User's Guide

Table of Contents

1. Introduction.

1.1 Scope

The library is designed to decode PDF417 Codes in accordance with ISO/IEC 16022 Symbology specification. Symbol quality assessment is provided in accordance with ISO/IEC 15415 standard.

Library interface is the same for Windows, Linux, and certain embedded platforms. Both static and dynamic libraries are available.

The Quality Parameters option is user-selectable.

Library processes **8-bit** images only.

1.2 Normative references

ISO/IEC 15438:2006 - Symbology specification – PDF417
ISO/IEC 15415 - Symbol quality - Two-dimensional symbols

1.3 SDK package composition

Decoding SDK package contains:

- Linux DLL/SO designed to perform PDF417 Code search, recognition and decoding.
- C++ source code of the Demo program that illustrate the library usage.
- Current User's Guide.

1.4 Features Description

The Library features are described in the Table below:

PDF417 SDK	
Features	**Features Description**
Direct	allows to decrease image capture time (if barcode orientation is known) by providing for three optional settings – "Left->Right", "Horizontal", "Vertical", "All Direction"
Speed	speed control setting with 3 optional modes: • **Fast Mode** - High speed decoding (~ 2x faster than Robust Mode) - for applications where decoding time is critical and image quality is relatively good; • **Robust Mode** - lower speed but highest accuracy level – for poor quality images; • **Smart Mode** - almost as good as **Robust** Mode in terms of decode rate, but still fast enough
Symbol Quality	Quality Parameters evaluation in accordance with ISO 15415, 15416

Multi Mode	decodes up to 100 barcodes in one image via variable setting
BW Filter	reduces or enlarges the bar widths with respect to spaces (print gain correction)

All listed features are user-selectable.

1.5 Program session

Typical program session looks as follows:

Step 1. Connect decoder
Step 2. Create and set decoder options
Loop
    Step 3. Capture/read bitmap image
    Step 4. Process image
    Step 5. Request image and symbols info
    … // further application-specific data processing and interaction with user
End Loop
Step 6. Delete decoder options
Step 7. Disconnect decoder.

2. The Basic Interface Structures

The library includes the following structures:

struct TPDF417_OptMode	The set of decoder options
struct TPDF417_ImageInfo	Features of decoded image
struct TPDF417_Info	Features of decoded symbols
struct TPDF417_Quality	Quality Parameters of decoded symbols

2.1 Decoder options.

```
struct TPDF417_OptMode
{
  int MaxSymbolCount;   //
  int SpeedMode;        // SP_Robust, SP_Fast, …
  int CalcGrade;        //
  int DirectMode;       // DM_LeftRight, DM_Horizontal, DM_All
  int BWFilter;         // 0: none, +1 => BW+1,  -1 => BW-1
```

```
  int TimeOut;          // milliseconds, zero - not applied
  int ScanStep;         // step of scan lines
};
```

2.2 Image info.

```
struct TPDF417_ImageInfo
{ // averall image properties
    int SymbolCount; // number of well decoded PDF417 symbols within image
    int ErrorCode;   // nonzero, if no one symbols had been decoded
    char* ErrorString; // pointer to error string
};
```

2.3 Symbol info.

Each decoded symbol is described by the following structure:

```
struct TPDF417_Info
{ //result of decoding per each PDF417 symbol

  int ErrorCode;     // Result
  int rowcols[8];    // symbol corners
  bool rcFlag;       // true if a corners of symbol was found
  int RSErrorCount;  // number of Reed Solomon errors
  int ColCount;      // horizontal dimension (from 1 to 30)
  int RowCount;      // vertical dimension   (from 4 to 90)
  char* ErrorString; // pointer to error string
  int ECLevel;       // Error Correction Level (from 0 to 8)
  int pchlen;        // length of decoded byte array
  unsigned char* pch;// decoded byte array
  char* c_str;       // decoded char array (points to the same allocation in
memory)
  TPDF417_Quality Quality; // parameters of symbol quality (if possible)

};
```

2.4 Symbol Quality

```
struct TDM_Quality
{
  // 0..100                    , -1 = have not been calculated
  float symbol_contrast;           // symbol contrast(SC)  = Rmax-Rmin
  float min_reflectance;           // Rmin
  float max_reflectance;           // Rmax
  float global_threshold;          // global_threshold(GT) = (Rmax+Rmin)/2
  float min_edge_contrast;         // ECmin
  float modulation;                // modulation (MOD)     = ECmin/SC
  float defects;                   //
  float decodability;              // V
  float width_height_proportion;
```

```
  float unused_error_correction;    // UEC

  // grades
  // 0..4,  0 = Worst , 4 = Best, -1 = have not been calculated
  float decode_grade;
  float symbol_contrast_grade;
  float min_reflectance_grade;
  float min_edge_contrast_grade;
  float modulation_grade;
  float defects_grade;
  float decodability_grade;
  float unused_error_correction_grade;

  float overall_grade;

};
```

