,        //set the hatch angle of hatch NO. 1
int    bEnableHatch2,       //enable hatch1 NO.2
int    nPenNo2,             //hatch pen
int    nHatchAttrib2,       //hatch attribute
double dHatchEdgeDist2,     //hatch edge distance
double dHatchLineDist2.     //hatch line distance
double dHatchStartOffset2,  //hatch start offset distance
double dHatchEndOffset2,    //hatch end offset distance
double dHatchAngle2         //angle of hatch line
);
 bEnableContour             // enable contour or not
 bEnableHatch1              //enable hatch
 nPenNo1                    //hatch pen no
 nHatchAttrib1:             //attribute of hatch，which is a combination of
                              the following values:
const int HATCHATTRIB_ALLCALC = 0x01; //compute all object as one
const int HATCHATTRIB_BIDIR   = 0x08; // reciprocating hatch
const int HATCHATTRIB_EDGE    = 0x02; // re-mark the edge
const int HATCHATTRIB_LOOP    = 0x10; // ring-like hatch
 dHatchEdgeDist1    // hatch edge distance
dHatchLineDist1     //hatch line distance
 dHatchStartOffset1  //hatch start offset distance
 dHatchEndOffset1   //hatch end offset distance
 dHatchAngle1       //angle of hatch line
```

DISCRIPTION: call lmc1_SetHatchParam to set the parameters of hatch. The parameters will be used for the latter hatched object.

RETURN VALUE: common error code

lmc1_SetHatchParam2

INTENTION: Set the hatch parameter

DEFINITION: int lmc1_SetHatchParam2(

```
BOOL     bEnableContour,    //enable the contour of object to be marked
 int nParamIndex,            //hatch order number is 1,2,3
  int bEnableHatch,          //enable hatch
  int nPenNo,                //hatch pen no
  int nHatchType,            // Hatch type:0 unidirectional, 1 bidirectional, 2
                                return, 3 bow, 4 bow not reverse
  BOOL bHatchAllCalc,        // compute all object or not
  BOOL bHatchEdge,           //around edge once time
  BOOL bHatchAverageLine,// Automatic average distribution line
   double dHatchAngle,       //hatch line angle
  double dHatchLineDist,     // hatch edge distance
  double dHatchEdgeDist,     // hatch line distance
   double dHatchStartOffset, // hatch start offset distance
  double dHatchEndOffset,    // hatch end offset distance
  double dHatchLineReduction,//line reduction
  double dHatchLoopDist,     //ring line distance
  int nEdgeLoop,             //ring count
  BOOL nHatchLoopRev,        //loop reverse
  BOOL bHatchAutoRotate,     //enable auto rotate angle or not
  double dHatchRotateAngle   //enable rotate angle    );
```

DISCRIPTION: call lmc1_SetHatchParam2 to set the parameters of hatch. The parameters will be used for the latter hatched object.

RETURN VALUE: common error code

lmc1_SetHatchEntParam

INTENTION: Set the hatch parameter

DEFINITION: int lmc1_SetHatchEntParam(

```
 TCHAR* pHatchName,         //name of hatch object
BOOL     bEnableContour,    //enable the contour of object to be marked
 int nParamIndex,           //hatch order number is 1,2,3
  int bEnableHatch,         //enable hatch
  int nPenNo,               //hatch pen no
  int nHatchType,           // Hatch type:0 unidirectional, 1 bidirectional, 2
                               return, 3 bow, 4 bow not reverse
  BOOL bHatchAllCalc,       // compute all object or not
  BOOL bHatchEdge,          //around edge once time
  BOOL bHatchAverageLine,// Automatic average distribution line
```

double dHatchAngle, //hatch line angle
double dHatchLineDist, // hatch edge distance
double dHatchEdgeDist, // hatch line distance
double dHatchStartOffset, // hatch start offset distance
double dHatchEndOffset, // hatch end offset distance
double dHatchLineReduction,//line reduction
double dHatchLoopDist, //ring line distance
int nEdgeLoop, //ring count
BOOL nHatchLoopRev, //loop reverse
BOOL bHatchAutoRotate, //enable auto rotate angle or not
double dHatchRotateAngle //enable rotate angle);

DISCRIPTION: edit hatch parameter for hatch object
RETURN VALUE: common error code

lmc1_SetHatchEntParam2

INTENTION: Set the hatch parameter
DEFINITION: int lmc1_SetHatchEntParam2(
TCHAR* pHatchName, //name of hatch object
BOOL bEnableContour, //enable the contour of object to be marked
int nParamIndex, //hatch order number is 1,2,3
int bEnableHatch, //enable hatch
int nPenNo, //hatch pen no
BOOL bContourFirst //mark contour first
int nHatchType, // Hatch type:0 unidirectional, 1 bidirectional, 2 return, 3 bow, 4 bow not reverse
BOOL bHatchAllCalc, // compute all object or not
BOOL bHatchEdge, //around edge once time
BOOL bHatchAverageLine,// Automatic average distribution line
double dHatchAngle, //hatch line angle
double dHatchLineDist, // hatch edge distance
double dHatchEdgeDist, // hatch line distance
double dHatchStartOffset, // hatch start offset distance
double dHatchEndOffset, // hatch end offset distance
double dHatchLineReduction,//line reduction
double dHatchLoopDist, //ring line distance
int nEdgeLoop, //ring count
BOOL nHatchLoopRev, //loop reverse
BOOL bHatchAutoRotate, //enable auto rotate angle or not
double dHatchRotateAngle //enable rotate angle
BOOL bHatchCrossMode, // corss hatch mode
int dCycCount //count);

DISCRIPTION: edit hatch parameter for hatch object
RETURN VALUE: common error code

lmc1_GetHatchEntParam

INTENTION: Get the hatch parameter

DEFINITION: int lmc1_GetHatchEntParam(

