IDePass SDK INTRODUCTION

A8Capture SDK Document

Contents

Preface

Need copy the whole directory "./lib" to your application, and don't change the directory structure, then you can call the main dll of "A8Capture.dll".

The following definitions also can get from "PubDefines.h" and "A8Capture.h"

1、 Supported card types

ID	Description	Remark
0	Default	Often used to capture original image
1	Chinese second ID	A8
2	Hong Kong & Macao residence permit on the mainland	A8, living on the mainland
3	Foreigner residence permit on the mainland	A8, living on the mainland
4	Hong Kong and Macao home visit permit	A8, go to mainland
5	Taiwan home visit permit	A8, go to mainland
6	Hong Kong and Macao pass permit	A8, go to Hong Kong or Macao
7	International ID	A8, with three line MRZ
8	Chinese passport(not for IDE200)	A6, can recognize chinese
9	International passport(not for IDE200)	A6,

2、 Supported barcode types

ID	Description	Remark
0x2	AZTEC	
0x4	Codebar	
0x8	Code 39	
0x10	Code 93	
0x20	Code 128	
0x40	Data Matrix	
0x80	EAN 8	
0x100	EAN 13	
0x200	ITF	
0x400	Maxicode	
0x800	PDF 417	
0x1000	QR	Q
0x2000	RSS 14	
0x4000	RSS Expanded	
0x8000	UPC A	
0x10000	UPC E	
0x20000	UPC EAN Extension	

3、 OCR contents

```
typedef struct __CardDetails {
        char cnName[32];                // Chinese name
        char cnSurname[32];             // Chinese surname
        char cnGivenname[32];           // Chinese given name
        char enName[120];               // English name
        char enSurname[120];            // English surname
        char enGivenname[120];          // English given name
        char cnGender[4];               // Chinese gender
        char enGender[4];               // English gender
        char nation[22];                // Nation
        char dateOfBirth[22];           // Date of birth
        char address[72];               // Address
        char identityNumber[36];        // Identity number
        char cardNumber[36];            // Card number
        char cnAuthority[32];           // Authority of issue
        char enAuthority[64];           // Authority of issue
        char dateOfIssue[22];           // Date of issue
        char dateOfExpiry[22];          // Date of expiry
        char dateOfDepart[22];          // Date of depart
        char timesOfIssue[16];          // Times of issure
        char types[36];                 // card types
        char cnPlaceOfBirth[36];        // Place of birth
        char enPlaceOfBirth[72];        // Place of birth
        char cnPlaceOfIssue[36];        // Place of issue
        char enPlaceOfIssue[72];        // Place of issue
        char career[32];                 // Career
        char cardVersion[6];             // Card version
        char firstMRZ[46];               // The first line MRZ
        char secondMRZ[46];                // The second line MRZ
        char thirdMRZ[46];                // The third line MRZ
        unsigned char image[1024];    // Portrait of Chinese second id
        unsigned char finger[1024]; // Fingerprint of Chinese second id
}CardDetails;
```

4、 Barcode decode contents

```
typedef struct _BarcodeResult {
        char data[4096];    // decoded result
        int    dataLen;         // result length
        int    type;              // barcode type
        int    x;              //barcode position in the image
        int    y;              //barcode position in the image
        int    w;               //barcode width
        int    h;              //barcode height
}BarcodeResult;
```

5、 Return value define

ID	Description	Remark
0	success	
-100	device offline	
-101	device open fail	
-102	read file error	
-103	write file error	
-104	load 3rd dll error	
-105	load 3rd funcitons error	
-106	write register error	
-107	read register error	
-108	init asic error	
-109	not find black retangle wile calibrating	
-110	write calibrate data error	
-111	read calibrate data error,may had not cablirated	
-112	calibrate fail	
-113	image format error	

-114	open ID reader error	
-115	ID authenticate error	
-116	read ID contents error	
-117	not support functions	
-118	parameters error	
-119	device is busying	
-120	OCR init failure	
-121	OCR parameters error	
-122	OCR not decteced ROI	
-123	OCR fail	
-124	not support card type	
-125	no barcode be detected	
-126	illegal argument while decode barcode	
-127	barcode decode failed	
-128	barcode decode result verifiy failed	
-129	no authorization,please connect the vendor	

