Data Matrix Encoding SDK

User's Guide

Table of Contents

1. Introduction.

1.1 Scope

This document is applicable to the Data Matrix **Encoding** SDK.

SDK is notated as **DM_SN_YY**, where **YY**=32|64, and notation "32|64" means 32 bit or 64 bit version.

Library interface is build for Windows OS. Only dynamic library is available.

The library is designed to encode data into Data Matrices ECC200 in accordance with ISO/IEC 16022 Symbology specification.

1.2 Normative references

ISO/IEC 16022 - Symbology specification - Data Matrix

1.3 SDK composition

Decoding SDK contains:

- Windows DLL (**EncLib.DLL**) to encode data into Data Matrices.
- Demo-program build in Borland CBuilder v 3.0 that illustrate the DLL usage
- Current User's Guide.

2. EncLib.DLL Interface

The EncLib.DLL contains the following functions:

```
void ClearEncode (void);
        // /Prepares the encoding software for new encoding.
void  SetDefaultEncodeType (int EncodeType);
        // Sets a type of encoding (if necessary).
void  AddToEncode (const char * Source);
        //Adds array of characters with null terminator to the message for encode.
        //It can be called several times.
void AddBufToEncode (const unsigned char * Source, int Length);
        //Adds array of any symbols by length to the message for encode.
        //It can be called several times.
```

int **CheckMatrixSize** (int NM, int &Height, int &Width)
    // Checks or sets the size of Data Matrix symbol in modules (elementary cells).
void **Make_ecc200** (bool [144][144] &Matrix);
    // Gets the matrix of a Data Matrix symbol
    //each 0/1 of its value corresponts to
    // an white or black module of the Data Matrix symbol
int **MapToCanvas** (bool [144][144] & Matrix ,int Height, int Width, HDC Destination, int LeftMargin, int TopMargin , int Step, int Pattern, int Reserved);
    // Draws an Data Matrix symbol on "h_Dest" device context

2.1 ClearEncode

void ClearEncode (void);

Description
This function prepares the encoding software for new encoding.

2.2 SetDafaultEncodeType

void SetDafaultEncodeType (int EncodeType**)**

Description
Sets a type of encoding (if necessary).
Parameters

encAuto	= 0	// Default
encC40	= 230	
encBase256	= 231	
encANSIX12	= 238	
encText	= 239	
encEDIFACT	= 240	
encASCII	= 254	

In case of parameter EncodeType is set to "encAuto", the function calculates minimal possible size of the Data Matrix symbol by inserting special switch latches of encodation in optimal positions of the initiall message

In the all other cases the function encodes all chars under the specified type of encode, if it is possible.

2.3 CheckMatrixSize

int GCheckMatrixSize(int NMatrix **, int** &Height, **int** &Width**)**

Checks or sets depending on value of parameter 1 the size of Data Matrix symbol in modules (elementary cells).

2.4 Parameters

Input Parametrs
NMatrix.
possible first param value for CheckMatrixSize

ms_auto	=	-1 ;
ms_10_10	=	0 ;
ms_12_12	=	1 ;
ms_14_14	=	2 ;
ms_16_16	=	3 ;
ms_18_18	=	4 ;
ms_20_20	=	5 ;
ms_22_22	=	6 ;
ms_24_24	=	7 ;
ms_26_26	=	8 ;
ms_32_32	=	9 ;
ms_36_36	=	10 ;
ms_40_40	=	11 ;
ms_44_44	=	12 ;
ms_48_48	=	13 ;
ms_52_52	=	14 ;
ms_64_64	=	15 ;
ms_72_72	=	16 ;
ms_80_80	=	17 ;
ms_88_88	=	18 ;
ms_96_96	=	19 ;
ms_104_104	=	20 ;
ms_120_120	=	21 ;
ms_132_132	=	22 ;
ms_144_144	=	23 ;
ms_18_8	=	24 ;
ms_32_8	=	25 ;
ms_26_12	=	26 ;
ms_36_12	=	27 ;
ms_36_16	=	28 ;
ms_48_16	=	29 ;

In case of parameter NMatrix is set to "ms_auto", the function calculates minimal possible size of the Data Matrix symbol by inserting special switch latches of encodation in optimal positions of the initiall message

In the all other cases the function checks whether the message could be encoded in the Data Matrix symbol with its size set by parameter NMatrix. (In these cases the parameter NMatrix is an index of possible Data Matrix symbols)