```
TCHAR* pHatchName,        //name of hatch object
BOOL&      bEnableContour, //enable the contour of object to be marked
int nParamIndex,              //hatch order number is 1,2,3
  int& bEnableHatch,          //enable hatch
  int& nPenNo,                //hatch pen no
  int& nHatchType,            // Hatch type:0 unidirectional, 1 bidirectional,
                                2 return, 3 bow, 4 bow not reverse
  BOOL& bHatchAllCalc,        // compute all object or not
  BOOL& bHatchEdge,           //around edge once time
  BOOL&  bHatchAverageLine,// Automatic  average  distribution  line
double& dHatchAngle,          //hatch line angle
  double& dHatchLineDist,     // hatch edge distance
  double& dHatchEdgeDist,     // hatch line distance
   double& dHatchStartOffset, // hatch start offset distance
  double& dHatchEndOffset,    // hatch end offset distance
  double& dHatchLineReduction,//line reduction
  double& dHatchLoopDist,     //ring line distance
  int& nEdgeLoop,             //ring count
  BOOL& nHatchLoopRev,        //loop reverse
  BOOL& bHatchAutoRotate,     //enable auto rotate angle or not
  double& dHatchRotateAngle   //enable rotate angle) ;
```

DISCRIPTION: Get hatch parameter for hatch object

RETURN VALUE: common error code

lmc1_GetHatchEntParam2

INTENTION: Get the hatch parameter

DEFINITION: int lmc1_GetHatchEntParam2(

```
TCHAR* pHatchName,        //name of hatch object
BOOL&      bEnableContour, //enable the contour of object to be marked
int nParamIndex,              //hatch order number is 1,2,3
  int& bEnableHatch,          //enable hatch
BOOL& bContourFirst         //mark contour first
  int& nPenNo,                //hatch pen no
  int& nHatchType,            // Hatch type:0 unidirectional, 1 bidirectional,
                                2 return, 3 bow, 4 bow not reverse
  BOOL& bHatchAllCalc,        // compute all object or not
  BOOL& bHatchEdge,           //around edge once time
  BOOL&  bHatchAverageLine,// Automatic  average  distribution  line
```

```
double& dHatchAngle,           //hatch line angle
  double& dHatchLineDist,      // hatch edge distance
  double& dHatchEdgeDist,      // hatch line distance
   double& dHatchStartOffset, // hatch start offset distance
  double& dHatchEndOffset,    // hatch end offset distance
  double& dHatchLineReduction,//line reduction
  double& dHatchLoopDist,      //ring line distance
  int& nEdgeLoop,              //ring count
  BOOL& nHatchLoopRev,         //loop reverse
  BOOL& bHatchAutoRotate,      //enable auto rotate angle or not
  double& dHatchRotateAngle    //enable rotate angle
) ;
```

DISCRIPTION: Get hatch parameter for hatch object
RETURN VALUE: common error code

lmc1_HatchEnt

INTENTION: Hatch object
DEFINITION: int lmc1_HatchEnt(TCHAR*pEntName,
TCHAR*pEntNameNew)
PEntName //the object name
pEntNameNew //the object name after hatch
DISCRIPTION: hatch object
RETURN VALUE: common error code

lmc1_UnHatchEnt

INTENTION: Deleted hatch
DEFINITION: int lmc1_UnHatchEnt(TCHAR*pHatchEntName)
pHatchEntName //the object name after hatch
DISCRIPTION: delete hatch
RETURN VALUE: common error code

Add or Delete Object

lmc1_ClearEntLib

INTENTION: clear all object in database
DEFINITION: int lmc1_ClearEntLib();
DISCRIPTION: call lmc1_ClearEntLib to clear all objects in database.
RETURN VALUE: common error code

lmc1_DeleteEntLib

INTENTION: delete object in database

DEFINITION: int lmc1_DeleteEnt(TCHAR*pEntName); the name of object which one need to be delete.

DISCRIPTION: delete objects in database.

RETURN VALUE: common error code

lmc1_AddTextToLib

INTENTION: add new text object into database.

DEFINITION: int lmc1_AddTextToLib(

```
        TCHAR* pStr,
        TCHAR* pEntName,
        double dPosX,
        double dPosY,
        double dPosZ,
        int       nAlign
        double dTextRotateAngle,
        int nPenNo,
        BOOL bHatchText          //hatch the text object or not.
        );
```

pStr: //the character string

pEntName: //the name of character string object

dPosX //left-bottom point's X coordinate of the character string object

dPosY //left-bottom point's Y coordinate of the character string object

dPosZ //Z coordinate of the character string object

nAlign //align way 0－8

```
                //meaning of the align way:
                //     6 ---    5 ---    4
                //     |                  |
                //     |                  |
                //     7         8        3
                //     |                  |
                //     |                  |
                //     0 ----- 1 --- -- 2
```

dTextRotateAngle //rotation angle (in radian) that the character string object rotates around base point.

nPenNo //the number of pen to mark text

bHatchText //hatch the text object or not

DISCRIPTION: call lmc1_AddTextToLib to add new text object into database.

RETURN VALUE: common error code

