



Advanced Image Processing Products - <http://www.htsol.com>

SeeCar DLL

General & Technical Information

Jan 5, 2003

Table of Contents

1. PURPOSE & SCOPE	4
2. REFERENCES	4
3. OVERVIEW.....	4
3.1. Product description	4
3.2. System Architecture.....	5
3.3. Sample Applications.....	6
3.4. SeeCarDLL advantages	9
4. ARCHITECTURE	10
4.1. Overview.....	10
4.2. Format.ini Configuration File.....	11
4.3. DLL Input and Output.....	11
4.4. Image type	12
4.5. Other functions.....	13
5. CHANGING SITE-SPECIFIC PARAMETERS	14
5.1. Captured image size.....	14
5.2. Full-Frame or Field.....	14
5.3. Deinterlace	14
5.4. Margins	15
5.5. Max/Min Plate Length & Height.....	15
6. SOFTWARE DEVELOPMENT KIT (SDK)	16
6.1. Overview.....	16
6.2. SeeCarManager Recognition Class functions.....	16
7. SUPPORT AND MORE INFORMATION	17

8. APPENDIX A : SEECAR.H FILE..... 18

1.Purpose & Scope

This document provides general and technical information required to install, interface and operate the SeeCar Vehicle License Plate Identification **DLL** (Windows Dynamic-Link Library) product, or the Linux static library (LSL).

2.References

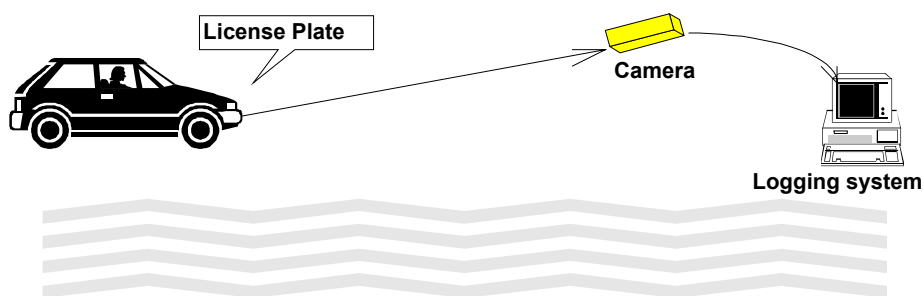
- 2.1 SeeLane - Hi-Tech Solution's Access control and monitoring system
- 2.2 SeeTraffic - Hi-Tech Solution's LPR system for medium to high speed vehicles

Note: In this document, all references to "Vendor" means Hi-Tech Solutions Ltd.

3.Overview

3.1.Product description

SeeCar is a product line of License Plate Recognition (LPR) products that are used to automatically read a license plate on a car. It is available in several configurations. This manual refers to the Windows DLL or Linux static library product configuration (both are identical in use, and will be referred as DLL in the remainder of this document).



SeeCarDLL is part of all HiTech Solution's LPR products. For example, it is integrated into SeeLane – slow to medium speed LPR system (see reference #2.1).

Although the recognition library can be ordered separately, it is recommended to install the SeeLane full turn-key system – which includes additional subsystems: application, frame grabber, I/O card, and debug and operational utilities.

3.2. System Architecture

The **SeeCar DLL** (Dynamic-Link Library) is a **subsystem** in the application. It runs in a PC under the User's or Hi-Tech's application program. It contains plate recognition software, with public function calls that the user can call from the application program.

The following illustration shows this configuration in a typical live system, where the User's Application runs in a PC. It interfaces one or more cameras, captures the image containing the License Plate with a frame grabber board into memory. Then it calls SeeCar DLL. The DLL processes the image and returns the recognition result and status.