6、 Functions

```
/*
 @brief Open the device and initialize all resources.
 @param modules[in] is external device type.
             1 is only A8 Capture
              2 can support OCR
              4 can support barcode
              8 is china's second generation card reader
@param hWnd[in] is the handle of a window or control, will show
           camera preview while is not null(reserved).
@return 0 is succes, else is failed.
 */
 int IO_OpenDevice(int modules, void* hWnd);

/*
 @brief Close the device and release all resources.
 @param void
 @return void
 */
 void IO_CloseDevice();

/*
@brief calibrate the device, should call the function while the device used on the host for the first
time.
```

```
 @param void
 @return 0 is succes, else is failed.
 */
 int IO_Calibrate();

/*
@brief Capture image buffer from device. If the parameter of a certain light is empty, this image is
not generated.
@param ir[out] is infrared image, need to allocate enough memory in advance, NULL is not
capture.
 @param irw[out] is infrared image width.
 @param irh[out] is infrared image height.
 @param irbits[out] is infrared image bits per pixel(8、24).
 @param wh[out] is color image, need to allocate enough memory in advance, NULL is not
   capture.
 @param whw[out] is color image width.
 @param whh[out] is color image height.
 @param whbits[out] is color image bits per pixel(8、24).
 @param uv[out] is ultraviolet image, need to allocate enough memory in advance, NULL is not
   capture.
 @param uvw[out] is ultraviolet image width.
 @param uvh[out] is ultraviolet image height.
 @param uvbits[out] is ultraviolet image bits per pixel(8、24).
 @param cardtype[in] is card type
 @return 0 is succes, else is failed.
 */
 int IO_CaptureBuf(unsigned char *ir,int *irw,int *irh,int *irbits,
                   unsigned char *wh, int *whw, int *whh, int *whbits,
                    unsigned char *uv, int *uvw, int *uvh, int *uvbits,
                   int cardtype);

/*
@brief Capture image files from device. If the parameter of a certain light is empty,this image is
not generated.
@param irFileName[in] is infrared image file name, will not generate if the filename is NULL.
@param whFileName[in] is color image file name, will not generate if the filename is NULL.
@param uvFileName[in] is ultraviolet image file name, will not generate if the filename is NULL.
 @param cardtype[in] is card type
 @return 0 is succes, else is failed.
 */
 int IO_CaptureFile(const char *irFileName,
                    const char *whFileName,
                    const char *uvFileName,
                    int cardtype);
```

```
/*
@brief Capture the full image files without cut from device. If the parameter of a certain light is
empty,this image is not generated.
@param irFileName[in] is infrared image file name, will not generate if the filename is NULL.
@param whFileName[in] is color image file name, will not generate if the filename is NULL.
@param uvFileName[in] is ultraviolet image file name, will not generate if the filename is NULL.
 @return 0 is succes, else is failed.
 */
 int IO_CaptureFileFullImage(const char *irFileName,
                             const char *whFileName,
                             const char *uvFileName);

/*
 @brief save image as a file.
 @param img[in] is image buffer.
 @param w[in] is image width.
 @param h[in] is image height.
 @param bits[in] is image bits per pixel(8、24).
 @param quality[in] is image quality while save as jpeg
 @return 0 is succes, else is failed.
 */
int   IO_SaveFile(const char *fileName,unsigned char *img, int w, int h, int bits,int quality);

 /*
 @brief load image buffer form file.
 @param fileName[in] the image file path and name.
@param img[out] is image buffer, after used,the buffer need call IO_BuffFree to release.
 @param w[out] is image width.
 @param h[out] is image height.
 @param bits[out] is image bits per pixel(8、24).
 @return 0 is succes, else is failed.
 */
int IO_LoadFile(const char *fileName, unsigned char **img, int   *w, int *h, int *bits);

 /*
 @brief release image buffer.
 @param img[in] is image buffer, this buffer is generated by IO_LoadFile.
 @return void.
 */
 void IO_BufFree(unsigned char *img);

/*
 @brief beep duration.
```

```
@param delay[in] is duration,unit is millisecond.
@return 0 is succes, else is failed.
*/
int IO_Beep(int delay);

/*
@brief led light.
@param mode[in] is Lighting color,
            0 is off. 1 is green on,2 is red on,3 is all on.
@return 0 is succes, else is failed.
*/
int IO_SetLed(int mode);

/*
@brief camera sensor light.
@param mode[in] camera sensor light, 0 is off.
            1 is ir,2 is color(white),3 is uv.
@return 0 is succes, else is failed.
*/
int IO_SetSensorLight(int mode);

/*
@brief check card on the device or not.
@param status[out]    0 is unknown status, 1 is the card is not on the device, 2 is the card on the
device
@return 0 is succes, else is failed.
*/
int IO_GetCardStatus(int *status);

/*
@brief get the device pid and vid.
@param vid[out] the device vid
@param pid[out] the device pid
@return 0 is succes, else is failed.
*/