```
#define CIRTEXTFLAG_REVERSE  0x0001 //reverse
#define CIRTEXTFLAG_UPDOWN  0x0002 //up and down reverse
```

lmc1_AddCircleTextToLib

INTENTION: add new circle object into database.
DEFINITION: int lmc1_AddCircleTextToLib(TCHAR *pStr,
TCHAR* pEntName,
double dCenX,
double dCenY,
double dCenZ,
int nPenNo,
int bHatchText,
double dCirDiameter,
double dCirBaseAngle,
BOOL bCirEnableAngleLimit,
double dCirAngleLimit,
int nCirTextFlag);

pStr //character string
pEntName //character string name
dCenX // left-bottom point's X coordinate of the character string object
dCenY // left-bottom point's Y coordinate of the character string object
dCenZ // z coordinate of the character string object
nPenNo //Pen no of the text object
bHatchText //hatch text or not
dCirDiameter //circle diameter
dCirBaseAngle //text basic angle
bCirEnableAngleLimit //enable angle limit or not
dCirAngleLimit //the angle limit
inCirTextFlag //the text direction on the circle

DISCRIPTION: add new circle object into database.
RETURN VALUE: common error code

lmc1_GetCircleTextParam

INTENTION: Get circle text parameter
DEFINITION: int lmc1_GetCircleTextParam(TCHAR* pEntName,
double& dCenX,
double& dCenY,
double& dCenZ,
double& dCirDiameter,
double& dCirBaseAngle,
BOOL& bCirEnableAngleLimit,

double& dCirAngleLimit,
int& nCirTextFlag);

pEntName //character string name
dCenX // left-bottom point's X coordinate of the character string object
dCenY // left-bottom point's Y coordinate of the character string object
dCenZ // z coordinate of the character string object
dCirDiameter //circle diameter
dCirBaseAngle //text basic angle
bCirEnableAngleLimit //enable angle limit or not
dCirAngleLimit //the angle limit
inCirTextFlag //the text direction on the circle

DISCRIPTION: Get circle text parameter.
RETURN VALUE: common error code

lmc1_SetCircleTextParam

INTENTION: Set circle text parameter
DEFINITION: int lmc1_SetCircleTextParam(TCHAR* pEntName,
double dCenX,
double dCenY,
double dCenZ,
double dCirDiameter,
double dCirBaseAngle,
BOOL bCirEnableAngleLimit,
double dCirAngleLimit,
int nCirTextFlag);

pEntName //character string name
dCenX // left-bottom point's X coordinate of the character string object
dCenY // left-bottom point's Y coordinate of the character string object
dCenZ // z coordinate of the character string object
dCirDiameter //circle diameter
dCirBaseAngle //text basic angle
bCirEnableAngleLimit //enable angle limit or not
dCirAngleLimit //the angle limit
inCirTextFlag //the text direction on the circle

DISCRIPTION: Set circle text parameter.
RETURN VALUE: common error code

lmc1_AddCurveToLib

INTENTION: add new curve object into database.
DEFINITION: int lmc1_AddCurveToLib(
double ptBuf[][2], //array of the curve vertex

int ptNum, //number of the curve vertex
TCHAR* pEntName, //name of the curve object
int nPenNo, //the pen number of curve object
int bHatch //hatch the curve object or not
);

DISCRIPTION: call lmc1_AddCurveToLib to add curve object into database.
RETURN VALUE: common error code

lmc1_AddPointToLib

INTENTION: add a group points into database.
DEFINITION: int lmc1_AddPointToLib(
double ptBuf[][2], //array of the point
int ptNum, //number of the point
TCHAR* pEntName, //name of the point object
int nPenNo, //the pen number of point object
);

DISCRIPTION: call lmc1_AddPointToLib to add point object into database.
RETURN VALUE: common error code

lmc1_AddDelayToLib

INTENTION: add delay into database.
DEFINITION: int lmc1_AddDelayToLib(
Double dDelayMs); //time of delay, ms

DISCRIPTION: call lmc1_AddDelayToLib to add delay into database.
RETURN VALUE: common error code

lmc1_AddWritePortToLib

INTENTION: add Input port into database.
DEFINITION: int lmc1_AddWritePortToLib(
int nOutPutBit //value of Input port, 0-15
BOOL bHigh //enable high or low
BOOL bPluse //enable pulse mode or not
Double dPulseTimeMs); //time of pulse, ms

DISCRIPTION: add a input port into database
RETURN VALUE: common error code

lmc1_AddFileToLib

INTENTION: add the appointed file into database.
DEFINITION：int lmc1_AddFileToLib(
TCHAR* pFileName, //file name

```
TCHAR* pEntName,      // name of the file object
double dPosX,         // X coordinate of left-bottom point
double dPosY,         // Y coordinate of left-bottom point
double dPosZ,         // Z coordinate of the file object
int    nAlign,        // align way 0－8
double dRatio,        // scaling ratio
int nPenNo,           //the pen number of the file object
BOOL bHatchFile       // hatch the file object or not
);
```

DISCRIPTION: call lmc1_AddFileLib to add new file object into database. The following file formats are supported : ezd, dxf， dst, plt, ai, bmp, jpg, tga, png, gif, tiff

RETURN VALUE: common error code

lmc1_AddBarCodeTolib

INTENTION: add a new bar code object into database.

DEFINITION: int lmc1_AddBarCodeToLib(

```
TCHAR* pStr,
TCHAR* pEntName,
double dPosX,
double dPosY,
double dPosZ,
int    nAlign,
int    nPenNo,
int    bHatchText,
int    nBarcodeType,
WORD   wBarCodeAttrib,
double dHeight,
double dNarrowWidth,
double dBarWidthScale[4],
double dSpaceWidthScale[4],
double dMidCharSpaceScale,
double dQuietLeftScale,
double dQuietMidScale,
double dQuietRightScale,
double dQuietTopScale,
double dQuietBottomScale,
int    nRow,
int    nCol,
int    nCheckLevel,
int    nSizeMode,
double dTextHeight,
```

```
        double dTextWidth,
        double dTextOffsetX,
        double dTextOffsetY,
        double dTextSpace,
        TCHAR* pTextFontName
        );

    pStr                //character string of the bar code
    pEntName            //name of the bar code object
    dPosX,              //X coordinate of left-bottom basic point of the barcode
    dPosY               //Y coordinate of left-bottom basic point of the barcode
    dPosZ               //Z coordinate of the bar code object
    nAlign,             //align way 0－8
    nPenNo              //the pen NO. of the barcode object
    bHatchText          //hatch the barcode object or not
    nBarcodeType        //type of barcode, see following:
        #define BARCODETYPE_39              0
        #define BARCODETYPE_93              1
        #define BARCODETYPE_128A            2
        #define BARCODETYPE_128B            3
        #define BARCODETYPE_128C            4
        #define BARCODETYPE_128OPT          5
        #define BARCODETYPE_EAN128A         6
        #define BARCODETYPE_EAN128B         7
        #define BARCODETYPE_EAN128C         8
        #define BARCODETYPE_EAN13           9
        #define BARCODETYPE_EAN8            10
        #define BARCODETYPE_UPCA            11
        #define BARCODETYPE_UPCE            12
        #define BARCODETYPE_25              13
        #define BARCODETYPE_INTER25         14
        #define BARCODETYPE_CODABAR         15
        #define BARCODETYPE_PDF417          16
        #define BARCODETYPE_DATAMTX         17
        #define BARCODETYPE_USERDEF =18,
        #define BARCODETYPE_QRCODE = 19,
        #define BARCODETYPE_MICROQRCODE = 20
    wBarCodeAttrib attribute of barcode.
//const WORD BARCODE_ATT_CHECKNUM   = 0x0004;//check self
//const WORD BARCODE_ATT_REVERSE    = 0x0008;//reverse
//const WORD BARCODE_ATT_SHORTMODE  = 0x0040;// show the character
//const WORD BARCODE_ATT_DOTMODE    = 0x0080;//point mode
//const WORD BARCODE_ATT_CIRCLEMODE = 0x0100;//circle mode
//const WORD BARCODE_ATT_ENABLETILDE  = 0x0200;//DataMatrix //enable
```

//const WORD BARCODE_ATT_RECTMODE = 0x0400;//rectangle mode
//const WORD BARCODE_ATT_SHOWCHECKNUM = 0x0800; // Display check code text
//const WORD BARCODE_ATT_HUMANREAD = 0x1000;// Display character recognition character
//const WORD BARCODE_ATT_NOHATCHTEXT = 0x2000;//did not hatch character
//const WORD BARCODE_ATT_BWREVERSE = 0x4000;//reverse black and white
//const WORD BARCODE_ATT_2DBIDIR = 0x8000;//double array

dHeight //height of bar code
dNarrowWidth: //width of the narrowest module
dBarWidthScale: //ratio of bar width to narrowest module
dSpaceWidthScale: //ratio of space width to the narrowest module
dMidCharSpaceScale //ratio of character space width to the narrowest module
dQuietLeftScale: //ratio of the left blank width to the narrowest module
dQuietMidScale: //ration of the middle blank width to the narrowest module
dQuietRightScale: //ratio of the right blank width to the narrowest module
dQuietTopScale: //ratio of the top blank width to the narrowest module
dQuietBottomScale: //ratio of the bottom blank width to the narrowest module
nRow: //row number of two-dimension barcode
nCol: //column number of two-dimension barcode
nCheckLevel, //pdf417 error recovery level 0-8
nSizeMode, //DataMatrix size mode 0-30