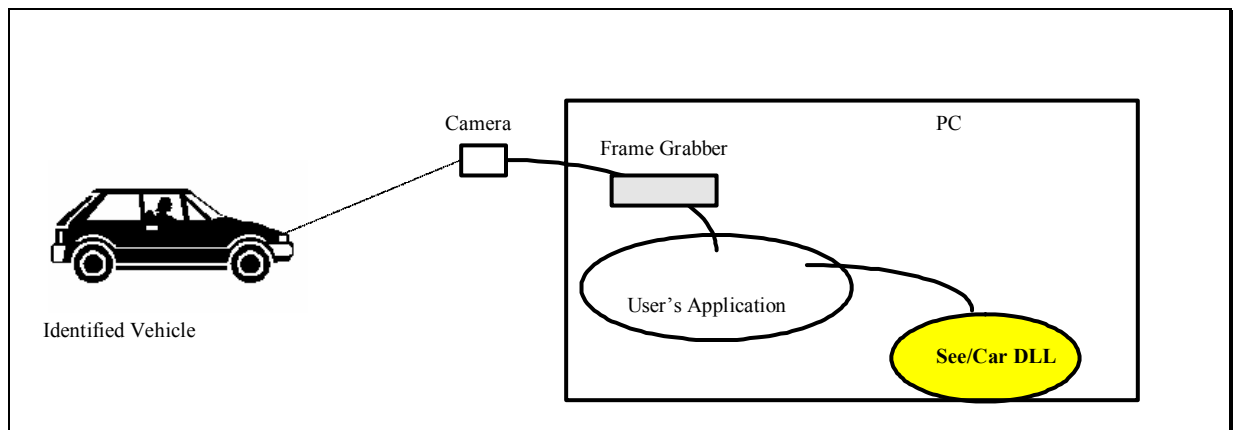


Figure 3.2.1 : See Car DLL integration - **On-Line** Mode

Another option (off-line mode) is to process image files previously captured in the field. The User's program opens each image file, reads the pixels into memory and calls the DLL.

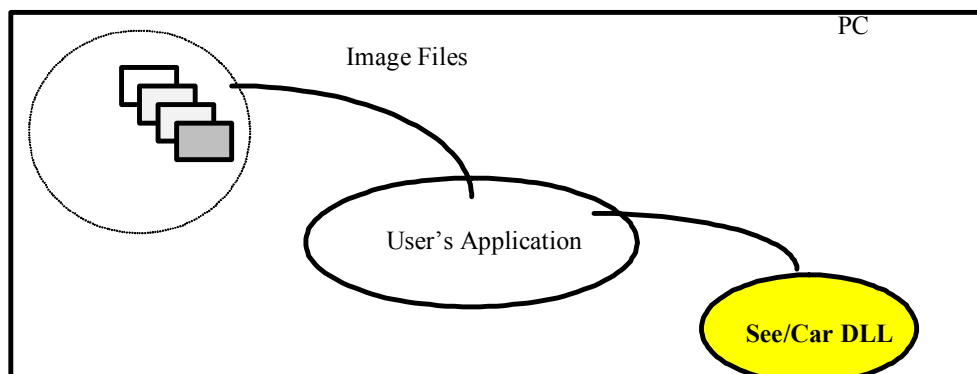


Figure 3.2.2 : See Car DLL integration - **Off-Line** Mode

3.3. Sample Applications

The system can be part of the following applications (a sample of possible use of the system using SeeCar DLL):

3.3.1. Automated Parking Lot

The cars reaching the entrance gate will trigger the projector light, which will turn on. The camera will 'see' the plate and the User's program will capture an image. It will call the DLL to identify its number, and send the recognition string via RS232 or network to the host computer. The host computer will open the gate. When the car exits through the exit gate, the same process is repeated. The host computer in this case will locate the car in the stored numbers, and will calculate the parking fee and open the gate (or in case of membership parking, will just open the gate).

The history of car entries and exits will be used for parking fee calculations, data logging, marketing information and statistics. Additionally, the list can be compared at any time with actual parked cars for security reasons.

In this configuration, it is assumed that a host computer that receives the car plate readings and performs the required financial calculations, and also controls the gates controls the parking lot. The PC using the SeeCar DLL works as the 'eyes' for this host computer.



Figure 3.3.1 : Gate Access control and Members parking illustration

Note that this application is performed by our access control application, SeeLane (ref #2.1). This application handles the DLL and the hardware and operates as a stand-alone access control system, which can be integrated into different types of applications. An additional system, SeeTraffic(ref #2.2), is geared to medium to fast speed vehicles and operates as a recognition system which interfaces the DLL and the hardware. Please refer to the applications' documentation.

3.3.2.Gate to Factory or secured area

An alternative operation mode can be that the PC with the SeeCar DLL works as a stand-alone system, without a host computer. In such applications, a pre-stored list of authorized cars will be stored in the PC as a file. The car will reach the gate, turn the projector on, and will be identified by the DLL. Then, the PC will open the gate and store or print the identified vehicle.