Output parameters:
Height, Width - height and width of Data Matrix symbol in modules (elementary cells)

Return Value
The function returns the following results:

> 0	Everything is OK, result one of the const values ms_10_10 .. ms_144_144 ,..., ms_18_8 .. ms_48_16
-1	Error Too long text for this matrix size
-2	Error Too long text
-3	Error Unresolved character for ANSIX12 encodation
-4	Error Unresolved character for Edifact encodation

2.5 AddToEncode

void AddToEncode (char * Source **)**

Input Parametrs
Adds array of characters with null terminator to the message for encode. It can be called several times..

2.6 AddBufToEncode

void AddBufToEncode (unsigned char * Source, **int** Length **)**

Input Parametrs
Adds array of bytes to the message for encode. It can be called several times.

2.7 Make_ecc200

typedef bool TMatrix[144][144];
void Make_ecc200 (TMatrix **&);**

Description

This function fills the boolean 2D-array for a custom decoding. This array includes the alignment and finder patterns of Data Matrix.

Parameters

MappingMatrix[144][144]

The 2D boolean array is supposed to be allocated before the function call. After the function implementation a **true** value in *MappingMatrix[row][col]* denotes the foreground module of Data Matrix. **False** value corresponds to background module.

f some rectangle like a Data Matrix is found.

2.8 MapToCanvas

```
void MapToCanvas  (
                   TMatrix &Matrix,
                  int Height, int Width,
                  HDC Destination,
                  int Left, int Top,
                  int Step,
                  int Pattern,
                  int Reserved);
```

An auxiliary function
Draws an Data Matrix symbol on "h_Dest" device context

Input parameters

Matrix	- matrix of DataMatrix symbol (look Make_ecc200)
h_Dest	- Handle of an device context
Left	- left Posizion of the Data Matrix symbols (in pixels)
Top	- top Position of the Data Matrix symbols (in pixels)
Step	- step between neighbouring modules of Data Matrix symbol(in pixels). Can be more or less than size of the modules.
Pattern	- size of a module (one side of square module) of the Data Matrix symbol(in pixels)
Reserved	- must be zero

2.9 Loading and calls

Call the API function **LoadLibrary***(lpLibFileName)* to load the DLL handle into your application program. It gives you access to DLL interface procedures.

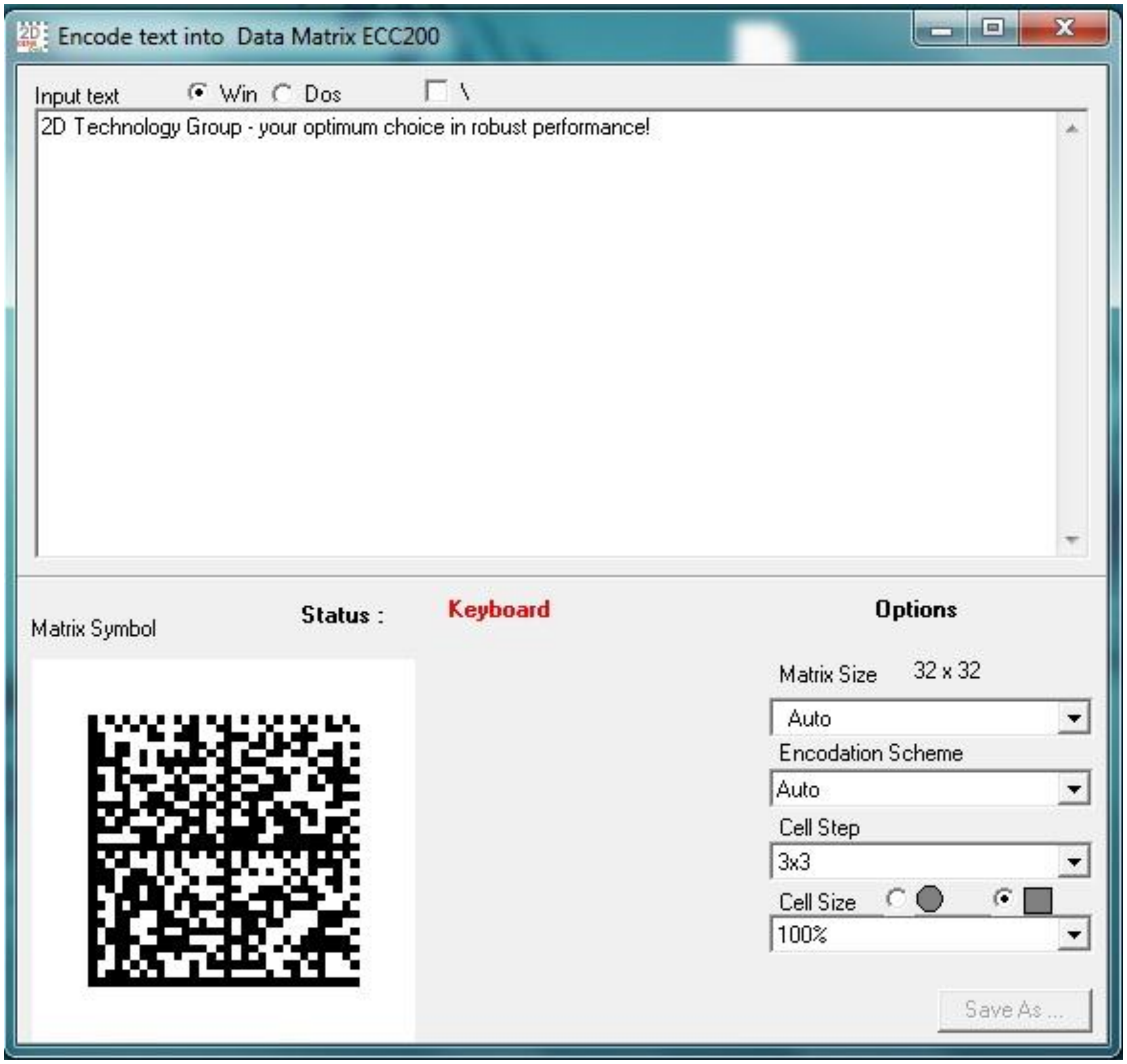
Call API function **FreeLibrary** to release the system resources before the application is finished.