```
#define DATAMTX_SIZEMODE_SMALLEST   0
#define DATAMTX_SIZEMODE_10X10      1
#define DATAMTX_SIZEMODE_12X12      2
#define DATAMTX_SIZEMODE_14X14      3
#define DATAMTX_SIZEMODE_16X16      4
#define DATAMTX_SIZEMODE_18X18      5
#define DATAMTX_SIZEMODE_20X20      6
#define DATAMTX_SIZEMODE_22X22      7
#define DATAMTX_SIZEMODE_24X24      8
#define DATAMTX_SIZEMODE_26X26      9
#define DATAMTX_SIZEMODE_32X32      10
#define DATAMTX_SIZEMODE_36X36      11
#define DATAMTX_SIZEMODE_40X40      12
#define DATAMTX_SIZEMODE_44X44      13
#define DATAMTX_SIZEMODE_48X48      14
#define DATAMTX_SIZEMODE_52X52      15
#define DATAMTX_SIZEMODE_64X64      16
```

```
#define DATAMTX_SIZEMODE_72X72      17
#define DATAMTX_SIZEMODE_80X80      18
#define DATAMTX_SIZEMODE_88X88      19
#define DATAMTX_SIZEMODE_96X96      20
#define DATAMTX_SIZEMODE_104X104    21
#define DATAMTX_SIZEMODE_120X120    22
#define DATAMTX_SIZEMODE_132X132    23
#define DATAMTX_SIZEMODE_144X144    24
#define DATAMTX_SIZEMODE_8X18       25
#define DATAMTX_SIZEMODE_8X32       26
#define DATAMTX_SIZEMODE_12X26      27
#define DATAMTX_SIZEMODE_12X36      28
#define DATAMTX_SIZEMODE_16X36      29
#define DATAMTX_SIZEMODE_16X48      30

#define QRCODE_SIZEMODE_SMALLEST   0
#define QRCODE _SIZEMODE_VERSION1      1
#define QRCODE _SIZEMODE_VERSION2      2
#define QRCODE _SIZEMODE_VERSION3      3
#define QRCODE _SIZEMODE_VERSION4      4
#define QRCODE _SIZEMODE_VERSION5      5
#define QRCODE _SIZEMODE_VERSION6      6
#define QRCODE _SIZEMODE_VERSION7      7
#define QRCODE _SIZEMODE_VERSION8      8
#define QRCODE _SIZEMODE_VERSION9      9
#define QRCODE _SIZEMODE_VERSION10      10
#define QRCODE _SIZEMODE_VERSION11      11
#define QRCODE _SIZEMODE_VERSION12      12
#define QRCODE _SIZEMODE_VERSION13      13
#define QRCODE _SIZEMODE_VERSION14      14
#define QRCODE _SIZEMODE_VERSION15      15
#define QRCODE _SIZEMODE_VERSION16      16
#define QRCODE _SIZEMODE_VERSION17      17
#define QRCODE _SIZEMODE_VERSION18      18
#define QRCODE _SIZEMODE_VERSION19      19
#define QRCODE _SIZEMODE_VERSION20      20
#define QRCODE _SIZEMODE_VERSION21      21
#define QRCODE _SIZEMODE_VERSION22      22
#define QRCODE _SIZEMODE_VERSION23      23
#define QRCODE _SIZEMODE_VERSION24      24
#define QRCODE _SIZEMODE_VERSION25      25
#define QRCODE _SIZEMODE_VERSION26      26
#define QRCODE _SIZEMODE_VERSION27      27
#define QRCODE _SIZEMODE_VERSION28      28
```

```
#define QRCODE _SIZEMODE_VERSION29      29
#define QRCODE _SIZEMODE_VERSION30      30
#define QRCODE _SIZEMODE_VERSION31      31
#define QRCODE _SIZEMODE_VERSION32      32
#define QRCODE _SIZEMODE_VERSION33      33
#define QRCODE _SIZEMODE_VERSION34      34
#define QRCODE _SIZEMODE_VERSION35      35
#define QRCODE _SIZEMODE_VERSION36      36
#define QRCODE _SIZEMODE_VERSION37      37
#define QRCODE _SIZEMODE_VERSION38      38
#define QRCODE _SIZEMODE_VERSION39      39
#define QRCODE _SIZEMODE_VERSION40      40
```

```
    dTextHeight          //height of the character string
    dTextWidth           //width of the character string
    dTextOffsetX         //X offset of the character string
    dTextOffsetY         //Y offset of the character string
    dTextSpace           //space of the character string
    pTextFontName        //font name of the character string
```

DISCRIPTION: call lmc1_AddBarCodeToLib to add bar code object into database.
RETURN VALUE: common error code

lmc1_GetBarcodeParam

INTENTION: Get barcode parameter
DEFINITION: int lmc1_GetBarcodeParam(TCHAR* pEntName,
WORD&wBarCodeAttrib,
int& nSizeMode,
int& nCheckLevel,
int& nLangPage,
double& dDiameter,
int&nPointTimesN,
double& dBiDirOffset);