This application is suitable for army installations, factories and other secured facilities.

Note that this application is also supported by SeeLane application.

3.3.3.Toll road

The system can be used to data collect the car plates that pass a toll road gate, for billing or data collection.



Figure 3.3.3: Toll road sample installation

Note that this application is also supported by SeeLane or SeeTraffic applications.

3.3.4. Police uses and Law Enforcement

3.3.4.1. Stolen Cars

A predefined list of stolen cars can be entered to the system, and its detection will signal an alarm (a sound card 'siren' for example) when such a car will pass.

Note that this application is also supported by our SeeCarTrap application, working with SeeLane or SeeTraffic.

3.3.4.2. Speed Violations

Another system application can be used for law enforcement, either for speed violation or entry to illegal traveling zones. In this case the system can identify cars on-line, or process image files off-line (for ticketing).

The automatic identifications accelerate the processing of the violations. The manual operation takes a few minutes per image, while the automatic processing can be performed at a rate of several images per second. This is a 1:1000 quicker turnaround rate.

Note that this application is also supported by our SeeFilm application.

3.3.5. Other applications

Please refer to our Web Site for additional sample installations.

3.4. SeeCarDLL advantages

The SeeCarDLL recognition library has the following advantages over existing LPR software:

- simple integration into the existing or new applications
- has a high recognition rate (multiple checks for each letter/digit)
- covers a wide range of plate size (80-300; recommended 150 pixels side-to-side)
- deals with very small plates (down to ~80 pixels per plate on typical plates)
- deals with wide range of contrast images (recommended: 50 gray levels font to background)
- allowable rotation ± 30 degrees (depends on plate type; recommended smallest angle)
- configurable operation (using standard ini file)
- fast response - one of the fastest packages available. Pentium 1900Mhz at typical 25msec per image to return the recognition string. This allows the user's application to utilize most of the PC resources.
- low cost solution - standard PC with minimal resources
- Adaptable to different country standards (with a simple configuration file change)
- Available in 32bit versions, for use with standard compilers
- source code for sample applications is available (cuts time-to-market)
- Highly integratable applications (see SeeLane or SeeTraffic)

4. Architecture

This section describes the architecture of the SeeCar DLL.

4.1. Overview

4.1.1. Functions

The system performs the following functions within the DLL:

- Image handling - handles the input image and prepares it for the processing
- Image Enhancement - improves the quality of the image
- Plate finding - locates the license plate within the image
- Slope calculation - measures the angle of the plate
- Characters detection - locates each of the plate's characters
- Characters identification - identifies each of the detected characters
- Validation - compares the results
- Output - sends the results back to the User's program

The effect of this operation is like a reduction program: a huge array of information (the image) is converted to a short ASCII string containing the identified plate. The conversion ratio is roughly 512 X 512 (average image size in pixels, each pixel a Byte) to about 10 characters!

4.1.2. Application program

Note that the DLL does not capture the image (via a frame grabber for on-line applications or via a file read for off-line applications). This is the role of the application program which performs all the hardware and system functions of the application. Furthermore, the DLL does not process the recognition results and outputs it to the display or computer network.

Sample applications in source file will be supplied to the developer as part of the DLL package. Other applications, such as a parking lot system application, can be licensed.

4.1.3. Structure

The DLL is composed of the following elements:

- DLL object code which performs its functions
- Security plug which is required to be plugged in the parallel port (or, as a special option, to a USB port)

- “Format.ini” configuration file - which describes the parameters of the plate and identification. A detailed explanation is provided in the following subparagraph.

4.2.Format.ini Configuration File

Since the DLL is designed to operate in various configurations, installations and local standards, a special text file is available to optionally change the configuration and installation of the system. The file is in a standard initialization (“.ini”) file format as commonly used in Windows environment. This file is a text file with a set of fields (each enclosed in square[] brackets) and a set of assignments (par_i=value_j). Comments are enclosed after “;”. The file can be edited using standard text editors. Thus, the vendor or user can shape the characteristics of the system in order to fit it to the required environment.

This file (format.ini) has the following types of parameters:

- **Formats** - Define formats that describe the local standard. (For example, in a certain country there are 3 different standards of the plate then each standard is described as a separate format). This is prepared by the Vendor before shipping the product, and tested against a set of sample test images.
- **Internal Parameters** - a set of values used for the optimized execution of the recognition software. This also is prepared by the vendor and optimized for a certain country standard.
- **External Parameters** - a few parameters are site specific and can be either determined by the Vendor, or set on site by the installer. These are described in a later section.