int IO_GetVidPid(unsigned short *vid,unsigned short *pid);

/*
@brief get the device serial number
@param sn[out] the serial number,should malloc buffer in advance
@param len[out] the serial number string length.
@return 0 is succes, else is failed.
*/
int IO_GetSN(char *sn,int *len);
```

```
/*
@brief get the device firmware version
@param version[out] the firmware version,should malloc buffer in advance
@param len[out] the firmware version string length.
@return 0 is succes, else is failed.
*/
int IO_GetFirmwareVersion(char *version, int *len);

/*
@brief reboot the device
@param void
@return 0 is succes, else is failed.
*/
int IO_RebootDevice();

/*
@brief read the china second ID contents;
@param imgFileName[in] the head portrait file path and name.
@param cardDetails[out] the second ID contents.
@return 0 is succes, else is failed.
*/
int IO_ReadID(const char *imgFileName, CardDetails *cardDetails);

/*
@brief get the second ID reader SAM ID
@param samID[out] the SAM ID,should malloc buffer in advance
@param len[out] the SAM ID string length.
@return 0 is succes, else is failed.
*/
int IO_GetSAMID(char *samID, int *len);

/*
@brief recognize image buffer(OCR)
@param img[in] the image buffer.
@param w[in] the image width.
@param h[in] the image height.
@param bits[in] the bits of per pixel(8 or 24).
@param cardType[in] the card type.
@param cardDetails[out] the contents of recognized.
@return 0 is succes, else is failed.
*/
int IO_RecognizeBuf(unsigned char *img,int w,int h,int bits,
                        int cardType, CardDetails    *cardDetails);
```

```
 /*
 @brief recognize image file(OCR)
 @param imgFileName[in] the image file.
 @param cardType[in] the card type.
 @param cardDetails[out] the contents of recognized.
 @return 0 is succes, else is failed.
 */
int IO_RecognizeFile(const char *imgFileName, int cardType,
                         CardDetails *cardDetails);

 /*
 @brief capture image file and recognize(OCR).
@param imgFileNameIr[in] is infrared image file name, will not generate if the filename is NULL.
@param imgFileNameWh[in] is color image file name, will not generate if the filename is NULL.
@param imgFileNameUv[in] is ultraviolet image file name, will not generate if the filename is
NULL.
 @param cardType[in] the card type.
 @param cardDetails[out] the contents of recognized.
 @return 0 is succes, else is failed.
 */
 int IO_CaptureAndRecognizeFile(const char *imgFileNameIr,
                                    const char *imgFileNameWh,
                                    const char *imgFileNameUv,
                                    int cardType,
                                    CardDetails *cardDetails);

/*
 @brief decode image buffer(Barcode)
 @param img[in] the image buffer.
 @param w[in] the image width.
 @param h[in] the image height.
 @param bits[in] the bits of per pixel(8 or 24).
 @param barType[in] the barcode type.
 @param muti[in] false is only decode one barcode; true can decode two barcodes
 @param result[out] the barcode contents array,need malloc buffer in advance.
 @param resultSize[out] the barcode contents array size.
 @return 0 is succes, else is failed.
 */
 int IO_DecodeBuf(unsigned char *img, int w, int h,int bits,
                      unsigned int    barType, bool muti,
                      BarcodeResult *result, int *resultSize);

 /*
```

```
@brief decode image file(Barcode)
@param imgFileName[in] the image file name.
@param barType[in] the barcode type.
@param muti[in] false is only decode one barcode; true can decode two barcodes
@param result[out] the barcode contents array,need malloc buffer in advance.
@param resultSize[out] the barcode contents array size.
@return 0 is succes, else is failed.
*/
int IO_DecodeFile(const char *imgFileName,
                  unsigned int    barType, bool muti,
                  BarcodeResult *result, int *resultSize);

/*
@brief decode image file(Barcode)
@param imgFileNameIr[in] is infrared image file name, will not generate if the filename is NULL.
@param imgFileNameWh[in] is color image file name, will not generate if the filename is NULL.
@param imgFileNameUv[in] is ultraviolet image file name, will not generate if the filename is
NULL.
@param barType[in] the barcode type.
@param muti[in] false is only decode one barcode; true can decode two barcodes
@param result[out] the barcode contents array,need malloc buffer in advance.
@param resultSize[out] the barcode contents array size.
@return 0 is succes, else is failed.
*/
int IO_CaptureAndDecodeFile(const char *imgFileNameIr,
                            const char *imgFileNameWh,
                            const char *imgFileNameUv,
                            unsigned int    barType, bool muti,
                            BarcodeResult *result, int *resultSize);


/*
@brief Open the device and initialize all resources.
@param modules[in] is external device type.
            1 is only A8 Capture
            2 can support OCR
            4 can support barcode
            8 is china's second generation card reader
@param hWnd[in] is the handle of a window or control, will show