```
    pEntName          //object name
    wBarCodeAttrib    //barcode attritube
    nSizeMode         //size mode
    nCheckLevel       // Error correction level
    nLangPage         // Language encoding page
    nPointTimesN      //point times
    dBiDirOffset      // Bidirectional scanning compensation
```

DISCRIPTION: Get barcode parameter
RETURN VALUE: common error code

lmc1_SetBarcodeParam

INTENTION: Set barcode parameter

DEFINITION: int lmc1_SetBarcodeParam(TCHAR* pEntName,
WORD&wBarCodeAttrib,
int nSizeMode,
int nCheckLevel,
int nLangPage,
double dDiameter,
int nPointTimesN,
double dBiDirOffset);

pEntName //object name
wBarCodeAttrib //barcode attritube
nSizeMode //size mode
nCheckLevel // Error correction level
nLangPage // Language encoding page
nPointTimesN //point times
dBiDirOffset // Bidirectional scanning compensation

DISCRIPTION: Set barcode parameter

RETURN VALUE: common error code

10. External axis

lmc1_Reset

INTENTION: enable and reset the coordinate of extend axis

DEFINITION: int lmc1_Reset(BOOL bEnAxis0 , BOOL bEnAxis1);

bEnAxis0 //enable extended axis 0 or not
bEnAxis1 //enable extended axis 1 or not

DISCRIPTION: Before calling any other function about extended axis, you must call lmc1_Reset first to enable the appointed axis. When the extended axis moves to the limited position, lmc1_Reset can be called to reset the coordinate.

RETURN VALUE: common error code

lmc1_AxisCorrectOrigin