A short description of **some** of the parameters are provided in the next section.

4.3.DLL Input and Output

The DLL’s main function (sc_get_id_string) accepts the following inputs and outputs, as described in seecar.h (the header file for the DLL):

```
SEECAR_RET_STATUS SEEAPI sc_get_id_string (  
    byte MEM* p_byte,           /* source image bytes buffer */  
    id_string plate_string,     /* resulting plate string */  
    byte* plate_type           /* resulting plate type */  
);
```

The function receives:

- (1) a pointer to the input image in memory (*p_byte),
- (2) a pointer to the returned string (plate_string),

- (3) The identified format is returned with `plate_type` pointer (one of the possible formats, as described in the `formats.ini` file). This information is usually not used by the User's application.

In addition to the parameters, a return code (`SEECAR_RET_STATUS`) is also returned with the status of the recognition. The possible error codes are:

0=SC_SUCCESS,	/* successfully processed */
1=SC_FAILURE,	/* failed to process */
2=SC_NO_PLUG,	/* security plug not found */
3=SC_NO_MEM	/* not enough memory */

4.4. Image type

The major parameter that is passed to the DLL is a pointer to the image stored in memory. The contents of the image were previously captured or read by the calling application. The image should have the following characteristics:

- 256 levels (8-bit) gray-level, where: white is 255, black is 0.
- Image size is described in `format.ini` configuration file (e.g., 512 X 512)
- both frame (both even and odd lines) or field (only even or odd lines) type images could be passed to the DLL (a parameter in the `format.ini` configuration file)

In case of a B&W camera, the images are directly extracted to the image memory. In case of color images, the application program should convert the color information into gray-level (using the palette information and using the intensity equation conversion).

4.5. Other functions

Refer to SeeCar.h include file for a updated and formal description of the interface with the DLL. This header file includes additional function calls that are available in the DLL. The following functions exist:

seecar_init - initializes the DLL (should be called once)

seecar_free - free DLL resources after termination (should be called once upon exit)

get_plate_rect - returns bounding rectangle of the recognized plate

get_plate_image - returns plate image array

get_plate_width - returns configuration file image width

get_plate_height - returns configuration file image height

reset_image_sizes - change image sizes

get_id_confidence - returns overall grade and per-font grade array

A sample H file is included in Appendix A.

You can also use the sample applications included in the Software Development Kit (SDK) to learn how to integrate the DLL in your application.

5.Changing Site-specific Parameters

This section describes how certain parameters could be changed in the configuration file (format.ini). Note that in most sites the values can be the same. Furthermore, the Vendor can make these changes for you on supplied sample images.

We normally optimize the format.ini file parameters for you, based on a large set of images (that can be used for the testing of the configuration file). This is since the parameters are complicated and require deep knowledge of the recognition process.

5.1.Captured image size

The following lines describe the captured image size in pixels:

```
capture_width      = 768           ; captured picture size
capture_height     = 288           ;
```

Note that this example is for a standard CCIR field image. (The field data is an image with even or odd lines only).

5.2.Full-Frame or Field

The DLL can accept either Full-Frame images (frame=1) or single Field images (Frame=0). The line that requires this setting is:

```
frame = 0           ; whole frame = 1; even/odd field only = 0
```

5.3.Deinterlace

In case of full-frame only, you can determine whether to use the whole frame image contents (even and odd rows) or to perform deinterlacing correction. In the latter case the parameter should be set to 1. This prevents the effect of the moving car, since there is a time delay between even and odd rows in interlaced cameras.

```
deinterlace       = 1
```

For progressive-scan or Non-interlaced cameras, the value could be 0 (no deinterlacing).

5.4.Margins

The image margins should be defined in order to prevent the DLL to process titles or data that is known to exist on all images. For example, a law-enforcement image contains the speed violation data on the bottom line (20 pixels). In this case the margin in the bottom side should be set to 20:

bottom_margin = 20 ;;due to bottom speed-data text

For all 4 sides of the image, this example could be expanded as follows. Note that the value 3 is the minimum margin.

left_margin = 3 ;;the minimum margin
right_margin = 14
top_margin = 3
bottom_margin = 20

5.5.Max/Min Plate Length & Height

In order to prevent the software to lock on false alarms, the following statements could limit the size of the plate (in pixels):

min_hor_plate_length = 70
max_hor_plate_length = 200
min_hor_plate_height = 20
max_hor_plate_height = 60

6. Software Development Kit (SDK)

6.1. Overview

Our software CD-R disk comes packed with software development tools in order to shorten the integration phase.