3. Demo application

Decoding Library comes with the Demo application build in Borland CBuilder v 3.0.

3.1 GUI

GUI of this application illustrates all the major features of the Encoding Library:



2D Technology Group - your optimum choice in robust performance!

Matrix Size 32 x 32
Auto
Encodation Scheme
Auto
Cell Step
3x3
Cell Size
100%

- **Input text**:
  - **Win** – Windows code pages
  - **DOS** - DOS code pages
- **Options**
  - **Matrix Size**

Allows user to specify the matrix size (drop-down menu). Default value "**Auto**" (Recommended) means that the program will choose the smallest size that accommodates the data.

- **Encodation Scheme**
  The data may be encoded using any combination of six encodation schemes (drop-down menu) provided for by the ISO/IEC 16022. Default value "**Auto**" means that the program will choose the best scheme (having the highest degree of compaction) for a given set of data.
- **Cell Step** – Module size / distance between the centers of the modules (pixels)
- **Cell Size** – Module fill level (%)

3.2 Example

This is an example of DLL usage in C++ (Borland CBuilder v3.0) program

```
#include "EncLib.H"

int EncLibResult;
bool Matrix[144][144];

TencLib EncLib;

void EncodeText(const char * Txt)
{ char * Err;


  EncLib -> ClearEncode();

  EncLib -> SetDefaultEncodeType(encAuto);

  EncLib -> AddToEncode (" Begin Text->");
  EncLib -> AddToEncode (Txt);
  EncLib -> AddToEncode ("<- End Text");

  EncLibResult= EncLib -> CheckMatrixSize(ms_auto,Height,Width);

  Err = "Ok";

  if (EncLibResult>=0)
    { Err = "Ok";
      EncLib -> Make_ecc200(Matrix);
      EncLib ->MapToCanvas (Matrix, Height,Width, Image1.Canvas.Handle,
20,200,6,5,0);
    }
  else
  if (EncLibResult == -1)
    {Err ="Too long text for this matrix size";}
  else
  if (EncLibResult == -2)
```

```
      {Err ="Too long text";}
   else
   if (EncLibResult == -3)
      {Err ="Unresolved character  for ANSIX12";}
   else
   if (EncLibResult == -4)
      {Err ="Unresolved character  for Edifact";}
   else
      {Err ="Unknown Error ";}

   Label1.Caption = Err;

}
```