INTENTION: calibrate the origin of extended axis

DEFINITION: int lmc1_AxisCorrectOrigin(int axis);

axis //axis number 0 = axis 0 1 = axis 1

DISCRIPTION: call lmc1_AxisCorrectOrigin to calibrate the origin of extended axis

automatically

RETURN VALUE: common error code

lmc1_AxisMoveTo

INTENTION: move the extended axis to appointed position.

DEFINITION:int lmc1_AxisMoveTo(int axis,double GoalPos);

axis //axis number 0 = axis 0 1 = axis 1

GoalPos: //absolute coordinate of position

DISCRIPTION: call lmc1_AxisMoveTo to move extended axis to the appointed absolute coordinate position. The moving speed is the biggest speed set in parameter.

RETURN VALUE: common error code

lmc1_AxisMoveToPulse

INTENTION: move the extended axis to appointed pulse position.

DEFINITION:int lmc1_AxisMoveToPulse(int axis, int GoalPos);

axis //axis number 0 = axis 0 1 = axis 1

GoalPos: //absolute coordinate of pulse position

DISCRIPTION: call lmc1_AxisMoveToPulse to move extended axis to the appointed absolute coordinate pulse position. The moving speed is the biggest speed set in parameter.

RETURN VALUE: common error code

lmc1_GetAxisCoor

INTENTION: get the coordinate of extended axis.

DEFINITION: int lmc1_GetAxisCoor(int axis);

axis: //axis number 0 = axis 0 1 = axis 1

DISCRIPTION: call lmc1_GetAxisCoor to get the coordinate of extended axis.

RETURN VALUE: coordinate of appointed extended axis

lmc1_GetAxisCoorPulse