We provide VC++ and VB sample applications sources for testing the DLL interface in Windows (or the static library for Linux).

Two basic programs are included:

- (1) a single image recognition program (reads an image file, calls the DLL, and outputs the recognition) and
- (2) a multiple-image recognition program (reads a number of image files of the same vehicle, then optimizes and verifies the recognition results). *See details below.*

A COM interface to VB is also available upon request. Other special requests could also be prepared, please contact us and we would try to assist your development team.

6.2. SeeCarManager Recognition Class functions

The SDK includes a multiple-image recognition program which is based on a VC++ class. The functions that are included in this program are a second level of programming (a level of the DLL), which can be used to program in a higher level than the DLL.

The functions in this class include:

Create - initialize on first use of the class

add_image - adds a new image for the current recognized car

IdentifyImages - retrieves the result

For each vehicle, the function **add_image** should be called several times with all the images captured for this vehicle. Then **IdentifyImages** will return the overall result.

7.Support and more Information

You can contact us for more information at:

Hi-Tech Solutions POBox #133 Migdal-Haemek Israel 10500
Email info@htsol.com
Israel. Fax (+972) 46 44 1870
Israel Tel (+972) 46 44 1890

Visit our Home page <http://www.htsol.com> and get additional information or download demo applications!

The web site contains many **recognition players** (under **Demo** icon – at the left side of each web page). The recognition players are Country-specific and differ mostly in the configuration file –format.ini - and the images – which are local plates. The player runs a number of images and performs recognition on each image. This is similar to the DLL operation, only that the recognition library is statically compiled into the player. The player may be also used to check your own images.

8. Appendix A : SeeCar.H file

This appendix contains a sample seecar.h file which contains function prototypes.

Please refer to the actual header file for latest revision.

```

/*****
* File:                seecar.h
*
* Project:             See/Car generic DLL
*
* Version:             March 13 2001
*
* Contains:            DLL interface routines
*
*****/
* Overview
* -----
* This header contains the declarations required to use the
* See/Car DLL.
*
*****/

#ifndef _SEECAR_H_
#define _SEECAR_H_

/* --- require a 32 bit compiler --- */
#if !defined(_WIN32) && !defined(__WIN32__) && !defined(GCC)
#error Unable to determine the 32 bit compiler type.
#endif

/* --- auto-define the USE_SC_DLL --- */
#if !defined(SIM) && !defined(CREATE_DLL) && !defined(DEMO) &&
!defined(GCC)
    #if !defined(USE_SC_DLL)
        #define USE_SC_DLL
    #endif
#endif

/*
 * DLL calling convention
 */
#ifdef CREATE_DLL
    #include <windows.h>
    #if defined(__WIN32__) || defined(_WIN32)
        #define SEEAPI __declspec(dllexport)
    #else
        #define SEEAPI _export
    #endif
#else
    #include <windows.h>
    #if defined(__WIN32__) || defined(_WIN32)
        #define SEEAPI __declspec(dllimport)
    #else
        #define SEEAPI
    #endif
#endif

```

```

#endif

#else
#define SEEAPI
#if defined(__WIN32__) || defined(_WIN32)
#include <windows.h>
#else
typedef int BOOL;
#define FAR
#endif
#endif /* CREATE_DLL */

#ifdef __cplusplus
extern "C" {
#endif

typedef enum{ SC_SUCCESS, /* successful */
              SC_FAILURE, /* failed to process */
              SC_NO_PLUG, /* security plug not found*/
              SC_NO_MEM, /* not enough memory */
              SC_NO_INIT, /* DLL is not initialized */
              SC_NO_CONFIG /* unable to read config file*/
} SEECAR_RET_STATUS;

/*****
* SEECAR_INIT
*
* Function: Initializes the DLL. This function must be called
before using
* the DLL.
*
* Args:
* debug - should always be FALSE
* fcallback - watch dog callback, should always be
NULL.
*
* Return:
* SC_SUCCESS - initialization is OK
* SC_FAILURE - unable to initialize
* SC_NO_MEM - memory allocation error
* SC_NO_PLUG - HASP protection plug not found
* SC_NO_CONFIG - unable to read configuration file
FORMAT.INI
*
* Notes:
*/
SEECAR_RET_STATUS SEEAPI seecar_init(
    BOOL debug, /* enable debug info printing */
    void FAR (*fcallback)(void)); /* callback function
*/

/*****
* SEECAR_FREE
*
* Notes:
* Release the DLL resources
*/
void FAR SEEAPI seecar_free( void );