2.5 The Constants.

```
// TPDF417_SpeedMode
const int
  SP_Fast   = 1,
  SP_Robust = 0,
  SP_Smart  = 2;

// TPDF417_DirectMode
const int
  DM_LeftRight  = 0, // Scan only horizontal barcodes with Start Pattern
                     // on left side of image
  DM_Horizontal = 1, // Scan only horizontal barcodes with Start Pattern
                     // on left or right side of image
  DM_All        = 2; // Scan barcode with any orientation
// TPDF417_CalcGradeMode
const int
  CG_No         = 0, //
  CG_Yes        = 1; // Calculate Bar code print quality
                     // See ISO/IEC 15415, ISO/IEC 15416


const int// error codes

//
##############################################################################
  PDF417_OK                    =   0,
  PDF417_Rectangle_Error       =  -1, // no Start and Stop Patterns were found
  PDF417_ColCount_Error        =  -4,
  PDF417_RowCount_Error        =  -5,
  PDF417_ECCodeword_Error      =  -6,
  PDF417_ErasuresLimit_Error   =  -7, // too many codewords with improper
clasters
  PDF417_RS_Error              =  -8, // too many errors in Reed Solomon code
  PDF417_CharEncodation_Error  =  -9, // unexpected encodation scheme
  PDF417_StartPattern_Error    = -10, // Stop pattern is not found
  PDF417_StopPattern_Error     = -11, // Stop pattern is not found
```

```
  PDF417_Indikators_Error     = -12, // incorrect data in row indicators
  PDF417_Geometry_Error       = -13; // unallowed module size

const int// error codes
  PDF417_NOMEMORY             = -99,
  PDF417_UNKNOWN              = -100,
  PDF417_DISCONNECTED         = -200;
//=======================================
```

3. The Interface Procedures and Functions - C++

3.1 Connect_PDF417_Decoder

PPDF417_Decoder Connect_PDF417_Decoder(int maxcolcount, int maxrowcount)

Description.

Function generates new instance of class encapsulating the decoder functionality.

Parameters.

Maximum values of vertical and horizontal image sizes.

Return value.

Pointer to structure TPDF417_Decoder if decoder is created, **null** otherwise

3.2 Disconnect_PDF417_Decoder

void Disconnect_PDF417_Decoder(PPDF417_Decoder& pdecoder)

Description.

Procedure destroys decoder class and frees memory.

Parameters.

pdecoder - the variable containing the pointer on structure TPDF417_Decoder.

3.3 Create_PDF417_Options

PPDF417_Options PDF417_Options_Create(PPDF417_Decoder pdecoder, TPDF417_OptMode optmode)

Description.

Function generates new instance of class encapsulating the decoder with desirable options.

Parameters.

The pointer on structure TPDF417_Decoder.

3.4 Delete_PDF417_Options

void PDF417_Options_Delete(PPDF417_Options& `poptions`)

Description.
Procedure destroys instance of the decoder with options.
Parameters.
The variable containing the pointer on structure TPDF417_Options.

3.5 DecodePDF417_Bits

```
int DecodeBitsF              // base decoding function
               (PPDF417_Options poptions
               ,int rowcount // The number of rows in the image
               ,int colcount // The number of columns
               ,TRow* ppbits // Points to array of pointers:
                             //{pbits[0],...,pbits[rowcount-1]} to bitmap
                  lines
               );
```

Description.
The function processes an image and fills ImageInfo and array of SymbolInfo's.

Parameters.
`poptions` – pointer on structure TPDF417_Options
rowcount – number of image rows,
colcount – number of image columns,
ppbits – array of pointers to image rows. Every row is a byte array with pixels brightness.
(We have **typedef unsigned char* TRow**;)

Return value.
<0, if no one symbol was decoded, >=0 otherwise.

3.6 GetPDF417_ImageInfo

TPDF417_ImageInfo* Get_PDF417_ImageInfo(PPDF417_Options poptions); // inspects image info after decoding

Parameters.
poptions – pointer on structure TPDF417_Options
Return value.
Pointer to Image Info.

3.7 GetPDF417_Info

TPDF417_Info* Get_PDF417_Info(PPDF417_Options poptions, int dmNum); // inspects symbol info after decoding

Parameter.

poptions – pointer on structure TPDF417_Options.

dmNum - number of decoded symbol in image.

If no symbols were decoded then dmNum = 0 extracts Info for most possible symbol location.

Return value.

Pointer to Symbol Info.