INTENTION: get the pulse of extended axis.

DEFINITION: int lmc1_GetAxisCoorPulse(int axis);

axis: //axis number 0 = axis 0 1 = axis 1

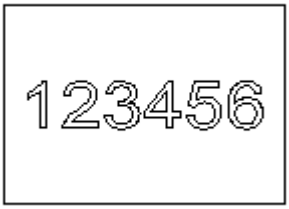
DISCRIPTION: call lmc1_GetAxisCoorPulse to get the pulse of extended axis.

RETURN VALUE: pulse of appointed extended axis

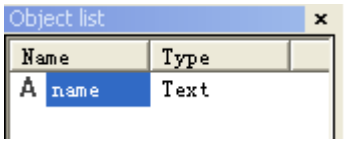
11. **DEVELOPMENT STEPS**

We will show an example about how to do the Development

Client need mark one line text in a rectangle, which is shown as following, and the content of text must be achieved from a network server.



Common step of Development as follows:
Step 1. Create a template file as test.ezd, then create a new text object and named it to “name”. Adjust the size, position and parameters to reach the machining effect. Then save file and exit EzCad2.



Step 2. Program software for calling markezd.dll.

a) First step : Dynamic Load MarkEzd.dll

HINSTANCE hEzdDLL = LoadLibrary(_T("MarkEzd.dll"));

b) Second step: get the pointer of the function to be called

lmc1_Initial=(LMC1_INITIAL)GetProcAddress(hEzdDLL, _T("lmc1_Initial"));
lmc1_Close=(LMC1_CLOSE)GetProcAddress(hEzdDLL, _T("lmc1_Close"));
lmc1_LoadEzdFile=(LMC1_LOADEZDFILE)GetProcAddress(hEzdDLL,_ T("lmc1_LoadEzdFile"));
lmc1_Mark=(LMC1_MARK)GetProcAddress(hEzdDLL,_T("lmc1_Mark"));
lmc1_ChangeTextByName=(LMC1_CHANGETEXTBYNAME)GetProcAdd ress(hEzdDLL,_T("lmc1_ChangeTextByName"));

c) Third step: Call the function

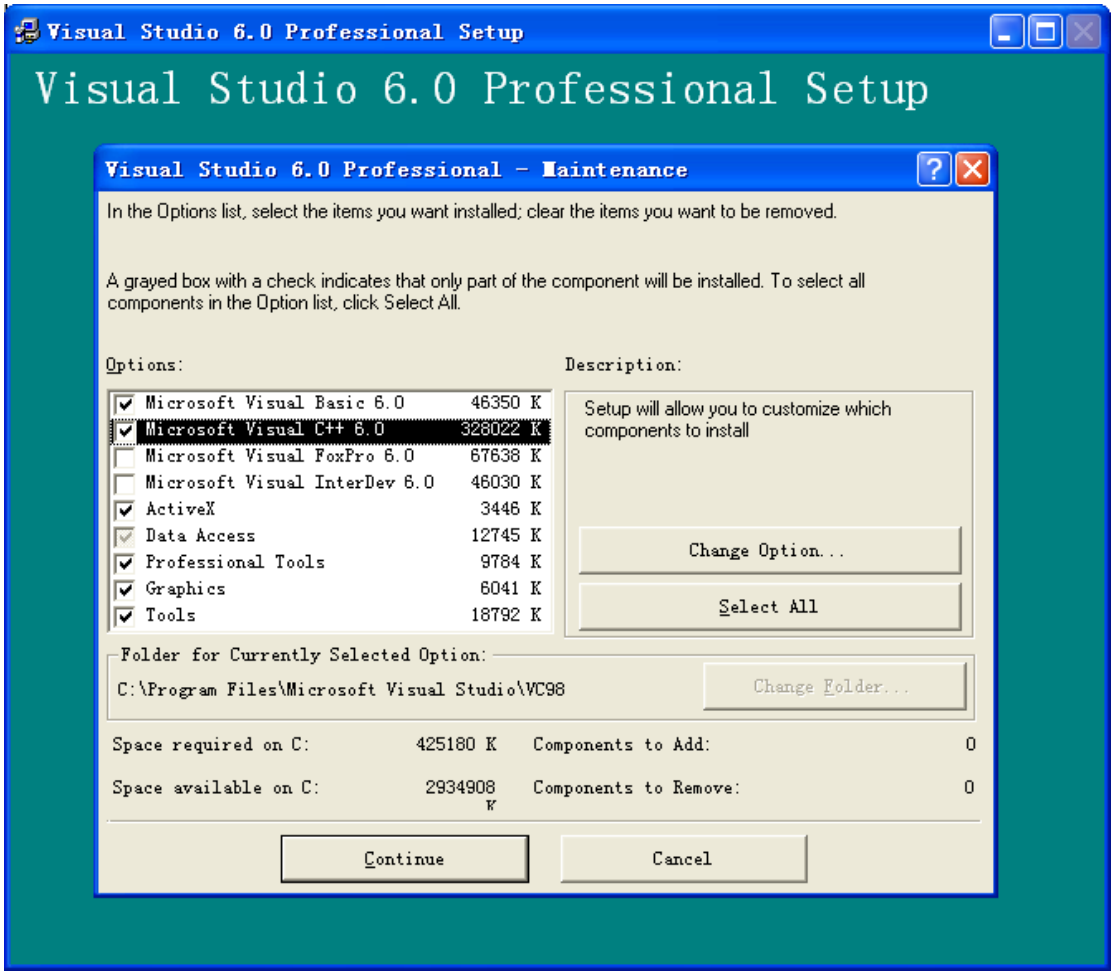