        camera preview while is not null.
@param callback[out] the callback function.
@return 0 is succes, else is failed.
*/
int IO_OpenDeviceCB(int modules, void* hWnd, CallbackResult callback);
```

```
/*
@brief Close the device and release all resources.
@param callback[out] the callback function.
@return void
*/
int IO_CloseDeviceCB(CallbackResult callback);

/*
@brief calibrate the device, should call the function while the device used on the host of the first time.
@param callback[out] the callback function.
@return 0 is succes, else is failed.
*/
int IO_CalibrateCB(CallbackResult callback);

/*
 @brief Capture image buffer from device. If the parameter of a certain light is empty,this image
 is not generated.
@param ir[out] is infrared image, need to allocate enough memory in advance.
@param wh[out] is color image, need to allocate enough memory in advance.
@param uv[out] is ultraviolet image, need to allocate enough memory in advance.
@param callback[out] the callback function.
@return 0 is succes, else is failed.
*/
int IO_CaptureBufCB(unsigned char *ir, unsigned char *wh,
                    unsigned char *uv, CallbackCaptureBuf callback);

/*
@brief Capture image files from device. If the parameter of a certain light is empty,this image is not generated.
@param irFileName[in] is infrared image file name, will not generate if the filename is NULL.
@param whFileName[in] is color image file name, will not generate if the filename is NULL.
@param uvFileName[in] is ultraviolet image file name, will not generate if the filename is NULL.
@param callback[out] the callback function.
@return 0 is succes, else is failed.
*/
int IO_CaptureFileCB(const char *irFileName, const char *whFileName,
                     const char *uvFileName, CallbackCaptureFile callback);

/*
@brief read the china second ID contents;
@param imgFileName[in] the head portrait file path and name.
@param cardDetails[out] the second ID contents.
```

```
 @param callback[out] the callback function.
 @return 0 is succes, else is failed.
 */
 int IO_ReadIDCB(const char *imgFileName, CallbackReadID callback);

 /*
 @brief capture image file and recognize(OCR).
@param imgFileNameIr[in] is infrared image file name, will not generate if the filename is NULL.
@param imgFileNameWh[in] is color image file name, will not generate if the filename is NULL.
@param imgFileNameUv[in] is ultraviolet image file name, will not generate if the filename is
NULL.
 @param cardType[in] the card type.
 @param cardDetails[out] the contents of recognized.
 @param callback[out] the callback function.
 @return 0 is succes, else is failed.
 */
 int IO_CaptureAndRecognizeFileCB(const char *imgFileNameIr,
                    const char *imgFileNameWh, const char *imgFileNameUv,
                    int cardType, CallbackCaptureAndRecognizeFile callback);

 /*
 @brief decode image file(Barcode)
@param imgFileNameIr[in] is infrared image file name, will not generate if the filename is NULL.
@param imgFileNameWh[in] is color image file name, will not generate if the filename is NULL.
@param imgFileNameUv[in] is ultraviolet image file name, will not generate if the filename is
NULL.
 @param barType[in] the barcode type.
 @param muti[in] false is only decode one barcode; true can decode two barcodes
 @param result[out] the barcode contents array,need malloc buffer in advance.
 @param resultSize[out] the barcode contents array size.
 @param callback[out] the callback function.
 @return 0 is succes, else is failed.
 */
 int IO_CaptureAndDecodeFileCB(const char *imgFileNameIr,
                           const char *imgFileNameWh, const char *imgFileNameUv,
                           unsigned int   barType, bool muti,
                           CallbackCaptureAndDecodeFile callback);
```

7、 Setting file

The setting file is “ImageProcessA8.ini”. can change it for improve image quality and OCR success rate.

The following values can be changed, if not in, don’t change please.

key	Default	Description

	value	
exposure	-6	camera exposure value, the greater the value, the greater the brightness
isCalibrateBrightnessIR	0	If calibratge brightness of IR image, 0 = not, 1=yes,if set 1, please use a white card/thick white paper to do calibration.
isCalibrateBrightnessWH	0	If calibratge brightness of WH image, 0 = not, 1=yes, if set 1, please use a white card/thick white paper to do calibration.
isCalibrateBrightnessUV	0	If calibratge brightness of UV image, 0 = not, 1=yes, if set 1, please use a white card/thick white paper to do calibration.
->left	0	X positing for cut image, (X + width <= the width of original image)
->top	0	y positing for cut image, (y + height <= the height of original image)
->with	0	width positing for cut image,0 is not cut, (X + width <= the width of original image)
->height	0	height positing for cut image, y positing for cut image,0 is not cut. (y + height <= the height of original image)
->gain	1024	the greater the value, the greater the brightness
->offset	0	the greater the value, the smaller the brightness

Questions?

For any technical question, please contact us at :

support@rtscan.net