```

```

/*****
* SC_GET_ID_STRING
*
* Function: Identify an image. The identified code will be returned
*           in the parameter 'plate_string' and the format
*           number in 'plate_type'.
*
* Args:
*       p_byte - pointer to data bits
*       plate_string - identified code buffer
*       plate_type - format number
*
* Ret:
*       SC_SUCCESS - image was successful identified
*       SC_FAILURE - unable to identify image
*       SC_NO_INIT - the DLL has not been initialized
*                   (call SEECAR_INIT())
*       SC_NO_PLUG - HASP protection plug is not found
*
*/

SEECAR_RET_STATUS FAR SEEAPI sc_get_id_string(
    unsigned char* p_byte, /* source image bytes buffer */
    char* plate_string, /* resulting plate string */
    char* plate_type /* resulting plate type */
);

#if !defined(SIM) && !defined(GCC)

/*****
* GET_PLATE_RECT
*
* Function: The function returns the bounding rectangle of the
plate from the
*           last identified image.
*
* Args:
*       rect -pointer to rectangle which will hold the plate area
*
* Return:
*       TRUE -rect is set with the correct values
*       FALSE-rect is not set because last image wasnt identified
*
* Note:
*/
BOOL FAR SEEAPI get_plate_rect(RECT *rect);

/*****
* CHANGE_FRAME_PARAM
*
* Function: change PARAMframe value (field->frame, frame->field)
* Return:
*       new value of PARAMframe (TRUE=frame, FALSE=field)
*/
BOOL FAR SEEAPI change_frame_param();

#endif /* ! SIM */

/*****
* GET_PLATE_IMAGE

```

```

*
*   Function: The function returns the rotated & detilted plate
*             image before enhancement of last identified plate
*
*   Args:
*       x_size - image width (pointer)
*       y_size - image height (pointer)
*
*   Return:
*       pointer to image bits.  NULL if the ID number undefined
*
*   Note:
*   Important : if FORMAT.ini parameter show_zoom is zero, then
*               function returns NULL allways
*/
char FAR SEEAPI *get_plate_image(int* width,int* height);

/*****
*   GET_IMAGE_WIDTH
*
*   Return:
*       The image width as read from configuration file.
*       0 if error reading configuration file.
*/
unsigned int          FAR SEEAPI  get_image_width(void);

/*****
*   GET_IMAGE_HEIGHT
*
*   Return:
*       The image height as read from configuration file.
*       0 if error reading configuration file.
*/
unsigned int          FAR SEEAPI  get_image_height(void);

#if !defined(SIM)
/*****
*   RESET_IMAGE_SIZES
*
*   Args:
*       x_size - new image width
*       y_size - new image height
*   Return:
*       SC_SUCCESS - initialization is OK
*       SC_FAILURE - unable to initialize
*/
SEECAR_RET_STATUS FAR SEEAPI  reset_image_sizes(unsigned int
x_size,unsigned int y_size);
#endif

/*****
*   G E T _ I D _ C O N F I D E N C E
*   Function returns confidence of whole recognized string & each
*   position in this string in percent (0 - 100, where 0% is total fail
*   and 100% is total success).
*
*   Notes: 1. call this function only after sc_get_id_string &
*           in case it returns SC_SUCCESS.
*          2. min_grade in format.ini file limits the returned value
*           to a level under which the recognition fails
*
*   Returned int : Confidence of current identification in % (0-100)
*
*   Returned as parameter: symbols_marks - array of confidence marks

```

```
*           for each font in identified string in percent (user
*           defined array, same size as in plate_string of
*           sc_get_id_string() )
*/
int FAR SEEAPI get_id_confidence(char* symbols_marks);

#endif      /* _SEECAR_H_ */
```