1） Initialization lmc1 board: **lmc1_Initial()**
2） Open test.ezd: **lmc1_LoadEzdFile(_T(“test.ezd”))**.
3） Get the text content from network server. User must write this segment program oneself.

4） Change the content of the named text object.
lmc1_ChangeTextByName (_T(“name”)， szTextAchievedFromNetwork);
5） Call **lmc1_Mark()** for machining
6） If go on, return to 3).
7） Close lmc1 board: **lmc1_Close().**

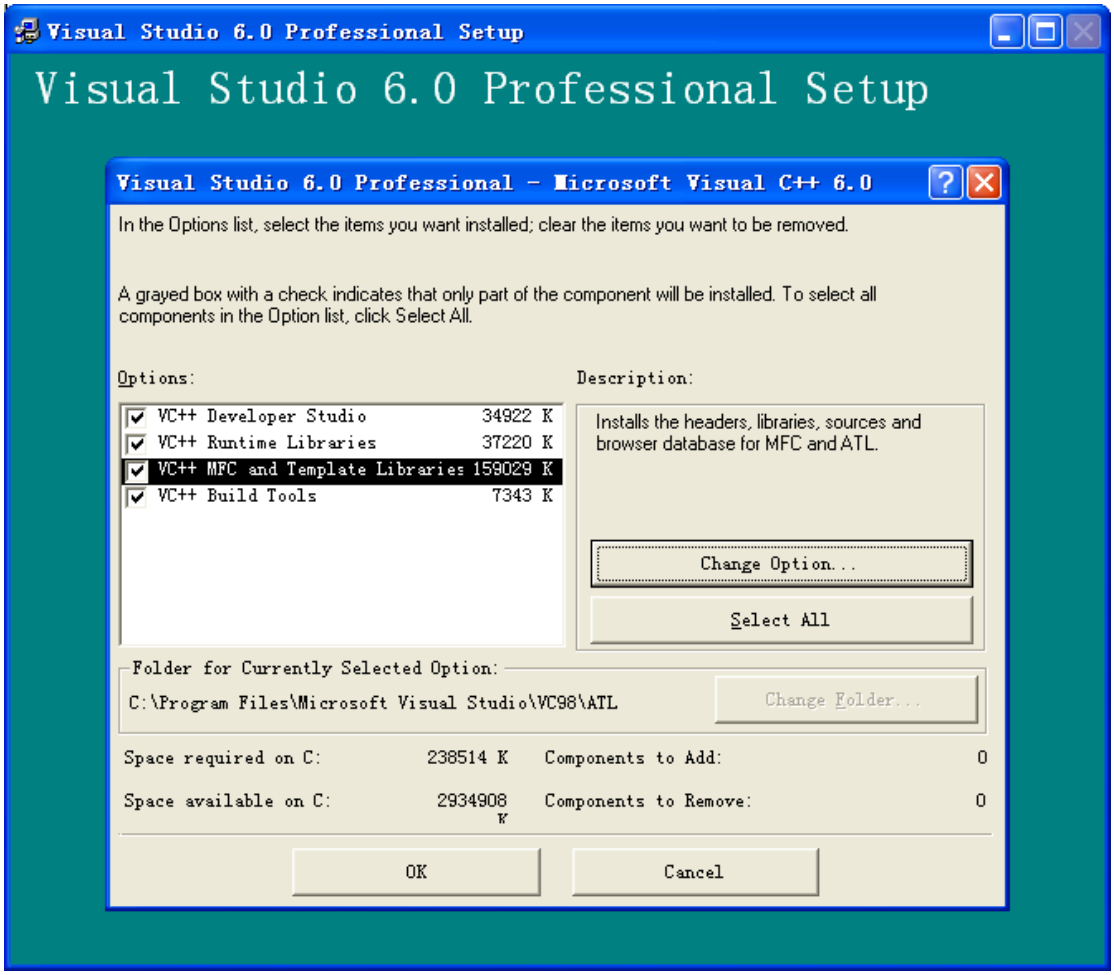
d) Fourth step, Release markezd.dll: **FreeLibrary(hEzdDLL)**

12. Appendix: Set project to Unicode type in VC++

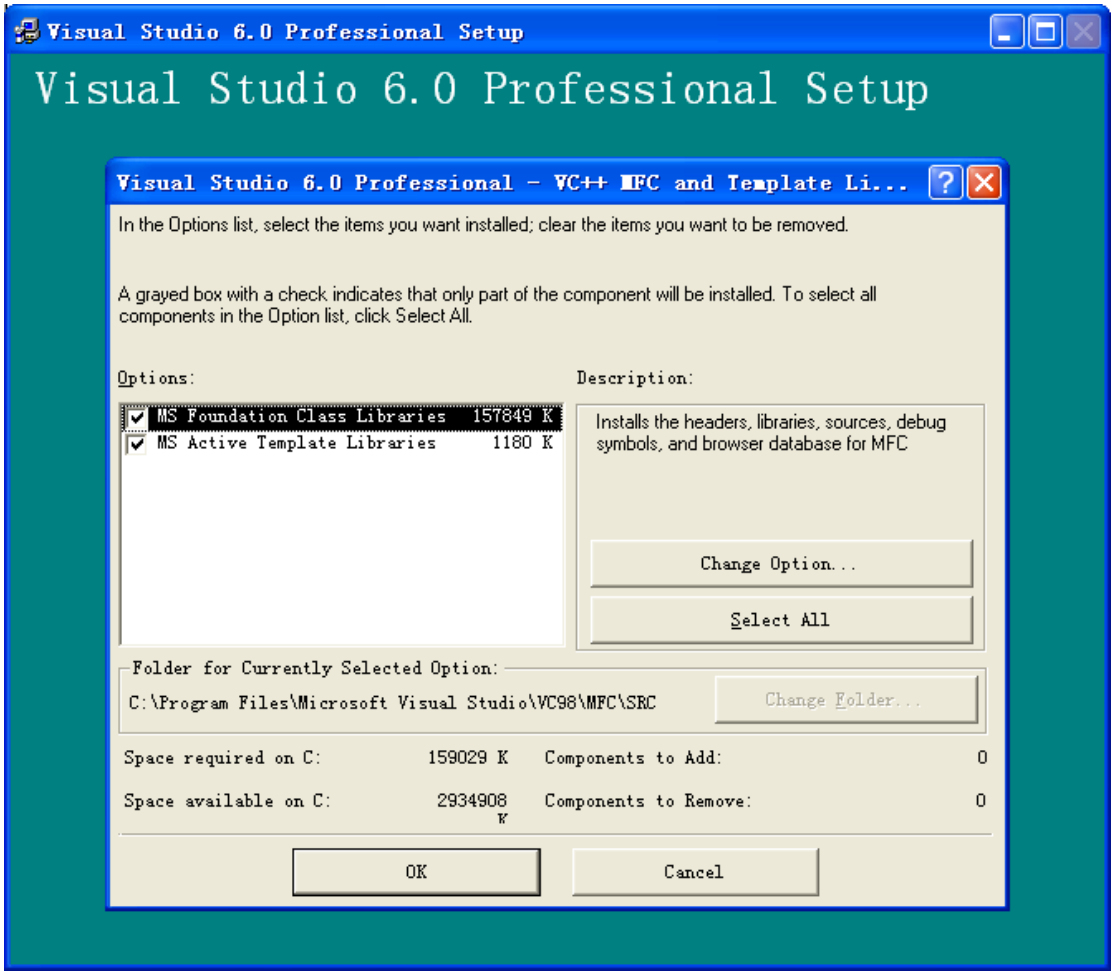
1. Choose “Microsoft Visual C++ 6.0” when install visual studio, and click “Change Option”.



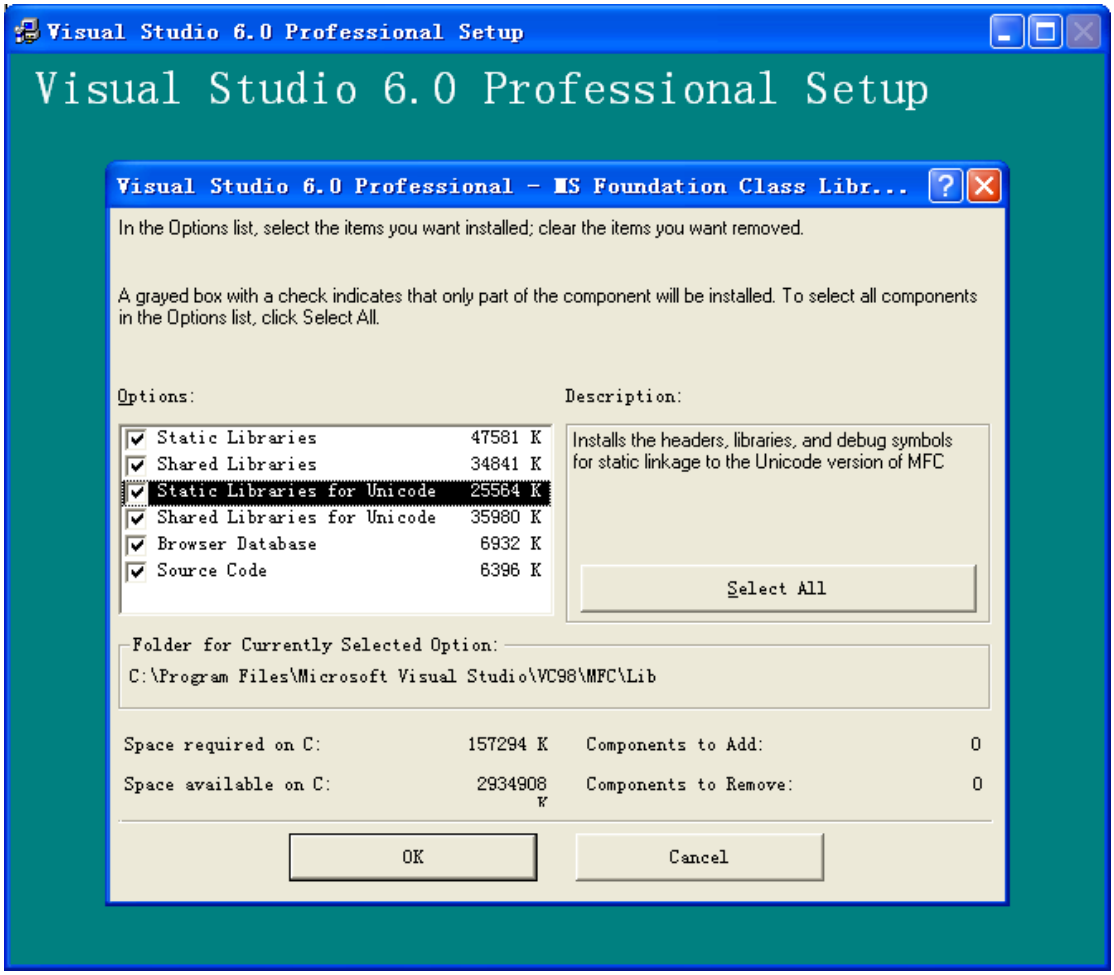
2. Choose “VC++ MFC and Template Libraries” and click “Change Option”.



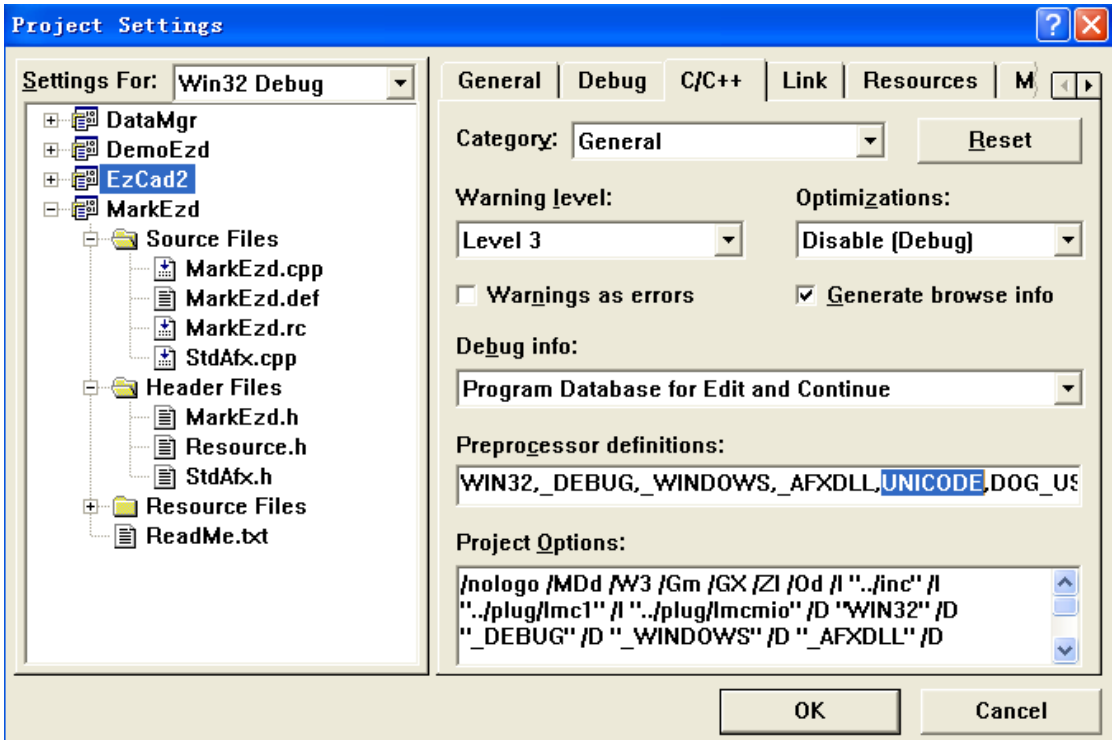
3. Choose “MS Foundation Class Libraries” and click “change option”.



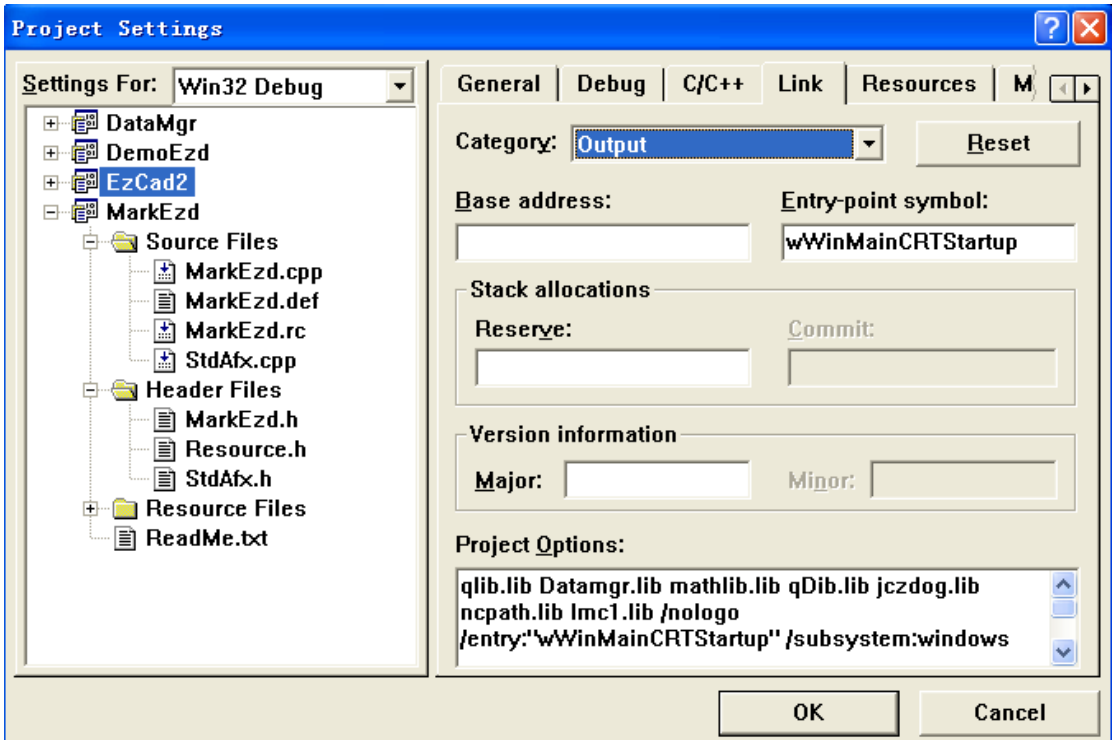
4. Choose the options as following picture, and click “OK”.



5. Open the project, choose menu Project->Settings. Choose “C/C++”, add “UNICODE” and delete “MCBS” in “Preprocessor definitions”



6. Choose “Link”, select “Output” in Category, and add “wWinMainCRTStartup” in “Entry-point symbol”



7. Change all “char” to “TCHAR” in source code.
8. Change all character string included by double quotation marks “…” to _T(“…”)
9. Compile and link the project again.